JOURNAL OF COMPUTATIONAL PHYSICS **129,** 30–40 (1996)
ARTICLE NO. 0231

Implementation of Hierarchical Clustering Methods

Arturo Serna

LAEC, Observatoire de Paris-Meudon, F-92195 Meudon, France

Received December 4, 1995; revised May 21, 1996

We present an implementation of hierarchical clustering methods to distribute a set of objects into a set of groups. Our code is particularly conceived to identify and analyze substructures in galaxy clusters or in large-scale catalogues. However, its general scheme allows for very easy adaptation to any other kinds of systems and physical problems. The algorithms to draw the hierarchical tree associated to a given sample of data, as well as those for analyzing and interpreting the results obtained from this technique, are also presented here.

I. INTRODUCTION

Several problems in physics require one to determine how objects in a sample are gathered into groups of a specific nature. For example, much effort is currently devoted to the detection of substructures in galaxy clusters, as well as in large-scale catalogues. Both topics are of major importance in cosmology because they may provide us with very valuable information on the cosmological scenarios of formation of clusters and/or on the internal dynamics of these systems. Several works show that the existence of a substructure in a galaxy cluster can be related to intrinsic parameters such as its evolutionary status, and the internal amount and distribution of dark matter [1].

The development of methods to detect and identify groups in a sample of data is therefore necessary to link observations with theoretical predictions and models. A huge number of papers have been devoted to this subject, and numerous methods of substructure detection have been proposed in the past. In astrophysics, most of these methods are however purely phenomenological and they merely look for coincidences in the positions and/or the velocities or galaxies.

A different approach in attempting to identify subgroups defined by their dynamical meaning has been recently proposed by Serna and Gerbal [2] (hereafter referred to as paper I). This method is based on the mathematical theory of clustering analysis, whose basic objective is often stated as sorting the observations into groups such that the degree of ''natural association'' is high between members of the same group and low between members of different groups.

Clustering analysis provides different mathematical methods to distribute a set of objects into a set of groups. Among these methods, hierarchical clustering gives the clearest insight into the structure of a cluster and allows a relatively easy identification of the members of a group. This kind of method can be formulated in terms of operational concepts, and it has been applied in the field of astronomy by Materne [3], Tully [4, 5], Gourgoulhon *et al.* [6], and Serna and Gerbal [2].

The aim of this paper is to present an implementation of the hierarchical clustering methods. Special attention will be paid to identifying and separating those points which depend on the nature of the physical problem under consideration. Thus, although our code will be applied to identify substructures in galaxy clusters, its adaptation to any other physical system should be very simple. Some examples found in the literature of other physical problems requiring a grouping analysis are clustering of electrons and holes in disordered solids, signal taxonomy studies, and identification of patterns in CCD images.

The paper is organized as follows: We describe in Section 2 the basic aspects of a hierarchical clustering method. The description of the main body of our code is then given in Section 3. The algorithms to draw the resulting hierarchical tree are presented in Section 4, while the procedures to analyze and interpret the results are given in Section 5. Finally, some additional comments are discussed in Section 6.

II. HIERARCHICAL CLUSTERING METHODS

A hierarchical method is a grouping scheme in which each cluster is obtained through mergers or agglomerations of smaller ones. In other words:

1. For a given sample of N objects (e.g., galaxies), this kind of method initially considers each object as constituting a group by itself, that is, it starts by considering a collection of N groups: $(\{G_1\}, \{G_2\}, ..., \{G_n\})$.

2. According to some specified criterion, we must then find the two most related groups, that is, those which most probably belong to the same structure. These two groups are then considered as actually constituting a unique group.

0021-9991/96 $18.00

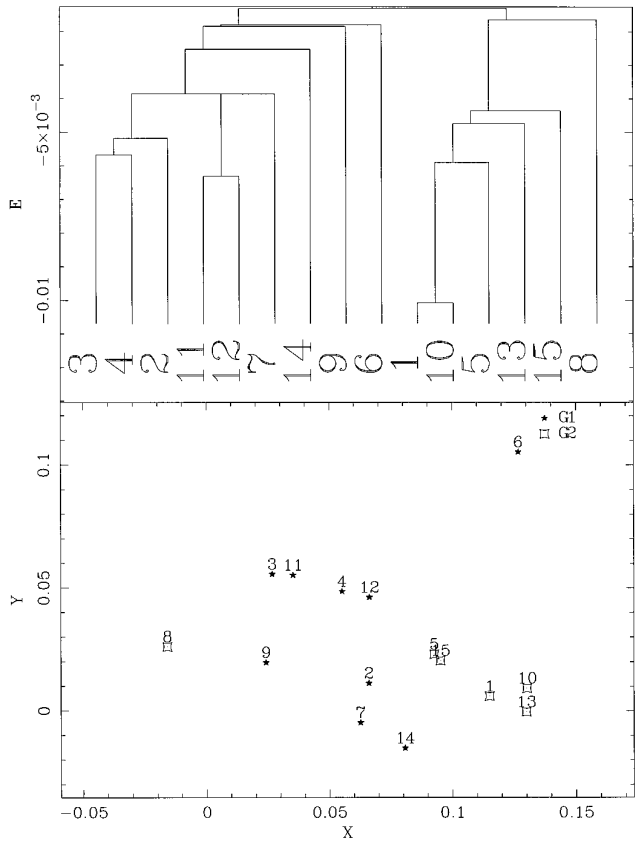

FIG. 1. Example of the hierarchical tree (upper panel) corresponding to a system as that visualized in the lower panel.

