An improved algorithm for the Jardine–Sibson method of generating overlapping clusters

A. J. Cole and D. Wishart

Computing Laboratory, University of St. Andrews, St. Andrews, Fife

An improved algorithm for generating k-partition overlapping clusters is discussed. Jardine and Sibson (1968) proposed the basic algorithm and suggested that it was likely that an improved algorithm could be found. In addition an attempt is made to automate the subsequent process of cluster recognition.

(Received July 1969)

Suppose that for a population P, comprising N individuals, a similarity matrix D is computed using a suitable coefficient of association, and a linkage parameter k and similarity threshold h are chosen. Each individual may be represented by a vertex on a graph, and all pairs of vertices which correspond to pairs of individuals having a similarity of at least h are connected. All maximal complete subgraphs are found and all pairs of such subgraphs that intersect in at least k vertices are further connected. This leads to the concept of a chain of connected subgraphs. In particular, since a sufficiently dense set of points in Euclidean space could form such a chain cluster having any shape or variance, the method when applied to metric data is of the 'natural class' type (Wishart, 1968). The method induces overlapping clusters since two intersecting maximal complete subgraphs which have less than k overlap vertices are distinguished as separate clusters. When $k = 1$ no overlap occurs and the procedure is identical to single linkage.

Method I below is a summary of the method proposed by Jardine and Sibson for deriving a numerical representation of this 'k-partition' clustering from the similarity matrix D. The modified matrix U that results from these operations on D expresses the connections completed within all the chain clusters. Method II describes our alternative method. A method of constructing the clusters from U is discussed later.

Method I:

1. In one 'scan' of D, all possible subsets of P comprising exactly $k + 2$ objects are considered.

2. For each subset determine the least and second least similarities h' and h'' respectively. If $h' < h''$ replace h' by h'' in D. If $h' = h''$ leave D unchanged.

3. Step 2 is repeated on the reduced similarity matrix until every subset contains a non-unique minimum similarity (if dissimilarities are used, the expressions 'least', $h' < h''$ and minimum are replaced by 'greatest', $h' > h''$ and maximum respectively: see, for example, Jardine and Sibson, 1968). This condition is detected when no further transformations of the reduced matrix occur during one complete scan. The reduced matrix is then the modified matrix U.

Similar algorithms are used by Johnson (1967) and Roux (1968) for the ultrametric transformation in the special case $k = 1$.

Method I evidently requires a considerable amount of unproductive work. During the second and subsequent scans of D, the examination of the majority of the $\binom{N}{k+2}$ subsets of P will yield no additional transformation, and the last scan of D is completely unproductive since the matrix is merely checked for the lack of any further modification. Also, the work required for the algorithm is proportional to

$$W_I = \frac{p}{2}(k+2)(k+1) \times \frac{N!}{(N-k-2)!(k+2)!}$$

where p is the number of scans required. Clearly, when $N \gg k$

$$W_I \approx \frac{p}{2k!} N^{(k+2)}$$

and the population will be restricted by most computers to $N \leqslant 20$.

One approach to the reduction of W_I is to eliminate as much of the unproductive work as possible. Williams, Lambert and Lance (1966) define a single linkage algorithm which uses an ordering of the similarities in D, tagged by their corresponding object pairs. The method works progressively through this ordered similarity list deriving the single linkage hierarchic dendrogram in the process, and since the similarities associated with each fusion on the dendrogram correspond precisely to the ultrametric distances when $k = 1$, the method incidentally develops the elements of U. We choose therefore to adapt this approach to the construction of U for all k, and propose the following basic algorithm together with its subsequent improvements as a means of inserting similarities directly from the ordered similarity list into U.

Method II:

1. Tag the elements of the similarity matrix D and order them by descending similarity in an array Q

Initialise an empty matrix U which is to receive the reduced matrix associated with D.

2. Sequentially remove similarities from Q. If we are considering the pth similarity Q_p corresponding to the object pair (i, j) and if U_{ij} has been filled, then we ignore Q_p and proceed to Q_{p+1}. If U_{ij} is not filled, then we set $R = Q_p$, put the object pair (i, j) at the head of a new 'insertion list' L and then proceed to 3.

3. From the insertion list L, select object pairs (g, h) corresponding to entries that are to be made in U. Initially L contains one pair only, but the procedure below may cause additional object pairs to be added. If U_{gh} is filled, then, we consider the next pair in L. If U_{gh} is not filled, we set $U_{gh} = R$ and proceed to 4.

4. Now consider all $(k + 2)$-element subsets of P containing both g and h. Associated with each subset $S = (\alpha_1, \alpha_2, \ldots, \alpha_k, g, h)$ we have the set η of similarities $\{U_{ij};\ i, j \epsilon S\}$, which can be partitioned into two subsets η', η'' where η' contains the elements U_{ij} of η which are not filled in U, and η'' is the complement of η' in η containing those elements U_{ij} which have been filled in U. We observe that the least similarity in η'' is R and no element of η' is greater than R by virtue of the ordering in Q. If, for some subset S, η' is a single unfilled element d' corresponding to $U_{g'h'}$, then $d' \leqslant R$. Hence, either d' is a unique minimum in which case Method I requires that we replace $U_{g'h'}$, by R, or else $d' = R$. We therefore set $U_{g'h'} = R$ and add the pair (g', h') to L.

