

# Fast Simulation of Large-Scale Growth Models

Tobias Friedrich<sup>1</sup> and Lionel Levine<sup>2</sup>

<sup>1</sup> Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

<sup>2</sup> Massachusetts Institute of Technology, Cambridge, MA 02139, USA

**Abstract.** We give an algorithm that computes the final state of certain growth models without computing all intermediate states. Our technique is based on a “least action principle” which characterizes the odometer function of the growth process. Starting from an approximation for the odometer, we successively correct under- and overestimates and provably arrive at the correct final state. The degree of speedup depends on the accuracy of the initial guess.

Determining the size of the boundary fluctuations in growth models like internal diffusion-limited aggregation (IDLA) is a long-standing open problem in statistical physics. As an application of our method, we calculate the size of fluctuations over two orders of magnitude beyond previous simulations.

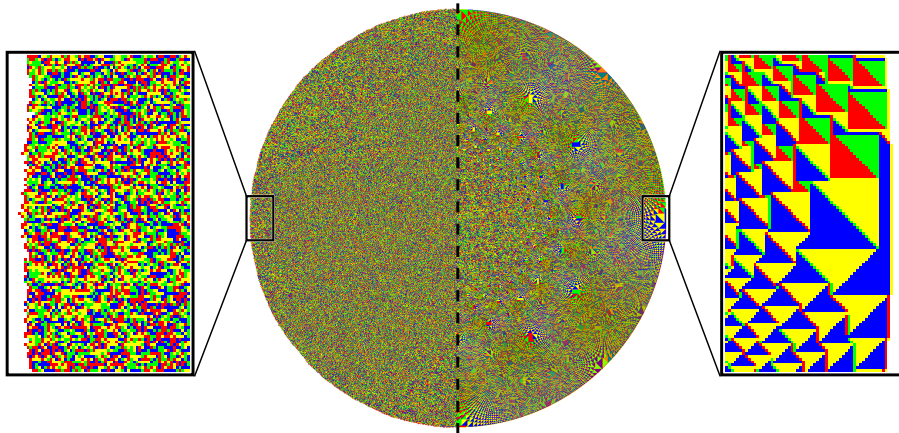
## 1 Introduction

In this paper we study the *abelian stack model*, a type of growth process on graphs. Special cases include *internal diffusion limited aggregation* (IDLA) and *rotor-router aggregation*. We describe a method for computing the final state of the process, given an initial approximation. The more accurate the approximation, the faster the computation.

### IDLA

Starting with  $N$  chips at the origin of the two-dimensional square grid  $\mathbb{Z}^2$ , each chip in turn performs a simple random walk until reaching an unoccupied site. Introduced by Meakin and Deutch [25] and independently by Diaconis and Fulton [10], IDLA models physical phenomena such as solid melting around a heat source, electrochemical polishing, and fluid flow in a Hele-Shaw cell. Lawler, Bramson, and Griffeath [22] showed that as  $N \rightarrow \infty$ , the asymptotic shape of the resulting cluster of  $N$  occupied sites is a disk (and in higher dimensions, a Euclidean ball).

The boundary of an IDLA cluster is a natural model of a random propagating front (Figure 1, left). From this perspective, the most basic question one could ask is, what is the scale of the fluctuations around the limiting circular shape? Until recently this was a long-standing open problem in statistical physics. It is now known that the fluctuations in dimension 2 are of order at most  $\log N$  [3, 19]; however, it is still open to show that the fluctuations are at least this large. We give numerical evidence that  $\log N$  is in fact the correct order, and estimate the constant in front of the log.



**Fig. 1:** IDLA cluster (left) and rotor-router cluster with counterclockwise rotor sequence (right) of  $N = 10^6$  chips. Half of each circular cluster is shown. Each site is colored according to the final direction of the rotor on top of its stack (yellow=W, red=S, blue=E, green=N). Note that the boundary of the rotor-router cluster is much smoother than the boundary of the IDLA cluster. Larger rotor-router clusters of size up to  $N = 10^{10}$  can be found at [1].

### Rotor-router aggregation

James Propp [21] proposed the following way of derandomizing IDLA. At each lattice site in  $\mathbb{Z}^2$  is a *rotor* that can point North, East, South or West. Instead of stepping in a random direction, a chip rotates the rotor at its current location counterclockwise, and then steps in the direction of this rotor. Each of  $N$  chips starting at the origin walks in this manner until reaching an unoccupied site. Given the initial configuration of the rotors (which can be taken, for example, all North), the resulting growth process is entirely deterministic. Regardless of the initial rotors, the asymptotic shape is a disk (and in higher dimensions, a Euclidean ball) and the inner fluctuations are proved to be  $O(\log N)$  [24]. The true fluctuations appear to grow even more slowly, and may even be bounded independent of  $N$ .

