Dr. PAUL CONSTANTINESCU

On the Clustering of a Set of Elements with respect to a Set of Criteria

The author gives a definition to the cluster (class) and examines possible solutions of the clustering problem. He further introduces certain algorithms for solving it and gives a full description of an algorithm in ALGOL.

Although a number of sciences and other fields of activity have for a long time taken into consideration the problems of clustering a set of elements with respect to different criteria — that is the problems of taxonomy — these have become of late, as a result of computer development, quite complex problems. Their solving requires the use of modern computers to an ever greater extent.

An increasing amount of information involved in various problems of engineering, economics, science, arts, etc., has brought about the necessity of using, as far as possible, a series of schemes for clustering and systematizing the necessary elements and the data in order to solve these problems. The process of clustering large sets of elements which is used nowadays for solving most varied problems is, generally speaking, similar to that, already classic in mathematics, of dividing into classes the different sets provided with algebrical or topological structures. Taxonomic problems, however, are usually considered in sets lacking such properties; the classification is obtained in respect with certain criteria given by a function applying the couples of elements in the set $I = (1, 2,..., n)$. For a large class of problems the scope of this function is the set of all the couples formed by distinct elements (owing to their particular nature). In this case, the mathematical definition of the *class of elements* in a fairly general clustering scheme (allowing to tackle problems of different fields) is worth taking into account. At least two ideas seem to deserve being considered with respect to this definition:

a) Suppose the numbers associated to the couples of elements belong to the set $I = (1, 2,..., n)$ of natural numbers. If we represent the half circles

C_1, C_2,...C_n in the half plane $x > 0$ of radius, respectively, 1, 2, ..., n, (Fig. 1) we shall consider that the subsets $E_1, ..., E_q$, of the set of elements $\in$ is in C_k if and only if *all the* $\frac{q(q-1)}{2}$ *couples we can form with these q elements* are associated to numbers inferior or equal to k, (k being associated at least to one couple).

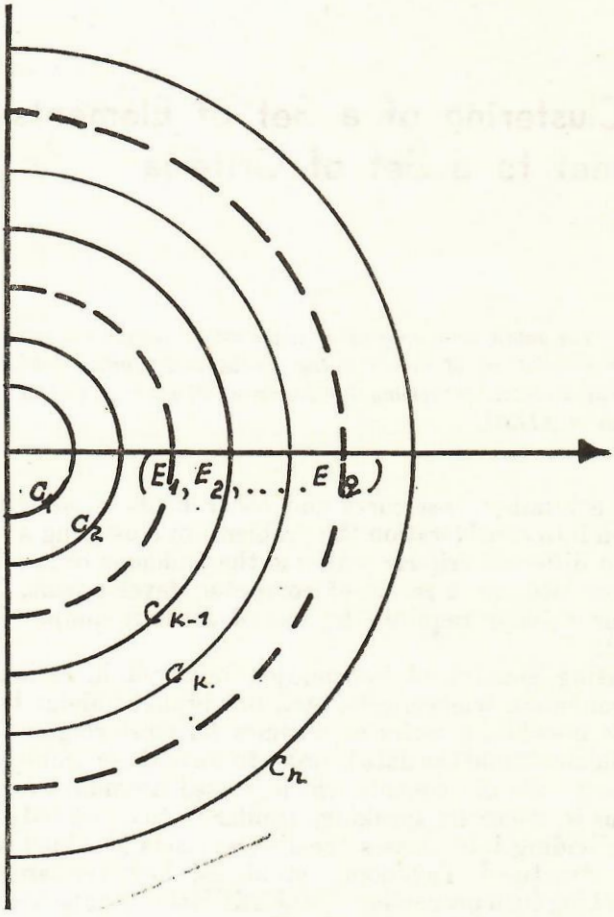

Fig. 1.

If two subsets which are in C_k (but not in C_{k-1}) are disjoint, they can be considered *classes (clusters)* in our classification scheme. Out of the domains bounded by C_1, C_1 and C_2,..., C_{n-1} and C_n, we can select a set of clusters whose reunion is the set E constituting a classification of this set.

We should point out that in this model (widely applied in technics, biology and psychology) a cluster in the domain bounded by C_{k-1} and C_k is obtained from elements whose possible couples are associated to numbers *at most equal to k.*

b) As regards other problems (such as, for instance, clustering the vertices of a graph with a view to optimize designing and programming) a cluster in the domain bounded by C_{k-1} and C_k, which were produced only by elements whose possible couples are associated numbers *exactly equal to* k, seems to be a useful technique.

A particular case of this second model is studied in [1].

We shall further give a definition of the cluster in case a) and examine the possible solutions of the clustering problem as described by the author in [2]. In order to solve these clustering problems with applications in psychology, biology and optimization we shall also introduce certain algorithms used by the author, [3], [4]. A full description of an algorithm will be formulated in ALGOL, namely the ALGOL restriction used in the Atlas computer (S.R.C. Atlas Computer Laboratory, Chilton, Didcot, Berkshire, England). This algorithm helped to solve the problems referred to in [3], [4], [5].

*

Let $\mathfrak{C} = (E_1, E_2, \ldots, E_n)$ be a set of elements and $\pi = (P_1, P_2, \ldots, P_n)$ a set of properties.

An element $E_1 \in \mathfrak{C}$ may or may not have a property $P_k \in \pi$. We can associate to the element E_1 and the property P_k a variable e_{ik} (0.1), i.e.:

$$e_{ik} = \begin{cases} 0 \text{ if } E_i \text{ has not the property } P_k, \\ 1 \text{ if } E_i \text{ has the property } P_k. \end{cases}$$

We can associate to the sets $\mathfrak{C}$ and π the matrix E with m rows and n columns:

$$E = \begin{pmatrix} e_{11} & e_{12} & \ldots & e_{1k} & \ldots & e_{1n} \\ e_{21} & e_{22} & \ldots & e_{2k} & \ldots & e_{2n} \\ \vdots & & & & & \\ e_{m1} & e_{m2} & \ldots & e_{mk} & \ldots & e_{mn} \end{pmatrix}$$

Considering the metric space H^n of the vectors

$$e_1 = (e_{i1}, e_{i2}, \ldots, e_{in})$$

for which are determined

$$e_{1i} + e_j = (e_{i1} + e_{j1}, \ldots, e_{in} + e_{jn})$$

$$\lambda e_i = (\lambda e_{i1}, \ldots, \lambda e_{in}), \ \lambda \in (0,1)$$

where $+$ is the sum of modulo 2:

$+$	0	1
0	0	1
1	1	0

In this space we consider the distance

$$d(e_i, e_j) = p_{ij}$$

where p_{ij} represents the number of components 1 of the vector $e_i + e_j$ (the

weight of this vector). Between the vectors e and the elements E_i we have an one to one correspondence $e_{1i} \leftrightarrow E_{1i}$.

Other definitions of the distance may be considered according to each separate case. But this is not material to what follows.

Likewise, the fact that we considered the space H^n over the field (0, 1) is not a restriction. The vectors can have components in other fields (rational or real). The theorem of existence and the algorithms we present do not depend on the field over which space H^n is considered. However, this particular case of field (0, 1), allows a comparatively easy solving of the following sort of problems:

Suppose all the components of the vectors e_1, e_2, ..., e_k are given, but some components of the vectors $e_{k+1}, e_{k+2}, ..., e_m$ are unknown, i.e. the relations between certain elements E_{k+1}, ..., E_m and certain properties P_1, ..., P_n are unknown. In order to cluster the set $e_1, ..., e_m$ we can complete these components by solving a problem of optimum, for instance, selecting from the hyperfaces (on the hypercube) associated to the elements $E_{k+1}, ..., E_m$ the vectors e_{k+1}, ..., e_m, so that this set of vectors should be at maximum (minimum) length from the set $e_1, ..., e_m$. Such problems are easily solved on the hypercube [3].

