Structural and computational depth of diffusion-limited aggregation

D. Tillberg and J. Machta*
Department of Physics, University of Massachusetts, Amherst, Massachusetts 01003, USA
(Received 30 September 2003; published 28 May 2004)

Diffusion-limited aggregation (DLA) is studied from the perspective of computational complexity. A parallel algorithm is exhibited that requires a number of steps that scales as the depth of the tree defined by the cluster. The existence of this algorithm suggests a connection between a fundamental computational and structural property of DLA.

DOI: 10.1103/PhysRevE.69.051403 PACS number(s): 61.43.Hv, 82.40.Ck, 05.45.Df, 89.75.Kd

Diffusion-limited aggregation (DLA), introduced some 20 years ago by Witten and Sander [1], is a model of pattern formation and an example of self-organized criticality. The dynamical rules for DLA are simple but the patterns produced are complex and have thus far defied full theoretical understanding; though there has been recent progress for two-dimensional DLA [2,3]. DLA models a number of physical systems including electrodeposition, fluid flow in porous media, and the growth of bacterial colonies [4].

The stochastic growth rules for DLA can be couched in terms of random-walk dynamics. In the present work we consider two-dimensional diffusion-limited aggregates composed of circular particles of unit diameter. The initial condition is a single seed particle at the origin, $\mathbf{r}_0=0$. The growth of the aggregate occurs one particle at a time and proceeds by starting the first particle at a random position on the "birth circle," $r_1(0)=r_b>1$. This particle does a random walk, $\mathbf{w}_1(t)$, until it either drifts out to the "death circle," $r_1>r_d\gg r_b$, or comes in contact with the particle at the origin, $r_1=1$. If it contacts the particle at the origin, it sticks and the aggregate grows. If it reaches the death circle, it is reborn at a random point on the birth circle and the process is repeated until the particle sticks. After n steps of this construction, the aggregate consists of n connected particles contained in a circle of radius R_n. On the nth step, a particle is launched on the birth circle, $r_b>R_n+1$, and follows a random walk $\mathbf{w}_n(t)$ until it sticks on one of the existing particles in the aggregate at sticking position $\mathbf{r}_n$ such that there is a $k<n$ and $|\mathbf{r}_n-\mathbf{r}_k|=1$.

DLA is believed to form fractal clusters such that $R_N \sim N^{1/d_f}$, where R_N is the average radius of a cluster of N particles and d_f is the fractal dimension, estimated by numerical simulation to be 1.715 ± 0.004 [5]. DLA clusters define a graph where the nodes are the particles and the edges are the contacts between particles. Since each particle sticks to a single predecessor, this graph is a tree rooted at the origin. One of the quantities that we will be interested in is the *structural depth* $\mathcal{D}_s$ of this tree, defined as the length of the path from the root to the outermost leaf. Numerical simulations [6] support the conjecture that the structural depth grows linearly in the radius of the cluster, $\mathcal{D}_s \sim R$. The growth mechanism of DLA strongly favors adding particles at the outermost tips of the cluster and thus exerts a tension that tends to make the growth radial. Figure 1(a) color codes particles in a cluster according to their distance to the seed particle along the tree, sometimes referred to as the chemical distance to the origin. This figure reveals the close relation between chemical distance and Euclidean distance from the origin.

In addition to its structural properties DLA has computational properties that help elucidate its complexity. The growth rules for DLA and the resulting patterns suggest that DLA has a great deal of *history dependence*. The construction of a DLA cluster requires a long sequence of steps and some random choices made early in the growth process are frozen in and have a large impact on the structure of the aggregate thereafter. Indeed, the stochastic history dependence present in DLA can be considered a marker for complexity. Biological evolution, the ultimate process for generating complexity, also has the feature that the present state emerges from a long sequence of stochastic steps and that accidents occurring in early epochs are frozen in and create the ground rules for later epochs.

The intuitive notion of history dependence in stochastic models such as DLA can be formalized in the context of computational complexity theory. The idea that the length of a history can be measured in the framework of computation was introduced by Bennett [7,8] though our treatment differs by emphasizing parallel rather than sequential computation and ensembles rather than individuals. We define the *computational depth* $\mathcal{D}_c$ of statistical physics models such as DLA as the minimum number of parallel computational steps needed to generate a typical system state of the model. Both statistical physics and computational complexity theory are concerned with scaling properties, for example, in the case of DLA, we are interested in the leading large N behavior of R, $\mathcal{D}_s$, and $\mathcal{D}_c$.