Rotor-router aggregation is remarkable in that it generates a nearly perfect disk in the square lattice without any reference to the Euclidean norm  $(x^2 + y^2)^{1/2}$ . Perhaps even more remarkable are the patterns formed by the final directions of the rotors (Figure 1, right).

### Computing the odometer function

The central tool in our analysis of both models is the *odometer function*, which measures the number of chips emitted from each site. The odometer function determines the shape of the final occupied cluster via a nonlinear operator that we call the *stack Laplacian*. Our main technical contribution is that even for highly

non-deterministic models such as IDLA, one can achieve *fast exact calculation via intermediate approximation*. Approximating the two growth processes by an idealized model called the divisible sandpile, we can use the known asymptotic expansion of the potential kernel of random walk on  $\mathbb{Z}^2$  to obtain an initial approximation of the odometer function. We present a method for carrying out subsequent local corrections to provably transform this approximation into the exact odometer function, and hence compute the shape of the occupied cluster. Our runtime strongly depends on the accuracy of the initial approximation.

## Applications

Traditional step-by-step simulation of both aforementioned models in  $\mathbb{Z}^2$  requires a runtime of order  $N^2$  to compute the occupied cluster. Using our new algorithm, we are able to generate large clusters faster: Our observed runtimes are about  $N \log N$  for the rotor-router model and about  $N^{1.5}$  for IDLA. By generating many independent IDLA clusters, we estimate the order of fluctuations from circularity over two orders of magnitude beyond previous simulations. Our data strongly support the findings of [26] that the order of the maximum fluctuation for IDLA in  $\mathbb{Z}^2$  is logarithmic in  $N$ . Two proofs of an upper bound  $C \log N$  on the maximum fluctuation for IDLA in  $\mathbb{Z}^2$  have recently been announced: see [2, 3] and [19]. While the implied constant  $C$  in these bounds is large, our simulations suggest that the maximum fluctuation is only about  $0.528 \ln N$ .

For rotor-router aggregation we achieve four orders of magnitude beyond previous simulations, which has enabled us to generate fine-scaled examples of the intricate patterns that form in the rotors on the tops of the stacks at the end of the aggregation process (Figure 1, right). These patterns remain poorly understood even on a heuristic level. We have used our algorithm to generate a four-color 10-gigapixel image [1] of the final rotors for  $N = 10^{10}$  chips. This file is so large that we had to use a Google maps overlay to allow the user to zoom and scroll through the image. Indeed, the degree of speedup in our method was so dramatic that memory, rather than time, became the limiting factor.

## Related Work

Unlike random walk, in a rotor-router walk each vertex serves its neighbors in a fixed order. The resulting walk, which is completely deterministic, nevertheless closely resembles a random walk in several respects [6–8, 11, 16, 18]. The rotor-router mechanism also leads to improvements in algorithmic applications. Examples include autonomous agents patrolling a territory [28], external mergesort [5], broadcasting information in networks [12, 13], and iterative load-balancing [17].

Abelian stacks (defined in the next section) are a way of indexing the steps of a walk by location and time rather than time alone. This fruitful idea goes back at least to Diaconis and Fulton [10, §4]. Wilson [29] (see also [27]) used this stack-based view of random walk in his algorithm for sampling a random spanning tree of a directed graph. The final cycle-popping phase of our algorithm

is directly inspired by Wilson’s algorithm. Our serial algorithm for IDLA also draws on ideas from the parallel algorithm of Moore and Machta [26].

Abelian stacks are a special case of *abelian networks* [9], also called “abelian distributed processors.” In this viewpoint, each vertex is a finite automaton, or “processor.” The chips are called “messages.” When a processor receives a message, it can change internal state and also send one or more messages to neighboring processors according to its current internal state. We believe that it might be possible to extend our method to other types of abelian networks, such as the Bak-Tang-Wiesenfeld sandpile model [4]. Indeed, the initial inspiration for our work was the “least action principle” for sandpiles described in [14].

