Discussion Paper
No. 90
Center for Spatial Information Science
University of Tokyo

COMPUTATIONAL METHOD FOR THE POINT CLUSTER ANALYSIS ON NETWORKS

Kokichi Sugihara, University of Tokyo
Atsuyuki Okabe, University of Tokyo
Toshiaki Satoh, PASCO Corp.
July 8, 2008

Computational Method for the Point Cluster Analysis on Networks

Kokichi Sugihara Atsuyuki Okabe Toshiaki Satoh
University of Tokyo PASCO Corp.

Abstract

We present a general framework of hierarchical methods for point cluster analysis on networks, and then consider individual clustering procedures and their time complexities defined by typical variants of distances between clusters. The distances considered here are the closest-pair distance, the farthest-pair distance, the diameter distance, the average distance, the median-pair distance, the radius distance and the generalized radius distance.

1 Introduction

We develop a hierarchical cluster analysis for points on a network. Stated a little more explicitly, we formulate computational methods for finding the hierarchical structure of clusters consisting of point-like events that occur on or along a network. Typical applications are to find 'hot spots' of purse snatches on streets and the spatial concentration of fashionable stores in downtown streets. In this paper we concentrate on hierarchical approaches, although nonhierarchical approaches might be possible.

Generally, a cluster analysis classifies events (alternatively referred to as objects, data units, subjects, cases or elements (Anderberg, 1973, p. 11)) into a set of clusters according to some 'similarity' among the events. In the ordinary cluster analysis, the attributes of an event are represented by a vector of variables in a space, termed the *attributes vector* in the *attribute space*, which is usually multidimensional Euclidean space. An event is represented by a point, called an *event point*, in the attribute space, and the distance between event points, called the *event distance*, is given by a function of the attribute vectors of the two event points. The event distance is often given by the Euclidean distance between two event points in the attribute space. A cluster is represented by a set of points in the attribute space, and the distance between two clusters, referred to as the *cluster distance* (or *cluster similarity*) is given by a function of the attribute vectors of the event-points forming the two clusters. There are various kinds of cluster distances and this variety allows many variations in cluster analysis (Anderberg, 1973, Chapter 6).

Cluster analysis on networks is conceptually similar to ordinary cluster analysis but it has the following three features distinct from the ordinary cluster analysis.

The first distinct feature is that the attribute variables of an event include the location variables of the event, that is, the coordinates of the incidence spot of the event on a network

embedded in a plane. In a general cluster analysis on networks, the attribute variables of an event consist of location variables and nonlocation variables.

Consequently, the attribute space of the general cluster analysis on networks is a hybrid space consisting of the physical distance space associated with the location variables and the nonphysical distance space associated with the nonlocation variables. This makes the treatment of the event distance and the cluster distance very complex. Because of the distinct nature of cluster analysis on networks, this paper focuses on the special case in which the only attribute variables of an event are the coordinates of the incidence location of the event on the network. This implies that the cluster analysis on networks to be developed in this paper is the cluster analysis of points in the physical space, i.e., the space formed by a network embedded in a plane. The general case will be developed elsewhere, based on the results in this paper.

The second distinct feature is that the event distance is measured by the length of the shortest path between the points on a network, and the cluster distance is defined as a function of the shortest-path distances between the points forming the clusters. As with ordinary cluster analysis, the function can be defined in various ways and this variety brings many variations in the cluster analysis on networks. We develop several such distances.

One might argue that the distance between event points can be measured by Euclidean distance, even though the events occur on a network. This is appropriate in some cases, but not in others. The choice depends on what events are to be analyzed. Consider, for example, the incidence spots of purse snatches on streets in a downtown area. Because purse snatchers commit on streets and run away along streets, it is more appropriate to use the shortest-path distance.

Another example is spatial clusters of retail stores in an urbanized area. Well-known spatial clusters are the diamond rows on Manhattan Island, New York, and the used book rows in Kanda, Tokyo. Because shoppers access these stores through streets, the spatial clusters are formed along the streets. Shoppers feel that a used book row on the opposite bank of a river is a long way from a used book row on this bank.

These examples suggest that if events are formed by activities achieved through networks, it is more appropriate to use the shortest-path distance rather than the Euclidean distance. In the real world, there are many events that occur on a network. Examples are car crashes on streets, road-kills of animals on forest roads, urban crime sites, tree spacing along the roadside, seabirds located along a coastline, beaver lodges built in a watercourse, levee crevasse distribution on riverbanks, leakages in gas and oil pipelines, breaks in a wiring network, and disconnections on the Internet. In addition to these events, another broad class of events that can be represented by the distribution of points on a network is the class of events that occur alongside rather than directly on a network, such as facilities located alongside street networks within densely inhabited areas. In fact, the entrances to almost all facilities in a city are adjacent to a street and users access amenities through these. Consequently, the distribution of almost all facilities within an

urbanized area can be represented by the distribution of points on networks. Therefore, cluster analysis on networks has many potential applications.

The third distinct feature of cluster analysis on networks is that it is oriented to geometric computation. The cluster distances in the ordinary cluster analysis are determined by algebraic operations on attribute vectors, and the computational method is fairly simple. On the other hand, the cluster distances in cluster analyses on networks are determined through the computation of shortest-path distances between event points. This computation requires knowledge of the topology of the network. Therefore, the computational method is completely different from the ordinary cluster analysis. At first glance, the extension from ordinary cluster analysis to the point cluster analysis on network seems to be easily achieved by replacing Euclidean distance with the shortest-path distance, but as will be shown in this paper, this extension is not so easy. We develop efficient computational methods for various cluster distances on a network.

As mentioned above, cluster analysis on networks has great potential demand, but its development has lagged. There are some reasons for this lag. First, the availability of network data and location data of point-like events (e.g., the incidence spots of accidents) was low. Second, the management and operation of network data was difficult. However, in recent years these difficulties have been overcome by the online web service of disaggregated spatial data and geographical information systems (GISs). The time is now ripe for developing cluster analysis on networks, but few papers can be found in the literature. Motivated by the great potential demand and lack in research, we attempt to develop cluster analysis on networks.

