# The Dispersion Time of Random Walks on Finite Graphs

Nicolás Rivera[*]
nicolas.rivera@cl.cam.ac.uk
University of Cambridge
Cambridge, United Kingdom

Thomas Sauerwald[*]
thomas.sauerwald@cl.cam.ac.uk
University of Cambridge
Cambridge, United Kingdom

Alexandre Stauffer[†]
a.stauffer@bath.ac.uk
University of Bath
Bath, United Kingdom

John Sylvester[*]
john.sylvester@cl.cam.ac.uk
University of Cambridge
Cambridge, United Kingdom

## ABSTRACT

We study two random processes on an $n$-vertex graph inspired by the internal diffusion limited aggregation (IDLA) model. These processes can also be regarded as protocols for allocating jobs in a distributed network of servers. In both processes $n$ particles start from an arbitrary but fixed origin. Each particle performs a simple random walk until it first encounters an unoccupied vertex, at which point the vertex becomes occupied and the random walk terminates. In one of the processes, called *Sequential-IDLA*, a single particle moves until settling and only then does the next particle start whereas in the second process, called *Parallel-IDLA*, all unsettled particles move simultaneously. The second process is akin to running the first in parallel. Our main goal is to analyze the so-called dispersion time of these processes, which is the maximum number of steps performed by any of the $n$ particles.

In order to compare the two processes, we develop a coupling which shows the dispersion time of the Parallel-IDLA stochastically dominates that of the Sequential-IDLA; however, the total number of steps performed by all particles has the same distribution in both processes. This coupling also gives us that dispersion time of Parallel-IDLA is bounded in expectation by dispersion time of the Sequential-IDLA up to a multiplicative $\log n$ factor. Moreover, we derive asymptotic upper and lower bound on the dispersion time for several graph classes, such as cliques, cycles, binary trees, $d$-dimensional grids, hypercubes and expanders. Most of our bounds are tight up to a multiplicative constant.

## CCS CONCEPTS

• **Theory of computation → Random walks and Markov chains**.

## KEYWORDS

Random Walks on Graphs; Parallelisation of random processes; Interacting particle systems

**ACM Reference Format:**
Nicolás Rivera, Thomas Sauerwald, Alexandre Stauffer, and John Sylvester. 2019. The Dispersion Time of Random Walks on Finite Graphs. In *31st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '19), June 22–24, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3323165.3323204

[*]Supported by T.S.' ERC Starting Grant 679660 (DYNAMIC MARCH).
[†]Supported by the EPSRC Early Career Fellowship EP/N004566/1.

## 1 INTRODUCTION

The internal diffusion limited aggregation (IDLA) model, first introduced independently by Diaconis & Fulton [19] and Meakin & Deutch [36], is a protocol for recursively building a randomly growing subset (aggregate) of vertices of a graph. Initially, the aggregate consists of only one vertex, denoted as the *origin*, and we let a particle be settled at that vertex. Then, at each step, we start a new particle from the origin and let it perform a random walk until it visits a vertex not contained in the aggregate. At this point, we say that the new particle settles at that vertex, and the vertex is added to the aggregate. We then add a new particle at the origin, and iterate this procedure over and over again.

IDLA was introduced on the infinite lattice $\mathbb{Z}^d$. Here we consider a finite connected $n$-vertex graph $G$. Note that after $n$ particles have settled, the aggregate occupies the whole of $G$. During this time, each particle performed some number of random walk steps before it settles. Clearly, this number depends on the geometry of the aggregate when the particle started moving. We define the *dispersion time* as the largest number of random walk steps performed by any one of the $n$ particles before reaching an unoccupied vertex.

We refer to the above protocol as *Sequential-IDLA*, in allusion to the fact that a particle cannot begin to move until the one before it settles. However, alternative scheduling protocols could be defined, in the sense that we could choose to add and move a new particle from the origin before the previous one has settled. In this way, there could be several unsettled particles moving at the same time, but they must abide by the rule that whenever a particle jumps to an unoccupied vertex, it must settle there. We call any process of this sort a *dispersion process*. We are interested in understanding the affect of different scheduling protocols on the dispersion time. For this, we will consider the following protocol. Start all $n$ particles from the origin at time 0 (thus one of them will instantaneously settle at the origin). Then, all particles perform one random walk step simultaneously; if one or more particles jump to an unoccupied vertex, then one such particle settles there. Iterate this procedure until all particles have settled. We call this second process *Parallel-IDLA*.

Both dispersion processes can be regarded as a set of simple local protocols for resource allocation. Specifically, the sequential dispersion process is quite similar to a local-search based reallocation scheme from [14], where a job continues to reallocate itself to a neighbour with less load until it has found a local minimum. Furthermore, the parallel dispersion process is related to the "QoS

Load Balancing" model [1], a particular instance of selfish load balancing (see also [10, 11] for similar protocols). In the QoS model, tasks perform random walks in parallel and terminate only if they have found a resource on which the estimated processing time is acceptable according to some agent-specific threshold. Our dispersion processes can be also viewed as a spatial coordination game, where the goal is to achieve a state in which players are all making *distinct* choices. As mentioned in [5], such games serve as a model for the dynamics in location games or habitat selection of species.

Recall that the dispersion time is the maximum number of steps taken by any of the $n$ particles in either IDLA process. For the complete graph $K_n$ the Sequential-IDLA process has essentially the same dynamics as the famous coupon collector process and the dispersion time corresponds to the longest wait between collecting successive (new) coupons. Thus the discrepancy between the Sequential and Parallel dispersion times for $K_n$ measures the effect of parallelising the coupon collector process on the longest time between coupons. This motivates the study of dispersion time on different networks which we can view as a generalization of the coupon collector process. In the general setting we address the question: what is the cost of parallelising the IDLA process? Addressing this question requires us to determine or at least estimate the Parallel *and* Sequential dispersion times.

The total time taken by all walks, as opposed to the longest walk, is also natural to study for these models. Returning briefly to the complete graph we see that the sum of the walk lengths in the Sequential-IDLA corresponds to the time to collect all coupons - this is what is typically studied for the coupon collector. Our couplings show that for any fixed graph the sum of all walk lengths, later denoted by $W$, is the same for Parallel and Sequential IDLA. From one perspective this motivates the study of $W$ for general graphs, this is work in progress by the authors. However, in this paper we are more interested in the discrepancies between the Sequential and Parallel processes, some of which are captured by the dispersion time.

Since in IDLA particles perform random walks, both dispersion processes can be regarded as a protocol for exploring and covering an unknown network. However, as opposed to previously studied models of covering a graph with multiple random walks [4, 9, 17], the length of the particles' trajectories may vary wildly in the dispersion process. This introduces strong correlations between different particles, a challenge which is not present in the cover time of multiple random walks.

## 1.1 Our Contributions

The results we prove can be broadly placed into three categories: Coupling results which allow us to make qualitative statements such as "is sequential dispersion *faster* than parallel dispersion". Secondly general bounds which give us quantitative estimates on how slow or fast the process can be on general graphs or graph classes such as trees or regular graphs. Finally we have also calculated the dispersion up to constant factors for many well known graph families such as cliques, cycles, binary trees, $d$-dimensional grids, hypercubes and expanders. Knowing the dispersion time for several different graph topologies aids our understanding of the dispersion time and how it is related to different graph parameters. We shall briefly outline our results in this section before presenting them in full in the following sections, complete proofs of all results can be found in the full length version of this paper on ArXiv [40].

The first fundamental question is whether one can relate the two dispersion times. We answer this question by developing a coupling, based on "cutting & pasting" particle trajectories. To state our results we must fix some notation, let $\tau_{seq}^{v}(G)$ and $\tau_{par}^{v}(G)$ denote the dispersion time of Sequential-IDLA and Parallel-IDLA on $G$ with origin $v$, respectively. Unless otherwise specified $G$ will be connected, simple $n$ vertex graph and $v \in V(G)$ arbitrary. We use our coupling to prove the following results:

- $\tau_{seq}^{v}(G) \leq \tau_{par}^{v}(G)$,
- $\mathbf{E}[\,\tau_{seq}^{v}(G)\,] \leq \mathbf{E}[\,\tau_{par}^{v}(G)\,]$,
- $\mathbf{E}[\,\tau_{par}^{v}(G)\,] = O\Big(\log(n) \cdot \mathbf{E}[\,\tau_{seq}^{v}(G)\,]\Big)$,
- The total number of steps is equidistributed.

The intuition behind Parallel-IDLA being "slower" than Sequential-IDLA is that, due to competition between particles trying to settle concurrently, the lengths of particle trajectories in Parallel-IDLA vary more than in Sequential-IDLA.

In Section 4.3 we introduce a variant of the Parallel-IDLA where each particle has an exponential clock of rate 1 and moves every time the clock rings until the particle settles. We name this variant the continuous-time Uniform-IDLA and let $\tau_{c-unif}^{v}$ denote its dispersion time. We also consider the Sequential and Parallel-IDLA with lazy walks and let $\tau_{L-seq}^{v}$ and $\tau_{L-par}^{v}$ denote their dispersion times respectively. Under some mild conditions we show that

- $\tau_{c-unif}^{v}(G) = (1 + o(1)) \cdot \tau_{par}^{v}(G)$,
- $\tau_{seq}^{v}(G) = (2 + o(1)) \cdot \tau_{L-seq}^{v}(G)$,
- $\tau_{par}^{v}(G) = (2 + o(1)) \cdot \tau_{L-par}^{v}(G)$,

hold in expectation and w.h.p., for further details, see Section 4. These relations are very useful as using lazy or continuous-time walks allow one to sidestep issues such as periodicity of the simple random walk or multiple particles settling simultaneously.

