CHAPTER 4 - CONNECTIVITY CLUSTERING

4.1 INTRODUCTION

Connectivity clustering is particularly important in pattern recognition; and it immediately generalizes to spaces of dimension greater than 2. Rather than attempting to find "useful" clusters, as in the synoptic clustering problem, instead "intuitive" or "natural" patterns are to be analyzed.

Unlike the algorithms discussed in Chapter 3, the algorithms of this Chapter do not require a distance; any dissimilarity may be employed. As before, possible parallel implementations are discussed, and these will probably play an increasing role as new machine architectures become more widespread.

Section 4.2 focusses on the algorithms described in this Chapter, - the single link hierarchical clustering method and the minimal spanning tree (MST). This section gives an indication of where these two closely related methods may be most fruitfully used.

Section 4.3 gives background material on implementations of algorithms. It does not attempt to detail all algorithms which could be employed, but those that it does discuss are among the most efficient and subsequent sections will improve on the basic ideas behind them.

Section 4.4 details implementations of algorithms which may incorporate fast nearest neighbour searching algorithms (see Chapter 2). For general purpose applications, the algorithms of this section could probably be said to represent the most recommendable algorithms when efficiency is of paramount importance. Although the minimal spanning tree is focussed on in this Chapter, the final algorithm of section 4.4 is suitable for any graph (or linkage) method, - such as for example the complete link method.

Since the MST and single linkage methods work on dissimilarities, we may further consider the case when only some of the n(n-1)/2 possible (symmetric) dissimilarities are presented to the algorithm. The problem of such sparse graphs arises, for example, in constrained clustering (to be described in Chapter 5). Section 4.5 details an efficient algorithm for the sparse graph problem. It further discusses a special case of this problem when the graph represents a set of planar points or objects (i.e. the graph is said to be planar).

As with all clustering techniques, there are limits to the applicability of the MST or single linkage method for distinguishing patterns. One way to extend their applicability is to use information provided by the problem in addition to the basic idea of connectivity. Section 4.6 discusses a number of algorithms of this type, where the objective is to find modes or peaks in point density or some analogous measure (irrespective of pattern shape).

4.2 SINGLE LINK METHOD IN PERSPECTIVE.

Hierarchical clustering methods based on the geometric paradigm have been explored in Chapter 3. In this Chapter, methods based on graph-theoretic principles will be studied.

The general hierarchical clustering algorithm described in Chapter 1 allows $O(n^2)$ time and $O(n^2)$ space implementations of the single and complete linkage methods, and of the weighted and unweighted average linkage methods (see Table 1, Chapter 3 for the dissimilarity update formulas required for these methods). At all stages of the agglomerations, the results obtained so far may be graphically presented, without the need to plot cluster centres (see Sneath and Sokal, 1973, for a number of worked examples).

The single link method uses a very weak requirement for cluster formation. The complete link method is more restrictive, and produces hierarchies which are nearly as "balanced" as the minimum variance method (cf. section 3.2). An average linkage method might be preferred for noisy data, so that spurious clusters which might be produced by the over-lax single link method or the over-demanding complete link method can be avoided.

The single link method will be focussed on in this Chapter. Besides being the oldest hierarchical clustering method (it was initially used in the early 1950s), and one of the most widely-used hierarchical methods because of computationally efficient algorithms, it is also of great interest for point pattern recognition. A very wide range of algorithms have, in fact, been developed for the single linkage method: Rohlf (1982) reviews algorithms with complexities ranging from $O(n \log n)$ to $O(n^5)$. Many of these algorithms have first constructed the MST, and subsequently transformed this into the single link hierarchy.

These two problems - single linkage clustering and the MST - are closely related. Information is lost in transforming the MST into the hierarchy, so that the reverse transformation is not possible. Rohlf (1973) describes an efficient $O(n^2)$ worst case algorithm for transforming the MST into the hierarchy. It may be simply sufficient to sort the n-1 edges of the MST, thus providing a representation of the hierarchy (i.e. the sequence of agglomerations; it might be preferred to adopt a clustering labelling standard, such as to number clusters from n+1 to at most 2n-1; or to label clusters by the lowest index number of their object-members). To do this requires O(n log n) time.

The algorithms described in this Chapter will be based on the MST. Being an important structure itself, and being easily transformable into the hierarchic representation, both indicate the centrality of the MST from a practical stand-point.

The following properties apply to the single link hierarchical method, and to its associated minimal spanning tree.

(1) A number of authors (see for example Jardine and Sibson, 1971) have found the mathematical properties of this method so appealing that they have preferred it to all other hierarchical methods. Among properties not shared by other methods which they have pointed to are:
   - every partition has classes which are optimal with reference to the connectivity criterion used; partitions obtained from the minimum variance method, in contrast, are suboptimal with respect to within-cluster minimum variance.
   - Monotonic transformation of input dissimilarities (i.e. a transformation which preserves order) has no effect on the hierarchy; this may be of importance where a question is raised over the scaling of directly-constructed dissimilarities.