5. When all subsets containing objects g and h have been considered, we return to 3 and select a new pair from the insertion list L.

6. When L is finally exhausted, we check U for unfilled elements. If U is completely filled then we exit; otherwise we return to 2 and consider new similarities from Q for insertion.

A comparison between Methods I and II

We shall omit the rather tedious formal proof that Method II derives the correct reduced matrix U. However, it is not immediately clear that the final matrix U obtained by Method II cannot be further modified by a Method I-type scan. We shall call steps 3–6 of Method II one 'cycle' of the algorithm.

Let ρ be the set of all subsets of P containing $(k + 2)$ objects. Partition ρ into ρ_i' and ρ_i'', where ρ_i'' contains all the subsets of ρ which, at the end of the ith cycle, are currently maximal complete subgraphs in U (that is, subsets S for which all $\frac{1}{2}(k + 2)(k + 1)$ similarities η have been filled in U). Let ρ_i' be the complement of ρ_i'' in ρ; that is

$$\rho = \rho_i' + \rho_i'' \text{ and } \rho_i' \,.\, \rho_i'' = \phi$$

We observe that for each subset $S \epsilon \rho_i'$, $|\eta'| \geqslant 2$, since step 4 of the algorithm always completes the subgraphs for those subsets having $|\eta'| = 1$. Let

$$\epsilon_i = \rho_i'' - \rho_{i-1}''$$

be the set of all maximal complete subgraphs of $(k + 2)$ objects which are completed during the ith cycle; then for each $S \epsilon \epsilon_i$ we have $|\eta'| \geqslant 2$ at the start, and $|\eta'| = 0$ at the end of the ith cycle. Hence at least 2 of the similarities η for S are filled during the ith cycle, and therefore, by virtue of the ordering on Q, R is the *non-unique* minimum similarity in η for all $S \epsilon \epsilon_i$. It follows that the similarities η cannot be further modified by a Method I-type reduction on any $S \epsilon \epsilon_i$. But

$$\rho_i'' = \epsilon_i + \rho_{i-1}'' = \epsilon_i + \epsilon_{i-1} + \rho_{i-2}'' = \ldots$$
$$= \epsilon_i + \epsilon_{i-1} + \ldots + \epsilon_1$$

Hence ρ_i'' contains no subset S for which η can be modified by a Method I-type reduction. Now suppose at the tth cycle of Method II all elements of U have been filled and the algorithm terminates, then $\rho_t' = \phi$, and $\rho_t'' = \rho$. Hence for all $S \epsilon \rho$, the similarities in η cannot be further modified by a Method I-type ultrametric reduction. It follows that the matrix U obtained by Method II cannot be further modified by Method I.

Stage 4 of Method II requires a search through all $(k + 2)$—subsets of P which contain the pair of objects (g, h). Since there are $\binom{N-2}{k}$ such subsets for each of the $\frac{1}{2}N(N-1)$ entries to be filled into U, the ratio of the work required for Methods I and II is

$$\frac{W_{II}}{W_I} = \frac{\frac{1}{2}N(N-1)}{p} \times \frac{(N-2)^C k}{N^C(k+2)} = \frac{1}{2p}(k+2)(k+1)$$

where p is the number of scans of U which are required for the completion of Method I. Since p is usually in the range 3 to 5, it is clear that Method II requires considerably more work than Method I except for the marginal case when $k = 1$. However, the approach of Method II now suggests the following three ways of reducing the amount of work W_{II}:

(i) After any one entry in U, the number of subsets that must be examined at stage 4 can be minimised by excluding certain objects from the $(N - 2)$ possibilities.

(ii) During stage 4, the size of these subsets can be reduced under certain conditions when a local value of k, which is smaller than the general value of k, is adopted. In general, this also reduces the number of subsets that must be examined.

(iii) We also consider situations in which the general value of k can be reduced, and it is shown, in the paragraph on ending conditions, that the algorithm can be terminated as soon as k rows of U have been completely filled. Hence the factor $\frac{1}{2}N(N-1)$ for W_{II} can be improved.

For the ensuing development we shall adopt certain new terms which are defined as follows:

1. If, at some stage of Method II, the similarity U_{ij} is filled in U then we say that objects i and j are 'connected'; similarly, if U_{ij} is not filled then i and j are 'not connected' or 'disconnected'.

2. Any subset of q objects for which all pairs of objects are connected is called a 'complete q-subset'.

3. Any subset of q objects for which all but one of the possible pairs of objects are connected is called an 'almost complete q-subset', which we abbreviate to an 'a.c. q-subset'.

4. In any subset of objects S, s_i is the number of objects in S which are connected to the ith member of S. This convention is applied to the base subset B (defined below), and the object universe P, where b_i and p_i are the respective connection counts for the ith object.

Step 4 of Method II can now be described as a search

for all a.c.$(k+2)$-subsets of P which contain objects g and h, and in the next three sections we discuss means of improving the efficiency of this search.

