Internal DLA: Efficient Simulation of a Physical Growth Model

Karl Bringmann[1], Fabian Kuhn[2], Konstantinos Panagiotou[3], Ueli Peter[4], and Henning Thomas[5]

[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 [15], 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 its "shape," has been the topic of many works. In particular, numerical simulations in [15] 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. [6,13,2,1]), by Jerison, Levine and Sheffield [11].

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. The latter allows one to run large experiments and it might also save energy and money, as experiments on simulated models are usually carried out on expensive clusters. 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 [16]. Among other results they showed that the simulation of one particle (given a string of random bits) is complete for the class $\mathcal{CC}$, which is the subset of $\mathcal{P}$ characterized by circuits that are composed of comparator gates only. Moreover in [9], 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*[5]*.*

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

[5] 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. Different jump procedures are discussed in Section **??**, and our best jump procedure follows 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 $B_z(r) := \{w \in \mathbb{Z}^2 \mid |z - w| \le r\}$. 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 [11]). *For every $\gamma > 0$ exists a constant $\alpha = \alpha(\gamma) < \infty$ such that for sufficiently large r*

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

Additionally using $r_O \le n$, 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 ([10]). *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 *if $J(A, z)$ can be computed in time t_J in expectation and with high probability. Moreover, we define the* expected jumping distance $\Delta_J \colon A \to \mathbb{R}_{\geq 0}$ *by*

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

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 $i = 0, 1, 2, \ldots$ 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*

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

and assume that $\delta_J(A) \leq \bar{\delta}_J$ in expectation and with high probability over A. 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[6] $O(n^{1/2} \log n)$, 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 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.

[6] 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* $\Pr[\tau^* \geq k\,(\delta_J^2 \log n + (r_O - r_I + \log n)^2)] \leq \exp(-\Omega(k))$.

In particular, we have $\mathbb{E}[\tau^*] \leq O\left(\delta_J^2 \log n + (r_O - r_I + \log n)^2\right)$.

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 $O(\delta_J^2 \log n)$ 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 $O(\delta_J^2 \log n)$ jumps we leave phase 1 with probability $\Omega(1)$. Together with (2) we obtain that, wherever we start, within $O(\delta_J^2 \log n + \sigma^2)$ jumps we hit ∂A with probability $\Omega(1)$. Hence, within $O(k(\delta_J^2 \log n + \sigma^2))$ 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 $\min_{z \in B_I} \frac{(\Delta_J(z))^2}{2(r_O+2-|z|)^2}$ for all $0 \leq k < \tau$, i.e., for any $z_k \in B_I$

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

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 $O(\delta_J^2 \log n)$.

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 $g(z_{k+1}) \leq \log(r_O + 2 - |z_{k+1}|)$. Hence, the expectation of $g(z_{k+1})$ conditioned on z_k, $z_k \in B_I$, is at most[7]

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

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

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

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 $|b| \leq a$:

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

Combining (4) and (5) yields

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

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

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

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

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

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 radius $\sigma := r_I - |z|$, so that S is contained in $B_0(r_I) \subseteq A$. Let $z_0, z_1, \dots$ 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 $J_{long}(z) := z_\tau$ with

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

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 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 $t_{J_{long}} = O(1)$ and for any $z \in B_I$ an expected jumping distance of $\Delta_{J_{long}}(z) = \Omega(\sqrt{T}) = \Omega\left(\frac{r_I - |z|}{\sqrt{\log n}}\right)$. 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 the mistake we make by sampling only z_T.

First consider Algorithm 1, which does not yet correctly sample a jump according to the distribution of $J_{long}(z)$. 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 $\frac{1}{2}\sigma$).

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

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

Algorithm 1 Algorithm Long-Jump-Incomplete

repeat
  $z' := \mathrm{RW}_T(z)$
until $z' \in \frac{1}{2}S$
return z'.

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 $w \in \bar{S}$
  compute $P_{rest}(w)$ according to equation (7)
  return $w \in \bar{S}$ 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 $P_{rest}(w)$ for $w \in \bar{S}$ form a probability distribution if we choose $p_{fail} \geq 28\mathrm{e}^{c_J/2} n^{-\min\{c_J/8, 5c_J/16-1\}}$.*

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

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

This reduces the calculation of P_{Alg1} to the calculation of $P_{RW}(w)$ for all $w \in \frac{1}{2}S$. For $w \not\equiv_T z$ we have $P_{RW}(w) = 0$, 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

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

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 $P_{RW}(W)$ 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 $O(1 + p_{fail} \cdot n^3)$, and the probability of having runtime larger than $O(1)$ is at most $O(p_{fail})$. Hence, for sufficiently large constant c_J, so that Lemma 6 allows to choose p_{fail} sufficiently small, we obtain a runtime of $t_{J_{long}} = O(1)$, 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 $\Delta_{J_{long}}(z)$ of long jumps, proving the second part of Lemma 5.