- Small changes in input dissimilarities produce small changes in the hierarchy produced, and thus the single link method is a stable method.

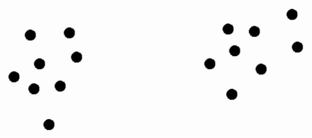
(2) In practice, the single link method has the chaining disadvantage which makes it particularly unsuitable for synoptic clustering.

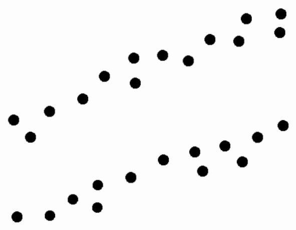
(3) However it is easy to graphically represent the results of linkage based clustering methods on 2-dimensional data; and the hierarchical method or the minimal spanning tree are very suitable for many types of pattern recognition problems.

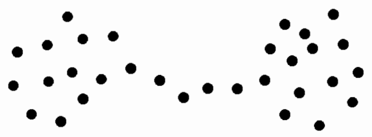
We will briefly look at problems in this latter area.

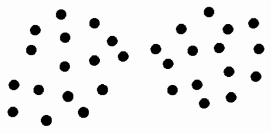
Machine vision includes pattern recognition and image processing. The automatic recognition of groups of points is an important problem in the former area. Such point patterns may be arrived at in different ways, - for example, by being derived from a digitized image with a considerably more complex background structure.

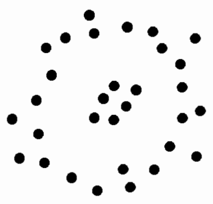
Practically all proposed clustering algorithms would perform well when presented with well-separated, compact groups (see Fig. 4.1). For elongated clusters, the minimum variance method would perhaps cut the clusters in two in its search for compactness (Wishart, 1969, shows such an example using astronomical data). The single link method - with its chaining effect - or the MST would be ideally suited instead. In the case of linked, globular clusters an estimate may be made of the density in the region of each point (e.g. the number of other points within a specified radius: this approach will be taken in mode analysis, below). This will indicate which points form part of the interconnecting links, constitute noise points, or are otherwise to be ignored. For touching globular groups, the


Well-separated, compact groups.


Elongated clusters.


Linked globular groups.


Touching globular groups.


Concentric groups.

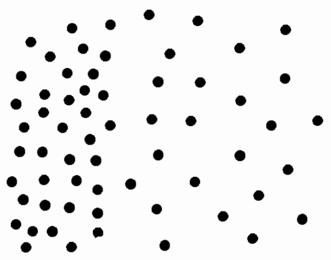

Groups characterised by differing densities.

Fig. 4.1 - Point patterns whose constituent groups are to be automatically recognized.

MST may not be of direct use. The minimum variance method should, however, find the clusters, and mode analysis (see below) may also be profitably applied here. Finally, in the cases of concentric groups, or of groups characterised by differing densities, the MST may be used. In the former case, a large link will indicate where the MST ought to be broken in order to leave the two components. In the latter case, a histogram of edge dissimilarity weights should uncover two distinguishable sets of dissimilarities: edges of small dissimilarity will relate to the high density part of the point pattern, and edges of greater dissimilarity will relate to the low-density region. Deleting all edges of dissimilarity weight greater than some thresold in the MST causes the resultant tree to connect only high-density points. Similarly, the low-density component may be ascertained. Note, though, that some extra treatment may be required for points on the boundary of the two regions in order to avoid misclassification (see Zahn, 1971).

Recent research has branched into two directions. On the one hand, a "shortest spanning path" (i.e. a path, spanning all points, which is as short as possible in totalled dissimilarities) achieves many of the same results as does the MST, but with greater computational ease: see Slagle et al. (1974), Slagle et al. (1975) and Lee (1981). On the other hand, other graph theoretical structures have been used with which the MST may be related as a special case: see Jarvis and Patrick (1973), Urquhart (1982), Sibson (1980), Ahuja (1982) and Fairfield (1983). Although these approaches are of importance and offer advantages in specific problems, the MST remains of great interest as a general-purpose technique.

4.3 TRADITIONAL MINIMAL SPANNING TREE ALGORITHMS

The algorithm for the single linkage hierarchical clustering described in Chapter 1 required $O(n^2)$ storage space for the dissimilarity matrix, and $O(n^2)$ processing - $O(n)$ iterations, each necessitating $O(n)$ updating of the dissimilarity matrix. These performance results may be considered as baseline results. They appear to be very satisfactory since the input string presented to the algorithm is $O(n^2)$ long. However they may be bettered, without detriment to the exactness of the output, using algorithms described in sections 4.4 and 4.5.

