Strategies for Optimize Off-Lattice Aggregate Simulations

S. G. Alves,* S. C. Ferreira Jr.,† and M. L. Martins‡
Departamento de Física, Universidade Federal de Viçosa, 36570-000, Viçosa, MG, Brazil

Received on, 27 September, 2007

We review some computer algorithms for the simulation of off-lattice clusters grown from a seed, with emphasis on the diffusion-limited aggregation, ballistic aggregation and Eden models. Only those methods which can be immediately extended to distinct off-lattice aggregation processes are discussed. The computer efficiencies of the distinct algorithms are compared.

Keywords: Off-lattice aggregation; Diffusion-limited aggregation; Ballistic aggregation; Eden model

I. INTRODUCTION

Growth processes occurring far from equilibrium are widespread in nature and technology. Examples include electrodeposition [1], viscous fingering [2], bacterial colonies [3], and neurite formation [4]. Computer models for the growth of clusters, generally constituted of identical particles, are useful tools for the understanding of aggregation phenomena. The main contribution of such models is to provide pathways to investigate the underlying physical ingredients ruling the properties observed in growth phenomena. One of the most intriguing features of the fractal structures found in nature and computer models is the scale invariance emerging without fine-tuning of any parameter, in contrast with usual critical phenomena in which scale invariance only emerges at a critical point [5].

The foremost example of nonequilibrim growth model is the diffusion-limited aggregation (DLA) model introduced by Witten and Sander [6] in 1981. The rules of the DLA model are based on an iterative stochastic process in which the particles, one at a time, follow Brownian trajectories until they touch and stick in an aggregate. Despite its simple rules, the DLA model leads to very complex aggregates with multiscale properties [7, 8] and multifractality in the growth-site probability distribution [9, 10].

If the random walks in the DLA model are replaced by ballistic trajectories at random directions, we have the ballistic aggregation (BA) model [11, 12] proposed by Vold to describe colloid aggregation. Differently from DLA, the BA model generates asymptotically non-fractal clusters (fractal dimension equal to the space dimension) characterized by a power law approach to the asymptotic regime [13, 14].

Finally, a third standard aggregation process was proposed by Eden [15] as a basic model for the biological pattern formation as, for instance, tumor growth and bacterial colonies. In this model, new particles are sequentially added to the empty neighborhood of the cluster without overlap with previously aggregated particles [16, 17]. Although the Eden model is unrealistic from the biological point of view, it produces compact aggregates with a nontrivial interface scaling usually analyzed through the interface width w [18]. Intensive numerical simulations indicate a power-law growth of the interface width with the time, $w \sim t^{\beta}$, and exponent $\beta = 1/3$ [14, 17, 19, 20], corresponding to the Kardar-Parisi-Zhang (KPZ) universality class [21].

The DLA, BA, and Eden models can be implemented and simulated in a relatively easy way by constraining the particle positions to the sites of an underlying lattice. However, it is very well established that lattice anisotropy imposes strong effects on the cluster shape and scaling [22–25]. Although some procedures have been proposed to remove the anisotropy of on-lattice clusters [26–28], their successes were limited and off-lattice simulations impose themselves as a general framework for the investigation of the scaling properties and universality classes of these aggregates. Clearly, the aggregation of a large number of particles is necessary to reach the asymptotic behavior which, in turn, demands very efficient algorithms for large scale off-lattice simulations with rigorous statistical sampling. In this paper, we review several strategies used to optimize computer algorithms for off-lattice aggregates. Only those procedures which can be applied to general off-lattice simulations are focused here. More sophisticated but less general procedures, as conformal maps [29], are avoided. Indeed, the conformal mapping is the most efficient strategy to simulate two-dimensional aggregates, but it cannot be used in higher dimensions.

II. ALGORITHMS FOR OFF-LATTICE AGGREGATION

In this section we present the description of distinct optimizations for two-dimensional clusters. The generalization for higher dimensions is straightforward. In all cases, simulations start with a single particle at the origin.

