REVEALING CONCEPTUAL STRUCTURE IN DATA BY INDUCTIVE INFERENCE

by

Ryszard S. Michalski
Robert Stepp

Chapter in the book, "Machine Intelligence 10," D. Michie, J. E. Hayes and Y-H Pao, Editors. John Wiley & Sons Publishing, New York, 1982.

MACHINE INTELLIGENCE 10

Edited by

J. E. HAYES
Research Associate
University of Edinburgh

DONALD MICHIE
Professor of Machine Intelligence
University of Edinburgh

and

Y-H PAO
Dively Distinguished Professor of Engineering, and
Professor of Electrical Engineering and Computer Science
Case Western Reserve University, USA

ELLIS HORWOOD LIMITED
Publishers · Chichester

Halsted Press: a division of
JOHN WILEY & SONS
New York · Brisbane · Chichester · Toronto

8

Revealing conceptual structure in data by inductive inference

R. S. Michalski and R. Stepp
University of Illinois
Urbana, USA

ABSTRACT

In many applied sciences there is often a problem of revealing a structure underlying a given collection of objects (situations, measurements, observations, etc.). A specific problem of this type is that of determining a hierarchy of meaningful subcategories in such a collection. This problem has been studied intensively in the area of cluster analysis. The methods developed there, however, formulate subcategories ('clusters') solely on the basis of pairwise 'similarity' (or 'proximity') of objects, and ignore the issue of the 'meaning' of the clusters obtained. The methods do not provide any description of the clusters obtained. This paper presents a method which constructs a hierarchy of subcategories, such that an appropriately generalized description of each subcategory is a single conjunctive statement involving attributes of objects and has a simple conceptual interpretation. The attributes may be many-valued nominal variables or relations on numerical variables. The hierarchy is constructed in such a way that a flexibly defined 'cost' of the collection of descriptions which branch from any node is minimized.

Experiments with the implemented program, CLUSTER/paf, have shown that for some quite simple problems the traditional methods are unable to produce a structuring of objects most 'natural' for people, while the method presented here was able to produce such a solution.

1. INTRODUCTION

Computer programs able to reveal an underlying conceptual structure in a set of data can be useful components of AI systems. Knowledge about the structure of the data can help, for example, in reducing the search space in problem solving, in dividing knowledge acquisition tasks into useful subcases, or in organizing large databases (or rule bases) and summarizing their contents. It is believed that the problem of intelligent structuring of data by computer will become one of the important tasks for AI research in the '80s.

A simple form of data structuring is *clustering*, which is a process of determining a hierarchy of subcategories within a given collection of objects. In the traditional methods of clustering, the basis for forming the subcategories is a 'degree of similarity' between objects: the subcategories are collections of objects whose intra-group similarity is high and inter-group similarity is low. The process of determining the hierarchy of subcategories can be done either in a 'bottom-up' or a 'top-down' fashion. The bottom-up methods (called 'hierarchical' in the literature on cluster analysis) recursively merge single objects or collections of objects into larger collections, ending with the original complete set of objects at the top of the hierarchy (dendrogram). The top-down ('non-hierarchical') methods recursively split the starting collections(s) of objects into subgroups, ending when single objects are assigned to the leaves of the hierarchy.

The bottom-up methods are mostly used in numerical taxonomy. Depending on the way in which object-to-group and group-to-group degrees of similarlity are calculated, different versions of the technique are obtained, such as 'single' linkage, 'complete' linkage, or 'average' linkage [14].

The top-down methods generally operate by making a series of cluster boundary perturbations while searching for the groupings which exhibit minimal dispersion of objects around the cluster means. Some top-down methods, e.g., ISODATA, have additional heuristics which help to select the optimal number of clusters.

The allied process of clustering features rather than objects involves the techniques of factor analysis and multi-dimensional scaling. Many clustering methods are sensitive to the irrelevant variables present in the data. Factor analysis and multidimensional scaling can be used to select the most 'relevant' variables before proceeding to cluster the objects. These methods, however, are designed primarily for numerical variables. They cannot handle many-valued nominal variables, which occur often in AI applications.

All the traditional techniques have one major disadvantage. Since the only basis for forming clusters is the degree of similarity (between objects or groups of objects), the resulting clusters do not necessarily have any simple conceptual interpretation. The problem of 'meaning' of the obtained clusters is simply left to the researcher. This disadvantage is a significant one because a researcher typically wants not only clusters, but also wants an explanation of them in human terms.

This paper describes a method of determining a hierarchical structure underlying a given collection of objects, in which each node represents a certain generalized description of a corresponding subcategory of objects. The descriptions are conjunctive concepts involving attributes of objects. The attributes can be nominal variables or relations on numerical variables. Such descriptions have a very simple human interpretation. The presented method is an example of what we call generally a *conceptual clustering*.

The label *conceptual clustering* can be applied to any method which determines a structure in a collection of objects, in which the nodes represent 'concepts' characterizing the corresponding subcategories, and the links represent relationships between the concepts. (By the term 'concept' we mean a human oriented description, which involves properties of objects and relations among them.) In the method described, the concepts are conjunctive descriptions of subcategories, and the links interconnecting the levels of the hierarchy represent the 'next level of generality'[1] relation between the descriptions (i.e., the predecessor description is a generalization of all successor descriptions).

Section 2, which follows, discusses the distinction between the conventional similarlity measure and the 'conceptual cohesion' measure, which underlines the presented method. Section 3 gives the basic terminology of the descriptive language used (the variable-valued logic system VL_1), and of the inductive inference technique. Section 4 gives an overview of the conceptual clustering algorithm and and its implementation in the program CLUSTER/paf. Finally, section 5 presents an example illustrating the method and compares the results obtained from conceptual clustering to those obtained from numerical taxonomy.

2. THE SIMILARITY MEASURE VERSUS CONCEPTUAL COHESIVENESS

The techniques of traditional cluster analysis are distinctly nonconceptual because they do not attempt to discover the meaning of the clusters or endeavour to arrange objects into those subcategories with the most succinct conceptual interpretation. As mentioned before, this behaviour is attributed to the use of standard distance or similarity measures as the only basis for clustering. In order to be able to do 'conceptual clustering,' one has to know more than the degree of similarlity between any two objects or groups of objects. Specifically, the notion of similarity should be replaced by a more general notion of 'conceptual cohesiveness', which we will now describe.

The similarlity between any two objects in the population to be clustered is characterized in the conventional data analysis methods by a single number – the value of the similarity function applied to symbolic descriptions of objects ('data points'). These descriptions are typically vectors, whose components represent scores on selected qualitative or quantitative variables used to describe objects. Frequently a reciprocal of a distance measure is used as a similarity function. The distance measure for such purposes, however, does not have to satisfy all the postulates of a distance function (specifically, the triangle inequality). A comprehensive review of various distance and similarity measures is provided in Diday & Simon [2] and Anderberg [1].

As mentioned before, the conventional measures of similarity are 'context-free,' i.e., the similarlity between any two data points A and B is a function of these points only:

$$\text{Similarity}(A, B) = f(A, B) \qquad (1)$$

Recently some authors [4] have been introducing 'context-sensitive' measures of similarity:

$$\text{Similarity}(A, B) = f(A, B, E) \tag{2}$$

where the similarity between A and B depends not only on A and B, but also on other points ('context points') in the collection to be clustered E.

Both previous clustering approaches cluster data points only on the basis of knowledge of the individual data points. Therefore such methods are fundamentally unable to capture the 'Gestalt property' of objects, i.e., a property which is characteristic of certain configurations of points considered as a whole, but not when considered as independent points. In order to detect such properties, the system must know not only the data points, but also certain 'concepts'. To illustrate this point, let us consider a problem of clustering data points in Fig. 1.

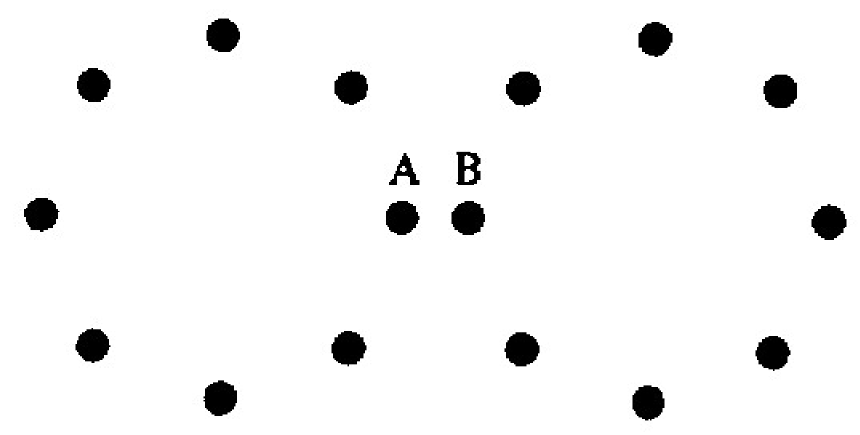


Fig. 1 – An illustration of conceptual clustering.

A person considering the problem in Fig. 1 would typically describe it as 'two circles'. Thus, the points A and B, although being very close, are placed in separate clusters. Here, human solution involves partitioning the data points into groups not on the basis of pairwise distance between points, but on the basis of 'concept membership'. That means that the points are placed in the same cluster if together they represent the same concept. In our example, the concepts are circles.

This idea is the basis of conceptual clustering. From the view of conceptual clustering, the 'similarity' between two data points A and B, which we will call the *conceptual cohesiveness*, is a function not only of these points and the context points in E, but also of a set of concepts C which are available for describing A and B together:

$$\text{Similarity}(A, B) = f(A, B, E, C) \tag{3}$$

To illustrate a 'conceptual cohesiveness' measure, let us assume that C is the set of concepts which are geometrical figures, such as circles, rectangles, triangles, etc. A measure of conceptual cohesiveness can be defined, for example, as

$$S(A, B, E, C) = \max_{i} \frac{\#e(i) - 1}{\text{area}(i)} \tag{4}$$

where i indexes all geometrical figures which are specified in C and which cover points A and B,

#e(i) is the total number of data points from E covered by figure i,

area(i) is the area of figure i.

(The constant "-1" in the numerator assures that the 'conceptual cohesiveness' reduces to a conventional similarity measure, i.e., a reciprocal of distance, when no context points in E are taken into consideration and C is a straight line of unit thickness intersecting the data points.)

This measure is mentioned solely to illustrate the difference between traditional similarity and conceptual cohesiveness. It is not used to actually implement the method of conceptual clustering described here.