The algorithms described in this section may be regard as alternatives to that described in Chapter 1; and they construct a MST rather than directly building the single link hierarchy. Their performance results do not improve on the performance of the algorithm looked at in Chapter 1. However they are of particular importance for two reasons. Firstly, the computationally efficient algorithms described in section 4.4 (probably the most efficient, general-purpose, current algorithms) are directly inspired from these algorithms. Secondly, it may be necessary to construct a MST or a single link hierarchy, on a sparse graph. This is a graph where the number of edges, m, is less than n(n-1)/2 and so a performance result is desired in terms of m and n. In Chapter 5, application-areas for such a problem will be described. Again, for this problem, efficient algorithms which are described in section 4.5 of this chapter are direct derivations of the algorithms now described.

In section 4.4 a single fragment algorithm allowing for the incorporation of fast NN-finding techniques will be studied. The Prim-Dijkstra algorithm, by comparison, involves brute-force NN-searching. It is assumed that the algorithm will work on the stored matrix of dissimilarities, requiring $O(n^2)$ space.

Algorithm A. Prim-Dijkstra MST algorithm

```
Step 1.  Find the closest vertex to an arbitrary vertex. Call these two
         vertices a fragment of the MST.
Step 2.  Determine the closest vertex, not in the fragment to any vertex
         in the fragment.
         Add this vertex to the fragment.
Step 3.  If all n vertices are not included in the fragment then return
         to Step 2.
```

There are O(n) iterations (Steps 2,3) in this algorithm since a MST must contain n-1 edges. Ordinarly Step 2 will require $O(n^2)$ operations, leading to an overall complexity of $O(n^3)$. An $O(n^2)$ implementation may be achieved as follows. For each vertex not in the fragment, maintain the closest vertex to it which is in the fragment. Initially $O(n^2)$ operations are required for this. On each iteration (Step 2), as a vertex (say, v) is added to the fragment, check to see if any nearest neighbour (among fragment members) of a vertex outside the fragment can now be bettered by v. This requires checking O(n) values, but allows the implementation of Step 2 in O(n) time.

The foregoing algorithm grew a single fragment through n-1 iterations. As an alternative approach, it is often fruitful to attempt to build the desired structure in a parallel fashion, - by allowing more than one fragment which will be subsets of the eventual MST. Sollin's algorithm does this, again assuming prior calculation of all dissimilarities (edge weights) which requires $O(n^2)$ or $O(m)$ operations. (In section 4.4 a multiple fragment algorithm bypassing this requirement will be described). An example of this algorithm is to be seen in Fig. 4.2.

Consider all vertices, to begin with, as singleton fragments of the MST.

Algorithm B. Sollin's MST algorithm.

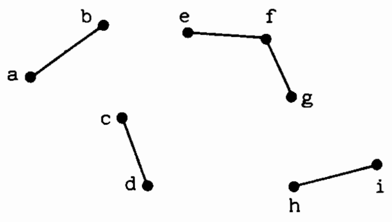
```
Step 1.  For each fragment in turn, determine its closest fragment.
         All edges so specified will be part of the MST.
Step 2.  Scan each of these edges, deleting the corresponding fragments
         from the list of fragments, and placing the new merged fragment
         onto the fragment-list.
Step 3.  While the fragment-list has more than one member, return to
         Step 1.
```

The closeness relation in Step 1 is not necessarily symmetric: in fact we have here a generalization of NN-graphs (see Chapter 3), where a RNN pair (symmetric) or part of a NN-chain (asymmetric) may be found among fragments. These least cost edges between fragments, determined in Step 1, must be part of the MST. This may be shown as follows. By construction we have minimal cost, but do we have a tree? If a cycle were possible we would have, for example, f_2 (fragment 2) as NN to f_1, f_3 as NN to f_2, and f_1 as NN to f_3. But then the least interconnecting link between f_2 and f_1 is greater than the least interconnecting link between f_3 and f_2 (otherwise f_1 would be NN to f_2), which in turn is greater than the analogous link between f_3 and f_1. Here we have a contradiction since by definition the link between f_1 and f_2 is less than that between f_3 and f_1. This proof may be extended to cycles with more than 3 vertices. Note that care must be taken in programming this algorithm to incorporate arbitrary choice-making in the case of equal dissimilarities, in order to avoid the creation of cycles.