We define $t_{seq}$ and $t_{par}$ to be the worst-case expected Seq/Par-IDLA dispersion times over all possible starting vertices in $V$. Let $t_{hit}(G)$ denote the maximum expected hitting time of a random walk from $v$ to $w$ over all pairs of vertices $(v, w)$. This is a relatively simple and well studied quantity, and we show a basic but useful upper bound on the dispersion time in terms of $t_{hit}$:

- $\mathbf{Pr}\left[\,\sup_{v\in V} \tau_{par}^{v}(G) > 8 \cdot t_{hit}(G) \cdot \log_2(n)\,\right] \leq \frac{1}{n}$,
- $t_{par}(G) = O(t_{hit}(G) \cdot \log(n))\,.$

The same results also hold for $\tau_{seq}$ and $t_{seq}$. These results imply the following worst-case bounds:

- For any $n$-vertex graph, $t_{seq}, t_{par} = O\big(n^3 \log n\big)$.
- For any regular $n$-vertex graph, $t_{seq}, t_{par} = O\big(n^2 \log n\big)$.

The above bounds are attained by the lollipop and cycle respectively and are thus best possible up to constant factors. In fact for any fixed $r < \infty$ one can construct a family of $r$-regular graphs for which the second bound above is tight. For example when $r = 3$ one can iteratively augment an even cycle by adding an edge between two

**Table 1: The last two columns summarize our results, proofs of these results can be found in the full version of this paper [40]. The results in the first three columns are for comparison and can be found in standard textbooks for example [3, 30]. The constant $\kappa_{cc}$ above has an explicit formula given by (4) and it evaluates to roughly 1.255, to be contrasted with $\pi^2/6 \approx 1.644$. The constant $\kappa_p$ is non-explicit, though specified by Theorem 5.1, simulations run by Nikolaus Howe (student) suggest $\kappa_p \approx 0.6\ldots$.**

| Graph family name | Cover time $t_{cov}$ | Hitting time $t_{hit}$ | Mixing time $t_{mix}$ | Dispersion time $t_{seq}$ | $t_{par}$ |
|---|---|---|---|---|---|
| Path | $n^2$ | $n^2$ | $O(n^2)$ | $\kappa_p \cdot n^2 \log n$ | |
| Cycle | $n^2/2$ | $n^2/2$ | $O(n^2)$ | $\Theta(n^2 \log n)$ | $\Theta(n^2 \log n)$ |
| 2-dimensional grid | $\Theta(n \log^2 n)$ | $\Theta(n \log n)$ | $\Theta(n)$ | $\Omega(n \log n)$ | $O(n \log(n)^2)$ |
| d-dimensional grid, $d > 2$ | $\Theta(n \log n)$ | $\Theta(n)$ | $\Theta(n^{2/d})$ | $\Theta(n)$ | $\Theta(n)$ |
| Hypercube | $\Theta(n \log n)$ | $\Theta(n)$ | $\log n \log\log n$ | $\Theta(n)$ | $\Theta(n)$ |
| Binary tree | $\Theta(n \log n)$ | $\Theta(n \log n)$ | $n$ | $\Theta(n \log(n)^2)$ | $\Theta(n \log(n)^2)$ |
| Complete graph | $\Theta(n \log n)$ | $\Theta(n)$ | 1 | $\kappa_{cc} \cdot n$ | $(\pi^2/6) \cdot n$ |
| Expanders | $\Theta(n \log n)$ | $\Theta(n)$ | $O(\log n)$ | $\Theta(n)$ | $\Theta(n)$ |
| Lollipop | $\Theta(n^3)$ | $\Theta(n^3)$ | $\Theta(n^2)$ | $\Theta(n^3 \log n)$ | $\Theta(n^3 \log n)$ |

vertices of degree two who are at distance two to obtain a 3-regular graph with the same asymptotic dispersion time as the cycle.

Let $t_{hit}(\pi, S)$ denote the expected hitting time of $S \subseteq V$ by a random walk from stationarity. We also prove bounds in terms of hitting times of subsets of decreasing sizes.

- $t_{par}(G) \leq 60 \cdot \sum_{j=1}^{\lceil \log_2 n \rceil} \left( t_{mix} + \max_{S \subseteq V:\, |S| \geq 2^{j-2}} t_{hit}(\pi, S) \right),$
- $t_{seq}(G) \leq 30 \cdot \max_{j \leq \lceil \log_2 n \rceil} \left\{ j \cdot \left( t_{mix} + \max_{S \subseteq V:\, |S| \geq 2^{j-2}} t_{hit}(\pi, S) \right) \right\}.$

These two bounds refine the simple $O(t_{hit} \cdot \log n)$ bound and for graphs with suitably good expansion such as the Hypercube, see Table 1, these results give us a bound of $O(t_{hit})$.

Based on the intuition that the last walk in the Sequential-IDLA should have a hard target to hit, one would expect that the worst-case hitting time $t_{hit}$ provides at least an approximate lower bound on the dispersion time. This intuition turns out to be false in general, as evidenced by a certain class of bounded-degree trees (see Proposition 3.7) which exhibit a gap of almost $\sqrt{n}$ between $t_{hit}$ and $t_{seq}$. We prove the following lower bounds

- If $G$ has maximum degree $\Delta$, then $t_{seq}(G) = \Omega(|E|/\Delta)$.
- For any tree $T$, we have $t_{seq}(T) = \Omega(n)$.
- If $G$ is regular, $t_{seq}(G) = \Omega(t_{mix}) = \Omega\left(\frac{1}{1-\lambda_2}\right) = \Omega\left(\frac{1}{\Phi}\right),$

where $t_{mix}$ is the $\ell_1$ mixing time, $\Phi$ is the conductance and $\lambda_2$ is the second largest eigenvalue associated with the lazy random walk.

As mentioned before we calculate dispersion time for many well known graph families, the two rightmost columns of Table 1 show our results. As seen in Table 1 we can determine the expected dispersion time in Parallel and Sequential-IDLA up to multiplicative constant factors in all graphs listed apart from the 2-dimensional grid, where there is a discrepancy of order $\log n$ between the lower and upper bounds. This remains an interesting open problem which seems to require quite detailed knowledge of the shape of the aggregate on a *finite* box/tori. As discussed in Section 1.3 below, this is a non-trivial problem even in the *infinite* 2d-grid.

## 1.2 Techniques used

The first tool we invent to analyse these processes is the Cut & Paste bijection between the histories of IDLA processes. The bijection allows us to couple the dispersion time of the Parallel-IDLA with that of the Sequential-IDLA and other variants such as Uniform IDLA (where at each step a random unsettled particle moves), as well as IDLA processes with lazy or continuous-time walks. Bounding dispersion times via these other variants is useful for avoiding issues such as simultaneous arrivals at unoccupied vertices and allows us to apply mixing time bounds. At a base level the stochastic domination of $\tau^v_{seq}$ by $\tau^v_{par}$ means we can sandwich both quantities with a bound on $\tau^v_{par}$ from above and on $\tau^v_{seq}$ from below.

Although these two processes are closely related, the different sources of dependence arising from the contrasting scheduling protocols provide several challenges. In the Sequential-IDLA interaction between the walkers comes via the configuration of vertices settled by the previous walks. This can make proving a tight lower bound on $\tau^v_{seq}$ tricky and often some knowledge of the geometry of the aggregate after a certain time is helpful. What is needed are results reminiscent of the "shape theorems" discussed in Section 1.3 below. This requirement for detailed knowledge of the aggregate appears to be crucial in achieving a tight lower bound on $\tau^v_{seq}$ for the binary tree and 2-dimensional grid. In comparison with the Sequential-IDLA interactions are less passive in the Parallel process as particles jostle to be the first to settle a vertex. This interaction can increase the length of the longest walk as is witnessed by the Cut & Paste bijection.

## 1.3 Related work

As pointed out by Diaconis & Fulton [19], there are several mathematical reasons for studying IDLA, including using it to take a product of sets - a special case of the "smash product". The limit shape of the aggregate on $\mathbb{Z}^d$ was first studied by Lawler, Bramson and Griffeath [29] who showed that, after adding $n$ particles and properly rescaling the aggregate by $n^{1/d}$, in the limit as $n \to \infty$ this converges to an Euclidean ball. There has been a series of improvements to this "shape theorem" of [29], by bounding the rate of convergence to the euclidean ball. The first refinement was made by Lawler [28] and the state of the art was achieved recently by two independent groups of authors [6–8, 26, 27].

Several authors have also proved shape theorems on other infinite graphs and groups including combs, tree, non-amenable groups and Bernoulli percolation on $\mathbb{Z}^d$ [13, 21, 24, 25, 41]. In all of these cases the limit shape is always a ball with respect to the underlying graph metric. Limit shapes in $\mathbb{Z}^d$ for other variants of IDLA have also been established. These variations include using non-standard random walks such as for drifted [35] and cookie walks [39] or starting the walks from different positions [20]. The time for the process started with some initial aggregate to "forget" this starting state has also been studied [33].

One model where interaction between particles prevents settling at a site is a two-type particle system called "Oil and Water" where particles of opposite types displace each other [16]. There have been some papers on models related to the Parking function of a graph where cars drive randomly around a graph searching for vacant spots [18, 23]. More commonly, however, interaction is directly between particles and not with the host graph such as predator prey/coalescing models [17]. The problem of uniformly distributing $n$ non-communicating memoryless particles across $n$ unoccupied sites is also considered from a game theoretic perspective [5].

Other models related to IDLA include rotor-router aggregation, chip firing, Abelian sandpile and activated random walks [12, 31, 42]. The rotor-router walk, or Propp machine, is the process which drives rotor-routor aggregation. This is a deterministic version of the simple random walk and has been used in load balancing [2].