## 2 Formal Model

The underlying graph for the abelian stack model can be any finite or infinite directed graph  $G = (V, E)$ . Each edge  $e \in E$  is oriented from its source vertex  $\mathbf{s}(e)$  to its target vertex  $\mathbf{t}(e)$ . Self-loops (edges  $e$  such that  $\mathbf{s}(e) = \mathbf{t}(e)$ ) and multiple edges (distinct edges  $e, e'$  such that  $\mathbf{s}(e) = \mathbf{s}(e')$  and  $\mathbf{t}(e) = \mathbf{t}(e')$ ) are permitted. We assume that  $G$  is *locally finite* – each vertex is incident to finitely many edges – and *strongly connected*: for any two vertices  $x, y \in V$  there are directed paths from  $x$  to  $y$  and from  $y$  to  $x$ . At each vertex  $x \in V$  is an infinite stack of *rotors*  $(\rho_n(x))_{n \geq 0}$ . Each rotor  $\rho_n(x)$  is an edge of  $G$  emanating from  $x$ , that is,  $\mathbf{s}(\rho_n(x)) = x$ . We say that rotor  $\rho_0(x)$  is “on top” of the stack.

A finite number of indistinguishable chips are dispersed on the vertices of  $G$  according to some prescribed initial configuration. For each vertex  $x$ , the first chip to visit  $x$  is absorbed there and never moves again. Each subsequent chip arriving at  $x$  first shifts the stack at  $x$  so that the new stack is  $(\rho_{n+1}(x))_{n \geq 0}$ . After shifting the stack, the chip moves from  $x$  to the other endpoint  $y = \mathbf{t}(\rho_1(x))$  of the rotor now on top. We call this two-step procedure (shifting the stack and moving a chip) *firing* the site  $x$ . The effect of this rule is that the  $n$ -th time a chip is emitted from  $x$ , it travels along the edge  $\rho_n(x)$ .

We will generally assume that the stacks are *infinitive*: for each edge  $e$ , infinitely many rotors  $\rho_n(\mathbf{s}(e))$  are equal to  $e$ . If  $G$  is infinite, or if the total number of chips is at most the number of vertices, then this condition ensures that firing eventually stops, and all chips are absorbed.

We are interested in the set of *occupied sites*, that is, sites that absorb a chip. The *abelian property* [10, Theorem 4.1] asserts that this set does not depend on the order in which vertices are fired. This property plays a key role in our method; we discuss it further in §3.

If the rotors  $\rho_n(x)$  are independent and identically distributed random edges  $e$  such that  $\mathbf{s}(e) = x$ , then we obtain IDLA. For instance, in the case  $G = \mathbb{Z}^2$ , we can take the rotors  $\rho_n(x)$  to be independent with the uniform distribution on the set of 4 edges joining  $x$  to its nearest neighbors  $x \pm \mathbf{e}_1, x \pm \mathbf{e}_2$ . The special case of IDLA in which all chips start at a fixed vertex  $o$  is more commonly described as follows. Let  $A_1 = \{o\}$ , and for  $N \geq 2$  define a random set  $A_N$  of  $N$  vertices

of  $G$  according to the recursive rule

$$A_{N+1} = A_N \cup \{x_N\} \quad (1)$$

where  $x_N$  is the endpoint of a random walk started at  $o$  and stopped when it first visits a site not in  $A_N$ . These random walks describe one particular sequence in which the vertices can be fired, for the initial configuration of  $N$  chips at  $o$ . The first chip is absorbed at  $o$ , and subsequent chips are absorbed in turn at sites  $x_1, \dots, x_{N-1}$ . When firing stops, the set of occupied sites is  $A_N$ .

A second interesting case is deterministic: the sequence  $\rho_n(x)$  is periodic in  $n$ , for every vertex  $x$ . For example, on  $\mathbb{Z}^2$ , we could take the top rotor in each stack to point to the northward neighbor, the next to the eastward neighbor, and so on. This choice yields the model of rotor-router aggregation defined by Propp [21] and analyzed in [23, 24]. It is described by the growth rule (1), where  $x_N$  is the endpoint of a rotor-router walk started at the origin and stopped on first exiting  $A_N$ .

### 3 Least Action Principle

A *rotor configuration* on  $G$  is a function  $r: V \rightarrow E$  such that  $\mathbf{s}(r(v)) = v$  for all  $v \in V$ . A *chip configuration* on  $G$  is a function  $\sigma: V \rightarrow \mathbb{Z}$  with finite support. Note we do not require  $\sigma \geq 0$ . If  $\sigma(x) = m > 0$ , we say there are  $m$  chips at vertex  $x$ ; if  $\sigma(x) = -m < 0$ , we say there is a hole of depth  $m$  at vertex  $x$ .

For an edge  $e$  and a nonnegative integer  $n$ , let

$$R_\rho(e, n) = \#\{1 \leq k \leq n \mid \rho_k(\mathbf{s}(e)) = e\}$$

be the number of times  $e$  occurs among the first  $n$  rotors in the stack at the vertex  $\mathbf{s}(e)$  (excluding the top rotor  $\rho_0(\mathbf{s}(e))$ ). When no ambiguity would result, we drop the subscript  $\rho$ .