We can describe the distances for all the couples of vectors by means of the matrix D of the order m:

$$D = \begin{pmatrix} d_{11} & d_{12} \dots d_{1m} \\ d_{21} & d_{22} \dots d_{2m} \\ \vdots & \\ d_{m1} & d_{m2} \dots d_{mm} \end{pmatrix}$$

This matrix is symmetrical, $d_{ii} = 0$ $(i = 1, ..., m)$ and $d_{ij} \in I = \{1, 2, ..., n\}$ for $i \neq j$.

We can associate to the matrix D a graph G obtained with the aid of $n + 1$ levels (level 0, level 1,..., level n); for each couple of elements (E_i, E_j) we draw the branches starting from E_i and E_j and meeting on level $d_{ij} = d(e_i, e_j)$ thus determining a vertex (E_i, E_j) on that level (Fig. 2):

By considering for each couple (E_i, E_j) the corresponding vertex, we obtain a graph G associated to matrix D. We shall call initial vertices of the graph G the elements from $\in$. In this graph we can see on each level the couples of elements having as a distance the number associated to the respective level. The matrix D or the graph G may be taken as a starting point for a large range of clustering problems.

In connection with matrix D and graph G we shall note that:

Remark 1

All the algorithms we shall further present are based on the supposition that the levels are associated to non negative integers. We make this remark on the strenght that, the elements of the matrix E being real numbers approximately represented, if we multiply them by certain constants (for instance, the different weights associated to the columns corresponding to the properties P_i) we can turn the numbers d_{ij} into integers. Moreover, in case the nature of the problem does not allow the association of weights, we shall consider the weights which bring all the numbers p_{ij} in a smaller interval $I = (0, ..., n)$.

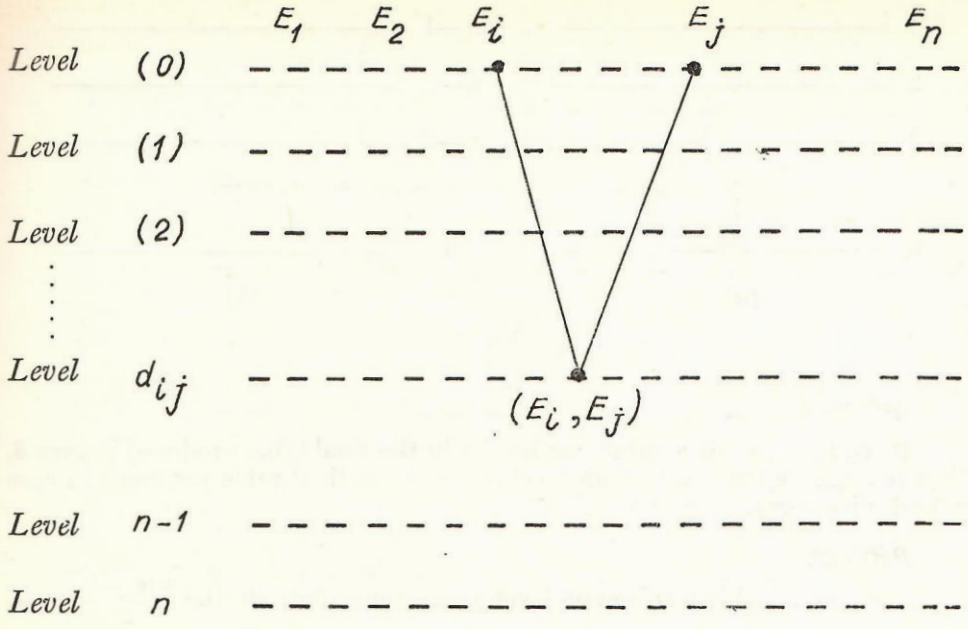

Fig. 2

Remark 2

Whether the matrix D represents experimental data [3], [4], or is calculated out of the matrix E [2], [5], by taking into account the distances between athe elements or the similarities (as we shall see further), the symmetry of this matrix may be looked upon as a natural hypothesis. Even in such problems where the elements of the matrix D are not symmetrical (for instance, when clustering the programs according to the number of transfers operated among them), we can consider the symmetrical matrix D^+ in which we have, for instance, the elements $d^+_{ij} = d_{ij} + d_{ij} = d^+_{ji}$ as a starting point for the clustering problem.

As a consequence of *Remark 1*, we shall demonstrate a lemma of duality in connection with the metric — interpreted as distance or as similarity, which can be considered in the vector space associated to the elements.

If the similarity between e_j and e_k is $s(e_j, e_k) = \sigma$ (for different definitions see [1], then we shall define the distance $d(e_j, e_k) = \delta = n - \sigma$, where n is the highest natural number of the interval I containing the numbers p_{ij}; reciprocally, if according to a certain definition the distance between e_j and e_k is $d(e_j, e_k) = \delta$, then we can define the similarity $s(e_j, e_k) = \sigma = n - \delta$.

The final table (Fig. 3, a) showing the subsets of elements out of which the clusters are selected, the table being drawn up on the same levels as the graph G produced in case σ (when similarities are taken into account), is dual to the final table (Fig. 3, b) produced in case σ (when distances are considered), in the sense of *Lemma 1:*

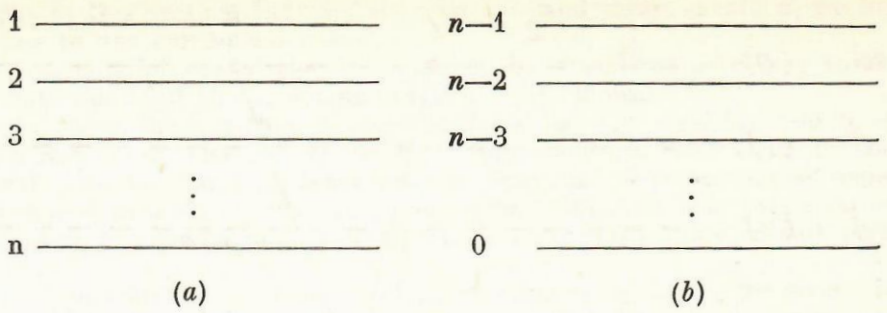

Fig. 3

Lemma 1.

If $(e_1, e_2, \ldots, e_g)$ is a subset on level q in the final table produced in case δ, then $(e_1, e_2, \ldots, e_g)$ is a subset on level $n-q$ in the final table produced in case σ and vice-versa.

PROOF.

If $(e_1, e_2, \ldots, e_g)$ is a subset on level q, it means that all the $\frac{g(g-1)}{2}$ couples of vectors $(e_1, e_2), (e_1, e_3), \ldots, (e_1, e_g), \ldots, (e_{g-1}, e_g)$ are on the first q levels in graph G. Hence, $\delta = d(e_i, e_2) \leqslant q$ for all the $\frac{g(g-1)}{2}$ couples. Consequently

$$q \leqslant s(e_1, e_j) = \sigma < n$$

where $\sigma = q$ if and only if $d - q = q$ and $\delta = q$.

If in the final table produced in case σ the $\frac{g(g-1)}{2}$ couples are on levels $n-1, n-2, \ldots, n-q$ ($\sigma = n - \delta$), then the subset $(e_1, e_2, \ldots, e_g)$ is in this table on level $n-q$. We proceed analogically in order to prove the reciprocal case.

Lemma 1 allows us to work out the clustering algorithms for one of the two cases, and transfer by duality the results obtained to the other case. We shall now consider the case δ in which the distances between the vectors are considered according to a suitable definition of the distance with reference to the respective problem.

*

We notice that, given the matrix D, the graph G results univocal and reciprocal. The same amount of information is contained in both ways of representing the data of the clustering problem.

If in the graph G all the couples formed by $E_{i_1}, \ldots E_{i_h}$ elements are bound by branches on the same level r, we shall consider in the graph G only the vertex $E^r_{i_1 i_2 \ldots i_h}$ which is connected with the initial vertices $E_{i_1}, \ldots, E_{i_h}$ as shown in Fig. 4.

