Smart covering for a box-counting algorithm

Marcos Yamaguti* and Carmen P. C. Prado†
Instituto de Física, Universidade de São Paulo, Caixa Postal 66318, 05315-970, Brazil

(Received 28 May 1996; revised manuscript received 14 November 1996)

We describe a box-counting algorithm that does not use a grid to cover a multifractal set. It leads to better results, especially in the $q<0$ region of the curves $f(\alpha)$ and D_q. We present tests of this algorithm for the attractor of the logistic map on the onset of chaos, for a binomial measure, and for the attractor of the Hénon map. [S1063-651X(97)05104-0]

PACS number(s): 02.70.−c, 61.43.Hv, 05.45.+b

Multifractal objects have been identified in many physical situations [1], ranging from problems of aggregation to the behavior of chaotic dynamical systems. The multifractal sets are usually characterized [2,3] by the generalized dimensions D_q and the associated spectrum of singularities $f(\alpha)$.

In the actual calculations, we usually introduce a covering which consists of a collection of boxes that contain the whole multifractal set, and associate a probability p_i with each box. The definition of the generalized dimensions D_q and the spectrum of singularities $f(\alpha)$ are based on these probabilities, which are supposed to be the integral of the invariant measure inside the boxes. As the theoretical coverings are required to have some properties that are very difficult to achieve in practice, the expressions of D_q and $f(\alpha)$ for most fractal sets cannot be calculated analytically, and we are forced to resort to numerical methods.

The best known numerical methods [4–6] to characterize the multifractal sets are based on box-counting algorithms [7–9] based on the construction of very simple coverings. Usually, we just consider a grid, that is, a set of boxes of the same size arranged side by side as in a chessboard (as shown in Fig. 1). However, we have already pointed out that grids are not very good coverings [10]. Indeed, these simple grids can lead to the existence of boxes that have a small probability p_i because they are almost entirely filled with empty space (instead of reflecting the weight of a region with just a few fractal points). These ''spurious'' boxes of the standard box-counting algorithms (see some examples in Fig. 1) are responsible for significant distortions of the curves for D_q and $f(\alpha)$ (especially of the $q<0$ branches, which are associated with regions of small probabilities). Also, the distortions introduced by those spurious boxes do not vanish in the limit of small boxes [10]. This kind of difficulty may be overcome if we are able to consider boxes which do not belong to a particular grid.

In this paper we describe an alternate box-counting algorithm to construct much better coverings. The boxes still have the same size and orientation, but are not arranged in a grid. As this type of covering is closer to the theoretical definitions, we are able to eliminate most of the spurious boxes, and obtain significantly better results for D_q and $f(\alpha)$. We present some tests for the attractor of the logistic map on the onset of chaos, for a binomial measure, and for the attractor of the Hénon map.

This paper is divided as follows. In Sec. I, we present the theoretical definitions associated with the problem. In Sec. II, we describe the method to construct a better covering for a box-counting algorithm, and in Sec. III we present some results obtained with this new method. In Sec. IV, we summarize our conclusions. In the Appendix the algorithm is described in detail.

I. DEFINITIONS OF THE MULTIFRACTAL THEORY

The statistical properties of a multifractal set are characterized by the generalized dimensions D_q and the associated spectrum of singularities $f(\alpha)$. In the multifractal theory [2,3] the definition of the generalized dimensions is introduced through the following steps, where the definition of an l covering of a set A is used. Consider a set of sets U_i (boxes) with the properties (1) the size l_i of the set U_i obeys the condition $l_i \leq l$ (l is the size of the biggest box); (2) $U_1 \cup U_2 \cup \cdots U_N \supset A$ (the union of all boxes contains the whole set); and (3) $U_i \cap U_j$ is empty, for every pair $i,j \leq N$ and $i=j$ (the boxes do not overlap).

The probability p_i is defined as the integral of the invari-

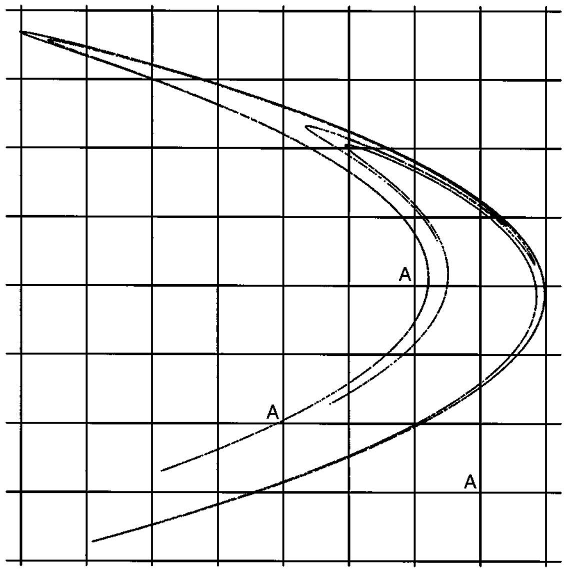

FIG. 1. The Hénon attractor covered by a grid that leads to the existence of spurious boxes (labeled by the letter A).

