A new approach to the computation of the Jardine-Sibson B_k clusters

F. J. Rohlf

Department of Ecology and Evolution, State University of New York at Stony Brook, Stony Brook, New York 11794, USA

A new approach to the computation of Jardine and Sibson's (1968a) (fine) k-cluster method B_k is described. The algorithm makes use of the fact that the set of dissimilarities which are invariant under the B_k clustering transformation can be represented as an abstract graph. The proposed algorithm for computing this graph obtains both the (weak) k-ultrametric dissimilarity matrix and the k-clusters simultaneously.

(Received March 1972, revised March 1974)

This paper reports on a new approach to the computation of Jardine and Sibson's (1968a; 1968b) nonhierarchic stratified cluster analysis methods B_k. This type of cluster analysis has a number of desirable mathematical properties (for example, it uses only rank order information from the dissimilarity matrix and the implied k-ultrametrics are continuous functions of the input dissimilarities (Jardine and Sibson, 1971). However, it should also be pointed out that other properties are considered more important by other workers (e.g. Fisher and Van Ness, 1971). Method B_1 is the single linkage method; B_2 yields a system of clusters in which clusters at a given level may overlap so as to have at most one object in common; in general B_k yields a system of clusters in which any pair of clusters can have at most $k-1$ objects in common at a given level. As one increases k one obtains increasingly better fit between the original dissimilarity matrix and the k-ultrametric dissimilarity matrix implied by the results of the cluster analysis.

The presently available programs for the computation of B_k clusters are based upon algorithms which successively transform the original dissimilarity matrix into a k-ultrametric matrix of dissimilarities (B_1 represents an important special case for which simpler techniques are well known). A simple procedure for carrying out this transformation was described by Jardine and Sibson (1968b). It is based upon *repetitive* complete searches of *all possible* sets of $k+2$ objects (until no modifications can be made). More complex procedures which eliminate much of the redundant computations were published by Cole and Wishart (1970). In addition a program by Dr. J. K. M. Moody is described in Jardine and Sibson (1971, Appendix 3). A disadvantage of these algorithms is that they yield the k-ultrametric dissimilarity matrix rather than information on cluster membership. For this reason a separate and rather complex program is needed to analyse the k-ultrametric matrix to isolate all clusters at a specified threshold level of dissimilarity. Algorithms for doing this are described by Cole and Wishart (1970) and in Jardine and Sibson (1971, Appendix 5 based upon the work of J. K. M. Moody and J. Hollis). The procedure described below has the advantage that the clusters at each distinct threshold level can be found directly and can be programmed quite easily in FORTRAN or any other higher level language.

Relationship between B_k and B_kG

Jardine and Sibson (1971) have pointed out that it is useful to view a cluster analysis as being a function which transforms the original input dissimilarity coefficient, d_{ij}, into a new output dissimilarity coefficient, u_{ij}, which has various properties depending upon the type of cluster analysis employed. The output dissimilarities represent the similarity among objects implied by the results of the cluster analysis. If a hierarchical clustering method is applied which yields nested clusters then the output dissimilarities will satisfy the ultrametric inequality

$$u_{ij} \leq \max\{u_{ik}, u_{jk}\}$$

for all objects i, j, and k in the study. An example is shown in **Table 1.** Jardine and Sibson (1968a) have proposed a generalisation, B_k, of the single linkage method in which the output dissimilarities satisfy the (weak) k-ultrametric inequality.

$$u_{ij} \leq \max\{u_{xy} : x \in S \cup \{i,j\}, y \in S\}$$

where S is a completely connected set of objects of size k and i, j, x, and y, are any objects in the study (Jardine and Sibson, 1971). Examples for $k = 2$ and 3 are given in **Table 2.** For

Table 1 **Dissimilarities between 10 hypothetical objects. Original dissimilarity matrix (above the diagonal) and ultrametric dissimilarities (below the diagonal) resulting from the application of clustering method B_1. Asterisks identify the invariant elements discussed in the text.**

