History of a Stochastic Growth Model

Murray Eden and Philippe Thévenaz

Office of Research Services
National Institutes of Health, Bethesda MD 20892-5766, USA

ABSTRACT

One of the earliest models of stochastic growth was originally developed for simulating the appearance of various biological patterns; in particular, bacterial colonies. Although it received little attention from biologists, some twenty years later it was adopted by crystallographers, solid state researchers, other physicists and chemists. Because of the model's flexibility it is being used by them after modifications appropriate to the application, in order to simulate their physical study objects under a variety of conditions. Only within the last few years has there been any interest in using this and similar digital models to represent the possible products of biological processes. It is also worth noting that aside from its relevance to probabilistically influenced pattern formation, the model has possible use in image processing for image compression and as an information-lossless way to code, regions, contours, or line segments.

Keywords: Stochastic algorithms, discrete growth processes, information-lossless image coding, Eden model

1. INTRODUCTION

Although the use of stochastic growth models to simulate many diverse phenomena has been developing rapidly within the last fifteen years, it had its origins more than forty five years ago. Turing published a paper in 1952 that considered a continuum one-dimensional growth in a ring of cells in which a random concentration fluctuation of some critical chemical component caused an instability in the linear growth of certain of the cells in the ring.[1] Turing also included the illustration of a two-dimensional simulation which resembled the pattern on a Holstein cow's hide, although he provided no details on the way in which the pattern was generated. Vold designed a simulation for colloidal aggregation[2] by considering a line of discrete elements and then "dropping" randomly distributed particles that adhered to the line or to particles that had been dropped earlier in the simulation. The models of Eden[3,4] were intended to simulate the growth of a colony of non-motile bacteria growing on a flat surface in a one-dimensional layer and are described below.

The early papers occasioned relatively little interest or exploration of such models. First of all, these models were considered to be overly simplistic and bore little resemblance to the phenomena they were intended to model. Secondly, the speed and power of the computers of the fifties did not permit the development of sufficiently large simulations and statistically significant samples*.

Modern computational power and ease of programming has made it possible to develop much more complex models that more nearly resemble the physical or biological objects they simulate. Nevertheless, they are to a large extent elaborations of the simpler ones. A typical example, one of the rare applications to biology, is the work of Drasdo *et al.*[5] They developed a hierarchy of stochastic models in order to simulate a monolayer of epithelial cells. The models included factors such as cell migration, rotation, growth, division, among others and applied them to a global description of birth-death processes as well as to very specific features of intracellular dynamics. More recently, Savakis and Maggelakis discussed the use of the Eden model as well as the diffusion limited aggregation process (DLA) in an attempt to simulate wound healing and tissue regeneration.[6]

The proliferation of uses for discrete stochastic models continues. They have been applied to two-component spreading phenomena[7] such as occur in liquid invasion in porous media, grain coalescence in alloys, fracture propagation in solids, damage spreading in dielectrics, electrical or neural networks and virus propagation, growth of towns,[8] models of river network evolution,[9] surface diffusion[10] as in paper towel wetting, sedimentation, vapor

*The Eden model was first run in 1953-1954 on the von Neumann-Goldstine-Bigelow designed computer at the Institute for Advanced Study, Princeton, New Jersey. "Colonies" of about twenty cells were displayed as rectangular holes in a Hollerith punch card.

deposition, impurities in crystals and molecular beam epitaxy to name some of the more common applications. Percolation is a related stochastic process which has had considerable application to physical and chemical processes.[11] Its general mathematical properties continue to be an area of active research.[12]

In this report we review a few of the algorithms beginning with what is regarded to be the simplest and consider some of its variations. Three of these algorithms have been described and analyzed[13] with the initial state being a straight line of varying (discrete) length l. We note that modifications in the algorithms that on first sight might be considered trivial, may have dramatic effect on the objects generated. Several of the parameters present in any such simulation are discussed. Their effects are illustrated in a set of experimental computer-generated patterns. Finally, an account is presented of the use of the similar algorithms in coding images in an information lossless manner.

2. GROWTH ALGORITHMS

Consider a graph Γ given by its (possibly infinite) symmetric adjacency matrix $A = A^{\mathsf{T}} = [a_{ij}]$, where $a_{ij} = 1$ indicates that nodes N_i and N_j are connected (neighbors,) and where $a_{ij} = 0$ indicates that they are not. By convention, we exclude self-adjacency by setting $a_{ii} = 0$. Let us partition the corresponding set of nodes $\Omega = \{N_i\}$ into a set Λ of labeled nodes and its complement $\bar{\Lambda} = \Omega \backslash \Lambda$. We impose $\Lambda \neq \emptyset$; in other words, there should be at least one labeled node. Our goal is to grow the set Λ at the expense of the set $\bar{\Lambda}$, while enforcing connectivity: each added node has to be adjacent to at least one labeled node, even though neither the graph Γ nor the initial set Λ need be connected themselves. Let $\Theta = \left\{ N_i \in \Omega \,\middle|\, (N_i \in \Lambda) \wedge \left(\sum_{N_j \in \bar{\Lambda}} a_{ij} > 0\right) \right\}$ be the set of all those labeled nodes that have at least one unlabeled neighbor, and let $\Phi = \left\{ N_i \in \Omega \,\middle|\, (N_i \in \bar{\Lambda}) \wedge \left(\sum_{N_j \in \Lambda} a_{ij} > 0\right) \right\}$ be the set of all those unlabeled nodes that have at least one labeled neighbor. These two sets Θ and Φ may be understood to be the internal and the external side respectively of the frontier between Λ and $\bar{\Lambda}$. The growth algorithm presented in Reference 3 labels nodes iteratively as follows: according to some prescribed stochastic rule, select a subset $\Xi = \{N_i\} \subseteq \Phi$, label the nodes of this subset (which modifies Λ and $\bar{\Lambda}$,) update Θ and Φ, and start over. The algorithm ends when $\Theta = \Phi = \emptyset$.

Algorithm A. The subset $\Xi = \{N_i\} \subseteq \Phi$ is a singleton ($\mathrm{card}(\Xi) = 1$.) Its associated probability is $P_A(N_i) = 1/\mathrm{card}(\Phi)$.

Algorithm B. The subset Ξ is the singleton $\{N_i\}$. Its associated probability $P_B(N_i) = \sum_{N_j \in \Theta} a_{ij} / \sum_{N_k \in \Phi} \sum_{N_l \in \Theta} a_{kl}$ is proportional to the number of its labeled adjacent nodes.