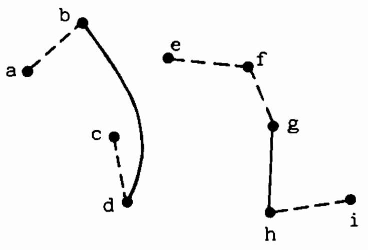
In Step 1 there are O(n) operations for each of n vertices. At most $\lceil n/2 \rceil$ (the least integer $\geq n/2$) fragments are produced. On the next pass through Step 1, $O(n^2)$ operations will be again required, leading to at most $\lceil n/4 \rceil$ fragments. Continuing, it is seen that there are O(log n) iterations yielding $O(n^2 \log n)$ performance.

$$\begin{array}{c|ccccccccc|}
 & a & b & c & d & e & f & g & h & i \\
\hline
a & 0 & 1 & 5 & 6 & 6 & 5 & 7 & 5 & 6 \\
b & 1 & 0 & 5 & 3 & 6 & 8 & 5 & 7 & 5 \\
c & 5 & 5 & 0 & 2 & 5 & 6 & 8 & 6 & 6 \\
d & 6 & 3 & 2 & 0 & 5 & 6 & 4 & 7 & 5 \\
e & 6 & 6 & 5 & 5 & 0 & 2 & 3 & 6 & 7 \\
f & 5 & 8 & 6 & 6 & 2 & 0 & 1 & 5 & 5 \\
g & 7 & 5 & 8 & 4 & 3 & 1 & 0 & 3 & 6 \\
h & 5 & 7 & 6 & 7 & 6 & 5 & 3 & 0 & 1 \\
i & 6 & 5 & 6 & 5 & 7 & 5 & 6 & 1 & 0 \\
\hline
\end{array}$$

Given matrix of dissimilarities between 9 vertices.



Using the lightest edges incident on each vertex, the above four fragments are found. The set of lightest edges from these fragments yield:



The final lightest edge between fragments connects d to g yielding MST.

<u>Fig. 4.2</u> – Sollin's algorithm for constructing a MST.

In the case of a sparse graph, incident edges may be stored as linked lists, or as some other convenient data structure. Step 1 will require a scan of all edges on each occasion, i.e. $O(m)$ operations. Thus the performance in this case is $O(m \log n)$.

4.4 MINIMAL SPANNING TREE USING FAST NEAREST NEIGHBOUR SEARCHING

In Chapter 3, single and multiple cluster algorithms for geometric clustering methods have been described, which allowed for the incorporation of fast NN searching routines. In this section, suggested approaches to single link hierarchical clustering will be discussed. These approaches are based on the MST (which may be subsequently transformed into the hierarchy).

The following proposition has been used in Chapter 3:

Proposition 1: Given a point-set, any pair of RNNs is a class or cluster of an agglomerative hierarchic clustering, if the hierarchic clustering method satisfies the reducibility property.

The single link method satisfies this property, but a stronger result is:

Proposition 2: Any NN-chain is a subset of MST.

Any NN-chain, originating in an arbitrary point and ending in a pair of RNNs, therefore defines in reverse order a sequence of nested clusters in a single linkage hierarchic clustering. Instead of single and multiple cluster algorithms for geometric cluster methods, we have here single and multiple fragment algorithms, where a fragment is "grown" from a NN-chain. The following is a single fragment algorithm.

Algorithm C. MST algorithm using NN-chains.

Step 1. Construct a NN-chain; let q be the last point added, and call the NN-chain a fragment.

Step 2. Find r, the nearest point to q which is not in the fragment.

Step 3. If, for some i in the fragment, $d(i,NN(i)) < d(q,r)$ then see if there is an s not in the fragment such that

$$d(i,NN(i)) < d(i,s) < d(q,r).$$

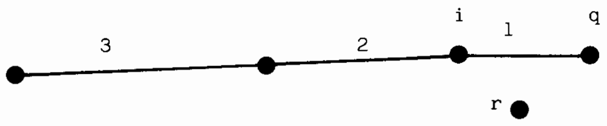
If so, find the least such $d(i,s)$, and connect s to i; otherwise connect r to q.

Step 4. Redefine q to be the point whose link to the fragment is of least dissimilarity, and return to Step 2 until all points are in the fragment.

Step 3 is explained as follows (cf. Fig. 4.3). The point r could be connected to q. However it must be checked if a closer point could instead be connected to some other point in the fragment. The edge $(i,NN(i))$ is in the fragment if i is in the fragment. If $d(i,NN(i)) > d(q,r)$, and if some s is connected to i then $d(i,s) < d(q,r)$, which together imply: $d(i,s) < d(i,NN(i))$. But then s is the NN of i, and from this contradiction it is seen that we were justified in connecting r to q.

Step 3 possibly necessitates "climbing back" some way in the fragment (i.e. it requires the finding of NNs of a number of vertices in the fragment) and is best implemented using a list of vertices in the fragment, ordered by the smallest dissimilarity which connects them to the fragment. The analysis of this algorithm depends on the average number of iterations between Steps 2 and 3, - i.e. if constant or $O(n)$, we get overall complexity of $O(n^2)$ or $O(n^3)$, respectively. An approximate minimal spanning tree algorithm, where this number of iterations is held constant, may be adequate but in general worst-case $O(n)$ must be assumed.

