Internal DLA: Efficient Simulation of a Physical Growth Model
(Extended Abstract)

Karl Bringmann[1], Fabian Kuhn[2], Konstantinos Panagiotou[3], Ueli Peter[4], and Henning Thomas[4]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
[2] Department of Computer Science, University of Freiburg, Germany
[3] Department of Mathematics, LMU München, Germany
[4] Institute of Theoretical Computer Science, ETH Zurich, Zürich, Switzerland

Abstract. The internal diffusion limited aggregation (IDLA) process places n particles on the two dimensional integer grid. The first particle is placed on the origin; every subsequent particle starts at the origin and performs an unbiased random walk until it reaches an unoccupied position.

In this work we study the computational complexity of determining the subset that is generated after n particles have been placed. We develop the first algorithm that provably outperforms the naive step-by-step simulation of all particles. Particularly, our algorithm has a running time of $O(n \log^2 n)$ and a sublinear space requirement of $O(n^{1/2} \log n)$, both in expectation and with high probability. In contrast to some speedups proposed for similar models in the physics community, our algorithm samples from the *exact* distribution.

To simulate a single particle fast we have to develop techniques for combining multiple steps of a random walk to large jumps without hitting a forbidden set of grid points. These techniques might be of independent interest for speeding up other problems based on random walks.

1 Introduction

Internal diffusion limited aggregation (IDLA) is a random process that places n particles on the two-dimensional integer grid $\mathbb{Z}^2$. Let $A(i) \subset \mathbb{Z}^2$ denote the set of occupied grid points after placing i particles. The first particle is placed on the origin, i.e., $A(1) = \{(0,0)\}$. From there on, $A(i+1)$ is constructed from $A(i)$ by adding the first grid point in $\mathbb{Z}^2 \setminus A(i)$ that is reached by a random walk on $\mathbb{Z}^2$ starting at the origin.

Particle diffusion processes are of considerable significance in various branches of science. In fact, the IDLA process was introduced by Meakin and Deutch [8], who used it as a model to describe the dynamics of certain chemical and physical processes like corrosion or the melting of a solid around a source of heat. Since then, the study of the typical properties of $A(n)$, and most prominently

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 247–258, 2014.

its "shape," has been the topic of many works. In particular, numerical simulations in [8] indicated that the surface of $A(n)$ is typically extremely smooth such that the fluctuations from a perfect circle are only of logarithmic order. Proving this rigorously turned out to be a difficult and challenging mathematical problem, which was resolved only recently, after many attempts by several different authors (see e.g. [3,7,2,1]), by Jerison, Levine and Sheffield [6].

In the present paper, we try to understand IDLA from a computational perspective by giving an efficient algorithm for determining the set $A(n)$. This line of research is driven by the pursuit to get efficient algorithmic tools for coping with random walks and by the wish to speed up models from physics, so that one may perform larger experiments. Moreover, understanding such models from a computational perspective might add to their understanding in general.

Using the aforementioned results it is easy to see that a direct simulation of every individual step for determining $A(n)$ is likely to require a total time of $\Omega(n^2)$, i.e., time $\Omega(n)$ per particle. Indeed, since $A(n)$ typically resembles a perfect circle, it has a radius of order $n^{1/2}$. Moreover, the random walk of a particle can be viewed as a combination of two independent one-dimensional random walks, one along the horizontal and one along the vertical axis. Thus, if a particle is placed initially at the origin, one of these two random walks has to travel a distance of order $n^{1/2}$ in some direction in order to escape $A(n)$. A quadratic running time then follows immediately from the well-known fact that a one-dimensional random walk of length ℓ in expectation only deviates $\Theta(\ell^{1/2})$ hops from its initial position.

The computational complexity of determining $A(n)$ was studied by Moore and Machta [9]. Among other results they showed that the simulation of IDLA (given a string of random bits) is complete for the class $\mathcal{CC}$ (even in the case of one particle), which is the subset of $\mathcal{P}$ characterized by circuits that are composed of comparator gates only. Moreover in [4], Friedrich and Levine give an algorithm that samples $A(n)$. They do not provide an analysis of the complexity (and it seems a quite difficult task to do so), but their experiments indicate that it scales like $O(n^{3/2})$, while they inherently use space $\Omega(n)$.

In this paper we develop a time and space-efficient algorithm for determining the set $A(n)$. We present the first algorithm that provably improves upon the "naive" step-by-step simulation of the particles.