Reducing the number of subsets

Let $S = \{\alpha_1, \alpha_2, \ldots, \alpha_k, g, h\}$ be an a.c. $(k+2)$-subset, then by definition exactly one of the connections within S is missing. Hence k of the members of S have $(k+1)$ connections within S, while the remaining two members are disconnected and have exactly k other connections. Since the connection U_{gh} is completed during step 3 of Method II, g and h cannot be the disconnected pair of objects, and therefore

$$\text{either} \quad s_g \geqslant k \text{ and } s_h = k+1$$
$$\text{or} \quad s_g = k+1 \text{ and } s_h \geqslant k \qquad (1)$$

Also for every object $i \in S$, we have

$$s_i \geqslant k \qquad (2)$$

and it follows that each $i \in S$, $i \neq g, h$ is at least connected to g or h. We can, therefore, restrict the objects α_i, which are placed in S during the search for a.c.$(k+2)$-subsets, to those objects which are at least connected to g or h, and we define the base subset B as the set of all such objects together with objects g and h. Associated with each object $i \in B$ we have the number of connections b_i between i and the other objects in B.

Note that for each object $i \in S$, where S is any subset of $(k+2)$ objects taken from B, $s_i \leqslant b_i$; it follows, from (1) above, that no such a.c.$(k+2)$-subset S exists unless

$$\text{either} \quad b_g \geqslant k \text{ and } b_h \geqslant k+1$$
$$\text{or} \quad b_g \geqslant k+1 \text{ and } b_h \geqslant k \qquad (3)$$

Similarly, suppose that for some object $i \in B$, $i \neq g, h$, we have

$$b_i < k$$

then for any subset S containing i, $s_i \leqslant b_i < k$, and hence S is not an a.c.$(k+2)$-subset, by (2) above. Therefore, no a.c.$(k+2)$-subset exists which includes an object i for which $b_i < k$, and hence such objects can be removed from B. If there are any removals, the b_i's for the remaining objects are recomputed and the procedure iterates until there are no further removals. The search for a.c.$(k+2)$-subsets can be concluded if, at any stage of this procedure,

either (i) B is now empty,
or (ii) B is not empty, but either g or h have been removed,
or (iii) B is not empty, but condition (3) is no longer satisfied,

in which case we return to stage 3 of the algorithm. Otherwise, B contains a list of objects, including g and h, which may form a.c.$(k+2)$-subsets and we proceed to consider methods for reducing the value of k and hence the size of the subsets that must be examined in the subsequent search.

Reducing the size of the subsets

We denote the number of objects in the base subset B by $|B|$, and hence

$$b_i \leqslant |B| - 1$$

Suppose that for some object $i \in B$, $b_i = |B| - 1$; that is, i is connected to every other object in B. Then if i is removed from B we can reduce the value of k by 1 in the search for a.c.$(k+2)$-subsets of B. Let B' be the subset of B which excludes object i, and let $S' = \{\alpha_1, \alpha_2, \ldots, \alpha_{k+1}\}$ be any a.c.$(k+1)$-subset of objects taken from B', then the subset $S = \{\alpha_1, \alpha_2, \ldots, \alpha_{k+1}, i\}$ is an a.c.$(k+2)$-subset since $b_i = |B| - 1 \Rightarrow i$ is connected to every $\alpha_j \in S'$. Notice that the removed object i could be either object g, object h or some other member of B, but we always require that S contains both objects g and h (the only a.c.$(k+2)$-subsets which can be found contain both objects g and h by virtue of the insertion of U_{gh}). This result can be generalised to the extent that if t members $i \in B$ satisfy $b_i = |B| - 1$, and these t objects are removed to form the residual base subset B', then an a.c.$(k+2-t)$-subset S' of B' becomes an a.c. $(k+2)$-subset S of B with the addition of the t completely connected objects previously removed from B. Furthermore, the single disconnected pair of objects in S' will be the same disconnected pair in S, and in the limiting case when $t \geqslant k$ but $|B'| > 0$, any disconnected pair of objects in B' is associated with an a.c.$(k+2)$-subset of B.

To implement these results we use a local value k_L of k which applies throughout the search for a.c.$(k+2)$-subsets of B. Initially $k_L = k$, and each object $i \in B$ for which $b_i = |B| - 1$ is removed from B and k_L is reduced by 1. When all such removals are complete, and provided that $k_L > 0$, B is searched for a.c.(k_L+2)-subsets subject to the inclusion of either or both objects g and h provided that either or both objects g and h are retained in B; when $k_L \leqslant 0$ and $|B| > 0$, B is searched for disconnected pairs of objects. When such a subset or disconnected pair is found, its single residual disconnection is completed in U with the current similarity R, and the associated object pair is added to L (as described at Stage 4). If after all removals B is empty, no a.c. $(k+2)$-subsets are to be found and the search is concluded. In the computer program KDEND, the search for (k_L+2)-subsets is further reduced by first considering triples of objects from B. When a complete or a.c. triple is found in B, other objects are tested for addition until a complete or a.c. 4-subset is found, and so on, until an a.c.(k_L+2)-subset has been obtained. The choice of objects for addition is, at each stage, restricted to those which have neither been considered in a previous base triple, nor have been considered at a previous stage in the generation of the current subset. Furthermore, when objects are being tested for addition to a current subset of size r, it is only necessary to examine r connections to determine if the addition yields a complete or a.c.$(r+1)$-subset. This procedure, which is used only when $k_L > 1$, further reduces the amount of work required for the consideration of all subsets of size (k_L+2) taken from B.