*Electronic address: yamaguti@onsager.if.usp.br
†Electronic address: prado@uspif.if.usp.br

ant measure inside U_i. Numerically, the set is represented by points, so the probability p_i is the fraction of points inside the box U_i.

To define the generalized dimensions, let us consider the following points related to the spectrum of mass $\tau(q)$.

(1) Define a function $\Gamma(A,q,\tau,\{U_i\},l)=\Sigma_i^N(p_i^q/l_i^\tau)$.

(2) $$\Gamma(A,q,\tau,l)=\begin{cases}\sup\limits_{\{l_i\}}\{\Gamma(A,q,\tau,\{U_i\},l)\} & \text{for } q>1\\ \inf\limits_{\{l_i\}}\{\Gamma(A,q,\tau,\{U_i\},l)\} & \text{for } q<1.\end{cases}$$

(3) Then take the limit $\Gamma(A,q,\tau)=\lim_{l\to 0}\Gamma(A,q,\tau,l)$.

(4) Finally, impose the condition $\Gamma(A,q,\tau)=1$ to establish a single relation between τ and q, given by the function $\tau(q)$.

There is a relationship between the spectrum of mass $\tau(q)$ and the curves D_q and $f(\alpha)$. In fact, we have $\tau(q)=(q-1)D_q$, and $f(\alpha)$ is the Legendre transformation of $\tau(q)$.

The numerical calculations of D_q and $f(\alpha)$ using a box-counting algorithm usually follow the steps of the theoretical definition. Let us consider the definition of $\Gamma(A,q,\tau,l)$. It is important to emphasize that this step establishes that the best covering of the set A must be used. As we mentioned before, in the usual box-counting algorithms, the covering is obtained through the introduction of a grid. This kind of covering is very simple and usually very different from the optimal cover required by the theory. The use of a grid distorts the $f(\alpha)$ curve. For instance, it allows the existence of many spurious boxes. On the other hand, the box-counting algorithm presented in this paper eliminates most of the ''spurious'' boxes, and provides a much better covering of the multifractal set.

II. OUR ALGORITHM

Our algorithm is a modification of an idea of Oiwa and Ferrara [11]. In Oiwa's algorithm the covering of the multifractal set is provided by boxes of the same size and orientation, but they are not arranged in a grid. Several of these coverings are constructed for the same set; every new covering is constructed from the old one through the partitioning (a partition is a real division in a region of space; the parts resulting from this division are isolated, and this partition remains in all subsequent new coverings constructed in the set) of the boxes. The size of the boxes on the new covering is half of the size of the boxes of the old covering, and the first covering is taken as a square box that contains the whole multifractal set. The boxes are partitioned if the interval of points inside them is bigger than half the size of the box. Otherwise, the box is shrunk, and no division is needed. The shrinking of a box eliminates the empty space and results in a much better procedure than piling boxes side by side.

A covering constructed with this algorithm for the Hénon attractor is shown in Fig. 2. The details of the algorithm will be described in the Appendix. The Hénon attractor is a two-dimensional set, so after each two subsequent sequences of partitions, we have a covering for the multifractal.

Since the coverings are constructed by partitioning, they are all correlated. These partitionings really divide the multifractal set, isolating each part, which can now be regarded as an individual new set, so the algorithm does not have a global notion of the geometry of the original fractal structure. Although Oiwa's algorithm does eliminate part of the empty space in the boxes, it still leads to the presence of many spurious boxes, and the resulting curves for $f(\alpha)$ and D_q are still deformed.

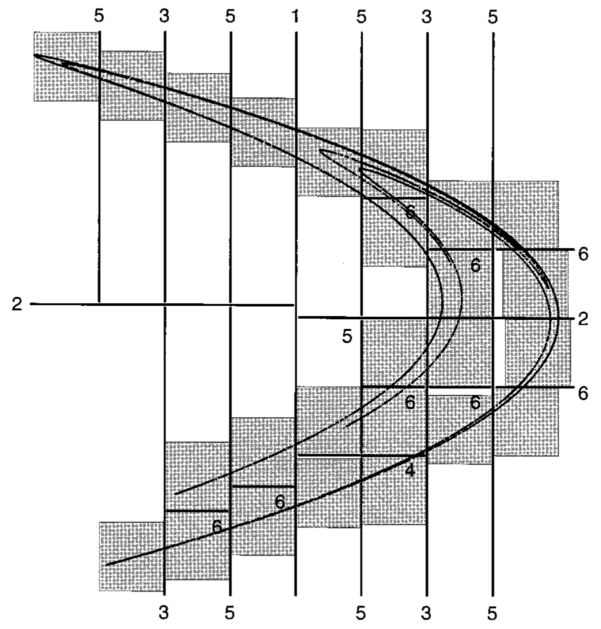

FIG. 2. Covering of the Hénon attractor according to Oiwa's algorithm. The sequence of the partitionings is indicated by the numbers. The boxes of the covering are adapted to the geometry of the set. Although having the same size and orientation, they do not belong to a grid.