	A	B	C	D	E	F	G	H	I	J
A	0	4	22	18	25	22	1	14	23	25
B	3	0	21	16	23	15	3	8	19	32
C	13	13	0	2	6	9	24	26	22	34
D	13	13	2*	0	7	10	21	24	21	44
E	13	13	6*	6	0	12	26	29	23	46
F	13	13	9*	9	9	0	20	17	11	34
G	1*	3*	13	13	13	13	0	5	21	22
H	5	5	13	13	13	13	5*	0	13	24
I	13	13	11	11	11	11*	13	13*	0	23
J	22	22	22	22	22	22	22*	22	22	0

Table 2 **(weak) 2-ultrametric dissimilarities (above the diagonal) resulting from the application of clustering method B_2 to the data in Table 1 and (weak) 3-ultrametric dissimilarities (below the diagonal) resulting from the application of clustering method B_3. Asterisks identify the invariant elements.**

	A	B	C	D	E	F	G	H	I	J
A	0	4*	17	17	17	17	1*	8	17	23
B	4*	0	16	16*	16	15*	3*	8	17	23
C	21	21*	0	2*	6*	9*	17	17	17	23
D	18*	16*	2*	0	7*	10*	17	17	17	23
E	21	21	6*	7*	0	10	17	17	17	23
F	20	15*	9*	10*	12*	0	17	17*	11*	23
G	1*	3*	21	20	21	20*	0	5*	17	22*
H	14*	8*	21	20	21	17*	5*	0	13*	23
I	20	19*	21	20	21	11*	20	13*	0	23*
J	24	24	24	24	24	24	22*	24*	23*	0

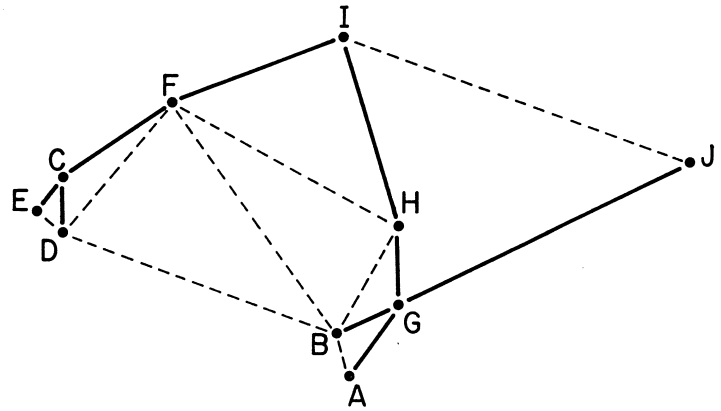

Fig. 1 Minimum spanning tree (solid lines) and the B_2G graph (solid and dashed lines) for the data in Table 1. Edges correspond to the elements of the U_1 and U_2 matrices in Tables 1 and 2 which are marked with asterisks (*).

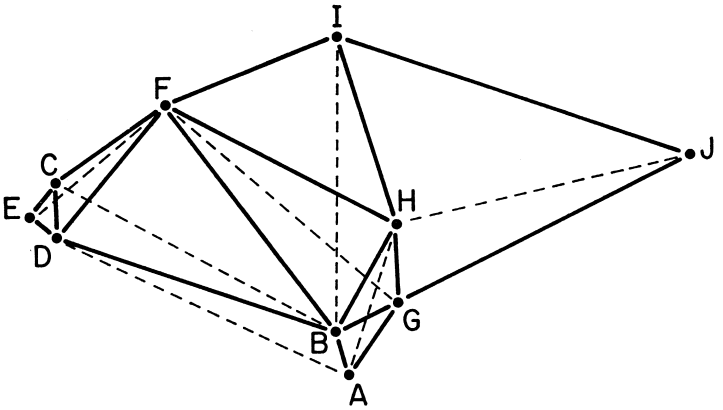

Fig. 2 B_3G graph for the data in Table 1. Edges correspond to the elements of the U_3 matrix in Table 2 which are marked with asterisks (*). Dashed lines indicate the edges which must be added to B_2G to obtain the B_3G.

$k = 1$ this reduces to the ordinary ultrametric inequality.