A better, multiple fragment algorithm with two separate stages (Steps 1, 2, and 3; and Steps 4 and 5) is as follows.



Final two nodes of fragment are i and q.

NN(i) = q.

d(i,q) = 1.

Closest node to q, not in fragment, is r.

d(q,r) = 1.9.

Search for closest node to i, not in fragment.

Find r: d(i,r) = 1.8.

Therefore s (see description of Algorithm C) is q, and q is connected to i.

Fig. 4.3 - Example of Algorithm C.

Algorithm D. Parallel MST algorithm.

Step 1. Pick an arbitrary point.

Step 2. Construct a NN-chain from this point.

Step 3. Pick another isolated point, and return to Step 2 until all points are in one of p NN-chains.

Step 4. Connect the closest point in an arbitrary NN-chain (or fragment) to some other NN-chain (or fragment), using Steps 2, 3 and 4 of Algorithm C.

Step 5. Return to Step 4 until all points are in one fragment.

Algorithm D will work better if the NN-chains are long, i.e. if the points chosen in Steps 1 and 3 are in sparse regions. Following the iterated Steps 2 and 3, there are p NN-chains and hence p-1 edges remaining to be found in the minimal spanning tree.

The principal computational advantage of Algorithms C and D lies in their ability to incorporate efficient NN-finding techniques. The latter algorithm has been found to be of O(n log n) average complexity, when a MDBST approach (see Chapter 2) is employed (Bentley and Friedman, 1978).

We will conclude this section with an adaptation of Algorithm C which is suitable for any graph (or linkage) method (e.g. the complete or average linkage methods described in Table 1, section 3.3). For these methods, inter-cluster dissimilarity cannot be calculated in O(1) time unless we have the entire set of dissimilarities directly accessable. Algorithm C may be amended as follows.

Algorithm E. Algorithm for any linkage-based method.

Step 1. Select an object arbitrarily.

Step 2. Determine and store all inter-object dissimilarities.

Step 3. Grow the NN-chain from the object chosen, until a pair of RNNs are obtained.

Step 4. Agglomerate these objects.

Step 5. Update the dissimilarity table, using the Lance-Williams formula.

Step 6. From the node in the NN-chain which preceded the RNNs, or from an arbitrary node (object or cluster) if the NN-chain is empty, go to Step 3 until only one node remains.

Clearly, $O(n^2)$ storage is required here. As before, there are $O(n)$ growths of the NN-chain, each requiring $O(n)$ operations; plus $O(n)$ contractions of the NN-chain, each requiring $O(n)$ updating of the dissimilarity table. In total, therefore, computational complexity is $O(n^2)$.

4.5 MINIMAL SPANNING TREE OF SPARSE AND PLANAR GRAPHS

Special cases of the MST problem arise when not all edges exist. In a sparse graph, the number of edges **may not be $O(n^2)$. The** first algorithm discussed below has computational complexity $O(m \log n)$. Thus this result comes close to $O(m)$, and m might be quite small. The number of edges is small in the case of planar graphs: the second algorithm discussed in this section achieves the very satisfactory computational performance of $O(n)$. These algorithms are due to Cheriton and Tarjan (1976).

Let us begin with the problem of sparse graphs where we seek the best performance in terms of n and of m (bearing in mind that the latter will always be greater for non-degenerate problems). The following algorithm is a particular implementation of Sollin's algorithm (see section 4.3, Algorithm B).

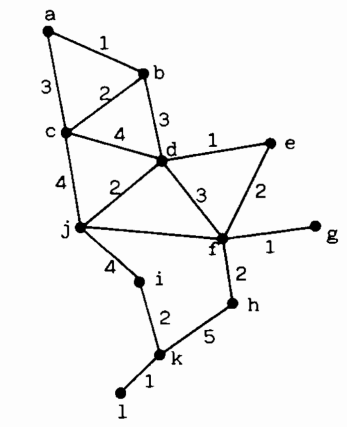
For each vertex v in the vertex-set V, let the number of incident edges be denoted by n_v . Thus, $\sum_{v \in V} n_v = 2\,m$

A preprocessing stage to Algorithm B is as follows. Divide each set of n_v edges into groups of size k ($\leq$ k for the final group: for simplicity, we will assume that every group contains precisely k edges). Sort each of these groups. For each vertex, we have n_v/k groups, and the sort operation will require $O(k \log k)$ comparisons. Hence $(n_v/k)\, k \log k$ will be the order of magnitude of the number of operations required for vertex v. For all vertices, the number of operations required is of the order of $\sum_{v \in V} n_v \log k = 2\, m \log k$.