Many interacting particle systems, such as the abelian sandpile model, satisfy a so-called "least action principle" which is key to their analysis [32]. Such a principle roughly states that the natural behavior of the system is in a sense optimal and, if the process is perturbed, then the outcome will have a higher energy. One may try to find a least action principle for Sequential-IDLA by conjecturing that if we allow a random walk not to settle sometimes when visiting an unoccupied vertex (thereby performing more random walk steps), then this can only increase the dispersion time. We show that this is not the case with the following counterexample.

Let $\xi_x^i = 1$ iff the site $x$ is vacant after $i-1$ walkers have settled and $W(X)$ denote the number of walk $X$. The normal "first vacant site is settled" rule is then $\rho = \inf\{t : \xi_{X(t)}^{W(X)} = 1\}$. Let $G_1 := G_1(n, v)$ be the "clique+edge": a clique $K_{n-1}$ attached by one edge to a single vertex $v$.

Proposition 1.1. *Define the following stopping rule on $G_1$*

$$\widetilde{\rho} = \inf\left\{t : (t \geq 3n\log(n) \textbf{ or } X(t) = v) \textbf{ and } \xi_{X(t)}^{W(X)} = 1\right\}$$

*Then the parallel or sequential process stopped according to $\widetilde{\rho}$ disperses in $O(n\log n)$ time, whereas for the standard rule $\rho$, $t_{seq}(G) = \Omega(n^2)$.*

To the best of our knowledge, the dispersion time and IDLA on a finite graph has not been studied before. Moore and Machta consider running IDLA walks synchronously for the purposes of simulating the limit shape [37] in parallel models of computation, however their results don't appear to overlap with ours. Simulating the process efficiently has also been studied more recently [22]. Thacker and Volkov [43] study a border DLA based growth model on finite graphs. This paper if of note as DLA a process related to IDLA and likewise is usually studied for infinite graphs however the questions studied by Thacker and Volkov differ from ours.

## 2 PRELIMINARIES

Throughout this work $G = (V, E)$ will always denote an undirected, unweighted, connected graph with $n$ vertices. We say that a graph $G$ is almost-regular if the ratio between maximum degree $\Delta(G)$ and minimum degree $\delta(G)$ is bounded from above by a constant. We call an almost-regular graph an expander if $1 - \lambda_2 = \Omega(1)$, where $\lambda_2$ is the second largest absolute eigenvalue. We say that a walk is lazy if it stays at its current vertex with probability $1/2$, thus the transition matrix $\widetilde{P}$ of the lazy walk is given by $\widetilde{P} = (I + P)/2$ where $P$ is the transition matrix of the simple random walk.

To recap we let $\tau_{par}^v(G)$ denote the dispersion time of the Parallel-IDLA process on $G$ started from $v$, that is the first iteration at which every vertex hosts (exactly) one particle. Similarly $\tau_{seq}^v(G)$ denotes the dispersion time of the Sequential-IDLA process on $G$ started from $v$, that is the longest time it takes a single particle to settle. Let $t_{seq}(G) = \max_{v\in V} \mathbf{E}[\tau_{seq}^v(G)]$ and $t_{par}(G) = \max_{v\in V} \mathbf{E}[\tau_{par}^v(G)]$. We shall drop the dependence on $G$ from our notation when the graph is clear from the context.

Further, let $t_{hit}(u, v) = \mathbf{E}[\tau_{hit}(u, v)]$, where $\tau_{hit}(u, v)$ is the time for a random walk to reach $v$ from $u$ and finally let $t_{hit}(G) := \max_{u,v\in V(G)} t_{hit}(u, v)$. For a probability distribution $\mu$ on $V$ and a set $S \subset V$ let $t_{hit}(\mu, S)$ denote the expected time for the walk starting from $\mu$ to hit any vertex in $S$.

Some results in the paper hold in expectation, some hold w.h.p. (with probability $1 - o(1)$) and others hold in both senses. One does not necessarily imply the other as the following counter example shows that in general (either) dispersion time does not concentrate. Recall $G_1$, the "clique+edge" graph from Proposition 1.1, and let $G_2 := G_2(n, v)$ be the "clique+hub+edge" graph consisting of a single edge $\{v, v^*\}$ attached at $v$ to $n/\log n$ vertices of $K_{n-2}$.

Proposition 2.1. *Let $D^v(G)$ denote either $\tau_{par}^v(G)$ or $\tau_{seq}^v(G)$. Then there exists $x \in V(G_1)$ and $y \in V(G_2)$ such that*

$$\mathbf{Pr}\left[D^x(G_1) \leq O\left(\mathbb{E}[D^x(G_1)]/n\right)\right] = \Omega(1),$$
$$\mathbf{Pr}\left[D^y(G_2) \geq \Omega\left(\mathbb{E}[D^y(G_2)] \cdot n\right)\right] = \Omega(1/n).$$

*Road Map.* The rest of this paper is organized as follows. We first present our general upper and lower bounds in Section 3 before turning to the coupling results in Section 4. In Section 5 we discuss how the results from Section 3 and Section 4 were applied to specific networks to give the results in Table 1, for some of these networks a more refined analysis is required. We conclude the paper in Section 6 with a summary of our results and some open problems. Note that in this conference version of the paper we have removed all the proofs apart from in the first part of Section 4. Where space allows we have tried to replace some of the proofs with some heuristic justification for the result in question. See [40] for full proofs.

## 3 GENERAL BOUNDS

### 3.1 Upper bounds

The first upper bound we present holds for any graph and only requires knowledge of the maximum hitting time of a random walk between two vertices. Although this result can be also recovered from the more general Theorem 3.3, it serves as a good "warm-up".

Theorem 3.1. *Let $G$ be any connected graph with $n$ vertices. Then*

- $\mathbf{Pr}\left[\sup_{v\in V}\tau^v_{par}(G) > 8\cdot t_{hit}(G)\cdot \log_2 n\right] \le \frac{1}{n}$
- $t_{par}(G) = O(t_{hit}(G)\cdot \log n)$.

*The same results also hold for $\tau^v_{seq}$ and $t_{seq}$.*

The rough idea of the proof is that if all $n$ walks are run for long enough to cover the whole graph then regardless of the interactions between the particles all of them will settle somewhere since they have each had the chance to visit every vertex. This simple bound is tight in many cases as seen in Table 1. The next result is a simple consequence, yet it actually provides the correct asymptotic worst-case bounds for the dispersion time.

Corollary 3.2 (General quantitative bounds on graphs).

- *For any $n$-vertex graph, $t_{seq}, t_{par} = O(n^3\log n)$.*
- *For any regular $n$-vertex graph, $t_{seq}, t_{par} = O(n^2\log n)$.*

Proof. Apply bounds on $t_{hit}$ from [34, Thm. 2.1] to Theorem 3.1. □

Both bounds above are sharp up to a multiplicative constant as witnessed by the lollipop and the cycle respectively, see Table 1. Also notice that both upper bounds exceed the corresponding upper bounds on the cover time [3, Thm. 6.12, Thm. 6.15] by a $\log n$-factor.

The upper bound in Theorem 3.1 matches Matthews bound for the cover time up to constant [30, Thm. 11.2]. While Theorem 3.1 is tight for the cycle, it is not tight for most "well-connected" graphs like expanders, high-dimensional grids and hypercubes. Thus for well connected graphs the dispersion time is usually of order $t_{hit}$. The behaviour of the log factor in the dispersion time contrasts with that of the log factor which may appear in the cover time.

*3.1.1 General Bounds in terms of Hitting Times of Sets.* In this section we shall achieve more refined bounds by considering hitting times of sets as opposed to vertices, and also mixing times. To avoid periodicity related issues we assume the trajectory of the particles is a lazy random walk. As shown in Theorem 4.7, running parallel (or sequential) IDLA with lazy walks slows down the process only by a factor of $2+o(1)$, thus any results established for the dispersion time with lazy walks also apply for non-lazy walk (up to a $2+o(1)$ factor) and vice versa. Define $\tau^v_{par}(G,k)$ to be the first time that less than $2^k-1$ vertices are left to be settled in the Parallel-IDLA, and let $t^k_{par}(G) = \max_{v\in V}\mathbf{E}\left[\tau^v_{par}(G,k)\right]$ denote the worst-case expectation. Clearly $\tau^v_{par}(G,1) = \tau^v_{par}(G)$, which is the standard parallel dispersion time.

Theorem 3.3. *Consider the Parallel-IDLA process with lazy walks. Then, for any connected $n$-vertex graph and any $k \ge 1$, we have*

$$t^k_{par}(G) \le 60\cdot \sum_{j=k}^{\lceil \log_2 n\rceil}\left(t_{mix} + \max_{S\subseteq V:\ |S|\ge 2^{j-2}} t_{hit}(\pi, S)\right).$$

One consequence of this theorem for $k = \log_2 n - 1$ is that within $O(t_{mix})$ steps, at least $n/2$ random walks are settled (this follows since by the duality between hitting time of large sets and mixing time [38], $\max_{S\subseteq V:\ |S|\ge n/4} t_{hit}(\pi, S) = O(t_{mix})$.

Note that the upper bound can be estimated directly to be at most $60\lceil\log_2 n\rceil\cdot(t_{mix}+t_{hit}) \le 120\lceil\log_2 n\rceil\cdot t_{hit}$, so this bound is (up to a multiplicative constant) a refinement of Theorem 3.1.

Let us now turn to the sequential process, where we can derive a similar bound, which turns out to be slightly stronger.