They are then replaced, or "merged," by a simple group $G_{kl} = \{G_k, G_l\}$. So we are left with $N - 1$ groups: ($\{G_1\}$, $\{G_{kl}\}$, ..., $\{G_{n-1}\}$).

3. The merging procedure is repeated until only one cluster is left, G_1, which contains all the N objects of the sample.

The merging sequence can be easily visualized under the form of a hierarchical tree (h-tree). Such a graph shows the smallest groups (or pairs) to the largest structures including several subgroups (see Fig. 1).

Obviously, the hierarchical method described above can be applied to several kinds of problems. In order to construct a particular grouping method, we must specify:

1. What kind of groups do we want to find?; that is, what is our group definition?

2. Which is the physical parameter that best characterizes the degree of association between the groups defined in that way?

3. Which is the most appropriate merging criterion, in terms of the above parameter?

An objective "group" definition is often difficult to find. For example, substructures in galaxy clusters can be defined in several different ways. In paper I, we consider the fact that the membership of a structure must be much more stable, against the dynamical evolution of the cluster, than that of any random subset of galaxies without a real dynamical meaning. As a matter of fact, strongly bound structures are less sensitive to tidal forces and, hence, their lifetimes are typically longer than those of weakly bound galaxy subgroups. We then decided to call "subclusters" those groups of galaxies presenting such a relative increase in their membership lifetime.

Provided with a definition of the "substructure" we must now decide, in a collection of N groups, the two which most probably belong to the same structure. The conventional approach is to compute a measure of association for each one of the $\frac{1}{2}N(N-1)$ different pairs of objects. This association measure, s, is often called the *affinity parameter*. We have shown (paper I) from N-body simulations that gravitationally bound groups (with a relative increase in the lifetimes of their memberships) are very well characterized when the binding energy E_{ij} is used as the affinity parameter:

$$E_{ij} = -G\frac{m_i m_j}{|r_i - r_j|} + \frac{1}{2}\mu_{ij}(v_i - v_j)^2, \tag{1}$$

where m_i, r_i, and v_i denote the mass, position, and velocity of galaxy i, while

$$\mu_{ij} = m_i m_j/(m_i + m_j) \tag{2}$$

is the reduced mass of i and j.

The choice of s is not enough, however, to determine the hierarchy of relationships among the objects of a sample. At intermediate stages of the merging sequence, some groups are in fact constituted by several objects, while the affinity parameter is defined for individual objects. Consequently, it is also necessary to specify a linkage method, that is, some criterion consistent with our choice for s but giving the association measure between multielement groups. This linkage method can be used to construct a similarity array describing the strength of all the relationships among the groups existing at a given stage of the merging sequence. The clustering analysis theory provides three different linkage techniques to construct the affinity array S_{ij} for a given choice of s:

1. *Single linkage.* Each group is characterized by the largest s value needed to connect any member of the merged group to some other member of that group:

$$S_{ij} = \min_{\substack{\mu\in G_i \\ \nu\in G_j}} (E_{\mu\nu}). \tag{3}$$

2. *Complete linkage.* Each group is characterized by the largest s value needed to connect every member of the merged group to every other member of that group:

$$S_{ij} = \max_{\substack{\mu \in G_i \\ \nu \in G_j}} (E_{\mu\nu}). \tag{4}$$

3. *Centroid clustering.* Each merged group is characterized by the value of s between the averages or mass center properties of its "parent" groups

$$S_{ij} = E_{\overline{G}_i \overline{G}_j}, \tag{5}$$

where $E_{\overline{G}_i \overline{G}_j}$ is the affinity (relative energy) between the average of $i \in G_i$ and that of $j \in G_j$.

A comparative study of the main properties of these different linkage methods has been extensively performed in the literature (see, e.g., Ref. [7]). The single-linkage method is one of the very few techniques which can outline clusters with a non-ellipsoidal form, and the resulting h-tree is invariant to any monotonic transformation of the affinity parameter. It also has interesting connections with certain aspects of graph theory, as the problem of finding the minimum spanning tree. Its main disadvantage is that it can fail to distinguish between two groups poorly separated in the s space. Complete linkage is also invariant to monotonic transformations of the affinity measure. In contrast to the single-linkage method, it can only be interpreted in terms of the relationships within individual groups, while the differences between groups are not determined very reliably. Although it has some aspects common to a maximally connected subgraph, it has no special interpretation in graph theory terms. Finally, the centroid method and its variants have the great disadvantage that reversals can occur, that is, the S_{ij} value corresponding to the groups to be merged may rise and fall from stage to stage of the hierarchical merging sequence. When these reversals occur, the resulting h-tree has no meaningful interpretation.

III. IMPLEMENTATION OF *H*-METHODS

We will describe now our computational implementation of the hierarchical clustering method. The general scheme of such a program consists of three main parts (see Fig. 2):

1. Calculation of all the information needed to extract, from a given sample, its corresponding hierarchical tree.
2. Drawing the resulting hierarchical tree.
3. Analysis of that tree in order to identify the groups existing in the sample as well as to estimate the significance and the main physical features of such groups.