This paper consists of thirteen sections. A framework for the point cluster analysis on networks is given in Section 2, and a general algorithm and its time complexity are presented in Section 3. Section 4 remarks on a special case where all the vertices of the network are included in the set of event points. In Sections 5 to 11, individual distances to measure the similarity between clusters are investigated; the distances discussed there are the closest-pair distance, the farthest-pair distance, the diameter distance, the average-pair distance, the median-pair distance, the radius distance, and the generalized radius distance. In Section 12, the possibility of an extension of Ward's method to networks is discussed, and concluding remarks are given in Section 13.

2 Hierarchical Clustering

Let $N = (V, L)$ be a network having vertex set V and link set L. Each element of L is a size-two subset of V. Thus, for each $l \in L$, we can write $l = \{v, v'\}$, $v, v' \in V$, and call l a *link* connecting v and v'. We make the following assumption:

Assumption 1 (planarity). The network N is planar.

That is, we can draw the network N on a plane in such a way that the elements of V are placed at distinct points and the elements of L are represented by curved line segments connecting the

associated two vertices without mutual intersection.

Throughout this paper, we use the following notation. For a finite set X, $|X|$ denotes the number of elements in X, and for two sets X and Y, $X\backslash Y$ denotes the set of all the elements that belong to X but do not belong to Y.

Next, let P be a set of prespecified points on the network. An element of P may be a vertex in V or a point on a link in L. We are interested in constructing the cluster structure of the points in P.

Suppose that $p \in P$ is a point in the interior of a link $l \in L$, that is, p is a point on l but is not a terminal point of l. We divide the link $l = \{v, w\}$ at p into two links $l' = \{v, p\}$ and $l'' = \{p, w\}$ and consider p as a new vertex incident to l' and l''. Thus the network N is changed by deleting the link l, adding l' and l'' as new links and adding p as a new vertex. We repeat this division procedure for all points in $P\backslash V$, and denote the resulting network by $\overline{N} = (V \cup P, \overline{L})$, where $\overline{L}$ is the set of all links obtained from L by dividing the links at points in $P\backslash V$. We call $\overline{N}$ the *extended network* of N with respect to P.

In the following, we concentrate on the extended network $\overline{N} = (V \cup P, \overline{L})$, and consider the cluster analysis of the vertices in P. We call the elements of P *event vertices*, while we call the elements $V \cup P$ just *vertices*. Note that $\overline{N}$ is also planar.

Throughout this paper we denote the number of elements in P by n, that is, $|P| = n$.

We assume the following.

Assumption 2. $|V \cup P| = \mathrm{O}(n)$.

This assumption implies that the number of event vertices is large when compared with the size of V. We make this assumption only to simplify the discussion of the time complexity. If V is much larger than P, the following complexity discussion is kept valid by redefining n as $n = |V \cup P|$.

Because the network is planar, the number of elements in $\overline{L}$ is also of $\mathrm{O}(n)$.

Let us name the event vertices as $v_1, v_2, \ldots, v_n$, that is, $P = \{v_1, v_2, \ldots, v_n\}$. Suppose that for any pair of disjoint subsets $X, Y \subset P$, the nonnegative real function $d(X, Y)$ is defined; $d(X, Y)$ is called the *distance* between X and Y. We will consider various definitions of $d(X, Y)$ in later sections, but at present we treat $d(X, Y)$ as a black box, ignoring its internal structure.

We can construct a hierarchical structure for the points in P in the following way. Initially for each $v \in P$, the size-one subset $\{v\}$ is regarded as a cluster. Hence, there are n clusters $\{v_1\}, \{v_2\}, \ldots, \{v_n\}$; we consider the set $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ of clusters as the level-0 structure. Next, we search for the minimum distance between two clusters. Suppose that we find

$$d(\{v_i\}, \{v_j\}) = \min_{\{v\},\{v'\} \in C_0} d(\{v\}, \{v'\}), \tag{1}$$

that is, the distance between the two clusters $\{v_i\}$ and $\{v_j\}$ is the shortest. Then, we merge $\{v_i\}$ and $\{v_j\}$ into one larger cluster $\{v_i, v_j\}$, and consider the resulting set $C_1 = \{\{v_1\}, \ldots, \{v_{i-1}\},$

$\{v_{i+1}\}, \ldots, \{v_{j-1}\}, \{v_{j+1}\}, \ldots, \{v_n\}, \{v_i, v_j\}\}$ of clusters as the level-1 structure. Hence the level-1 structure C_1 consists of $n-1$ clusters. An example with 6 event vertices is shown in Fig. 1. At the bottom of this figure, there are six nodes, each of which corresponds to a cluster consisting of one vertex; they are considered as the level-0 cluster set. Suppose that the distance between $\{v_4\}$ and $\{v_5\}$ is the smallest. Then, $\{v_4\}$ and $\{v_5\}$ are merged into $\{v_4, v_5\}$, which is represented by a new node placed one higher level that has $\{v_4\}$ and $\{v_5\}$ as two child nodes.

The level 2 and higher structures are constructed in a similar way. Suppose that we have already obtained the level-k structure consisting of $n-k$ clusters $C_k = \{X_1, X_2, \ldots, X_{n-k}\}$, $X_i \cap X_j = \emptyset$, $X_1 \cup X_2 \cup \cdots \cup X_{n-k} = P$. Then, we find the pair of clusters X and X' that admits the minimum distance:

$$d(X, X') = \min_{Y, Y' \in C_k} d(Y, Y'). \tag{2}$$

We merge the two clusters X and X' into the larger one $X \cup X'$, and consider the resulting set $C_{k+1} = (C_k \backslash \{X, X'\}) \cup \{X \cup X'\}$ as the level $k+1$ structure. Thus in our example, the binary tree shown in Fig. 1 is constructed.