Since the B_k methods are examples of what Jardine and Sibson (1971) call subdominant methods the output dissimilarities must not only satisfy the (weak) k-ultrametric inequality but must also be as large as possible subject to the restriction that $u_{ij} \leq d_{ij}$ (i.e. the elements which are invariant under the B_k clustering transformation) contains sufficient information to determine B_k clusters and hence all of the other u_{ij}. These invariant elements are indicated by asterisks in Table 1 and 2. As shown in Rohlf (1974) it is convenient to define an abstract graph, B_kG, whose vertices consist of the objects in the study and edges connect a pair of vertices i and j if and only if $u_{ij} = d_{ij}$. Edge e_{ij} is of length u_{ij}. The B_kG graphs are at least k-connected. A graph is said to be k-connected if every pair of distinct vertices i and j are joined by at least k chains which have no common vertices (except, of course, for i and j). A chain is the set of edges one traverses in moving from one vertex to another in the graph (Busacker and Saaty, 1965). There can be, however, many additional edges present beyond what would be required for the graph to be k-connected. It appears to be difficult to characterise the graph in general in any way other than that it represents the set of edges corresponding to the dissimilarities invariant under the B_k clustering transformation.

If $k = 1$ and the d_{ij} are distinct the resulting graph B_1G is the minimum spanning tree MST, (**Fig. 1,** solid line). This is, of course, not a very efficient method for its computation. If the d_{ij} are not distinct then there may be more than one MST. In such a case B_1G would be the union of all the possible MST's.

For $k \geq 2$ the graph is more difficult to characterise. A notable feature is the presence of what may be called k-chains. These can be defined as an ordered progression of completely connected sets of size $k + 1$, 'adjacent' pairs of which have k vertices in common. For $k = 1$ this is the usual notion of a chain. In Fig. 1 (solid *and* dashed lines) we have the 2-chain progression: (A, G, B), (G, B, H), (B, H, F), (H, F, I) and in **Fig. 2** we have, for example, the 3-chain progression: (A, G, B, H), (G, B, H, F), (B, H, F, I). Unfortunately (from the point of view of simplicity) there may be additional edges present in the graphs and not all vertices may be part of a k-chain. However, even if they are not (e.g. object J) they will have at least k connections to other vertices so that the graph will still be k-connected. The graph as a whole has the property that just enough of the shortest edges from each vertex are included so as to assure that if the following B_k cluster rules are applied one will end up with a single set containing all of the objects (ties can, of course, result in many additional edges).

1. All objects which are a part of the same CL-set (completely linked set) are defined to be in the same set.
2. All objects which are a part of the same k-chain are defined to be in the same set (any unconnected pairs of vertices in the set are considered to be connected from this point on).
3. All objects which have at least k connections to objects in the same set are defined to belong to that set (with unconnected pairs of vertices connected).
4. Any sets which have at least k objects in common can be merged to form a single set (this new set is made a completely connected set by connecting any unconnected pairs of vertices).
5. Steps 1 to 4 are to be applied iteratively until no further changes can be made in the membership of the sets.

Each edge in these graphs correspond to a clustering level in the k-dendrogram at which two or more objects or clusters fuse. Gower and Ross (1969) have pointed out the relationship between single linkage cluster analysis and the MST (i.e. between B_1 and B_1G). They show that given the MST one can easily produce the clusters obtained by single linkage cluster analysis. Rohlf (1973) presented an efficient technique for simultaneously producing a single linkage cluster analysis (including a dendrogram) while computing the MST. This work led to an investigation of the possibilities of computing B_kG directly and either at the same time or subsequently computing the B_k clusters and (weak) k-ultrametric dissimilarities. Rohlf (1974) discusses a set of simplified rules for obtaining the B_k clusters given only B_kG as well as examples of their application.

Algorithms

A number of possible algorithms for computing MST's were investigated for possible generalisation into procedures for finding B_kG's. It does not appear to be possible to generalise Prim's (1957) algorithm for computing a MST into a more general algorithm for computing B_kG even in the case where there are no ties. The obvious generalisation of this algorithm would seem to be: (at a given step in the computation) to add to the fragment that vertex whose kth smallest dissimilarity to any vertex already in the fragment is the smallest. This yields graphs in which all vertices either belong to k-chains or have k connections to other vertices but they do not always match B_kG nor are they always unique for a given set of data. It is a rapid algorithm so that this approach may be useful from problems involving large numbers of objects in which an approximate solution may be satisfactory.