Theorem 1. *IDLA can be simulated in $O(n \log^2 n)$ time and $O(n^{1/2} \log n)$ space, both in expectation and with high probability*[1]*.*

Our algorithm simulates all particles consecutively. It crucially uses that the shape of $A(n)$ is almost a perfect circle, as discussed above. Let the *in-circle* be the largest circle centred at the origin that contains only occupied grid points. As long as the current particle $n+1$ is within the in-circle of $A(n)$, the random walk will typically stay in $A(n)$ for many steps. Specifically, if the current distance of the particle to the in-circle of $A(n)$ is d, then typically in the next $\Theta(d^2)$ random walk steps the particle will stay in $A(n)$. We want to utilize this fact by combining

[1] With probability $1 - O(n^{-c})$ for a constant $c > 0$ that can be made arbitrary large.

many steps to a single *jump* of the particle, without simulating all of these steps explicitly. Building on this, we use drift analysis to show that typically $O(\log n)$ such jumps are sufficient to simulate one particle. Intuitively, such a combination of steps to a jump simply amounts to sampling the position of the particle after $T \approx d^2$ steps, which can be done by sampling two binomial random variables $\textsc{Bin}(T, \frac{1}{2})$. However, there is an obstacle to this simple intuition: Within $\Theta(d^2)$ steps we leave $A(n)$ with positive probability, so simply jumping to the outcome of $\Theta(d^2)$ steps necessarily introduces an error. As we want to design an *exact* sampling algorithm, we have to overcome this hurdle.

We present a general framework that utilizes jumps to efficiently simulate IDLA in Section 3. A particular jump procedure is discussed in Section 4.

2 Preliminaries

2.1 Notation

We denote by $\textsc{Bin}(n, p)$ a binomial distribution with parameters n and p and by $\log n$ the natural logarithm of n. For $z = (x, y) \in \mathbb{Z}^2$ we let $|z| = (x^2 + y^2)^{1/2}$ be its 2-norm. For $z \in \mathbb{Z}^2$ and $r > 0$ we define the ball with radius r around z as $B_z(r) := \{w \in \mathbb{Z}^2 \mid |z - w| \leq r\}$. We write $\Gamma(z)$ for the set of grid neighbors of $z \in \mathbb{Z}^2$, and for an arbitrary set $S \subseteq \mathbb{Z}^2$ we write ∂S for the set of all position that can be reached from S, i.e.

$$\partial S := \{z \in \mathbb{Z}^2 \setminus S \mid \Gamma(z) \cap S \neq \emptyset\} \quad \text{and} \quad \bar{S} := S \cup \partial S.$$

Whenever it is clear from the context, which particle we are simulating, we will write A for $A(i)$. For an IDLA shape A let $r_I = r_I(A)$ and $r_O = r_O(A)$ be its in- and outradius (rounded for technical reasons), i.e.,

$$r_I := \left\lfloor \min_{x \in \mathbb{Z}^2 \setminus A} |x| \right\rfloor \quad \text{and} \quad r_O := \left\lfloor \max_{x \in A} |x| \right\rfloor + 1.$$

Moreover, we say that $B_0(r_I)$ is the *in-* and $B_0(r_O)$ the *out-circle* of A.

2.2 The Shape of IDLA

Recently, Jerison, Levine and Sheffield proved a long open conjecture which stated that $A(n) = B_0(\sqrt{n/\pi}) \pm O(\log n)$ with high probability.

Theorem 2 (Theorem 1 in [6]). *For every $\gamma > 0$ exists a constant $\alpha = \alpha(\gamma) < \infty$ such that for sufficiently large r*

$$\Pr\left[B_0(r - \alpha \log r) \subset A(\lfloor \pi r^2 \rfloor) \subset B_0(r + \alpha \log r)\right] \geq 1 - r^{-\gamma}. \tag{1}$$

Additionally using $r_O \leq n$, this theorem implies that $r_O - r_I = O(\log n)$, both in expectation and with high probability.

2.3 Random Walks on $\mathbb{Z}$ and $\mathbb{Z}^2$

Let $z = z_0, z_1, z_2, \ldots$ be a random walk starting in $z \in \mathbb{Z}^2$. Here we always consider the standard random walk on $\mathbb{Z}^2$ that chooses each adjacent grid point with probability 1/4. We write $\mathrm{RW}_T(z) = z_T$ for the outcome of a random walk of length T starting in z and abbreviate $\mathrm{RW}_T(0) = \mathrm{RW}_T$. Note that $\mathrm{RW}_T(z) \sim z + \mathrm{RW}_T$.

We also reach each adjacent grid point with probability 1/4 by flipping two coins $c_1, c_2 \in \{1, -1\}$ and choosing the next position to be

$$z + c_1 \cdot (1/2, 1/2) + c_2 \cdot (-1/2, 1/2).$$

This yields the following reformulation of a 2-dimensional random walk as a linear combination of two independent 1-dimensional random walks. In particular, the following lemma allows us to quickly sample from RW_T (if one can sample binomial random variables quickly, we refer to the full version of this paper for a thorough discussion of this assumption).