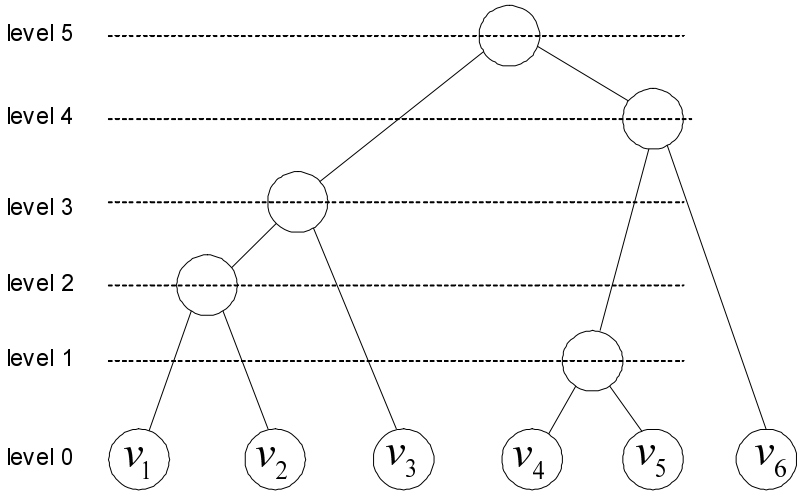

Figure 1. Binary tree structure representing hierarchical clustering.

As we go up the level of structure by one, the number of clusters decreases by one. Hence, the level $n-1$ structure consists of the single cluster P. Therefore, by this procedure we can construct the sequence of n cluster sets, $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}, C_1, \ldots, C_{n-1} = \{P\}$. This procedure is called *hierarchical clustering*, and the resulting sequence of cluster sets the *hierarchical structure of the clusters*.

At each step of the construction of the hierarchical structure, we merge the closest two clusters into one. Hence it is natural to think that the distance between the two clusters we merge in a lower level is smaller than that in a higher level. To guarantee this property, we make the next assumption.

Assumption 3 (monotonicity of the distance). For $i = 1, 2, \ldots, n-1$, the following property is satisfied. Suppose that at the ith level of clustering, two clusters $X_1, X_2 \in C_{i-1}$ are merged into $X_1 \cup X_2$. Then, for any other cluster $Y \in C_{i-1} \backslash \{X_1, X_2\}$, the following inequality holds:

$$\min\{d(X_1, Y), d(X_2, Y)\} \leq d(X_1 \cup X_2, Y). \tag{3}$$

Assumption 3 implies the following. Suppose that X_1, X_2 and Y are as stated in Assumption 3. Because the closest pair is merged, we find:

$$d(X_1, X_2) \leq \min\{d(X_1, Y), d(X_2, Y)\}. \tag{4}$$

From inequalities (3) and (4), it follows that the merged cluster $X_1 \cup X_2$ has larger distances to other clusters than $d(X_1, X_2)$. Therefore, Assumption 3 guarantees that the distances between two clusters that are merged is monotone nondecreasing as we move from lower levels to upper levels of the hierarchy.

Intuitively, the clusters are subsets of event vertices that are mutually closer. Because we have the network structure, it is natural to expect that clusters are connected substructures of the network. This is the most remarkable property of cluster analysis on networks when compared with the general cluster analysis. To guarantee this property we impose the following restriction.

Let X be a subset of P. X is said to be *intra-cluster connected in* P, if any two vertices in X are connected by a path in $\overline{N}$ that does not contain any vertex in $P \backslash X$. In other word, we delete all the event vertices other than X and the links incident to them from the network $\overline{N}$, and define the connectedness of X in the remaining network.

For example, consider the network $\overline{N} = (V \cup P, \overline{L})$ shown in Fig. 2, where the filled circles represent the event vertices (i.e., elements of P) and the empty circles represent nonevent vertices (i.e., elements of $V \backslash P$). Let X be the subset of the event vertices surrounded by the broken closed curve in the figure. In $\overline{N}$, the two vertices in X are connected by the unique path passing through the central vertex. In Fig. 2(a), the central vertex does not belong to P, and hence X is intra-cluster connected. On the other hand, in Fig. 2(b), the central vertex belongs to P, and hence X is not intra-cluster connected.

On the basis of these definitions, we make the following rule.

Rule 1. At any level of clustering, two clusters X and Y can be merged only when $X \cup Y$ is intra-cluster connected in P.

Let us say that two clusters X and Y are *adjacent* if $X \cup Y$ is intra-cluster connected in P.

Under Rule 1, it is only necessary to examine adjacent pairs of clusters to find the closest pair. However, we should note that the number of adjacent pairs can be as large as $O(n^2)$ even though the network is planar. This can be understood by the example shown in Fig. 3. The

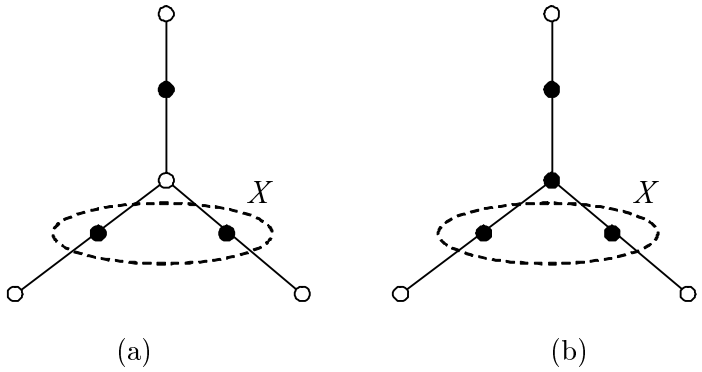

Figure 2. An intra-cluster connected subset and a disconnected subset.

network shown in Fig. 3(a) has a central vertex and five peripheral vertices connected to the central vertex. Suppose that all the peripheral vertices belong to P, but the central vertex does not. Then, at the initial stage of clustering, the five peripheral vertices form individual clusters and they are mutually adjacent. If we make a graph with vertices representing the clusters and edges representing the adjacency relation in Fig. 3 (a), we obtain the complete graph shown in Fig. 3(b). We can extend this example to a network with n peripheral vertices, in which the number of adjacent pairs of clusters is as large as $\mathrm{O}(n^2)$.