The notion of computational depth requires that we specify the parallel computer used to simulate the system. Computational complexity theory [9,10] provides a standard, idealized model of parallel computation called the parallel random access machine or PRAM. The PRAM consists of many simple processors all connected to a single global memory. All processors run the same program synchronously, though each processor has a distinct label so that the program can direct different processors to do different calculations. In a single computational step, each processor carries out an elementary logical or arithmetic operation and ex-

*Electronic address: machta@physics.umass.edu

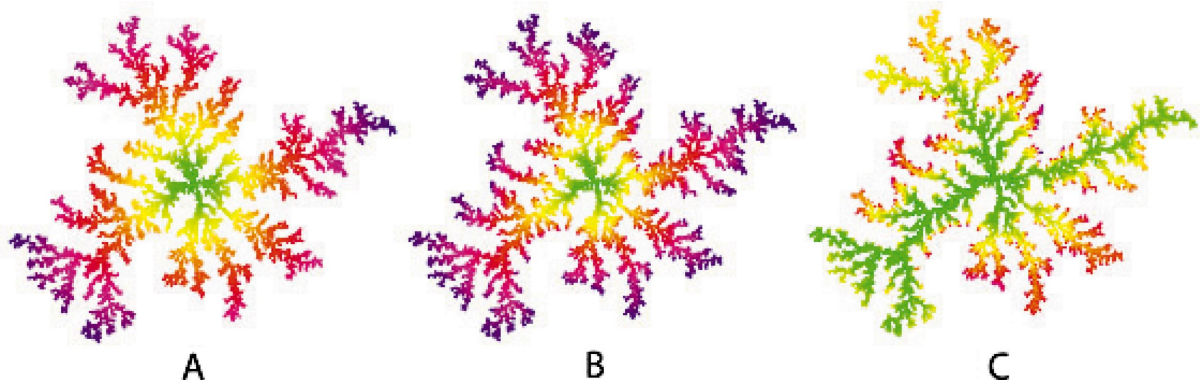

FIG. 1. (Color) Color-coded pictures of the same cluster. (a) The chemical distance to the origin; (b) the parallel step on which the particle first becomes semisecure at the correct position; (c) the deficit. Green (light) represents the lowest values and blue (dark) the highest.

changes information with the global memory. Since we are considering a stochastic system, random numbers are needed and we assume a sufficient supply is stored in the global memory before the simulation begins. The PRAM is a massively parallel model of computation where the number of processors and memory cells are allowed to grow *polynomially* (i.e., as a power) in the size of the system, in our case the number of particles N. Since any processor in the PRAM can communicate with any memory element in a single time step, the PRAM is not a realistic, scalable model of parallel computation and the considerations of this paper are not intended to provide a practical means of simulating DLA. Instead, the goal is to elucidate whether assembling a DLA cluster requires a long sequence of steps. The PRAM is one of many equivalent models of parallel computation and the scaling behavior for $\mathcal{D}_c$ reflects an intrinsic property of DLA rather than a particular strength or weakness of the PRAM. With this preamble we can now define computational depth as the average number of steps needed to construct a system state on a PRAM using the fastest parallel algorithm.

The primary objective of this paper is to show that $\mathcal{D}_c \lesssim \mathcal{D}_s$ and to motivate the conjecture that in fact $\mathcal{D}_c \sim \mathcal{D}_s$ [17]. Previous work has placed upper bounds on the computational depth of various pattern formation processes in statistical physics. For example, it has been shown that the clusters formed by the Bak-Sneppen model [11], internal DLA [12], invasion percolation, the Eden model, and ballistic deposition [13] all have *polylog* depth $\mathcal{D}_c \lesssim \log^{O(1)} N$, where N is the number of degrees of freedom of the system. These results show that these models do not have a strong intrinsic history dependence since constructing typical states can be carried out in a small number of parallel steps. These models generate trees whose structural depth greatly exceeds the computational depth, $\mathcal{D}_c \ll \mathcal{D}_s$. The situation is more complicated for DLA. It has been shown [14] that random-walk dynamics for DLA defines a **P**-complete problem, which strongly suggests that there is no polylog depth parallel construction for DLA, though it does not rule out a better power-law than the linear scaling with N of the conventional, one walk at a time growth rule. The question posed here is what is the best power-law scaling for a parallel simulation of DLA. Previous work [15,2] showed that $\mathcal{D}_c \lesssim N^{0.74} \sim R^{1.26}$. Here we improve that bound by exhibiting a faster parallel algorithm than the one described in Ref. [15].