Lemma 1. *Let $z \in \mathbb{Z}^2$ and $T \in \mathbb{N}$. Let S_T be the sum of T independent uniform $\{1, -1\}$ random variables, $S_T \sim 2\,\mathrm{Bin}(T, 1/2) - T$, and let X, Y be independent copies of S_T. Then*

$$\mathrm{RW}_T(z) \sim z + X \cdot (1/2, 1/2) + Y \cdot (1/2, -1/2).$$

Note that our random walks are "bipartite" in the sense that in even timesteps one can reach only the "even" positions of the grid $\{(x, y) \in \mathbb{Z}^2 \mid x + y \equiv 0 \pmod 2\}$, and similarly for odd timesteps. We write $z \equiv_T x$ if z can be reached from x by a walk of length $\ell \in \mathbb{N}$ with $\ell \equiv T \pmod 2$.

The outcome of a one-dimensional random walk of length T has standard deviation $\Theta(\sqrt{T})$. Intuitively, this implies that with at least constant probability the two-dimensional random walk RW_T is further than $\sqrt{T}$ away from the origin. Moreover, in any direction ξ the expected jump length is large.

Lemma 2. *For any $T \in \mathbb{N}$ we have $\Pr[|\mathrm{RW}_T| \geq \sqrt{T}] \geq \Omega(1)$.*

Moreover, let τ be a symmetric stopping time, i.e., for all $z \in \mathbb{Z}^2$ we have $\Pr[\mathrm{RW}_\tau = z] = \Pr[\mathrm{RW}_\tau = z']$ where z' is obtained from z by rotating it by $90°$. Then for any $\xi \in \mathbb{R}^2$ with $|\xi| = 1$ we have

$$\mathbb{E}[|\xi \cdot \mathrm{RW}_\tau|] = \Omega(\Pr[|\mathrm{RW}_\tau| \geq \sqrt{T}] \cdot \sqrt{T}).$$

2.4 Drift Analysis

Let Ω be some state space, $Y_k \in \Omega$ ($k \in \mathbb{N}$) a stochastic process and $g : \Omega \to \mathbb{R}_{\geq 0}$ a function on Y_k. Let the hitting time τ be the smallest k such that $g(Y_k) = 0$. We say that $g(Y_k)$ has an additive drift of at least ε if for all $0 \leq k < \tau$

$$\mathbb{E}\left[g(Y_{k+1}) - g(Y_k) \mid Y_k\right] < -\varepsilon. \tag{2}$$

The following theorem bounds the expected hitting time by the inverse of the additive drift.

Theorem 3 ([5]). *In the situation of this section we have $\mathbb{E}[\tau] \leq \frac{g(Y_0)}{\varepsilon}$.*

3 A General Framework

The main idea of our algorithms is to combine many steps of a particle's random walk to a jump as long as the current particle $n+1$ is in the in-circle of $A = A(n)$. In this section, we first formalize the notion of a jump. After that we provide a framework that yields an IDLA simulation algorithm for any given jump procedure. Throughout this paper a *step* refers to a single step in a particle's random walk and a *jump* refers to several steps at once.

3.1 The Concept of a Jump

Ideally, a *jump* does multiple steps of a random walk at once to save the effort of simulating every single step. Jumps should be concatenable to form longer portions of a random walk. More formally, let $z = z_0, z_1, \dots$ be a random walk starting in z and $\tau = \tau(A, z)$ a stopping time of this random walk. Then $z \mapsto z_\tau$ defines a jump procedure, and the concatenation of two such jumps is again the outcome of a random walk at a certain stopping time. This concatenation property allows us to add up jumps until we finally hit the boundary ∂A. A jump should make at least one single step of the random walk in order to have guaranteed progress, i.e., we require $\tau \geq 1$ (with probability 1). Moreover, in order to have a correct simulation of IDLA, jumps must stop at the latest when the random walk leaves A, since then the particle's simulation is complete. Additionally, all jump procedures considered in this paper are symmetric around z.

There are two important goals for the design of a jump procedure. First, the (expected) *runtime* to compute the outcome of a jump should be as small as possible. In particular, it should be faster than simulating the random walk step-by-step. Second, intuitively a jump should be the combination of as many single steps as possible. This can be formalized by requiring the *expected jumping distance* to be large. The following definition captures this concept of a jump.