The idea of conceptual clustering has been introduced by Michalski [12], and evolved from earlier work by him and his collaborators on generating the uniclass covers' (i.e., disjunctive descriptions of a class of objects specified by only positive examples of the class). A computer program and various experimental results on determining uniclass covers are described by Stepp [15].

3. TERMINOLOGY AND DEFINITIONS

In this section, relevant formal concepts and definitions will be briefly summarized. A complete presentation can be found in [12].

Value set (or domain) of a variable

Let $x_1, x_2, \ldots, x_n$ denote discrete variables which are selected to describe objects in the population to be clustered. For each variable a *value set* or *domain* is defined, which contains all possible values this variable can take for any object in the population. We shall assume that the value sets of variables x_i, $i = 1, 2, \ldots, n$ are finite, and therefore can be represented as:

$$D_i = \{0, 1, \ldots, d_i - 1\}, \quad i = 1, 2, \ldots, n\,. \tag{5}$$

In general, the value sets may differ not only with respect to their size, but also with respect to the structure relating their elements (reflecting the scale of measurement). In this paper we will distinguish only between nominal (qualitative) and linear (quantitative) variables whose domains are unordered and linearly ordered sets, respectively.

Event space

An *event* is defined as any sequence of values of variables $x_1, x_2, \ldots, x_n$:

$$e = (r_1, r_2, \ldots, r_n) \tag{6}$$

where $r_i \in D_i$, $i = 1, 2, \ldots, n$.

The set of all possible events, Σ, is called the *event space*:

$$\Sigma = \{e_i\}_{i=1}^{\mathrm{d}} \tag{7}$$

where $\mathrm{d} = d_1 \cdot d_2 \cdot \ldots \cdot d_n$ (the *size* of the event space).

Syntactic distance

Given two events e_1, e_2 in Σ, the *syntactic distance*, $\delta(e_1, e_2)$, between e_1 and e_2 is defined as the number of variables which have different values in e_1 and e_2.

Selectors

A relational statement

$$[x_i \# R_i] \tag{8}$$

where R_i is one or more elements from the domain of x_i, and $\#$ stands for the relational operator $=$ or $\neq$ is called a VL_1 *selector*† or, briefly, a *selector.* The selector $[x_i = R_i]$ ($[x_i \neq R_i]$) is interpreted as 'value of $x_i \in \{R_i\}$' ('value of $x_i \notin \{R_i\}$'). In the case of linear variables, the operator "=" in $[x_i = R_i]$ can be replaced by relational operators $\geqslant, >, <, \leqslant$ for an appropriate R_i, as indicated below.

Here are a few examples of a selector, in which variables and their values are represented by linguistic terms:

[height = tall]	
[length $\geqslant$ 2]	
[colour = blue, red]	(colour is blue or red)
[size $\neq$ medium]	(size is not medium)
[weight = 2..5]	(weight is between 2 and 5, inclusively)

Complexes

A logical product of selectors is called a *logical complex* (*l-complex*):

$$\bigwedge_{i \in I} [x_i \# R_i] \tag{9}$$

where $I \subseteq \{1,2,\ldots,n\}$, and $R_i \subseteq D_i$. An event e is said to *satisy* an *l-complex* if values of variables in e satisfy all the selectors in the complex. For example, event $e = (2,7,0,1,5,4,6)$ satisfies the l-complex $[x_1 = 2,3]$ $[x_3 \leqslant 3]$ $[x_5 = 3..8]$ (concatenation of selectors implies conjunction).

An l-complex can be viewed as an exact symbolic representation of the events which satisfy it. For example, the above l-complex is the symbolic representation of all events for which x_1 is 2 or 3, x_3 is smaller than or equal to 3, and x_5 is between 3 and 8.

† VL_1 stands for variable-valued logic system VL_1 which uses such selectors.

Any set of events for which there exists an l-complex satisfied by these events and only by these events is called a *set complex* (*s-complex*). Henceforth, if α is an s-complex, then by α we will denote the corresponding l-complex.

Quantitative properties of clusters

Let E be a set of events in Σ, which are data points to be clustered. The events in E are called *data events* (or *observed events*), and events in $\Sigma \backslash E$ (i.e., events in Σ which are not data events) are called *empty events* (or *unobserved events*). Let α be a complex which covers some data events and some empty events. The number of data events (*p*oints) in α is denoted by $p(\alpha)$. The number of empty events in α is called the *sparseness* and denoted by $s(\alpha)$. The total number of events in α is thus $t(\alpha) = p(\alpha) + s(\alpha)$.

If s-complex α is represented by l-complex $\hat{\alpha} = \bigwedge_{i \in I} [x_i \# R_i]$, the number $t(\alpha)$ can be computed as:

$$t(\alpha) = \prod_{i \in I} c(R_i) \cdot \prod_{i \notin I} d_i \tag{10}$$

where $I \subseteq \{1, 2, \ldots, n\}$,
$c(R_i)$ is the cardinality of R_i,
d_i is the cardinality of the value set of variable x_i.

The l-complex $\hat{\alpha}$ can be viewed as a generalized description of the data points in α. The sparseness, as defined above, can be used as a simple measure of the degree to which the description $\hat{\alpha}$ generalizes over the data points. If the sparseness is zero, then the description covers only data points ('zero generalization'). As the sparseness for a given complex increases, so does the degree to which the description $\hat{\alpha}$ generalizes over the data points. A formal definition of the 'degree of generalization,' based on the information-theoretic uncertainty of the location of data points in α, is given in [12].

Star

The *star* $G(e|F)$ of e against the event set F is the set of all maximal under inclusion complexes covering the event e and not covering any event in F. (A complex α is *maximal under inclusion* with respect to property P, if there does not exist a complex α^* with property P, such that $\alpha \subset \alpha^*$).

Cover

Let E_1 and E_2 be two disjoint event sets, $E_1 \cap E_2 = \phi$. A *cover* $\mathrm{COV}(E_1|E_2)$ of E_1 *against* E_2, is any set of complexes, $\{\alpha_j\}_{j \in J}$, such that for each event $e \in E_1$ there is a complex α_j, $j \in J$, covering it, and none of the complexes α_j cover any event in E_2. Thus we have:

$$E_1 \subseteq \bigcup_{j \in J} \alpha_j \subseteq \Sigma \backslash E_2 \ . \tag{11}$$

A cover in which all complexes are pairwise disjoint sets is called a *disjoint cover.* If set E_2 is empty, then a cover $\mathrm{COV}(E_1|E_2) = \mathrm{COV}(E_1|\phi)$ is simply denoted as $\mathrm{COV}(E_1)$. A disjoint cover $\mathrm{COV}(E)$ which consists of k complexes is called a *k-partition* of E.

4. THE METHOD AND IMPLEMENTATION

This section describes an algorithm for conjunctive conceptual clustering, which consists of an outer layer and an inner layer, described in sections 4.1 and 4.2 below. Section 4.3 describes the procedure used to construct the k-partition from stars which optimizes a certain criterion of clustering quality.

4.1 Inner layer

The inner portion of the algorithm called PAF was introduced in [12] as a 'constrained' conjunctive conceptual clustering technique. Given a set of data events, E, to be clustered and an integer, k, PAF partitions the set E into k clusters, each of which has a conjunctive description in the form of a complex. (In the complete algorithm, E and k are determined by the outer algorithm). The obtained partition is optimal or suboptimal with regard to a user selected measure of clustering quality.

The general structure of PAF is based on the dynamic clustering method developed by Diday and his collaborators (Diday & Simon [1], Hanani [5]). Underlying the notions of the dynamic clustering method are two functions:

g: the *representation function*, which, given a k-partition of E, produces a set of k cluster representations which best describe the clusters,

f: the *allocation function*, which, given a set of k cluster representations, produces a k-partition in which each cluster is composed of those objects which best fit the corresponding cluster representations.

The method works iteratively, starting with a set of k initial, randomly chosen cluster representations. A single iteration consists of an application of function f to the given representations, and then of function g to the obtained partition. Each iteration ends with a new set of representations. The process continues until the chosen criterion of clustering quality, W, ceases to improve. (Criterion W measures the 'fit' between a partition and its representation.) It has been proved that this method always converges to a local optimum [2].

In PAF, the notions of dynamic clustering are not rigorously followed, although the general approach is similar. PAF has a step which is analogous to function g, which from a given set of clusters of objects, produces the best representations of them (in the form of complexes). The process involves selecting a representative event (seed) from each cluster and covering one seed against the others to generate generalized descriptions of the events. This is done by procedures STAR and NID described in section 4.3 below. The function f of dynamic clustering is represented in PAF by a procedure which, for a given complex, determines the set of covered data events.

PAF will now be described by showing how it operates on the data given in Fig. 2, with parameter k set to 2 (i.e., the algorithm will split the data into two subcategories), and with minimizing total sparseness as the criterion of clustering quality. There are ten objects, each described by the values of four variables x_1, x_2, x_3, and x_4, with three-valued domains, $D_i = \{0, 1, 2\}$, $i = 1, 2, 3, 4$. Variables x_1 and x_2 are linear, and variables x_3 and x_4 are nominal.

Event	x_1	x_2	x_3	x_4
e_1	0	0	0	1
e_2	0	1	0	0
e_3	0	2	1	2
e_4	1	0	0	2
e_5	1	2	1	1
e_6	2	0	1	0
e_7	2	1	0	1
e_8	2	1	1	2
e_9	2	2	0	0
e_{10}	2	2	2	2
Value set type:	L	L	N	N

(L – linearly ordered; N – nominal)

Fig. 2 – An exemplary data set describing ten objects using four variables.

Figure 3 presents the data set from Fig. 2 graphically, using a planar representation of the event space spanned over variables x_1, x_2, x_3, and x_4.