The parallel algorithm for DLA proposed here assembles the cluster iteratively. On each step, every particle is moved to a tentative position so that the true cluster is approximated with increasing fidelity. Before the iterative process is begun, an ordered list of N sufficiently long random-walk trajectories $\{\mathbf{w}_i(t) | i = 1, \ldots, N\}$ is generated and stored in memory [18]. The cluster generated by the parallel algorithm is the same as the cluster that would result using these random-walk trajectories and the standard sequential rules for DLA.

In each step of the parallel algorithm, all particles are moved along their trajectories to tentative sticking points defined by a temporary cluster called the *semisecure* cluster. On the first step, the semisecure cluster consists only of the seed particle and every other particle sticks at the point where it first contacts the seed particle. For large N, the cluster after the first step will be nearly a disk of radius $3/2$ composed of the seed particle surrounded by other particles attached to the seed at the point they first touch it. The semisecure cluster for the second step consists of the seed particle and all particles whose path to the seed did not cross the sticking point of any lower-numbered particle. For $d=2$, the semisecure cluster for the second step consists of the seed and no more than six nonoverlapping particles touching the seed site. On the second step, every particle is moved independently along its trajectory until it first contacts this semisecure cluster.

The semisecure cluster on each parallel step is a template for growth and contains the subset of particle locations on a given step that are not obviously incorrect. A particle is not semisecure if it passes through a predecessor particle on the path to its sticking point. Semisecure particles are not necessarily at their final locations—it is possible that a particle on a given step may be in the semisecure cluster in one location but sticks in a different location on a later step. However, a semisecure particle such that all its predecessor particles are semisecure is in its final location.

For the mth step in the algorithm, we have a semisecure cluster defined by $N^{(m)}$ particles at locations $\{\mathbf{s}_j^{(m)} | j \in S^{(m)}\}$, where $S^{(m)}$ is the set of semisecure particle indices at step m and $\mathbf{s}_j^{(m)}$ is the location of semisecure particle j on step m. Each particle (semisecure or not) is then moved to a location

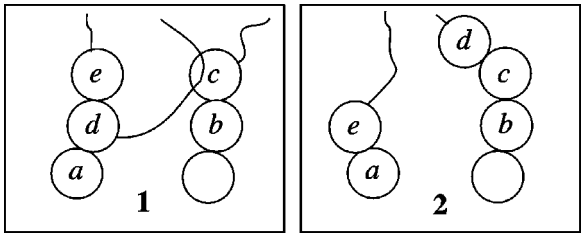

FIG. 2. A sequence of two parallel steps that increases the deficit of particle *e* and all particles that later connect to it. Suppose that for the step shown in panel 1, all particles but *c* are semisecure and that *b* has just become semisecure and *e* has been semisecure in its location for the past *k* steps. For the step shown in panel 2, *d* and *e* are not semisecure, *d* due to interference with *c* and *e* because it is no longer connected to the origin. In step shown in panel 2, *e* and *d* become semisecure in new locations and *e*'s deficit is increased by $k+2$.

based on the template provided by this semisecure cluster. The sticking point of a particle is determined by independently moving it along its trajectory until it first contacts a semisecure particle location with an index lower than its own. The sticking point $\mathbf{r}_i^{(m)}$ of particle i at step m is the location along its trajectory at the least time τ_i^m such that there is contact with a lower-numbered semisecure particle; $\mathbf{r}_i^{(m)}=\mathbf{w}_i(\tau_i^{(m)})$ such that there is a $j\in S^{(m)}$, $j<i$, and $|\mathbf{w}_i(\tau_i^{(m)})-\mathbf{s}_j^{(m)}|=1$ and for all $t<\tau_i^{(m)}$ and $k\in S^{(m)}$, $|\mathbf{w}_i(t)-\mathbf{s}_k^{(m)}|>1$.

After particles are moved to their sticking points, the current semisecure cluster is discarded and the locations of the sticking points are used to determine the next semisecure cluster. Each particle is categorized as "semisecure" or "not semisecure." A particle i is semisecure for step $m+1$ if the path to its sticking point in step m does not intersect the sticking point of any predecessor particle on step m. More formally, $i\in S^{(m+1)}$ and $\mathbf{s}_i^{(m+1)}=\mathbf{r}_i^{(m)}$, if for all $j<i$ and all $t<\tau_i^m$, $|\mathbf{w}_i(t)-\mathbf{r}_j^{(m)}|>1$. A semisecure particle such that all its predecessors are also semisecure is called *secure*. It is easy to see that a secure particle is in the correct location in the cluster and is guaranteed not to change its location on subsequent steps. The parallel algorithm has constructed a correct DLA cluster of size M for the given trajectories as soon as particle M is secure.