Definition 1. *A* jump procedure *is a randomized algorithm J with input (an IDLA structure) $A \subset \mathbb{Z}^2$ and a point $z \in A$ and output $J(A, z) = z_\tau$, where $z = z_0, z_1, \dots$ is a random walk and $\tau = \tau_J(A, z)$ is any stopping time. We require the jump to make at least one single step of a random walk and to stop at the latest when leaving A for the first time, i.e.,* $\Pr[1 \leq \tau \leq \tau_{\partial A}] = 1$,*where $\tau_{\partial A} = \min\{t \mid z_t \in \partial A\}$ is the hitting time of ∂A. Additionally, J shall be symmetric around z, i.e.,* $\Pr[J(A, z) = z + w] = \Pr[J(A, z) = z - w]$ *for all $w \in \mathbb{Z}^2$.*

We say that J has runtime bound $t_J = t_J(n)$ *if $J(A, z)$ can be computed in time t_J in expectation and with high probability (over the randomness of $A = A(n)$ and $\{z\} = A(n+1) \setminus A(n)$ and the internal randomness of J). Moreover, we define the* expected jumping distance *as*

$$\Delta_J(A, z) := \min_{|\xi|=1} \mathbb{E}[|\xi \cdot (J(A, z) - z)|].$$

When A is clear from the context we also write $J(z)$ for $J(A, z)$.

3.2 From Jumps to IDLA

Any jump procedure can be iterated to find the point where the random walk first leaves the IDLA structure A. Let $z_0 := (0,0)$ and $z_{i+1} := J(A, z_i)$, for every $i = 0, 1, 2, \dots$ as long as z_i is still in A. Moreover, let $\tau^* = \tau^*(J, A) := \min\{i \mid z_i \in \partial A\}$ and $J^* = J^*(A) := z_{\tau^*}$. Note that since J is a randomized algorithm, J^* and τ^* are random variables. Clearly, J^* is distributed exactly as the endpoint of an IDLA particle. This way, any jump procedure gives rise to a simulation algorithm for IDLA.

The following theorem gives an upper bound on the running time of an IDLA simulation with jump procedure J.

Theorem 4. *Let J be a jump procedure with runtime bound t_J. Let Δ_J be its expected jumping distance, $c_J > 0$ some constant, $B_I := B_0(r_I - c_J \log n)$, set*

$$\delta_J(A) := \max_{z \in B_I} \frac{r_O - |z|}{\Delta_J(A, z)}$$

and assume that for some $\bar{\delta}_J = \bar{\delta}_J(n)$ we have $\delta_J(A) \le \bar{\delta}_J$ in expectation and with high probability (over the randomness of $A = A(n)$). Then we can construct an algorithm for simulating IDLA with runtime

$$O(n \cdot t_J \cdot \log n \cdot (\bar{\delta}_J^2 + \log n))$$

and space usage[2] *$O(n^{1/2} \log n)$, both in expectation and with high probability.*

To see that $O(n^{1/2} \log n)$ bits are sufficient (in expectation) to store $A(n)$, note that by Theorem 2 we have with high probability $B_0(\sqrt{n} - O(\log n)) \subseteq A(n) \subseteq B_0(\sqrt{n} + O(\log n))$, and $B_0(\sqrt{n} + O(\log n)) \setminus B_0(\sqrt{n} - O(\log n))$ contains $O(n^{1/2} \log n)$ grid cells, for each of which we can store whether it is occupied in 1 bit.

In Section 4 we present a jump procedure with $t_J = O(1)$ and $\bar{\delta}_J^2 = O(\log n)$, and thereby provide a proof for Theorem 1.

In order to run efficient IDLA simulations, we need a data structure that has the following properties.

Lemma 3. *We can construct a data structure for A that allows us to*
- *query r_I and r_O in $O(1)$ time,*
- *check $z \in A$ in $O(1)$ time, and*
- *add $z \in \mathbb{Z}^2$ to A.*

Adding the n particles of an IDLA simulation one-by-one to this data structure overall needs $O(n)$ time and $O(n^{1/2} \log n)$ space, both in expectation and with high probability.

In this extended abstract we omit the description of the data structure and the proof of Lemma 3. In the following section, we analyse the expected number of jumps that we need to simulate. Then, Theorem 4 is merely a consequence of Theorem 2, Lemma 3 and Lemma 4 below, and we therefore omit its proof.

[2] Not including the space used by the jump function.

3.3 Number of Jumps

To bound the expected runtime of the simulation of a particle using a jump procedure J, we only have to bound the hitting time τ^* of ∂A. The following lemma provides such a bound.