A. The trajectories

Firstly, we describe optimizations for models in which particles of unitary diameter follow trajectories before stick to the aggregate, as is the case for the DLA and BA models. In both cases, the particles are released at random from a launching radius r_l larger than the cluster radius r_{max} and follow their trajectories up to touch the aggregate or cross a killing radius

*Electronic address: sidiney@ufv.br
†Electronic address: silviojr@ufv.br ; (Corresponding Author.)
‡Electronic address: mmartins@ufv.br

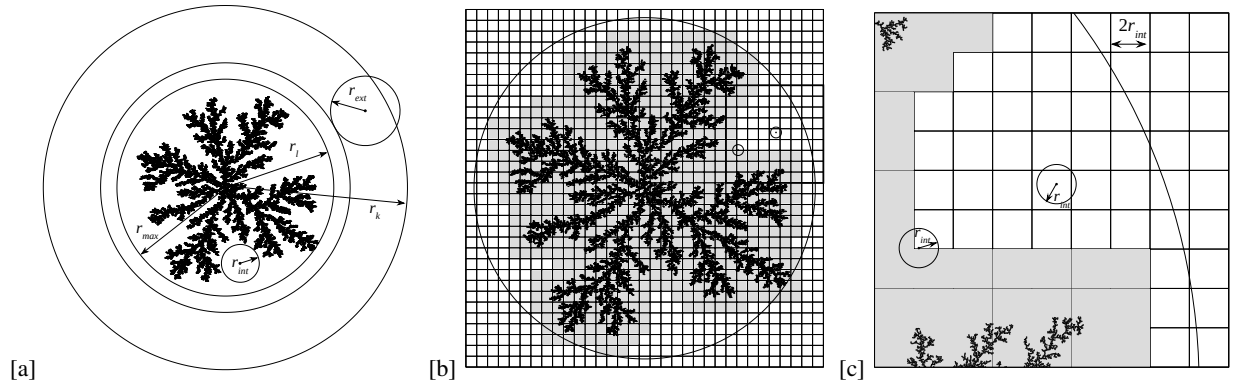

FIG. 1: (a) Schematic representation of the "optimized random trajectories". (b) A DLA aggregate and a mesh of cells $2r_{int} \times 2r_{int}$. Long steps are forbidden in the gray boxes and allowed in the white ones. Also, two long steps are illustrated. (c) A zoom of the region inside the large square in (b).

r_k much larger than the system size. In the DLA, where particles follow discrete time random walks of unitary steps, a standard method is to allow the particles outside the launching circle take long random steps of length r_{ext} if these steps do not brings up a particle inside the launching circle, as illustrated in Fig. 1(a). An adequate choice is $r_{ext} = \max(r - r_{max} - \delta, 1)$, where r is the distance of the walker from the origin and a small tolerance $\delta = 5$ was used. Also, the Brownian walks in large empty areas in the inner region which delimits the cluster ($r < r_{max}$) are very computer time consuming, specially for large aggregates. Ball and Brady [30] proposed a strategy which allows the particles inside the launching circle to take a long step of length r_{int} if they do not cross any part of the aggregate, as illustrated in Fig. 1(a). Similar procedures have been used in other works [31–33].

In the BA model, the particles follow ballistic trajectories and the clusters do not exhibit large empty inner regions as in the DLA model. Hence, the trajectories can be efficiently implemented simply using a long step of size r_{ext} as in DLA model. An important difference between BA and DLA implementations is that in the first the launching radius should be as large as possible in order to avoid growth instabilities promoted by shadowing effects [34, 35] while in the DLA, this radius can be taken a few particle diameters larger than the cluster radius.

A smart strategy to determine the length of the internal steps r_{int} is decisive for the algorithm efficiency. In order to accomplish this task, we define a square region of side L centered on the initial seed which delimits the entire aggregate. This region should be sufficiently large in order to guarantee that aggregate does not exceed its boundary. Then, the region is divided in a coarse-grained mesh with cells of size $2r_{int} \times 2r_{int}$ as illustrated in Figs. 1(b) and 1(c). Each cell of the mesh is associated to an element of a $K \times K$ square matrix $\mathcal{A}$, where $K = L/(2r_{int})$, which assumes 1 if the cell or one of its nearest or next-nearest neighbors contains any particle of the aggregate or assumes 0 otherwise. The boxes depicted in gray ($\mathcal{A}_{ij} = 1$) are those in which the random walk can cross the cluster after a step of length r_{int}, since they contain or are adjacent to a part of the cluster. Consequently, long steps starting from gray boxes are forbidden. There are two options for a walker on a gray box: the particle executes a unitary step or tries a shorter step of length r'_{int}, where $1 < r'_{int} < r_{int}$, using other auxiliary coarse-grained mesh $\mathcal{A}'$ with cells of size $2r'_{int} \times 2r'_{int}$. Indeed, several auxiliary meshes can be used in order to maximize the efficiency. In this paper, we report simulation for 3 meshes with $r_{int} = 4,\ 8$, and 16.