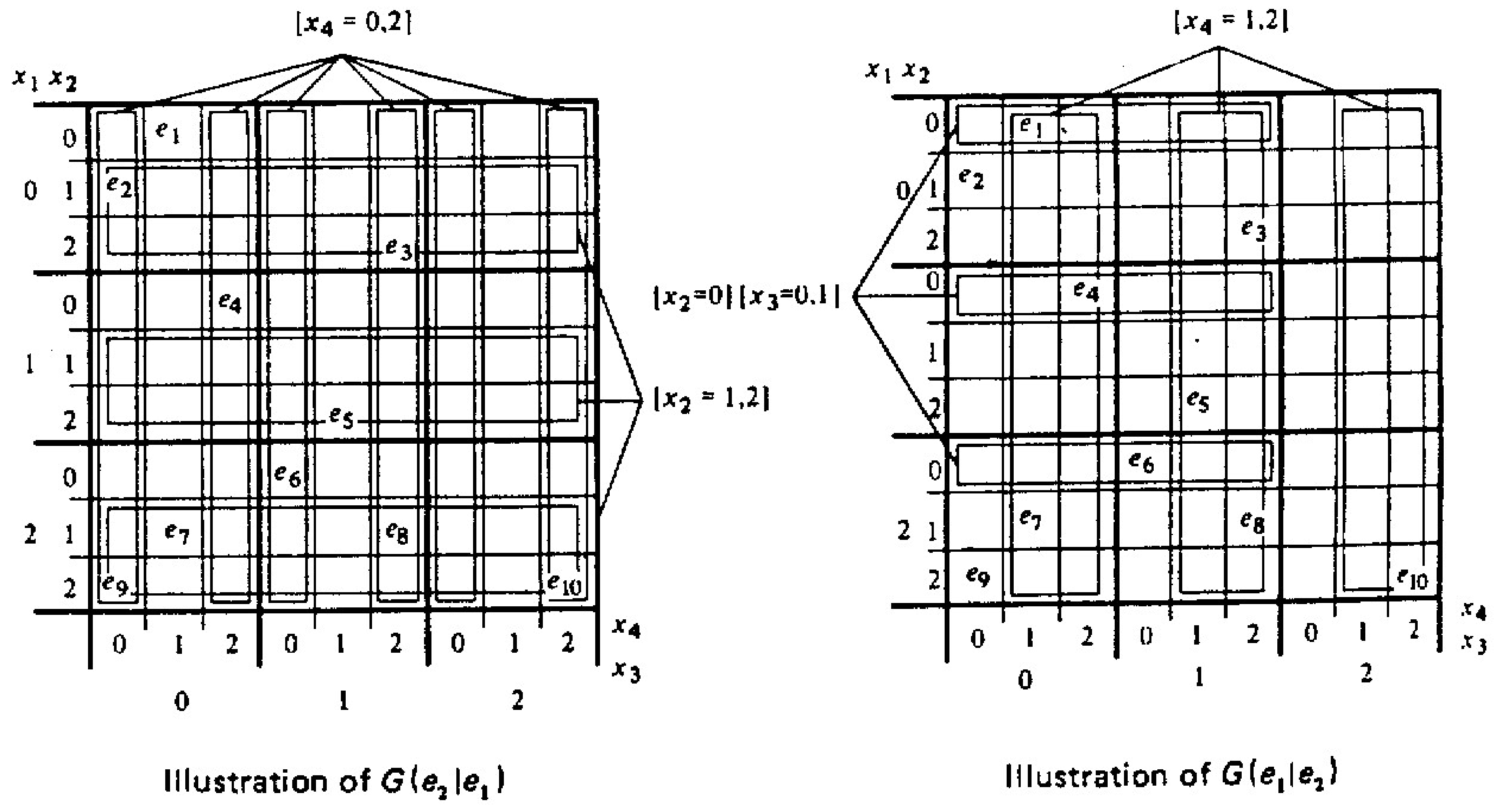


Fig. 3 – A planar representation of the event space spanned over variables x_1, x_2, x_3, x_4 showing the complexes of two stars.

PAF will now be explained as it operates on the data of the sample problem. A flow diagram of PAF is presented in Fig. 4.

Iteration 1:

Step 1 (Fig. 4, block 1):

E_0 is a subset composed of k data events from E (*seeds*). The seeds can be selected arbitrarily, or they can be chosen as events which are most syntactically distant from each other. In the latter case the algorithm will generally converge faster. For selecting such events, program ESEL [11] can be used. For the sample problem, let $E_0 = \{e_1, e_2\}$.

Step 2 (Fig. 4, block 2):

A star, $G(e_j|E_0 \backslash e_j)$, where $e_j \in E_0$, is generated for each seed against the remaining seeds. In our case, stars $G_1 = G(e_1|e_2)$ and $G_2 = G(e_2|e_1)$ are generated. The program produced the following stars by applying the STAR generating procedure outlined in section 4.3. Each star consists of two complexes:

$$G(e_1|e_2) = \{[x_2 = 0]\,[x_3 = 0, 1], [x_4 = 1, 2]\}$$
$$G(e_2|e_1) = \{[x_2 = 1, 2], [x_4 = 0, 2]\}$$

These stars are pictured in Fig. 3.

Step 3 (Fig. 4, block 3):

From each star a complex is selected, such that that resulting set of k complexes

(a) is a disjoint cover of E, and

(b) is an optimal or suboptimal cover among all possible such covers, according to a selected quality of clustering criterion. In the sample problem, the quality of clustering criterion is minimizing total sparseness. There are four combinations of complexes to consider:

			Sparseness
(a)	complex 1:	$[x_2 = 1, 2]$	47
	complex 2:	$[x_2 = 0]\,[x_3 = 0, 1]$	15
			—
			62
(b)	complex 1:	$[x_2 = 1, 2]$	
	complex 2:	$[x_4 = 1, 2]$	
(c)	complex 1:	$[x_4 = 0, 2]$	(These covers are not disjoint. NID (see section 4.3) is applied to each; however in this instance, the resulting sparsenesses are not less than 62.)
	complex 2:	$[x_2 = 0]\,[x_3 = 0, 1]$	
(d)	complex 1:	$[x_4 = 0, 1]$	
	complex 2:	$[x_4 = 1, 2]$	

Combination (a) is selected since it has minimum total sparseness.

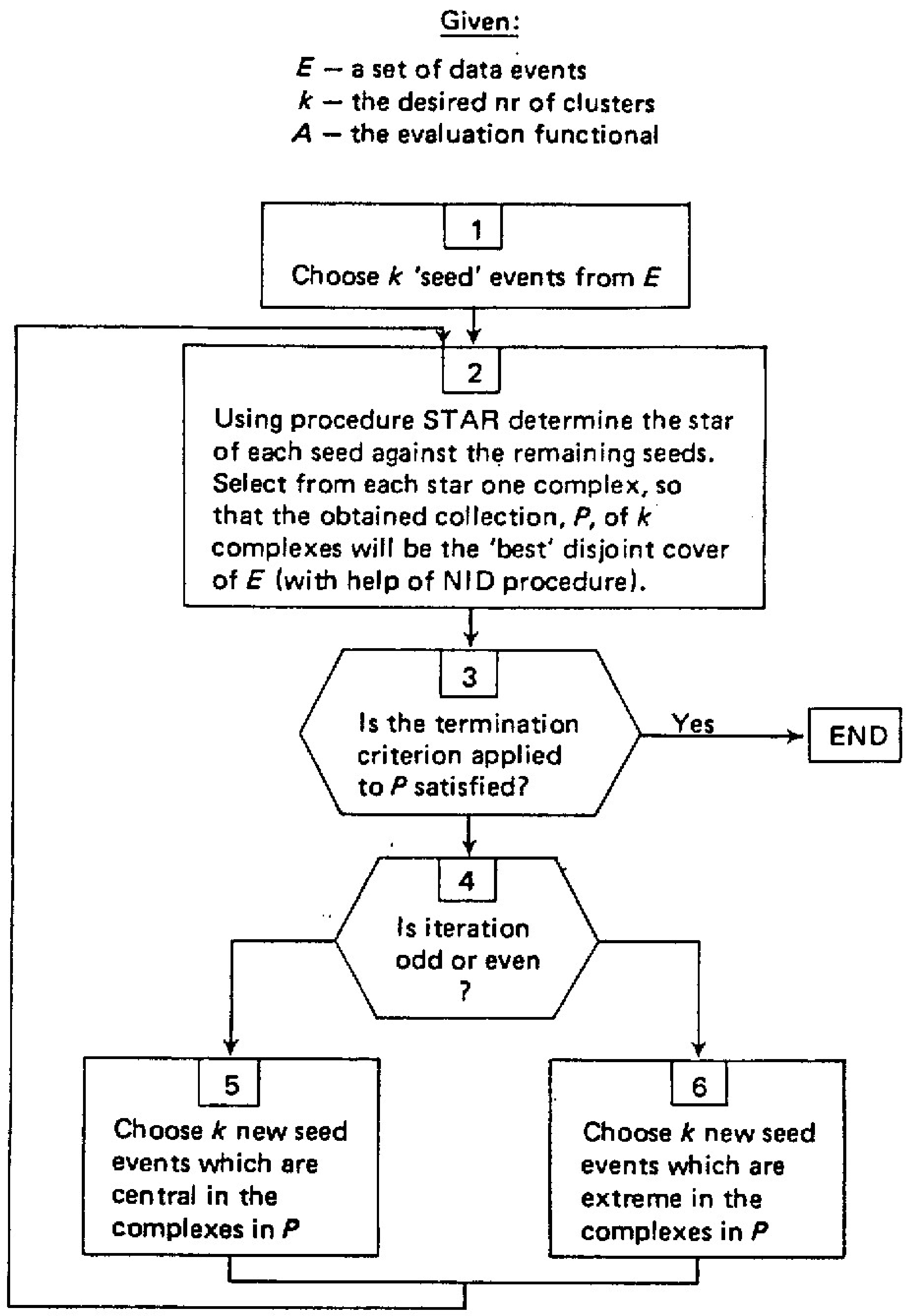

Given:
E – a set of data events
k – the desired nr of clusters
A – the evaluation functional
1
Choose k 'seed' events from E
2
Using procedure STAR determine the star of each seed against the remaining seeds. Select from each star one complex, so that the obtained collection, P, of k complexes will be the 'best' disjoint cover of E (with help of NID procedure).
3
Is the termination criterion applied to P satisfied?
Yes
END
4
Is iteration odd or even ?
5
Choose k new seed events which are central in the complexes in P
6
Choose k new seed events which are extreme in the complexes in P

Fig. 4 – A flow diagram of the inner layer of algorithm PAF.

Step 4 (Fig. 4, block 4):

The termination criterion of the algorithm is applied to the obtained cover. The termination criterion is a pair of parameters (b,p) where b (the *base*) is a standard number of iterations the algortihm always performs, and p (the *probe*) is the number of additional interactions beyond b performed after each iteration which produces an improved cover. In our example, $b = 2$ and $p = 1$.

Step 5 (Fig. 4, blocks 5 and 6):

A new set of seeds is determined. If the iteration is odd, then the new seeds are data events in the centres of complexes in the cover (according to the syntactic distance). If the iteration is even, then the new seeds are data events maximally distant from the centres (according to the 'adversity principle'†). In the sample problem, syntactically central events are to be found for the complex $[x_2 = 1,2]$, covering $\{e_2, e_3, e_5, e_7, e_8, e_9, e_{10}\}$, and complex $[x_2 = 0]$ $[x_3 = 0,1]$, covering $\{e_1, e_4, e_6\}$. Central events are the ones for which the sum of syntactic distances between them and all other events in the same cluster is minimal. The following sum-of-distances tables apply in this example.