Lemma 4. *With the notation of Section 3.2 for any IDLA structure A and $k > 0$ we have* $\Pr[\tau^* \geq k\,(\delta_J^2(A)\log n + (r_O - r_I + \log n)^2)] \leq \exp(-\Omega(k))$.
In particular, we have $\mathbb{E}[\tau^*] \leq O\left(\delta_J^2(A)\log n + (r_O - r_I + \log n)^2\right)$.

Proof. Consider again the stochastic process $z_0 = (0,0)$, and $z_{k+1} = J(A, z_k)$ for $k > 0$. Set $\sigma := \sqrt{2}(r_O - r_I + c_J \log n)$. We analyze this process in phases. The process starts in phase 1 and changes to phase 2 the first time it reaches a position $z_k \notin B_I$. For the next σ^2 jumps the process stays in phase 2. After that it returns to phase 1, except if we are again outside B_I, then we directly start another phase 2. This repeats until we hit ∂A. For these phases we prove the following.

(1) Starting phase 1 anywhere in B_I, we stay in this phase for at most $O(\delta_J^2(A)\log n)$ jumps in expectation.
(2) Starting phase 2 anywhere outside B_I, the probability of hitting ∂A before the end of the phase is $\Omega(1)$.

Using Markov's inequality, (1) implies that after at most $O(\delta_J^2(A)\log n)$ jumps we leave phase 1 with probability $\Omega(1)$. Together with (2) we obtain that, wherever we start, within $O(\delta_J^2(A)\log n + \sigma^2)$ jumps we hit ∂A with probability $\Omega(1)$. Hence, within $O(k(\delta_J^2(A)\log n + \sigma^2))$ jumps we hit ∂A with probability $1-\exp(-\Omega(k))$, yielding both expectation and concentration of the hitting time.

The proof of (2) follows by Lemma 2 and standard calculations and we therefore omit it in this extended abstract.

To show (1) we apply additive drift analysis to prove that the stochastic process $z_0, z_1, \ldots, z_\tau$ (for $z_0 \in B_I$ and $\tau := \min\{k \mid z_k \notin B_I\}$) has an expected hitting time as claimed. In order to apply Theorem 3 we need a suitable distance function $g : \mathbb{Z}^2 \to \mathbb{R}_{\geq 0}$. We let

$$g(z) := \begin{cases} \log(r_O + 2 - |z|), & z \in B_I \\ 0, & z \in \mathbb{Z}^2 \setminus B_I \end{cases}.$$

In the following we will show that g has an additive drift of $\min_{z \in B_I} \frac{(\Delta_J(A,z))^2}{2(r_O+2-|z|)^2}$ for all $0 \leq k < \tau$, i.e., for any $z_k \in B_I$

$$\mathbb{E}\left[g(z_{k+1}) - g(z_k) \mid z_k\right] \leq -\min_{z \in B_I} \frac{(\Delta_J(A,z))^2}{2(r_O + 2 - |z|)^2}. \tag{3}$$

Applying Theorem 3 together with $g(z) \leq O(\log n)$ then yields an expected hitting time of $\mathbb{Z}^2 \setminus B_I$ of $O(\delta_J^2(A)\log n)$.

Whenever $z_k \in A$ we know that $z_{k+1} \in \bar{A} \subseteq B_0(r_O + 1)$. In this case we can bound $g(z_{k+1}) \leq \log(r_O + 2 - |z_{k+1}|)$. To shorten notation we let $L(x) := \log(r_O +$

$2 - x$) for any $x \in \mathbb{R}$ in the remainder of this proof. Hence, the expectation of $g(z_{k+1})$ conditioned on z_k, $z_k \in B_I$, is at most[3]

$$\sum_{x\in\mathbb{Z}^2} \Pr\left[z_{k+1} = x \mid z_k\right] \cdot L(|x|) \leq \sum_{x\in\mathbb{Z}^2} \Pr\left[z_{k+1} = x \mid z_k\right] \cdot L\left(x\frac{z_k}{|z_k|}\right),$$

since the length of the projection of x is bounded by $|x|$ in any direction. Using the transformation $y_x := x - z_k$ and the symmetry of jump procedures we can rewrite this as

$$\begin{aligned}&\sum_{x\in\mathbb{Z}^2} \Pr\left[z_{k+1} = x|z_k\right] \cdot L\left(|z_k| - y_x\frac{z_k}{|z_k|}\right)\\ &= \frac{1}{2}\sum_{x\in\mathbb{Z}^2} \Pr\left[z_{k+1} = x|z_k\right] \cdot \left(L\left(|z_k| - y_x\frac{z_k}{|z_z|}\right) + L\left(|z_k| + y_x\tfrac{z_k}{|z_k|}\right)\right), \qquad (4)\end{aligned}$$

where $|y_x\frac{z_k}{|z_k|}| \leq r_O + 1 - |z_k|$ for all x with $\Pr\left[z_{k+1} = x|z_k\right] > 0$.