The two previous algorithms are built on the principle of achieving the expansion of Λ by reducing the external frontier Φ by one element at a time. The next four algorithms make use of the dual approach, achieving the reduction of $\bar{\Lambda}$ by expanding the internal frontier Θ by one or more elements.

Algorithm C. First, pick a labeled node $N_j \in \Theta$ with equal probability $P_C(N_j) = 1/\mathrm{card}(\Theta)$. The subset $\Xi = \{N_i \in \Omega \,|\, (a_{ij} = 1) \wedge (N_i \in \Phi)\}$ contains all those neighbors of N_j that are unlabeled.

Algorithm D. First, pick a labeled node $N_j \in \Theta$ with probability $P_C(N_j) = 1/\mathrm{card}(\Theta)$. The subset Ξ is a singleton $\{N_i\}$ that is chosen from among the unlabeled neighbors of N_j with equal probability. Since N_j may not be the only labeled node adjacent to N_i, the probability for the node N_i to be selected is $P_D(N_i) = \sum_{N_j \in \Theta} \left(a_{ij}\, P_C(N_j) / \sum_{N_k \in \Phi} a_{jk}\right)$.

Algorithm E. First, pick a labeled node $N_j \in \Theta$ with a probability $P_E(N_j) = \sum_{N_k \in \Phi} a_{jk} / \sum_{N_l \in \Theta} \sum_{N_m \in \Phi} a_{lm}$ that is proportional to the number of its unlabeled adjacent nodes. The subset $\Xi = \{N_i \in \Omega \,|\, (a_{ij} = 1) \wedge (N_i \in \Phi)\}$ contains all those neighbors of N_j that are unlabeled.

Algorithm F. First, pick a labeled node $N_j \in \Theta$ with a probability $P_E(N_j) = \sum_{N_k \in \Phi} a_{jk} / \sum_{N_l \in \Theta} \sum_{N_m \in \Phi} a_{lm}$ that is proportional to the number of its unlabeled adjacent nodes. The subset Ξ is a singleton $\{N_i\}$ chosen from among the neighbors of N_j with equal probability. Since N_j is not the only node to which N_i may be adjacent, the probability for the node N_i to be selected is $P_F(N_i) = \sum_{N_j \in \Theta} \left(a_{ij}\, P_E(N_j) / \sum_{N_k \in \Phi} a_{jk}\right)$. Note that this algorithm is stochastically equivalent to Algorithm B, which is easy to verify by the substitution of the expression for P_E into P_F.

The six previous algorithms depend neither on any special structure of the underlying graph Γ nor on the space dimension. For the next algorithms, we introduce an additional requirement according to which the graph Γ represents a regular lattice. In two dimensions, we typically consider different adjacency relations, specifying three, four, six or eight neighbors to any node (*e.g.*, as given by hexagonal or square lattices.) In three dimensions, we typically consider six, eighteen or twenty-six neighbors for a cubic lattice. We associate a separate list to each principal direction $\mathbf{v}$ in the lattice, and we distribute the initial set $\Phi = \bigcup_{\mathbf{v}} \Phi_{\mathbf{v}}$ into those directional lists in an arbitrary way (it need not necessarily be a partition.) During the growth process, we shall keep track of which directional relation is used and we shall update the relevant list. Note that we never recompute Φ, but we use the updated $\Phi_{\mathbf{v}}$ instead; also, the content of the set Θ is not explicitly relevant in the next four algorithms.

Algorithm G. First, pick at random a directional list $\Phi_{\mathbf{u}}$. One may or may not consider the length of the lists in this process (which might generate additional algorithms,) but an empty list will be rejected. Then, select the subset $\Xi \subseteq \Phi_{\mathbf{u}}$ as a singleton containing the node N_i that is at the tail of the list (first-in-last-out approach.) Label this node and remove it from $\Phi_{\mathbf{u}}$ and from all the other lists $\Phi_{\mathbf{v}}$ in which it appears. Taking directional considerations into account, update the lists $\Phi_{\mathbf{v}}$ by appending to their tail the unlabeled neighbors, if any, of the newly labeled node N_i. The algorithm ends when every directional list $\Phi_{\mathbf{v}}$ is empty.

Algorithm H. Same as Algorithm G, but the node N_i comes from the head of the list $\Phi_{\mathbf{u}}$ (first-in-first-out approach.) The probability of selecting a node N_i is null if N_i is not at the tail (Algorithm G) or the head (Algorithm H) of some list $\Phi_{\mathbf{v}}$, and is $P_{GH}(N_i) = r_i/V$ otherwise, where $V = \text{card}\left(\{\mathbf{v} \,|\, \mathbf{\Phi_v} \neq \emptyset\}\right)$ is the number of non-empty directional lists, and where r_i is a redundancy factor that counts the number of lists in which the same node N_i is present (at the tail or the head of the lists for Algorithms G and H, respectively.)

Algorithm I. Same as Algorithm G, but the node N_i comes from anywhere in the list $\Phi_{\mathbf{u}}$. The probability for a node N_i to be selected is $P_I(N_i) = (1/V)\ \sum_{\{\mathbf{v}|\mathbf{N_i}\in\mathbf{\Phi_v}\}} (1/\text{card}(\Phi_{\mathbf{v}}))$ in the case of equiprobable lists.

Algorithm J. Same as Algorithm I, but the probability of selecting a particular list is made proportional to its length. Algorithm J is stochastically equivalent to Algorithms B and F, provided there is an appropriate initial distribution $\Phi = \bigcup_{\mathbf{v}} \Phi_{\mathbf{v}}$.

One can also decompose the set Θ into directional lists. By contrast with Algorithms G, H, I and J, it is preferable (although not necessary) to consider an initial decomposition in which the lists $\Theta_{\mathbf{v}}$ form a partition of Θ. The algorithms we present now benefit from this property and ensure that it is maintained during growth.

Algorithm K. First, pick at random a list $\Theta_{\mathbf{u}}$. One may or may not consider the length of the lists in this process (which might generate additional algorithms,) but an empty list will be rejected. Then, pick a labeled node N_j, either at the tail of the list (Algorithm KG,) head of the list (Algorithm KH,) or anywhere in between (Algorithms KI and KJ.) The subset $\Xi = \{N_i \in \Omega \,|\, (a_{ij} = 1) \wedge (N_i \in \Phi)\}$ contains all those neighbors of N_j that are unlabeled, if any. Label them and append them to the tails of the relevant directional lists. Remove N_j from the list $\Theta_{\mathbf{u}}$. The algorithm ends when every directional list $\Theta_{\mathbf{v}}$ is empty.