The overlap between particles can occur after a unitary step if the preceding step brings the random walker at a distance from the cluster particle where it sticks lower than the unity. In this case, one just brings back the particle to the adjacent position along the opposite direction of the movement.

B. Determination of the neighborhood

The search mechanism for determining when and where the walker has contacted the aggregate represents the major time consuming step in large off-lattice simulations. The spatial coordinates of the particle belonging to the cluster are stored in one-dimensional arrays at the sequence of aggregation. So, the inspection of these arrays is performed whenever the walkers are in the nearby of the aggregate. If none optimization is adopted, all aggregated particles may be checked to verify if a contact occurred or not. At least three optimizations can be used. In the first and simplest one, we just verify the list in the reverse order in which the particles were added to the cluster, because the chance is larger for the aggregation to take place on the more external particles than on the inner ones. This procedure is considered default in this work. In the second one, particle positions are mapped on a square lattice by approximating their real coordinates to the nearest integer, producing an on-lattice cluster. In an auxiliary square lattice $\mathcal{Z}$, we label as occupied those sites belonging to the previous on-lattice cluster as well as their nearest and next-nearest neighbors. The search for contact is done only if the nearest integer coordinates of the walker represent an occupied site of the lattice $\mathcal{Z}$. This procedure is schematically described in Fig. 2(a). In the third optimization procedure, a coarse-grained mesh $\mathcal{W}$ of cells with size $\ell \times \ell$ can be used to limit the verification to a re-

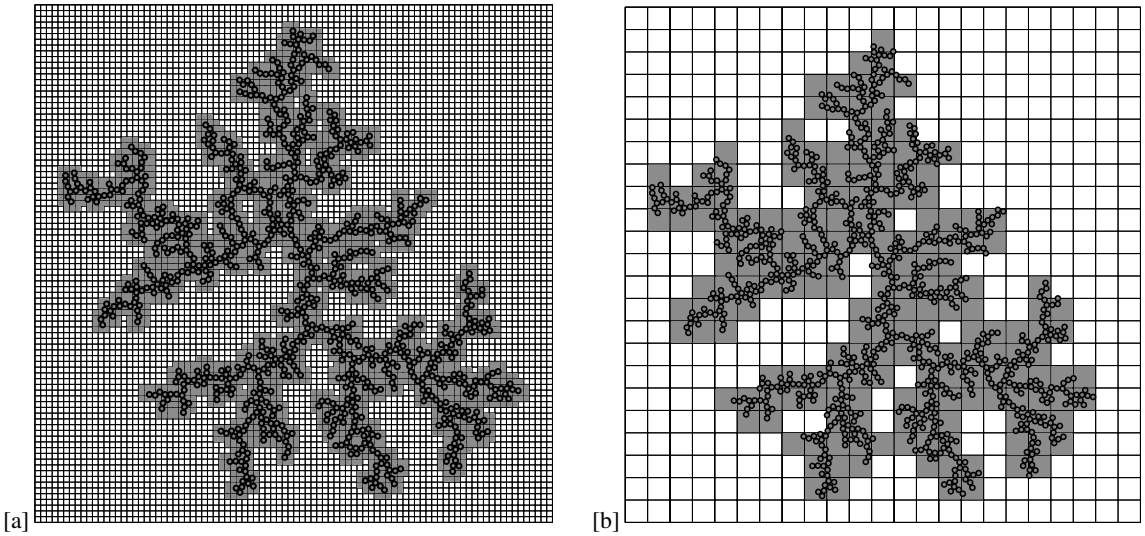

FIG. 2: Illustration of the optimizations for off-lattice aggregation processes. (a) An auxiliary square lattice is used to determine when the walker is neighboring the cluster. The cluster particles are represented by black circles and their neighbors are depicted in gray. (b) A mesh with cells of size 4×4 used to restrict the search for contacts nearby the walker.