Complex $[x_2 = 1,2]$

Event	Sum of distances to other events
e_2	18
e_3	16
e_5	18
e_7	16
e_8	$\underline{15}$
e_9	15
e_{10}	16

Complex $[x_2 = 0][x_3 = 0,1]$

Event	Sum of distances to other events
e_1	5
e_4	$\underline{5}$
e_6	6

Note: Selected events shown with underlined distances. Ties are broken in favour of events not previously used as seeds.

The new seeds are $E_0 = \{e_4, e_8\}$.

Iteration 2:

Step 2:

The stars $G_1 = G(e_4|e_8)$ and $G_2 = G(e_8|e_4)$ are generated:

$$G(e_4|e_8) = \{[x_2 = 0][x_3 = 0,1], [x_1 = 0,1]\ [x_3 = 0,1], [x_3 = 0,2]\}$$
$$G(e_8|e_4) = \{[x_1 = 2], [x_2 = 1,2], [x_3 = 1,2]\}$$

Step 3:

The combinations of complexes which are disjoint covers are:

		Sparseness
(a)	complex 1: $[x_1 = 2]$	22
	complex 2: $[x_1 = 0,1][x_3 = 0,1]$	$\underline{31}$
		53
(b)	complex 1: $[x_2 = 1,2]$	47
	complex 2: $[x_2 = 0][x_3 = 0,1]$	$\underline{15}$
		62

Combination (a) is selected (it has the minimum total sparseness).

† This principle states that if the most outstanding event truly belongs to the given cluster, then when serving as the cluster representation, the 'fit' between it and other events in the same cluster should still be better than the 'fit' between it and events of any other cluster.

Step 4:

This is step 4 of iteration 2. Since $b = 2$, this is the last of the base iterations.

Step 5:

Complex $[x_1 = 2]$ covers $\{e_6, e_7, e_8, e_9, e_{10}\}$ and complex $[x_1 = 0,1][x_3 = 0,1]$ covers $\{e_1, e_2, e_3, e_4, e_5\}$. Since this iteration is an even one, the new seeds are those whose sum of syntactic distances to other events in the same cluster is maximal. After ties are broken, these events are $E_0 = \{e_2, e_{10}\}$.

Iteration 3:

Step 2:

The stars $G_1 = G(e_2|e_{10})$ and $G_2 = G(e_{10}|e_2)$ are generated:

$$G(e_2|e_{10}) = \{[x_1 = 0,1][x_3 = 0,1], [x_2 = 0,1][x_3 = 0,1], [x_3 = 0,1][x_4 = 0,1]\}$$

$$G(e_{10}|e_2) = \{[x_1 = 1,2], [x_2 = 2], [x_4 = 1,2]\}$$

Step 3:

The only combination of complexes which is a disjoint cover is:

		Sparseness
(a)	complex 1: $[x_2 = 2]$	23
	complex 2: $[x_2 = 0,1][x_3 = 0,1]$	30
		53

Step 4:

This iteration is the first (and in this example also the last) 'probe' iteration. If the clustering quality on this iteration is better than the previous best clustering, another p iterations are scheduled, else the algorithm stops after completing p probing iterations. The best total sparseness of iteration 3, namely 53 is not an improvement over the previous best sparseness, also 53. Since $p = 1$ in this example, the termination criterion is satisfied at this point. There are two alternative clusterings produced, each with a total sparseness of 53:

alternative 1:	alternative 2:
$[x_1 = 2]$	$[x_2 = 2]$
$[x_1 = 0,1][x_3 = 0,1]$	$[x_2 = 0,1][x_3 = 0,1]$

4.2 Outer algorithm

The outer layer of the algorithm makes recursive applications of the inner layer in order to create a hierarchical description (a concept tree) underlying the data. The tree grows in a top-down fashion until the 'continuation-of-growth' criterion fails. This criterion requires that the 'fit' (measured by sparseness) of the concepts to the events they describe must be sufficiently better at each next (lower) level of the hierarchy. When this criterion is not met, the latest obtained subcategories become leaves of the tree.

At each step, the group of objects associated with a particular node in the hierarchy is divided into subcategories, and their descriptions are assigned to the offspring nodes. The degree of the parent node (the number of subcategories or clusters) is specified by parameter k. Usually there is no *a priori* knowledge of how many subcategories to form. Interesting solutions from the viewpoint of a human user, however, should involve only a small number of subcategories (e.g., between 2 and 7) so that it is computationally feasible to determine the best number of subcategories by constructing first the best 2-partition, then the best 3-partition, and so on, while evaluating relative quality of the obtained partitions.

The clustering quality measure used here must relate a quantitative measure of complexes (e.g., the total sparseness, S) to the number of clusters. As the number of clusters k increases, the total sparseness will likely decrease (since smaller complexes will better 'fit' the data). One possible global criterion of clustering quality is to minimize $S^*(k+\beta)$, where β is chosen to properly balance the effect of k on the solution.

It should be noted that selecting too small a value of k does not necessarily distort the resulting conceptual hierarchy. It may simply cause the creation of additional levels within the same structure. This is illustrated in Fig. 5.

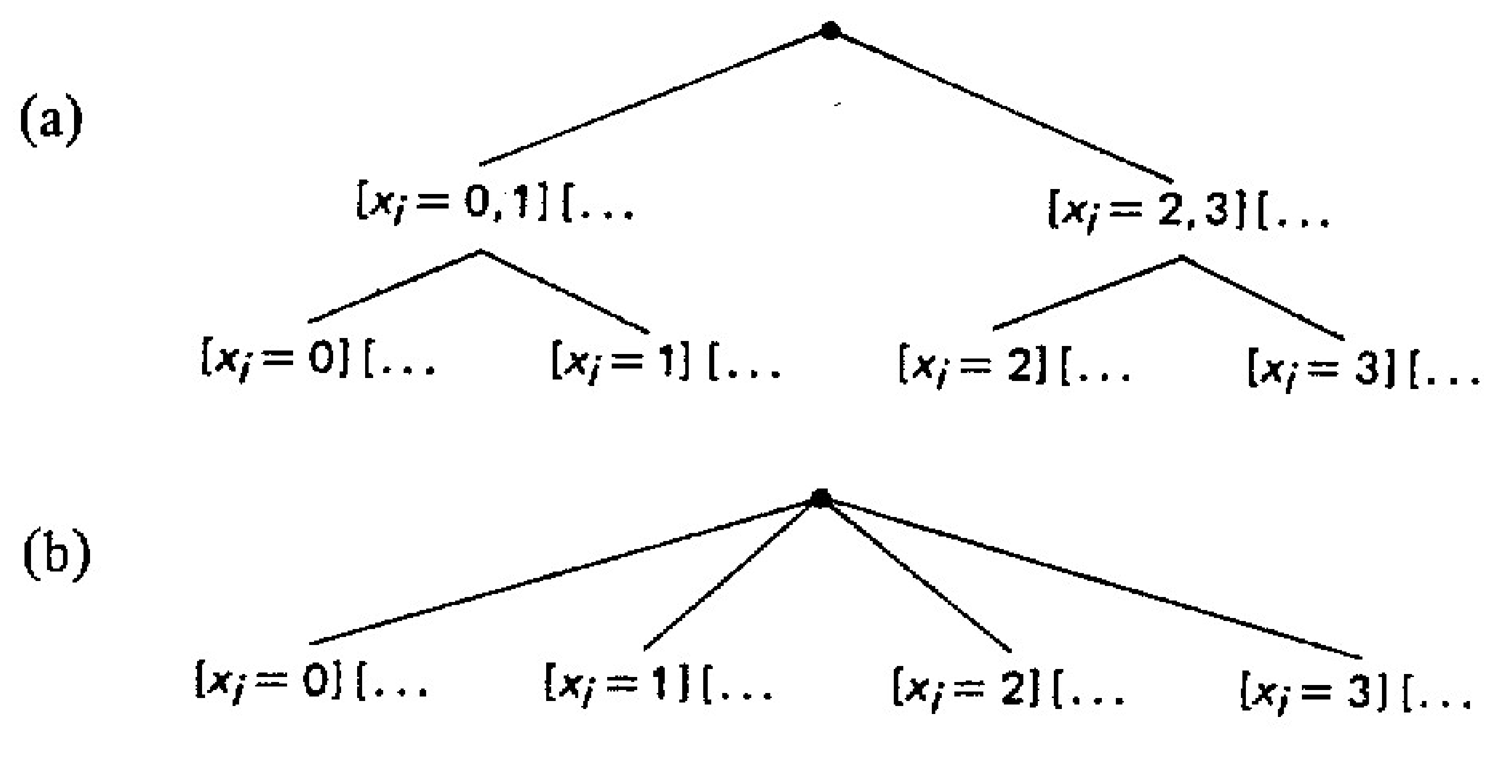


Fig. 5 – Two comparable structures with different degrees of root nodes.

4.3 Procedures STAR and NID, and the method for determining the best cover

The star $G(e|F)$ has been defined as the set of maximal complexes covering event e and no events in the set F. This section will explain how stars are constructed, and are subsequently used to conduct a best-first search to find an optimal or suboptimal k-partition of an event set. Assume first that $F = \{e_1\}$, $e_1 \neq e$. To generate the star $G(e|e_1)$ one determines all variables in e which have different values than in e_1. Suppose, without losing generality, these variables are $x_1, x_2, \ldots, x_k$, and $e_1 = (r_1, r_2, \ldots, r_k, \ldots, r_n)$. Assuming that the variables are nominal. the complexes of the star $G(e|e_1)$ are $[x_i \neq r_i]$, $i = 1, 2, \ldots, k$ (or equivalently, $[x_i = D_i \backslash r_i]$), since these are the largest complexes which cover e

and do not cover e_1. The number of complexes in a star $G(e|F)$, when F is a single event, is at most n (the number of variables), and at least 1, since $e_1 \neq e$.

Assume now that $F = \{e_1, e_2, \ldots, e_s\}$. A star $G(e|F)$ is constructed by building first stars $G(e|e_1)$, $i = 1, 2, \ldots, s$, and then set-theoretically multiplying these stars by each other, using absorption laws to eliminate redundancy. A detailed description of this process, including the treatment of linear variables, is given in [12].

The upper bound on the size of a star is n^m, where m is the number of events in F. Absorption laws will usually eliminate many redundant complexes, but the size of a star may still become unmanageable. Therefore a *bounded star* is used which has a specified upper limit, MAXSTAR, on the number of complexes it may contain. Whenever a star exceeds this number, the complexes are ordered in ascending order according to sparseness (or in general, to any assumed criterion) and only the first MAXSTAR complexes are retained. The position of a complex in the sequence so ordered is the *rank* of the complex.