Write  $\mathbb{N}$  for the set of nonnegative integers. Given a function  $u: V \rightarrow \mathbb{N}$ , we would like to describe the net effect on chips resulting from firing each vertex  $x \in V$  a total of  $u(x)$  times. In the course of these firings, each vertex  $x$  emits  $u(x)$  chips, and receives  $R_\rho(e, u(\mathbf{s}(e)))$  chips along each incoming edge  $e$  with  $\mathbf{t}(e) = x$ . This motivates the following definition.

**Definition 1.** *The stack Laplacian of a function  $u: V \rightarrow \mathbb{N}$  is the function  $\Delta_\rho u: V \rightarrow \mathbb{Z}$  given by  $\Delta_\rho u(x) = \sum_{\mathbf{t}(e)=x} R_\rho(e, u(\mathbf{s}(e))) - u(x)$ , where the sum is over all edges  $e$  with target vertex  $\mathbf{t}(e) = x$ .*

Given an initial chip configuration  $\sigma_0$ , the configuration  $\sigma$  resulting from performing  $u(x)$  firings at each site  $x \in V$  is given by

$$\sigma = \sigma_0 + \Delta_\rho u. \quad (2)$$

The rotor configuration on the tops of the stacks after these firings is also easy to describe. We denote this configuration by  $\text{Top}_\rho(u)$ , and it is given

by  $\text{Top}_\rho(u)(x) = \rho_{u(x)}(x)$ . Vertices  $x_1, \dots, x_m$  form a *legal firing sequence* for  $\sigma_0$  if  $\sigma_j(x_{j+1}) > 1$ ,  $j = 0, \dots, m-1$  where  $\sigma_j = \sigma_0 + \Delta_\rho u_j$  and  $u_j(x) = \#\{i \leq j : x_i = x\}$ .

In words, the condition  $\sigma_j(x_{j+1}) > 1$  says that after firing  $x_1, \dots, x_j$ , the vertex  $x_{j+1}$  has at least two chips. We require at least two because in our growth model, the first chip to visit each vertex gets absorbed.

The firing sequence is *complete* if no further legal firings are possible; that is,  $\sigma_m(x) \leq 1$  for all  $x \in V$ . If  $x_1, \dots, x_m$  is a complete legal firing sequence for the chip configuration  $\sigma_0$ , then we call the function  $u := u_m$  the *odometer* of  $\sigma_0$ . The odometer tells us how many times each site fires.

**Abelian Property** [10, Theorem 4.1] Given an initial configuration  $\sigma_0$  and stacks  $\rho$ , every complete legal firing sequence for  $\sigma_0$  has the same odometer function  $u$ .

It follows that the final chip configuration  $\sigma_m = \sigma_0 + \Delta_\rho u$  and the final rotor configuration  $\text{Top}_\rho(u)$  do not depend on the choice of complete legal firing sequence.

Given a chip configuration  $\sigma_0$  and rotor stacks  $(\rho_k(x))_{k \geq 0}$ , our goal is to compute the final chip configuration  $\sigma_m$  without performing individual firings one at a time. A fundamental observation is that by equation (2), it suffices to compute the odometer function  $u$  of  $\sigma_0$ . Indeed, once we know that each site  $x$  fires  $u(x)$  times, we can add up the number of chips  $x$  receives from each of its neighbors and subtract the  $u(x)$  chips it emits to figure out the final number of chips at  $x$ . This arithmetic is accomplished by the term  $\Delta_\rho u$  in equation (2). In practice, it is usually easy to compute  $\Delta_\rho u$  given  $u$ , an issue we address in §4.

Our approach will be to start from an approximation of  $u$  and correct errors. In order to know when our algorithm is finished, the key mathematical point is to find a list of properties of  $u$  that characterize it uniquely. Our main result in this section, Theorem 1, gives such a list. As we now explain, the hypotheses of this theorem can all be guessed from certain necessary features of the final chip configuration  $\sigma_m$  and the final rotor configuration  $\text{Top}_\rho(u)$ . What is perhaps surprising is that these few properties suffice to characterize  $u$ .

Let  $x_1, \dots, x_m$  be a complete legal firing sequence for the chip configuration  $\sigma_0$ . We start with the observation that since no further legal firings are possible,

- $\sigma_m(x) \leq 1$  for all  $x \in V$ .

Next, consider the set  $A := \{x \in V : u(x) > 0\}$  of sites that fire. Since each site that fires must first absorb a chip, we have

- $\sigma_m(x) = 1$  for all  $x \in A$ .