Theorem 3.4. *Consider the Sequential-IDLA process with lazy walks. Then, for any graph $G = (V, E)$, we have*

$$t_{seq}(G) \le 30\cdot \max_{1\le j\le\lceil\log_2 n\rceil}\left\{j\cdot\left(t_{mix} + \max_{S\subseteq V:\ |S|\ge 2^{j-2}} t_{hit}(\pi, S)\right)\right\}.$$

Neglecting constant factors, both upper bounds look comparable, but in fact it is not difficult to verify that the upper bound on $t_{seq}(G)$ is at most the upper bound on $t_{par}(G)$, up to constants. Conversely, the gap between the two upper bounds can be shown to be at most $O(\log n)$ however we do not know of any graph where the two dispersion are not the same up-to a multiplicative constant.

Note that both statements recover the basic $O(t_{hit}(G)\cdot\log n)$ upper bound, but as soon as there is a sufficient speed-up for hitting times of larger sets (and the mixing time is not too large), these bounds may give a bound of $O(t_{hit}(G))$ for certain graphs. We will see that this is indeed the case for several fundamental classes of graphs in Section 5, where we apply the previous bounds, and in particular Theorem 3.3.

Several bounds for expected hitting times of sets can be obtained by analyzing return probabilities, some of them are very tight. Since those bounds are more related to Markov chain properties than the IDLA process we restrict their statements and proofs to the online version of this paper [40].

## 3.2 Lower bounds

Theorem 3.5. *Let $G$ be a connected $n$-vertex graph with maximum degree $\Delta$, then $t_{seq}(G) = \Omega(|E|/\Delta)$. Hence in particular, $\Omega(n)$ is a lower bound for almost-regular graphs.*

The key idea used to prove this theorem is the fact that there is an ordering of the $n$ vertices so that if $u$ precedes $v$, then $t_{hit}(u,v) \le t_{hit}(v,u)$. Thus there is a vertex $w$ so that for any other vertex $v$, we have $t_{hit}(w,v) \ge t_{hit}(v,w)$. If we take $w$ as the origin then expected time to hit the last vertex is at least half the commute time. This is then bounded from below via the effective resistance.

Theorem 3.5 is tight up to constant when $G$ is the complete graph $K_n$, see Table 1. We also present a refined lower bound for trees.

Theorem 3.6. *Let $T$ be any $n$-vertex tree, then $t_{seq}(T) \ge 2n-3$.*

Proof of the above follows from the Essential edge Lemma [3, Lem. 5.1] since in a tree the last vertex to be settled must be a leaf.

Let $S_n$ be the $n$-vertex star and notice that $t_{seq}(S_n) \sim 2t_{seq}(K_{n-1}) \approx 2.6n$, thus Theorem 3.6 is tight up to a small multiplicative constant.

It would be natural to hope $t_{seq} = \Omega(t_{hit})$ should hold since one would expect the vertices with largest hitting times to be explored later by the sequential process and thus contribute to the dispersion time. We refute this with the following counter example.

Proposition 3.7. *Fix $0 < \varepsilon < 1/2$ and let $T$ be the complete binary tree on $n$ vertices with a path of length $n^{1/2-\varepsilon}$ attached to the root of the tree at one endpoint. Then*

$$t_{seq}(T) = O\left(n\cdot\log(n)^2\right) \qquad \textit{and} \qquad t_{hit}(T) = \Omega(n^{3/2-\varepsilon}).$$

The following lower bound is tight up to a $\log n$ factor as witnessed by the cycle, see Table 1.

Proposition 3.8. *Consider the Sequential-IDLA with lazy walks on an almost regular graph $G$. Then*

$$t_{seq}(G) = \Omega(t_{\text{mix}}) = \Omega\left(\frac{1}{1-\lambda_2}\right) = \Omega\left(\frac{1}{\Phi}\right),$$

*where $\Phi$ and $\lambda_2$ are, respectively, the conductance and second largest eigenvalue associated to the lazy random walk on $G$.*

## 4 COUPLING AND STOCHASTIC DOMINATION

In this section we shall prove the following stochastic domination.

Theorem 4.1. *Let $G$ be a finite graph and $v \in V(G)$. Then*

$$\tau^v_{seq}(G) \preceq \tau^v_{par}(G).$$

An immediate corollary of this result is the inequality

$$\mathbf{E}\left[\, \tau^v_{seq}(G) \,\right] \le \mathbf{E}\left[\, \tau^v_{par}(G) \,\right],$$

we also prove the reverse inequality up to $\log n$ factors.

Theorem 4.2. *Let $G$ be a finite graph and $v \in V(G)$. Then*

$$\mathbf{E}\left[\, \tau^v_{par}(G) \,\right] = O\Big(\log(n) \cdot \mathbf{E}\left[\, \tau^v_{seq}(G) \,\right]\Big).$$

The proofs of the above theorems are based on a coupling between the Sequential and Parallel-IDLA processes. To construct this coupling we consider a (Parallel or Sequential) IDLA process on $G$ as an irregular 2-dimensional array $L$ where each element $L(i, j) \in V$. This array $L$ (also referred to as a block) has $n$ rows representing the $n$ particles. Column $t$ represents time $t$, and thus $L(i, t)$ represents the vertex visited by walk $i$ at time $t$. We let $\rho_i$ denote the length of walk $i$, hence the index of each row $i$ goes from 0 to $\rho_i$. We denote by $\mathcal{I}_L$ the set of all indices $(i, t)$ of the array $L$.

Given $(i, s), (j, t) \in \mathcal{I}_L$, we say that $(i, s)$ is smaller than $(j, t)$ in sequential order, written $(i, s) <_S (j, t)$ if either $(i < j)$ or $(i = j, s < t)$. Thus in sequential order, the block is read

$$L(1, 0), \dots L(1, \rho_1), L(2, 0), \dots L(2, \rho_2), \dots, L(n, 0), \dots, L(n, \rho_n).$$

Likewise we say that $(i, s)$ is smaller than $(j, t)$ in parallel order, denoted by $(i, s) <_P (j, t)$ if either $(s < t)$ or $(s = t, i < j)$. So in parallel order, the block is read

$$L(1, 0), \dots, L(n, 0), L(1, 1), \dots, L(n, 1), \dots, L(1, r), \dots, L(n, r), \dots$$

where if $r > \rho_i$ then $L(i, r)$ is empty so it is skipped.

Note that if $L$ is a block representing a parallel or Sequential-IDLA the following property holds

$$L(i, \rho_i) \ne L(j, \rho_j) \text{ for each pair } i \ne j. \qquad (1)$$

If $L$ satisfies (1) then $\{L(i, \rho_i) : i \in [n]\} = V$ and the final element of each row is unique.

A block $L$ satisfying (1) represents a Sequential-IDLA process if and only if each row $i$ represents a path in $G$ from vertex $L(i, 0) = v$ to $L(i, \rho_i)$ and for all $(i, j) \in \mathcal{I}_L$

$$(i, t) \text{ is the first occurrence of } L(i, t) \in L \text{ w.r.t. } <_S \text{ iff } t = \rho_i. \quad (2)$$

This says that when $L$ is read in sequential-order the first time a new vertex is read it ends the current row.

Similarly a block $L$ satisfying (1) represents a realization of a Parallel-IDLA process if and only if each row $i$ represents a path in $G$ from vertex $L(i, 0) = v$ to $L(i, \rho_i)$ and and for all $(i, t) \in \mathcal{I}_L$

$$(i, t) \text{ is the first occurrence of } L(i, t) \in L \text{ w.r.t. } <_P \text{ iff } t = \rho_i. \quad (3)$$

For a 2-dim array $L$ we denote its total length (the work done) by $W(L)$, this is the total number of moves recorded by $L$ and thus $W(L) := \rho_1 + \cdots + \rho_n$. Let $\text{Seq}^m_v$, or $\text{Par}^m_v$, denote the set of all sequential, respectively parallel, blocks representing realisations of IDLA processes starting from $v$ and total length $m$, i.e. $W(L) = m$.

To build the coupling between Sequential and Parallel-IDLA, we are going to use a series of "Cut & Paste" transformations. Consider $(i, t) \in \mathcal{I}_L$, then define $\mathbf{CP}_{(i,t)}(L)$ as the block constructed by taking $L$ and cutting the cells $(i, t+1), \dots, (i, \rho_i)$ and pasting it after the unique $(k, \rho_k)$ with $L(i, t) = L(k, \rho_k)$.

**Example:** Represented below are $L$ - a block on $V = \{1, 2, 3, 4\}$, and $\mathbf{CP}_{(4,1)}(L)$ - the result of applying the cut & paste $\mathbf{CP}_{(4,1)}$ to $L$.

$$L = \begin{array}{|c|c|c|c|c|c|}\hline 1 \\ \hline 1 & 2 \\ \hline 1 & 2 & 2 & 3 \\ \hline 1 & 2 & 1 & 2 & 3 & 4 \\ \hline \end{array} \qquad \mathbf{CP}_{(4,1)}(L) = \begin{array}{|c|c|c|c|c|c|}\hline 1 \\ \hline 1 & 2 & 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 2 & 3 \\ \hline 1 & 2 \\ \hline \end{array}$$

While $\mathbf{CP}_{(1,0)}(L) = \mathbf{CP}_{(2,1)}(L) = \mathbf{CP}_{(3,3)}(L) = \mathbf{CP}_{(4,5)} = L$. Note that if $L$ satisfies property (1), then $L' = \mathbf{CP}_{(i,t)}(L)$ also satisfies (1). Property (1) is an important invariant for our algorithms.

### 4.1 Algorithms

We propose two algorithms **StP** and **PtS**, formally specified by Algorithms 1 and 2 below. The algorithm **StP** transforms a sequential process into a parallel and **PtS** transforms a parallel process into a sequential. The key component of both algorithms is the "cut & paste" operation **CP**.