Recall that the expected jumping distance at $z \in B_I$ is defined as

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

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

References

1. A. Asselah and A. Gaudilliere, *From logarithmic to subdiffusive polynomial fluctuations for internal DLA and related growth models*, arXiv preprint arXiv:1009.2838 (2010). (Cited on page 2.)
2. ———, *Lower bounds on fluctuations for internal DLA*, Probability Theory and Related Fields (2011), 1–15. (Cited on page 2.)
3. D. Belazzougui, F. C. Botelho, and M. Dietzfelbinger, *Hash, displace, and compress*, Algorithms - ESA 2009 (A. Fiat and P. Sanders, eds.), Lecture Notes in Computer Science, vol. 5757, Springer, 2009, pp. 682–693. (Cited on page 20.)
4. K. Bringmann and T. Friedrich, *Exact and efficient generation of geometric random variates and random graphs*, Automata, Languages, and Programming (ICALP'13) (F. V. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg, eds.), Lecture Notes in Computer Science, vol. 7965, Springer, 2013, pp. 267–278. (Cited on pages 16 and 18.)
5. L. Devroye, *Non-uniform random variate generation*, Springer-Verlag, 1986. (Cited on page 15.)
6. P. Diaconis and W. Fulton, *A growth model, a game, an algebra, Lagrange inversion, and characteristic classes*, Rend. Sem. Mat. Univ. Politec. Torino **49** (1991), no. 1, 95–119. (Cited on page 2.)
7. D. P. Dubhash and A. Panconesi, *Concentration of measure for the analysis of randomized algorithms*, Cambridge University Press, 2009. (Cited on page 22.)
8. M. Farach-Colton and M. Tsai, *Exact sublinear binomial sampling*, Algorithms and Computation (ISAAC'13) (L. Cai, S. Cheng, and T. Lam, eds.), Lecture Notes in Computer Science, vol. 8283, Springer, 2013, pp. 240–250. (Cited on page 16.)
9. T. Friedrich and L. Levine, *Fast simulation of large-scale growth models*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (2011), 555–566. (Cited on page 2.)
10. J. He and X. Yao, *A study of drift analysis for estimating computation time of evolutionary algorithms*, Natural Computing **3** (2004), no. 1, 21–35. (Cited on page 4.)
11. D. Jerison, L. Levine, and S. Sheffield, *Logarithmic fluctuations for internal DLA*, J. Amer. Math. Soc **25** (2012), no. 1, 271–301. (Cited on pages 2 and 3.)
12. C. Lanczos, *A precision approximation of the gamma function*, Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis **1** (1964), no. 1, 86–96. (Cited on page 18.)
13. G. F. Lawler, M. Bramson, and D. Griffeath, *Internal diffusion limited aggregation*, The Annals of Probability (1992), 2117–2140. (Cited on page 2.)
14. G. F. Lawler and V. Limic, *Random walk: A modern introduction*, Cambridge Studies in Advanced Mathematics, Cambridge University Press, 2010. (Cited on pages 13 and 15.)
15. P. Meakin and J. M. Deutch, *The formation of surfaces by diffusion limited annihilation*, The Journal of chemical physics **85** (1986), 2320. (Cited on page 1.)
16. C. Moore and J. Machta, *Internal diffusion-limited aggregation: Parallel algorithms and complexity*, Journal of Statistical Physics **99** (2000), 661–690. (Cited on page 2.)
17. J. Spouge, *Computation of the gamma, digamma, and trigamma functions*, SIAM Journal on Numerical Analysis **31** (1994), no. 3, 931–944. (Cited on pages 18 and 19.)

A Proofs and Lemmas omitted from Section 2

A.1 Random Walks

In this section we present several easy or known facts about random walks that are used throughout this paper. For some of these facts we need explicit constants in the error terms. Thus, although variants incorporating O-notation could be found, e.g, in [14], we need to reprove them with explicit constants.

First, we need bounds for the probability of RW_T to end up in a particular point $z \in \mathbb{Z}^2$. A proof of this statement amounts to Stirling's approximation on binomial coefficients with explicitly bounding the error term constants.

Lemma 7. *Let $T \in \mathbb{N}$ and $z \in \mathbb{Z}^2$ with $|z| \le \frac{1}{2}T$ and $z \equiv_T (0,0)$. Then we have*

$$\Pr[\mathrm{RW}_T = z] \le \frac{2}{T} \exp\Big(- \frac{|z|^2}{2T}\Big(1 - \frac{2|z|}{T}\Big)\Big),$$
$$\Pr[\mathrm{RW}_T = z] \ge \frac{1}{3T} \exp\Big(- \frac{|z|^2}{2T}\Big(1 + \frac{|z|}{T}\Big)\Big).$$

Proof. Let $z = x \cdot (1/2, 1/2) + y \cdot (1/2, -1/2)$. Without loss of generality we can assume that $x, y \ge 0$. With the notation of Lemma 1 we have

$$\Pr[\mathrm{RW}_T = z] = \Pr[X = x] \cdot \Pr[Y = y].$$

Counting the number of paths on $\mathbb{Z}$ from 0 to x (or y) yields

$$\Pr[\mathrm{RW}_T = z] = 2^{-T}\binom{T}{\frac{T+x}{2}} \cdot 2^{-T}\binom{T}{\frac{T+y}{2}}. \tag{8}$$