There is one additional rule for constructing the semisecure cluster. It may happen that the semisecure cluster defined by the above rules is a multiply connected "forest" composed of several "trees." In this case, only the tree rooted at the seed particle is retained and the remaining particles, not connected to the seed, are removed from the semisecure cluster. This rule removes particle indices from $S^{(m+1)}$ if they are not connected to the seed. An example of this phenomenon is shown in Fig. 2. Suppose the semisecure cluster for the first panel consists of *a*, *b*, *d*, and *e* and that *b* has just become semisecure in the previous step. If the index *c* is less than *d* then on the step shown in the first panel *d* intersects *c* and is no longer semisecure for panel 2. The result is that *e* is not connected to the seed and is removed from the semisecure cluster for panel 2, which consists of *a*, *b*, and *c*.

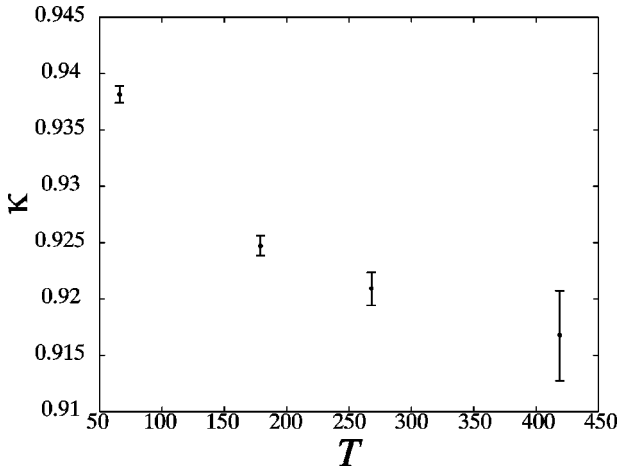

FIG. 3. The ratio $\kappa=\mathcal{D}_s/T$ of the average structural depth of the cluster divided by the number of parallel computational steps vs T.

As discussed in detail in Ref. [15], sufficiently long random-walk trajectories can be generated, sticking points of these trajectories to an existing cluster found and interferences among particles identified in polylog parallel time using polynomially many processors . Using a parallel algorithm for connected components [16], one can identify the tree connected to the origin in polylog time. Thus, the setup stage and each step of the algorithm require polylog time so that, up to logarithmic factors, the parallel time used by the algorithm scales as the number of steps.

How well does this algorithm perform? It is easy to see that at least one new particle becomes secure on each iteration so that the parallel algorithm requires no more than N iterations. In fact, the performance is much better than this weak bound. We have simulated the parallel algorithm on a sequential computer for N up to 20 000. Let T be the number of iterations required by the algorithm and let $\kappa=\mathcal{D}_s/T$ be the ratio of the structural depth to the number of iterations. Figure 3 shows $\kappa=\mathcal{D}_s/T$ plotted against T. The average is over several thousand clusters for the small sizes and 63 clusters for $N=20\,000$. Although κ decreases slightly as T and N increase, the data strongly suggest that κ approaches an asymptote near 0.9. Thus, up to logarithmic factors, the computational depth of DLA is no greater than the structural depth. In terms of a dynamic exponent for the algorithm, defined by $T\sim R^z$, the data suggest that $z=1$.

The algorithm of Ref. [15] is similar to the present algorithm except that the template used in step $m+1$ is the secure cluster from step m rather than the semisecure cluster. The semisecure cluster is larger than the secure cluster but, unlike the secure cluster, it contains particles at incorrect positions which must later be moved. Our numerical results show that the larger size of the semisecure cluster more than compensates for the errors in the semisecure cluster. The dynamic exponent of the "secure" algorithm of Ref. [15] is 1.26 compared to 1 for the "semisecure" algorithm described here.