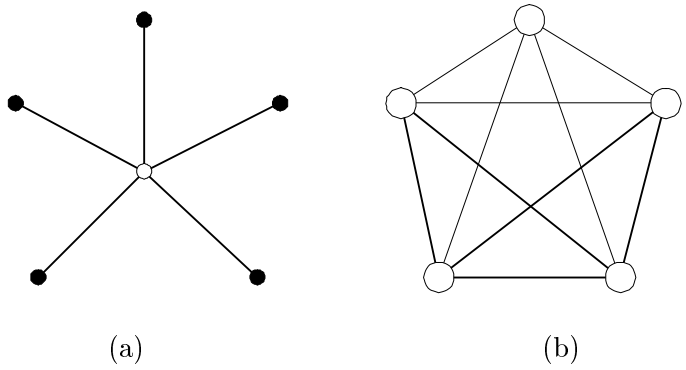

Figure 3. Example of a network in which there are as many as $\mathrm{O}(n^2)$ adjacent pairs of clusters.

3 A General Algorithm and Its Time Complexity

The general algorithm for the hierarchical clustering can be described in the following way.

Algorithm 1 (network hierarchical clustering in general)
Input: Network $N = (V, L)$, set $P = \{v_1, v_2, \ldots, v_n\}$ of event vertices, and the distance function d.

Output: Hierarchical structures of the clusters of the event vertices.

Procedure:

1. Construct the extended network $\overline{N} = (V \cup P, \overline{L})$ by dividing the links at the event vertices.
2. Initialize the cluster set as $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$.
3. Compute the distances $d(X, Y)$ for all adjacent pairs of clusters, and store them in storage Q.
4. Choose the pair (X, Y) with the minimum distance from Q and delete it, and do the following.
   - 4.1. Merge X and Y into $X \cup Y$, and put
     $C_i \leftarrow (C_{i-1} \backslash \{X, Y\}) \cup \{X \cup Y\}$.
   - 4.2. For all $Z \in C_{i-1} \backslash \{X, Y\}$ adjacent to X or Y, delete pairs (X, Z) and (Y, Z) from Q if they are in Q.
   - 4.3. For all $Z \in C_{i-1} \backslash \{X, Y\}$ adjacent to $X \cup Y$, compute $d(X \cup Y, Z)$ and insert the new pairs $(X \cup Y, Z)$ to Q.
5. If Q is empty, stop. Otherwise go to Step 4. □

Suppose that, for an adjacent pair of $X, Y \subset P$ of clusters, the value of the distance $d(X, Y)$ can be computed in time $t(n)$. On the basis of this general setting, here we consider the time complexity of Algorithm 1. In later sections we will break down the procedure and improve this time complexity for individual variants of the distance function.

To represent the hierarchical structure of the clusters, we use the binary tree data structure, as shown in Fig. 1. At each construction of one higher level, we generate a new node and connect it with the associated two clusters as its child nodes. Hence, the update of the data structure can be done in constant time, so we concentrate on the time necessary to find the pair of clusters with the minimum distance.

Steps 1 and 2 take $\mathrm{O}(n)$ time. In Step 3, we must compute the distances of adjacent pairs, of which there can be as many as of $\mathrm{O}(n^2)$. Hence Step 3 requires $\mathrm{O}(n^2\, t(n))$ time. In Step 4, the pair with the minimum distance is found in $\mathrm{O}(n^2)$ time. The merge procedure in Step 4.1 can be done in constant time. The deletion of irrelevant pairs from Q in Step 4.2 requires $\mathrm{O}(n)$ time. The number of new pairs to be inserted in Q in Step 4.3 can be as large as $\mathrm{O}(n)$, and hence Step 4.3 requires $\mathrm{O}(n\, t(n))$ time. Step 5 can be done in constant time. Because Steps 4 and 5 are repeated $n - 1$ times, the total time to execute Steps 4 and 5 is of $\mathrm{O}(n^2\, t(n))$.

If we use a priority queue as the data structure for the storage Q, the selection of the minimum-distance pair can be reduced from $\mathrm{O}(n^2)$ to $\mathrm{O}(\log n)$ [Aho et al. 1974]. However, this algorithmic technique does not help in decreasing the time complexity, because Step 4.3 dominates the computational cost.

Thus we obtain the next theorem.

Theorem 1. For a general distance function $d(X, Y)$ requiring $t(n)$ time for evaluation, the total hierarchical structure of clusters can be constructed in $\mathrm{O}(n^2\, t(n))$ time.

4 Planar Case

As we have seen in Fig. 3, the number of adjacent pairs of clusters can be as large as $O(n^2)$ in the worst case. However, if all the vertices of the original network N are event vertices (i.e., $V \subset P$), the planarity of the network is preserved when the adjacent vertices are merged into clusters. Hence in this case the number of adjacent pairs is bounded by $O(n)$. We call the network $\overline{N} = (V \cup P, \overline{L})$ is *pure* if $V \subset P$; in this case $V \cup P = P$ and hence $\overline{N}$ can also be written as $\overline{N} = (P, \overline{L})$.

Suppose that $\overline{N}$ is pure. Then the number of adjacent pairs of clusters is of $O(n)$, and hence Step 3 of Algorithm 1 is executed in $O(nt(n))$ (instead of $O(n^2 t(n))$), and the minimum-distance pair can be found in Step 4 in $O(n)$ time (instead of $O(n^2)$). Therefore, one might expect that the total time complexity of Algorithm 1 can be reduced.