Now we use the following estimate that holds for any $a, b \in \mathbb{R}$ with $a > 0$ and $|b| \leq a$:

$$\log(a + b) + \log(a - b) \leq 2\log(a) - \frac{b^2}{a^2}. \qquad (5)$$

Combining (4) and (5) yields

$$\begin{aligned}\mathbb{E}\left[g(z_{k+1})|z_k\right] &\leq \frac{1}{2}\sum_{x\in\mathbb{Z}^2} \Pr\left[z_{k+1} = x|z_k\right] \cdot \left(2L(|z_k|) - \frac{(y_x \cdot z_k/|z_k|)^2}{(r_O + 2 - |z_k|)^2}\right)\\ &= g(z_k) - \frac{\mathbb{E}\left[(y_{z_{k+1}} \cdot z_k/|z_k|)^2 \middle| z_k\right]}{2(r_O + 2 - |z_k|)^2} = g(z_k) - \frac{\mathbb{E}\left[|(z_{k+1} - z_k)\frac{z_k}{|z_k|}|^2 \middle| z_k\right]}{2(r_O + 2 - |z_k|)^2}\\ &\leq g(z_k) - \frac{\mathbb{E}\left[|(z_{k+1} - z_k)\frac{z_k}{|z_k|}| \middle| z_k\right]^2}{2(r_O + 2 - |z_k|)^2} \qquad (6)\end{aligned}$$

where the last inequality follows from Jensen's inequality. Considering the definition of the expected jumping distance Δ_J (Definition 1) with $\xi = \frac{z_k}{|z_k|}$ we obtain

$$\mathbb{E}\left[g(z_{k+1})|z_k\right] \leq g(z_k) - \frac{(\Delta_J(A, z_k))^2}{2(r_O + 2 - |z_k|)^2}$$

which proves the drift inequality (3) and, thus, the lemma. □

4 Long Jumps

Consider a particle at position $z \in B_I = B_0(r_I - c_J \log n)$ (for some sufficiently large constant $c_J > 0$) and consider the ball $S := B_z(\sigma)$ with midpoint z and

[3] Here we define the corresponding summand to be 0 whenever the log is undefined.

radius $\sigma := r_I - |z|$, so that S is contained in $B_0(r_I) \subseteq A$. Let $z_0, z_1, \ldots$ be a random walk starting in $z_0 = z$, let $\tau_{\partial S} := \min\{i \mid z_i \in \partial S\}$ be its hitting time of the boundary of S, and similarly let $\tau_{\partial A} := \min\{i \mid z_i \in \partial A\}$. Our procedure will directly jump to $J_{\text{long}}(z) := z_\tau$ with

$$\tau := \min\{\tau_{\partial S}, T\} \quad \text{and} \quad T := \left\lfloor \frac{\sigma^2}{c_J \ln(n/\mathrm{e})} \right\rfloor.$$

Whenever $z \notin B_I$, we simply make one step of the random walk, i.e., $\tau := 1$. This way we make sure that $\tau \geq 1$ (for all $z \in A$). Note that here we use $\tau_{\partial S}$ to ensure $\tau \leq \tau_{\partial S} \leq \tau_{\partial A}$, meaning that we stop at the latest when leaving A. Since τ is a stopping time and J_{long} is symmetric, this is a valid jump procedure according to Definition 1. It is not clear at first sight that J_{long} can be sampled efficiently for all $z \in B_I$. However, we present an algorithm in the next section and prove in Section 4.2 that its expected runtime is constant. Finally, we determine the expected jumping distance of J_{long} in Section 4.3. Overall, we obtain the following result, which together with Theorem 4 proves our main result.

Lemma 5. *The jump procedure J_{long} has runtime bound $t_{J_{\text{long}}} = O(1)$ and for any $z \in B_I$ an expected jumping distance of $\Delta_{J_{\text{long}}}(A, z) = \Omega(\sqrt{T}) = \Omega\left(\frac{r_I(A)-|z|}{\sqrt{\log n}}\right)$. Furthermore, it has a space usage of $O(1)$ memory cells (in expectation and with high probability).*

4.1 An Algorithm for Sampling Long Jumps

Observe that with high probability a random walk of length T starting in z does not leave S. Hence, the minimum of $\tau_{\partial S}$ and T is typically obtained at T. We will design an algorithm that samples the position of z_T (restricted to a certain subset) very efficiently. Additionally, we have to patch this approximate algorithm by a second (slow) algorithm that is executed only with small probability and that compensates for any mistakes we might make by sampling only z_T.