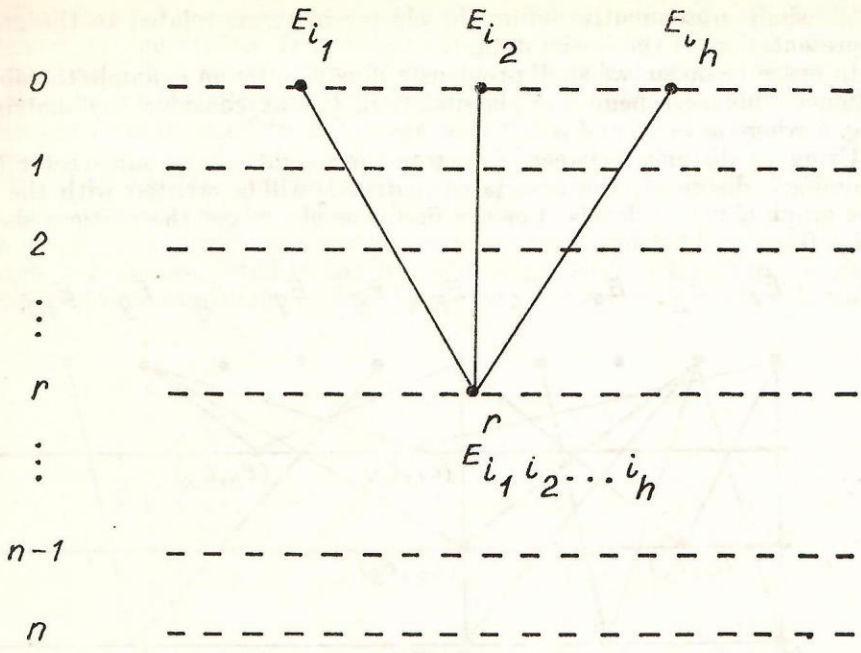

Fig. 4

1	0	0	0	1	1	0	0	0	0	0	0	1
1	0	0	0	1	1	1	0	0	1	1	0	1
1	0	0	0	0	1	1	0	0	0	0	1	0
1	1	0	0	1	1	1	0	0	1	0	0	0
1	0	0	0	1	0	0	0	0	0	1	0	1
0	1	1	1	1	0	0	1	1	0	0	0	1
0	0	1	1	0	0	0	0	1	0	0	0	0
1	0	1	1	0	1	1	0	1	1	1	1	0
1	0	0	0	1	1	0	0	0	0	1	0	1
0	1	1	0	1	0	0	1	0	0	0	0	0

Fig. 5

We shall subsequently define the cluster in terms related to the graph G representation of the initial data.

In order to do so, we shall previously illustrate by an example the above mentioned "binary scheme" of classification. Let us consider the matrix E of Fig. 5 where $m = 10$ and $n = 13$.

Using as distance between 2 vectors the weight of the sum-vector (the "Hamming" distance), the associated matrix D will be written with the aid of the graph G with n levels. For the first 3 levels we get the vertices shown in Fig. 6.

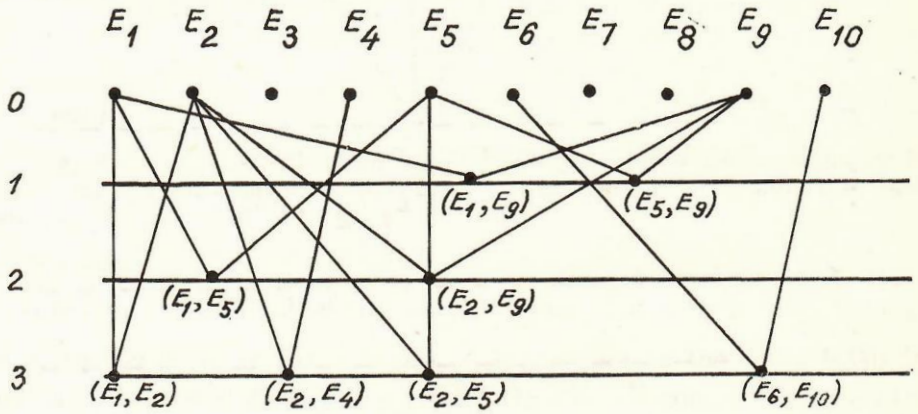

Fig. 6

In this case, in order to draw all the vertices associated to the $\frac{9.10}{2}$ couples of elements, we shall consider 11 levels (on two evels out of 13, there are no vertices). But this is rather a slow way of representing initial data. We notice that the presence of branches in the graph G is immaterial; the graph G — as we shall see further — only suggests the next step in the algorithm. A *representation* G' associated to the graph G, which does not take into consideration the branches but only the vertices marked by the couple of elements they result from, contains the same amount of information as the graph G' or the matrix D. *Representation* G' of the first three levels, in the case of the example considered, is given in Fig. 7.

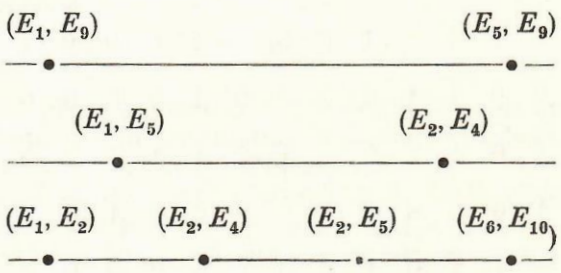

Fig. 7

This form of representation is the closest to introducing the information contained in the matrix D into the computer memory. We shall also point out that instead of the elements E_i we can only represent the indices i, and separate the set of indices in first position in the couples marking the points of the representation G' from the set of indices in second position in the couples.

We shall consider two tables, G_1 and G_2 ,where these sets of indices will be respectively represented, by establishing an one-to-one correspondence between the rows of these tables and the levels of the graph G: the order number of each row is associated to the corrresponding level in the graph G. Figure 8 gives a G' representation of all 11 levels with the aid of the tables G_1 and G_2.

G_1:

1	5						
1	2						
1	2	2	6				
1	1	3	6				
2	3	3	4	7			
1	2	3	4	4	5	7	
1	1	3	4	5	5	9	
3	6	7	8				
1	2	4	4	5			
2	2	6					
3	8						

G :

9	9						
5	5						
2	4	5	10				
3	4	4	7				
3	8	6	9	10			
10	8	5	5	10	10	8	
6	7	7	8	6	7	10	
10	9	6	9				
8	10	9	7	8			
6	7	8					
6	10						

Fig. 8

Actually, the tables G_1 and G_2 constitute a starting point for the basic algorithm we are going to describe.

In order to achieve the next step in the algorithm and give the definition of the cluster, a few preliminary definitions are required.

Definition 1

We shall say that the level p is *superior* to the level r if $p \leqslant r$ ($p, r \in I = (0, 1, \ldots, n)$) (or that r is *inferior* to p). According to this definition we shall call the vertices on the last level, final vertices — while those on the first level (level 0) will be called initial vertices.

Definition 2

We shall call a *tree* a graph G associated to a matrix D if it has only one final vertex (on the level n or on a superior one) and if from each vertex (including the initial ones) only one branch goes out to the inferior levels.

Definition 3

We call a *representation* G' of the graph G the representation of the vertices marked by couples or rows of elements from graph G on the respective levels, without the corresponding branches.

Definition 4

A G' representation of the graph G is a *tree-representation* if it has only one final vertex and if on each level it has only $E^r_{i_1} \dots i_h$ vertices for which the sets of indices $J_i = (i_1, \dots, I_h)$ are disjoint, and if each set on a level other than the level 0, when it contains at least one element of a set placed on a higher level, contains all the elements of this set.