Using Stirling's approximation

$$n! = r_1\sqrt{2\pi n}\Big(\frac{n}{\mathrm{e}}\Big)^n \quad \text{with } 1 \le r_1 \le 1.1$$

yields after some simplifications

$$2^{-T}\binom{T}{\frac{T+x}{2}} = r_2\sqrt{\frac{2T}{\pi(T^2 - x^2)}}\Big(1 - \frac{x^2}{T^2}\Big)^{-T/2}\Big(1 + \frac{x}{T}\Big)^{-x/2}\Big(1 - \frac{x}{T}\Big)^{x/2},$$

with $(1.1)^{-2} \le r_2 \le 1.1$. Note that by assumption $x, y \le |z| \le T/2$. This allows to bound $T^2 \ge T^2 - x^2 \ge \frac{3}{4}T^2$. Furthermore, for $a \ge -\frac{1}{2}$ we have $\exp(a(1-a)) \le 1 + a \le \exp(a)$. Together we get

$$\begin{aligned} 2^{-T}\binom{T}{\frac{T+x}{2}} &\le 1.1\sqrt{\frac{8}{3\pi T}} \exp\Big(\frac{x^2}{2T}\Big(1 + \frac{x^2}{T^2}\Big) - \frac{x^2}{2T}\Big(1 - \frac{x}{T}\Big) - \frac{x^2}{2T}\Big) \\ &\le 1.1\sqrt{\frac{8}{3\pi T}} \exp\Big(- \frac{x^2}{2T}\Big(1 - \frac{2x}{T}\Big)\Big), \end{aligned}$$

and, analogously,

$$2^{-T}\binom{T}{\frac{T+x}{2}} \geq (1.1)^{-2}\sqrt{\frac{2}{\pi T}}\exp\Big(-\frac{x^2}{2T}\Big(1+\frac{x}{T}\Big)\Big).$$

Plugging this into equation (8), using $|z|^2 = x^2+y^2$ and $x^3+y^3 \leq (x^2+y^2)^{3/2} = |z|^3$, and rounding the factors, in total we obtain

$$\begin{aligned}\Pr[\mathrm{RW}_T = z] &\leq (1.1)^2\frac{8}{3\pi T}\exp\Big(-\frac{x^2+y^2}{2T}+\frac{x^3+y^3}{T^2}\Big)\\ &\leq \frac{2}{T}\exp\Big(-\frac{|z|^2}{2T}\Big(1-\frac{2|z|}{T}\Big)\Big),\end{aligned}$$

and, analogously,

$$\Pr[\mathrm{RW}_T = z] \geq \frac{1}{3T}\exp\Big(-\frac{|z|^2}{2T}\Big(1+\frac{|z|}{T}\Big)\Big),$$

Proof (Lemma **??***).* For any small constant T the claim clearly holds. In the remaining cases we proceed as follows. For any $d \geq 1$ the slice $S := \{z \in \mathbb{Z}^2 \mid d \leq |z| < 2d\}$ contains $\Omega(d^2)$ grid points. Using Lemma 7 this yields

$$\Pr[\mathrm{RW}_T \geq d] \geq \Pr[\mathrm{RW}_T \in S] \geq \Omega\left(\frac{d^2}{T}\exp\left(-\frac{(2d)^2}{2T}\Big(1+\frac{2d}{T}\Big)\right)\right).$$

For $d = \sqrt{T}$ the right hand side is $\Omega(1)$.

For the second statement, let $w_1, \ldots, w_4$ be symmetric points around the origin, i.e., $w_1, \ldots, w_4$ form a square with midpoint the origin. Then there is a point w_i which forms an angle of at most 45 degrees with ξ, so that we have $|\xi \cdot w_i| = \Omega(|w_i|)$. Thus, we have

$$\sum_{i=1}^{4}|\xi \cdot w_i| = \Omega(|w_i|) = \Omega\Big(\sum_{i=1}^{4}|w_i|\Big).$$

As the stopping time is symmetric, we can use the above symmetry argument as follows,

$$\begin{aligned}\mathbb{E}[|\xi \cdot \mathrm{RW}_\tau|] &= \sum_{w\in\mathbb{Z}^2}\Pr[\mathrm{RW}_\tau = w]\cdot|\xi\cdot w|\\ &= \sum_{w\in\mathbb{Z}^2}\Pr[\mathrm{RW}_\tau = w]\cdot\Omega(|w|).\end{aligned}$$

A rough upper bound now yields the claim,

$$\begin{aligned}\mathbb{E}[|\xi \cdot \mathrm{RW}_\tau|] &\geq \Omega\Big(\sum_{\substack{w\in\mathbb{Z}^2\\ |w|\geq\sqrt{T}}}\Pr[\mathrm{RW}_\tau = w]\cdot\sqrt{T}\Big)\\ &= \Omega\big(\Pr[|\mathrm{RW}_\tau| \geq \sqrt{T}]\cdot\sqrt{T}\big).\end{aligned}$$

Finally, the Chernoff bound yields a tail bound for the probability of RW_T to end up too far away from the origin. Together with the reflection principle (see [14]) this yields a tail bound for the probability of being too far away from the origin at any time $0 \leq t \leq T$.

Lemma 8. *For any $T, k \in \mathbb{N}$ we have*

$$\Pr[|\mathrm{RW}_T| > k] \leq 4\mathrm{e}^{-\frac{k^2}{2T}}.$$

Moreover, let $0 = z_0, z_1, \ldots$ be a random walk and set $\tau := \min_{t \geq 0}\{|z_t| > k\}$. Then

$$\Pr[\tau \leq T] \leq 8\mathrm{e}^{-\frac{k^2}{2T}}.$$

Proof. Let $X_1, X_2, \ldots$ be independent copies of a uniform $\{1, -1\}$ random variable and let $S_i := \sum_{j=1}^{i} X_i$. By Lemma 1 we have $\mathrm{RW}_T \sim X \cdot (1/2, 1/2) + Y \cdot (1/2, -1/2)$, where X, Y are independent copies of S_T. Since $|\mathrm{RW}_T| = (X^2 + Y^2)^{1/2}/\sqrt{2}$, if $|\mathrm{RW}_T| > k$ then in particular one of $|X|$ and $|Y|$ is larger than k. Hence, we have

$$\Pr[|\mathrm{RW}_T| > k] \leq 2\Pr[|S_T| > k] = 2\Pr[|\textsc{Bin}(T, 1/2) - T/2| > k/2].$$

With a Chernoff bound we obtain

$$\Pr[|\mathrm{RW}_T| > k] \leq 4\mathrm{e}^{-\frac{k^2}{2T}}.$$

The second claim now follows from the reflection principle (see [14]) which states that

$$\Pr[\tau \geq T] \leq 2\Pr[|\mathrm{RW}_T| > k].$$

A.2 Sampling Binomial Random Variables

In this paper we assume that a binomial random variable $\textsc{Bin}(T, \frac{1}{2})$ can be sampled in constant time. Together with Lemma 1 this allows to sample the outcome of a random walk RW_T in constant time. In this section we discuss whether this assumption is justified.