We now propose a procedure to further reduce the spurious boxes on the covering. According to our procedure, a spurious box is removed by moving a whole group of boxes. Although the properties of these groups of boxes may be regarded, at first sight, as very restrictive, we will see that boxes with these properties are in fact very much usual. Clusters that cannot be treated by our algorithm are described in the Appendix.

The boxes that we are able to move in order to eliminate the spurious boxes (see Fig. 3) form a group with the following properties.

(1) The boxes of the group are placed side by side, along the same direction of the space, as in a line embedded in that space.

(2) Each box of the group has only two neighboring boxes, one at each side, except for the boxes at the end of the group, which might have only one neighbor.

(3) The boxes at the end of the group must have some empty space inside them. This empty space must be located at the side of the box that has no neighbor, that is, at the end of the group. The sum of the empty space present in the boxes located at the end of the group has to be greater than the size of the box, so that one box can be eliminated by moving the whole group.

An example of this new step is given in Fig. 4, where we have the same covering shown in Fig. 2, after the application

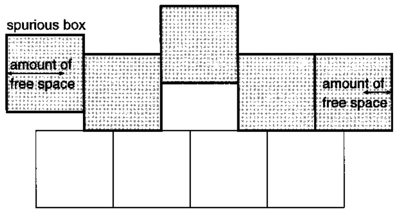

FIG. 3. Two-dimensional space: Example of the group (gray) treated by the algorithm.

of this procedure. The boxes that were moved are shown in thick lines.

The future partitions are made on this new corrected covering. Then, the new covering is corrected again and again, until we obtain a covering with boxes as small as desired for that particular multifractal set.

The algorithm is useful for all sets that are represented by points and, as already stated, it increases the accuracy of the calculations of the negative branch of $f(\alpha)$. We must point out, however, that in very particular situations it may be better to use the usual approach. That will be the case of the application of the method of histograms in sets with an extremely rough distribution of probabilities (such as Cantor sets embedded in one-dimensional space). When applying the method of histograms for those sets, the average among different positions of the grid becomes an essential step in the calculation of $f(\alpha)$. It smooths the distribution, which leads to a smooth $f(\alpha)$ curve. The algorithm we propose tries to adapt itself to the set, generating a unique distribution of probabilities. This distribution is closer to the ideal one. As it is made by peaks, it is very irregular and consequently any method that makes use of histograms will lead to irregular $f(\alpha)$ curves.

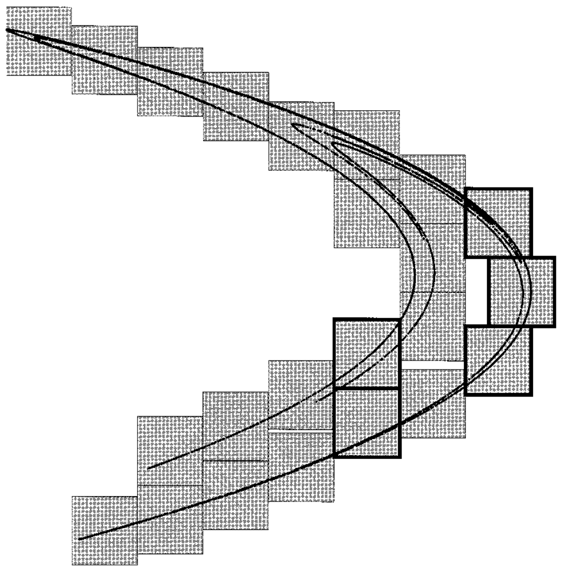

FIG. 4. Covering constructed by our algorithm. The boxes are the same as in Fig. 2, except those in thick lines, that have been moved. This translation eliminates spurious boxes from the covering, and corrects the distortions of the results.

III. APPLYING OUR ALGORITHM

There are several methods that use box-counting algorithms to evaluate $f(\alpha)$ and D_q. In this paper we have used the canonical method [5] to evaluate the $f(\alpha)$ function and observe the effects of our algorithm. The canonical method uses a box-counting algorithm to obtain the probabilities p_i associated with the covering of a set. The equations of the canonical method are given by

$$\mu_i(\epsilon,q)=\frac{p_i^q(\epsilon)}{\Sigma_j p_j^q(\epsilon)}, \tag{1}$$

$$\alpha(q)=\lim_{\epsilon\to 0}\frac{\Sigma_i p_i \ln\mu_i}{\ln\epsilon}, \tag{2}$$

$$f(\alpha(q))=\lim_{\epsilon\to 0}\frac{\Sigma_i \mu_i \ln\mu_i}{\ln\epsilon}, \tag{3}$$

where ϵ is the size of the boxes.

Using the curves $\Sigma_i p_i \ln\mu_i$ and $\Sigma_i \mu_i \ln\mu_i$ as a function of $\ln\epsilon$, we finally get $f(\alpha)$ in a parametric way.

Three sets were used to test our algorithm: (i) the attractor of the logistic map [12] $x_{i+1}=Ax(1-x)$, with $A=3.569\,946$; (ii) a binomial measure [12] with $l_1=0.4$, $l_2=0.35$, $p_1=0.6$; and (iii) the attractor of the Hénon map [13], $x_{i+1}=1-Ax_i^2+y_i$ and $y_{i+1}=Bx_i$, with $A=1.4$ and $B=0.3$.