Algorithm L. Algorithms LG, LH, LI and LJ are the same as Algorithms KG, KH, KI and KJ, respectively, but the subset Ξ is a singleton $\{N_i\}$ that is chosen with equal probability from among the unlabeled neighbors of N_j, if any. The node N_j is removed from the list $\Theta_{\mathbf{u}}$ only when the condition $\Xi = \emptyset$ is encountered. There exists an equivalence relation between Algorithm KJ and Algorithm C and also between Algorithm LJ and Algorithm D. We call these relations stochastic, in the sense that nodes have the same probability of being selected for labeling if one provides the appropriate initial distribution $\Theta = \bigcup_{\mathbf{v}} \Theta_{\mathbf{v}}$.

We propose to qualify Algorithms A to LJ as being *incremental* or *accretional*, according whether Ξ is necessarily a singleton or not, respectively. In addition, we qualify them as being *defensive* if the first algorithmic step is to pick a node from the internal frontier Θ between Λ and $\bar{\Lambda}$, and *offensive* if the first algorithmic step is to select a node from the external frontier Φ. Since the latter terminology is based on algorithmic considerations, two algorithms (*e.g.*, Algorithm B and Algorithm F) might be classified differently, even though they are stochastically equivalent. Yet another algorithmic classification scheme separates those algorithms that need no set of directional lists from

Table 1. Classification of some growth algorithms.

	Set-based			List-based	
	Incremental	Accretional		Incremental	Accretional
Offensive	A, B		Offensive	G, H, I, J	
Defensive	D, F	C, E	Defensive	LG, LH, LI, LJ	KG, KH, KI, KJ

	$\circ N1$				
$\circ W1$	$\bullet 1$	$\circ E1$	$\circ N6$		
$\circ W2$	$\bullet 2$	$\circ E2$	$\bullet 6$	$\circ E6$	
$\circ W3$	$\bullet 3$	$\bullet 4$	$\bullet 5$	$\circ E5$	
	$\circ S3$	$\circ S4$	$\circ S5$		

Figure 1. Initial configuration. The graph Γ is a 4-connected square lattice. The labeled nodes are $\{\bullet i\} \in \Lambda$, and some unlabeled nodes are $\{\circ Di\} \in \bar{\Lambda}$, where D refers to a direction from the labeled node $\bullet i$. We have here that $\Theta = \Lambda$ and $\Phi = \{\circ i\}$.

those that do. We name the former *set-based* and the latter *list-based*, respectively. According to this terminology, Table 1 presents the classification of the algorithms we have described above. Figure 1 illustrates some of them.

Let us examine the fate of some nodes in Figure 1, in the light of the set-based algorithms. For example, we have $P_A(\circ N6) = 1/12$, $P_B(\circ N6) = P_F(\circ N6) = 1/14$, $P_C(\circ N6) = 1/6$, $P_D(\circ N6) = 1/18$, $P_E(\circ N6) = 3/14$. The examination of another node yields $P_A(\circ E2) = 1/12$, $P_B(\circ E2) = P_F(\circ E2) = 3/14$, $P_C(\circ E2) = 1/2$, $P_D(\circ E2) = 2/9$ and $P_E(\circ E2) = 1/2$. Supposing that the node $\circ N6$ has been selected for labeling, the succeeding configuration is the same for all algorithms if they are incremental. It is given in Figure 2. The configuration for accretional algorithms is given in Figure 3. In the present case, it happens to be the same for all such algorithms because the unlabeled node $\circ N6$ had only one neighboring labeled node $\bullet 6$. Note that the latter becomes entirely surrounded by other labeled nodes, which means that we have now a strict inclusion relation $\Theta \subset \Lambda$, contrary to the configuration in Figure 1 where we had $\Theta = \Lambda$. It is obvious that at least one such neighbor free cell is created with each step in accretional algorithms.

	$\circ N1$		$\circ N7$		
$\circ W1$	$\bullet 1$	$\circ E1$	$\bullet 7$	$\circ E7$	
$\circ W2$	$\bullet 2$	$\circ E2$	$\bullet 6$	$\circ E6$	
$\circ W3$	$\bullet 3$	$\bullet 4$	$\bullet 5$	$\circ E5$	
	$\circ S3$	$\circ S4$	$\circ S5$		

Figure 2. Next configuration in the case of an incremental algorithm.

3. ALGORITHMIC MODIFICATIONS

Even though some of the growth algorithms are stochastically equivalent (*e.g.*, B = F = J, C = KJ and D = LJ,) their catalog is potentially endless because the choice of probability distributions is arbitrary. The catalog can be made even larger by taking a less local view of the graph Γ, considering adjacency relations that span more than

	$\circ N1$		$\circ N7$		
$\circ W1$	$\bullet 1$	$\circ E1$	$\bullet 7$	$\circ E7$	
$\circ W2$	$\bullet 2$	$\bullet 8$	$\bullet 6$	$\bullet 9$	$\circ E9$
$\circ W3$	$\bullet 3$	$\bullet 4$	$\bullet 5$	$\circ E5$	
	$\circ S3$	$\circ S4$	$\circ S5$		

Figure 3. Next configuration in the case of an accretional algorithm.

two nodes. Moreover, additional algorithms may arise with competitive growth, which appears when the initial set $\Lambda = \bigcup_n \Lambda_n$ is made of several disconnected sub-graphs that grow independently on the same underlying graph Γ. In this case, it is simplest to limit the interaction between the sub-graphs to the competition for space $\bar{\Lambda}$, but more complicated interactions might be considered as well, for example depending on the size of a connected Λ_n, or on the age of its nodes relative to those of its competitors. As examples of still more growth algorithms, we leave as an exercise to the interested reader the completion of Table 1 with accretional-offensive ones.

Anisotropic conditions of growth might simulate a gradient of chemical concentration, nutrient, electric charge, *etc.* This simulation may be achieved by assigning differing probabilities of growth to different directions, which is particularly easy to implement for list-based algorithms (for example Algorithms G to LJ.) We have included examples of such modifications in our experimental demonstrations.

Another algorithmic change that has had wide use will create "trees" rather than more-or-less compact clusters.[14,15] The simplest algorithmic way to grow trees assigns $P = 0$ to any unlabeled node already adjacent to two or more labeled nodes. The variety of options discussed above can be used in growing trees as well as clusters. Further complexity can be simulated by conditioning the probability assignments on factors such as age of a cell (the length of time it has been in a labeled state,) on the density of cells within a radius r, distance to an attractor (*e.g.*, to simulate phototropism,[16]) *etc.*

With few exceptions, growth models have been restricted to two dimensions. Nevertheless, Jullien and Botet[13] generated relatively small clusters in three and four dimensions on linear baselines of maximum extent $l = 48$ (a square $48 \times 48 = 2304$) and $l = 12$ (a cube with 1728 nodes). In order to model the development of the structure of slightly poisoned silica colloids Sintes *et al.*[17] developed clusters of $50,000$ cells in a cube with $l = 64$, on a tetragonal lattice.