The trivial algorithm to sample $\textsc{Bin}(T, \frac{1}{2})$ performs T coin flips, which takes time $\Theta(T)$. The literature on sampling contains a large amount of algorithms that sample binomial random variables much faster, namely in constant time (see, e.g., [5]), and implementations of these algorithms are readily available. However, all these algorithms are exact only in the Real RAM model of computation, where a memory cell can store a real number and operations on real numbers take constant time - on real-life computers these algorithms are not exact.

A more realistic machine model is the Word RAM, where each cell can store a w-bit integer, called *word*, and typical operations (arithmetic operations, bit operations, ...) on words take constant time. For sampling it makes sense to assume that one also can generate a random word in constant time. Usually

one assumes $w \geq \Omega(\log T)$. Until recently there was no algorithm known that samples $\textsc{Bin}(T, \frac{1}{2})$ on the word RAM faster than the trivial coin flipping. The first algorithm with sublinear preprocessing and polylogarithmic sampling time was proposed in [8]. We improve upon this algorithm here and show that constant expected sampling time is possible, even without any preprocessing. We remark that this is in line with a similar result for geometric random variables [4].

Theorem 5. *On the Word RAM, a binomial random variable* $\textsc{Bin}(T, \frac{1}{2})$ *can be sampled in expected time* $O(1)$*. Moreover, it can be sampled in a runtime that is larger than* $t > 0$ *with probability* $\exp(-t^{\Omega(1)})$*.*

In the remainder of this section we prove the above theorem.

We may assume that T is even, say $T = 2n$, otherwise we split into $\textsc{Bin}(T-1, \frac{1}{2}) + \textsc{Bin}(1, \frac{1}{2})$ and sample the latter in constant time. We may also assume $n \geq 2$, otherwise we sample $\textsc{Bin}(2n, \frac{1}{2})$ as a sum of $2n$ random bits in constant time.

Upper bound for binomial coefficients. Let $m \in \mathbb{N}$ be an approximation of $\sqrt{2n}$ with $m \in [\sqrt{2n}, \sqrt{2n}+3]$. We obtain such an approximation by using any (rough) approximation of the function $\sqrt{x}$, yielding $m' \in [\sqrt{2n}-1, \sqrt{2n}+1]$, and considering $m := \lceil m' \rceil + 1$.

Lemma 9. *In the above situation we have for any* $k \in \mathbb{Z}$

$$\binom{2n}{n + k \cdot m} \leq 2^{2n} \frac{4}{2^{|k|} m}.$$

Proof. Since $\binom{2n}{n+i} = \binom{2n}{n-i}$ we may assume $k \geq 0$. For $0 \leq i \leq n$, standard calculations yield

$$\begin{aligned}\binom{2n}{n+i} \Big/ \binom{2n}{n} &= \prod_{j=1}^{i} \frac{n+1-j}{n+j} \leq \prod_{j=1}^{i} \left(1 - \frac{j}{n+1}\right) \\ &\leq \exp\left(-\sum_{j=1}^{i} \frac{j}{n+1}\right) = \exp\left(-\tfrac{i(i+1)}{2(n+1)}\right) \leq \exp\left(-\tfrac{i^2}{2n}\right).\end{aligned}$$

A well-known bound following from Stirling's approximation is

$$\binom{2n}{n} \leq \frac{2^{2n}}{\sqrt{\pi n}},$$

so that we get

$$\binom{2n}{n+i} \leq \frac{2^{2n}}{\sqrt{\pi n}} \exp\left(-\tfrac{i^2}{2n}\right).$$

For $i = k \cdot m$ and $m \geq \sqrt{2n}$ we have

$$\exp\left(-\tfrac{i^2}{2n}\right) \leq \exp(-k^2) \leq 2^{1-k}.$$

Finally, for $m \leq \sqrt{2n} + 3$ and $n \geq 2$ we have $\sqrt{\pi n} \geq m/2$, which yields the claim.

Partition $\mathbb{Z}$ into buckets of m consecutive numbers, $B_k := \{km, km+1, \ldots, km+m-1\}$ for $k \in \mathbb{Z}$. Let

$$f(i) := \frac{4}{2^{\max\{k,-k-1\}}m}$$

for any $i \in B_k$, $k \in \mathbb{Z}$. Moreover, set

$$p(i) := \binom{2n}{n+i}/2^{2n},$$

with $p(i) := 0$ for $|i| > n$. Note that we have $\Pr[\textsc{Bin}(2n, \frac{1}{2}) = n + i] = p(i)$.

Lemma 10. *We have*

1. $f(i) \geq p(i)$ *for all* $i \in \mathbb{Z}$ *and*
2. $\sum_{i\in\mathbb{Z}} f(i) = 16$.