At each iteration of algorithm PAF (Fig. 4), k stars are produced, for each seed event against the remaining $k-1$ seed events. From each star one complex is then selected in such a way that the resulting set will consist of k disjoint complexes (a k-partition), and be optimal according to the assumed criterion. If un-bounded stars were used, each could be composed of up to $n^{(k-1)}$ complexes and therefore up to $(n^{(k-1)})^k$ sets of complexes would have to be inspected in order to determine the optimal k-partition! The best-first search strategy outlined below solves this problem.

Performing an efficient search for the best set of complexes can be viewed as finding an optimal path through the search tree whose nodes are complexes. The ith level of the tree corresponds to the ith star. The height of the tree is k, and a path of length k corresponds to a particular k-partition. The *pathrank* of a path is the sum of the ranks of the complexes along the path. The sequences of complexes (paths) are considered in order of increasing pathrank so that complexes with minimum sparseness are considered first.

Frequently the cover with minimum total sparseness will not be disjoint. A procedure called NID tries to transform a *n*on-disjoint cover *i*nto a *d*isjoint one by making small adjustments in cluster memberships. The NID procedure is not always successful, and when it cannot create a disjoint cover of all events in E, it creats the disjoint cover which covers as many events in E as possible. The events it is unable to cover are reported as 'exceptional events'.

4.4 A note on the CLUSTER/paf implementation of the algorithm

The algorithm described above has been implemented in a program CLUSTER/paf, written in PASCAL. The program contains about 3500 PASCAL statements and requires about 20K 60-bit words of memory for clustering problems of the kind shown in Fig. 2. CLUSTER/paf is connected to a relational data base, and all the input examples and parameters are supplied to it in the form of relational tables.

Data points and complexes are stored in one common data structure, the COMPLEX[†], which is an array of SELECTORS declared as objects of the PASCAL type 'set of interger'. The integer-coded values in the reference set of each variable, x_i, are stored in SELECTOR$_i$. When a particular variable, x_j, is not present in a complex, SELECTOR$_j$ is assigned the set representing the entire domain of the variable. Each COMPLEX is thus of fixed length even though the number of selectors in the corresponding complex varies. This data structure provides an easy way to compute the total number of events in a complex $t(\alpha)$, the number of data points covered by a complex $p(\alpha)$, and from these two, the sparseness of the complex $s(\alpha)$.

The program supports five criteria for evaluating the quality of a k-partition:

- the total sparseness of the complexes,
- the degree of inter-cluster disjointness (the total number of disjoint selectors in the k-partition),
- the imbalance in cluster populations (the uneveness of the distribution of events in complexes of the k-partition),
- the dimensionality (the number of different variables used in the complexes of the k-partition),
- the total number of selectors in the k-partition.

Selected criteria from the above list are applied in a lexicographical order defined by the user.

5. AN EXAMPLE

The simple example problem described below was designed to illustrate the major differences between traditional clustering techniques and the conceptual

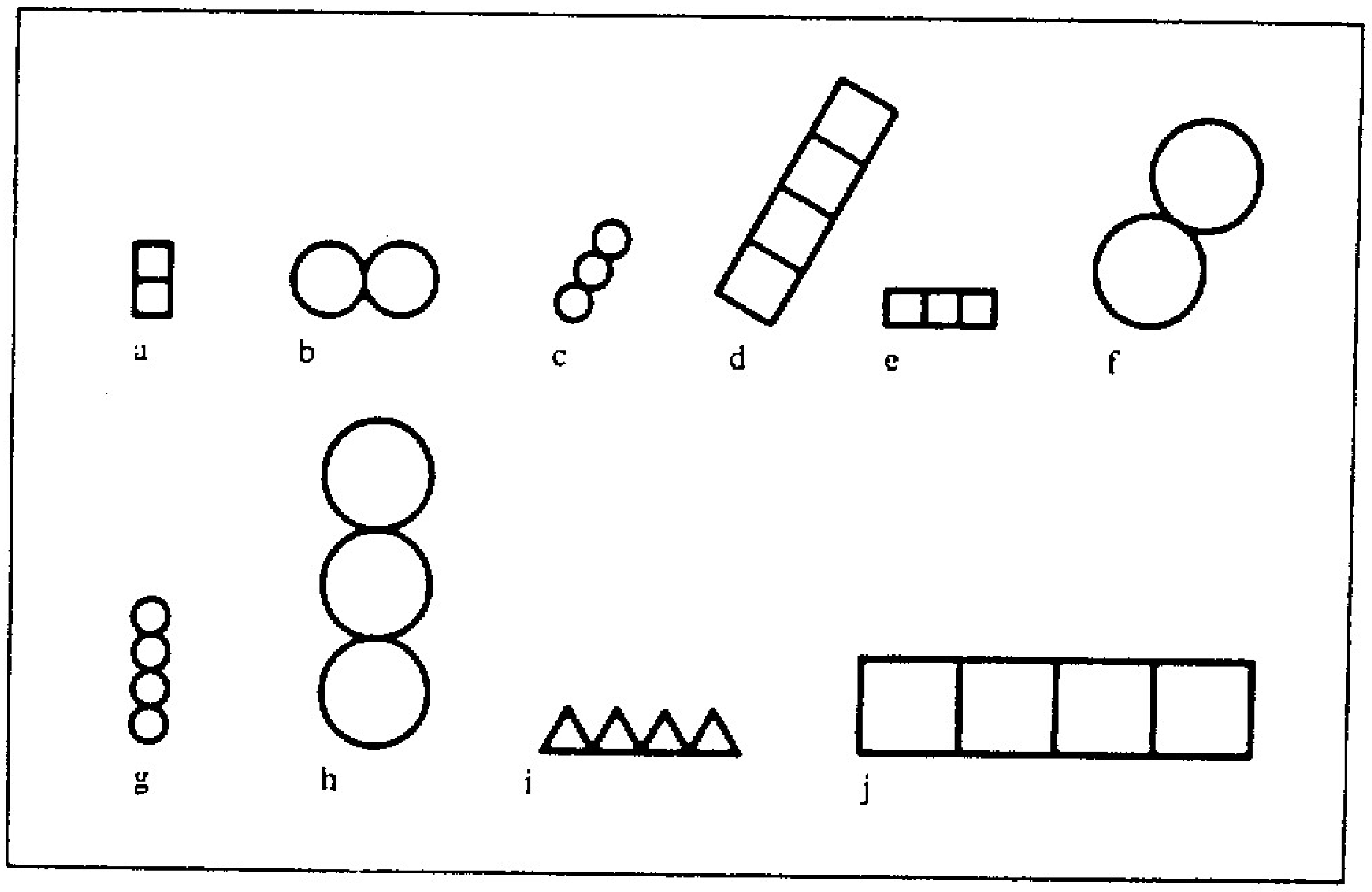


Fig. 6 – BEAD STRINGS.

† Upper-case will be used to denote data structures within CLUSTER/paf.

clustering method described in the paper. Results from applying the traditional numerical taxonomy techniques to the problem were obtained by using a program NUMTAX[†], which implements the standard algorithms from Sokal & Sneath [14]. Results from applying the conjunctive conceptual method were obtained by using the program CLUSTER/paf.

The problem is to structure hierarchically the ten objects shown in Fig. 6. Each object is described by values of the four variables: size of beads, number of beads, shape of beads, and orientation of beads. Figure 7 shows the table of descriptions of the beads. The symbolic values of variables in Fig. 7 were encoded as integer values to prepare the data for NUMTAX and CLUSTER/paf. The encoding was:

x_1:	size of beads	x_3:	shape of beads
	0 – large		0 – circle
	1 – medium		1 – square
	2 – small		2 – triangle
x_2:	number of beads	x_4:	alignment of beads
	0 – 2 beads		0 – vertical
	1 – 3 beads		1 – diagonal
	2 – 4 beads		2 – horizontal

The encoded values for this problem were given in Fig. 2, and used to illustrate the inner PAF algorithm in section 4.

string symbol	size of beads	number of beads	shape of beads	orientation of beads
a	small	2	square	vertical
b	medium	2	circle	horizontal
c	small	3	circle	diagonal
d	medium	4	square	diagonal
e	small	3	square	horizontal
f	large	2	circle	diagonal
g	small	4	circle	vertical
h	large	3	circle	vertical
i	small	4	triangle	horizontal
j	large	4	square	horizontal

Fig. 7 – The description of BEAD STRINGS.

5.1 Results from NUMTAX

Given a set of events, the numerical taxonomy program, NUMTAX, organizes the events into a hierarchy which reflects the numerical distances between consecutively larger subcategories of events, which at the highest level, merge to

† NUMTAX was written by R. Selander of the University of Illinois.

become the union of all events (a dendrogram). Figure 8 shows the dendrogram generated by NUMTAX using the reciprocal of unweighted Euclidean distance as the measure of sisimilarlity. (The distances were calculated from standardized data (*z*-scores)). Eighteen different distance measurements were tried in the experiment by using various combinations of a data transformation (raw data, ranging, *z*-scores); a similarity measure (reciprocal Euclidean distance, product-moment correlation, simple matching coefficient); and weighting (unweighted, weighted by the number of events in a group).