The first and second sets are one dimensional, and were chosen because they are rare examples of sets for which it is possible to calculate $f(\alpha)$ theoretically. Once the groups of boxes we are able to move in our correction procedure are of a restricted type, the algorithm cannot eliminate all the spurious boxes it finds. So, we made different coverings using different orientations of boxes, and consider in the final calculation that one which corresponded to the inf/sup criteria of step 2 of the theoretical definition of $\tau(q)$. Also, we have studied one-dimensional sets embedded in a two-dimensional space to be able to consider different orientations of the coverings.

The binomial measure was represented by 200 000 points, and the attractor of the logistic map by 300 000. The parameters of the method, in both sets, were $q=0,\pm1,\pm2,\ldots,10$. $\epsilon=2^{-i}$, $i=1,2,\ldots,10$. Fifty-one displacements of the grid were used in the usual box counting, and 51 rotations of boxes in our box counting, in an angle of 45°. The points of the binomial measure were generated by an iterated function system [14]. The results for the attractor of the logistic map are shown in Fig. 5. For the binomial measure, they are shown in Fig. 6.

The two-dimensional set of Hénon attractor was represented by a group of 2×10^5 points. The parameters of the method were taken as $q=0,\pm1,\pm2,\ldots,\pm10$, and $\epsilon=2^{-i}$, $i=1,2,\ldots,8$. As in the one-dimensional case, 97 displacements of the grid were used in the usual box counting, and 97 rotations of boxes, by an angle of 90°, in our box counting. There is no analytical solution for the attractor of the Hénon map, so we compare our results with the $f(\alpha)$ obtained from an application of the fixed mass method [15] (which has been chosen because it does not require the

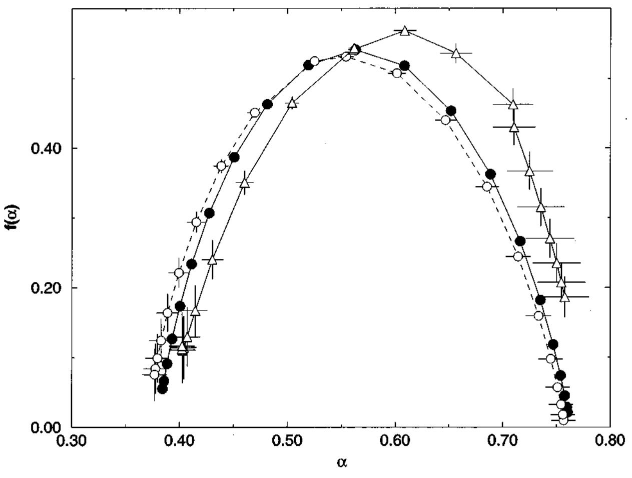

FIG. 5. $f(\alpha)$ curve: Attractor of the logistic map in the onset of chaos. Black circles: theoretical results. White circles: our box-counting algorithm. White triangles: usual box-counting algorithm.

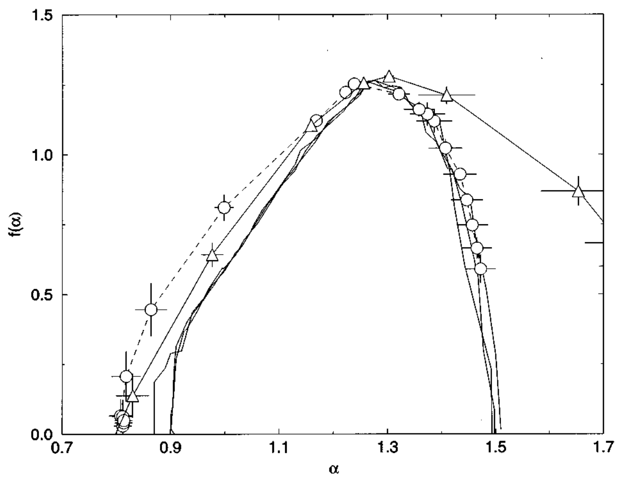

FIG. 7. $f(\alpha)$ curve: Attractor of the Hénon map $x_{i+1}=1.0-Ax_i^2+y_i$ and $y_{i+1}=By_i$ with $A=1.4$ and $B=0.3$. Solid lines: fixed mass method. White circles: our box-counting algorithm. White triangles: usual box-counting algorithm.

calculation of the probabilities of the set using a box-counting algorithm). We need to point out that, although in the case of the Hénon map (see Fig. 7) our method leads to results better than the ones obtained by the usual box counting, and comparable to the results obtained by the fixed mass approach, for sets with a very irregular distribution of probabilities (as Cantor sets) the fixed mass approach does not work well. In these cases our algorithm combined with the canonical or microcanonical approach will lead to better results.