Definition 5

We shall call *absorption* of certain vertices of the set of vertices $E^{q_j}_{i_{j_1}} \dots i_{j_k}$, $(i_{j_1}, \dots, i_{j_k}) = J_j$, *by the vertex* $E^q_{i_1} \dots, i_l$ where $q = \max(q_j)$ and $J = (i_1, \dots, i_l) = \bigcup_J J_j$ if the representation G' is substituted by a representation G'', where the respective vertices of the set $E^{b_j}_{i_{j_1}} \dots i_{j_k}$ are not represented and where the vertex $E^q_{i_1} \dots i_l$ is represented.

The conversion of the representations G' into the representations G'' underlies the classification algorithm. By operating such transformations we obtain different trees or tree-representations which, as we shall see, ofer solutions to the clustering problem.

According to **definition 5** we shall state that some of the elements E_{i_1}, $E_{i_2}, \dots E_{i_l}$ have been absorbed by the set $(E_{i_1}, E_{i_2}, \dots, E_{i_l})$ (represented by a vertex in G or in G').

Remark 1

A part (possibly all) of the vertices $E^{q_j}_{i_{j_1}} \dots i_{j_k}$ can be absorbed by $E^q_{i_1} \dots i_l$ **(Definition 5)** (if and only if all the $E^{q_j}_{i_{j_1}} \dots i_{j_k}$ vertices representing all the possible $\frac{l(l-1)}{2}$ couples of elements $E_{i_1}, \dots, E_{i_l}$ are placed on levels superior to q.

Remark 2

We are particularly able to obtain through absorptions, from the vertices placed on the same level r and representing all the possible couples of elements $E_{i_1}, \dots, E_{i_h}$, a single vertex $E^r_{i_1} \dots i_h$ (see Fig. 4).

Definition 6 (Cluster)

A set of initial elements $E_{i_1}, \dots i_l (< m)$ represented in a tree, or a tree-representation, by the vertex $E^q_{i_1 i_2} \dots i_l$ is called a *cluster* of set $\mathcal{C}$.

We shall prove the following lemma:

Lemma 2

A *tree-representation* G' is a representation of a tree G and reciprocally.

PROOF

We shall prove that if the representation G' is a tree-representation, then the graph G is a tree. Actually, if on each level the sets J_j are disjoint, and if each set, when containing at least one element of a set placed on a superior level contains the whole set as a subset, then from each vertex on the superior levels only one branch goes out to the vertices on the inferior levels. And reciprocally: if the graph G is a tree, then the sets of indices marking the vertices in the representation G' of the graph G are disjoint on each level and the containing of one element from a set of a superior level entails the inclusion of the whole set.

Consequently (according to the cluster definition) the clustering problem we propose to solve may be thus formulated: let us associate with the graph G (the representation G'), which represents the distances between the elements of the set $\mathcal{E}$, at least one tree (one tree-representation). The final table in which we shall represent the associated trees will show on each level different sets of indices among which we shall select the clusters composing a tree (a tree-representation). According to lemma 2 and definition 6, the clusters on each level are disjoint sets of indices.

Before proceeding to the elaboration of the algorithm, we shall first discuss its possible solutions, namely, the possibilities to associate a tree with a graph G.

Definition 7

A treee (tree-representation) which has, beside its initial vertices, one single final vertex marked by a set $\mathcal{E}$ is called a trivial tree (trivial tree-representation). An example of trivial tree-representation is given in Fig. 9, provided that the final vertex be on level n.

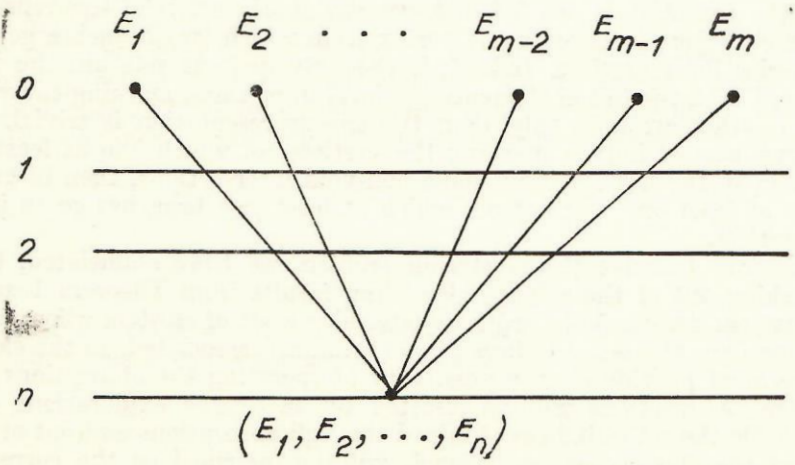

Fig. 9

The theorem of the existence of non-trivial trees will be proved:

Theorem 1

A representation can be transformed in a non-trivial tree-representation if and only if there are at least two vertices for which the sets J and J' are *disjoint*, where $J \cup J' = \{1, 2, \ldots, m\}$, and if at least one vertex is not on the last or on the first level ($1, 2, \ldots, m$ are the indices of the set $\in$ of elements).

PROOF

The condition is sufficient.

Let E_j and $E_{j'}$, be the vertices representing the elements whose indices are in J, respectively in J'. Let E_j and $E_{j'}$ be on level r, respectively r' $(r < r')$.

Let us consider E_j^* and E_j^{**} on level $p < r$ for which $J^* \cap J^{**} = K \in \{1, 2, \ldots, m\}$.

a) If $J^* \cup J^{**} \subset J$ then E_j^* and E_j^{**} can be absorbed by E_j^*.

b) If $J^* \subset J$ and $\{J^{**} - k\} \subset J'$ then E_j and E_j^{**} can be absorbed by vertex $E_{J \cup J'_r}$.

The same reasoning may be analogically used for the other cases. Therefore, if the condition given in the enunciation of Theorem 1 is fulfilled then we are able to obtain through absorption, on each level, only vertices marked by disjoint sets of indices.

We shall also prove that one single final vertex is obtained by means of absorption. Thus, if we have for instance 2 final vertices $E_{\alpha\beta}$ and $E_{\gamma\delta}$ for which $\alpha, \beta \in J$ and $\gamma, \beta \in J'$ then both can be absorbed by $E_{J \cup J'}$. We can reason analogically in other cases where two or more final vertices are concerned.

The condition is necessary

If the condition is not fulfilled, we can obtain a trivial tree-representation, or else there is at least one vertex from which two branches go out to the inferior level vertices. Indeed, if the only disjoint sets are the indices associated with the initial elements (on level 0) or those marking the vertices placed on the last level only, then the tree representation is trivial. If for every two sets of indices marking the vertices (of which one at least is on level 0 or on the last level) we have non-void intersections, then in graph G there is at least one vertex from which at least two branches go to inferior level vertices.

The way to solve the clustering problem we have enunciated, that is the working out of the solving algorithm, results from Theorem 1 and our previous considerations. In order to determine a set of clusters whose reunion should produce the set of indices $I_m = (1, \ldots, m)$ associated to the elements in $\in$, we must provide for the possibility of operating the absorptions in the algorithm. Absorptions will be effected by successive explorations of the levels, while the set of indices obtained through absorptions and out of which we select the clusters on each level, will be inscribed on the corresponding level.

We compute the absorptions by using the TRIANG procedure (see annexed the Clustering Program in ALGOL).

In order to place the set $J_i = (i_1, \ldots i_k)$ on level q it is necessary and sufficient that all the marks representing the $\frac{k(k-1)}{2}$ couples $(i_1, i_2), \ldots, (i_{k-1}, i_k)$ should be present on the first levels q. In order to find all the sets of indices meeting this condition *(Remark 1)* we shall consider the variable b which describes the rows in G_1 and G_2 (the levels in G) and we shall draw up for each value of b two tables $T_1[1 : n]$ and $T_2[1 : n, 1 : f]$ where f represents the maximum number of indices to be found on one of the rows in G_2 (or G_1). All the distinct numbers appearing on the first rows b in G_1 are inscribed in increasing order in $T_1[1 : n]$. All the elements in G_2 on the first rows b associated to the element $T_1[i]$ — namely, the second index of every couple on the first levels b (of the graph G) having $T_1[i]$ as prime element in the couple — are inscribed in increasing order, on each row i, in $T_2[1 : n, 1 : f]$. The rest of the elements in T_1 an T_2 are zero elements.