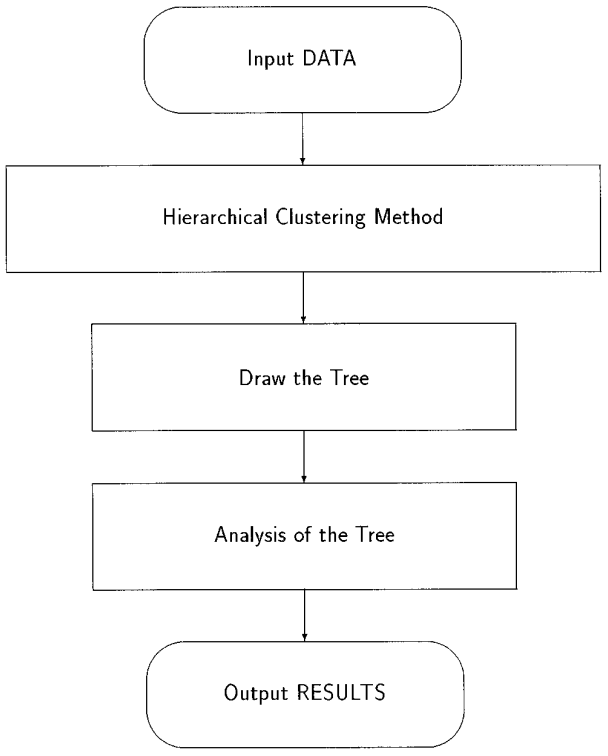

FIG. 2. General diagram of a hierarchical clustering analysis.

A. *Hierarchical Clustering Method*

The degree of complexity of a hierarchical tree can be very high, especially for samples containing a large number of objects and subclusters. Because of this, its numerical processing seems a priori rather complex. Fortunately, all the information contained in a hierarchical tree can be stored and easily extracted by using a *merge list.* For example, in the pioneering implementation of these methods proposed by Anderberg [7], the merge list was a $N \times 7$ array where he had to specify the different paths to move within it.

We propose instead a more simple merge list consisting of just four columns, $\mathbf{M}_{\text{list}}^{k,l}$ ($k = 1, ..., N$; $l = 1, ..., 4$). Its kth row corresponds to the kth stage in the merging sequence. The information stored in its four l columns is the following:

1. m, the identification number of the greatest of the two merged clusters at the kth stage. This number will identify the resulting merged group.
2. n, the identification number of the smallest of the two merged clusters at the kth stage.
3. N_m, number of objects in cluster m.
4. N_n, number of objects in cluster n.

TABLE I

Merge List for the *H*-Tree of Figure 1

k	m	n	N_m	N_n	k	m	n	N_m	N_n
1	1	10	1	1	8	11	7	2	1
2	11	12	1	1	9	3	11	3	3
3	1	5	2	1	10	3	14	6	1
4	3	4	1	1	11	3	9	7	1
5	3	2	2	1	12	3	6	8	1
6	1	13	3	1	13	1	8	5	1
7	1	15	4	1	14	3	1	9	6

In addition, the affinity between the two merged groups at each stage is stored in the real array $\mathbf{s}^j_{\text{val}}$ ($j = 1, ..., N$).

For example, the merge list corresponding to the tree displayed in Fig. 1 is that shown in Table I. It is straightforward exercise to see how Fig. 1 leads to Table I, and vice versa.

In order to construct the merge list, we proceed as indicated in the flow diagram shown in Fig. 3. That is:

1. We initially compute the array $\mathbf{S}^{i,j}_{\text{pair}}$ containing the value of the affinity parameter s_{ij} between all the different pairs of individual objects in the sample. If we are interested in finding substructures in galaxy clusters, a suitable choice for s_{ij} is that given by Eq. (1).

2. At each stage of the merging sequence, we compute the affinity array $\mathbf{S}^{i,j}_{\text{group}}$ containing the value of S_{ij} for all the different pairs of groups existing at that stage. In the case of galaxy groups, the best choice of S_{ij} is that obtained by using the single-linkage of Eq. (3).

3. We extract the two groups k, l presenting the highest affinity, that is, those which satisfy

$$\mathbf{S}^{k,l}_{\text{group}} = \min_{ij}(\mathbf{S}^{i,j}_{\text{group}}) \equiv \mathbf{s}^m_{\text{val}}. \tag{6}$$

4. These two groups are replaced by a single group, so that we are left with $N - 1$ groups. The array which contains the elements belonging to each group is then updated by setting $G_k \cup G_l \rightarrow G_k$ and $G_l \rightarrow 0$. the $\mathbf{M}_{\text{list}}$ array is also updated by setting

$$\mathbf{M}^m_{\text{list}} = (k, l, N_k, N_l),$$

where N_k and N_l are the numbers of objects in the groups G_k and G_l, respectively.

5. If more than a group is left, we repeat this procedure from the step 2.

The search for the next merging pair and the subsequent updating of the affinity array is the main computational burden for a hierarchical clustering analysis. It is then desirable to minimize the computational effort expended in these two steps.

Obviously, a first fact that we should use is that both the affinity parameter and the affinity array are generally symmetric: $s_{ij} = s_{ji}$, $S_{ij} = S_{ji}$. This is the case in which s_{ij} is the binding energy E_{ij}, and S_{ij} is obtained through a single linkage. Then, we need only to compute the triangular arrays $\mathbf{S}^{i,j<i}_{\text{pair}}$ and $\mathbf{S}^{i,j<i}_{\text{group}}$.