If κ is indeed asymptotically near 1, it implies that on most steps of the parallel algorithm, one new level is added to the tree defined by the cluster. It is obvious that the algorithm cannot add more than one level to the tree in one parallel step so $\kappa\leqslant 1$. It is instructive to consider the way

that T becomes larger than the structural depth. After each parallel step, every semisecure particle can be assigned a *deficit*. A particle's deficit is defined as the difference between the step on which it most recently became semisecure at its current location and its chemical distance from the origin. Figure 2 shows how the deficit of a particle can increase by an arbitrary amount. The two panels show two successive parallel steps. For the step shown in panel 1, suppose all particles except c are semisecure and particle b *first* becomes semisecure for this step. Trajectories and sticking points for particles c, d, and e are shown. The ordering for the particles is assumed to be $b<c<d<e$ and $a<d$. Suppose that e had become semisecure at its current location k steps before the step shown in panel 1. In panel 1, c interferes with d so d and e are no longer semisecure for the step shown in panel 2. For the step after panel 2 all particles are semisecure in locations shown in panel 2. The deficit of e after panel 2 is increased by $k+2$ over what it was before panel 1 since its chemical distance is decreased by 1 and it has become semisecure in a new location $k+1$ steps later. When the algorithm is finished and all particles are secure, deficits along branches of the tree are a nondecreasing function of chemical distance. The maximum, taken over leaves of the tree, of the sum of the deficit and chemical distance to the origin gives the running time of the algorithm in parallel steps.

Figure 1 shows three images of the same cluster colored to reveal different properties. This cluster consists of 20 000 particles, it has a structural depth of 439 and requires 479 steps to assemble. In Fig. 1(a) the particles are colored according to their structural depth (chemical distance from the origin). Note that the contours of equal structural depth are nearly circular. In Fig. 1(b) particles are colored according to the parallel step on which they first become semisecure at their final locations. This image reveals that growth generated by the parallel algorithm conforms more to the shape of the branches of the cluster. In Fig. 1(c) particles are colored according to their deficits, Fig. 1(c) shows the difference between the quantities in Figs. 1(b) and 1(a) and it reveals that the dominant branches have small deficits but that large deficits can develop in branches which are less robust and screened by the dominant branches. The reason for this tendency can be seen in the example of Fig. 2 where a theft from one branch to another increases the deficit on the branch that loses the particle.

The algorithm proposed here assembles DLA clusters by adding nearly one level of structural depth in each parallel step. Our intuition is that it is not possible to generate DLA clusters in substantially fewer steps than the structural depth of the cluster so that $\mathcal{D}_c \sim \mathcal{D}_s$. This conjecture links fundamental structural and computational properties of DLA. Like other lower bounds in computational complexity theory, this conjecture is likely to be difficult to prove.

This work was supported by NSF Grants DMR-9978233 and DMR-0242402.

[1] T. A. Witten and L. M. Sander, Phys. Rev. Lett. **47**, 1400 (1981).
[2] F. Barra, B. Davidovitch, and I. Procaccia, Phys. Rev. E **65**, 046144 (2002).
[3] R. C. Ball and E. Somfai, Phys. Rev. E **67**, 021401 (2003).
[4] T. Vicsek, *Fractal Growth Phenomena* (World Scientific, Singapore, 1992).
[5] S. Tolman and P. Meakin, Phys. Rev. A **40**, 428 (1989).
[6] P. Meakin, I. Majid, S. Havlin, and H. E. Stanley, J. Phys. A **17**, L975 (1984).
[7] C. H. Bennett, in *The Universal Turing Machine—A Half-Century Survey*, edited by R. Herken (Oxford University Press, New York, 1988), pp. 227–257.
[8] C. H. Bennett, in *Complexity, Entropy and the Physics of Information*, edited by W. H. Zurek, SFI Studies in the Sciences of Complexity Vol. 7 (Addison-Wesley, Reading, MA, 1990), p. 137.
[9] C. H. Papadimitriou, *Computational Complexity* (Addison-Wesley, Reading, MA, 1994).
[10] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to Parallel Computation: P-completeness Theory* (Oxford University Press, New York, 1995).
[11] J. Machta and X.-N. Li, Physica A **300**, 245 (2001).
[12] C. Moore and J. Machta, J. Stat. Phys. **99**, 661 (2000).
[13] J. Machta and R. Greenlaw, J. Stat. Phys. **77**, 755 (1994).
[14] J. Machta and R. Greenlaw, J. Stat. Phys. **82**, 1299 (1996).
[15] K. Moriarty, J. Machta, and R. Greenlaw, Phys. Rev. E **55**, 6211 (1997).
[16] A. Gibbons and W. Rytter, *Efficient Parallel Algorithms* (Cambridge University Press, Cambridge, England 1988).
[17] The symbols $\lesssim$ and $\sim$ are to be interpreted, respectively, as asymptotic inequality and equality up to logarithmic factors, e.g., $N^3 \log^2 N \lesssim N^3$.
[18] Random walk trajectories may include several restarts from the birth circle each time the death circle is reached.