To our knowledge, except for Sintes *et al.*,[17] planar clusters have been usually grown on a 4-connected square lattice; that is, neighbors adjacent to a given node N are the set of its nearest nodes, and the connections run through the sides of the square cell holding the node N. Obviously, growth on an 8-connected lattice may also be considered, where the additional connections run through the diagonals of the cells in addition to those running through the sides. Alternatively, we may consider two-dimensional growth on the hexagonal lattice of Figure 4. In this instance, each cell can be either 6-connected or 3-connected, in which case a third of the lattice nodes—shown in gray in Figure 4—doesn't belong to Ω.

We can grow the set Λ on a finite set Ω, while keeping a strictly regular structure for the lattice. We typically achieve this by considering that the geometry of our lattice is toroidal (a torus can be tiled with squares, and hexagons as well.) Initially, let the set Λ consist of a certain number of disconnected single nodes that are placed arbitrarily. A personalized label is attributed to each of these nodes and is propagated by the growth process. The same label is applied to each of the nodes' offspring. Because Ω is finite and connected, we can permit growth until we have $\bar{\Lambda} = \emptyset$. This construction produces results that are somewhat reminiscent of a section through a block of solid tissue or a section through a cluster of soap bubbles.

4. ALGORITHMIC EXPERIMENTS

We have generated a variety of patterns to illustrate the different outcomes. Although several statistical functions, notably, the "thickness" of the edge or the convex hull as it changes with area, have been estimated in a number

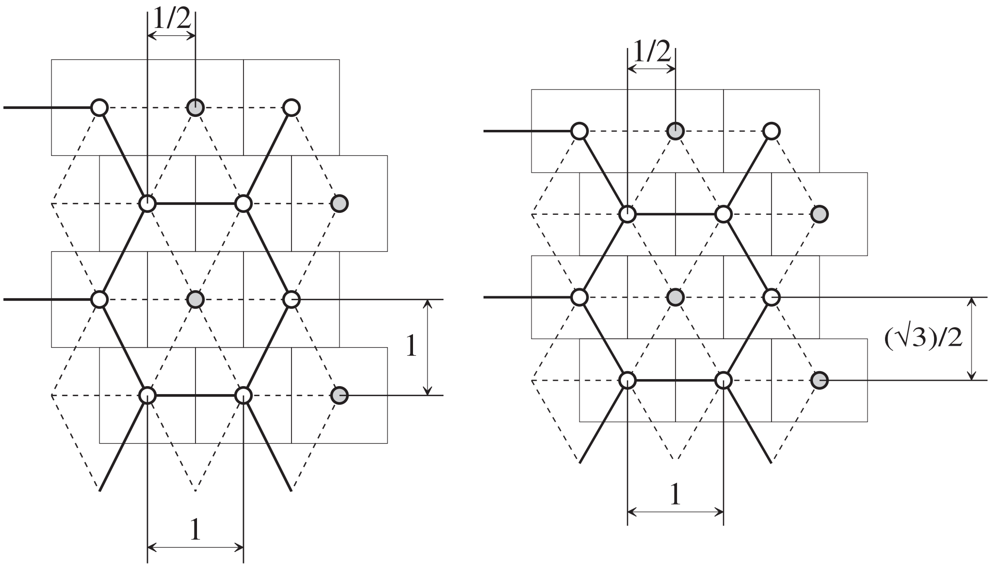


Figure 4. A geometrically correct hexagonal array for display purposes may be made by displacing each even numbered (horizontal) line of a square lattice by half the lattice spacing and by reducing the vertical distance between lattice lines to $\frac{1}{2}\sqrt{3}$.

of the papers cited, only a few qualitative judgments will be presented here. They clearly show that the resulting clusters are wildly different, as can be seen by comparing Figures 5, 6 and 7, for example.

The patterns were grown on regular lattices. The same value was used to seed the pseudo-random numbers in each case. The displays for the first set of clusters used 10 successive gray levels, each repeated for 10 successive labeling operations, permitting visual discrimination of successive rings of growth. For the last set of patterns that cover the lattice fully, a single gray level was assigned to each of the individual clusters. Unless mentioned otherwise, in trials of 4-connected or 6-connected sets equal probabilities were assigned to each direction. In the 8-connected case the nearest neighbors—those on the sides—were assigned probabilities $P_s = (2 - \sqrt{2})/4$, while $P_d = (\sqrt{2} - 1)/4$ was assigned to the diagonals, thus simulating isotropism in an Euclidean sense since we have $P_s/P_d = \sqrt{2}$.

In the case of a 4-connected square lattice, Figure 5 illustrates at left the incremental algorithm H. The accretional version of this algorithm (KH) is illustrated at the right of Figure 5. In this case, the set of labeled nodes takes a remarkably elongated form. In other trials, with other seeds for the pseudo-random number generator, the same elongation occurs; sometimes it extends in the other principal direction. In the 6-connected case, three-branched star patterns emerge, and the 8-connected case leads to + and × crosses. Both Algorithms H and KH are of the first-in-first-out type (FIFO.) The stochastic algorithm H produces results that are very similar to its deterministic version, because the latter would generate perfect diamonds. On the other hand either incremental or accretional first-in-last-out algorithms (FILO) generate a highly mobile pattern somewhat resembling Brownian motion. This is in sharp contrast with their deterministic version, which would produce a single (non-branched) trail that would explore Γ in a systematic way. Apart from the width of the trails, there is little obvious difference between the outcome of Algorithms G and KG, both shown in Figure 6 (note that in these two last cases, growth proceeds for only one out of the three initial labeled nodes, because the two others will be visited again only at the very end of the growth.) As can be seen by comparing Figures 5 and 6, the sets produced by the FIFO and the FILO ordering are strikingly different.

Comparisons of patterns resulting from the incremental-offensive algorithms I or J to those of their accretional-defensive version KI or KJ, show the same general behavior, as can be seen in Figures 7 and 8. The accretional patterns appear to have smoother borders, which was expected because connected clusters are added at each step instead of randomly separated single cells.