Ending conditions

In the previous section it was shown that we can remove from B any object i for which $b_i = |B| - 1$, and reduce the local k_L value by 1. If we consider the universe of objects P as base subset, where p_i is the number of overall connections for object i, then the result can be generalised as follows: if there exists an

object $i \in P$ such that $p_i = N - 1$, then we can remove object i from further consideration and reduce the general value of k by 1. Furthermore, when $k = 0$ after such a reduction, all empty elements of U can be filled with the current similarity R; that is, the matrix U can be completed as soon as k rows or columns are filled. In the computer program, the overall connection counts p_i are stored and updated at each insertion into U. When one such count reaches $N - 1$, the associated object i is deleted from further consideration and the value of k is reduced by 1; however, if such an insertion occurs during a scan of the base subset B and $k \neq 0$, it is important to retain object i with the current local value of k_L unmodified until the scan of B has been completed. When an insertion reduces k to zero, we exit from further scanning of B and complete all empty elements of U with the current similarity R. We are now finished, and U contains the final reduced matrix.

The Cluster Recognition Algorithm

Jardine and Sibson (1968) appear to have avoided the problem of automatically isolating clusters from the reduced matrix U, and simply advocate their construction by hand for any chosen similarity threshold h. This process, although not difficult, can be tedious and, therefore, we proposed an algorithmic solution to the problem.

A binary linkage matrix L is defined as follows: for any chosen similarity threshold h, we set

$$L_{ij} = 1 \text{ if } U_{ij} \geqslant h$$

or

$$L_{ij} = 0 \text{ if } U_{ij} < h$$

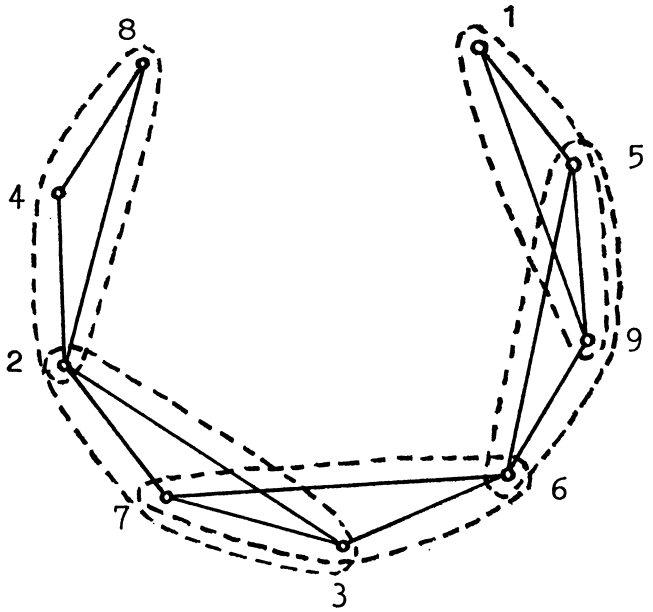
and

$$L_{ii} = 1 \text{ for all } i$$

Fig. 1 shows the linkage matrix L obtained when $h = 5 \cdot 50$, from the reduced matrix U derived in the example used by Jardine and Sibson when $k = 3$. Also shown is a linkage diagram for this set of 9 objects, on which connected objects are joined and clusters are indicated by dotted lines. We notice that there are two types of objects: 'explicit' objects belong to one cluster only, and 'overlap' objects belong to two or more clusters. Also, for each cluster of objects we know that every member is connected to every other member. The problem of cluster recognition is to isolate the maximal complete subgraphs expressed in L, and remove them one at a time until all such clusters have been found. Furthermore, in removing connections from L, it is important to retain those connections to overlap objects which are used to describe other clusters. For example, suppose we remove cluster (159) (shown in Fig. 1) and in doing so delete the (5, 9) connection, then cluster (569) will not be recognised later. Our approach is to search for explicit (see below) objects, remove the clusters containing them, and then delete only those connections to each explicit object. Hence, in our example of cluster (159) we discover that object 1 is explicit and so we reset the first row and column of L to zero.

An 'overlap' object is defined as an object i, connected to objects $\{\alpha_1, \alpha_2, \ldots\}$, for which at least one of the pairs (α_j, α_k) is not connected. Any other object is termed 'explicit'. It follows that only overlap objects belong to more than one cluster. In our example, object 1 is explicit because it is connected to 5 and 9, and the pair (5, 9) is also connected. By contrast, object 5 has connections to objects 1, 6 and 9, and since the pair (1, 6) is not connected, 5 is an overlap object. We observe that objects 1, 4 and 8 are the only explicit objects in Fig. 1.

The first step in the cluster recognition algorithm is to search for isolated objects. Each such object is characterised by a 1 in the diagonal and zeros elsewhere of its



	1	2	3	4	5	6	7	8	9
1	1	0	0	0	1	0	0	0	1
2	0	1	1	1	0	0	1	1	0
3	0	1	1	0	0	1	1	0	0
5	1	0	0	0	1	1	0	0	1
6	0	0	1	0	1	1	1	0	1
7	0	1	1	0	0	1	1	0	0
8	0	1	0	1	0	0	0	1	0
9	1	0	0	0	1	1	0	0	1

Fig. 1. An example of overlapping clusters for a 9-object population, and the associated linkage matrix derived from the ultrametric. Clusters are indicated by dotted lines, and solid lines join objects whose similarity exceeds the threshold. In the linkage matrix significant similarities are coded 1. Observe that only objects 1, 4 and 8 are explicit.

associated row and column in L. The diagonal 1 element is replaced by 0 for each isolated object, and a 'single object cluster' message is printed. Next we look for explicit objects and their parent clusters by the following terminating procedure:

(1) We scan L for objects that have 1's in their associated row: suppose we find that the ith. object has connections in L, then vectors I and I' are set equal to the ith row of L.