Proof. The first statement follows from Lemma 9 and monotonicity of binomial coefficients: For $i \in B_k$ with $k \geq 0$ we have $i \geq km$ so that $\binom{2n}{n+i} \leq \binom{2n}{n+km}$, which yields $p(i) \leq \binom{2n}{n+km}/2^{2n}$. Together with Lemma 9 this proves $p(i) \leq f(i)$. For $k < 0$ we argue similarly using $i \leq (k+1)m \leq 0$.

For the second statement we use symmetry of f around 0 and $|B_k| = m$ to obtain

$$\sum_{i<0} f(i) = \sum_{i\geq 0} f(i) = \sum_{k\geq 0} |B_k| \frac{4}{2^k m} = 4 \sum_{k\geq 0} 2^{-k} = 8.$$

Note that the above Lemma implies that $\bar{f}(i) := f(i)/16$ gives a probability distribution.

Rejection sampling. We use rejection sampling with the function f to sample from $\textsc{Bin}(2n, \frac{1}{2})$ as follows. We first sample $i \in \mathbb{Z}$ with probability distribution $\bar{f}$ (where $\bar{f}(i) = f(i)/16$). Then we sample a Bernoulli random variate $\textsc{Ber}(p(i)/f(i))$ (which is 1 with probability $p(i)/f(i)$). If it turns out 1 then we return $n+i$ and are done. Otherwise we reject i, i.e., we throw away i and repeat the whole process.

1. Sample $i \in \mathbb{Z}$ with probability distribution $\bar{f}$.
2. With probability $p(i)/f(i)$: Return $n + i$.
3. Otherwise: Reject i and goto 1.

Let us first argue about correctness and runtime of this algorithm and then fill in the details of how to implement steps 1 and 2. Note that the probability of returning $n+i$ in a particular iteration of this algorithm is $\bar{f}(i) \cdot \frac{p(i)}{f(i)} = p(i)/16$. Thus, the probability of returning $n+i$ is proportional to $p(i) = \Pr[\textsc{Bin}(2n, \frac{1}{2}) = n + i]$, so that we have an exact sampling algorithm. Moreover, since $p(i)$ is a probability distribution we have $\sum_{i\in\mathbb{Z}} p(i)/16 = 1/16 = \Omega(1)$. Hence, in every iteration the stopping probability of the algorithm is constant, so that the expected number of iterations of this algorithm is constant, and the number of iterations has an exponential tail.

To implement step 1 of this algorithm, we first flip a random bit whether $i \geq 0$ or $i < 0$ (making use of the fact $\sum_{i<0} f(i) = \sum_{i>0} f(i)$). Without loss of generality let $i \geq 0$ so that $f(i) = \frac{4}{2^k m}$ for $i \in B_k$, $k \geq 0$. The block B_k containing i is distributed geometrically, so we can draw random bits $X_1, X_2, \ldots$ and let $k \geq 0$ maximal with $X_1 = \ldots = X_k = 1$. Finally, we pick i uniformly at random in block B_k. This runs in expected constant time (with exponential tail).

To implement step 2, we have to sample $\textsc{Ber}(p(i)/f(i))$ in expected constant time. In general, to sample a Bernoulli random variate $\textsc{Ber}(p)$ it suffices to be able to compute an additive 2^{-L}-approximation of p in time $L^{O(1)}$ (see, e.g., [4]). Indeed, we can draw $\textsc{Ber}(p)$ by taking a uniformly random number $R \in [0,1)$ and returning whether $R \leq p$. We perform this check by computing 2^{-L}-approximations $\tilde{p}$ of p and $\tilde{R}$ of R (by taking its first L random bits) and checking whether these approximations allow to decide whether $R \leq p$ (which is possible if $|\tilde{R} - \tilde{p}| \geq 2^{1-L}$). If the decision is not possible, we increase L by 1 and repeat. Since R is uniformly random in $[0,1)$, this procedure stops after an expected constant number of iterations (with exponential tail). With the L-th iteration taking time $L^{O(1)}$, the total runtime is larger than t with probability $\exp(-t^{\Omega(1)})$, as promised.

Note that since $0 \leq p(i)/f(i) \leq 1$ it suffices to compute a *relative* $2^{-\Omega(L)}$-approximation of $p(i)/f(i)$ in time $L^{O(1)}$.

Approximate Calculations on the Word RAM. On the Word RAM we can perform usual logical and arithmetic operations on two w-bit integers in constant time. We can compute with longer integers by representing them as lists of words. This allows, e.g., to add two L-bit integers in time $O(1 + L/w)$. In general, a usual logical or arithmetic operation on two L-bit integers can be performed in time $O(1 + (L/w)^{O(1)})$. Moreover, we can use floating-point approximations of reals by storing both mantissa and exponent as long integers. This allows to perform typical operations on two floating-point numbers with L-bit mantissas and E-bit exponents in time $O(1 + ((L+E)/w)^{O(1)})$. For the numbers that we will consider, the exponents are $O(L + \log n)$-bits numbers. If $L \leq \log(n)$ the usual Word RAM assumption of $w \geq \Omega(\log n)$ implies that we can compute with these exponents in constant time. In any case, the runtime for handling the exponents is bounded by $L^{O(1)}$. For this reason, we will only discuss the mantissa in the following.

Observe that once we have a floating-point approximation of $p(i)$ with precision $2^{-\Theta(L)}$, we easily obtain a floating point approximation of $p(i)/f(i)$ with precision $2^{-\Theta(L)}$, since $f(i)$ is build of elementary functions. Moreover, to approximate $p(i) = \binom{2n}{n+i}/2^{2n}$ it suffices to be able to approximate factorials.