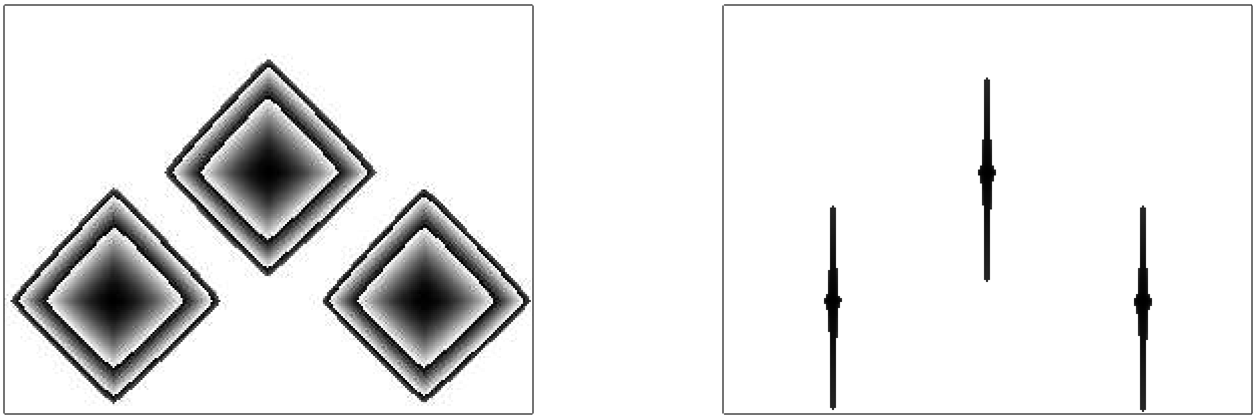


Figure 5. Illustration of Algorithm H (left) and Algorithm KH (right), 4-connected, initial Λ as three separate nodes, isotropic growth probabilities.

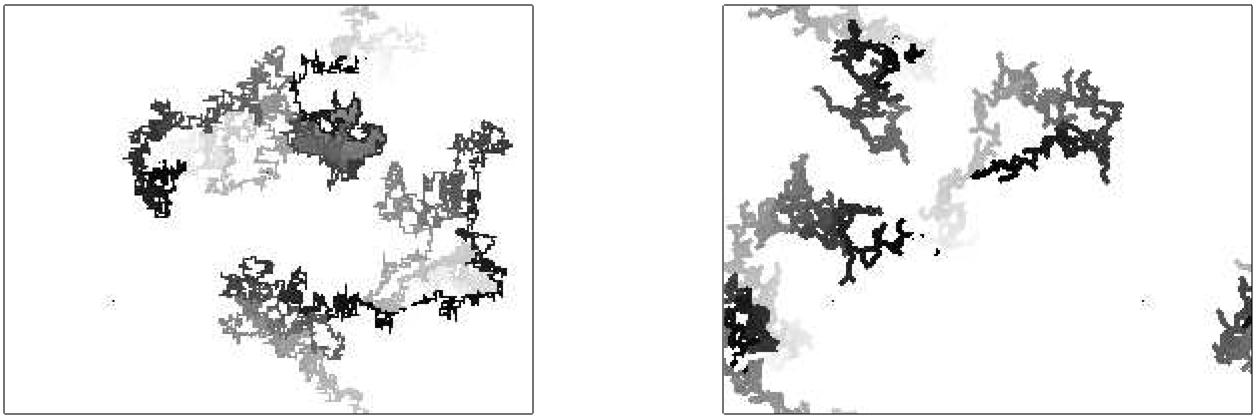
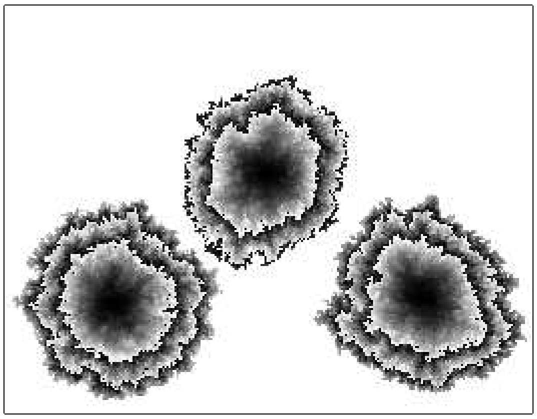


Figure 6. Illustration of Algorithm G (left) and Algorithm KG (right), 4-connected, initial Λ as three separate nodes, isotropic growth probabilities.

We modified slightly Algorithms I and KI by assigning the probability $P_v = 0.4$ to the lists responsible for vertical growth, and $P_h = 0.1$ for horizontal growth. We display the resulting patterns in Figure 9, where the difference between incremental and accretional growth is striking. The aspect ratio for the former is about 2.9, while the aspect ratio for the latter is about 15 for very much smaller clusters.

In Figure 10, we display 8-connected clusters under incremental-offensive growth with probabilities that are isotropic in an Euclidean sense. In Figure 11, we display 6-connected clusters under incremental-offensive and accretional-defensive growth. Once again, there is little qualitative difference except for the smoother border in the latter case, along with a less balanced size of the clusters.

We ran a series of experiments filling the lattice. Figure 12 illustrates raster-filling trials with five, eight, and sixteen kernel cells, respectively, using Algorithm I. The kernels (initial Λ) were placed in an approximate quincuncial configuration in the first two experiments and in a rectangular array in the last one. Clearly, the relative location of the initial kernels affects the appearance. It is worth noting that each of the five regions in Figure 12 (left) forms a more-or-less straight boundary with all the others. In Figure 12 (center) each region is bounded by exactly five others, again with reasonably straight boundaries. In Figure 12 (right) the mean number of abutting regions is six, but with considerable variation in abutters from four to eight. Boundaries between regions diagonally oriented one to the other are very short and seem to be local accidents of the growth process.



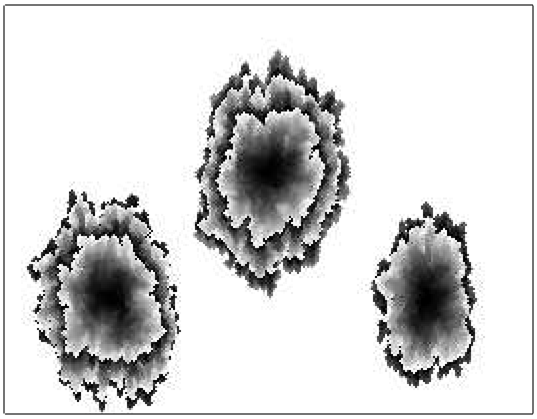
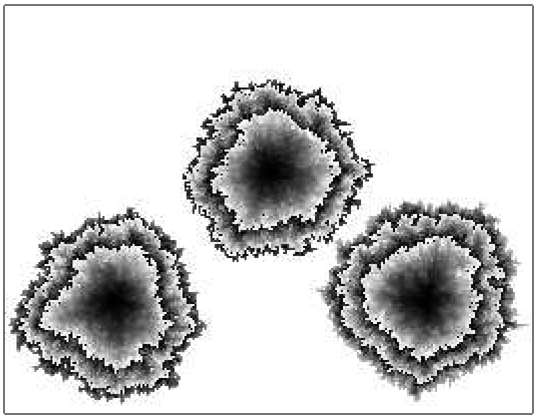


Figure 7. Illustration of Algorithm I (left) and Algorithm KI (right), 4-connected, initial Λ as three separate nodes, isotropic growth probabilities.