(2) For each '1' in I, corresponding to a connection (i, j), we replace I' by the product of vector I' and the jth. row in L.

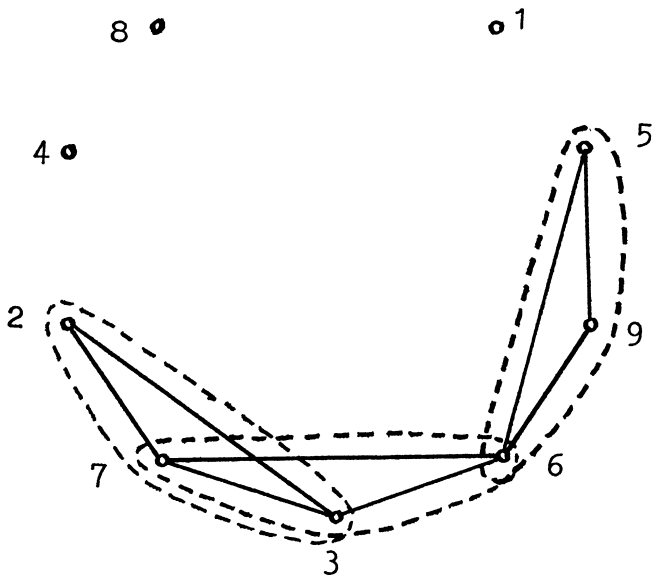
(3) When all 1's in I have been treated in this way, the current vector I' is compared with the original vector I: if they are the same, then object i is explicit; if they differ, then i is an overlap object. We ignore any overlap objects and continue the scan of L until an explicit object is found, when we proceed to 4.

(4) Suppose there are c 1's in I, then each 1 corresponds to a member of the current cluster C, whose size is c. Since the row and column of L corresponding to each explicit object in C will be identical to I, the number of connections in L for each such object will be c. We now form a list $E = \{\alpha_1, \alpha_2, \ldots\}$ of objects $\alpha_i \in C$ such that the row in L corresonding to α_i contains exactly c 1's; E is therefore a list of all the explicit objects in C.

(5) The rows and columns of L corresponding to each $\alpha_i \in E$ are now reset to zero, and we return to (1) and look for new explicit objects and their parent clusters. When every element of L is zero, the procedure is terminated and all clusters have been found.

This process, when applied to the linkage diagram of **Fig. 1**, will first recognise cluster (159), find the explicit object 1 and then reset row 1 and column 1 of L to zero. Next, explicit object 4 will be discovered and the parent cluster (248) recognised. In this case, E will contain explicit objects 4 and 8, and so the corresponding rows and columns of L will be reset to zero. **Fig. 2** shows the stage that has now been reached. Two clusters have been peeled off the chain of overlapping clusters with the result that objects 2, 5 and 9, which were previously overlap objects, are now explicit. When the algorithm is reapplied, the clusters (237), (367) and (569) are recognised, at which point L contains zeros everywhere and the scan is concluded.

So far we have omitted one special case: the situation when every object is in an overlap position. This may occur with the entire population or with a subset of objects, but in any event, it is detected when steps 1–3 of the algorithm fail to find an explicit object while there are still 1's in L. **Fig. 3** shows the connections between six overlap objects, for which the clusters are (1245), (1246), (1345) and (1346). We observe that if any overlap object i has p_i residual connections in L (the connection L_{ii} is included), then the maximum size of cluster to which i can belong cannot exceed $p_i - 2$. Let p_{max} be the maximum number of residual connections in L for any one object, then a search through all subsets comprising from 2 to $p_{max} - 2$ overlap objects will reveal all overlapping clusters. Our approach is to first consider all subsets of size $p_{max} - 2$ objects, and form a list of those subsets which are maximal complete subgraphs in L. Next, all subsets of size $p_{max} - 3$ are considered and any maximal complete subgraph is added to the list provided that it does not form a subset of a cluster previously found. As each new cluster is discovered, a check is made to test whether the present list of clusters accounts for all the residual connections in L. When this happens the process is terminated. Although this procedure appears to be lengthy, in practice the total overlap condition occurs for small spherical groupings of overlapping clusters (as in **Fig. 3**) or for connected circuits of overlapping clusters, because of the



	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	1	0	0
3	0	1	1	0	0	1	1	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	1	0	0	1
6	0	0	1	0	1	1	1	0	1
7	0	1	1	0	0	1	1	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	1	0	0	1

Fig. 2. Residual clusters and linkage matrix for the 9-object population in Fig. 1 after 2 cycles of the cluster recognition algorithm. Clusters (248) and (159) have been recognised and removed from the linkage matrix with the result that objects 2, 5 and 9 are now explicit. The next cluster that will be recognised is (237).

severe requirement that every object must belong to more than one cluster. Consequently p_{max} is usually small, and the exhaustive search for clusters is often terminated at or near the subset size $p_{max} - 2$ level. In the example shown in Fig. 3, $p_{max} = p_1 = 6$ so that only the 15 subsets of size 4 need be considered.