Both algorithms work as follows: a pointer moves through the input array $L$ in a fixed order and when the pointer sees a vertex label for the first time this label is added to the set $\mathcal{S}$ of seen vertices and a cut & paste transform **CP** is applied to $L$ at this position before the pointer continues. The difference is that in **StP** the pointer explores columns then rows (i.e. in parallel order $<_P$), whereas **PtS** reads rows then columns (i.e. in sequential order $<_S$).

Broadly speaking the algorithms try to read the input array as if it was of the type specified by the output and if the input fails to have this form then it will edit it using the cut & paste transform until it has the correct form.

Notice the ordering of the **for** and **while** loops is reversed between the **StP** and **PtS** algorithms.

The set $\mathcal{S} = \mathcal{S}(L, t)$ stores the different values of $L(i, j)$ observed after $t$ iterations of the innermost loop. The algorithms terminate once they have scanned the whole array, this is the first time when $|\mathcal{S}| = n$. Sometimes the algorithms may apply $\mathbf{CP}_{(i,j)}$ with $j = \rho_i$, this leaves $L$ unchanged.

Lemma 4.3. *The following holds,*

- **PtS** *is a bijection from* $\text{Par}^m_v$ *to* $\text{Seq}^m_v$.
- **StP** *is a bijection from* $\text{Seq}^m_v$ *to* $\text{Par}^m_v$.

Proof. We first observe that during the running of the **PtS** and **StP**, Algorithms 1 & 2, the only changes made to the input array $L$ are a sequence of cut & paste transforms $\mathbf{CP}_{i_1,t_1}, \mathbf{CP}_{i_2,t_2} \dots$. Since

```
Result: transforms a sequential array $L$ into a parallel array
$\mathcal{S} \leftarrow \emptyset$;
$t \leftarrow 0$;
while $|\mathcal{S}| < n$ do
    for $i = 1..n$ do
        if $(i,t) \in \mathcal{I}_L$ and $L(i,t) \notin \mathcal{S}$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{L(i,t)\}$;
            $L \leftarrow \mathbf{CP}_{(i,t)}(L)$;
        end
    end
    $t \leftarrow t + 1$;
end
return $L$;
```
**Algorithm 1:** Sequential to Parallel (**StP**)

```
Result: transforms a parallel array $L$ into a sequential array
  $\mathcal{S} \leftarrow \emptyset$;
1 for $i = 1..n$ do
      $t \leftarrow 0$;
2     while $(i,t) \in \mathcal{I}_L$ do
3         if $L(i,t) \notin \mathcal{S}$ then
4             $\mathcal{S} \leftarrow \mathcal{S} \cup \{L(i,t)\}$;
5             $L \leftarrow \mathbf{CP}_{(i,t)}(L)$;
              exit
          end
          $t \leftarrow t + 1$;
      end
  end
  return $L$;
```
**Algorithm 2:** Parallel to sequential (**PtS**)

each cut & paste transformation preserves Property (1) it follows that **PtS** and **StP** preserve Property (1). Likewise cutting & pasting preserves total length, thus so do **PtS** and **StP**. Recall that the operator $\mathbf{CP}_{(i,t)}$ cuts and pastes the random walk trajectory $(i, t+1), \dots, (i, \rho_i)$ onto the unique $(k, \rho_k)$ with $L(i,t) = L(k, \rho_k)$. Thus row $k$ in $L' = \mathbf{CP}_{(i,t)}$ is a valid path from vertex $L(0,k)$ to $L(i, \rho_i)$.

For **PtS** we must check that if $L \in \mathrm{Par}_v^m$, then $\mathbf{PtS}(L) \in \mathrm{Seq}_v^m$, i.e. $\mathbf{PtS}(L)$ satisfies (2). Recall that the **PtS** algorithm reads the input array $L$ in sequential order and when a vertex label is seen for the first time at some position $(i,j)$ it applies the cut & paste transform $\mathbf{CP}_{(i,j)}$ and the pointer moves to the next row. If $(i, j+1)$ is non-empty then $\mathbf{CP}_{(i,j)}$ pastes the remainder of row $i$ to some row $i'$ with endpoint value $L(i,j)$. Observe that $i' > i$ since $(i,j)$ is the first occurrence of $L(i,j)$ in sequential order. Thus each new vertex found w.r.t. $<_S$ forms an endpoint as it is cut when it is first discovered and nothing else can be pasted onto that row later by the algorithm. This proves that $\mathbf{PtS}(L)$ is a valid Sequential-IDLA block.

Likewise for **StP** let $L \in \mathrm{Seq}_v^m$ and we check $\mathbf{StP}(L)$ satisfies (3). Suppose when reading $L$ in parallel order $(i,j)$ is the first occurrence of $L(i,j)$, **StP** will apply $\mathbf{CP}_{(i,j)}$ and continue to read the array in parallel order. Position $(i,j)$ is now fixed as the end point of row $i$ as no later copy & paste can alter this row. This holds since to paste something else onto row $i$ we would have to see vertex $L(i,j)$ for the first time (again) later in parallel order which cannot happen.

For injectivity let $\mathbf{F}$ represent either of the maps **PtS**, **StP**, and $L, L'$ be distinct arrays both from $\mathrm{Par}_v^m$ or $\mathrm{Seq}_v^m$ respectively. Assume for a contradiction that $\mathbf{F}(L) = \mathbf{F}(L')$. Since $L \neq L'$ there is a first position $(i,j)$ at which they differ w.r.t. $<_S$ or $<_P$, i.e. $L(i,j) \neq L'(i,j)$. It cannot be the case that $L(i,j) = \emptyset$ and $L' \neq \emptyset$, or vice versa, since otherwise the arrays must differ at position $(i, j-1)$ which occurs before $(i,j)$ in either ordering. Let $(i,j)$ be the current position when $\mathbf{F}$ is is running on $L$ and $L'$. If $L(i,j) \notin \mathcal{S}(t, L)$ and $L'(i,j) \notin \mathcal{S}(t, L')$ then $\mathbf{CP}_{(i,j)}$ is applied and the position $(i,j)$ is now fixed in both arrays, i.e. $\mathbf{F}(L)(i,j) \neq \mathbf{F}(L')(i,j)$, a contradiction. Similarly if $L(i,j) \in \mathcal{S}(t,L)$ and $L'(i,j) \in \mathcal{S}(t, L')$ then no transform is applied and the positions are fixed. Otherwise the element at $(i,j)$ is seen in one array and not in the other, w.l.o.g. assume $L(i,j) \notin \mathcal{S}(t,L)$ and $L'(i,j) \in \mathcal{S}(t,L)$. In this case a $\mathbf{CP}_{(i,j)}$ is applied to $L$ but not to $L'$ and both positions fixed, again we have a contradiction as $\mathbf{F}(L)(i,j) \neq \mathbf{F}(L')(i,j)$.

For bijectivity since $\mathbf{StP} : \mathrm{Seq}_v^m \to \mathrm{Par}_v^m$ and $\mathbf{PtS} : \mathrm{Par}_v^m \to \mathrm{Seq}_v^m$ are both injections and $\mathrm{Seq}_v^m, \mathrm{Par}_v^m$ are finite it follows that $|\mathrm{Seq}_v^m| = |\mathrm{Par}_v^m|$. Thus **StP**, **PtS** are surjections. □

One can prove **StP** has inverse **PtS** but we do not use this fact.

Lemma 4.4. *Let $L \in \mathrm{Seq}_v^m$. Then $\max_{i \in \mathcal{I}_L} \rho_i \leq \max_{i \in \mathcal{I}_{\mathrm{StP}(L)}} \rho_i$.*

Proof. Assume for a contradiction that

$$\max_{i \in \mathcal{I}_L} \rho_i > \max_{i \in \mathcal{I}_{\mathrm{StP}(L)}} \rho_i.$$

This means that each row attaining maximum length in $L$ must have a section cut and pasted to a row of shorter length by the **StP** algorithm. However the **StP** algorithm runs in parallel order and cannot paste onto a cell which it has already read. Thus any row suitable to receive the end of the current row must have its end point in the same column or a column to the right of the current one. This cannot decrease the length of the longest row. □

We now have what we need to prove that $\tau_{seq}^v(G) \leq \tau_{par}^v(G)$.

Proof of Theorem 4.1. By Lemma 4.3 **StP** is a bijection between $\mathrm{Par}_v^m$ and $\mathrm{Seq}_v^m$. Thus we can pair every sequential process $L$ of total length $W(L) = m$ with a unique parallel process $L'$ of total length $W(L') = m$. Both $L$ and $L'$ visit the same vertices with the same frequency and in the same order, thus the probability of each vertex sequence of total length $m$ in either process is the same. This implies that the total lengths of the processes are distributed identically.

Lemma 4.4 states that for this pair the longest row in $L'$ is at least as long as the longest row in $L$. Thus for any $k, m \geq 0$,

$$\mathbf{Pr}\left[\max_{i \in \mathcal{I}_L} \rho_i \geq k \,\middle|\, W(L) = m\right] \leq \mathbf{Pr}\left[\max_{i \in \mathcal{I}_{L'}} \rho_i \geq k \,\middle|\, W(L') = m\right].$$

The result follows as $\tau_{seq}^v(G)$ and $\tau_{par}^v(G)$ are given by the length of the longest row in either array. □

We shall now prove Theorem 4.2 which states that

$$\mathbf{E}\left[\,\tau_{par}^v(G)\,\right] = O\Big(\log(n) \cdot \mathbf{E}\left[\,\tau_{seq}^v(G)\,\right]\Big),$$

for any $G$ and $v \in V(G)$.