Hence, we are left with the following problem. Given n and L, compute a floating-point approximation of $n!$ with precision 2^{-L} (i.e., with relative error 2^{-L}). Note that the standard Stirling's approximation only allows to approximate $n!$ with fixed precision (depending on n). Classic formulas that allow arbitrary precision approximations of $n!$ are Lanczos approximation [12] and Spouge's approximation [17]. A fixed precision version of the former is, for in-

stance, implemented in the GNU Scientific Library[8]. These classic formulas allow to approximate $n!$ up to precision 2^{-L} in time $L^{O(1)}$. The only possible obstacle for us using these approximations is that they are typically analyzed on the Real RAM model of computation, where we can compute with real numbers in constant time and do not have to worry about floating-point approximations. In the following we go through Spouge's approximation to check that it indeed works on the Word RAM.

Spouge's approximation [17] states that for any $n, L \in \mathbb{N}$, $L > 2$,

$$n! \approx (n+L)^{n+1/2}\mathrm{e}^{-(n+L)}\left[c_0 + \sum_{k=1}^{L-1}\frac{c_k}{n+k}\right],$$

where

$$c_0 = \sqrt{2\pi}, \quad c_k = \frac{(-1)^{k-1}}{(k-1)!}(L-k)^{k-1/2}\mathrm{e}^{L-k},$$

with a relative error that is bounded by

$$L^{-1/2}(2\pi)^{-(L+1/2)} \leq 2^{-L-1}.$$

To evaluate this formula, it suffices to be able to compute (floating-point approximations of) π and the functions e^x and $\sqrt{x}$. The standard algorithms for this also work on the Word RAM.

It remains to show that it suffices to compute all terms of the formula with precision $2^{-\Theta(L)}$ in order to obtain $n!$ with precision $2^{-L} = 2^{-L-1} + 2^{-L-1}$ (the error from Spouge's approximation plus the error of floating-point approximation). Note that this requires an argument, since the terms $c_k/(n+k)$ change in sign and could cancel, making very high precision necessary. However, it is not hard to bound

$$\left|\frac{c_k}{n+k}\right| \leq 2^{\ell_1}$$

with $\ell_1 = O(L)$. Moreover, Spouge's approximation guarantee and $n! = \Theta((n/\mathrm{e})^{n+1/2})$ (by Stirling's approximation) imply

$$c_0 + \sum_{k=1}^{L-1}\frac{c_k}{n+k} \geq (1-2^{-L-1})\frac{n!}{(n+L)^{n+1/2}\mathrm{e}^{-(n+L)}} \geq 2^{-\ell_2},$$

with $\ell_2 = O(L)$. Together, these inequalities show that $\ell_1 + \ell_2 + C \cdot L = O(L)$ bits of precision for a summand $\frac{c_k}{n+k}$ (or c_0) yield a precision of $C \cdot L$ bits *relative to the sum* $c_0 + \sum_{k=1}^{L-1}\frac{c_k}{n+k}$. For C a sufficiently large constant, this precision suffices for the sum to have a precision of $L+1$ bits so that we compute a relative 2^{-L-1}-approximation of $n!$, as desired. Since there are $L^{O(1)}$ terms in Spouge's approximation, we obtain a total runtime of $L^{O(1)}$ to compute $n!$ with precision 2^{-L}. This finishes the proof of Theorem 5.

[8] http://www.gnu.org/software/gsl/

Remark. We remark that Theorem 5 implies that m samples from $\textsc{Bin}(T, \frac{1}{2})$ take total time $O(m)$, both in expectation and with high probability in m (we will show the latter in Lemma 11). Hence, the runtime bound of our main theorem is valid in expectation and with high probability using the above sampling algorithm for $\textsc{Bin}(T, \frac{1}{2})$. However, for the sake of readability we prefer to assume that we can sample $\textsc{Bin}(T, \frac{1}{2})$ in constant time in the remainder of this paper.

B Proofs and Lemmas omitted from Section 3

B.1 Data Structure

Jump procedures need access to the IDLA structure $A = A(n)$ in some way. In this section we describe a data structure for storing A that fits the needs of our jump procedures.

The most natural solution for storing the shape of A is a $2n \times 2n$ matrix in which each element contains the information whether the corresponding position is occupied. The size of this matrix is $\Theta(n^2)$ and already its initialization would therefore exceed our desired running time. As we have $r_O - r_I = O(\log n)$ with high probability, an $O(\sqrt{n}) \times O(\sqrt{n})$ matrix is sufficient (in most simulations). However, in this case A contains the whole center $B_0(r_I)$, so most of the information stored in the matrix is still redundant.

To save space we split the grid in slices $S_0, S_1, \ldots$, where slice S_i contains all $z \in \mathbb{Z}^2$ with $i \leq |z| < i + 1$. We store r_I and r_O, as well as $A \cap S_i$ for all $r_I \leq i < r_O$. This is sufficient information to reconstruct A, as $A \cap S_i = S_i$ for $i < r_I$ and $A \cap S_i = \emptyset$ for $i \geq r_O$.