Let $b = 3$ in our particular case. We have:

T_1:	T_2:
1	2 5 9 0 0 0 0 0
2	4 5 9 0 0 0 0 0
5	9 0 0 0 0 0 0 0
6	10 0 0 0 0 0 0 0
0	0 0 0 0 0 0 0 0
0	0 0 0 0 0 0 0 0
0	0 0 0 0 0 0 0 0
0	0 0 0 0 0 0 0 0
0	0 0 0 0 0 0 0 0
0	0 0 0 0 0 0 0 0

For each couple T_1, T_2, we draw up all the tables U_1, U_2 which are obtained from T_1 and T_2 as follows:

Out of T_2 we draw up all the tables U_2 for which the elements placed on parallel directions to the second diagonal of the square matrix formed by the first rows and columns containing non-zero elements of T_2, are equivalent. For instance, in the case of the above mentioned tables T_1 and T_2 we have:

U_1:	U_2:
1	2 5 9
2	5 9
5	9

Hence, on level $b = 3$ we can place the set of indices (1, 2, 5, 9).

In other cases, we shall generally have:

U_1:	U_2:
i_1	$i_2\ i_3 \ldots i_k$
i_2	$i_3\ i_4 \ldots i_k$
$\vdots$	$\vdots$
i_{k-1}	i_k

We shall prove the following theorem:

Theorem 2

The necessary and sufficient condition for all the couples $(i_1, i_2), \ldots, (i_{k-1}, i_k)$ to be present on a certain number of levels is that the table U_2 should be a right-angle isosceles triangle in which all the elements placed on parallel directions to the hypotenuse should be equal to respectively i_k, $i_{k-1}, \ldots i_3, i_2$.

PROOF

The condition is sufficient

Indeed, if the condition of the enunciation is fulfilled, then all the $\frac{k(k-1)}{2} = 1 + 2 + \ldots k - 1$ couples we can form with $i_1, i_2, \ldots, i_k$ are present on the levels considered.

The condition is necessary.

If we have $\frac{k(k-1)}{2}$ couples $(i_1, i_2), \ldots, (i_{k-1}, i_k)$ on a number of levels, then we can draw up the above mentioned tables U_1 and U_2 whose properties are enunciated in the theorem.

On the basis of Theorem 2 we find the sets of indices which can be put down on each level in the final table. However, only those not included in the sets of indices already placed on the previous levels or on the respective level will be written down in the final table. In the clustering program this computation is operated by the INCLUD procedure. The final table, in the case we have considered, is shown in Fig. 10.

1	1,9; 5,9;
2	1,5,9; 2,9; **
3	1,2,5,9; * 6,10; **
4	1,3; 3,4; ** 6,7; *
5	1,2,3,4,9; 6,7,10; **
6	1,2,3,4,5,9; * 1,10; 4,10; 4,5,10; 7,8;
7	1,4,5,9,10; 1,5,6,7,10; ** 3,5,7;
8	1,3,4,5,9,10; 1,5,6,7,9,10; 2,8,9; 2,3,8,9; 2,3,4,8,9; ** 3,5,7,9,10; 3,7,8,9;
9	1,2,3,4,5,8,9; * 1,3,4,5,7,8,9; 1,4,5,6,7,9,10; 1,2,9,10; 1,2,3,9,10; 1,2,3,4,9,10; 1,2,3,4,5,9,10; 3,4,5,7,9,10;
10	1,2,3,4,5,7,8,9; 1,4,5,6,7,8,9; 1,2,4,5,6,7,8,9; 1,2,3,4,5,7,9,10;
11	1,2,3,4,5,6,7,8,9,10; **

Fig. 10

We notice in this table that none of the sets of indices has been included in a previous set. One asterisk marks out the clusters making up the tree in Fig 11 a, two asterisks mark out the clusters making up the tree in Fig. 11 b. One can see from this very example the comparatively large number of trees which can be associated through absorptions to a graph G. The different possibilities to check Theorem 1, for each tree, are also to be noticed.

Other procedures, chiefly technical ones, as far as the program working out is concerned, are provided for in the clustering program. The PRES procedure, for instance, is used for eliminating the rows made up exclusively by 0 elements in G_1 and G_2.

*

The elements we have been discussing are common to all the clustering algorithms that have been worked out. As for the differences between those algorithms, they are implied by the standpoint adopted with regard to the problem of the number of sets of indices we propose to obtain in the final table and among which we select the clusters forming each tree.

Even if the number of elements m is about 30, that is comparatively small, we can obtain about 200 sets of indices on certain levels. The great number of clusters and the comparatively large one of possible trees create difficulties for the description of the solutions. In order to increase the number of elements m and keep up the complexity of the final table within acceptable limits of the necessary computation time, the following variants of the above mentioned fundamental algorithm have been worked out, which take into account the nature of the solutions needed by experts in various fields.

Clustering 1: Produces on each level almost all the sets out of which clusters can be selected. This program is advised for $m < 30$.

Clustering 2: Produces on each level at most m sets of indices (at the most 1 set for each possible initial index). We advise it for $m < 50$. The decrease of the number of trees supplied by the final table is comparatively much smaller than the decrease of the number of sets of indices on each level.

Clustering 3: Out of the sets produced by the clustering program 2, only the sets for which each initial element is not included in the previous sets written on the respective level, will be kept on each level. We advise it for the interval $m < 50$.

Clustering 4: On each level the sets of indices, excepting the last element of each set, are disjoint. We recommend it for the interval $m < 100$. In order to obtain a satisfactory number of trees out of the final table, the necessary number of levels out of which the clusters are selected may be, generally, smaller. For instance, if we have 100 le-