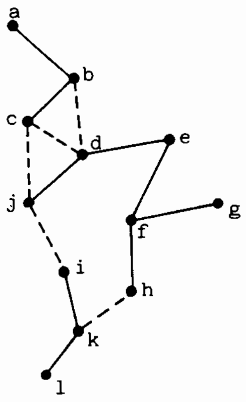
Algorithm B is now implemented as follows. Determining the lightest edge incident on a vertex requires $O(n_v/k)$ comparisons since this edge is to be found in some one of the n_v/k sorted groups. The lightest edges incident on all vertices are therefore obtained in $O(m/k)$ operations. When two vertices (later: fragments) are merged, their associated sorted groups of incident edges are simply appended together so that all such groups remain internally sorted. On subsequent executions of Step 1 in Algorithm B, again $O(m/k)$ processing is required. (Note that we may discard from all sorted groups those edges connecting vertices in the same fragment: in the entire algorithm, this cannot surpass the deleting of 2 m edges). Thus, overall, Algorithm B's complexity becomes $O(m/k \log n) + O(m \log k)$ where the latter term, as was seen above, is required for preprocessing. By chosing $k = \log n$, the second of these terms dominates and gives complexity $O(m \log \log n)$ for this implementation.

We turn attention now to constructing a MST from a planar graph. The following graph-theoretic result will be of central importance in the analysis of an algorithm for building the MST of a planar graph: for a planar graph, $m = O(n)$. Specifically, $m \leqslant 3n-6$ for $m > 1$. For proof, see for example Tucker (1980).

Referring to Sollin's algorithm (Algorithm B of section 4.3), $O(n)$ operations are required to establish the least cost edge from each vertex (since there are only $O(n)$ edges present). On the subsequent execution of Step 1, we may define a new, planar graph with as new vertices the fragments found so far (see Fig. 4.4). There will be at most $\lceil n/2 \rceil$ such new vertices. $O(n/2)$ processing will serve to replace multiple edges between the same pair of (new) vertices with the minimum edge weights, since the total number of multiple edges remains $O(n)$. Following this, $O(n/2)$ processing is required to merge fragments (Step 1 of Algorithm B). Continuing, we obtain the overall complexity as being $O(n)+O(n/2)+O(n/4)+ + \ldots = O(n)$.



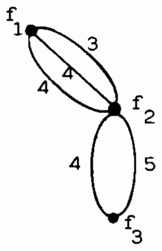
Given planar graph.



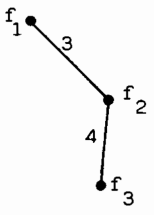
Heavy lines show fragments found in first iteration of algorithm.

Clean up of deletable edges has been carried out.

Dashed lines are edges connecting fragments.



New graph consisting of three fragments.



New graph with multiple edges removed.

Fig. 4.4 - Stages in an implementation of Sollin's algorithm on a planar graph.

4.6 EXTENSION: MODE ANALYSIS

The MST (and the single link method) use dissimilarities or distances. One difficulty with distance-based procedures is that any regularity in the data may give rise to many identical distances and to subsequent degenerate or misleading cluster results. This difficulty can be bypassed by incorporation of local (e.g. density) information, which will indicate which points are of greater importance. In general, we may use

- node weights: valuations on the points under consideration; or
- node and edge weights, where interpoint dissimilarities are additionally used.

We will begin with algorithms in the first category.

Node weights used in pattern recognition have generally involved an estimate of density at each point. Among such node weights are the following:

1) $|N(i)|$ where $N(i) = \{ j \mid d_{ij} \leqslant r \}$

   N(i) is the neighbourhood of point i, defined here as the set of points within radius r of i. The weight of node i is the cardinality of its neighbourhood.

2) $1/k \quad \Sigma \{ d_{ij} \mid j \in N(i) \}$

   where N(i) is the set of k nearest neighbours of i. The weight of node i, here, is the average distance to the k-nearest neighbours; it is a measure of <u>potential</u>, i.e. the inverse of density. Unlike the previous node weight, it is independent of the scaling of the original data.

3) In image processing nodes corresponding to pixels may be weighted by

the grey level intensity at that point. This is the most immediate node weighting scheme. Others may be specified, though, such as a measure of edge gradient at that pixel. The edge value is the difference in intensities between contiguous pixels; and edge gradient is the maximum such value between a pixel and its neighbours. The edge gradient may be useful for contour extraction, i.e. for determining significant boundaries.

The use of dissimilarity d, in the above, is almost invariably Euclidean, - the most natural choice for visual patterns of points. Let f_i be the weight associated with node i, using some one of the above definitions.