Most of the results obtained with our box-counting algorithm are much better than the results obtained with the usual algorithm. The results are still different from the analytical predictions, because of various effects. The plots of $\Sigma_i p_i \ln\mu_i$ and $\Sigma_i \mu_i \ln\mu_i$ were not perfectly straight lines; their points fluctuate around a line (these fluctuations are usually associated with an effect of lacunarity). The sizes of boxes in the algorithm were always equal, so, for instance in the generalized Cantor set, the correct probabilities of the prefractals are not reached. In the attractor of the Hénon map, there is a considerable difference between the results obtained with our and the usual algorithm, because in two dimensions the richness of the structure of the set can hardly be noted using a grid. The different arrangements of the boxes in the coverings used by our algorithm may produce better results in higher dimensions. For all these calculations, the processing time was small. In Table I we show how the CPU time and the amount of memory needed to apply our algorithm to the Hénon map (with standard parameters $A=1.4$ and $B=0.3$) increases with the number of points of the set. Those results were obtained with a SPARC 2. The coding of our algorithm is complex, but not all of its parts are necessarily processed, so it is still fast, even when compared with correlation integral methods [16,17].

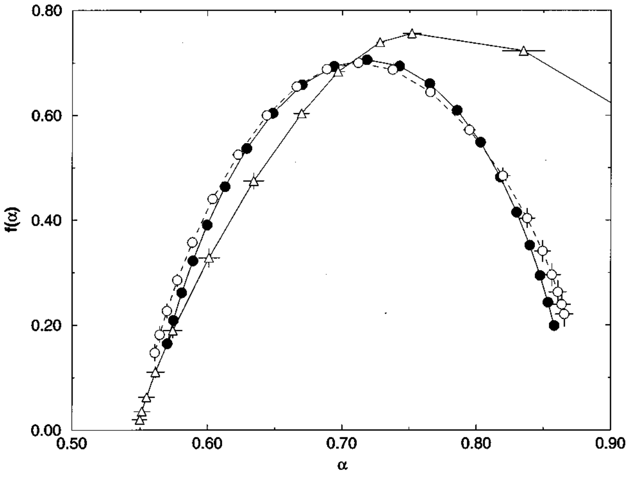

FIG. 6. $f(\alpha)$ curve: Binomial measure, $l_1=0.4$, $l_2=0.35$, $p_1=0.6$. Black circles: theoretical results. White circles: our box-counting algorithm. White triangles: usual box-counting algorithm.

IV. COMMENTS AND CONCLUSIONS

The box-counting algorithm based on a ''smart covering'' improves the calculation of $f(\alpha)$ and D_q very significantly. The new covering adjusts to the geometry of the set, in agreement with the ideas of the theoretical definitions. The significant improvement in the final results confirms our conjecture that the existence of spurious boxes in the covering of the traditional box-counting algorithms is the main source of error in the $q<0$ branch of the curves. Although the procedure of codification is a bit long, it is important to point out

TABLE I. CPU times and memory for applying the new algorithm to the Hénon attractor in a SPARC 2. All the parameters were kept constant except for the number of points.

Number of points	CPU time (h:s)	Memory (Mb)
100 000	1:23	11.8
500 000	6:34	16.9
100 0000	14:43	22.5
150 0000	19:17	28.7

Box Counting Algorithm — First Part

```
Repeat $dim$ times
    Repeat for all boxes $boxold$ of the covering
        Choose one component $comp$ of the space in the box $boxold$. We have
        chosen the one that has $table[comp] = 0$ and the biggest
        $interv = pntmax[comp] - pntmin[comp]$.
        Is the interval $interv$ bigger than $size/2$ ?
        Y:
            The box will be partitioned in component $comp$.
                This part is described in fig. 9.
        N:
            The box will be shrunk in component $comp$. A copy
            $boxnew$ of the box $boxold$ is added to the new covering.
            It will appear an empty space.
            $boxnew \rightarrow scprgt[comp]$ is increased by $size/2$.
        The boxes are centralized in the interval of points at this stage. This is
        done in a similar way of the steps of dividing the box. The variables
        $boxpos[comp]$, $spclft[comp]$ and $spcrgt[comp]$ may be changed.
        $table[comp]$ is set to 1. This indicates that this size of the box is
        smaller. It is necessary to check if the box is square. It is square if all
        values of $table[]$ are equal to one. In this case, $size$ is divided by two,
        and all the values of $table[]$ are set to zero. Otherwise, they are left
        unchanged.
    Next, we erase the old intermediate covering. Rename the new
    intermediate covering old and repeat all the steps to create a new
    intermediate covering. We call them intermediate covering because there
    may have rectangular boxes.
```

FIG. 8. Nassi-Schneiderman diagram of the first part of the algorithm.

that it is not time consuming, and that the final results are rewarding.

The first part of the algorithm can be used to deal with higher dimensions in a trivial way. In the second part, however, we have to introduce some modifications to be able to treat other kinds of clusters. The possibility of extending our algorithm to higher dimensions has to be further investigated to check whether a single kind of cluster of boxes could eliminate the majority of spurious boxes, as observed in two dimensions. In practice, these extensions are not frequently used, as all box-counting algorithms in dimensions higher than two use a larger amount of memory to store in the boxes. In the Appendix we describe in detail the process of coding our algorithm.