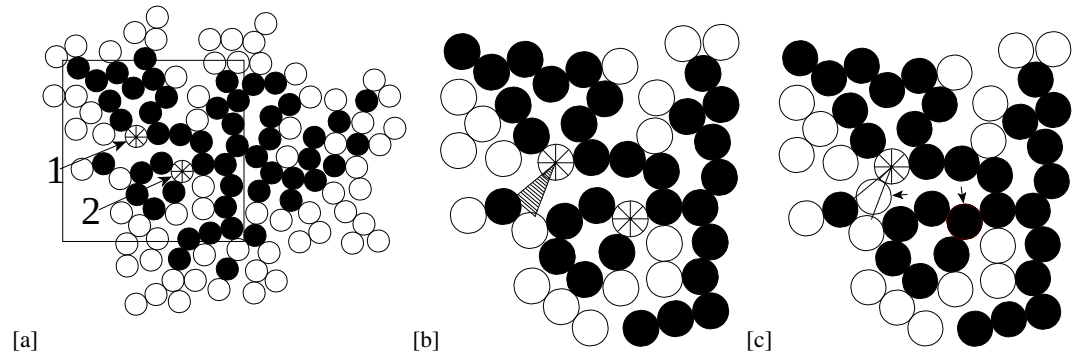

FIG. 3: Growth rules for the off-lattice Eden model. Active and inactive particles are represented by open and fullfiled discs, respectively. (a) A cluster and two active particles selected for the growth. The particle 1 has an empty region where a new adjacent particle can be added while the particle 2 does not. (b) The growth region adjacent to the particle 1 is shown as a dashed sector. (c) A new particle is added at a random direction in the growth region shown in (b) and the particle 2 is discarded from the list of active ones (both indicated by arrows).

gion around the walker position. In this strategy, the cells are sequentially labeled by an index $k = 1, 2, 3, \cdots$ when they are occupied by a particle of the cluster for the first time. Also, the number of particles N_k in the cells are stored. Finally, a third auxiliary one-dimensional array $\mathcal{F}$ divided in blocks with ℓ^2 elements is used to store the indexes of the particles in the arrays of coordinates. Each block is associated to a cell of the mesh. Once the analysis of the auxiliary square matrix $\mathcal{Z}$ has provided that the walker may be in contact with a particle of the cluster, the index k read in the mesh $\mathcal{W}$ is used to restrict the search for a contact in the array of coordinates using $\mathcal{F}$. The cell index of a walker at real coordinates (x, y) is given by $k = \mathcal{W}_{ij}$, where $i = \text{nint}(x/\ell)$, $j = \text{nint}(y/\ell)$, and $\text{nint}(x)$ function rounds x to the nearest integer. Indeed, the particles in the cell k are visited by varying the index of the array $\mathcal{F}$ from $n = n_k + 1$ to $n = n_k + N_k$, where $n_k = \ell^2 \times (k-1)$. Notice that the cell j of the mesh $\mathcal{W}$ and its neighbor cells should be verified to check the contacts on the cell edges. In the simulation results presented in the next section, $\ell = 4$ was used.

C. The Eden model

The off-lattice simulation of the Eden model was proposed by Wang et al. [16] and improved by Ferreira and Alves [17] as follows

- A particle with unitary diameter is chosen at random from a list of active ones (Fig. 3(a)). A particle is considered active when a new one adjacent to it can be added to the aggregate without any overlap.
- Once an active particle was chosen (Fig. 3(b)), its empty adjacent region, where there are no overlap between a