The algorithms of Kruskal (1956) can, however, be easily generalised. One method (his 'construction A') is—perform the following step as many times as possible: among the edges of the graph not yet chosen, select the shortest edge which does

not make the graph more than simply connected (i.e. does not connect objects already in the same single link cluster). This can be generalised to—perform the following steps as many times as possible among the edges of the graph not yet chosen: select the shortest edge which does not connect objects already in the same B_k cluster. In actually performing this algorithm there are two basic computational tasks. The first is of sorting the dissimilarity values so that one can process them one at a time in order of increasing dissimilarity. Since dissimilarity matrices can be quite large it is important that an efficient sorting procedure be used. Thus the rather commonly used 'bubble sort' is simply too inefficient to be considered. The second problem is that of determining when a dissimilarity value would correspond to an edge which enables one to merge sets using the rules given above. If (as edges are accepted) one builds up ML-sets (maximal CL-sets) and merges them whenever two or more such sets have k or more vertices in common, then one can reject an edge whenever it connects two vertices which were already in the same set. This algorithm has the advantage that the B_k clusters are explicitly computed. The entries in the k-ultrametric matrix can also be computed with a small amount of additional effort. While feasible this algorithm requires an efficient set of routines to create, delete, merge, and search list structures. The computational effort goes up rapidly with both n and k.

Other algorithms were investigated to avoid the time consuming operations of finding all ML-sets (which places a large demand upon the list structure manipulation routines). In the following algorithm less use is made of list structures so that it is sufficient to implement them using threaded lists in FORTRAN. This algorithm (as the previous one) processes edges one at a time from the smallest dissimilarity to the largest. The determination of whether an edge belongs to B_kG is made more efficient by creating the k-ultrametric matrix, U, as we go along. Edge e_{ij} belongs to B_kG if and only if u_{ij} has not yet been filled in. The updating procedure for the U-matrix requires iterative searches as before but only for the relatively few edges which actually belong to B_kG. Thus most elements of the input dissimilarity matrix can quickly be skipped over (this is also an important feature of the Cole and Wishart (1970) algorithm).

The proposed algorithm is as follows:

Step 1
Clear the array which will be used to contain the final (weak) k-ultrametric metrix U, and the vector keeping track of the degree of each vertex in the graph implied by the U-matrix (which is not the same as B_kG since U contains *all* of the implied edges). At this point the input dissimilarity matrix must have been sorted so that one can obtain the dissimilarities in order from smallest to largest. Clear lists L and M (lists of sets).

Step 2
Find the next input dissimilarity, d_{ij} whose corresponding element in the U-matrix, u_{ij}, has not been filled in. Edge i-j (of length d_{ij}) is added to B_kG. u_{ij} is set equal to d_{ij}, the set $\{i,j\}$ is added to L, and we proceed to Step 3.

Step 3
Delete the set at the top of list L and store it in C. Let NC equal the size of set C.

Step 4
Search the U-matrix to determine the B_k clusters.

(*a*) If $NC < k$, then search for objects V connected to all members of C for which the set $C \cup V \nsubseteq a$ cluster in M. If $m \geq 2$, objects V_i, are found then add to L the sets $C \cup \{V_i\}$ for $i = 2, \ldots, m$. Add the first object found, V_1, to C. If at least one ML-set was found go to Step 4, otherwise go to Step 5. Note if $NC + m \leq k$ then further processing can result in the recognition of new clusters but no changes are possible in the U-matrix.

(*b*) Else search for objects connected to at least k members of C. Add them to C and update U. If NC now equals n processing is complete, STOP. If $k > 2$ add to list L sets consisting of the two objects associated with each edge being updated in U and continue the search until no further additions can be made, then go to Step 5.

Step 5
Save C in M unless C is a subset of a set in M. If L is not empty delete from L any sets of size $\geq k$ which are subsets of C.

Step 6
Go to Step 3 if L is not empty. Otherwise the current clusters (in M) can be printed. This output can be rather large if n and k are not small. Go to Step 2 for the next d_{ij}.