Box Counting algorithm — Dividing the box

```
Two new structures $boxlft$ and $boxrgt$, representing the new boxes, are
created. Their values are all equal to $oldbox$, except for
$boxlft \rightarrow spcrgt[comp] = boxrgt \rightarrow spclft[comp] = 0$, because they are
neighbors.

The box will be divided as near as is possible of the middle value of $interv$. A
variable $mute$ is created.
$mute = pntmin[comp] + interv/2 - (boxpos[comp] + size/2)$. If
$mute > spcrgt[comp]$, do $mute = spcrgt[comp]$. If $-mute > spclft[comp]$, do
$mute = -spclft[comp]$.

Both boxes must be shifted by $mute$. This means
$boxlft \rightarrow boxpos[comp] = boxold \rightarrow boxpos[comp] + mute$ and
$boxrgt \rightarrow boxpos[comp] = boxlft \rightarrow boxpos[comp] + size/2$. The spaces around
the boxes change too. $boxrgt \rightarrow spcrgt$ decreases by $mute$, $boxlft \rightarrow spclft$
increases by $mute$.

The points are separated into the right boxes. Points with coordinate in
component $comp$ smaller than the value $boxrgt \rightarrow boxpos[comp]$ belong to the
$boxlft$, otherwise belong to the $boxrgt$. This determines the values of $mypnt$ in
the two structures. It also changes $row[]$.

The interval that contains the points inside the boxes in all components of
space must be determined. $pntmin[]$ and $pntmax[]$ are defined in both
structures.
```

FIG. 9. Nassi-Schneiderman diagram of the box division procedure, used in the first part of the algorithm.

Box Counting algorithm — Constructing a cluster

```
For each box of the covering, repeat for all components of space.
    A half empty box $box[0]$ in the component $comp$ must be found. The direction in
    which the cluster will be moved depends on the position of the empty space inside
    this box. See item 1. Consider $i = 1$.
    Repeat
        In the component $comp$ and the direction defined above, neighboring boxes are
        searched for.
        Was one neighbor of $box[i-1]$ found ?
        Y:
            If only one neighbor box $box[i]$ is found, the number of neighbors of
            this box in the opposite direction of the shift of the cluster must be
            determined.
            Is the box $box[i]$ neighbor of only $box[i-1]$ ?
            Y:
                If $box[i]$ has only $box[i-1]$ as neighbor, the spaces in the
                transversal component $trans$ have to be checked. (See fig. 11a),
                and item 2.
                there is space in component $trans$ to fit a box ?
                Y:
                    The box $box[i]$ is accepted in the cluster. The cluster
                    can be finished if the sum of empty space inside $box[i]$
                    and $box[0]$ is enough to fit one box. See item 3.
                    $i = i + 1$.
                    Can the cluster be finished ?
                    Y:
                        the cluster is shifted.
                            See fig. 14.
                    N:
                N:
            N:
        N:
    Until the formation of the cluster, or a violation of its conditions.
```

FIG. 10. Nassi-Schneiderman diagram of the cluster construction procedure, used in the second part of the algorithm.

APPENDIX: DESCRIBING THE ALGORITHM

1. Overview

The program was written in C. Some notations and symbols used in the description below are familiar to C users. At first sight this description may seem to be a little complicated and difficult to codify, but the ideas used here are simple and the improvements in the results are significant.

In order to build our coverings, we start with one square box that contains the whole set. Based on this initial covering, the algorithm will construct a new covering of boxes with half the size of the previous one. This is achieved in two parts. In the first part (Oiwa's algorithm), a new covering is constructed by partitioning the space. In the second part, spurious boxes are eventually found and eliminated through the displacement of a cluster of boxes. This procedure is repeated until the size of each box in the covering is as small as desired for that specific set.

In the results presented in this paper, just one cluster is analyzed by us, which is very simple, but the majority of the spurious boxes of the covering belong to this kind of cluster.

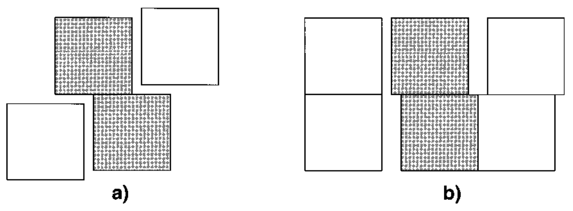

FIG. 11. (a) Step of construction of a cluster to eliminate spurious boxes. The cluster (grayed) cannot be replaced by a single box because of the position of the nearest boxes. (b) Correction of the space around the shifted box. At the left side of the shifted box, the boxes are aligned, so no change is needed. At the right side, the smallest distance to the nearest boxes has to be considered.

It consists of a chain of boxes along the same component of space that was used to determine that a box was spurious. These boxes are always placed side by side, and along this component they have as neighbors only boxes that belong to the cluster, except for the boxes at the end, that together should have enough empty space to permit the elimination of one box (see Fig. 3).

In the rest of this appendix we will describe the way we code this algorithm.