On the other hand, the computation of $\mathbf{S}_{\text{pair}}$ is only needed at the beginning of the clustering sequence. Since at that first stage all the individual objects are considered as groups, the affinity array between groups, $\mathbf{S}_{\text{group}}$, will be identical to $\mathbf{S}_{\text{pair}}$. At subsequent stages, $\mathbf{S}_{\text{pair}}$ remains obviously unchanged while $\mathbf{S}_{\text{group}}$ must be updated in order to take into account the modifications due to the last merging of groups. Although the number of elements of $\mathbf{S}_{\text{group}}$ is proportional to N^2, it is important to recall that the elements $\mathbf{S}^{i,j}_{\text{group}}$ with i different from i_{last} (= last merged

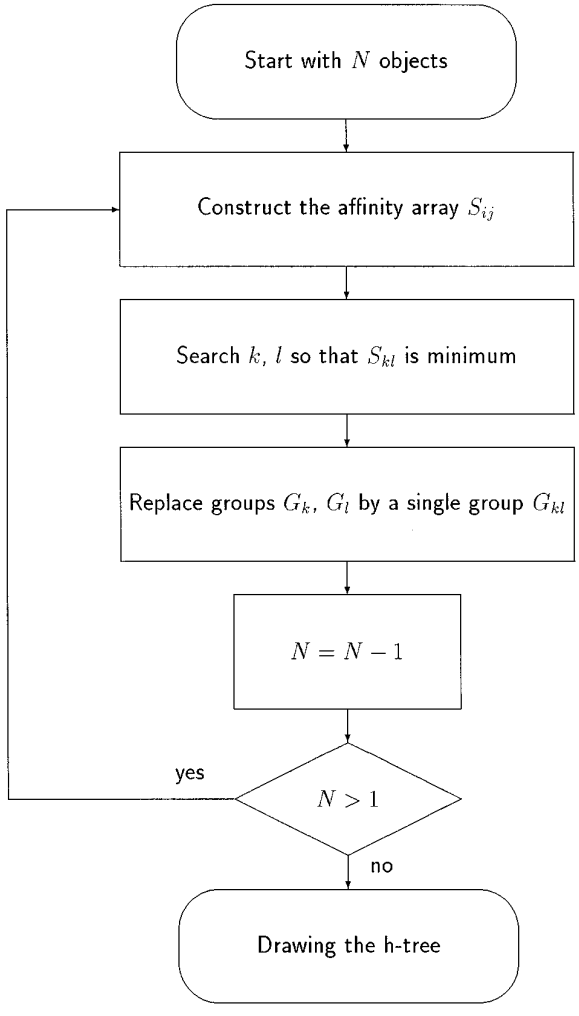

FIG. 3. Flow diagram corresponding to the hierarchical method.

group) need not be recalculated. Consequently, the computing time expended by our algorithm in the updating of $\mathbf{S}_{\text{group}}$ just grows proportionally to the number of groups existing at that stage.

We give in Figs. 4 and 5 the listing of our main algorithm for hierarchical clustering methods. The merge list $\mathbf{M}_{\text{list}}$ as well as the array $\mathbf{S}_{\text{group}}$ is calculated in subroutine *merging.* At each stage, this subroutine also updates the array $\mathbf{N}^{i,j}_{\text{groups}}$, which contains the identification of the objects belonging to each existing group, and calls to the subroutine *linkage,* where the affinity array $\mathbf{S}_{\text{group}}$ between groups is updated. The variables that we must provide to *merging* are: the number of objects in the sample, N, and their coordinates. For a galaxy cluster, such coordinates are the masses (*body*), positions ($x0$), and velocities ($v0$) of each of its N objects. Evidently, the catalogues of galaxies only contain apparent magnitudes, angular positions, and redshifts of their objects. Such data must be previously read and transformed into the mentioned coordinates, which in turn requires specifying the Hubble constant and the mass-to-luminosity ratio of the cluster. As output variables, subroutine *merging* gives the merge list, $\mathbf{M}_{\text{list}}$; the association measure between the two merging groups at each stage, $\mathbf{s}_{\text{val}}$; as well as the identification of the elements which constitute the resulting merged group, $\mathbf{M}_{\text{ident}}$.

Note that, before the merging of the two most similar groups, subroutine *merging* calls *merge_cond.* This last subroutine verifies whether the group which would result from the merging of k and l satisfies some additional condition that we wish to impose. For example, in galaxy clusters, we could impose that the collapse time of their substructures must be shorter than the age of the universe. If such a condition is satisfied (*m_cond = .true.*), the subroutine *merging* proceeds to the merging of k and l. Otherwise, it will try to find other better candidates. The incorporation of an additional condition is not always necessary in a hierarchical clustering method. In that case, we can ignore it by assigning the value *m_cond = .true.* at every stage of the merging sequence.

IV. DRAWING A HIERARCHICAL TREE

An h-tree gives an effective visual representation of the clustering results. It permits a rapid understanding of the hierarchical relationships and membership of each cluster at any level of aggregation. Because of this, much effort has been devoted to constructing efficient algorithms to draw such trees [8–10]. In this section, we present a new algorithm for h-tree construction.

The merge list contains all the information needed to draw a hierarchical tree and to identify the sample objects associated with each one of its branches. The first of these tasks consists of determining the set of $(N-1)$ horizontal segments and $2(N-1)+1$ vertical segments needed to draw the complete hierarchical tree. Such segments are specified by the coordinates of their extremes (x_1, y_1), (x_2, y_2), stored in the $[3(N-1)+1] \times 4$ array

$$\mathbf{P}^{i,j}_{\text{line}} \equiv (x^i_1, x^i_2, y^i_1, y^i_2).$$

The second task consists of finding the array $\text{ident}(j) \equiv \mathbf{I}^j$ ($j = 1, \ldots, N$), where the identification numbers of the objects appear ordered in the same way as that in the hierarchical tree. For instance, in the example shown in Fig. 1, the array *ident* is (3, 4, 2, 11, 12, 7, 14, 9, 6, 1, 10, 5, 13, 15, 8).

A. *Calculation of ident and of the Horizontal Segments in the h-Tree*

We recall that the merge list is constituted by $(N-1)$ rows and 4 columns. The last row of $\mathbf{M}_{\text{list}}$ corresponds to the last merger in the hierarchical sequence, that is, that sequence giving a system which contains all the objects in the sample. Our algorithm starts by considering this last merging, that is, the row $k = N-1$ of $\mathbf{M}_{\text{list}}$.