Finally, one modification to the cluster recognition algorithm has to be made. A cluster can contain subsets of explicit and overlap objects, respectively. When a cluster is recognised, those connections to the explicit objects are removed from L while the connections to the overlap objects are retained in L. If the overlap objects happen to belong to more than one other cluster, then this subset of overlap objects will be recognised later as a separate cluster. This means that not only clusters, but also subsets of clusters will be recognised by the algorithm. To correct this case, the computer program compiles a list of the clusters as they are found. When the cluster recognition phase is terminated, this list is searched for duplicated cluster subsets which are removed before the final classifications are printed.

Obtaining a hierarchy of clusterings

The cluster recognition algorithm is defined for one chosen similarity threshold h. However, if we apply the algorithm using each unique entry contained in U as threshold, we obtain the hierarchy of all the clusterings that can possibly be generated for any chosen value of h. In their introduction of Method I, Jardine and Sibson distinguish between the variants of the sequence which yield non-overlapping ($k = 1$) and overlapping ($k > 1$) clusters by the terms 'heirarchic' and 'non-hierarchic' respectively. This use of the term 'hierarchy' to describe a sequence of nested partitions which give rise to strictly disjoint subsets, differs from the conventional meaning where 'hierarchic' refers to those methods that produce ordered clusterings from a monotonic decreasing similarity threshold. For this reason, we prefer to use the terms 'non-overlapping' and 'overlapping' to describe the type of clusters which are obtained. In our terminology

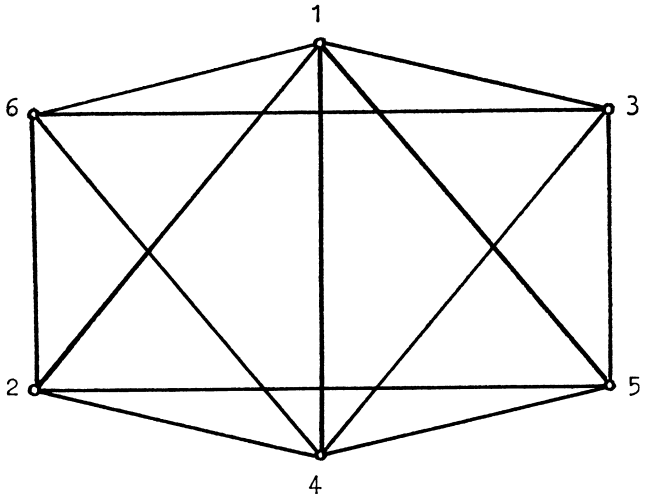


Fig. 3. An example of the complete overlap situation for 6 objects when $k = 4$. The clusters are (1245), (1246), (1345) and (1346).

therefore, we obtain a hierarchy of clusterings for all values of k by applying the ultrametric similarities contained in U to the cluster recognition algorithm in order of decreasing similarity.

In the computer program KDEND, the ordering of ultrametric similarities is stored in Q at step 2 of Method II. Any value Q_p, corresponding to an entry U_{ij} which is not filled, is retained in Q for subsequent use as a threshold with the cluster recognition algorithm.

Large populations

In a situation where we wish to describe a large population in terms of a few 'types', we tend to look for large clusters that signal a concurrence of pattern. These large groups need not be the most interesting—the peripheral and intermediate objects may attract our attention through being unusual—nevertheless, we must first detect and isolate the types before the classification can be examined in detail. If we consider the specific instance of a large population represented by a multivariate Euclidean sample distribution, then as $N \to \infty$ the distribution will approximate some complex theoretical density function F. Suppose that at some arbitrary level of probability p the volume $F \geqslant p$ can be partitioned by closed disjoint surfaces, then it has been reasoned (Wishart, 1968, 1969A) that each closed surface is an instance of a particular 'type' within the population. Conversely, if there is only one connected surface $F = p$, we can conclude that there is no evidence that the data can be subclassified 'naturally' at level p. Furthermore, by examining different values of p we select different levels of hierarchic classification: in taxonomy, certain levels might correspond to family, genera, species, etc. Wishart (1968) proposes the following 'one-level' method designed to detect such classes within a large population. A distance threshold r and density parameter k are chosen, and each sample point is visualised at the centre of a sphere of radius r, which is used to estimate the density of observations within the local spherical neighbourhood of the point. We assume that those spheres which contain at least k other points are located in the volume $F \geqslant p$, where p is now some unknown probability determined by r, k and N, and we detect the disjoint connected surfaces by joining all such spheres which intersect. In short, any point which lies within a distance r from at least k other points is 'dense', and clusters are formed by using single linkage on the dense points at threshold r. The method has been further developed as a hierarchical process which obtains the family of nested disjoint surfaces corresponding to all probabilities p, and does not require the choice of threshold. However, the classification obtained at one threshold level r can conveniently be used to compare Wishart's method with the k-partition.