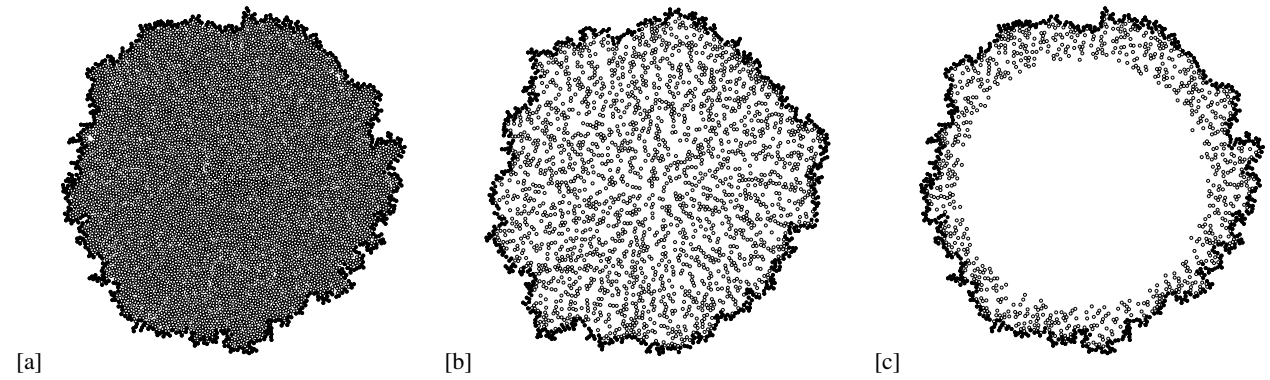

FIG. 4: (a) Eden cluster with 6000 particles. The border is represented by fullfiled symbols. Active particles for (b) standard and (c) optimized off-lattice algorithms for the Eden model are shown.

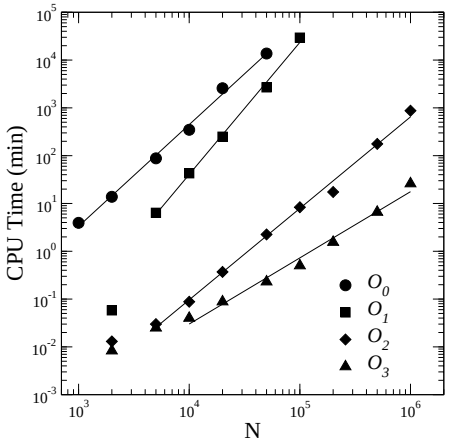

FIG. 5: CPU times as functions of the number of particles in the off-lattice DLA model for distinct optimization strategies. Lines are power fits.

new particle and those previously aggregated, is determined. A new particle is put in a direction randomly chosen among the allowed ones (Fig.3(b)).

- If the active particle does not have a growth region, it is labeled as inactive (Fig.3(c)).

In Fig, 3 the evolution rules are illustrated by two independent growth processes. Since the interest on Eden clusters is focused on the interface scaling, Ferreira and Alves [17] introduced an optimization where any active cell inside a central core of radius r_c is labeled as inactive. Since the inactivation of the particles near or belonging to the interface must be avoided, $r_c = 0.8\bar{r}$ was chosen, where $\bar{r}$ is the mean radius of the interface. This optimization was used only for $\bar{r} > 80a$. In Fig. 4, typical growth patterns with and without this last optimization, the corresponding borders [36], and the active particles are illustrated. Finally, the optimizations described in sub-section II B for determining the neighborhood of a particle can be used for the Eden model.

III. SIMULATIONS

All simulations were performed on the same computer, a Pentium IV 3.0 GHz with 2GB of RAM memory under Debian Linux operating system. One process was run by time. The algorithm codes were written in FORTRAN 90 language and compiled with the standard options of the Intel Fortran Compiler 9.1 [37].

A. Diffusion-limited aggregation

Off-lattice DLA clusters with N particles were grown using different combinations of the previously described optimizations. In all simulations, the launching and killing radius were taken as $r_l = r_{max} + 5$ and $r_k = 100r_l$, respectively. In 1981, when Sander and Witten published their seminal work introducing the DLA model [6] without any optimization, the largest cluster generated on square lattices produced with computers of that age did not reach 4000 particles. Nowadays, this sort of simulation can be performed in a few minutes with any standard home computer. In table I, the CPU times spent in off-lattice simulations of a single cluster for some optimization schedules are listed. Also, CPU times are shown as functions of N in Fig. 5.

Simulations without optimizations become prohibitively long for relatively small aggregates. For example, a single cluster with 5×10^4 particles demanded 10 days of simulations. If external steps are included in the original algorithm, for simplicity called by O_1, a great improvement of the efficiency is observed for very small clusters, but the simulations are also prohibitive for $N \sim 10^5$, since inner empty regions become of the same magnitude than the cluster size. Simulations become more than three orders of magnitude faster when the optimized neighborhood determination is included in O_1 optimization, now called O_2. Notice that the computational time grows approximately proportional to N^2 for both optimizations O_0 and O_2. Simulations one order faster and CPU times growing slower with increasing cluster size are performed when inner steps are included in O_2 optimization.