However, this does not happen. A counterexample is given in Fig. 4. Suppose that cluster X in Fig. 4(a), is adjacent to all the other clusters, and that X is merged with another cluster Y. Then, as shown in Fig. 4(b), the new cluster $X \cup Y$ is adjacent to all the other clusters, and hence Step 4.3 of Algorithm 1 still requires $O(n\,t(n))$ time. Thus the total time complexity does not decrease even if $\overline{N}$ is pure.

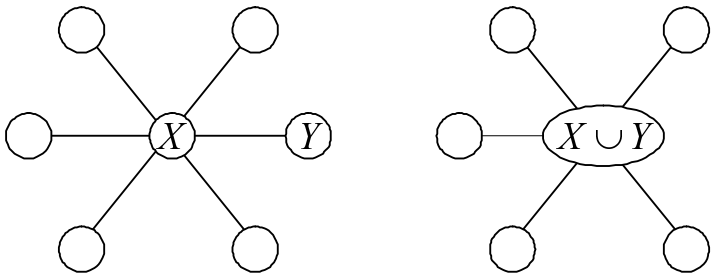

Figure 4. An example in which $O(n)$ new pairs arise when two clusters are merged.

For some of the specified distance functions, on the other hand, the time complexity decreases when we move from a general network to a pure network, as will be seen later.

5 Closest-Pair Distance

From now on, we consider individual distance functions, and the associated hierarchical clustering and its time complexity. The first distance function we consider is the closest-pair distance, which is a counterpart of the one used in the nearest-neighbor method (also called the single-linkage method) in ordinary cluster analysis [Johnson 1967, Gower and Ross 1969, Zahn 1971].

For two vertices $v, v' \in V \cup P$, let $D(v, v')$ represent the shortest-path distance between them in the network $\overline{N}$. For mutually disjoint subsets $X, Y \subset P$, let $d_1(X, Y)$ be defined as the

minimum distance between a vertex in X and a vertex in Y:

$$d_1(X,Y) = \min_{v \in X, v' \in Y} D(v,v'), \tag{5}$$

The distance function $d_1(X,Y)$ is called the *closest-pair distance*. This distance satisfies Assumption 3.

For a network with n vertices and e links, the shortest-path distances from one vertex to all the other vertices can be computed in $O(e \log n)$ time by Dijkstra's algorithm [Aho et al. 1974]. Because the network $\overline{N}$ contains $O(n)$ vertices and $O(n)$ links, the shortest-path distances from one vertex to all the others can be computed in $O(n \log n)$ time, and consequently the shortest-path distances between all the vertices can be computed in $O(n^2 \log n)$ time. Let X and Y be mutually disjoint subsets of P. Note that X and Y contain at most $O(n)$ vertices. Hence, using Dijkstra's algorithm, we can compute the shortest-path distances between all pairs of vertices in $X \cup Y$ in $O(n^2 \log n)$ time, and thus can compute the closest-pair distance by taking the minimum distance among all pairs with one in X and the other in Y. That is $t(n) = O(n^2 \log n)$. Therefore, from Theorem 1, we see that the time complexity for hierarchical clustering is of $O(n^4 \log n)$. However, we can decrease the time complexity drastically in the following way.

From the extended network $\overline{N} = (V \cup P, \overline{L})$, we construct another network $\widetilde{N} = (P, \widetilde{L})$ with the vertex set P, where $\widetilde{L}$ consists of links connecting all pairs of vertices in P. For each pair of vertices in P, we assign the shortest-path distance between them in $\overline{N}$ as the length of the associated link in $\widetilde{L}$.

A subnetwork $\widetilde{N}' = (P, \widetilde{L}')$ of the network $\widetilde{N} = (P, \widetilde{L})$, where $\widetilde{L}' \subset \widetilde{L}$, is called a *spanning tree* of $\widetilde{N}$ if $\widetilde{N}'$ is connected and acyclic. A spanning tree $\widetilde{N}'$ is called the *minimum spanning tree* if the sum of the lengths of the edges in $\widetilde{L}'$ is minimum among all the spanning trees of $\widetilde{N}$.

The minimum spanning tree can be constructed using the following procedure. We start with an empty set $\widetilde{L}_0$ of links, and the associated subnetwork $\widetilde{N}_0 = (P, \widetilde{L}_0)$ of $\widetilde{N}$. $\widetilde{N}_0$ consists of n connected components, each of which corresponds to a vertex in P. Next, we choose the link l with the minimum length from $\widetilde{L}$, and set $\widetilde{L}_1 = \widetilde{L}_0 \cup \{l\}$, obtaining the next subnetwork $\widetilde{N}_1 = (P, \widetilde{L}_1)$, which consists of $n-1$ connected components. We repeat similar procedures. That is, in the ith step, we choose the link l with the minimum length from $\widetilde{L}$ that connects two different connected components of $\widetilde{N}_{i-1} = (P, \widetilde{L}_{n-1})$, and set $\widetilde{L}_i = \widetilde{L}_{i-1} \cup \{l\}$, obtaining the ith subnetwork $\widetilde{N}_i = (P, \widetilde{L}_i)$. We repeat this procedure $n-1$ times. This gives us the subnetwork $\widetilde{N}_{n-1} = (P, \widetilde{L}_{n-1})$ with $n-1$ links and one connected component, which is the minimum spanning tree of $\widetilde{N}$.

Note that this construction procedure coincides with the construction of the hierarchical structure of the clustering; thus the ith-level cluster set corresponds to the set of connected components of the ith subnetwork $\widetilde{N}_i = (P, \widetilde{L}_i)$. Hence, the hierarchical clustering with respect to the closest-pair distance can be done by constructing the minimum spanning tree of $\widetilde{N}$.

It is known that the minimum spanning tree of a network with m links can be constructed in

$O(m \log m)$ time [Aho et al. 1974]. Because our network $\widetilde{N}$ contains $O(n^2)$ links, the hierarchical structure of the clusters with respect to the closest-pair distance can be constructed in $O(n^2 \log n)$ time.