First consider Algorithm 1, which does not yet correctly sample a jump according to the distribution of $J_{\text{long}}(z)$. It simply draws a point $z' = \mathrm{RW}_T(z)$ (by sampling from a binomial random variable, see Lemma 1) and rejects as long as $z' \notin \frac{1}{2}S$ (where $\frac{1}{2}S$ is the ball with midpoint z and radius $\frac{1}{2}\sigma$).

Algorithm 1. Algorithm Long-Jump-Incomplete

```
repeat
    z' := RW_T(z)
until z' ∈ (1/2)S
return z'.
```

For $w \in \mathbb{Z}^2$ let $P_J(w) := \Pr[J_{\text{long}}(z) = w]$ and denote the probability of Algorithm 1 to return w by $P_{\text{Alg1}}(w)$. To patch Algorithm 1 we choose a failure probability p_{fail} (to be fixed later). Then, with probability $1 - p_{\text{fail}}$ we run

Algorithm 1, but with probability p_{fail} we patch the algorithm by exhaustively computing the probabilities $P_J(w)$ and $P_{\text{Alg1}}(w)$ for all $w \in \bar{S}$ and returning $w \in \bar{S}$ with probability $P_{\text{rest}}(w)$, where

$$(1 - p_{\text{fail}}) \cdot P_{\text{Alg1}}(w) + p_{\text{fail}} \cdot P_{\text{rest}}(w) = P_J(w). \tag{7}$$

The above equation ensures that overall we draw $w \in \mathbb{Z}^2$ according to the right probability distribution P_J. The approach is summarized in Algorithm 2.

Algorithm 2. Algorithm Long-Jump-Complete

```
choose p uniformly at random from [0, 1].
if p < p_fail then
    calculate P_J(w) and P_Alg1(w) for all w ∈ S̄
    compute P_rest(w) according to equation (7)
    return w ∈ S̄ drawn according to the distribution P_rest(w)
else
    run Algorithm 1
end if
```

This algorithm is correct if p_{fail} can be chosen in such a way that P_{rest} is a probability distribution. The following lemma states for which values of p_{fail} this is the case.

Lemma 6. *The values $P_{\text{rest}}(w)$ for $w \in \bar{S}$ form a probability distribution if we choose $p_{\text{fail}} \geq 28\mathrm{e}^{c_J/2} n^{-\min\{c_J/8, 5c_J/16-1\}}$.*

In this extended abstract we omit the technical proof of Lemma 6. In the remainder of this section we analyze the runtime of our algorithm and prove a lower bound on the expected jump length.

4.2 Runtime of the Algorithm

In the fail compensation part of our algorithm we have to compute P_{Alg1} and P_J exactly. In this section we discuss how to do this efficiently, which yields a bound on the runtime of our algorithm.

Observe that for $P_{\text{RW}}(w) := \Pr[\text{RW}_T(z) = w]$ we have for all $w \in \frac{1}{2}S$ that

$$P_{\text{Alg1}}(w) = P_{\text{RW}}(w) / \sum_{w \in \frac{1}{2}S} P_{\text{RW}}(w).$$

This reduces the calculation of P_{Alg1} to the calculation of $P_{\text{RW}}(w)$ for all $w \in \frac{1}{2}S$. For $w \not\equiv_T z$ we have $P_{\text{RW}}(w) = 0$, so let $w \equiv_T z$. Then we can write $w = x \cdot (1/2, 1/2) + y \cdot (1/2, -1/2)$ with $x, y \in \mathbb{Z}$. With the notation of Lemma 1 we have

$$P_{\text{RW}}(w) = \Pr[X = x] \cdot \Pr[Y = y] = 2^{-T} \binom{T}{\frac{T+x}{2}} \cdot 2^{-T} \binom{T}{\frac{T+y}{2}}.$$

Note that this probability has denominator 4^T, so it can be stored using $O(T)$ bits. Moreover, as $\binom{T}{i}$ can be computed in $O(T)$ multiplications and divisions of a $O(T)$ bit number by a $O(\log T)$ bit number, we can calculate $P_{\mathrm{RW}}(W)$ in time $O(T^2 \log T)$. The total running time for calculating P_{Alg1} is therefore $O(\sigma^2 T^2 \log T)$ and the occupied space is $O(\sigma^2 T)$.