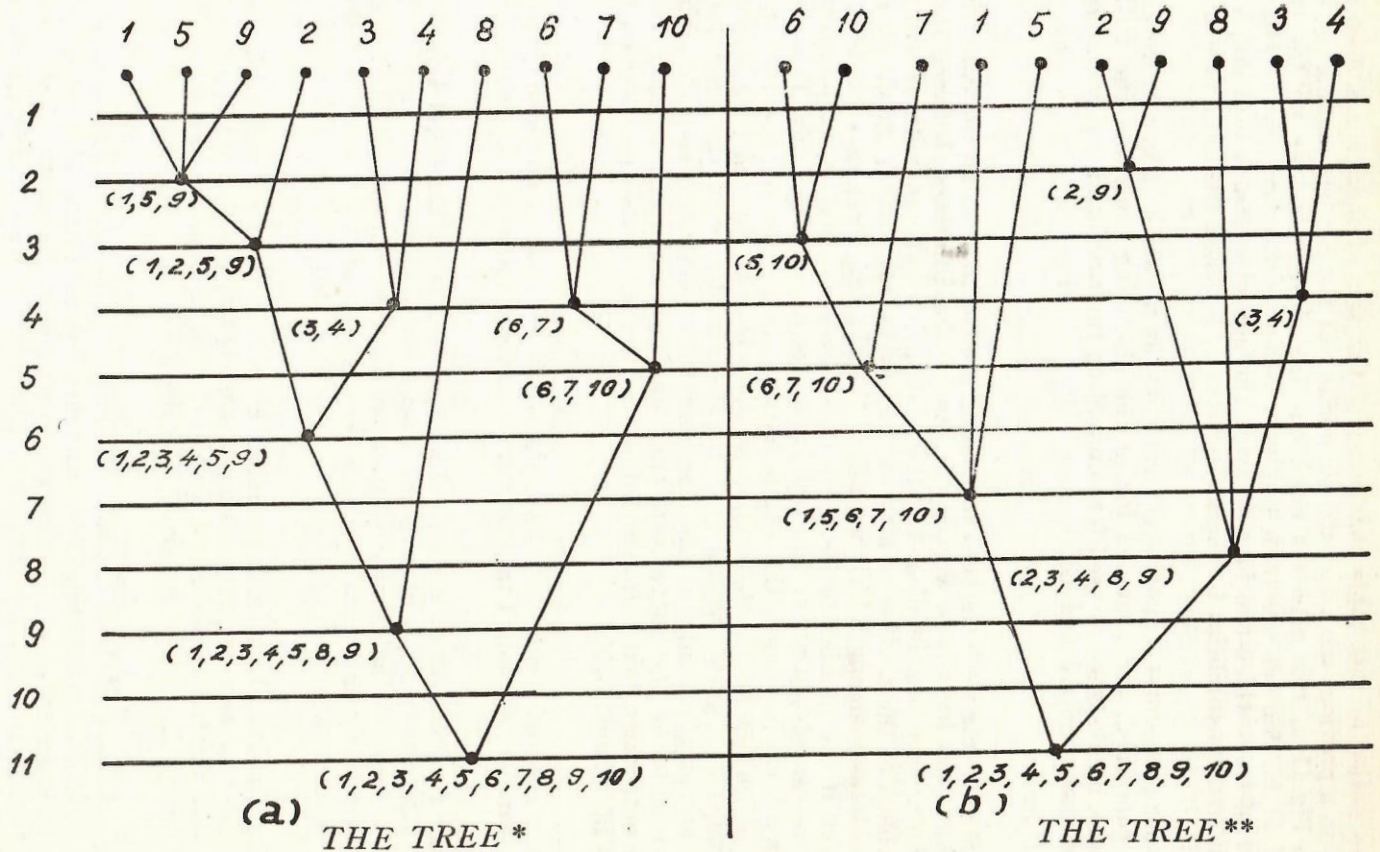
1 5 9 2 3 4 8 6 7 10
1
2
3
4
5
6
7
8
9
10
11
(1,5,9)
(1,2,5,9)
(3,4)
(6,7)
(6,7,10)
(1,2,3,4,5,9)
(1,2,3,4,5,8,9)
(1,2,3,4,5,6,7,8,9,10)
(a)
THE TREE *
6 10 7 1 5 2 9 8 3 4
(5,10)
(2,9)
(3,4)
(6,7,10)
(1,5,6,7,10)
(2,3,4,8,9)
(1,2,3,4,5,6,7,8,9,10)
(b)
THE TREE **

Fig. 11

vels, about 10—12 levels in the interval (40—60), that is, in the middle of the interval (1—100), will provide the clusters allowing to associate a sufficient number of trees — which constitute solutions to the clustering problem. In order to deal with higher values of m, an algorithm has been drown up in the program:

Clustering 5: Produces the sets of indices on a single level for each case of the clustering programs 1, 2, 3, 4. By repeating this program for a few levels we are able to obtain the sets of indices which will provide clusters for different trees. This program is advisable for $m < 1{,}000$.

For each of the above mentioned clustering programs various variants have been elaborated, according to the initial data of the clustering problem: whether the initial data are matrix E or D, whether these matrices are complete or not a.s.o.

The definition of the cluster given in this study covers a large range of clustering problems in various fields. This study contains, as a particular case, the definition of the cluster (class) as a "complete maximum subgraph of the matrix of similarities of the elements to be clustered" (See [1]). Such particular clusters may be obtained from definition 6, where only the elements placed at distance 1 (level 1) are taken into account.

With a view to make quite clear the concept of cluster used in this study — as related to previous conceptions — we shall describe the cluster as follows:

We shall generalize the notion of complete graph. A complete graph is one in which each vertex is connected with all the other vertices by a single branch.

Definition 8

We shall say that a graph is a complete p graph when each vertex is connected to any other vertex of this graph by p branches at most, but there s at least one other vertex to which it is connected by exactly p branches.

If we take the distance between 2 vertices of a graph to be the length of the minimum course (the minimum number of branches) between the 2 vertices, then, being a matrix D, we can associate to it a complete graph G_D interpreting the elements d_{ij} of the matrix as the distance between the vertices i and j of the graph G_D; namely, in the complete graph G_D associated to the matrix D we consider that the value d_{ij} is associated as a weight to each branch (i, j). We agree to interpret the weight associated to the branch (i, j) as the length of the minimum course between the nodes i and j. Considering this agreement, we can define the cluster as follows:

Definition 9

A cluster is a complete p subgraph $(p = 1, 2, \ldots, n)$ of graph G_D associated to matrix D, disjoint from the other complete p subgraphs.

In order to illustrate these agreements, let us take the following example: Let the matrix D be:

$$D = \begin{pmatrix} 0 & 2 & 3 & 4 \\ & 0 & 2 & 3 \\ & & 0 & 4 \\ & & & 0 \end{pmatrix}$$

The complete graph G_D having as associated weights the branches of the elements d_{ij}, is shown in Fig. 12.

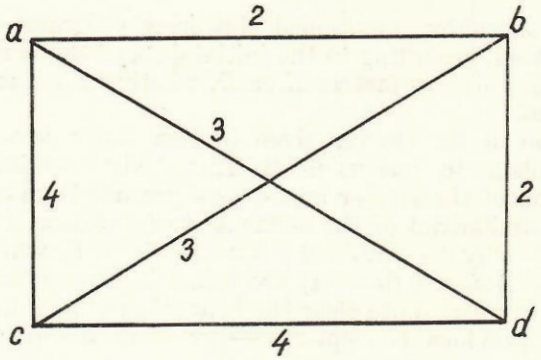

Fig. 12

On levels 1, 2, 3, 4 we have the following sets of elements: (Fig. 13).

We disregarded some subgraphs on each level. Thus, on level 3 we did not take into account the couples (b, c) and (a, d), and on level 4 we ignored the sets (a, c), (c, d), (a, b, c), (b, c, d), (a, c, d). We can consider, for instance, a tree composed of the clusters a, (b, d), (a, b, d), (a, b, c, d), (Fig. 14)

For smaller sets of elements (as for instance $m < 10$) this method of calculating the clusters can be applied without the help of a computer.

As for drawing up other algorithms in order to solve clustering problems we think worth mentioning the fact that the result of reducing a boolean

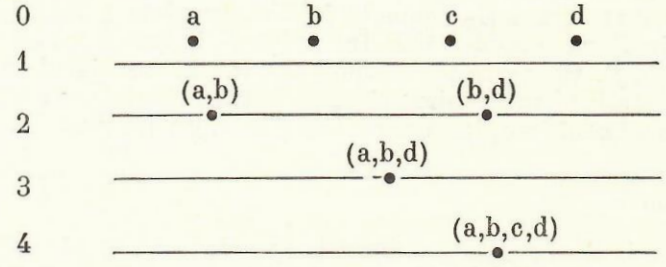

Fig. 13

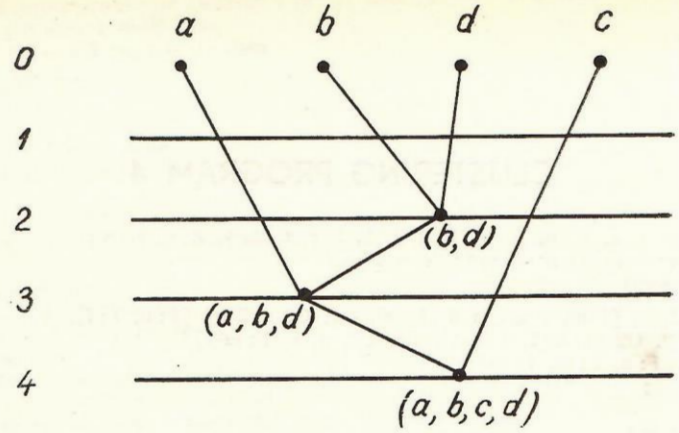

Fig. 14

function to its normal disjunctive form represents, under certain conditions, a solution to a clustering problem in that class of problems where matrix E has elements whose values are 0 or 1.