We would like to store each set $A \cap S_i$ by a bit array, however, the natural index set S_i is not a range of integers. As a workaround, we first construct a perfect hash function $h_i \colon S_i \to [n_i]$ to map S_i to a range of integers of length $n_i = O(|S_i|) = O(i)$ without collision. See, e.g., [3] for a construction of a perfect hash function with $O(n)$ construction time, $O(n)$ space usage and $O(1)$ evaluation time[9]. Additionally, we store a bit array $B_i[1..n_i]$ with $B_i[h_i(z)] = [z \in A]$, i.e., for each $z \in S_i$ the bit $B_i[h_i(z)]$ stores whether z is in A. In total this yields $O(1)$ time to access/modify $[z \in A]$ with space requirement $O(i)$. Note the similarities to storing a bit array for $A \cap S_i$, only that this is not directly possible.

To allow for efficient updates of our data structure after placing a particle, we also store $|S_i|$ and $|A \cap S_i|$ for each $r_I \leq i < r_O$. Then whenever a new grid point z is occupied (in slice S_i), we do the following. If slice S_i is not yet built, i.e., if r_O was at most i, then we increase r_O and initialize the data structure for $A \cap S_i = \{z\}$. Specifically, we enumerate S_i, compute $|S_i|$ and $|A \cap S_i| = |\{z\}| = 1$, build the perfect hash function h_i and initialize the bit array B_i. Otherwise, if the slice S_i was already built, then we simply update $B_i[h_i(z)] := 1$ and $|A \cap S_i|$. Whenever the slice S_{r_I} becomes full (i.e., $|A \cap S_{r_I}| = |S_{r_I}|$) we trash everything

[9] We remark that it is an easy exercise to explicitly construct a perfect hash function in our situation, simply based on the angles of the grid points inside a slice.

we stored for this slice $(h_{r_I}, B_{r_I}, |S_{r_I}|, |A \cap S_{r_I}|)$ and increase r_I. We repeat this as long as S_{r_I} is full. This way we satisfy an invariant of S_{r_I} being the innermost non-full slice and S_{r_O} being the lowest empty slice. In particular, r_I and r_O are always the correct in- and out-radius.

It is not hard to see that, when starting from the empty set, the updates of our data structure take time $O(r_O^2)$ in total, and that it uses $O(r_O(r_O - r_I))$ bits of space. Since $r_O - r_I$ (and, thus, also r_O) is bounded both in expectation and with high probability, we get Lemma 3.

B.2 Proof of Theorem 4

Theorem 4 is a consequence of Theorem 2, Lemma 4 and Lemma 3.

Proof (Theorem 4). We simulate the first $\sqrt{n}$ particles in the naive step-by-step way in expected time $O(n)$. For all the remaining particles we use the jump procedure. Since Theorem 2 together with $r_O \leq n$ implies $\mathbb{E}[(r_O - r_I)^2] \leq O(\log^2 n)$, we can read off Lemma 4 an expected number of jumps of $O(\log n(\bar{\delta}_J^2 + \log n))$, yielding an expected time of $O(t_J \log n(\bar{\delta}_J^2 + \log n))$ to simulate the i-th particle. Adding the runtime of our data structure (Lemma 3), in total over all particles we obtain an expected time as claimed. Lemma 3 also directly implies the statement about space usage.

Arguing about the concentration of the runtime requires some more work. Note that $r_O - r_I = O(\log n)$ for all $\sqrt{n} < i \leq n$ with high probability. If this is the case then the number of jumps needed for particle $\sqrt{n} < i \leq n$ is more than $k \log n(\bar{\delta}_J^2 + \log n)$ with probability $\exp(-\Omega(k))$ by Lemma 4. By the following lemma, the total number of jumps for all these particles is $O(n \log n(\bar{\delta}_J^2 + \log n))$ with high probability. Moreover, the runtime of the jump procedure is bounded by $O(t_J)$ with high probability, therefore it is also bounded by $O(t_J)$ for all $O(n \log n(\bar{\delta}_J^2 + \log n))$ jumps with high probability (adjusting the constant in the exponent of the high probability). In total, the claimed runtime bound $O(t_J \log n(\bar{\delta}_J^2 + \log n))$ also holds with high probability for the particles $\sqrt{n} < i \leq n$. For the naive simulation of the first $\sqrt{n}$ particles we are in a similar situation. With high probability $A(i)$ is contained in $B_0(O(n^{1/4}))$ for all $1 \leq i \leq \sqrt{n}$ (Theorem 2). If this is the case, then the simulation of one of the first $\sqrt{n}$ particles takes time larger than $k\sqrt{n}$ with probability $\exp(-\Omega(k))$ for any $k \geq 1$, since after each block of $\Theta(\sqrt{n})$ steps we leave the ball with radius $\Theta(n^{1/4})$ with constant probability (Lemma 2). Again using the following lemma, we obtain that this naive simulation takes time $O(n)$ in total with high probability, finishing the proof.

Lemma 11. *Let $X_1, \ldots, X_m$ be independent positive random variables with $\Pr[X_i > kv] \leq \exp(-\Omega(k^\varepsilon))$ for some $v > 0$ and $\varepsilon > 0$ and all $k \geq 1$. Then we have*

$$\sum_{i=1}^{m} X_i \leq O(mv)$$

with probability at least $1 - \exp(-m^{\Omega(1)})$.