2. Description of the variables

The main variables of the algorithm are the ones that represent the boxes of the covering. Consider that the points of the set are saved in an array *points* [ ], and the dimension of the system is represented by the variable *dim*. Each box of the covering is represented by the set of eight variables described below. They form a structure in the context of C.

(1) *boxpos[dim]* represents the position of one corner of the box.

(2) *size* saves the biggest size of the box (it can be rectangular).

(3) *table[dim]* indicates, for each component, if the size is equal to the biggest one, or if it is a half of it (there are no other values for box size).

(4) *pntmin[dim],pntmax[dim]* represent the smallest rectangle that contains all the points that are inside the box.

(5) *spclft[dim],spcrgt[dim]* save the amount of empty space (without boxes) around the box in that direction.

(6) *mypnt* saves the position of only one of the points of *points* [ ] that are inside the box. The way we save the other

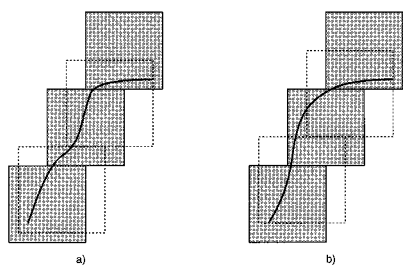

FIG. 12. Two-dimensional space: After the movement of the cluster, it is necessary to check if there are overlappings of boxes or uncovered points. In (a) the distribution of points in the middle box of the cluster (gray) makes it possible to replace the cluster by two boxes (dotted). In (b) this is not possible.

Box Counting algorithm — shifting a cluster

```
If the direction to move the cluster is to the left, invert all the positions of box[]. Now all the clusters will move to the right.

The displacement disp that will be applied to the cluster must centralize it, taking into account the interval of points inside it. $disp = box0 \rightarrow (pntmin[comp] - boxpos[comp]) - box[N] \rightarrow (pntmax[comp] - bospos[comp])/2$. Let $i = 1$

    The boxes have to be shifted in couples, $box[i]$ and $box[i+1]$, because points in one box may have to pass to its neighbor. Add disp to $box[i] \rightarrow boxpos[comp]$; $box[i] \rightarrow boxpos[comp] + size$ will be the limit used to separate the points. If the point in $box[i+1]$ has its value in the comp component greater than this limit, it will stay in the box. Otherwise, it will be moved to $box[i]$. This division of point will change row and may change $box[i+1] \rightarrow mypnt$.

    Recalculate the values of pntmin[] and pntmax[] of $box[i]$, that are the intervals that contain all points inside the box. Do $i = i + 1$.

Repeat for all couples of neighboring boxes that belong to this cluster, or until $i = N - 1$

The last box, $box[N]$, must have no points now. This is the box to be eliminated. The pointers of the covering that points to this box have to be changed.

Because of the elimination of one box, an empty space will appear. $box[0] \rightarrow spclft[comp]$ will be increased by disp, and $box[N-1] \rightarrow spcrgt[comp]$ will be increased by $size - disp$.

May be necessary to modify the spaces in the transversal component of the new boxes, see fig. 11b. The boxes in the left side of the cluster are aligned, so no change is needed there. In the right side, though, the smallest distance to the nearest boxes have to be placed in the variable spcrgt[trans].

The boxes now must be centralized in the interval of points that exist into them. This may change boxpos[], spclft[] and spcrgt.
```

FIG. 13. Nassi-Schneiderman diagram of the cluster shifting procedure, used in the second part of the algorithm.

points of the box will be explained below.

A covering is a chain of these structures. Each structure has a pointer to the next box of the covering and to the previous one.

The subset of points that belong to each box is kept using the same method presented by Grassberger [18]. We use an array *row* [ ] that has the same size of *points* [ ]. This array contains integers that represent positions in the array *points* [ ]. They are pointers to points of the set. For example, consider a box that contains the whole set. We choose $mypnt=0$ and $row[i]=i+1$, except for the last value, that should contain -1. If one wants to know which points belong to this box, just look at *mypnt*. It says that the point 0 belongs to the box. The next point that belongs to the box is $row[mypnt]=row[0]=1$. The next one is $row[row[mypnt]]=2$, and so on, until the value -1 is reached. It indicates that there are no more points in that box. The data in *row* [ ] must be updated to keep the information of the distribution of points in every cover.

3. The first part of the algorithm: Oiwa's algorithm

The algorithm starts defining the values of the points of the set (*points* [ ]) and the dimension of the set (dim). The points can be proportionally distorted to fit in a single square box that must be constructed with the values described above.

The first part of the algorithm consists of the steps shown in Figs. 8 and 9. The covering shown in Fig. 2 was constructed using only this part of the algorithm.

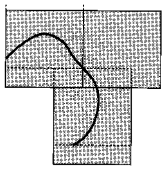

FIG. 14. Two-dimensional space: Example of a cluster that has not been treated by the algorithm. The solution for this case should be a vertical shift (dotted) of the cluster of three boxes (grayed).

4. The second part of the algorithm: Reducing the number of spurious boxes

The second part of the algorithm is divided in two steps. In the first step, a group of boxes is defined. This group will be moved in the second step of the algorithm, eliminating one spurious box. These two steps must be repeated for all components of the space and for all spurious boxes detected in the covering.