If the extended network $\overline{N}$ is pure, we can further decrease the time complexity. Assume that $\overline{N}$ is pure. Then we can write $\overline{N}$ by $\overline{N} = (P, \overline{L})$. In this case we can use the minimum spanning tree of $\overline{N}$ instead of $\widetilde{N}$. Indeed, the procedure for constructing the minimum spanning tree of $\overline{N}$ coincides with the construction of the hierarchical structure of clusters of P. Because $\overline{N}$ contains $O(n)$ links, the minimum spanning tree can be constructed in $O(n \log n)$ time. Thus, the hierarchical structure of clusters can be constructed in $O(n \log n)$ time.

In the above procedure for constructing the minimum spanning tree, two connected components are connected into a larger component by a link (but not by a path consisting of two or more links). This implies that the two clusters with the shortest distance are always adjacent. Even if we do not impose Rule 1, the clustering procedure obeys Rule 1 automatically.

Although the closest-pair distance gives an efficient method for hierarchical clustering, the resulting cluster structure is sometimes far from our intuition of clusters. An example of such a situation is depicted in Fig. 5, where the network consists of one path and the lengths of links increase from right to left. In this case, the ith-level cluster set consists of the cluster of the rightmost i vertices and the remaining isolated vertices. Therefore, the rightmost cluster contains two vertices with a shortest-path distance that is much larger than the distance between neighboring isolated vertices. This does not agree well with what we want to call clusters.

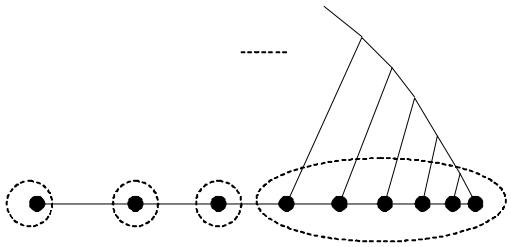

Figure 5. Hierarchical clustering with respect to the closest-pair distance.

Hence we consider other distances, although the associated hierarchical clustering is not so efficient as that with the closest-pair distance.

6 Farthest-Pair Distance

The second distance we consider is a network counterpart of the one used in the farthest-neighbor method (also called the complete-linkage method) in ordinary cluster analysis [Johnson 1967].

For mutually disjoint subsets $X, Y \subset P$, let $d_2(X, Y)$ be defined as the maximum shortest-

path distance in the network $\overline{N}$ between a vertex in X and a vertex in Y:

$$d_2(X,Y) = \max_{v \in X, v' \in Y} D(v,v'). \tag{6}$$

This distance $d_2(X,Y)$ is called the *farthest-pair distance* between X and Y. This distance satisfies Assumption 3.

Unlike the closest-pair distance, Rule 1 plays an essential role for the farthest-pair distance. Consider the network shown in Fig. 6. Suppose that we repeatedly merge the two clusters with the minimum farthest-pair distance. After the third repetition, we obtain five clusters, three of which consist of two vertices as shown by the broken closed curves in Fig. 6(a). We continue merging the two clusters with the minimum distance two more times, resulting in three clusters X, Y, Z as shown in Fig. 6(b). We see $d_2(X,Y) = 19, d_2(Y,Z) = 19, d_2(X,Z) = 18$, and hence if we do not impose Rule 1, the nonadjacent pair X and Z will be merged in the next step. Thus, Rule 1 is necessary.

As we have seen, the shortest-path distances between all pairs of vertices can be computed in $\mathrm{O}(n^2 \log n)$ time by Dijkstra's algorithm. Hence, the time required to compute the farthest-pair distance is $t(n) = \mathrm{O}(n^2 \log n)$. Therefore, direct application of Theorem 1 leads to the time complexity of $\mathrm{O}(n^4 \log n)$ for the hierarchical clustering.

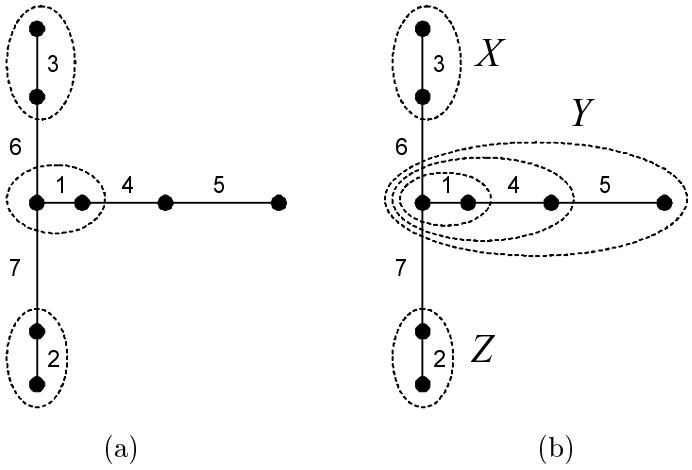

Figure 6. A network for which Rule 1 plays an essential role.

However, we can decrease this time complexity by the following algorithm.

Algorithm 2 (Clustering with the farthest-pair distance)

Input: Network $\overline{N} = (V \cup P, \overline{L})$

Output: Hierarchical structure of the clusters of vertices in P with respect to the farthest-pair distance.

Procedure:

1. Compute the shortest-path distances of all pairs of vertices in P.

2. Sort the shortest-path distances in increasing order.
3. Initialize the cluster set $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$.
4. For $i = 1, 2, \ldots, n-1$ do the following.
   4.1. Find the pair (v, v') such that
      (i) v and v' belong to different clusters $X, Y \in C_{i-1}$,
      (ii) all the pairs with one vertex in X and the other in Y have shortest-path distances not greater than $d_2(v, v')$, and
      (iii) $d_2(v, v')$ is the minimum among all pairs that satisfy the properties (i) and (ii).
   4.2. Merge the two clusters X and Y into $X \cup Y$, and name the resulting cluster set C_i. □

Algorithm 2 constructs the hierarchy of clusters with respect to the farthest-pair distance. This can be understood in the following way. Let (v, v') be the pair chosen at Step 4.1 in Algorithm 2. Because this pair satisfies properties (i) and (ii), the shortest-path distance between v and v' coincides with the farthest-pair distance between X and Y; i.e., $d_2(X, Y) = D(v, v')$. Moreover, the pair (v, v') satisfies the property (iii), and hence realizes the shortest distance between two clusters in C_{i-1}. Hence, the merge process of Step 4.2 coincides with the construction of the ith-level cluster set.