Proof of Theorem 4.2. Let $L$ be a Parallel-IDLA block and $\sigma$ be a random permutation of $\{2, \ldots, n\}$. Let $\sigma(L)$ be the block that results from permuting the rows of $L$ using $\sigma$. The block $\sigma(L)$ represents a Parallel-IDLA process where conflicts between particles are solved by giving priority to particles with least value of $\sigma(index)$ (instead of least $index$, as per the definition of Parallel-IDLA). Also, for simplicity we fix $\sigma(1) = 1$. Note that $L$ and $\sigma(L)$ have the same rows, and thus the maximum row-length is the same in both blocks. We remark that **PtS**, Algorithm 2, still produces a valid sequential array even if the input is $\sigma(L)$ instead of $L$.

Let $L$ be an arbitrary parallel array and consider a run of **PtS**, Algorithm 2, on $\sigma(L)$ where we do not reveal $\sigma$ in advance. Instead we reveal the permutation $\sigma$ row by row as **PtS** reads the array in sequential order (in other words, instead of running **PtS**$(\sigma(L))$, we equivalently run **PtS**$(L)$ but we read rows in random order, starting with row 1 ($= \sigma(1)$) of $L$, and then rows $\sigma(2), \sigma(3), \ldots, \sigma(n)$. This is equivalent to replacing $i$ by $\sigma(i)$ in lines 1-5 of Algorithm 2). Note that the Cut & Paste operation is unaffected by not revealing the order of the rows. This holds because the Cut & Paste transform only pastes behind unread rows, independent of their location in the array $L$ and what is more, there is only one row where we can paste a cut section by property (1). Consider the largest row (or choose one arbitrarily if there is more than one) in the original block $L$. We shall paint this row red and call the last cell $\xi$. During the running of **PtS**$(L)$ the marked cell $\xi$ moves from row to row because of the Cut & Paste operations. Here is the key observation: If $\ell$ is the length of the original red row and $\xi$ moves no more than $N$ times then in the output array **PtS**$(L)$ has a row of length at least $\ell/N$. This holds because the red row was partitioned $N$ times and thus one of the pieces has to have length at least $\ell/N$

Let $i_k$ be the iteration (how many rows we have read) by the $k^{th}$ time **PtS** reads a row containing the marked cell $\xi$. When we read a row which contains $\xi$ for first time in iteration $i_1$, we may apply a Cut & Paste somewhere in this row (if not we are done). If so $\xi$ would find itself at the end of an unread row $x_2$ of $L$, which will be read in a (random) iteration $i_2$, i.e. $\sigma(i_2) = x_2$. Note $i_2$ is a uniform random value in $\{i_1 + 1, \ldots, n\}$. In iteration $i_2$, we read the row with the marked cell and again, the algorithm might cut and paste this row behind an unread row $x_3$ which will be read at some time $i_3$, which is again uniformly random in $\{i_2 + 1, \ldots, n\}$, and so on. Each time we make a cut and paste the index $i_j$ of the recipient row will be in the latter half of the list $\{i_j + 1, \ldots, n\}$ with probability $1/2$. Thus since **PtS** works through this list in order the expected length of the list of possible positions for the next value $i_{j+1}$ halves every iteration. We cannot keep halving this list indefinitely because either at some point a row ended by $\xi$ is not cut or $\xi$ is in the last row to be read (which is never cut). Thus the number of times $\xi$ moves (i.e. expected times the longest row is cut) is at most $C \log n$ with probability at least $1/2$ by Markov's inequality. Denote by $X$ the (random) number of times we cut a row containing the marked cell $\xi$. Let $\ell$ be the length of the longest row of $L$, and $\ell'$ the random variable representing the length of the longest row of **PtS**$(L)$ using a random permutation $\sigma$. Conditional on cutting $L$'s longest row $X$ times, we have must have at least one row of length $\ell/X$ once the algorithm has terminated. Thus, given the block $L$ with largest row $\ell$, we have

$$\mathbf{E}\left[\,\ell' \mid L\,\right] > \mathbf{E}\left[\,\ell' \mid L, X \le C\log n\,\right] \cdot \frac{1}{2} \ge \frac{\ell}{2C\log n}.$$

By taking expectation over all blocks $L$ generated from a Parallel-IDLA with a random $\sigma$ we conclude the result. □

## 4.2 Uniform-IDLA

Recall that in the Sequential-IDLA we run the walks one by one in order and walk $i + 1$ starts only after walk $i$ has settled, while in the Parallel-IDLA all particles walk simultaneously until they settle, breaking ties by settling the particle with smallest index. In either Sequential or Parallel we are interested in the longest walk. Another natural way to run the IDLA process is in uniform order: we choose a random unsettled particle and move it to a random neighbouring vertex which it settles on if unoccupied. We call this process the Uniform-IDLA. This process can be seen as lying between the Sequential and Parallel-IDLA models. To sample from the Uniform-IDLA process, we first consider an infinite sequence $(R_i)$ where the $R_i$ are independent random variables sampled from $\{2, \ldots, n\}$. Then we run the Uniform-IDLA as following: First particle 1 settles at the origin, so the origin is occupied. Then, at each time-step $t \ge 1$, if particle $R_t$ is unsettled, it moves to a random neighbour, otherwise it stays in its current location. If such neighbour is not occupied, particle $R_t$ settles on it and the vertex is now occupied.

Given the random ordering $R$, we can find a bijection between the Uniform-IDLA and Parallel-IDLA. Given $R$, an $R$-block is defined in the same fashion as a parallel block, i.e. $L(i, j)$ represents the position of the $i$-th particle after $j$ jumps, but additionally, we associate to every $(i, j) \in \mathcal{I}_L$ an integer $T(i, j)$. This $T$ is called the timing array and defined as $T(i, j) = t$ if $R_t = i$ for $j$-th time and $T(i, 0) = 0$ for all particles $i$. Note that using the block and timing array we can reconstruct the uniform process as we have not only the paths but the time-steps when the particles moved.

The bijection between an $R$-block and a parallel block is defined algorithmically in the same fashion as before. To transform an $R$-uniform block into a parallel block we just apply **StP**, Algorithm 1, to the $R$-uniform block oblivious to $R$ since **StP** reads in parallel order. However to transform a parallel block into an $R$-uniform block, we must read the block in the order given by $T(i, t)$ (i.e. read the block with smallest value $T(i, t)$, then the second smallest, etc..) and apply $\mathbf{CP}_{i,j}$ whenever the vertex $L(i, j)$ is read for first time. It is very important that now when applying the Cut & Paste operation we move not only the cells containing a portion of the path but also the times $T(i, t)$ associated to those cells, i.e. if cell $(i, t)$ moves to $(j, s)$ then $T(j, s)$ gets the value of $T(i, j)$, while $T(i, t)$ is left undefined. Pseudo-code for the procedure we have just described is given in Algorithm 3.

Let $\mathrm{Unif}^m_{R,v}$ be the set of all Uniform IDLA blocks with ordering $R$ starting from $v$ with total number of random walk steps $m$. Denote by $\tau^v_{Unif,R}(G)$ the longest path in a Uniform IDLA process given $R$. Then, using the same arguments as the in the sequential-parallel case we obtain.

Theorem 4.5. *For any given ordering $R$ there is a bijection between $\mathrm{unif}^m_{R,v}$ and $\mathrm{Par}^m_v$. Moreover the number of steps taken by the longest*

```
Result: transforms a parallel array $L$ and order sequence $R$
        into a $R$-Uniform array
$\mathcal{S} \leftarrow \emptyset$;
$C \leftarrow$ list of cells $(i,t)$ ordered by $T(i,t)$ in increasing order;
$k \leftarrow 0$;
while $|\mathcal{S}| < n$ do
    $k \leftarrow k+1$;
1   $(i,t) \leftarrow C(k)$;
2   if $L(i,t) \notin \mathcal{S}$ then
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{L(i,t)\}$;
        $L \leftarrow \mathbf{CP}_{(i,t)}(L)$;
    end
end
return $L$;
```

**Algorithm 3:** Parallel to $R$-Uniform ($\mathbf{PtU}_R$)

*walk of the Uniform IDLA is stochastically dominated by the number of steps in the longest walk of the Parallel-IDLA.*

Observe however that the dispersion time of the Uniform array is not determined purely by the number of steps/length of the longest row but by the values $T(i,j)$ of the timing array.

## 4.3 Continuous-time IDLA

We consider continuous-time versions of the Sequential and Uniform IDLA process. For the Sequential-IDLA it is easy to consider its continuous-time analogue, we just have random walks that jump at times given by a Poisson process of intensity 1. Also, we can easily sample from the continuous-time Sequential-IDLA by sampling a discrete time IDLA and then considering independent exponential times of mean 1 between the jumps. Let $\tau^v_{c,seq}(G)$ be the time it took to the slowest particle to settle in the continuous-time Sequential-IDLA. Standard concentration inequalities shows that for any origin vertex $v$,

$$\tau^v_{seq}(G) = (1+o(1)) \cdot \tau^v_{c-seq}(G)$$

holds with high probability provided $\tau^v_{seq}(G) > n^\alpha$ w.h.p. for some $\alpha > 0$. The equality also holds in expectation by noticing that $\mathbf{E}\left[\,\tau^v_{seq}(G)\,\right]$ is polynomial (at most $O(n^3 \log n)$).