The most straightforward approach to the clustering of node-valued graphs is to use a single threshold: nodes of density-weight greater (or potential-weight less) than the threshold are members of the same cluster, if they are in addition contiguous to at least one other member of the cluster. By decreasing the threshold in the case of densities, or by increasing it in the case of potential, a hierarchy of embedded classes is obtained.

The clustering brought about by thresholding can also be expressed in terms of more traditional distance-based clustering. Define $\delta_{ij} = \infty$ if i and j are not contiguous; otherwise define $\delta_{ij} = -\min\{f_i, f_j\}$ where f is a density. As values of f are examined in increasing order of magnitude, i will be connected to j only if both f_i and f_j are greater than the density threshold. This clustering method may therefore be viewed as a single link method. Clustering by thresholding in the manner described is a common technique in image processing; and it has also been used for wealth data for geographic regions (i.e. f_i = per capita income for region i; see Hartigan, 1975).

The use of node weights (such as point densities, attributes of populations or states, etc.) is an intuitively clear starting point from which to carry out the automatic grouping process. But the use of inter-point distances, while being fraught with difficulty when many distances are identical, nonetheless allows a more fine-tuned analysis: for example, in the threshold-based clustering described above, no account is taken as to whether a new addition to a cluster is closely related to one or to many of the cluster members. In order to allow for varying degrees of relationship, a dissimilarity may be recreated from the node weights. One possibility for this is to construct directed arcs defined by $\delta_{ij} = f_j - f_i$ where i and j are contiguous. Therefore if $f_j > f_i$ then δ_{ij} is directed from i to j, while if $f_j < f_i$ then the arc is negatively weighted, and so is directed from j to i. Rather than the difference in densities, as this dissimilarity coefficient is, the density gradient (difference in density per unit distance) has usually been preferred (see e.g. Koontz et al., 1976). This is given by $\delta_{ij} = (f_i - f_j)/d_{ij}$ where d is the Euclidean distance, and δ is again an asymmetric dissimilarity. A generalization of the single linkage method (or the MST) has been used for such dissimilarities. It is to connect i to j if δ_{ij} is positive and maximum among nodes j which are contiguous to i, i.e. to construct components such that the density gradient is always upwards.

It is easily verified that each such component is a directed tree, so long as no δ_{ij} equals zero. In order to facilitate subsequent labelling and other processing of the components, cycles in the directed graph must be prevented, and arbitrarily directed edges are formed for $\delta_{ij} = 0$ following a test that a cycle will not result. Note that in this approach each component nominates a unique "centre" or local peak in density. It has also been proposed that local valleys in density are equally revealing of structure in the data (see Johnston et al., 1979). Such an alternative viewpoint on the data may be carried out by simply defining δ_{ij} as the negative of the mode-oriented approach.

A similar approach - determining components which are directed trees - has been used in image processing (see Fig. 4.5). Narendra and Goldberg (1980) define as a weight at each pixel (node) a measure of edge gradient. Having thus a value for f_i, the asymmetric dissimilarity δ_{ij} is constructed and the directed tree formed in the manner described above.

Another very different application of this directed forest approach has also been successfully employed, as follows. A histogram of intensities often permits visually different parts of the image to be distinguished, - different modes in the histogram correspond to distinct, but significantly numerous, sets of pixel intensities. The gradient climbing procedure, used in point pattern recognition, also allows the modes of the histogram to be determined (see Fig. 4.6). Smoothing of the histogram might be required - using for instance a 3-point moving average - and Wharton (1983) suggests an "adaptive smoothing" where non-mode parts of the histogram (below a user-specified threshold value) alone are smoothed in this way. For 4-band LANDSAT data, a 4-dimensional generalization of this approach has been employed by constructing a 4-dimensional histogram. This is simply a grid of regular cells in 4-dimensional space, each containing the frequency of occurrence of associated 4-valued pixel intensity vectors.

Note that the dissimilarity constructed in the foregoing examples has been anti-symmetric (i.e. $\delta_{ij} = - \delta_{ji}$). Other asymmetric (but not anti-symmetric) coefficients may also be constructed for point pattern recognition. For instance, Ozawa (1983) defines $\delta_{ij} = f_i \cdot \exp(-b \cdot d_{ij})$ where b is some scale constant. This dissimilarity will yield different values

$$\begin{tikzcd}
25 \arrow[r] & 35 \arrow[r] & 48 \arrow[r] & 52 \arrow[d] \\
20 \arrow[u] & 24 \arrow[r] & 46 \arrow[r] & 53 \\
22 \arrow[r] & 23 \arrow[u] & 18 \arrow[u] & 20 \arrow[d] \\
21 \arrow[u] & 20 \arrow[r] & 24 \arrow[r] & 55
\end{tikzcd}$$

Directed trees

1	1	1	1
1	1	1	1
1	1	1	2
1	2	2	2

Class labels

Fig. 4.5 - Clustering by connecting pixels to highest-valued pixel among four-neighbours.