At completion U will be completely filled in and will satisfy the (weak) k-ultrametric inequality. The algorithm is similar to Jardine and Sibson's (1968a, p. 476) suggested procedure for finding B_k-clusters by hand for small numbers of objects. Their procedure was to draw a graph whose vertices are the objects and whose edges join just those pairs of objects having dissimilarities $\leq h$. All ML-sets are then found. If two such sets have $\geq k$ vertices in common then the two sets are merged and further edges are drawn so as to make the resultant set of vertices completely linked. ML-sets are redetermined and the process of merging is iterated until no further changes can be made. The resultant ML-sets are the B_k-clusters at level h and all pairs of vertices are connected whose (weak) k-ultrametric dissimilarity is $\leq h$. The proposed algorithm performs these operations in a stepwise manner for each distinct dissimilarity level starting from the smallest. This method is practical if one makes efficient use of the results obtained in previous steps. For large t and small k no computation is necessary for most levels. Only at the critical 'splitting levels' (which correspond to the edges in B_kG) does one actually need to carry out operations like those described above.

At the lowest distinct threshold, h_0, corresponding to the smallest inter-object dissimilarity, d_{ij}, we set $u_{ij} = d_{ij}$. At this point there is only a single cluster containing objects i and j and B_kG consists of a single edge connecting vertices i and j. This is obvious since this set is the largest (and the only) ML-set at this threshold level and u_{ij} must be the largest dissimilarity $\leq d_{ij}$ (by the subdominance property).

Assuming the U-matrix and the B_kG are correct at an arbitrary threshold level h_1, let $h_2 = h_1 + \varepsilon$ such that a single *additional* dissimilarity, d_{ij}, is now less than or equal to h_2. Whether or not the edge $i - j$ is added to B_kG depends upon the state of the i, j element of the U-matrix.

If u_{ij} has already been defined, then objects i and j must already belong to the same cluster at level h_1. Thus no additional changes need to be made to the U-matrix at level h_2. In B_k cluster analysis objects in the same cluster at level h_1 must also be in the same cluster at level h_2. Edge $i - j$ need not be added to B_kG since the connection between i and j must be implied by the other edges (we assumed B_kG was correct at level h_1). Thus all d_{ij} for which the corresponding u_{ij} are already defined are ignored in Step 2.

If U_{ij} has not yet been defined then it is now set equal to d_{ij} (since u_{ij} must be $\leq d_{ij}$ by the subdominance property of B_k) and the edge e_{ij} is added to B_kG. We must now determine whether additional changes must be made to the U-matrix as well as the changes in cluster memberships. No further changes need be made to B_kG, however. Any further changes to be made in the U-matrix are a consequence of the (weak) k-transitivity condition. That is, all elements u_{ab} will now be set

equal to d_{ij} if and only if two presently unconnected objects a and b (i.e. u_{ab} is presently undefined) are completely connected to a completely connected set of size $\geq k$. Each time an element of the U-matrix is defined this changes the ML-sets so that additional changes may become possible. Each such u_{ab} that is defined corresponds to two objects a and b being placed together in the same cluster for the first time by the merging of two B_k clusters. Hence all *new* clusters at level h_2 can be found by determining the clusters ML-sets for those pairs of objects corresponding to the changed elements of U. Due to the amount of effort required to determine all ML-sets at a given threshold level it is useful to note that all *new* ML-sets at the given threshold level $h_2 = d_{ij}$ must contain either objects i and j or some other pair of objects which are connected in U for the first time at level h_2 (any other pairs are either already in the same cluster or not yet in the same cluster). Note that if $k \leq 2$ then only a single cluster is formed at each level which must obviously contain both objects i and j. For $k = 1$ this is true since the only criterion for cluster membership is that the set be connected. For $k = 2$ it is true since a new cluster can only be formed if a new edge is added so that a larger ML-set can be found. But a new edge can only be added by the application of the (weak) k-transitivity condition and each edge must, of course, connect two objects. Thus any new ML-sets will be merged since they must have at least two objects in common. This means that a complete determination of *all* ML-sets defined by the connections in the U-matrix is not necessary at each distinct threshold level h. The procedure used in the algorithm is to define a list, L, of sets of objects for which ML-sets must be determined. Initially this list contains only $\{i,j\}$. For $k \geq 3$ every pair of objects (a, b) for which u_{ab} is updated at the current level h is added to this list [Step 4(a)]. As each pair is checked it is deleted from the list [Step 3]. Eventually no further changes to U are possible and L becomes empty [Step 6] and the processing is complete at the current threshold level.