Another natural continuous-time process is the Uniform IDLA. In this process each particle has an exponential clock of rate 1. Then, as long as the particle is not settled, when the clock rings the particle moves to a random neighbour and settles if possible. Note that this is equivalent to running the discrete-time Uniform IDLA but putting exponentials of mean $1/(n-1)$ between each time-step (remember particle 1 occupies the origin and $R_t$ takes values in $\{2,\ldots,n\}$). Alternatively, we can sample the continuous-time Uniform IDLA by using $\mathbf{PtU}_R$, Algorithm 3. First, sample a (discrete-time) Parallel-IDLA. Then run Algorithm 3 but instead of using the list $C$ to choose the next cell $(i,t)$ (line 1), each row of the block has a exponential clock of mean 1. When the clock of row $i$ rings, the algorithm chooses the first unread cell of row $i$ (if it exists), and proceeds with line 2. We shall name this procedure $\mathbf{PtU}_C$. This algorithm can be shown correct due the bijection between $\mathrm{Unif}^m_{R,v}$ and $\mathrm{Par}^m_v$ for a fixed ordering $R$ established in Theorem 4.5.

Let $\tau^v_{c-unif}(G)$ be the time it takes the continuous-time Uniform IDLA (CTU-IDLA) started from $v$ to settle all the particles. The following result relates $\tau^v_{c-unif}(G)$ and $\tau^v_{par}(G)$ provided that w.h.p. the Parallel-IDLA process does not end to quickly on $G$.

Theorem 4.6. *If $n^\alpha \le \tau^v_{par}(G)$ holds w.p. $1-n^{-\beta}$ for some $\alpha, \beta > 0$, then*

$$\tau^v_{c-unif}(G) = (1+o(1)) \cdot \tau^v_{par}(G),$$

*w.p. $1-o(1)$ and in expectation.*

## 4.4 Lazy IDLA

Consider the lazy versions of the discrete-time Sequential and Parallel-IDLA models, where with probability $1/2$ particles remain at their current vertex and otherwise choose a neighbour uniformly. Note that all our previous results using the cut & paste bijections are also valid for lazy walks as for example one can simply consider the graph with the addition of (multi)-loops at each vertex. Indeed, they are valid for any block that is generated by using a Markov chain to move the particles.

Let $\tau^v_{L-seq}(G)$, $\tau^v_{L-par}(G)$, be the number of steps needed to complete the lazy Sequential, respectively lazy Parallel, IDLA process started from $v$.

Although we are mainly concerned with the simple random walk IDLA models would like to be able to switch to the lazy setting at times as it allows us to use mixing time results. For the Sequential it is fairly clear that up to lower order terms the lazy sequential is a factor of 2 slower than the Parallel, using the continuous time Uniform IDLA we can also show this for Parallel-IDLA under the same assumptions as Theorem 4.6.

Theorem 4.7. *If $\tau^v_{seq}(G) \ge n^\alpha$ holds w.p. at least $1-n^{-\beta}$ for some $\alpha, \beta > 0$, then*

$$\tau^v_{seq}(G) = (2+o(1)) \cdot \tau^v_{L-seq}(G).$$

*If $\tau^v_{par}(G) \ge n^\alpha$ holds w.p. at least $1-n^{-\beta}$ for some $\alpha, \beta > 0$, then*

$$\tau^v_{par}(G) = (2+o(1)) \cdot \tau^v_{L-par}(G),$$

*The equality in both equations above hold w.h.p. and in expectation.*

## 5 FUNDAMENTAL NETWORKS

In this section we discuss how to establish the dispersion for many well known network topologies and how properties of the network in question affect dispersion. The two right most columns of Table 1 contain the results mentioned in this section, full proofs of these results can be found in the full length version of this paper [40].

*Cycle/Path.* For these graphs the bound $O(t_{hit} \cdot \log n)$ (which gives $O\big(n^2 \log n\big)$) is tight. A simple lower bound of $\Omega(n^2)$ holds since with constant probability the last walk has to cover a distance of order $n$ and thus takes time $\Omega(n^2)$. This lower bound can be improved to match the upper bound by showing that with constant probability polynomially many walkers must cover a distance of order $n$ and that at least one of these walkers takes time $\Omega(n^2 \log n)$ to settle. For the path we prove a little more.

Theorem 5.1. *Let $P_n$ be the path with n-vertices. Let $M$ be the maximum of n independent random variables representing the hitting time of a random walk to the vertex n, starting from 1 on $P_n$. Then*

$$t_{seq}(P_n), t_{par}(P_n) = (1 \pm o(1)) \cdot \mathbf{E}[M].$$

This show that on the path the leading order terms are the same for both processes.

*2-dim Grid/Torus.* The planar grid/tori on $n$ vertices is the only graph stated where we do not know the dispersion time up to constants. The best upper bound of $O(n(\log n)^2)$ comes from the $O(t_{hit} \cdot \log n)$ upper bound (in this case the bound cannot be improved by considering hitting times of sets such as in Theorem 3.3). For the lower bound of $\Omega(n \log n)$ one appeals to the shape theorem for $\mathbb{Z}^2$ and argues that after $\alpha n$ particles have settled, for some $\alpha < 1$, the aggregate contains a euclidean ball of radius $\Omega(\sqrt{n})$ around the origin w.h.p.. It follows that one of the last $\Omega(n)$ many walkers will take time $\Omega(n \log n)$ to exit this ball with constant probability.

*Expanders, Hypercube, d-dim Grids/Tori where $d \geq 3$.* Each of these graphs is regular or almost regular and so a lower bound of $\Omega(n)$ holds by Theorem 3.5. These families are an example where the basic bound $O(t_{hit} \log n)$ is not tight and $O(t_{hit})$ is the correct answer. To achieve this bounds on hitting times of sets are calculated using return probabilities and rapid mixing for lazy random walks on these graph and then applied in conjunction with Theorem 3.3.

*Cliques.* For the complete graph $K_n$ we have

$$t_{seq} = (\kappa_{cc} + o(1)) \cdot n \qquad \text{and} \qquad t_{par} = (\pi^2/6 + o(1)) \cdot n.$$

The Sequential dispersion time $\tau^v_{sec}(K_n)$ is equivalent to the length of the longest walk in the coupon collector problem, denoted $T_n$, and thus is distributed as the maximum of $n$ independent geometric random variables with parameters $\frac{n-i+1}{n}$ for $1 \leq i \leq n$. Brennan et. al. [15] show that $\mathbf{E}[T_n]/n \longrightarrow \kappa_{cc}$ where

$$\kappa_{cc} := \sum_{i=1}^{\infty} \left( \frac{2}{i(3i-1)} - \frac{2}{i(3i+1)} \right) \approx 1.255. \tag{4}$$

For the Parallel dispersion time there are issues when two or more particles settle at the same time. These problems are avoided by moving to the continuous time Unifom-IDLA model (CTU-IDLA). We can do this since by Theorem 4.6 the dispersion times in the two models are the same up to lower order terms. If there are $k$ unsettled particles in the CTU-IDLA, then the time until the next particles settles is exponentially distributed with mean $(n-1)/k^2$. Thus summing these expected waiting times yields the result.

*Binary tree.* For the (complete) binary tree of height $h$ on $n = 2^h$ vertices the $O(t_{hit} \cdot \log n)$ upper bound is tight and thus, similarly to the path/cycle, one must show that there is at least one slow particle who takes time $\Omega(t_{hit} \cdot \log n)$ to settle. It is clear that the last vertex to be settled is a leaf and thus the last walk takes time $t_{hit} = \Omega(n \log n)$ with constant probability. However, to show that there is a particle which takes a $\log n$ factor longer than this requires one to establish some quite detailed results about the aggregate at an advanced state in the process. It is shown in the full paper [40] that w.h.p. there is some $t \leq n - n^a$ such that after $t$ particles have been released in the Sequential-IDLA all remaining unvisited vertices lie in a sub-tree which has its root at distance $\Omega(\log n)$ from the origin of the walkers (root of the binary tree). One can then show that at least one of the remaining $n^\alpha$ walks take $\Omega(t_{hit} \cdot \log n)$ steps before entering the sub-tree and settling.

## 6 CONCLUSIONS

### 6.1 Summary of our results

The aim of this project is to better understand IDLA processes on finite graphs. The main tool we developed to gain an insight on the processes is the Cut & Paste bijection. This bijection allows us to study directly the affect of the different scheduling protocols on the random walk trajectories. We use this bijection to couple the various IDLA variants allowing us to order or equate their dispersion times and show that $t_{seq}$ and $t_{par}$ are equal up to a multiplicative factor of order $\log n$.

In addition to the qualitative information provided by the bijection we also develop a collection of upper and lower bounds phrased in terms of graph and random walk quantities which are easier to compute. These quantities are max degree, number of edges, mixing time and hitting times of vertices/sets by a single random walk. These bounds enable us to establish the correct asymptotic order of the dispersion time for the Parallel and Sequential processes on several natural networks. They also provide some general bounds in terms of $n$ which are shown to be tight. In summary, our findings reveal that for almost all natural graphs the dispersion time is either of order $t_{hit}$ or $t_{hit} \cdot \log n$. However we also show with there is a graph, see Proposition 3.7, where the dispersion time is smaller than $t_{hit}$ by a multiplicative factor of almost $1/\sqrt{n}$.

### 6.2 Further directions

As pointed out earlier, our results establish the correct asymptotic order of the dispersion time for most natural networks. The only exception is the $2d$-grid, where the dispersion time is shown to be between $\Omega(n \log n)$ and $O(n \log^2 n)$. The known shape theorems for the *infinite* 2d-grid, empirical simulations as well as the result for binary trees all strongly suggest the dispersion to be of order $n \log^2 n$. This provides us with the first open problem .

Open Problem 1. *Determine the dispersion time of the $2d$-grid/torus.*

The second main open problem is whether for any graph, the sequential and parallel dispersion time are of the same order. Note however that we show, see Table 1, that already for the clique, the Parallel-IDLA is about 30 percent slower than the Sequential-IDLA. Thus, we cannot have equality between the two processes, even though the path is an example where the Parallel-IDLA and Sequential-IDLA have the same dispersion time up to lower order terms.