We call $j1$ and $j2$ those elements of *ident* so that

$$\mathbf{I}^{j1} = \mathbf{M}^{k,1}_{\text{list}}$$
$$\mathbf{I}^{j2} = \mathbf{M}^{k,2}_{\text{list}}.$$

Obviously, in the merging $k = n-1$, the largest of the merging groups coincides with the first element of *ident,* that is, $j1 = 1$ and $\mathbf{I}^{j1} = \mathbf{M}^{n-1,1}_{\text{list}}$.

Since the number of objects belonging to the group $G_{\text{ident}(j1)}$ is given by $\mathbf{M}^{k,3}_{\text{list}}$, the values of $j2$ and $\mathbf{I}^{j2}$ can be easily calculated from:

$$j2 = j1 + \mathbf{M}^{k,3}_{\text{list}}$$
$$\mathbf{I}^{j2} = \mathbf{M}^{k,2}_{\text{list}}.$$

We have thus determined the interval of elements in *ident* which corresponds to the objects of the group $G_{\text{ident}(j1)}$, as well as that corresponding to the objects of $G_{\text{ident}(j2)}$. The horizontal segment in the h-tree, which represents the merging of these two groups, begins in the middle of the interval $G_{\text{ident}(j1)}$ in *ident* and ends in the middle of the interval of *ident* corresponding to the elements of $G_{\text{ident}(j2)}$. In other words, the coordinates of the extremes of such a segment are:

$$x_1 = j1 + (\mathbf{M}^{k,3}_{\text{list}} - 1)/2$$
$$y_1 = \mathbf{s}^k_{\text{val}}$$
$$x_2 = j2 + (\mathbf{M}^{k,4}_{\text{list}} - 1)/2$$
$$y_2 = \mathbf{s}^k_{\text{val}},$$

```
c=====================================================
      subroutine merging(body,x0,v0,M_list,M_ident)
c=====================================================
      ......
      Declaration of variables and commons
      Initialization of ngroups(i,j)= i (for j=1)
                                      0 (for j>1)
                        M_list(i,j) = 0 (for any i,j)
      ......
c---------------------------------------
c loop over levels (i1=last merged group)
c---------------------------------------

      n1=n
      i1=0

c Calculate or update the S_g for current groups
c-----------------------------------------------

98    call linkage(body,x0,v0,ngroups,S_g,i1)

c Search the two next merging groups
c-----------------------------------

      bmin=1e30
      mergdo=0
33    do i=1,n
       if (ngroups(i,1).ne.0) then
        do j=1,i-1
         bminp=min(S_g(i,j),S_g(j,i))
         if (bminp.lt.bmin.and.ngroups(j,1).ne.0)then
          m_cond = .true.
          if (mergdo.eq.0) then
           call merge_cond(body,x0,v0,i,j,m_cond)
          end if

c update ij as the new best candidate

          if (m_cond.or.n1.le.2) then
           bmin=bminp
           imin=i
           jmin=j
          end if
         end if
        end do
       end if
      end do

c if a good merged candidate has not been obtained,
c switch-off the merge condition and repeat the loop

      if (bmin.eq.1e30) then
       mergdo=1
       goto 33
      end if
108   i1=min(imin,jmin)
      i2=max(imin,jmin)

c construct the merging list
c---------------------------

      il=n+1-n1
      M_list(il,1)=i1
      M_list(il,2)=i2
      M_list(il,3)=ngroups(i1,n+1)
      M_list(il,4)=ngroups(i2,n+1)

c the greatest group defines the merged group

      if (M_list(il,4).gt.M_list(il,3)) then
       i1=M_list(il,2)
       i2=M_list(il,1)
       M_list(il,2)=i2
       M_list(il,1)=i1
       naux=M_list(il,4)
       M_list(il,4)=M_list(il,3)
       M_list(il,3)=naux
      end if

c merge the most bound groups

      nb=0
      s_val(il) = bmin
      do k=M_list(il,3)+1,n
       nb=nb+1
       ngroups(i1,k)=ngroups(i2,nb)
       ngroups(i2,nb)=0
      end do
      ngroups(i1,n+1)=M_list(il,3)+M_list(il,4)
      ngroups(i2,n+1)=0

c save the merged group into M_ident-il

      do k=1,n
       M_ident(il,k)=ngroups(i1,k)
      end do
      write(*,*) 'found level =',n1

c go to the following level

      n1=n1-1
      if (n1.gt.1) then
       goto 98
      else if (n1.eq.1) then
       do i=1,n
        if (ngroups(i,1).ne.0) then
         imin=i
         jmin=j
         goto 108
        end if
       end do
      else
      end if
      return
      end
```