Now let us consider the time complexity of Algorithm 2. Step 1 can be done in $O(n^2 \log n)$ time by applying Dijkstra's algorithm n times. Step 2 can be done in $O(n^2 \log n)$ time by a standard sorting method. Step 3 can be done in $O(n)$ time. To find the pair (v, v') in Step 4.1, we use a counter $c(X, Y)$ for each pair of clusters X and Y. Initially $c(X, Y)$ is set to 0. We examine pairs (v, w) of vertices in increasing order of shortest-path distances, and increment the counter $c(X, Y)$ by one for the pair of clusters such that $v \in X$ and $w \in Y$. Because the number of pairs between X and Y is $|X| \times |Y|$, we can recognize that we encounter the pair (v, v') specified by the properties (i), (ii) and (iii) in Step 4.1 when the value of the counter $c(X, Y)$ reaches $|X| \times |Y|$.

It is not necessary to clear the counters $c(X, Y)$ at the beginning of each repetition of Step 4.1, because we can reuse the values of the counters in the earlier stages. Initially all the clusters are singletons, and hence we define $c(\{v_i\}, \{v_j\}) = 0$ for all pairs $v_i, v_j \in P$. Suppose that at one repetition of Steps 4.1 and 4.2 we merge two clusters X and Y into $X \cup Y$. At that time we define, for the other clusters $Z \in C_{i-1} \backslash \{X, Y\}$, new counters by

$$c(X \cup Y, Z) = c(X, Z) + c(Y, Z). \tag{7}$$

Therefore, for each pair (v, w) of vertices, the properties (i), (ii) and (iii) are checked in constant time, and here all the repetitions of Step 4.1 require time proportional to the number of pairs of vertices; i.e., $O(n^2)$. As we already saw, the merge process in Step 4.2 is done in constant time. Consequently, Step 4 is done in $O(n^2)$ time. Thus in total, Algorithm 1 runs in $O(n^2 \log n)$, which is a great improvement from the $O(n^4 \log n)$ for the general method.

Note that this argument on time complexity does not change even if $\overline{N}$ is pure. This is because the planarity of $\overline{N}$ is already employed in evaluating the computational cost of Dijkstra's algorithm.

7 Diameter Distance

For $X \subset P$, we define $\mathrm{Diam}(X)$ as follows:

$$\mathrm{Diam}(X) = \max_{u,v \in X} D(u,v), \tag{8}$$

and call it the *diameter* of X. The diameter is the shortest-path distance between the farthest pair of vertices in X.

Next, for mutually disjoint subsets $X, Y \subset P$, let $d_3(X,Y)$ be defined as the diameter of $X \cup Y$:

$$d_3(X,Y) = \mathrm{Diam}(X \cup Y), \tag{9}$$

which is called the *diameter distance*.

Suppose that we construct the hierarchical structure of clusters with respect to the diameter distance. In that case, if we merge two clusters X and Y into $X \cup Y$ in some step of the clustering, the inequality

$$\max\{\mathrm{Diam}(X), \mathrm{Diam}(Y)\} < d_3(X,Y) \tag{10}$$

holds. In this context, the diameter $d_3(X,Y)$ is realized by a pair of vertices, one in X and the other in Y, and hence the diameter distance coincides with the farthest-pair distance.

Therefore, the clustering with respect to the diameter distance coincides with the clustering with respect to the farthest-pair distance and, consequently, Algorithm 2 constructed in the last section can be used for the diameter distance.

8 Average-Pair Distance

The next distance we consider is a network counterpart of the one used in the average-linkage method in ordinary cluster analysis [Lance and Williams 1967].

Let $X \subset P$ be a subset of event vertices. We define $\mathrm{AD}(X)$ by

$$\mathrm{AD}(X) = \frac{2}{|X|(|X|-1)} \sum_{v,w \in X} D(v,w). \tag{11}$$

That is, $\mathrm{AD}(X)$ is the average of the shortest-path distances of all the pairs of vertices in X. We call $\mathrm{AD}(X)$ the *average diameter* of X.

For mutually disjoint subsets $X, Y \subset P$, we define

$$d_4(X,Y) = \mathrm{AD}(X \cup Y). \tag{12}$$

That is, $d_4(X,Y)$ is the average of the shortest-path distances of all pairs of vertices in $X \cup Y$. We call $d_4(X,Y)$ the *average-pair distance* between X and Y.

Because X and Y contain $\mathrm{O}(n)$ vertices, the number of possible pairs of vertices in X and those in Y is $\mathrm{O}(n^2)$, and the shortest-path distances from one vertex in X to all the vertices in Y can be computed in $\mathrm{O}(n \log n)$ time by Dijkstra's algorithm, the average-pair distance between X and Y can be computed in $\mathrm{O}(n^2 \log n)$ time. Therefore, from Theorem 1, the hierarchical structure of the clusters can be computed in $\mathrm{O}(n^4 \log n)$ time.