Proof. Without loss of generality we can assume $v = 1$. Assume that all X_i are at most $M > 0$ to be fixed later. By a union bound this happens with probability at least $1 - m\exp(-\Omega(M^\varepsilon))$. This assumption means that we instead consider bounded random variables Y_i with $\Pr[Y_i \leq k] = \Pr[X_i \leq k]/\Pr[X_i \leq M]$ for any $k \leq M$. Using the Azuma-Hoeffding bound (see for example Chapter 5 in [7]) we obtain

$$\Pr\Big[\sum_{i=1}^{m}(Y_i - \mathbb{E}[Y_i]) > t\Big] \leq \exp\left(\frac{-t^2}{2mM^2}\right).$$

Hence, for sufficiently large constant $c > 0$ we have

$$\Pr\Big[\sum_{i=1}^{m} Y_i > cm\Big] \leq \exp(-\Omega(m/M^2)),$$

and in total we get

$$\Pr\Big[\sum_{i=1}^{m} X_i > cm\Big] \leq \exp(-\Omega(m/M^2)) + m\exp(-\Omega(M^\varepsilon)).$$

Choosing $M = m^{1/(2+\varepsilon)}$ this error is $\exp(-\Omega(m^{\varepsilon/(2+\varepsilon)})) = \exp(-m^{\Omega(1)})$ as claimed.

C Proofs and Lemmas omitted from Section 4

Proof (Lemma 6).

Because of equation (7) and $\sum_w P_J(w) = \sum_w P_{Alg1}(w) = 1$ we have $\sum_w P_{rest}(w) = 1$. However, we have to prove that we can choose p_{fail} (and c_J) in such a way that $P_{rest}(w)$ is non-negative for all $w \in \bar{S}$. It suffices to choose p_{fail} such that for all $w \in \bar{S}$

$$(1 - p_{fail})P_{Alg1}(w) \leq P_J(w),$$

since then $P_{rest}(w) \geq 0$ according to equation (7). Without loss of generality we consider $w \in \frac{1}{2}S$ with $w \equiv_T z$, as otherwise we have $P_{Alg1}(w) = 0$.

We abbreviate $P_{RW}(w) := \Pr[\mathrm{RW}_T(z) = w]$. Since Algorithm 1 returns the endpoint of a random walk of length T conditioned on being contained in $\frac{1}{2}S$, we clearly have

$$P_{Alg1}(w) = \frac{P_{RW}(w)}{1 - \Pr[\mathrm{RW}_T(z) \notin \frac{1}{2}S]}. \tag{9}$$

Since $\Pr[\mathrm{RW}_T(z) \notin \frac{1}{2}S] = \Pr[|\mathrm{RW}_T| > \frac{1}{2}\sigma]$ and $T \leq \frac{\sigma^2}{c_J \log(n/\mathrm{e})}$ we can apply Lemma 8 to get

$$P_{Alg1}(w) \leq \frac{P_{RW}(w)}{1 - 4(n/\mathrm{e})^{-c_J/8}}. \tag{10}$$

For $P_J(w)$, any walk of length T starting in z and ending in w is counted, except if it hits ∂S. Hence, we have

$$P_J(w) \geq P_{RW}(w) - \Pr[\tau_{\partial S} \leq T].$$

We want to write the right hand side of this as $P_{RW}(w)\cdot(1-\rho)$, so that combined with equation (10) we obtain $P_J(w) \geq P_{Alg1}(w) \cdot (1-\rho')$. For this we need an upper bound for $\Pr[\tau_{\partial S} \leq T]$ (provided by Lemma 8) and a lower bound for $P_{RW}(w)$ (provided by Lemma 7). Combining the two yields (after some technical simplifications that we postpone)

$$\frac{\Pr[\tau_{\partial S} \leq T]}{P_{RW}(w)} \leq 24\mathrm{e}^{c_J/2} n^{1-5c_J/16}. \tag{11}$$

Thus,

$$P_J(w) \geq P_{RW}(w)(1 - 24\mathrm{e}^{c_J/2} n^{1-5c_J/16}),$$

and together with equation (10) we obtain

$$P_J(w) \geq (1 - 24\mathrm{e}^{c_J/2} n^{1-5c_J/16} - 4(n/\mathrm{e})^{-c_J/8}) P_{Alg1}(w).$$

Thus, we can safely set $p_{fail} \geq 28\mathrm{e}^{c_J/2} n^{-\min\{c_J/8, 5c_J/16-1\}}$.

In the following we show inequality (11). First note that by the definition of T and $\sigma \geq c_J \log n$ we have

$$T \geq \frac{\sigma^2}{c_J \log(n/\mathrm{e})} - 1 \geq \frac{\sigma^2}{c_J \log n} \cdot \left(1 + \frac{1}{\log n}\right) - 1 \geq \frac{\sigma^2}{c_J \log n} + c_J - 1 \geq \frac{\sigma^2}{c_J \log n},$$

for $c_J \geq 1$. Thus, $|w| \leq \frac{1}{2}\sigma$, $|w|/T \leq \frac{1}{2}$ and $\frac{\sigma^2}{T} \leq c_J \log n$. Using this, we can combine Lemmas 8 and 7 to obtain

$$\begin{aligned}\frac{\Pr[\tau_{\partial S} \leq T]}{P_{RW}(w)} \leq \frac{8(n/\mathrm{e})^{-c_J/2}}{\frac{1}{3T}\exp\big(-\frac{|w|^2}{2T}\big(1+\frac{|w|}{T}\big)\big)} &\leq 24T(n/\mathrm{e})^{-c_J/2} \exp\left(\frac{3\sigma^2}{16T}\right) \\ &\leq 24\mathrm{e}^{c_J/2} T n^{-5c_J/16}.\end{aligned}$$

Note that as S is completely filled with particles we have $n \geq \sigma^2 \geq T$, finally yielding

$$\frac{\Pr[\tau_{\partial S} \leq T]}{P_{RW}(w)} \leq 24\mathrm{e}^{c_J/2} n^{1-5c_J/16}.$$