Earlier we introduced the concept of a chain cluster for the k-partition, which consists of a straggle of maximal complete subgraphs that intersect in at least k vertices. We shall restrict our interest in the clusters of a large population to the chain clusters, or those distinct maximal complete subgraphs that at least have the potential to chain if there were any local connections. We observe that each individual maximal complete subgraph (which we shall call a 'unit') must possess at least $k + 1$ mutually connected objects. Furthermore, since each unit is maximally connected at threshold r its diameter cannot exceed r. Hence the units can be considered as spherical dense neighbourhoods, containing at least $k + 1$ objects, and clusters are formed by

connecting intersecting units (provided that they intersect in at least k vertices). This is essentially the concept on which Wishart's probabilistic method is based, and it is interesting to note that both methods degenerate to single linkage when $k = 1$. Of course, there are differences: two intersecting chain clusters are separated by the k-partition if they fail to intersect in more than $k - 1$ objects, and the k-partition also recognises maximal complete subgraphs which contain fewer than $k - 1$ objects. Nevertheless, with a large population which exhibits distinct data swarms we can expect the two methods to yield very similar major clusters (that is, if it were possible to use the k-partition method with a large population).

These considerations now suggest an algorithm for the specific solution of large clusters by the k-partition when a single initial threshold value is given. We observe that every member of a unit is connected to at least k other objects, and conversely any object which has less than k connections at threshold h cannot belong to a large cluster. We can therefore eliminate some of the objects which are not members of units at threshold h by removing those objects having less than k connections. Furthermore, since the connection counts within the residual population may be modified by these removals, we now recompute the connection counts, reapply the test for k connections and repeat this procedure until there are no further removals. The residual population now contains all the members of units, and possibly some others. We now perform the Method II ultrametric reduction on the similarity submatrix for the residual population, and use the cluster recognition algorithm at threshold h to identify the large cluster members. This algorithm will evidently work well for any size of population, provided that the similarity threshold is sufficiently large to yield a residual population of order less than 60. However, since this restriction reduces the effective generality of the k-partition method, we have not programmed the algorithm.

Conclusions

Fig. 4 shows the times required for Methods I and II using different population sizes. Each timing represents the average of two trials using different 2-cluster populations generated by a pseudo-normal number routine. The similarity matrix was computed from Euclidean distances, and the tests were completed by FORTRAN II programs on the IBM 1620 II, and FORTRAN IV programs on the IBM 360/Model 44. It became clear that the sequence $k = 1$ to 5 could only be reasonably computed with the IBM 360/Model 44 for up to 20 individuals by Method I and 60 individuals for Method II (these limits can probably be extended by 5 and 20 objects respectively on a faster computer). The largest population size that we tried was 35, for which Method II required 18 minutes for the completion of the sequence $k = 1$ to 5 on the IBM 360/Model 44. In view of the fact that the similarity matrix must be held in core for these algorithms, Method II is necessarily restricted to small data problems.

An important practical feature of the k-partition is the large number of clusterings that are obtained. In our trials with the 9-object population cited by Jardine and Sibson, the cluster recognition algorithm produced

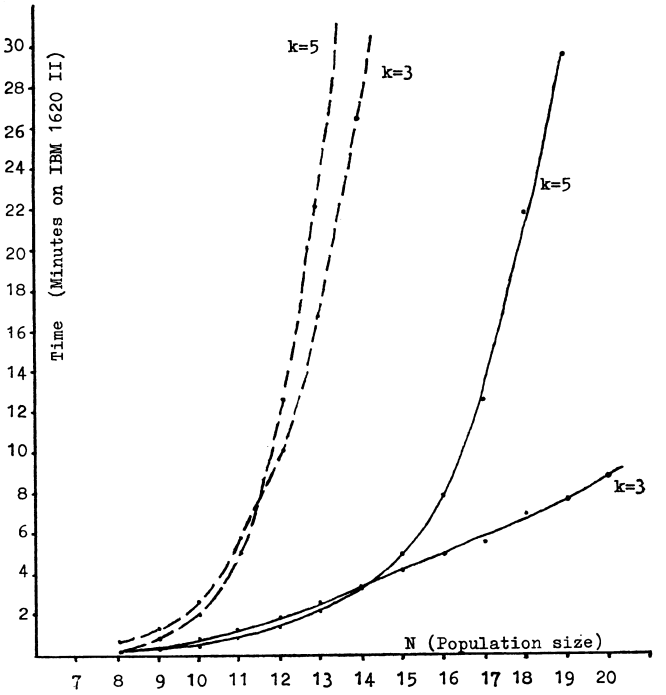


Fig. 4. An indication of the times required for the two algorithms. Dotted lines correspond to Method I; solid lines are Method II.

70 separate classifications for the sequence $k = 1$ to 5, and although not all of these groupings were unique, a user must be severely selective when presenting his results. This is in complete contrast to Wishart's hierarchic method which incorporates a selective mechanism that seldom presents the user with more than about 12 groupings for any population size.

On the whole, we feel that the detailed analysis of data structure which is offered by the k-partition is desirable for small populations; large data applications which suggest a natural class approach should be referred to the alternative probabilistic model.

Appendix

A computer program (KDEND) for the k-partition by the modified Method II algorithm with cluster recogtion has been incorporated in the 'CLUSTAN' package of FORTRAN Programs for cluster analysis. The program is included in the second version (CLUSTAN IA) and operates in conjunction with programs FILE and CORREL of CLUSTAN I. Further details can be obtained from D. Wishart.