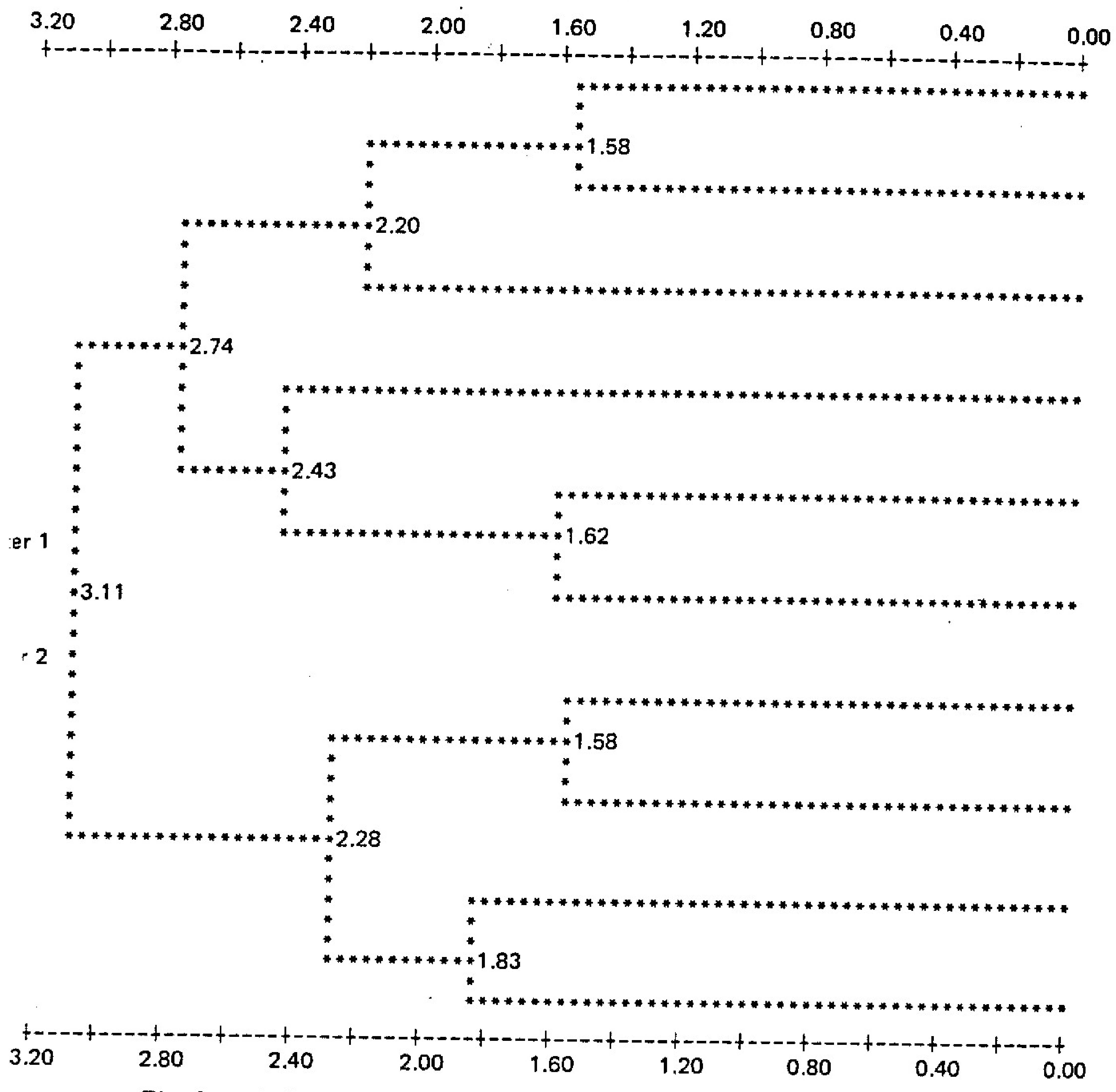
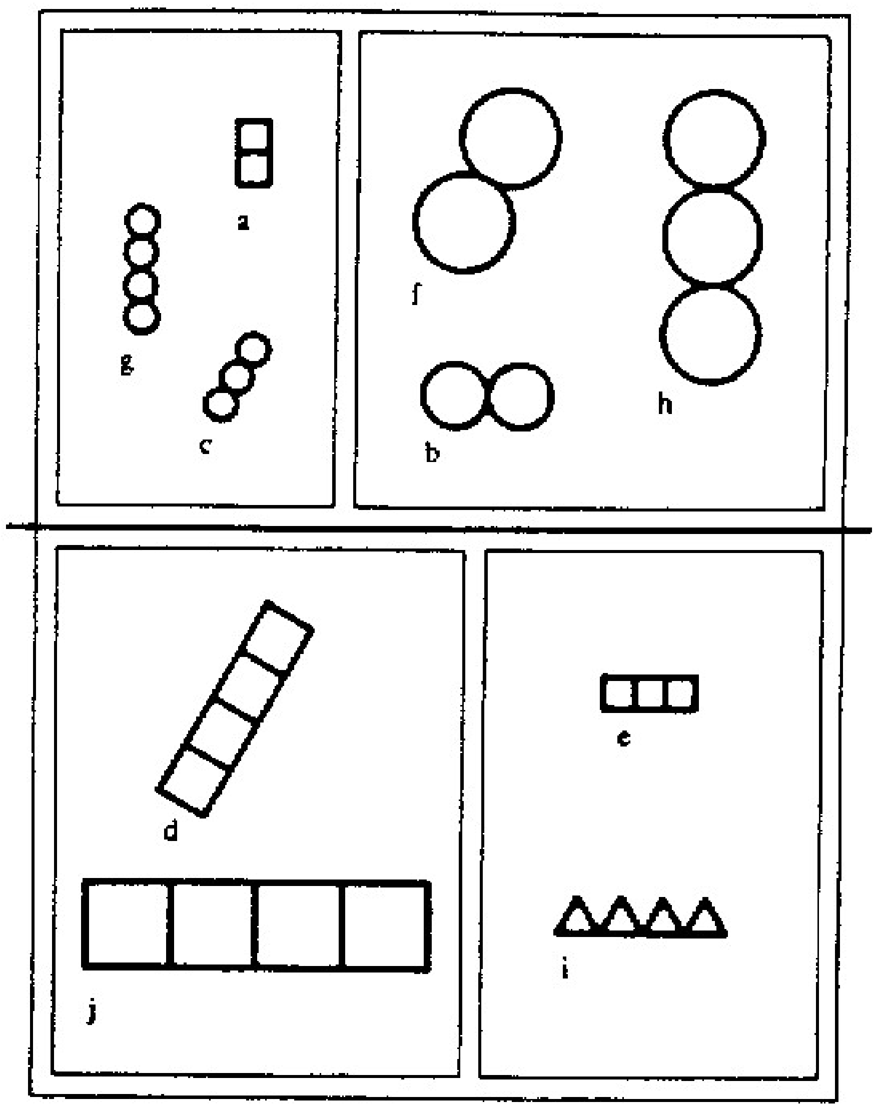


Fig. 8 – A dendrogram of BEAD STRINGS produced by NUMTAX. Coefficients assigned to the nodes represent the Euclidean distance (in the *n*-dimensional space) between the objects indicated by branches from the node. The distances are calculated using standardized data values, i.e., the values of a variable are divided by their standard deviation. The distance between groups of objects is defined as the average of the distances from objects in one group to those of the other group.

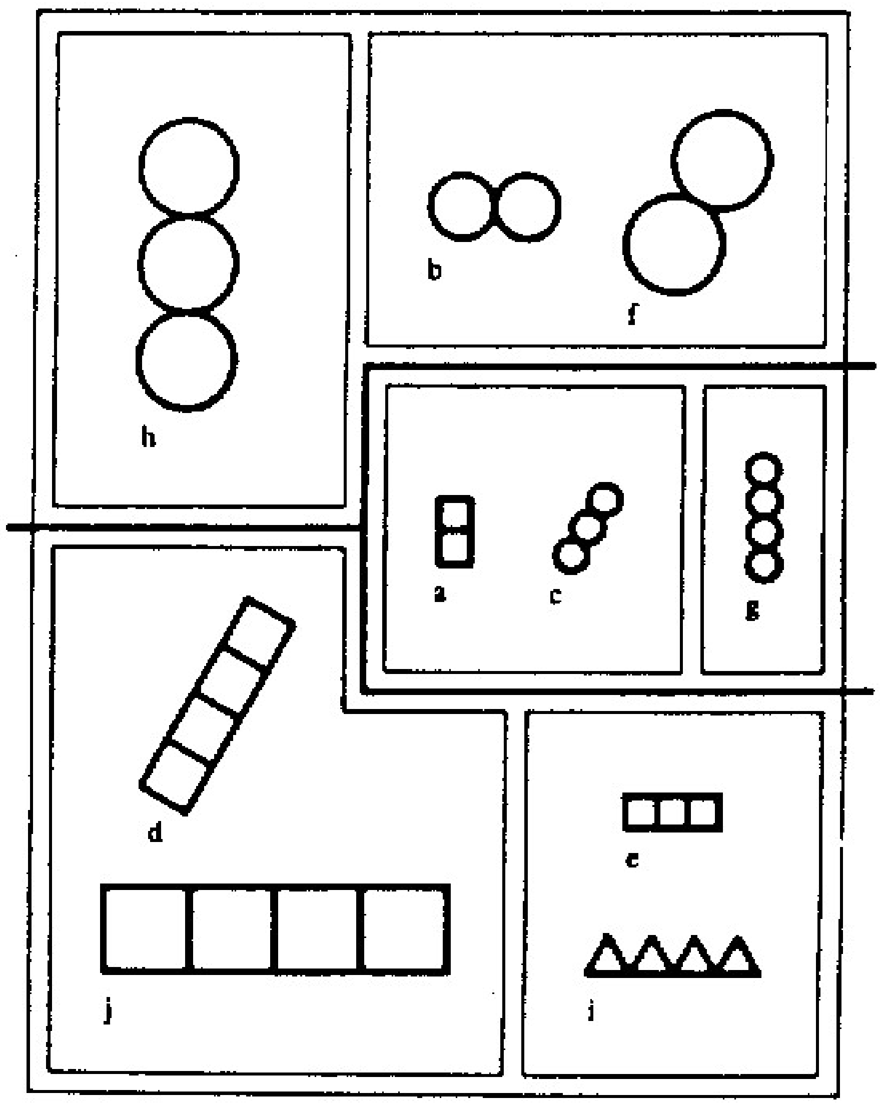
Because dendrograms are constructed bottom-up, the entire dendrogram must always be generated. The after this has been done, the dendrogram may be cut apart at some level to produce clusters. The collections of BEADS shown in Fig. 9 are the result of considering only the two major subtrees of the dendrogram of Fig. 8. The three clusters shown in Fig. 10 were obtained in a similar manner



1. [Bead shape = circle] or [Orientation = vertical

2. [Bead shape ≠ circle] & [Orientation ≠ vertical]

Fig. 9 – Clusters obtained from the dendrogram generated by NUMTAX (K = 2). Description of the clusters were obtained by program AQ11.