FIG. 4. Listing of subroutine *merging*.

```
c=======================================================
      subroutine linkage(body,x0,v0,ngroups,S_g,i1)
c=======================================================
      .......
      Declaration of variables and commons
      .......

c set bindings to be calculated

      if (i1.eq.0) then
        n1=1
        n2=n
      else
        n1=i1
        n2=i1
      end if

c Calculate binding energies in the first call
c--------------------------------------------

      if (i1.eq.0) then
       do i = 1,n
        do j = 1,i-1
         if (j.lt.i.and.body(j).gt.0.) then
          vrel2=0.
          distance = 0.
          do k = 1,ndim
           temp = x0(k,j) - x0(k,i)
           vrel2=vrel2+
     &          (v0(k,j)-v0(k,i))*(v0(k,j)-v0(k,i))
           distance = distance + temp*temp
          end do
          xmu=body(i)*body(j)/(body(i)+body(j))
          ekin=0.5*xmu*vrel2
          potemp = G*body(i)*body(j)/sqrt(distance)
          S_p(i,j)=-potemp+ekin
          S_p(j,i)=S_p(i,j)
         end if
        end do
       end do
      end if

c Compute or update the affinity array S_g(i,j)
c ---------------------------------------------

c loop over i-groups (if i1>0, just for i=i1)

      do i=n1,n2
       if (ngroups(i,1).ne.0) then
        ni=ngroups(i,n+1)

c loop over j-groups different from the i-group

        do j=1,n
         if (i.eq.j.or.ngroups(j,1).eq.0) goto 21
         if (i1.eq.0.and.j.ge.i) goto 21
         nj=ngroups(j,n+1)

c find the minimum value of S_p(i,j) with i=all the
c group-i elements and j = all the group-j elements

         bmin=1e30
         do ki=1,ni
          ipi=ngroups(i,ki)
          do kj=1,nj
            ipj=ngroups(j,kj)
            if (S_p(ipi,ipj).lt.bmin)
     &         bmin=S_p(ipi,ipj)
          end do
         end do

c the affinity between groups i and j is then bmin

         S_g(i,j)=bmin
         S_g(j,i)=bmin
21      end do
       end if
      end do
      return
      end
```

FIG. 5. Listing of subroutine *linkage.*

where $\mathbf{s}^k_{\text{val}}$ is the value of the association measure between both groups. Our algorithm descends progressively through the *merge list* until it arrives at the first row $k = 1$. At each step k of this algorithm, $j1$ is now obtained by searching out, among the already identified elements of *ident,* that verifying $\mathbf{I}^{j1} = \mathbf{M}^{k,1}_{\text{list}}$. The values of $j2$, $\mathbf{I}^{j2}$, and of the coordinates of the horizontal segment associated with that merging are then calculated as before. When the sequence has finished, all the elements of *ident* have been identified, as well as all the horizontal segments of the h-tree.

B. *Calculation of the Vertical Segments of the h-Tree*

We see in Fig. 1 that each one of the vertical segments of the h-tree has, as its upper extreme, a point (x_1, y_1) which coincides with one of the $2(N - 1)$ extremes of the horizontal segments previously obtained. The lower extreme is instead determined by the point in which the vertical line going through (x_1, y_1) intersects, for the first time, a horizontal segment located in lower y-positions. Our algorithm uses this property to determine each one of the vertical segments. In other words, the coordinates (x_h, y_h) of the extremes of every horizontal segment stored in $\mathbf{P}_{\text{line}}$ define the upper extreme of a vertical segment:

$$(x_1, y_1) = (x_h, y_h) = (\mathbf{P}^{j3,k}_{\text{line}}, \mathbf{P}^{j3,k+1}_{\text{line}}),$$

with

$$j3 = 1, ..., (N - 1) \qquad \text{and} \qquad k = 1 \text{ or } 3.$$

```
c=====================================================
      subroutine drawtree(M_list,s_val)
c=====================================================
      ........
      Declaration of variables and commons
      ........
c
c Set the range of rho values
1     rhomin=1e30
      rhomax=-1e30
      do i=1,n
        if (s_val(i).lt.rhomin) rhomin=s_val(i)
        if (s_val(i).gt.rhomax) rhomax=s_val(i)
      end do
      rhorange=rhomax-rhomin
      rhomin=rhomin-rhorange/float(n-1)
c
c Determine horizontal lines of the tree
c
      do i=1,n-1
        il=n-i
        if (i.eq.1) then
          jbeg1=1
          ident(1)=M_list(il,1)
        else
          jbeg1=0
          do j=1,n
            if (ident(j).eq.M_list(il,1)) jbeg1=j
          end do
        end if
        jbeg2=jbeg1+M_list(il,3)
        ident(jbeg2)=M_list(il,2)
        pline(i,1)=jbeg1+float(M_list(il,3)-1)/2.
        pline(i,2)=jbeg2+float(M_list(il,4)-1)/2.
        pline(i,3)=s_val(il)
        pline(i,4)=s_val(il)
      end do
c
c Determine vertical lines of the tree
c-------------------------------------
      il=n-1
      do i=1,n-1
        y1=s_val(n-i)
        do k=3,4
          y2=rhomin
          do j=i+1,n-1
            if (pline(i,k).ge.pline(j,3).and.
     *        pline(i,k).le.pline(j,4)) then
              y2=s_val(n-j)
              goto 10
            end if
          end do
10        il=il+1
          if (y2.gt.y1) then
            s_val(n-j)=y1-rhorange/100.
            goto 1
          end if
          pline(il,1)=pline(i,k)
          pline(il,2)=pline(i,k)
          pline(il,3)=y1
          pline(il,4)=y2
        end do
      end do
      il=il+1
      pline(il,1)=(pline(1,1)+pline(1,2))/2.
      pline(il,2)=pline(il,1)
      pline(il,3)=pline(1,3)+
     &            (pline(1,3)-pline(2,3))/10.
      pline(il,4)=pline(1,3)
c
c Draw the tree
c
      .........
      Draw segments defined by pline by using a
      graphic library (for example, pgplot)
      .........

      return
      end
```

FIG. 6. Listing of subroutine *drawtree.*

We look then for the row $j4$ of $\mathbf{P}_{\text{line}}$ so that

$$\mathbf{P}_{\text{line}}^{j4,1} \leq x_h \leq \mathbf{P}_{\text{line}}^{j4,3}.$$