Finally, observe that for any vertex  $x \in A$ , the rotor  $r(x) = \text{Top}_\rho(u)(x)$  at the top of the stack at  $x$  is the edge traversed by the last chip fired from  $x$ . The last chip fired from a given finite subset  $A'$  of  $A$  must be to a vertex outside of  $A'$ , so  $A'$  must have a vertex whose top rotor points outside of  $A'$ .

- For any finite set  $A' \subset A$ , there exists  $x \in A'$  with  $\mathfrak{t}(r(x)) \notin A'$ .

We can state this last condition more succinctly by saying that the rotor configuration  $r = \text{Top}_\rho(u)$  is *acyclic* on  $A$ ; that is, the spanning subgraph  $(V, r(A))$  has no directed cycles. Here  $r(A) = \{r(x) \mid x \in A\}$ .

**Theorem 1.** *Let  $G$  be a finite or infinite directed graph,  $\rho$  a collection of rotor stacks on  $G$ , and  $\sigma_0$  a chip configuration on  $G$ . Fix  $u_*: V \rightarrow \mathbb{N}$ , and let  $A_* = \text{supp}(u_*)$ . Let  $\sigma_* = \sigma_0 + \Delta_\rho u_*$ , and suppose that*

- $\sigma_* \leq 1$ ;
- $A_*$  is finite;
- $\sigma_*(x) = 1$  for all  $x \in A_*$ ; and
- $\text{Top}_\rho(u_*)$  is acyclic on  $A_*$ .

*Then the odometer function  $u$  is well-defined, and  $u_* = u$ .*

Note that to ensure that  $u$  is well-defined (i.e., that there exists a finite complete legal firing sequence) it is common to place some minimal assumptions on  $\rho$  and  $\sigma_0$ . For example, if  $G$  is infinite and strongly connected, then it suffices to assume that the stacks  $\rho$  are infinitive. Theorem 1 does not explicitly make any assumptions of this kind; rather, if a function  $u_*$  exists satisfying the conditions listed, then  $u$  must be finite (and equal to  $u_*$ ).

## 4 Algorithm: From Approximation to Exact Calculation

In this section we describe how to compute the odometer function  $u$  exactly, given as input an approximation  $u_1$ . The running time depends on the accuracy of the approximation, but the correctness of the output does not. How to find a good approximation  $u_1$  for  $N$  chips started at the origin in  $\mathbb{Z}^2$ , can be found in the full version of the paper [15].

Recall that  $G$  may be finite or infinite, and we assume that  $G$  is strongly connected. We assume that the initial configuration  $\sigma_0$  satisfies  $\sigma_0(x) \geq 0$  for all  $x$ , and  $\sum_x \sigma_0(x) < \infty$ . If  $G$  is finite, we assume that  $\sum_x \sigma_0(x)$  is at most the number of vertices of  $G$  (otherwise, some chips would never get absorbed). The only assumption on the approximation  $u_1$  is that it is nonnegative with finite support. Finally, we assume that the rotor stacks are infinitive, which ensures that the growth process terminates after finitely many firings: that is,  $\sum_{x \in V} u(x) < \infty$ .

For  $x \in V$ , write  $d_{out}(x) = \#\{e \in E \mid \mathfrak{s}(e) = x\}$  and  $d_{in}(x) = \#\{e \in E \mid \mathfrak{t}(e) = x\}$  for the out-degree and in-degree of  $x$ , respectively. The odometer function  $u$  depends on the initial chip configuration  $\sigma_0$  and on the rotor stacks  $(\rho_k(x))_{k \geq 0}$ . The latter are completely specified by the function  $R(e, n)$  defined in §3. Note that for rotor-router aggregation, since the stacks are periodic,  $R(e, n)$  has the simple explicit form  $R(e, n) = \left\lfloor \frac{n + d_{out}(x) - j}{d_{out}(x)} \right\rfloor$  where  $j$  is the least positive integer such that  $\rho_j(x) = e$ . For IDLA,  $R(e, n)$  is a random

variable with the Binomial( $n, p$ ) distribution, where  $p$  is the transition probability associated to the edge  $e$ . In this section we take  $R(e, n)$  as known. From a computational standpoint, if the stacks are random, then determining  $R(e, n)$  involves calls to a pseudorandom number generator.

Our algorithm consists of an approximation step followed by two error-correction steps: an annihilation step that corrects the chip locations, and a reverse cycle-popping step that corrects the rotors.