Let (i, j) be the next set of objects in the list, L, described above. The procedure for finding ML-sets is based on the following:

If vertices i or j are of degree $<k$ or i and j are both of degree k (as defined by the connections in the U-matrix), then i and j cannot both belong to an ML-set of size $>k+1$ since each object in an ML set of size m must have $m-1$ connections. No further changes to U are then possible at level h_2. It is possible that some clusters should be merged but such mergers cannot result in any change to U. This must be true according to the definition of the (weak) k transitivity condition. Any new clusters formed must contain both objects i and j. Since i and j cannot have been members of the same cluster previously (since they were unconnected) the new clusters to be found must be the set of ML-sets which contain both i and j at the current threshold level. These can be found rather simply by the following procedure (see Step 4(a)): place objects i and j in a set C and delete this set from L. Find all objects which are completely connected to all objects in set C. For each of these objects form a new set consisting of that object and all of the objects in set C. Store each of these sets in list L (unless it is a subset of a previously found cluster). If no such objects are found then store set C in M (the list of clusters). Repeat this process for each of the sets in L until L is empty. The results in M are the ML-sets (clusters) at the current level which contain both i and j.

If i or j is of degree $>k$ and neither is of degree $<k$ then changes to the U-matrix are possible but only if i and j actually belong to at least one ML-set of size $k+1$. Some (but not all) of the new clusters need not contain either objects i or j. At this point one must consider locating all ML-sets and merging any with $\geq k$ vertices (in common with the resultant set made a maximal complete subgraph by making the necessary changes to the U-matrix) iteratively until no further changes are possible as described by Jardine and Sibson (1968b). The procedure given above for finding all ML-sets could be used here also. One need only add a procedure for merging all such sets which have k or more vertices in common. Whenever two clusters are merged elements of the U-matrix must be updated for all pairs of objects which are placed into the same cluster. The actual procedure used for finding and merging ML-sets is as follows:

The procedure described above to find ML-sets involving objects i and j is used initially [Step 4(a)] but once a set C is of size k (always the case if $k \leq 2$) the rules can be modified [Step 4(b)]. Any ML-sets which need to be merged with C can be found by simply adding to C all objects which have at least k-connections to the objects presently in set C. When an object, a, is added to this set any object, b, in the set to which the new object is not already connected is now connected by setting u_{ab} equal to d_{ij}. This procedure works since if objects i and j belong to an ML-set, C, of size $\geq k+1$ and if there exists an ML-set B to which it should be merged, then there must exist at least k objects in B each of which has at least k connections to the objects in C. The remaining (if any) objects in B must (by definition) each have at least k connections to these vertices belonging to both B and C. Thus the iterative procedure [Step 4(b)] of adding vertices to C which have at least k connections to objects presently in C will always result in the proper merging of sets B and C. If i and j belong to more than one ML set then this process must be performed for each (the order makes no difference). It is computationally important to note that these steps need not be carried out on any set of size $\geq k$ if it is a subset of a set resulting from the application above steps to a previously found ML-set since it would result in the same final ML-set [Step 5]. If there exists an ML-set D which has fewer than k objects in common with C then none of the objects in D (other than those which belong to both C and D) can have at least k connections to the objects in C. Thus the algorithm will correctly fail to merge sets C and D.

If all the dissimilarity values are not distinct then one must modify Step 2 in the algorithm so that if one finds a tie then one tests each of the edges to determine whether the corresponding u_{ij} entries have been defined. If they have, then these edges are ignored. If more than one edge remains, then they are all added to B_kG and to list L so that Steps 3 to 5 can be carried out for each edge (the order is irrelevant). The clusters formed at this level must be accumulated until all of the tied edges are processed at which point the maximal clusters can be printed.

Discussion

The proposed algorithm is sufficiently complicated that problems were encountered verifying its correctness. As a partial check the program was tested against the simpler but less efficient method of searching all sets of size $k+2$ described by Jardine and Sibson (1968a) for a large number of

Table 3 Average CPU time (in min. for an IBM 370/155) for various combinations of n (number of objects) and k. The time to read and/or generate the dissimilarity coefficients has been excluded.

	k 1	2	3	4	5
20	0·0025	0·0030	0·019	0·022	0·027
40	0·013	0·016	0·24	0·26	0·26
n 60	0·033	0·044	1·15	1·13	1·21
80	0·084	0·095	3·1	3·1	3·2
100	0·16	0·19	7·1	7·2	7·3

random matrices. In all cases the resulting U-matrices were identical.