The way of solving clustering problems, that we have just described and which is based on finding the clusters of a tree, may be considered as complementary to the one, for instance, known in psychology as "factor analysis" And this, chiefly because the solving of problems, where the number n (of the properties in the set π) is a large one, is less easy through factor analysis than by means of the algorithms we have described — which practically need the same amount of time for solving high or low values of n; but the time required by these algorithms may rapidly increase along with the value of m (the number of elements in $\in$), whereas this increase affects to a smaller extent the algorithms of the factor analysis.

There is no end to the possible applications of the clustering method we have introduced. We shall only mention its main fields of application: biology, psychology, automatic translation, licences classification, automatic documentation, automatic summarizing, classification of technological processes, planning, clustering of prognosis problems in meteorology, acknowledgement of characters, working out of data, designing etc. Clustering processes are required to a greater extent in more and more areas; while in a field of activity the amount of information steadily increases, systematization and clustering become imperative in dealing with it.

In order to complete the description of the basic algorithm for the clustering of a set of elements with respect to a set of criteria, we shall annex herewith the Clustering Program 4 in the ALGOL code used by the Atlas computer.

CLUSTERING PROGRAM 4

```
begin integer f, o, e, b, p, m, n, h, i, j, k, l, c, t, x, y, r, u, c, w;
            m: = read; n = read; f: = 4^m;
begin integer array
            alpha [1:m, 1:m], a d [1: n], G1, G2, V2 q1 [1: n, 1: f],
            T2, U2 [1: m-1, 1:m-1], T1, U1, q, s [1: m-1],
            R[1: n, 1: m, 1: m];
procedure PRES;
begin
            j: = 1,
PUT: C: O; for i: = j step 1 until n do
            if G1 [i, 1] ≠ O then c: = c + 1;
            if c ≠ 0 then
            begin if G1 [j, 1] ≠ 0 then
            begin j: = j + 1; go to PUT
            end else
            begin for y: = j step 1 until n − 1 do
            begin a [y]: = a [y + 1];
            for i: = 1 step 1 until f do
            begin G1 [y, i]: = G1 [y + 1, i];
            G2 [y, i]: = G2 [y + 1, i];
              end
          end;
            a[n]: = 0;
            for i: = 1 step 1 until f do
            for i: = 1 step 1 until f do
            G1 [n, i]: = [G2 n, i]: = 0;
            go to PUT
          end
        end;
          n: j-1; go to CYCLE
        end of procedure, PRESS;
        procedure TRIANG;
begin
      for j: = 1 step 1 until m-1 do
begin U1 [j]: = 0; s[j]: = q [i];
      for i: = 1 step 1 until m-1 do
        U2 [j, i] : = 0
end;
      e: = q [1];
    for j: = 1 step 1 until e do
begin U1 [j]: = T1 [j];
    for i: = 1 step 1 until; q [j] do
        U2 [j, i]: = T2 [j, i]
end:
h: = t : = 1;
F: if U2 [h, t + 1] ≠ U2 [h + 1, t] then
    begin c: = 0; for j: = t + 1 step 1 until s [h + 1] do
      begin if U2 [h, t + 1] = U2 [h + 1, j] then c: = c + 1
    end;
end else go to T;
if c ≠ 0 then go to DOWN;
```

```
comment One crosses out the element j in U1 and the row j in U2;
UP: for j: = 2 step 1 until e do
begin if U1 [j] = U2 [h, t + 1] then
            U1[j] : = 0
end;
c: = 0;
for y: = 1 step 1 until e do
     begin if U1 [y] = 0 then c: = c + 1
     end;
     if c = 0 then go to H;
comment One renumbers the elements in U1 and the rows in U2:
       j : = 2;
     G: if j < e then
     begin if U1 [j] ≠ 0 then
     begin j: = j + 1; go to G
     end else
     begin for y: = j step 1 until e do
     begin s [y]: = s[y + 1]: U1 [y]: = U1 [y + 1];
          for i: = 1 step 1 until m-1 do
                    U2 [y, i]: + U2 [y + 1, i]
        end; e: = e − 1; go to H
         end
       end;
     if U1 [e] = 0 then
       begin s [e]: = 0; e: = e − 1
  end;
comment The elements out from cluster are crossed out, the elements in the row in discussion
     in U2 are renumbered and the lengths of these rows are adjusted;
H: 0 = U2 [h, t + 1];
     if o = 0 then
          begin if h < e-1 then
          begin t: = 1; h: = h + 1; go to T
          end else go to LEAP
          end;
       for y: = 1 step 1 until e do
          begin for i: = 1 step 1 until s[y] do
          begin if U2 [y, i] = 0 then
            begin for j: = i step 1 until s[y] do
                    U2[y, j]: = U2 [y, j + 1];
                    s[y]: = s[y] − 1; go to K
              end
            end;
K: end; go to L;
DOWN: o: = U2 [h + 1, t];
          if o = 0 then
          begin if h < e − 1 then
          begin t: = 1; h: = h + 1; go to T
          end else go to LEAP
         end;
         for y: = 1 step 1 until e do
            begin for i: = 1 step 1 until s[y] do
            begin if U2 [y, 1] = o then
            begin for j: = i step 1 until s[y] do
                    U2[y, j]: = U2 [y, j + 1];
                      s[y]: = s[y] − 1; go to REP
            end
         end;
     REP: end;
comment If some rows become zero-rows in U2 we renumber the elements in U1 and the
     rows in U2 again;
L: c: = 0;
```

```
      for i: = 1 step 1 until e do
            begin if U2 [i, 1] = 0 then c: = c + 1
            end;
            if c= o then go to T;
            j: = 2;
      AGAIN: if j<e then
      begin if U2 [j, 1,] ≠ then
            begin j: = j + 1; go to AGAIN
            end else
            begin for y: = j step 1 until e do
            begin s[y]: = s [y + 1]; U1[y]: = U1[y + 1];
              for i: = 1 step 1 until m-1 do
                    U2 [y, i]: = U2 [y + 1, i]
            end
            e: = e − 1; c: = 0;
            for i: = j step 1 until e do
              begin if U2 [i, 1,] = 0 then c: = c + 1
              end; if c ≠ 0 then go to AGAIN
          end
      end;
comment We continue to compare the elements in the rows (h — 1, h) and after this we
      pass to the couple (h, h + 1) etc;
T: c: = 0; for w: = 1 step 1 until s [1] −1 do
            for u: = 1 step 1 until w do
begin
      if U2 [1, w + 1] ≠ U2 [1 + u, w − u + 1] then c: = c + 1
end;
      if c = 0 then
      q1[b, p]: = s [1]; if b = 1 and p = 1 then
begin for y: = 1 step 1 until q1 [b, p] do
      R[b, p, y]: = U1[y];
      R[b, p, q1 [b, p] + 1] : = U2 [q1 [b, p], 1];
            for y: = 1 step 1 until q1 [b, p + 1] do
              print (R[b, p, y], 2, 0); space (7);
                  d[b]: = d[b] + 1; p : = p + 1;
LEAP: go to J
      end else
      INCLUD
end;
            if U2 [h, t + 1] ≠ U2 [h + 1, t] then go to F ;
if t < s [h] − 1 then
            begin t: = t + 1; go to F
            end else if h<e − 1 then
            begin h: = h + 1; t: = 1;. go to F
            end else go to LEAP
end of procedure TRIANG;
procedure INCLUD
begin
comment We write the new string R [b, p, y];
or j: = 1 step 1 until q1 [b, p] do R [b, p, j] : = U1 [j];
      R[b, p, q1 [b, p] + 1] : = U2 [q1 [b, p], 1];
   = r : = y : = 1; c: = 0;
comment We compare the new string with all the anterior strings and verify if is included
      in an anterior one;
E: if R[u, r, 1] = 0 then
      begin if p = 1 and u = b−1 then
      go to E1 else
      begin u: = u + 1; r: = y : = 1; go to E
      end
      end;
```

```
for h: = 1 step 1 until q1 [u, r] + 1 do
begin
if R [b, p, y] = R [u, r, h] then  c: = c + 1
efnd; if c = 0 then
if R [b, p, y] = R [u, r, h] then c: = c + 1
end; if c = 0 then
     begin if r< d[u] then
           begin r: = r + 1; y: = 1; go to E
           end else
           begin if u = b then go to E1
           else
           begin if p=1 and u=b−1 then go to E1
             else
             begin y: = r : = 1; u: = u + 1;
                    go to E
             end
           end
     end else if y ⩽ q1 [b, p] then
begin y: = y + 1; c: = 0; go to E
end;
     for j: = 1 step 1 until q1 [b, p] + 1 do
                    R[b, p, j]: = 0;
               q1 [b, p[ : = 0;
          go to J;
     E1: for j: = 1 step 1 until q1 [b, p] + 1 do print (R [b, pj,] , 2, 0);
          p: = p + 1; space (7); d[b]: = d [b] + 1;
          go to J
end of procedure INCLUD;
comment START;
     for i: = 1 step 1 until m-1 do
     for j: = 1 step 1 until m-i do
          alpha [i, j]: = read;
     for k: = 1 step 1 until n do
     begin a[k] : = 1; d[k[ : = 0
     end; We get from the matrix Mo = alpha, [i, j], the graph
          G(G1 şi G2);
     for i: = 1 step 1 until n do
     for j: = 1 step 1 until f do
          G1 [i, j]: = G2 [i, j] : = 0;
     for i: = 1 step 1 until m-1 do
     for j: = 1 step 1 until m-i do
       begin k: = alpha [i, j];
          G1 [k, a [k]]: = i; G2 [k, a [k]] : = j + i; a[k]: = a [k] + 1;
     if a[k] > f then
          begin write text (-G-);
                    go to STOP
          end
       end;
     for k: = 1 step 1 until n do
       begin new line (1); print (k, 2, 0); space (7);
            print (G1 [k, 1], 1, 0); a [k]: = a [k] − 1
          end; new line (2);
          PRES;
          CYCLE: for b: = 1 step 1 until n do
          begin
               j: = l : = p : = 1;
comment We use (count) the elements which are equal with l on the level b an dall the in-
     ferior levels in G1 in order to get T1 and T2;
IN B: for h: = 1 step 1 until m-1 do
     begin T1[h]: = q[h] : = 0;
```

```
      for i: = 1 step 1 until m-1 do
                T2[h, i]: = 0
      end;
    A: for h: = 1 step 1 until b do
        for i: = 1 step 1 until a [h]do
            V2 [h, i]: = 0; c: = 0;
       for h: = 1 step 1 until b do
       for x: = 1 step 1 until a[h] do
       begin if G1 [h, x] = 1 then c: = c + 1
          end;
      if c ≠ 0 then go to BUILD;
      if j = 1 then
      begin if l < m-1; then
      begin l: = l + 1; go to A
      end else go to OUT B
      end;
A1: if l< m-1 then
      begin l: = l + 1; c: 0;
        for h: = 1 step 1 until q [1]do
        begin if 1 = T2 [1, h] then  c: = c + 1
        end;
        if C ≠ 0 then
          begin for i: = j − 1 step 1 until q [1] do
             T2[1, i]: = T2 [1, i + 1];
             q[1]: = q[1] − 1; go to A
             end else go to A1
        end;
      go to D1;
comment One constructs the array V2;
BUILD: q[j]: = c;
      for h: = 1 step 1 until b do
      for x: = 1 step 1 until a [h] do
          if G1 [h, x] = l then V2 [h, x]: = G2 [h, x];
comment One constructs the element j in T1;
          T1[j]: = l;
comment From the array V2 we get the row j in T2 (elements in natural order);
          k: = x: = t : = 1;
       C: if V2 [k, x] = 0 then
      begin if x < a [k] then
        begin x: = x + 1; go to C;
        end else if k < b then
        begin k: = k + 1; x: = 1; go to C
        end else go to D
end;
      C1: T2 [j, t]: = V2 [k, x];
        for h: = 1 step 1 until b do
        for i: = 1 step 1 until  a[h] do
          begin if V2 [h, i] ≠ 0 and V2 [h, i] < T2 [j, t] then
           T2 [j, t]: = V2 [h, i]
      end;
      v: = V2 [k, x];
        for h: = 1 step 1 until b do
        for i: = 1 step 1 until a [h] do
        begin if V2 [h, i] = T2 [j, t] then
             V2[h, i] : = 0
        end;
        if v ≠ T2 [j, t] then
        begin t: = t + 1; go to C1
        end;
        t: = t + 1; if x < a[k] then
        begin x: = x + 1; go to C
```

```
        end else if k < b then
        begin k: = k + 1; x: = 1; go to C
        end
the following element in T1 and row in T2;
D: if j< q [1] and T2 [1, j] < m then
     begin l: = T2 [1, j]; j: j + 1; go to CHECK
     end;
D1: TRIANG;
J: if T1 [1] < m-1 then
     begin j: = l; l: = T1 [1] + 1;
CHECK: c: = 0;
        for i: = 1 step 1 until d [b] do
        for h: = 1 step 1 until q1 [b, i] + 1 do
        begin if l = R [b, i, h] then c: = c + 1
        end;
        if c ≠ 0 then
     begin j: = 1, l: = l + 1; go to CHECK
        end;
        if j = 1 then go to IN B
             else go to A
end;
OUT B: space (7); write text (— ROW —);
          print (b, 2, 0); new line (4)
end
end;
STOP: end
```