- (1) **Approximation.** Perform firings according to the approximate odometer, by computing the chip configuration  $\sigma_1 = \sigma_0 + \Delta_\rho u_1$ . Using Definition 1, this takes time  $O(d_{in}(x) + 1)$  for each vertex  $x$ , for a total time of  $O(\#E + \#V)$ . This step is where the speedup occurs, because we are performing many firings at once:  $\sum_x u_1(x)$  is typically much larger than  $\#E + \#V$ . Return  $\sigma_1$ .
- (2) **Annihilation.** Start with  $u_2 = u_1$  and  $\sigma_2 = \sigma_1$ . If  $x \in V$  satisfies  $\sigma_2(x) > 1$ , then we call  $x$  a *hill*. If  $\sigma_2(x) < 0$ , or if  $\sigma_2(x) = 0$  and  $u_2(x) > 0$ , then we call  $x$  a *hole*. For each  $x \in \mathbb{Z}^2$ ,
  - (a) If  $x$  is a hill, fire it by incrementing  $u_2(x)$  by 1 and then moving one chip from  $x$  to  $\mathfrak{t}(\text{Top}(u_2)(x))$ .
  - (b) If  $x$  is a hole, unfire it by moving one chip from  $\mathfrak{t}(\text{Top}(u_2)(x))$  to  $x$  and then decrementing  $u_2(x)$  by one.
 A hill can disappear in one of two ways: by reaching an unoccupied site on the boundary, or by reaching a hole and canceling it out. When there are no more hills and holes, return  $u_2$ .
- (3) **Reverse cycle-popping.** Start with  $u_3 = u_2$  and  $A_3 = \{x \in V : u_3(x) > 0\}$ . If  $\text{Top}(u_3)$  is not acyclic on  $A_3$ , then pick a cycle and unfire each of its vertices once. This may create additional cycles. Update  $A_3$  and repeat until  $\text{Top}(u_3)$  is acyclic on  $A_3$ . Output  $u_3$ .

## 5 Experimental Results

We implemented our algorithm for both growth models in  $\mathbb{Z}^2$ . The source code is available from [1]. In this section we discuss some of our results. More experimental findings can be found in the full version [15].

Traditional step-by-step simulation requires quadratic time to compute the occupied cluster [22, 26]. We found experimentally that our algorithm ran in significantly shorter time: about  $N \log N$  for the rotor-router model, and about  $N^{1.5}$  for IDLA.

### 5.1 Rotor-router aggregation

In the classic rotor-router model, the rotor stack is the cyclic sequence of the four cardinal directions in counterclockwise order. The absolute error in our odometer approximation

$$\|u_1 - u\|_1 = \sum_x |u_1(x) - u(x)|$$

appears to scale linearly with  $N$ . This quantity is certainly a lower bound for the running time of our algorithm. The measured runtimes indicate close-to-linear runtime behavior, which suggests that our multiscale approach to canceling out hills and holes is relatively efficient.

The asymptotic shape of rotor-router aggregation is a disk [23, 24]. To measure how close  $A_N$  is to a disk, we define the *inradius* and *outradius* of a set  $A \subset \mathbb{Z}^2$  by  $r_{in}(A) = \min\{|x| : x \notin A\}$  and  $r_{out}(A) = \max\{|x| : x \in A\}$ , respectively. We then define

$$\text{diff}(N) = r_{out}(A_N) - r_{in}(A_N).$$

A natural question is whether this difference is bounded independent of  $N$ . We certainly expect it to increase much more slowly than the order  $\log N$  observed for IDLA.

Kleber [21] calculated that  $\text{diff}(3 \cdot 10^6) \approx 1.6106$ . We can now extend the measurement of  $\text{diff}(N)$  up to  $N = 2^{36} \approx 6.8 \cdot 10^{10}$  and observe in this case for example  $\text{diff}(2^{36}) \approx 1.587$ . Our algorithm runs in less than four hours for this value of  $N$ ; by comparison, a step-by-step simulation of this size would take about 23000 years on a computer with one billion operations per second. In our implementation, the limiting factor is memory rather than time.

Up to dihedral symmetry, there are three different balanced period-4 rotor sequences for  $\mathbb{Z}^2$ : WENS, WNSE and WNES. The notation WENS means that the first four rotors in each stack point respectively west, east, north and south.

Figure 2a shows the radius difference  $\text{diff}(N)$  for various  $N$  for the three different rotor sequences. As these values are rather noisy, we have also calculated and plotted the averages

$$\overline{\text{diff}}(N) := \frac{1}{|I(N)|} \sum_{N' \in I(N)} \text{diff}(N') \quad (3)$$

with  $I(N) = [\frac{N}{2}, \frac{3N}{2}]$  for  $N \leq 10^6$ , and  $I(N) = [N - 5 \cdot 10^5, N + 5 \cdot 10^5]$  for  $N > 10^6$ .