1. Features of CLUSTAN I

This first version of CLUSTAN (Wishart, 1969C) contains programs FILE, CORREL and RESULT (which perform the basic operations of data input, principal components, computation of similarities and analysis of results) together with the probabilistic clustering method of Wishart (Wishart, 1968, 1969A) (program MODE), and 7 hierarchical fusion methods (program HIERAR: see Lance and Williams, 1967A; Wishart, 1969B).

By using magnetic disk as backing store, the standard

version of CLUSTAN can accommodate problems subject to the following general conditions:

Maximum number of individuals (N) = 999
Maximum number of continuous variables (MN) = 200
Maximum number of binary variables (MB) = 400
No missing data permitted.

2. Features of CLUSTAN IA

This group of programs is designed to complement and extend CLUSTAN I, and uses FILE and CORREL for the initial computations. The package, which is yet to be published, contains programs KDEND, DIVIDE (for 9 monothetic divisive options: see Lance and Williams, 1965; MacNaughton-Smith, 1965; Crawford and Wishart, 1967; Gower, 1967), CENTRO (centroid sorting with some non-combinatorial coefficients: see Lance and Williams, 1966) and RELOC (an iterative assignment procedure to optimise the error sum, group coefficient, etc.: see Jancey, 1966; Lance and Williams, 1967B; Crawford and Wishart, 1968). The package is further supplemented by routines PLINK and SCAT for off-line dendrogram and scatter-diagram plotting using a graph plotter.

3. Modifications of CLUSTAN

Although CLUSTAN is disk oriented, all disk input/output is controlled by one subroutine DISKIO which is easily modified for particular machine requirements, or simulation of disk in core and on magnetic tape. Modified versions of CLUSTAN, in Fortran II and IV, are currently available for use with the IBM 1620 Model II ($N \leqslant 400$), IBM 360 series ($N < 1{,}000$), IBM 7094, ICL 1905 1909 4100 4/50, Atlas, KDF9, CDC 6600 and UNIVAC 1100. Other implementations are being considered.

References

CRAWFORD, R. M. M., and WISHART, D. (1967). 'A rapid multivariate method for the detection and classification of groups of ecologically related species', *J. Ecol.*, Vol. 55, p. 505.

CRAWFORD, R. M. M., and WISHART, D. (1968). 'A rapid classification and ordination method and its application to vegetation mapping', *J. Ecol.*, Vol. 56, p. 385.

GOWER, J. C. (1967). 'A comparison of some methods of cluster analysis', *Biometrics*, Vol. 23, p. 623.

JANCEY, R. C. (1966). 'Multidimensional group analysis', *Aust. J. Bot.*, Vol. 14, p. 127.

JARDINE, N., and SIBSON, R. (1968). 'The construction of hierarchic and non-hierarchic classifications', *Comp. J.*, Vol. 11, p. 177.

JOHNSON, S. C. (1967). 'Hierarchical clustering schemes', *Psychometrika*, Vol. 32, p. 241.

LANCE, G. N., and WILLIAMS, W. T. (1965). 'Computer programs for monothetic classification ("Association analysis")', *Comp. J.*, Vol. 8, p. 246.

LANCE, G. N., and WILLIAMS, W. T. (1966). 'Computer programs for hierarchical polythetic classification ("Similarity analyses")', *Comp. J.*, Vol. 9, p. 60.

LANCE, G. N., and WILLIAMS, W. T. (1967A). 'A general theory of classificatory sorting strategies. I. Hierarchical systems', *Comp. J.*, Vol. 9, p. 373.

LANCE, G. N., and WILLIAMS, W. T. (1967B). 'A general theory of classificatory sorting strategies. II. Clustering Systems', *Comp. J.*, Vol. 10, p. 271.

MACNAUGHTON-SMITH, P. (1965). 'Some statistical and other numerical techniques for classifying individuals', (H.M.S.O. Home Office Research Unit Report No. 6), London.

ROUX, M. (1968). 'An algorithm to construct a particular kind of hierarchy', *Proc. of Univ. of St. Andrews Colloq. in Numerical Taxonomy* (Editor: A. J. Cole) p. 200. Also in: *Numerical Taxonomy*, Academic Press (London), p. 234.

WILLIAMS, W. T., LAMBERT, J. M., and LANCE, G. N. (1966). 'Multivariate methods in plant ecology V. Similarity analyses and information-analysis', *J. Ecol.*, Vol. 54, p. 427.

WISHART, D. (1968). 'Mode analysis: a generalisation of nearest neighbour which reduces chaining effects', *Proc. of Univ. of St. Andrews Colloq. in Numerical Taxonomy* (Editor: A. J. Cole), p. 233. Also in: *Numerical Taxonomy*, Academic Press (London), p. 283.

WISHART, D. (1969A). 'Numerical classification method for deriving natural classes', *Nature*, Vol. 221, p. 97.

WISHART, D. (1969B). 'An algorithm for hierarchical classifications', *Biometrics*, Vol. 22, p. 165–170.

WISHART, D. (1969C). 'Fortran II programs for 8 methods of cluster analysis (CLUSTAN I)', *Kansas Geol. Comp. Contr. Series*, No. 38, Kansas.