For computing P_J we use a simple iterative scheme. We recursively define X_w^t for $0 \le t \le T$ and $w \in \bar{S}$. For $t = 0$ we set

$$X_w^0 = \begin{cases} 1 & \text{if } w = z, \\ 0 & \text{otherwise,} \end{cases}$$

while for $t > 0$ we set

$$X_w^t = \begin{cases} \sum_{v \in \Gamma(w) \cap S} \frac{1}{4} X_v^{t-1} & \text{if } w \in S, \\ X_w^{t-1} + \sum_{v \in \Gamma(w) \cap S} \frac{1}{4} X_v^{t-1} & \text{if } w \in \partial S. \end{cases}$$

Observe that X_w^T is equal to $P_J(w)$ for every $w \in \bar{S}$, and each probability X_w^t can be stored using $O(T)$ bits. The total running time to calculate P_J is therefore $O(\sigma^2 T^2)$ and the space usage is $O(\sigma^2 T)$ bits.

As the ball S is completely filled with particles, we have $n \ge \sigma^2$. Using $T = \Theta(\sigma^2 / \log n)$ we get a runtime of $O(n^3)$ and a space usage of $O(n^2)$ for computing P_J and P_{Alg1}.

Clearly, Algorithm 1 runs in expected constant time. Moreover, as the probability of $\mathrm{RW}_T \notin \frac{1}{2}S$ is small (smaller than p_{fail}, as chosen in the last section), it even runs in $O(1)$ time with high probability. In total, the expected runtime of our algorithm for sampling long jumps is $O(1 + p_{\mathrm{fail}} \cdot n^3)$, and the probability of having runtime larger than $O(1)$ is at most $O(p_{\mathrm{fail}})$. Hence, for sufficiently large constant c_J, so that Lemma 6 allows us to choose p_{fail} sufficiently small, we obtain a runtime of $t_{J_{\mathrm{long}}} = O(1)$, both in expectation and with high probability. This proves the first part of Lemma 5.

4.3 Expected Jumping Distance

In this section we analyze the expected jumping distance $\Delta_{J_{\mathrm{long}}}(z)$ of long jumps, proving the second part of Lemma 5. Recall that the expected jumping distance at $z \in B_I$ is defined as

$$\Delta_{J_{\mathrm{long}}}(A, z) = \min_{|\xi|=1} \mathbb{E}[|\xi^T (J_{\mathrm{long}}(A, z) - z)|].$$

Since the stopping time τ of J_{long} is symmetric, we can use the second part of Lemma 2 to obtain $\Delta_{J_{\mathrm{long}}}(A, z) = \Omega(\Pr[|J_{\mathrm{long}}(A, z) - z| \ge \sqrt{T}] \cdot \sqrt{T})$. Observe that we have $\Pr[|J_{\mathrm{long}}(A, z) - z| \ge \sqrt{T}] \ge \Pr[|\mathrm{RW}_T| \ge \sqrt{T}]$, where the inequality comes from some walks in $\mathrm{RW}_{\min\{\tau_{\partial S}, T\}}(z)$ ending prematurely (if $\tau_{\partial S} \le T$). Together with the first part of Lemma 2, this shows $\Delta_{J_{\mathrm{long}}}(A, z) \ge \Omega(\sqrt{T}) = \Omega((r_I(A) - |z|)/\sqrt{\log n})$.

References

1. Asselah, A., Gaudilliere, A.: From logarithmic to subdiffusive polynomial fluctuations for internal DLA and related growth models, arXiv preprint arXiv:1009.2838 (2010)
2. Asselah, A., Gaudilliere, A.: Lower bounds on fluctuations for internal DLA. Probability Theory and Related Fields, 1–15 (2011)
3. Diaconis, P., Fulton, W.: A growth model, a game, an algebra, Lagrange inversion, and characteristic classes. Rend. Sem. Mat. Univ. Politec. Torino 49(1), 95–119 (1991)
4. Friedrich, T., Levine, L.: Fast simulation of large-scale growth models. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) RANDOM 2011 and APPROX 2011. LNCS, vol. 6845, pp. 555–566. Springer, Heidelberg (2011)
5. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3(1), 21–35 (2004)
6. Jerison, D., Levine, L., Sheffield, S.: Logarithmic fluctuations for internal DLA. J. Amer. Math. Soc. 25(1), 271–301 (2012)
7. Lawler, G.F., Bramson, M., Griffeath, D.: Internal diffusion limited aggregation. The Annals of Probability, 2117–2140 (1992)
8. Meakin, P., Deutch, J.M.: The formation of surfaces by diffusion limited annihilation. The Journal of Chemical Physics 85, 2320 (1986)
9. Moore, C., Machta, J.: Internal diffusion-limited aggregation: Parallel algorithms and complexity. Journal of Statistical Physics 99, 661–690 (2000)