The coordinates of the searched lower extreme are thus:

$$(x_2, y_2) = (x_1, \mathbf{P}_{\text{line}}^{j4,2}).$$

If $j3 = N - 1$ (terminal branches), the value assigned for y_2 must be some fixed value smaller than $\min_j(\mathbf{s}_{\text{val}}^j)$. By repeating this procedure, we thus find the $2(N-1)$ vertical segments. To them, we can add another, of arbitrary extension, above the middle of the highest horizontal segment. This last segment just represents the overall set of data. Once this is performed, any graphic library (i.e., *pgplot*) can be used to draw the $3(N-1)+1$ segments defined by the points $(\mathbf{P}_{\text{line}}^{j,1}, \mathbf{P}_{\text{line}}^{j,2})$ and $(\mathbf{P}_{\text{line}}^{j,3}, \mathbf{P}_{\text{line}}^{j,4})$. We give in Fig. 6 the listing of the subroutine *drawtree,* where this algorithm is performed.

V. ANALYZING A HIERARCHICAL TREE

Unlike some other topics in statistics, the methods employed in cluster analysis have a great degree of flexibility and subjectivity. Strategies of cluster analysis are at a rather embryonic stage of development and refinement.

A collection of $2(N-1)$ different "subgroups" appears to be involved in the sequence of $(N-1)$ hierarchical mergers stored in the *merge list.* Obviously, most of these "subgroups" just represent intermediate stages of the hierarchical method and they cannot be considered as substructures with a real physical meaning. In the analysis of the *h*-tree, we must decide whether some of such subgroups

can be considered as a true structure. If so, we must also calculate the most interesting physical features of such substructures.

The criterion to decide the significance of a group, and the physical properties that our analysis subroutine must calculate, depend both on the nature of the data sample and on our physical motivations. We will just provide here some basic ideas that we believe especially well suited to analyze galaxy clusters and large-scale galaxy catalogues.

A. *Extraction of Subgroups*

In the example shown in Fig. 1, it seems natural to consider {1, 10, 5, 13, 15, 8} and {3, 4, 2, 11, 12, 7, 14, 9, 6}, the last with the small-scale groups {3, 4, 2} and {11, 12, 7}, as the unique subgroups with a possible real dynamical meaning. We see in Table I that, except for $k = 14$ and 9, the smallest of the two groups involved in each merging is always constituted by only one object. These two stages ($k = 14$ and 9) correspond to the merging of the two above-quoted pairs of groups. Thus, the fact that such groups seem more natural than the other ones can be stated by the following condition: they are groups of at least $n_{\min}$ (>1) objects which aggregate to groups of at least $n_{\min}$ objects.

In our code, we have used that condition as the criterion to extract the possible real subgroups in a sample. That is, we look in the merge list for those mergers so that the number of objects, $\mathbf{M}_{\text{list}}^{k,4}$, in the smallest of the two merging groups is larger than $n_{\min}$. The membership of each one of these groups is then extracted from $\mathbf{M}_{\text{ident}}$ and stored for subsequent analyses (see Fig. 7).

Note that the most appropriate value of $n_{\min}$ could sometimes be larger than 2, for example, if we do not wish to identify simple galaxy pairs with possible substructures.

```
c=================================================
      subroutine extract(M_list,M_ident)
c=================================================
      .......
      Declaration of variables and commons
      including ngroups(n,n) and nglist(n,3)
      .......
c
c Extract and give name to the possible groups
c --------------------------------------------
      ng=0
      do i1=0,n-1
        i=n-i1
c enhanced groups have mergers with M_list(i,4)>=nmin
        if (M_list(i,4).lt.nmin) goto 100
        do k=3,4
          ng=ng+1
          nglist(ng,1)=M_list(i,k-2)
          nok=0
          do ii=1,ng-1
            ii1=ng-ii
            if (k.eq.3.or.ii.gt.1) then
              if (nglist(ii1,1).eq.M_list(i,1)) then
                nok=nglist(ii1,2)
                goto 10
              end if
            end if
          end do
10        nglist(ng,2) = nok*10+k-2
          nok=0
          if (k.eq.4) nok=M_list(i,3)
          do j=1,M_list(i,k)
            nstruct(ng,j)=M_ident(i,j+nok)
          end do
          do j=M_list(i,k)+1,n
            nstruct(ng,j)=0
          end do
          if (i.eq.n) then
            nglist(ng,3)=n
          else
            nglist(ng,3)=M_list(i,k)
          end if
        end do
100   end do
      return
      end
```

FIG. 7. Listing of subroutine *extract.*

B. *Analysis of the Extracted Substructures*

Once the possible substructures of a sample have been extracted, we must analyze their significance as well as their most interesting physical features. In spite of the fact that hierarchical clustering methods are not conceived from a statistical point of view, an estimate of the significance of the extracted subgroups can be useful for the interpretation of results. However, we must not forget that such estimates just constitute a helping tool for the scientist. The final decision on the real significance of a substructure must be made by also considering other information, like its physical features.

Materne [11] estimated the significance of a structure by assuming that the observed coordinates of the cluster galaxies satisfy a distribution which consists of the overlapping of the Gaussian distributions of each one of its structures, as well as that corresponding to the overall cluster.

According to this approximation, the most meaningful structures will have a most important contribution to the fitting of the total distribution of observed positions and velocities in the cluster. A measure of the statistical significance of the group k can be then calculated by approximating the distribution of

$$-2\log(L^k/L) \tag{7}$$

to the function χ^2 with a degree of freedom [11].