The algorithm depends on the dimension *dim* of the set. We will give a description in the two-dimensional space. In this part of the algorithm, it is necessary to find a cluster of boxes to be shifted, so a group of neighboring boxes has to be localized. We used a usual box-counting algorithm to do this. Consider the set of box positions *boxpos* [ ], and cover this set with a grid of boxes that have same size *size*. The boxes of the covering will be associated with only one box of the grid, and in this grid it is easy to find neighboring boxes. Considering two boxes for which the correspondent boxes in the grid are neighbors, we can make use of the distance between them to determine if they are really neighbors. An ordered list that associates the label of each box in the covering with the pointer of that box in the grid must be created.

We will represent the cluster by an array of structures *box* [ ]. The steps for constructing this structure are presented in Fig. 10. Some relevant details of the procedure of these steps are described below.

(1) The direction of the shift of the cluster is determined by looking at the position of the empty space of the box *box[0]*. If *pntmin[comp]-boxpos[comp]>size/2*, the shift that will be applied to cluster is to the left. If *pntmax[comp]-boxpos[comp]<size/2*, the shift will be to the right. In both cases the box is also half empty.

(2) The examination on the component *trans* is done by considering the distance between the corner of the boxes that form the cluster, *dist=box[i-1]→boxpos[trans]-box[i] → boxpos[trans]*. If *dist* is positive, the space for the box that will replace *box[i]* and *box[i-1]* is *dist=size-dist + box[i]→spcrgt[trans]+box[i-1]→ spclft[trans]*. Otherwise, *dist=size-dist+box[i-1] → spcrgt[trans]+box[i] →spclft[trans]*. If *dist* is greater than *size*, there is enough space for the new box [see Fig. 11(a)].

(3) To finish the cluster, the following conditions must be true: If the direction in which the cluster will be moved is to the right, *box[0]→(pntmin[comp]-boxpos[comp])-box[i]→ (pntmax[comp]-boxpos[comp])>0*. Otherwise, it is *box[i] →(pntmin[comp]-boxpos[comp])-box[0]→ (pntmax[comp]-boxpos[comp])>0*.

In the two-dimensional space, moving a cluster can break the conditions of a valid covering because it can produce overlapping of boxes or can leave uncovered points. That problem cannot be determined beforehand (see Fig. 12), so the previous values of the variables that define the cover, the boxes of the cluster, and *row*[ ], must all be saved. If some problem occurs after the shifting of the cluster, the state of the variables before it can be restored.

To move this cluster of N boxes, we will proceed according to the steps shown in Fig. 13.

Now, the probabilities assigned to the boxes can be determined. For instance, the sums $\Sigma_i p_i \ln\mu_i$ and $\Sigma_i \mu_i \ln\mu_i$ can be calculated for this size of box if the canonical method is applied in the multifractal set.

As already mentioned, we do not consider all clusters containing spurious boxes. In Fig. 14 we show a basic cluster that is not treated. The clusters that we have used are similar to one-dimensional structures in a two-dimensional space. There are other kinds of clusters that can be treated by the algorithm, but the clusters which we have considered are much more numerous, so the numerical results were not so compromised by this restriction.

[1] J. Feder, *Fractals* (Plenum, New York, 1988).
[2] G. Paladin and A. Vulpiani, Phys. Rep. **156**, 148 (1987).
[3] J.L. McCauley, Phys. Rep. **189**, 225 (1990).
[4] F. Argoul, A. Arneodo, and G. Grasseau, Z. Angew. Math. Mech. **68**, 519 (1988).
[5] A.B. Chhabra, C. Meneveau, R.V. Jensen, and K.R. Srenivasan, Phys. Rev. A **40**, 5284 (1989).
[6] C. Meneveau and K.R. Srenivasan, Phys. Lett. A **137**, 103 (1981).
[7] A. Block, W. von Bloh, and H.J. Schellnhuber, Phys Rev. A **42**, 1869 (1990).
[8] P. Grassberger, Int. J. Mod. Phys. C **4**, 515 (1993).
[9] T.C.A. Molteno, Phys. Rev. E **48**, R3263 (1993).
[10] M. Yamaguti, Master thesis, Universidade de São Paulo, Brazil, 1992.
[11] N. Oiwa and N.F. Ferrara (unpublished).
[12] T.C. Halsey, M.H. Jersen, L.P. Kadanoff, I. Procaccia, and B.I. Shraiman, Phys. Rev. A **33**, 1141 (1986).
[13] M. Hénon, Commun. Math. Phys. **50**, 69 (1976).
[14] M.F. Barnsley and S. Demko, Proc. R. Soc. London Ser. A **399**, 245 (1985).
[15] R. Baddi and G. Broggi, Phys. Lett. A **131**, 339 (1988).
[16] H.G.E. Hentschel and I. Procaccia, Physica D **8**, 435 (1983).
[17] F.H. Ling and G. Schmidt, J. Comput. Phys. **99**, 196 (1992).
[18] P. Grassberger, Phys. Lett. A **148**, 63 (1990).