Open Problem 2. *Is it true that for all graphs, $t_{par} = O(t_{seq})$?*

We know of no graph where this does not hold however it seems hard to prove. In order to prove this result, it might be useful to derive some general lower bounds on the dispersion time, which are in turn interesting on their own right. One lower bound which we conjecture to hold is that for any graph, the sequential (or parallel) dispersion time takes $\Omega(n)$.

The following conjecture is motivated by the idea that when you run **StP** algorithm (see Section 4) the random walk sections cut and pasted do not have to cover the graph. If true, then this conjecture would resolve the open problem above for some classes of graphs.

Conjecture 6.1. *Let $t_{cov}(G)$ be the cover time. Then*

$$t_{par}(G) \leq t_{seq}(G) + t_{cov}(G).$$

The counter example to concentration (Proposition 2.1) from Section 2 motivates the following open problem.

Open Problem 3. *What conditions must a graph satisfy for the dispersion time to concentrate around its expectation?*

Other interesting variants of the dispersion process are when the number of particles is either considerably smaller or considerably larger than the number of sites (it is conceivable to believe that the dispersion times are maximal if the two numbers are equal). Finally, another direction is to study a version of the dispersion process where the origin is sampled independently and uniformly at random for each particle.

## REFERENCES

[1] Heiner Ackermann, Simon Fischer, Martin Hoefer, and Marcel Schöngens. 2011. Distributed algorithms for QoS load balancing. *Distributed Computing* 23, 5-6 (2011), 321–330.

[2] Hoda Akbari and Petra Berenbrink. 2013. Parallel Rotor Walks on Finite Graphs and Applications in Discrete Load Balancing. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '13)*. ACM, New York, NY, USA, 186–195.

[3] David Aldous and James Allen Fill. 2002. Reversible Markov Chains and Random Walks on Graphs. (2002). Unfinished monograph, recompiled 2014.

[4] Noga Alon, Chen Avin, Michal Koucký, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. 2011. Many random walks are faster than one. *Combin. Probab. Comput.* 20, 4 (2011), 481–502.

[5] Steve Alpern and Diane J Reyniers. 2002. Spatial dispersion as a dynamic coordination problem. *Theory and decision* 53, 1 (2002), 29–59.

[6] Amine Asselah and Alexandre Gaudillière. 2013. From logarithmic to subdiffusive polynomial fluctuations for internal DLA and related growth models. *Ann. Probab.* 41, 3A (2013), 1115–1159.

[7] Amine Asselah and Alexandre Gaudillière. 2013. Sublogarithmic fluctuations for internal DLA. *Ann. Probab.* 41, 3A (2013), 1160–1179.

[8] Amine Asselah and Alexandre Gaudillière. 2014. Lower bounds on fluctuations for internal DLA. *Probab. Theory Related Fields* 158, 1-2 (2014), 39–53.

[9] Chen Avin, Michal Koucký, and Zvi Lotker. 2018. Cover time and mixing time of random walks on dynamic graphs. *Random Structures & Algorithms* 52, 4 (2018), 576–596.

[10] Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul W. Goldberg, Zengjian Hu, and Russell A. Martin. 2007. Distributed Selfish Load Balancing. *SIAM J. Comput.* 37, 4 (2007), 1163–1181.

[11] Petra Berenbrink, Martin Hoefer, and Thomas Sauerwald. 2014. Distributed Selfish Load Balancing on Networks. *ACM Trans. Algorithms* 11, 1 (2014), 2:1–2:29.

[12] Anders Björner, László Lovász, and Peter W. Shor. 1991. Chip-firing games on graphs. *European J. Combin.* 12, 4 (1991), 283–291.

[13] Sébastien Blachère and Sara Brofferio. 2007. Internal diffusion limited aggregation on discrete groups having exponential growth. *Probab. Theory Related Fields* 137, 3-4 (2007), 323–343.

[14] Paul Bogdan, Thomas Sauerwald, Alexandre Stauffer, and He Sun. 2012. Balls in bins via local search. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA, 16–34.

[15] Charlotte Brennan, J Kariv, and Arnold Knopfmacher. 2015. Longest Waiting Time in the Coupon Collectors Problem. *British Journal of Mathematics & Computer Science* 8 (01 2015), 330–336.

[16] Elisabetta Candellero, Shirshendu Ganguly, Christopher Hoffman, and Lionel Levine. 2017. Oil and water: a two-type internal aggregation model. *Ann. Probab.* 45, 6A (2017), 4019–4070.

[17] Colin Cooper, Alan Frieze, and Tomasz Radzik. 2010. Multiple random walks in random regular graphs. *SIAM J. Discrete Math.* 23, 4 (2010), 1738–1761.

[18] Michael Damron, Janko Gravner, Matthew Junge, Hanbaek Lyu, and David Sivakoff. 2017. Parking on transitive unimodular graphs. *Preprint* arXiv:1710.10529 (2017).

[19] P. Diaconis and W. Fulton. 1991. A growth model, a game, an algebra, Lagrange inversion, and characteristic classes. *Rend. Sem. Mat. Univ. Politec. Torino* 49, 1 (1991), 95–119 (1993). Commutative algebra and algebraic geometry, II (Italian) (Turin, 1990).

[20] Hugo Duminil-Copin, Itai Benjamini, Gady Kozma, and Cyrille Lucas. 2017. Internal Diffusion-Limited aggregation with uniform starting points. *Preprint* arxiv:1707.03241 (2017).

[21] Hugo Duminil-Copin, Cyrille Lucas, Ariel Yadin, and Amir Yehudayoff. 2013. Containing internal diffusion limited aggregation. *Electron. Commun. Probab.* 18 (2013), no. 50, 8.

[22] Tobias Friedrich and Lionel Levine. 2013. Fast simulation of large-scale growth models. *Random Structures Algorithms* 42, 2 (2013), 185–213.

[23] Christina Goldschmidt and Michał Przykucki. 2016. Parking on a random tree. *Preprint* arXiv:1610.08786 (2016).

[24] Wilfried Huss. 2008. Internal diffusion-limited aggregation on non-amenable graphs. *Electron. Commun. Probab.* 13 (2008), 272–279.

[25] Wilfried Huss and Ecaterina Sava. 2012. Internal aggregation models on comb lattices. *Electron. J. Probab.* 17 (2012), no. 30, 21.

[26] David Jerison, Lionel Levine, and Scott Sheffield. 2012. Logarithmic fluctuations for internal DLA. *J. Amer. Math. Soc.* 25, 1 (2012), 271–301.

[27] David Jerison, Lionel Levine, and Scott Sheffield. 2013. Internal DLA in higher dimensions. *Electron. J. Probab.* 18 (2013), No. 98, 14.

[28] Gregory F. Lawler. 1995. Subdiffusive fluctuations for internal diffusion limited aggregation. *Ann. Probab.* 23, 1 (1995), 71–86.

[29] Gregory F. Lawler, Maury Bramson, and David Griffeath. 1992. Internal diffusion limited aggregation. *Ann. Probab.* 20, 4 (1992), 2117–2140.

[30] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. 2009. *Markov chains and mixing times*. American Mathematical Society, Providence, RI. xviii+371 pages. With a chapter by James G. Propp and David B. Wilson.

[31] Lionel Levine and Yuval Peres. 2017. Laplacian growth, sandpiles, and scaling limits. *Bull. Amer. Math. Soc. (N.S.)* 54, 3 (2017), 355–382.

[32] Lionel Levine and Yuval Peres. 2017. Laplacian growth, sandpiles, and scaling limits. *Bull. Amer. Math. Soc. (N.S.)* 54, 3 (2017), 355–382.

[33] Lionel Levine and Vittoria Silvestri. 2018. How long does it take for Internal DLA to forget its initial profile? *Preprint* arXiv:1801.0853 (2018).

[34] László Lovász. 1996. Random walks on graphs: a survey. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*. Bolyai Soc. Math. Stud., Vol. 2. János Bolyai Math. Soc., Budapest, 353–397.

[35] Cyrille Lucas. 2014. The limiting shape for drifted internal diffusion limited aggregation is a true heat ball. *Probab. Theory Related Fields* 159, 1-2 (2014), 197–235.

[36] P. Meakin and J. M. Deutch. 1986. The formation of surfaces by diffusion-limited annihilation. *J Chem Phys* 85 (1986), 2320 – 2325.

[37] Cristopher Moore and Jonathan Machta. 2000. Internal diffusion-limited aggregation: parallel algorithms and complexity. *J. Statist. Phys.* 99, 3-4 (2000), 661–690.

[38] Yuval Peres and Perla Sousi. 2015. Mixing times are hitting times of large sets. *J. Theoret. Probab.* 28, 2 (2015), 488–519.

[39] Olivier Raimond and Bruno Schapira. 2011. Internal DLA generated by cookie random walks on $\mathbb{Z}$. *Electron. Commun. Probab.* 16 (2011), 482–490.

[40] Nicolás Rivera, Alexandre Stauffer, Thomas Sauerwald, and John Sylvester. 2018. The dispersion time of random walks on finite graphs. *Preprint* arXiv:1808.09219 (2018).

[41] Eric Shellef. 2010. IDLA on the supercritical percolation cluster. *Electron. J. Probab.* 15 (2010), no. 24, 723–740.

[42] Alexandre Stauffer and Lorenzo Taggi. 2018. Critical density of activated random walks on transitive graphs. *Ann. Probab.* 46, 4 (07 2018), 2190–2220.

[43] Debleena Thacker and Stanislav Volkov. 2018. Border aggregation model. *Ann. Appl. Probab.* 28, 3 (2018), 1604–1633.