It is now feasible to consider more extensive empirical testing of the usefulness of the B_k cluster techniques. Previously available algorithms were limited to about 60 objects due to the amount of computation involved for $k > 2$. Cole and Wishart's (1970) improved algorithm required 18 minutes to compute only the U-matrix for the sequence $k = 1, 2, \ldots, 5$ for 35 objects on an IBM 360/44. The present technique required only 0·5 minutes to compute both the U-matrix *and* the clusters on an IBM 370/155 (average for random matrices with no ties). It is difficult to realistically compare times on different computers, but the ratio of speeds is less than 36 to 1. **Table 3** furnishes average computational time (on an IBM 370/155) as a function of n for $k = 1, 2, \ldots, 5$. The large difference in time for $k = 2$ and $k = 3$ is due to the fact that for $k > 2$ a *much* larger number of sets must be tested as a result of going through Step 4(*a*). It should be emphasised that for the special case of $k = 1$ other algorithms are much more efficient. The fastest appears to be the method given in Rohlf (1973) which also produces B_1G (the minimum spanning tree) at the same time. The SLINK program of Sibson (1973) is only half as fast but may be useful if the matrix is so large that it must be kept on a sequential file. More efficient programs for other particular values of k can undoubtedly also be written.

As pointed out by Rohlf (1974), the fact that the program can output clusters as an automatic byproduct of the algorithm to find B_kG and U is a convenience but it is not essential to the application of the method. For large n the B_k clusters can be very complex unless $k = 1$ or 2. One may, therefore, not wish to attempt to construct a k-dendrogram. One can either report only those B_k clusters which are particularly distinct (large gaps) or depict the results by drawing in the edges in the B_kG on a principal components or (preferably) a non-metric multidimensional scaling analysis plot of the n objects. If the lengths of the edges are given then the clusters at any threshold level can usually be found quite easily by using the rules for determining cluster membership given above and in Rohlf (1974).

Acknowledgements

This work was supported in part by a grant GB-20496 (from The National Science Foundation). This paper is contribution number 52 of the program in Ecology and Evolution at the State University of New York at Stony Brook. Dr. J. S. Farris read the manuscript and made a number of suggestions which improved the efficiency of the program. Drs. N. Jardine and R. Sibson criticised various versions of the manuscript which stimulated its development to its present stage. The responsibility for any remaining errors rests, of course, with the author. A copy of the computer program (written in FORTRAN IV) is available upon request, both as an independent program and as part of the NTSYS system of programs.

References

Busacker, R. G., and Saaty, T. L. (1965). *Finite graphs and networks*, McGraw-Hill, New York, 294 pages.

Cole, A. J., and Wishart, D. (1970). An improved algorithm for the Jardine-Sibson method of generating overlapping clusters, *The Computer Journal*, Vol. 13, pp. 156-163.

Fisher, L., and Van Ness, J. W. (1971). Admissible clustering procedures, *Biometrika*, Vol. 58, pp. 91-104.

Gower, J. C., and Ross, G. J. S. (1969). Minimum spanning trees and single linkage cluster analysis, *Applied Statistics*, Vol. 18, pp. 54-64

Jardine, N., and Sibson, R. (1968a). A model for taxonomy, *Math. Biosciences*, Vol. 2, pp. 465-482.

Jardine, N., and Sibson, R. (1968b). The construction of hierarchic and non-hierarchic classifications, *The Computer Journal*, Vol. 11, pp. 177-184.

Jardine, N., and Sibson, R. (1971). *Mathematical taxonomy*, Wiley, New York, 286 pages.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the travelling salesman problem, *Proc. Am. Math. Soc.*, Vol. 7, pp. 48-50.

Prim, R. C. (1957). Shortest connection networks and some generalizations, *Bell System Technical Journal*, Vol. 36, pp. 1389-1401.

Rohlf, F. J. (1973). Hierarchical clustering using the minimum spanning tree, *The Computer Journal*, Vol. 16, pp. 93-95.

Rohlf, F. J. (1974). *Graphs implied by the Jardine-Sibson B_k, overlapping clustering methods, Journal American Statistical Association*, Vol. 69, pp. 705-710.

Sibson, R. (1973). SLINK: an optimally efficient algorithm for the single-link cluster, *The Computer Journal*, Vol. 16, pp. 30-34.