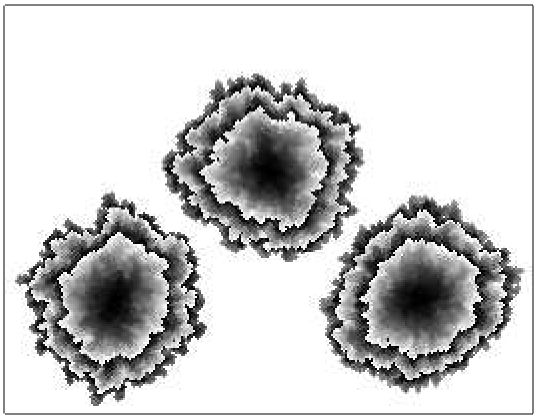


Figure 8. Illustration of Algorithm J (left) and Algorithm KJ (right), 4-connected, initial Λ as three separate nodes, isotropic growth probabilities.

5. USED AS A CODING ALGORITHM

Although the intent in the various algorithms and their modifications has been to represent stochastic processes, it is possible to use similar algorithms to code binary figures, especially if they are connected sets. Let ω represent the set of all nodes. Let $\Omega \subseteq \omega$ represent the set of nodes that build the figure. We associate to ω an n-connected lattice and to Ω a graph Γ on the lattice.

One way to code Ω starts by the choice of an arbitrary node $N_0 \in \Omega$. Initially, we let the set of labeled nodes be the singleton $\Lambda = \{N_0\}$. Then, we adopt some growth procedure that may be inspired by the FIFO or FILO algorithms G, H, KG, KH, LG or LH, but modified to become deterministic, for example by using a single list instead of multiple directional lists. According to the adopted procedure, we grow the set Λ on Γ and we generate a single new codeword each time a set Ξ of nodes is labeled (for obvious efficiency reasons, accretional algorithms would be preferred to incremental ones.) The codeword consists of of n bits that are determined by a membership function that assigns 1 if $N_i \in \Omega$, and 0 if $N_i \in \omega \backslash \Omega$, where $\{N_i\}$ is the set of nodes neighboring the node N currently examined. Remember that $N \in \Phi$ if the adopted procedure is offensive and that $N \in \Theta$ if it is defensive. The ordering of the bits in the codeword is prescribed by some deterministic rule; for example, given $n = 4$ and a two-dimensional Γ, we may decide to visit the neighbors of N in the order East, North, West, South. The new codeword, obtained by examining the membership function on the neighboring nodes $\{N_i\}$ of the current node N, is concatenated to the

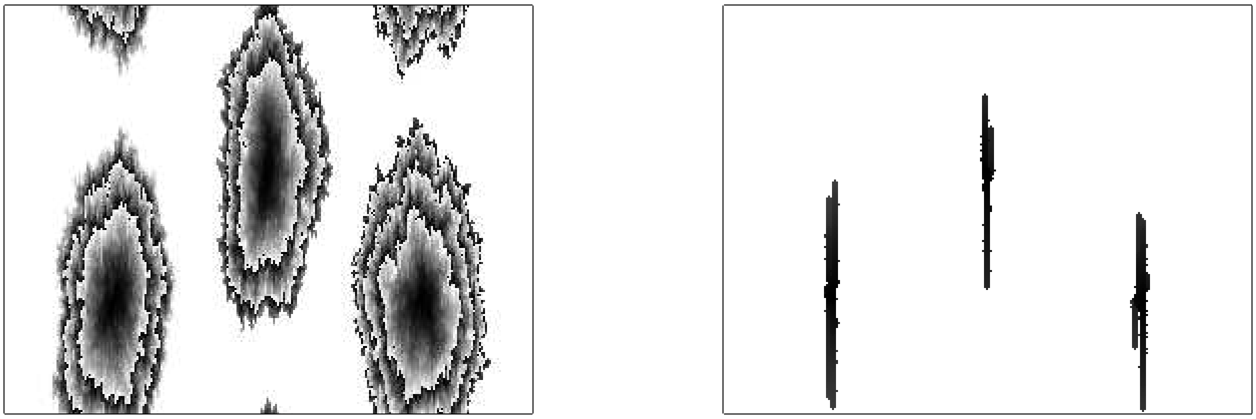


Figure 9. Illustration of Algorithm I (left) and Algorithm KI (right), 4-connected, initial Λ as three separate nodes, anisotropic growth probabilities.

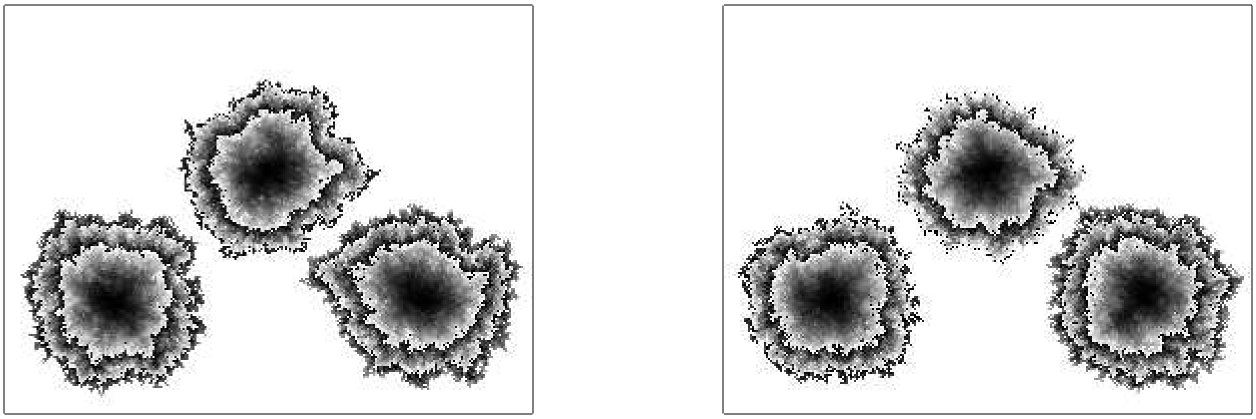


Figure 10. Illustration of Algorithm I (left) and Algorithm J (right), 8-connected, initial Λ as three separate nodes, isotropic growth probabilities.

previous ones, and the resulting string codes for the history of the set Φ or Θ during growth.