REFERENCES

1. BONNER, R. E., *On Some Clustering Techniques*, IPM Journal, January 1964.
2. CONSTANTINESCU, P., *The Classification of a Set of Elements with respect to a Set of Properties*, Computer Journal, January 1966
3. CONSTANTINESCU, P. and MOSS, G., *A Clustering Procedure tested on a Set o Human Chromosomes*, Bio-Medical Engineering (in press)
4. CONSTANTINESCU, P., *On Cluster Analysis*, British Journal for Statistical Psychology (in press)
5. CONSTANTINESCU, P., *Introducere în programarea automată*, Editura Tehnică, Bucureşti, 1967.

SUR LA CLASSIFICATION D'UN ENSEMBLE D'ELEMENTS PAR RAPPORT A UN ENSEMBLE DE PROPRIETES

Vu le volume croissant d'informations survenant dans divers problèmes techniques, économiques et scientifiques le solutionnement de ces problèmes exige l'emploi, aussi large que possible, de certains schémas pour la classification des données. Le processus de classification des grands ensembles est analogue à celui qui consiste à diviser différents ensembles doués de structures algébriques.

Dans cette étude, l'auteur donne la définition de la „gerbe" (classe) et examine les possibilités de résoudre le problème de la classification. Certains algorithmes utilisés en vue du solutionnement de ce problème y sont ensuite présentés ainsi que la description complète d'un algorithme en langage ALGOL, à savoir la restriction ALGOL employée pour un calculateur ATLAS.

КЛАССИФИКАЦИЯ НЕКОТОРЫХ МНОЖЕСТВ ЭЛЕМЕНТОВ В СООТНОШЕНИИ С МНОШЕСТВОМ ХАРАКТЕРИСТИК

Увеличение количества информации, касающейся разных вопросов техники, экономики, науки требует применения всевозможных схем классификации данных при решении этих проблем. Процесс классификции больших множеств является аналогичным процессу разделеления различных множеств, содержащих алгебраические структуры.

В исследовании дается определение снопа (класса) и изучается вопрос наличия решения проблемы классификации. Показаны алгоритмы для решения этой проблемы классификации п полностью характеризуется алгоритм на языке ALGOL, а именно ограничение ALGOL, ириненяемое к счетно-вычислительной машине ATLAS.