1. [Bead size ≠ small] & [Bead shape ≠ square]

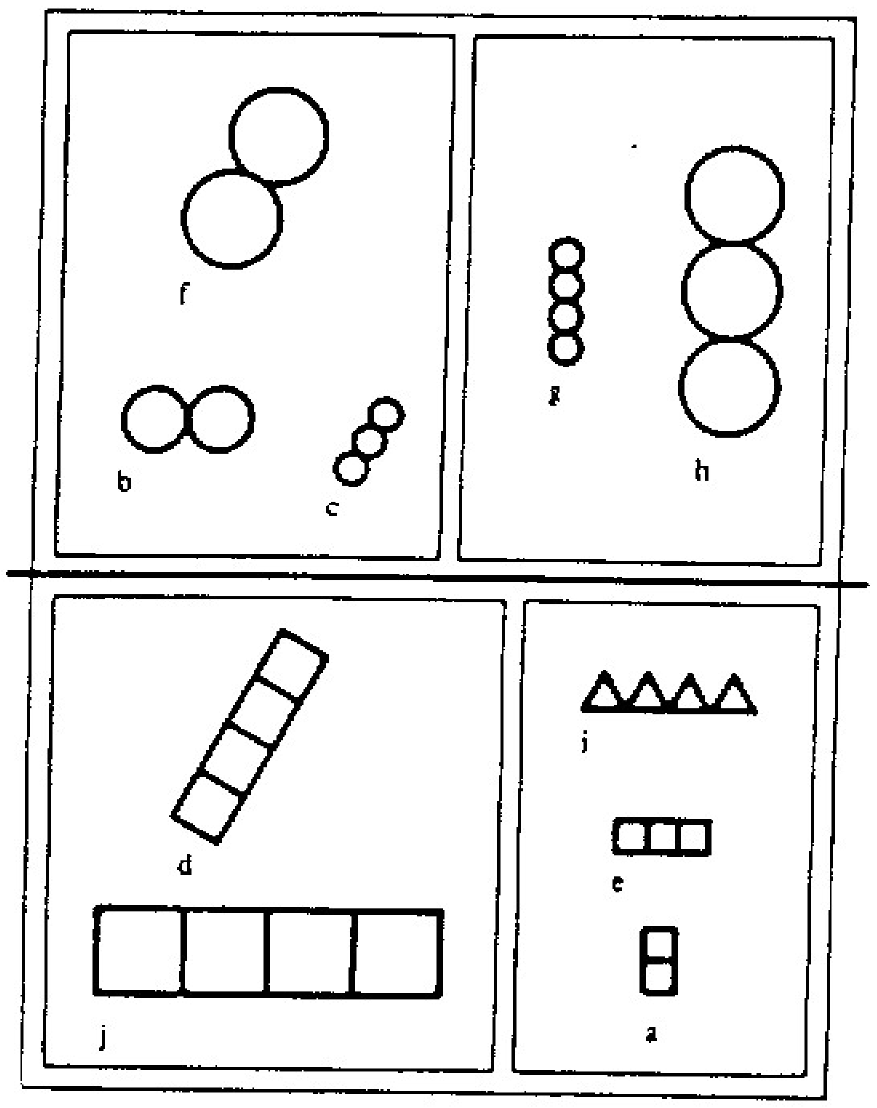
2. [Bead size = small] & [Orientation ≠ horizonta

3. [Bead size ≠ small] & [Bead shape = square] o
[Orientation = horizontal]
([Bead shape = square] or [Bead size = small])

Fig. 10 – Clusters obtained from the dendrogram generated by NUMTAX (K = 3). Descriptions of the clusters were obtained by program AQ11. In both figures, the inner lines show possible further partitions.

by cutting the dendrogram into three portions. The two illustrations are typical of the many dendrograms generated. None of the dendrograms form clusters which have simpler characterizations.

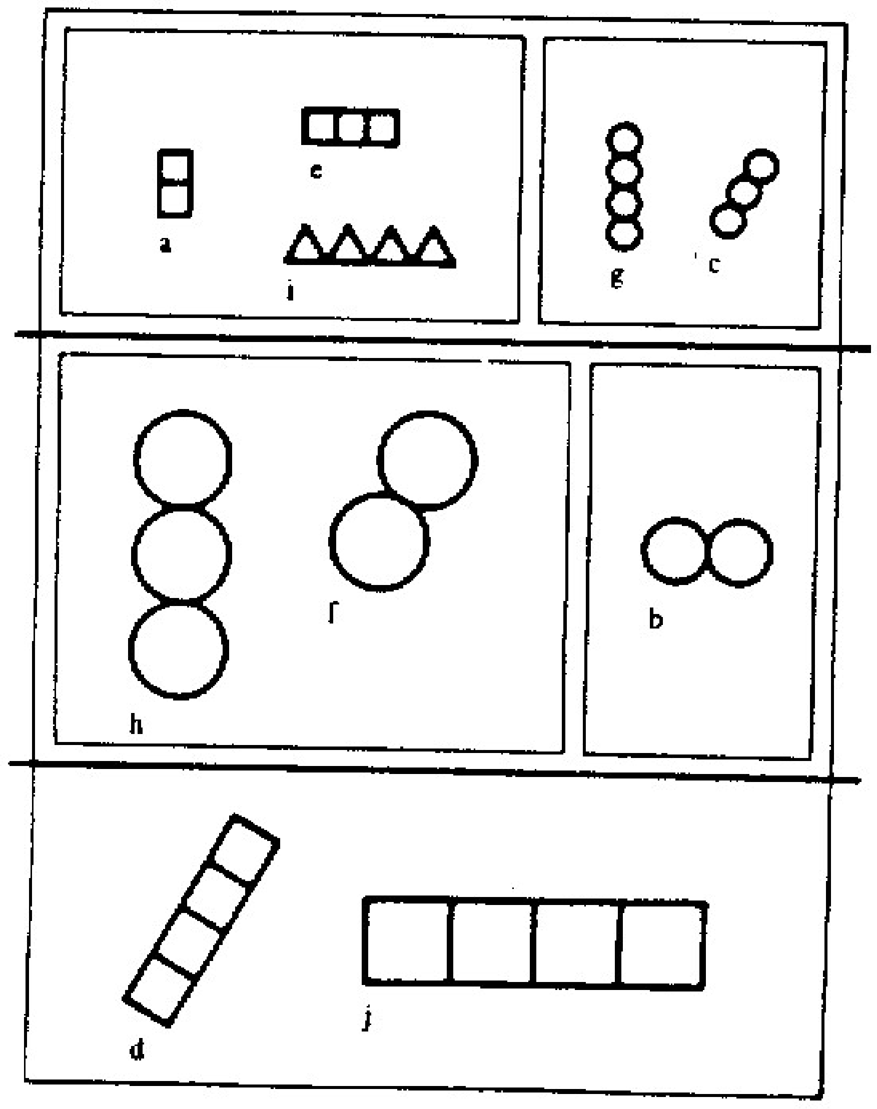
The clusters obtained from the dendrogram are not accompanied by any description. In order to determine the meaning of the clusters, an additional step which generates cluster characterizations is incorporated. The program used to provide these characterizations in this experiment was AQ11 [11]. This program accepts sets of events (the clusters defined by dendrogram subtrees) and produces maximally generalized descriptions of them in the form of a logical disjunction of complexes.



1. [Bead shape = circle]

2. [Beach shape ≠ circle]

Fig. 11 – Clusters generated by CLUSTER/paf for K = 2.



1. [Bead size = small]

2. [Bead size ≠ small] & [#Beads < 3] & [Bead shape = circle]

3. [Bead size ≠ small] & [#Beads = 4] & [Bead shape = square]

Fig. 12 – Clusters generated by CLUSTER/paf for K = 3.

5.2 Results from conceptual clustering

The conceptual clustering program CLUSTER/paf was run with two different criteria of clustering optimality:

(a) 'minimize sparseness, then minimize the number of selectors', and
(b) 'minimize the degree of disjointness, then minimize sparseness'.

Results when using criterion (a) are shown in Figs. 11 and 12. Figures 13 and 14 show results when using criterion (b).

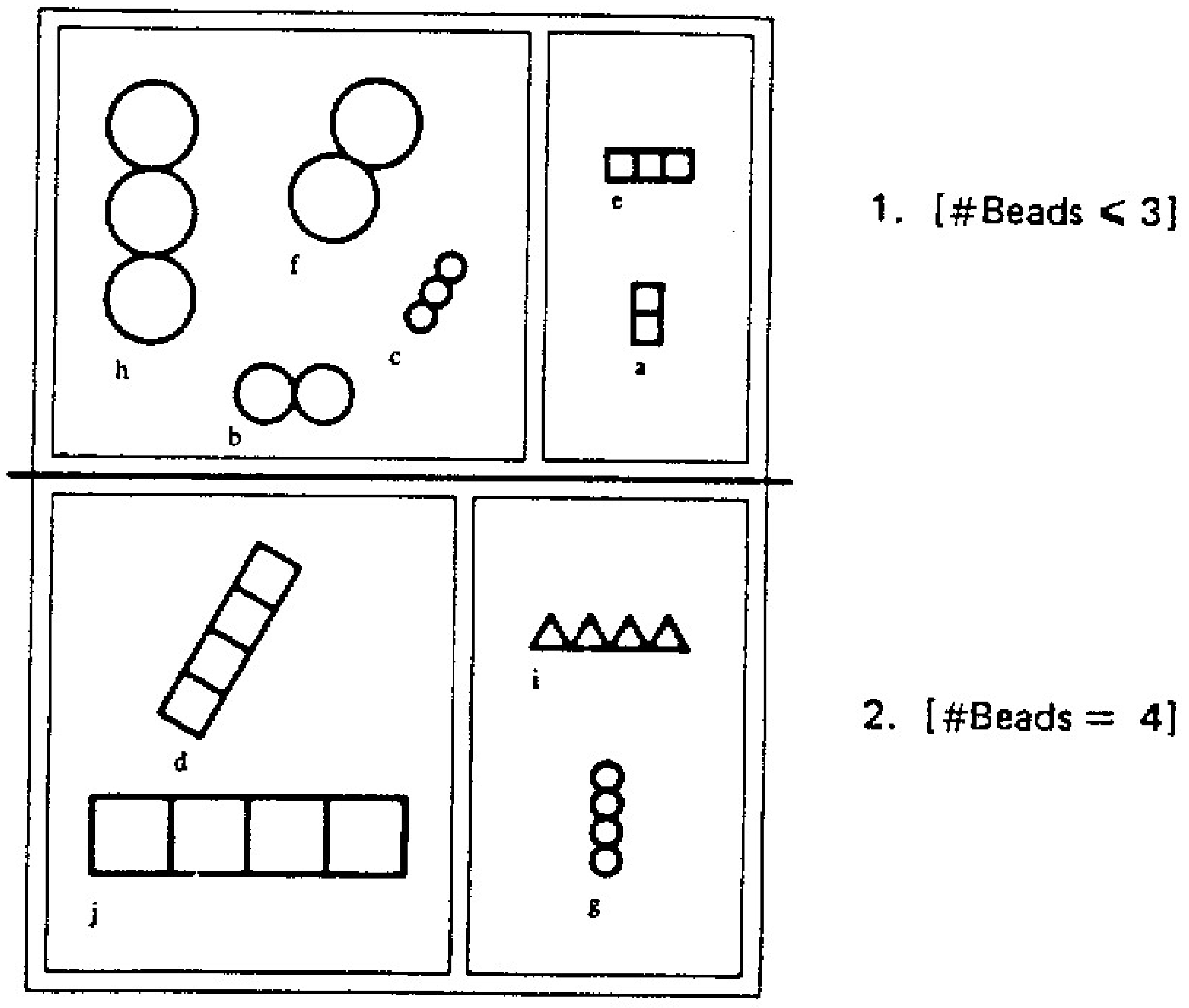


Fig. 13 – Another solution generated by CLUSTER/paf for K = 2.