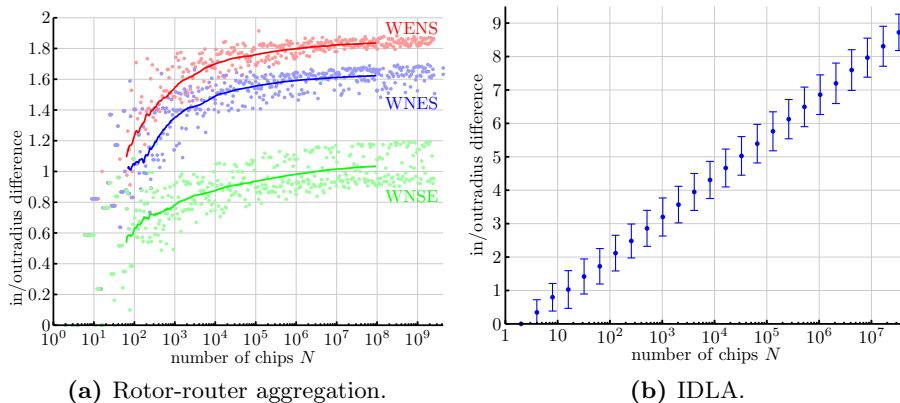
Note that in Figure 2a, the radius difference  $\overline{\text{diff}}(N)$  grows extremely slowly in  $N$ . In particular, it appears to be sublogarithmic.

We observe a systematic difference in behavior for the three different rotor sequences. The observed radius differences are lowest for WNSE, intermediate for WNES, and highest for WENS. For example,

$$\overline{\text{diff}}(10^8) \approx \begin{cases} 1.034 & \text{for WNSE,} \\ 1.623 & \text{for WNES,} \\ 1.837 & \text{for WENS.} \end{cases}$$

## 5.2 Internal Diffusion Limited Aggregation (IDLA)

In IDLA, the rotor directions  $\rho_k(x)$  for  $x \in \mathbb{Z}^2$  and  $k \in \mathbb{Z}$  are chosen independently and uniformly at random from among the four cardinal directions. In the



**Fig. 2:** Difference between the inradius and outradius for the rotor-router aggregate (left) and IDLA (right). In the left figure, the single dots are individual values of  $\text{diff}(N)$  while the darker curves show the averages  $\overline{\text{diff}(N)}$  as defined in equation (3). The right figure shows the average values and standard deviations.

course of firing and unfiring during steps 2 and 3 of our algorithm, the same rotor  $\rho_k(x)$  may be requested several times. Therefore, we need to be able to generate the same pseudorandom value for  $\rho_k(x)$  each time it is used. Generating and storing all rotors  $\rho_k(x)$  for all  $x$  and all  $1 \leq k \leq u_1(x)$  is out of the question, however, since it would cost  $\Omega(N^2)$  time and space.

Moore and Machta [26] encountered the same issue in developing a fast parallel algorithm for IDLA. Rather than store all of the random choices, they chose to store only certain seed values for the random number generator and generate random walk steps online as needed. In the full version of the paper [15] we describe how to adapt this idea to our setting for fast serial computation of IDLA.

The results of our large-scale simulations of IDLA are summarized in Figure 2b, extending the experiments of Moore and Machta [26] ( $N \leq 10^{5.25}$  with 100 trials) to over  $10^6$  trials for  $N \leq 2^{16}$  and over 300 trials for  $N \leq 2^{25} \approx 10^{7.5}$ . The observed runtime of our algorithm for IDLA is about  $N^{1.5}$ ; in contrast, building an IDLA cluster of size  $N$  by serial simulation of  $N$  random walks takes expected time order  $N^2$  (cf. [26, Fig. 3]).

The expected value of the difference  $\text{diff}(N)$  between outradius and inradius grows logarithmically in  $N$ : The data fits to  $\mathbb{E} \text{diff}(N) = 0.528 \ln(N) - 0.457$  with a coefficient of determination of  $R^2 = 0.99994$ . Error bars in figure Figure 2b show standard deviations of the random variable  $\text{diff}(N)$ . Note that the size of the standard deviation is approximately constant: it does not grow with  $N$ . This finding is consistent with the connection with Gaussian free field revealed in [20]: indeed, the maximum of the discrete two-dimensional Gaussian free field over the boundary of a disk of area  $N$  has mean of order  $\log N$  and variance of order 1.

## Acknowledgments

We thank James Propp for many enlightening conversations and for comments on several drafts of this article. We thank David Wilson suggesting the AES random number generator and for tips about how to use it effectively.