Note that, apart from the first, each additional codeword necessitates at most $n-1$ bits, often much less. As example of savings, only 3 bits are necessary for a two-dimensional 4-connected Γ_4 (instead of 4,) and a two-dimensional Γ_6 as well (instead of 6.) A mixture of 3 and 5 bits is sufficient for a two-dimensional Γ_8 (instead of 8.) In a cubic three-dimensional lattice, Γ_6 needs no more than 5 bits (instead of 6,) Γ_{18} needs a mixture of 9 and 11 bits (instead of 18,) while Γ_{26} needs 9 bits, 15 bits or 19 bits (instead of 26.) We present a more elaborate version of these savings in the context of chain coding (see later.) The appropriate number of bits in a codeword need not be specified explicitly because this number can be determined by the context during the decoding operation. However, these bit counts do not include the condition $\Xi = \emptyset$, which might be encountered in some of the algorithms and is thus in need of its own codeword.

Of course, a string of codewords requiring globally a length in excess of $n + (\mathrm{card}(\Omega) - 1)(n-1)$ for a figure of area or volume $\mathrm{card}(\Omega)$ is hardly efficient, even if it is unambiguous, and even if this length is given as a worst-case estimation and could be shortened by taking advantage of the savings we mentioned in the previous paragraph. Fortunately, for compact figures there will be many syntactic constraints. We expect entropy-coding methods would benefit from these constraints to greatly diminish the total amount of code required.

A somewhat more efficient code can be achieved on a regular two-dimensional lattice by coding the contour of a

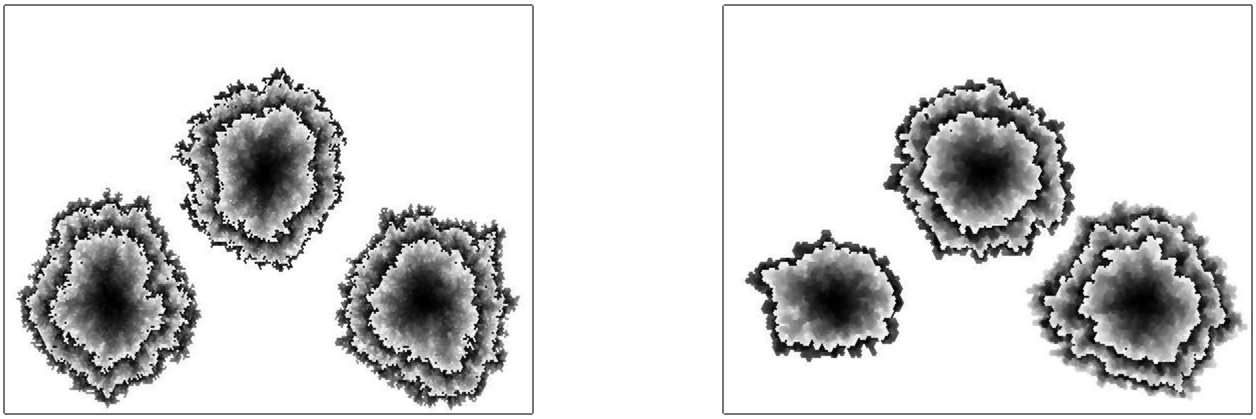


Figure 11. Illustration of Algorithm I (left) and Algorithm KI (right), 6-connected, initial Λ as three separate nodes, isotropic growth probabilities.

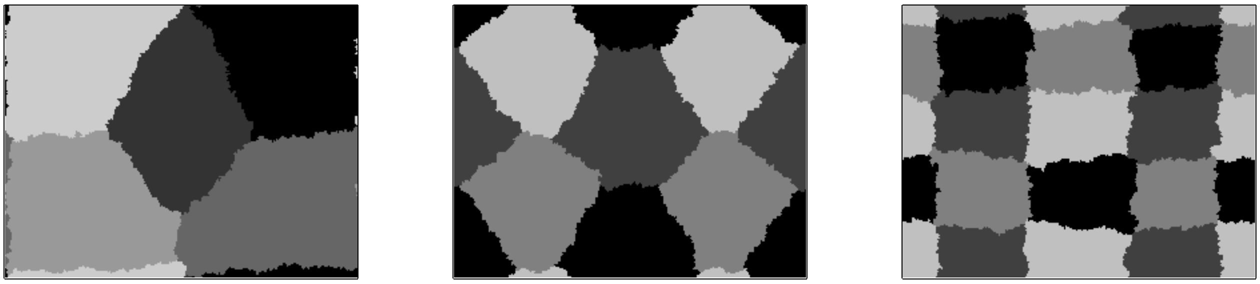


Figure 12. Illustration of Algorithm I, 4-connected, initial Λ as five (left,) eight (center,) and sixteen (right) separate nodes, isotropic growth probabilities, growth up to completion on a toroidal geometry.

connected set.[18] By convention, we represent each piece of the contour by a circular list of nodes, where each node is adjacent to its successor in the list. This list is oriented (there is a reading order,) and nodes may appear several times in the list. We construct each list in such a way that we may distinguish between ω and Ω by assigning one set to the left and the other to the right of each contour piece, respectively (this last constraint also ensures that the duplication of nodes is minimal.) With some additional information about the relative positions of the pieces, the global contour can code for holes without ambiguity. For example, on a two-dimensional square lattice, it is well known that unlabeled neighbors Φ_4 of a 4-connected labeled set are themselves piece-wise 8-connected[†]. Since it takes three bits to code for the eight possible directions, the length of the string of codewords will be $3\,\mathrm{card}(\Phi_4)$; similarly, the internal frontier of an 8-connected set requires a string of length $3\,\mathrm{card}(\Theta_8)$. By a dual argument, the string of codewords needed to represent a 4-connected Φ_8 reduces to a length $2\,\mathrm{card}(\Phi_8)$, while Λ_4 requires $2\,\mathrm{card}(\Theta_4)$ bits. These numbers do not include the part of the code responsible for describing the relative locations of the contour pieces. They must be further increased when the sub-graph associated to each piece is not Hamiltonian. We note that hexagonal lattices, whether 3-connected, 6-connected or 12-connected, introduce the need for fractional bits.