N	O_0	O_1	O_2	O_3
1×10^3	3.93×10^0	1.62×10^{-3}	1.6×10^{-3}	1.6×10^{-3}
2×10^3	1.38×10^1	5.81×10^{-2}	1.3×10^{-2}	8.3×10^{-3}
5×10^3	8.79×10^1	6.37×10^0	3.0×10^{-2}	2.5×10^{-2}
1×10^4	3.48×10^2	4.29×10^1	8.8×10^{-2}	4.0×10^{-2}
2×10^4	2.57×10^3	2.49×10^2	3.7×10^{-1}	8.8×10^{-2}
5×10^4	1.37×10^4	2.69×10^3	2.25×10^0	2.3×10^{-1}
1×10^5	—	2.94×10^4	8.33×10^0	5.0×10^{-1}
2×10^5	—	—	1.74×10^1	1.55×10^0
5×10^5	—	—	1.76×10^2	6.60×10^0
1×10^6	—	—	8.73×10^2	2.60×10^1
CPU time	$T\sim N^{2.1}$	$T\sim N^{2.8}$	$T\sim N^{1.9}$	$T\sim N^{1.4}$

TABLE I: Real CPU times in minutes for distinct optimizations applied to the DLA model. N is the number of particles; O_0 refers to the algorithm with the default optimization where the backward inspetion of the coordinate arrays is used; O_1 means that the long external steps of size r_{ext} were used; O_2 means that external steps and optimized neighborhood were used simultaneously; O_3 the previous optimizations plus the internal long steps of size r_{int} (Figs. 1 and 2) were adopted. The approximate dependence between CPU time and cluster size are indicated in the last line.

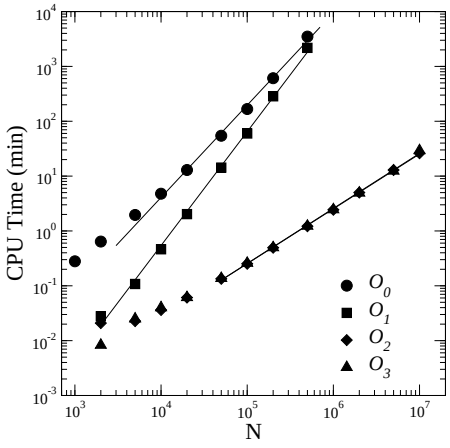

FIG. 6: CPU times as functions of the number of particles in the off-lattice BA model for distinct optimization strategies. Lines are power fits.

Also, notice that the computational time increases faster in O_1 than in the others optimizations, but for large clusters O_0 and O_1 optimizations are expected to be equivalent due to the presence of large empty inner regions.

B. Ballistic aggregation

Off-lattice simulations of the BA model are very similar to the DLA model. The main difference is that the unitary steps performed by the walkers are in a fixed direction randomly chosen at the beginning of the ballistic walk. Also, the launching and killing radius used were $r_l = 100r_{max} + 1000$ and $r_k = r_l + 10$. In table II, the computational times for the same

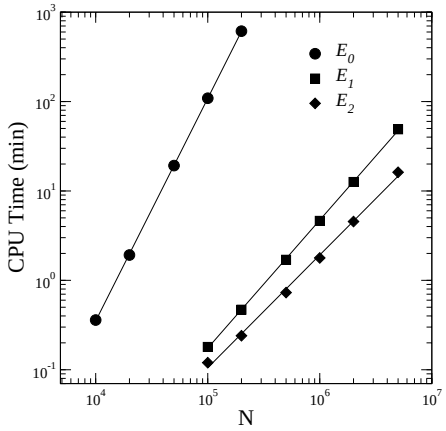

FIG. 7: CPU times as functions of the number of particles in the off-lattice Eden model using distinct optimization strategies. Lines are power fits.