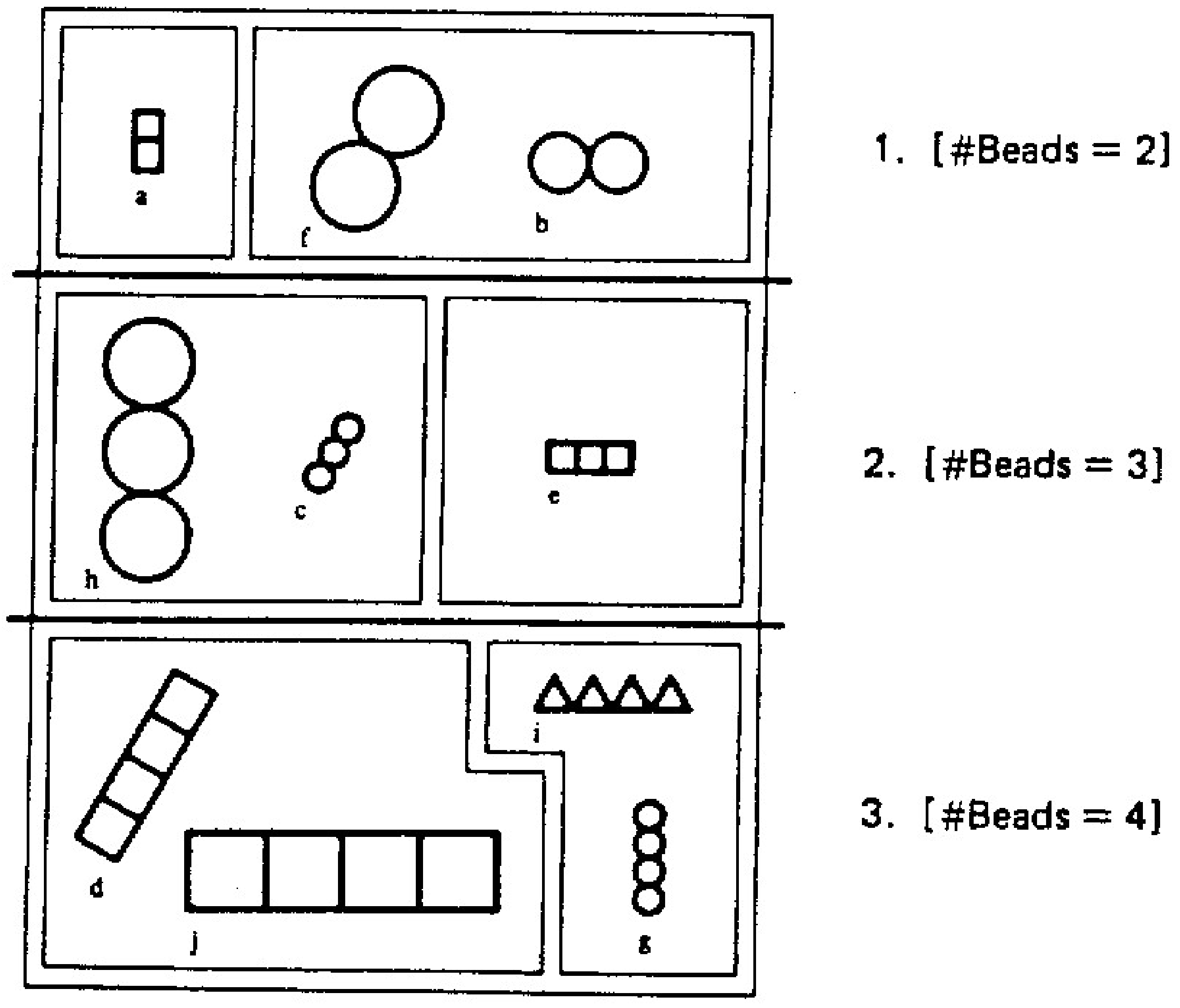


Fig. 14 – Another solution generated by CLUSTER/paf for K = 3.

5.3 Discussion of results

An experiment with human subjects solving this problem indicated that people categorize objects using the object's most noticeable properties. Typical solutions were, e.g.,

[shape of BEAD = circle] *vs.* [shape of BEAD ≠ circle] ,

and

[#BEADS = 2] *vs.* [#BEADS = 3] *vs.* [#BEADS = 4] .

The clusterings produced by NUMTAX seem 'unnatural' when compared to human clustering. The distinction can also be observed by looking at the cluster descriptions. The descriptions determined by program AQ11 for the clusters generated by NUMTAX involve disjunction, and seem to be more complex than the descriptions which people consider most natural. On the other hand, the descriptions produced by CLUSTER/paf matched the human solutions.

It should be noted that the descriptions generated by AQ11 for clusters obtained from NUMTAX were 'biased' towards the type of descriptions were used. In any case, however, since NUMTAX is not equipped with knowledge of any concepts, it cannot knowingly produce clusters corresponding to concepts.

CLUSTER/paf has been applied to some 'real world' clustering problems. One application was to cluster data describing 47 diseased soybean plants (each described by 34 many-valued variables). CLUSTER/paf accurately partitioned the diseased plants into the four disease categories presented in the sample, and described the clusters by the proper concepts, stated in terms of disease symptoms confirmed by plant pathologists.

6. CONCLUSION

he conceptual clustering method, PAF, determines a hierarchy of subcategories ithin a collection of objects. The subcategories are determin[illegible] in such a way iat an appropriate generalization of the description of each subcategory yields a ingle conjunctive statement which is disjoint from the similar statements haracterizing other subcategories. The difference between this method and traditional methods can be explained by extending the concept of the measure of similarity into the more general notion of conceptual cohesiveness.

A limitation of the program is that it describes subcategories of objects solely y conjunctive statements. Although a conjunctive statement is probably the nost common descriptive form used by humans, it is a quite limited form. An nteresting extension of the work would be to use other forms of descriptions, .g., involving logical implication, equivalence, and exception. Also, the program equires thttthat all potentially relevant variables for clustering are supplied in he input data (as in all other clustering methods).

The problems which are suitable to the PAF algorithm involve objects lescribable by variable-value pairs, i.e., objects without any structure of their wn. When the objects of interest do have structure which has to be taken into consideration (e.g., relationships between features of object subparts), the techniques presented here become inadequate. Clustering of such objects will require the use of a richer description language than that used here, such as first-order predicate logic or its equivalent.

7. ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under grant No. MCS-79-06614. The authors wish to thank Professor R. Selander of the University of Illinois for providing the NUMTAX numerical taxonomy program.

REFERENCES

[1] Anderberg, M. R., (1973). *Cluster Analysis for Applications.* New York and London: Academic Press.

[2] Diday, E., & Simon, J. C., (1976). Clustering analysis, *Communication and Cybernetics, 10*, (ed. Fu, K. S.). Berlin, Heidelberg, New York: Springer Verlag.

[3] Diday, E., (1978). Problems of clustering and recent advances, 11th Congress of Statistics, Oslo, Norway.

[4] Gowda, K. Chidananda, & Krishna, G., (1978). Disaggregative clustering using the concept of mutal nearest neighbourhood, *IEE Trans. On Systems, Man and Cybernetics, SMC-8, No. 12*, 888–894.

[5] Hanani, U., (1979). Multicriteria dynamic clustering. Rocquencourt: INRIA Reports.

[6] Michalski, R. S., (1974). Variable-valued logic: System VL_1, *Proceedings of the 1974 Int. Symp. on Multiple-Valued Logic*, West Virginia University, Morgantown, West Virginia, May 29–31.

[7] Michalski, R. S., (1975). Synthesis of optimal and quasi-optimal variable-valued logic formulas, *Proceedings of the 1975 Int. Symp. on Multiple-Valued Logic*, Bloomington, Indiana, May 13–16.

[8] Michalski, R. S., (1975). Variable-valued logic and its applications to pattern recognition and machine learning, *Multiple-Valued Logic and Computer Science*, (ed. D. Rine). Amsterdam: North-Holland.

[9] Michalski, R. S., (to appear). Studies in inductive inference and plausible reasoning, Technical Report, Urbana-Champaign: Department of Computer Science, University of Illinois.

[10] Michalski, R. S., (1978). A planar geometrical model for representing multidimensional discrete spaces and multiple-valued logic functions, *UIUCDCS-R-897*, Urbana-Champaign: Department of Computer Science, University of Illinois.

[11] Michalski, R. S., & Larson, J. B., (1978). Selection of most representative training examples and incremental generation of VL_1 hypotheses: the underlying methodology and the description of programs ESEL and AQ11, *UIUCDCS-R-867*, Urbana-Champaign: Department of Computer Science, University of Illinois.

[12] Michalski, R. S., (1980). Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts, (Special issue on knowledge acquisition and induction), *Policy Analysis and Information Systems, No. 3*, 219–244.

[13] Nilsson, N. J., (1980). *Principles of Artificial Intelligence*, Menlo Park: Tioga Publishing Company.

[14] Sokal, R. R., & Sneath, P. H., (1963). *Principles of Numerical Taxonomy.* San Francisco: Freeman.

[15] Stepp, R., (1979). Learning without negative examples via variable-valued logic characterizations: the uniclass inductive program AQ7UNI, *UIUCDCS-R-982*. Urbana-Champaign: Department of Computer Science, University of Illinois.

[16] Stepp, R., (1980). Learning by observation: experiments in conceptual clustering, Workshop on Machine Learning, Carnegie-Mellon University, Pittsburgh, Pennsylvania, July 16–19.

[17] Stepp, R., (to appear). A description and user's guide for CLUSTER/paf – a program for conjunctive conceptual clustering. Technical Report, Urbana-Champaign: Department of Computer Science, University of Illinois, Urbana, Illinois.

[18] Watanabe, S., (1968). Pattern recognition as an inductive process, *Methodologies of Pattern Recognition*, (ed. Watanabe, S.). New York and London: Academic Press.

[19] Watanabe, S., (1969). *Knowing and Guessing; a quantitative study of inference and information*. New York: Wiley.

[20] Winston, P. H., (1977). *Artificial Intelligence*. Reading, Mass: Addison-Wesley.