[†]Let us introduce an 8-connected regular lattice and a graph Γ_8 on this lattice that embeds a 4-connected square lattice and its associated graph Γ_4. The set Λ_4 is assumed to be itself 4-connected. In general, the set Φ_4 (relative to Λ_4 on the 4-connected lattice) is disconnected on the original graph Γ_4, while it is necessarily piece-wise connected on the graph Γ_8, where the number of pieces depends on the topology of the lattice (*e.g.*, plane, sphere, torus, Mœbius strip, Klein bottle,) and the number of holes of Λ_4, as counted on Γ_8. This relation is dual: let Λ_8 be an 8-connected set with an external frontier Φ_8. If the corresponding graph Γ_8 is restricted to be 4-connected by deleting all diagonal connections, then Φ_8 necessarily stays piece-wise connected on the restricted graph Γ_4 while Λ_8 generally becomes disconnected. On a hexagonal lattice, Φ_6 is piece-wise 6-connected on Γ_6, while Φ_3 is piece-wise 12-connected on Γ_{12}, which is a non-planar graph. These cross-relations simplify in the case of the internal frontier: Θ_n is piece-wise n-connected if Λ is n-connected.

The number of bits necessary for coding a contour can be made smaller than those presented above by considering the complete history of the string of directional codewords. In this case, these codewords will no longer be local. To understand why, we recall that the contour is made of oriented lists, which means that each node N_i has a set of predecessors $\Pi_i = \{N_0, N_1, \ldots, N_{i-1}\}$. The successor node N_{i+1}, if there is one, will be the next node in the list and is necessarily chosen among the n neighbors of N_i, unless all nodes of the list have been coded already, in which case the successor is the first node of the next piece of contour. If the codeword is local—as it was in the previous paragraph—the number of bits required to select 1 out of n possibilities is $\log_2 n$; but if one is willing to consider the status of all nodes and all neighbors of all nodes in Π_i as well, the length of the codeword associated to N_i for the specification of its successor N_{i+1} can be shortened, even if fractionally, because some of the nodes neighboring N_i may have been earlier candidates during the construction of the codeword for some node N_j, with $j < i$. In particular, at least N_{i-1} is such a node. These early candidates have to be withdrawn from the set of potential successors to N_i since their contour membership has been already estimated. Thus, the number of possibilities becomes smaller than the initial n cases.

With an additional convention, for example that the first contour node N_0 is always the Westmost node in the Northmost row of the contour on a square lattice (or a similar convention for hexagonal lattices,) it is possible to apply this shortening of the codewords also to the case $\Pi_0 = \emptyset$ because the convention ensures that the node N_0 has fewer than n neighbors (for example, on a square lattice, it has no North, North-West or West neighbor.) When the condition arises that the contour membership of all n neighbors to N_i has been previously determined, but all contour nodes have not been coded yet, the string of codewords has to specify the relative location of the next uncoded node N_k in the contour, and the coding has to resume from there, eventually skipping duplicate nodes, or eventually overriding a membership decision by including in the contour a node that was previously estimated as not belonging to the contour. This particular condition arises for example when constructing the codeword for the ultimate node of a list (unless the list has only one element,) when switching to a next piece of the contour, or when the graph associated to some contour piece is not Hamiltonian.

For figures that satisfy a criterion of discrete convexity, the length of the boundaries card(Φ) or card(Θ) will be of the order of $\sqrt{\text{card}(\Omega)}$. Such a boundary or contour code may be reasonable. We note that this is widely known as an example of a Freeman chain code. While extensions to three dimensions, as in video sequences are conceivable, to our knowledge an approach such as this one has not been tried, perhaps because of the difficulty of coding an enveloping surface instead of a curve. We note however that such a difficulty does not arise with volumetric coding, which is the approach we presented first in this section.

REFERENCES

1. A. Turing, "The chemical basis of morphogenesis," *Phil. Trans. Roy. Soc. London B* **237**, pp. 37–72, 1952.
2. M. Vold, "Sediment volume and structure in dispersions of anisometric particles," *J. Phys. Chem.* **63**, pp. 1608–1612, 1959.
3. M. Eden, *A Probabilistic Model for Morphogenesis*, pp. 359–370. Pergamon Press, New York, 1958.
4. M. Eden, "A two-dimensional growth process," *Proc. Fourth Berkeley Symposium in Mathematical Statistics and Probability* , pp. 223–239, Univ. California Press, (Berkeley, CA USA), 1961.
5. D. Drasdo, R. Kree, and J. McCaskill, "Monte carlo approach to tissue-cell populations," *Phys. Rev. E* **52**, pp. 6635–6657, 1995.
6. A. Savakis and S. Maggelakis, "Models of wound healing and tissue regeneration," *Proc. Soc. for Math. Biol. annual meeting* , pp. SMB–21, (Univ. of Washington, Seattle, WA USA), August 1996.
7. N. Vandewalle and M. Ausloos, "Two component spreading phenomena: Why the geometry masks the criticality?," *Phys. Rev. E* **54**, pp. 3006–3008, 1996.
8. L. Benguigui, "A new aggregation model: Application to town growth," *Physica A* **219**, pp. 13–26, 1995.
9. R. Leheny, "Simple model for river network evolution," *Phys. Rev. E* **52**(5 Part B), pp. 5610–5620, 1995.
10. C. Ryu and K. I.-M., "Effects of surface diffusion on the Eden model," *Phys. Rev. E* **53**, pp. 5643–5646, 1996.
11. J. Hammersley, *Bornes supérieures de la probabilité critique dans un processus de filtration*, pp. 17–37. CNRS, Paris, France, 1959.
12. I. Benjamini and O. Schramm, "Percolation beyond Z^d, many questions and few answers," *Electronic Communications in Probability* **1**(8), pp. 71–82, 1996.

13. R. Jullien and R. Botet, "Scaling properties of the surface of the Eden model in D = 2, 3, 4," *J. Phys. A, Math. Gen.* **18**, pp. 2279–2287, 1985.
14. F. Aarao Reis, "Scaling for random walks on Eden trees," *Phys. Rev. E* **54**, pp. R3079–3081, 1996.
15. A. Hasmy, E. Anglaret, and R. Jullien, "Local reactivity limited aggregation," *Phys. Rev. E* **54**, pp. 4454–4457, 1996.
16. D. Cohen, "Computer simulation of biological pattern generation processes," *Nature* **216**, pp. 246–248, 1967.
17. T. Sintes, R. Toral, and A. Chakrabarti, "Fractal structure of silica colloids revisited," *J. Phys. A, Math. Gen.* **29**, pp. 533–540, 1996.
18. M. Eden and M. Kocher, "On the performance of a contour coding algorithm in the context of image coding, part 1: Contour segment coding," *Signal Processing* **8**, pp. 381–386, 1985.