N	O_0	O_1	O_2
1×10^3	2.81×10^{-1}	1.51×10^{-2}	1.35×10^{-2}
2×10^3	6.45×10^{-1}	2.78×10^{-2}	2.09×10^{-2}
5×10^3	1.96×10^0	1.08×10^{-1}	2.27×10^{-2}
1×10^4	4.78×10^0	4.63×10^{-1}	3.62×10^{-2}
2×10^4	1.29×10^1	2.03×10^0	6.10×10^{-2}
5×10^4	5.44×10^1	1.43×10^1	1.34×10^{-1}
1×10^5	1.67×10^2	5.99×10^1	2.52×10^{-1}
2×10^5	6.10×10^2	2.85×10^2	4.94×10^{-1}
5×10^5	3.50×10^3	2.18×10^3	1.22×10^0
1×10^6	—	—	2.43×10^0
2×10^6	—	—	5.01×10^0
5×10^6	—	—	1.29×10^1
5×10^6	—	—	2.61×10^1

TABLE II: Real CPU time in minutes for distinct optimizations applied to BA model. Optimizations as in table I.

strategies used for DLA are listed. Like in the DLA model, long steps improve simulation efficiency for small clusters, but this gain decreases with increasing number of particles. However, optimized neighborhood determination provodes a gain of three orders of magnitude. In Fig. 6 the CPU times are drawn as functions of N. These times grow approximately as $T\sim N^{1.7}$, $T\sim N^{2.1}$, and $T\sim N^{1.0}$ for O_0, O_1, and O_2, respectively.

C. Eden model

The major challenge in off-lattice simulation of the Eden model is to determine which are the active cells. Since Eden model does not involve walkers, strategies as those of Figs. 1 and 2(a) do not have sense. But, an efficient determination of the empty neighborhood can be used as done for the DLA model. The original strategy proposed by Wang et al. [16] is called E_0 and when the local search of neighbors is included,

N	E_0	E_1	E_2
1×10^3	1.12×10^{-2}	1.33×10^{-2}	1.33×10^{-2}
2×10^3	2.04×10^{-2}	1.33×10^{-2}	1.33×10^{-2}
5×10^3	7.31×10^{-2}	1.60×10^{-2}	1.83×10^{-2}
1×10^4	3.63×10^{-1}	2.33×10^{-2}	2.33×10^{-2}
2×10^4	1.92×10^{0}	3.67×10^{-2}	3.33×10^{-2}
5×10^4	1.92×10^{1}	8.33×10^{-2}	6.00×10^{-2}
1×10^5	1.09×10^{2}	1.81×10^{-1}	1.20×10^{-1}
2×10^5	6.13×10^{2}	4.72×10^{-1}	2.40×10^{-1}
5×10^5	—	1.70×10^{0}	7.30×10^{-1}
1×10^6	—	4.63×10^{0}	1.78×10^{0}
2×10^6	—	1.26×10^{1}	4.55×10^{0}
5×10^6	—	4.89×10^{1}	1.62×10^{1}

TABLE III: Eden Model Optimizations. Symbols E_0, E_1, and E_2 described in text.

the model is denoted by E_1. CPU times are given in table III and Fig. 7. The last algorithm overcomes the first one in three or more orders of magnitude. If a central core of particles is excluded from the list of active ones, the optimization E_2, simulations become up to three times faster. Moreover, the efficiency gain increases with the number of particles. Indeed, CPU times grow approximately as $T \sim N^{2.5}$, $T \sim N^{1.4}$, and $T \sim N^{1.2}$ for E_0, E_1, and E_2, respectively.

IV. SUMMARY

Several optimizing strategies for the computer simulation of aggregation models dispersed throughout the literature were described in the present paper. It have been demonstrated that the combined implementation of such strategies can reduce in up to four order of magnitude the computer time demanded to perform large scale simulations of off-lattice aggregates with an increase of one order of magnitude in the allocated memory. Furthermore, these procedures can be applied to the simulations of other cluster growth processes beyond the traditional DLA, BA, and Eden models.

Acknowledgments

This work was partially supported by CNPq and FAPEMIG, Brazilian agencies. We thank to Nemésio M. Oliveira-Neto for non expertise reading of the manuscript and his valuable contribution to make the paper more accessible.