$$\begin{tikzcd}
5 \arrow[r] & 17 \arrow[r] & 35 \arrow[r] & 83 & 41 \arrow[l] & 22 \arrow[l] & 3 \arrow[l] & 3 \arrow[r] & 5 \arrow[r] & 25 \arrow[r] & 39 \arrow[r] & 53 & 22 \arrow[l] & 11 \arrow[l]
\end{tikzcd}$$

Fig. 4.6 - Clustering by gradient climbing to distinguish modes of a histogram (given here by the set of frequencies).

for δ_{ij} and for δ_{ji} depending on density defined at i and at j. Katz and Rohlf (1973) use another asymmetric coefficient.

This brief look at mode analysis indicates the range of approaches in an area which continues to expand rapidly. Many of the approaches described required local processing to establish the node or edge weights, and the connectivity clustering was then performed with reference only to the neighbourhood of the point (or pixel or other object). Thus these algorithms would appear to be well-suited to parallel implementations. A problem to be solved in certain cases concerns the improvisation required to arbitrarily choose among equally-valued dissimilarities.

4.7 REFERENCES

N.AHUJA, Dot pattern processing using Voronoi neighbourhoods. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4, 336-343 (1982).

J.L. BENTLEY and J.H. FRIEDMAN, Fast algorithms for constructing minimal spanning trees in coordinate spaces. IEEE Transactions on Computers C-27, 97-105 (1978).

D. CHERITON and R.E. TARJAN, Finding minimum spanning trees. SIAM Journal of Computing 5, 724-742 (1976).

J. FAIRFIELD, Segmenting dot patterns by Voronoi diagram concavity. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5, 104-110 (1983).

J.A. HARTIGAN, Clustering Algorithms,Wiley, New York (1975).

N. JARDINE and R. SIBSON, Mathematical Taxonomy, Wiley, New York (1971).

R.A. JARVIS and E.A. PATRICK, Clustering using a similarity measure based on shared near neighbours. IEEE Transactions on Computers C-22, 1025-1034 (1973).

B. JOHNSTON, T. BAILEY and R. DUBES, A variation on a nonparametric clustering method. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1, 400-408 (1979).

J.O. KATZ and F.J. ROHLF, Function-point cluster analysis. Systematic Zoology 22, 295-301 (1973).

W.L.G. KOONTZ, P.M. NARENDRA and K. FUKUNAGA, A graph-theoretic approach to nonparametric cluster analysis. IEEE Transactions on Computers C-25, 936-944 (1976).

R.C.T. LEE, Clustering analysis and its applications. In Advances in Information Systems Science, Edited by J.T. Tou, Vol. 8, pp. 169-292, Plenum Press, New York (1981).

P.M. NARENDRA and M. GOLDBERG, Image segmentation with directed trees. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2, 185-191 (1980).

K. OZAWA, CLASSIC: a hierarchical clustering algorithm based on asymmetric similarities. Pattern recognition 16, 201-211 (1983).

F.J. ROHLF, Algorithm 76: hierarchical clustering using the minimum spanning tree. The Computer Journal 16, 93-95 (1975).

F.J. ROHLF, Single link clustering algorithms. In P.R. Krishnaiah and L. N. Kanal (Editors), Handbook of Statistics Vol.2, pp. 267-284, North-Holland, Amsterdam (1982).

R. SIBSON, The Dirichlet tesselation as an aid in data analysis. Scandinavian Journal of Statistics 7, 14-20 (1980).

J.R. SLAGLE, C.L. CHANG and R.C.T. LEE, Experiments with some cluster analysis algorithms. Pattern Recognition 6, 181-187 (1974).

J.R. SLAGLE, C.L. CHANG and S.R. HELLER, A clustering and data-reorganizing algorithm. IEEE Transactions on Systems, Man, and Cybernetics SMC-15, 125-128 (1975).

P.H.A. SNEATH and R.R. SOKAL, Numerical Taxonomy. Freeman, San Francisco, 1973.

A. TUCKER, Applied Combinatorics. Wiley, New York, 1980.

R. URQUHART, Graph theoretical clustering based on limited neighbourhood sets. Pattern Recognition 15, 173-187 (1982); Erratum, Pattern Recognition 15, 427 (1982).

D. WISHART, Mode Analysis: a generalization of nearest neighbour which reduces chaining effects. In Numerical Taxonomy, Edited by A.J. Cole, Academic Press, London, pp. 272-281 (1969).

S.W. WHARTON, A generalized histogram clustering scheme for multidimensional image data. Pattern Recognition 16, 193-199 (1983).

C.T. ZAHN, Graph-theoretical methods for detecting and describing Gestalt clusters. IEEE Transactions on Computers C-20, 68-86 (1971).