In Eq. (7), L is the likelihood of the cluster partition

in N_S groups (including the background cluster) obtained through the h-analysis. L^k is instead the likelihood under the hypothesis that the group k does not exist,

$$L = \prod_{i=1}^{N} \left[\sum_{j=1}^{N_S} \phi(i,j) \right]; \qquad L^k = \prod_{i=1}^{N} \left[\sum_{\substack{j=1 \\ j \neq k}}^{N_S} \phi(i,j) \right], \qquad (8)$$

where $\phi(i, j)$ is the probability that the galaxy i belongs to the group j, assumed to be Gaussian,

$$\phi(i,j) = \frac{N_j}{(2\pi)^{3/2}(\sigma^j_{\alpha\delta})^2 \sigma^j_v} \times \exp\left[-\frac{(\alpha_i - \alpha_j)^2}{(\sigma^j_{\alpha\delta})^2} - \frac{(\delta_i - \delta_j)^2}{(\sigma^j_{\alpha\delta})^2} - \frac{(v_i - v_j)^2}{(\sigma^j_v)^2} \right],$$

N_k being the number of objects in the group j; α_j, δ_j, and v_j being the mean group coordinates; and $\sigma^j_{\alpha\delta}$ and σ^j_v being their standard deviations. The probabilities $\phi(i, j)$ must be normalized so that $\sum_{j=1}^{N_S} \phi(i, j) = 1$.

Note that, unlike Materne [11], we do not search for the partition giving the maximum likelihood. Since the stability of our method, as well as the dynamical meaning of the obtained groups, has been tested from N-body simulations, we assume that the results of the h-analysis are directly those which give the maximum likelihood.

Obviously, the substructures of a cluster do not always satisfy a Gaussian distribution. Consequently, Eqs. (7), (8) should be considered just as a first estimate of the significance of the extracted substructures. A more rigorous calculation would perhaps require Monte-Carlo simulations analyzing the modifications introduced by uncertainties in the data. In the case of a galaxy cluster, such uncertainties mainly come from the error bars in the sample data and, especially, from the transformation from observed coordinates (α, δ, apparent magnitude) to intrinsic coordinates (**r**, **v**, mass). These Monte-Carlo simulations have however the disadvantage that, in samples containing a relatively high number of objects, the computing time increases enormously. Because of this, the current version of our code does not use this last approach.

In any case, a reliable interpretation of results also requires one to consider the different physical features of each substructure such as, for instance, the virial mass, the mean harmonic radius, the spherical collapse time, the binding energy, the mean redshift as well as its standard deviation, variance, skewness, and kurtosis. The set of all these data, for each substructure obtained from the h-analysis, provides extremely valuable information for describing the internal physics of a galaxy cluster.

VI. FINAL COMMENTS

We have described in detail the fundamental strategy for implementing the hierarchical clustering analysis. In our code, special attention has been paid to identifying and separating all those points which depend on the type of data in the sample as well as on the physical nature of the searched structures. Although our code has been applied here to the case of substructures in galaxy clusters, its adaptation to any other kind of system should not be difficult.

The physical magnitude which determines the nature of the extracted structures is the affinity parameter, s_{ij}, and the type of linkage (Eqs. (3)–(5)). We have considered that s_{ij} is the relative binding energy between the objects of the sample (Eq. (1)), which is especially appropriate for identifying groups of galaxies. Although other kind of systems would require another definition for s_{ij} (and, in some cases, another type of linkage), but the general structure of the code, described in Section 3, as well as the algorithm to draw the h-tree (see Section 4), would be essentially identical. The subsequent analysis of the results would have instead to be readapted for that kind of problem except, perhaps, for what concerns the algorithm of extraction of possible real substructures (Section 5).

It must be noted finally that our clustering method has been constructed using the six (**r**, **v**) coordinates of each object. However, as mentioned in Section III A, astronomical observations only provide the projected positions and the line-of-sight velocities of galaxies. In paper I, we showed through N-body simulations that results are remarkably stable against projections, that is, that the ''observed'' (3D) structures are reasonably similar to the ''intrinsic'' (6D) ones. Consequently, a hierarchical method based on relative binding energies is very well suited to being applied in the analysis of galaxy clusters. The main difficulty of this method is, however, that the computing time strongly increases with the number of objects. Consequently, its application is limited to systems with, at most, some few thousands of objects. The analysis by this procedure of cosmological N-body simulations, which usually consider huge numbers of particles, is then prohibitively expensive in computing time.

Note. The package containing the full Fortran code described in this paper, as well as some examples and instructions, can be freely rerceived by e-mail request to serna@gin.obspm.fr.

REFERENCES

1. A. Serna, J.-M. Alimi, and H. Scholl, *Astrophys. J.* **427,** 574 (1994); and references therein.
2. A. Serna and D. Gerbal, *Astron. & Astrophys.,* in press (1995).
3. J. Materne, *Astron. & Astrophys.* **63,** 401 (1978).

4. R. B. Tully, *Astrophys J.* **237,** 390 (1980).
5. R. B. Tully, *Astrophys. J.* **321,** 280 (1987).
6. E. Gourgoulhon, P. Chamaraux, and P. Fouqué, *Astron. & Astrophys.* **255,** 69 (1992).
7. M. R. Anderberg, *Cluster Analysis for Applications* (Academic Press, London, 1973).
8. R. B. McCammon, *Geol. Soc. Amer. Bull.* **79,** 1663 (1967).
9. R. B. McCammon and G. Wenninger, in *Computer Contributions,* Vol. 48 (State Geol. Survey, Univ. of Kansas, Lawrence, 1970).
10. J. M. Parks, in *Computes Contributions,* Vol. 48 (State Geol. Survey, Univ. of Kansas, Lawrence, 1970).
11. J. Materne, *Astron. & Astrophys.* **74,** 235 (1979).