[1] M. Matsushita, M. Sano, Y. Hayakawa, H. Honjo, and Y. Sawada, Phys. Rev. Lett. **53**, 286 (1984).
[2] K. J. Måløy, J. Feder, and T. Jøssang, Phys. Rev. Lett. **55**, 2688 (1985).
[3] M. Matsushita and H. Fujikawa, Physica A **168**, 498 (1990).
[4] F. Caserta, H. E. Stanley, W. D. Eldred, G. Daccord, R. E. Hausman, and J. Nittmann, Phys. Rev. Lett. **64**, 95 (1990).
[5] H. E. Stanley, *Introduction to phase transitions and Critical Phenomena* (Oxford University Press, Cambridge, 1971).
[6] T. A. Witten and L. M. Sander, Phys. Rev. Lett. **47**, 1400 (1981).
[7] C. Amitrano, A. Coniglio, P. Meakin and M. Zanneti, Phys. Rev. B **44**, 4974 (1991).
[8] B. B. Mandelbrot, B. Kol, and A. Aharony, Phys. Rev. Lett. **88**, 055501 (2002).
[9] C. Amitrano, A. Coniglio, and F. Diliberto, Phys. Rev. Lett. **57**, 1016 (1986).
[10] L. M. Sander, Contemp. Phys. **41**, 203 (2000).
[11] M. J. Vold, J. Colloid. Sci. **18**, 684 (1963).
[12] P. Meakin, *Fractals, scaling and growth far from equilibrium* (Cambridge University Press, Cambridge, 1998).
[13] S. Liang and L. P. Kadanoff, Phys. Rev. A **31**, 2628 (1985).
[14] T. Vicsek, *Fractal Growth Phenomena* (World Scientific, Singapore, 1992).
[15] M. Eden, in J. Neyman (ed.), *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability Vol. 4: Biology and Problems of Health*, (University of California Press) (1961).
[16] C. Y. Wang, P. L. Liu, and J. G. Bassingthwaighte, J. Phys. A: Math. Gen. **28**, 2141 (1995).
[17] S. C. Ferreira Jr. and S. G. Alves, J. Stat. Mech.: theory and experiment P11007 (2006).
[18] The interface width is commonly defined as the standard deviation of the distances from the center of the lattice.
[19] J. Kertész and D. E. Wolf, J. Phys. A: Math. Gen. **21**, 747 (1988).
[20] P. Devillard and H. E. Stanley, Physica A **160**, 298 (1989).
[21] M. Kardar, G. Parisi and Y. C. Zhang, Phys. Rev. Lett. **56**, 889 (1986).
[22] S. Tolman and P. Meakin, Phys. Rev. A **40**, 428 (1989)
[23] N. R. Goold, E. Somfai, R. C. Ball, Phys. Rev. E **72**, 031403 (2005).
[24] J. G. Zabolitzky and D. Stauffer, Phys. Rev. A **34**, 1523 (1986).
[25] M. J. Batchelor and B. I. Henry, Phys. Lett. A **157**, 229 (1991).
[26] L. R. Paiva and S. C. Ferreira, J. Phys. A. Math. Gen. **40**, F43 (2007).
[27] S. G. Alves and S. C. Ferreira, J. Phys. A. Math. Gen. **39**, 2843 (2006).
[28] V. A. Bogoyavlenskiy, J. Phys. A. Math. Gen. **35**, 2533 (2002).
[29] B. Davidovitch, H. G. E Hentschel, Z. Olami1, I. Procaccia1, L. M. Sander, and E. Somfai, Phys. Rev E. **59**, 1368 (1999).
[30] R. C. Ball and R. M Brady, J. Phys. A: Math. Gen. **18**, L809 (1985).
[31] P. Meakin and T. Vicsek, Phys. Rev. A **32**, 685 (1986).
[32] S. G. Alves and S. C. Ferreira, Phys. Rev. E **73**, 051401 (2006).
[33] S. C. Ferreira, Eur. Phys. J. B **42**, 263 (2004).
[34] C. Tang and S. Liang, Phys. Rev. Lett. **71**, 2769 (1993).
[35] J. Yu and J. G. Amar, Phys. Rev. E **66**, 021603 (2002).
[36] The border is defined as the set of cells that forms an external layer impenetrable for incoming cells. Consequently, the spaces between consecutive border cells is smaller than a cell diameter.
[37] The free non-commercial version of the Intel Fotran Compiler for linux was take from `http://www.intel.com/cd/software/products/asmo-na/eng/compilers/flin/282048.htm`.