## Bibliography

- [1] available at <http://rotor-router.mpi-inf.mpg.de>.
- [2] A. Asselah and A. Gaudillière. From logarithmic to subdiffusive polynomial fluctuations for internal DLA and related growth models, 2010. arXiv:1009.2838.
- [3] A. Asselah and A. Gaudillière. Sub-logarithmic fluctuations for internal DLA, 2010. arXiv:1011.4592.
- [4] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: an explanation of the  $1/f$  noise. *Phys. Rev. Lett.*, 59(4):381–384, 1987.
- [5] R. D. Barve, E. F. Grove, and J. S. Vitter. Simple randomized mergesort on parallel disks. *Parallel Computing*, 23(4-5):601–631, 1997.
- [6] J. Cooper and J. Spencer. Simulating a random walk with constant error. *Combinatorics, Probability & Computing*, 15:815–822, 2006.
- [7] J. Cooper, B. Doerr, J. Spencer, and G. Tardos. Deterministic random walks on the integers. *European Journal of Combinatorics*, 28:2072–2090, 2007.
- [8] J. Cooper, B. Doerr, T. Friedrich, and J. Spencer. Deterministic random walks on regular trees. *Random Structures and Algorithms*, 37(3):353–366, 2010.
- [9] D. Dhar. Theoretical studies of self-organized criticality. *Physica A*, 369:29–70, 2006.
- [10] P. Diaconis and W. Fulton. A growth model, a game, an algebra, Lagrange inversion, and characteristic classes. *Rend. Sem. Mat. Univ. Politec. Torino*, 49(1):95–119, 1991.
- [11] B. Doerr and T. Friedrich. Deterministic random walks on the two-dimensional grid. *Combinatorics, Probability & Computing*, 18(1-2):123–144, 2009.
- [12] B. Doerr, T. Friedrich, and T. Sauerwald. Quasirandom rumor spreading. In *19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, pages 773–781, 2008.
- [13] B. Doerr, T. Friedrich, and T. Sauerwald. Quasirandom rumor spreading: Expanders, push vs. pull, and robustness. In *36th Int. Colloquium on Automata, Languages and Programming (ICALP'09)*, pages 366–377, 2009.

- [14] A. Fey, L. Levine, and Y. Peres. Growth rates and explosions in sandpiles. *J. Stat. Phys.*, 138:143–159, 2010.
- [15] T. Friedrich and L. Levine. Fast simulation of large-scale growth models, 2011. arXiv:1006.1003.
- [16] T. Friedrich and T. Sauerwald. The cover time of deterministic random walks. *Electr. J. Comb.*, 17(1):1–7, 2010. R167.
- [17] T. Friedrich, M. Gairing, and T. Sauerwald. Quasirandom load balancing. In *21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '10)*, pages 1620–1629, 2010.
- [18] A. E. Holroyd and J. G. Propp. Rotor walks and Markov chains. *Algorithmic Probability and Combinatorics*, 520:105–126, 2010.
- [19] D. Jerison, L. Levine, and S. Sheffield. Logarithmic fluctuations for internal DLA, 2010. arXiv:1010.2483.
- [20] D. Jerison, L. Levine, and S. Sheffield. Internal DLA and the Gaussian free field, 2011. arXiv:1101.0596.
- [21] M. Kleber. Goldbug variations. *Math. Intelligencer*, 27(1):55–63, 2005.
- [22] G. F. Lawler, M. Bramson, and D. Griffeath. Internal diffusion limited aggregation. *Ann. Probab.*, 20(4):2117–2140, 1992.
- [23] L. Levine and Y. Peres. Spherical asymptotics for the rotor-router model in  $\mathbb{Z}^d$ . *Indiana Univ. Math. J.*, 57(1):431–450, 2008.
- [24] L. Levine and Y. Peres. Strong spherical asymptotics for rotor-router aggregation and the divisible sandpile. *Potential Anal.*, 30:1–27, 2009.
- [25] P. Meakin and J. M. Deutch. The formation of surfaces by diffusion-limited annihilation. *J. Chem. Phys.*, 85, 1986.
- [26] C. Moore and J. Machta. Internal diffusion-limited aggregation: parallel algorithms and complexity. *J. Stat. Phys.*, 99(3-4):661–690, 2000.
- [27] J. G. Propp and D. B. Wilson. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *J. Algorithms.*, 27:170–217, 1998.
- [28] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Smell as a computational resource – a lesson we can learn from the ant. In *4th Israel Symposium on Theory of Computing and Systems (ISTCS '96)*, pages 219–230, 1996.
- [29] D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *28th Annual ACM Symposium on the Theory of Computing (STOC '96)*, pages 296–303, 1996.