We can decrease this time complexity in the following way. As we have already seen, the shortest-path distances for all pairs of vertices in V can be computed in $\mathrm{O}(n^2 \log n)$ time. In the initialization stage, we compute the shortest-path distances $D(v,w)$ for all $v, w \in V$, and set $d_4(\{v\},\{w\}) = D(v,w)$. Furthermore, we set $\mathrm{AD}(\{v\}) = 0$ for all $v \in V$. Suppose that at the ith iteration of clustering, we merge two clusters $X, Y \in C_{i-1}$ into $X \cup Y$. At this step, the average diameter of the new cluster $X \cup Y$ is obtained as:

$$\mathrm{AD}(X \cup Y) = d_4(X,Y). \tag{13}$$

For the average-pair distances between the new cluster $X \cup Y$ and another cluster $Z \in C_{i-1} \backslash \{X, Y\}$, we obtain:

$$\begin{aligned}
& d_4(X \cup Y, Z) = \mathrm{AD}(X \cup Y \cup Z) \\
= \; & \frac{2}{|X \cup Y \cup Z|(|X \cup Y \cup Z| - 1)} \left\{ \sum_{v,w \in X} D(v,w) + \sum_{v,w \in Y} D(v,w) + \sum_{v,w \in Z} D(v,w) \right. \\
& \left. + \sum_{v \in X, w \in Y} D(v,w) + \sum_{v \in X, w \in Z} D(v,w) + \sum_{v \in Y, w \in Z} D(v,w) \right\} \\
= \; & \frac{2}{|X \cup Y \cup Z|(|X \cup Y \cup Z| - 1)} \left\{ \left(\sum_{v,w \in X} D(v,w) + \sum_{v,w \in Z} D(v,w) + \sum_{v \in Y, w \in Z} D(v,w) \right) \right. \\
& + \left(\sum_{v,w \in Y} D(v,w) + \sum_{v,w \in Z} D(v,w) + \sum_{v \in Y, w \in Z} D(v,w) \right) \\
& + \left(\sum_{v,w \in X} D(v,w) + \sum_{v,w \in Y} D(v,w) + \sum_{v \in X, w \in Y} D(v,w) \right) \\
& \left. - \sum_{v,w \in X} D(v,w) - \sum_{v,w \in Y} D(v,w) - \sum_{v,w \in Z} D(v,w) \right\} \\
= \; & \frac{2}{|X \cup Y \cup Z|(|X \cup Y \cup Z| - 1)} \left\{ \frac{|X \cup Z|(|X \cup Z| - 1)}{2} d_4(X,Z) \right. \\
& + \frac{|Y \cup Z|(|Y \cup Z| - 1)}{2} d_4(Y,Z) + \frac{|X \cup Y|(|X \cup Y| - 1)}{2} d_4(X,Y) \\
& \left. - \frac{|X|(|X| - 1)}{2} \mathrm{AD}(X) - \frac{|Y|(|Y| - 1)}{2} \mathrm{AD}(Y) - \frac{|Z|(|Z| - 1)}{2} \mathrm{AD}(Z) \right\}.
\end{aligned} \tag{14}$$

We already know the average diameters $\mathrm{AD}(X)$, $\mathrm{AD}(Y), \mathrm{AD}(Z)$, and the distances $d_4(X,Y)$,

$d_4(X,Z), d_4(Y,Z)$, so the distance between the new cluster $X \cup Y$ and another cluster $Z \in C_{i-1}\backslash\{X,Y\}$ can be computed in constant time by eq. (14). Hence the distances between clusters can be updated in $O(n)$ time at each level of merging.

Thus we have the following algorithm.

Algorithm 3 (Clustering with the average-pair distance)
Input: Network $N = (V, L)$, and event vertex set $P = \{v_1, v_2, \ldots, v_n\}$.
Output: Hierarchical structure of the clusters with respect to the average-pair distance.
Procedure:

1. Initialize the cluster set as $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$.
2. For all $v \in V$, put $\mathrm{AD}(\{v\}) = 0$.
3. Construct the extended network $\overline{N} = (V \cup P, \overline{L})$, and compute the shortest-path distances $D(v,w)$ for all pairs $v, w \in V$, and set $d_4(\{v\},\{w\}) = D(v,w)$.
4. Store all the adjacent pairs of clusters in Q.
5. Choose and delete the adjacent pair (X,Y) of clusters with the minimum distance from Q, and do the following.
   5.1. Merge X and Y into a new cluster $X \cup Y$, and put $C_i = (C_{i-1}\backslash\{X,Y\}) \cup \{X \cup Y\}$.
   5.2. Assign the average diameter by $\mathrm{AD}(X \cup Y) = d_4(X,Y)$.
   5.3. For all $Z \in C_{i-1}\backslash\{X,Y\}$, delete the pairs (X,Z) and (Y,Z) from Q, insert new pairs $(X \cup Y, Z)$ to Q, and compute the distances by eq. (14).
6. If Q is empty, stop; otherwise go to Step 5. □

The time complexity of this algorithm can be evaluated in the following way. Because $V \cup P$ contains $O(n)$ vertices, Steps 1 and 2 can be done in $O(n)$ time. Step 3 requires $O(n^2 \log n)$ time. The storage Q must support insertion, deletion and finding a minimum. For this purpose, we can use a standard data structure such as a 2-3 tree or AVL tree, with which we can execute all three operations in $O(\log n)$ time [Aho et al. 1974]. Hence, Step 4 requires $O(n \log n)$ time. In Step 5, the pair (X,Y) with the minimum distance can be found in $O(\log n)$ time. Steps 5.1 and 5.2 can be done in $O(1)$ time. Step 5.3 requires $O(n \log n)$ time, because it consists of $O(n)$ deletions and insertions of elements with respect to Q. Step 5 is repeated $n-1$ times. Step 6 is done in constant time. Hence, in total the time complexity of Algorithm 2 is $O(n^2 \log n)$.

This time complexity does not change if we assume that $\overline{N}$ is pure, because we already utilize the planarity of $\overline{N}$.

9 Median-Pair Distance

The next distance we consider is a counterpart of the one used in the median method in ordinary cluster analysis [Gower 1967].

For subset X of P, let us define $\mathrm{MD}(X)$ as the median of the shortest-path distances in $\overline{N}$ of all pairs of vertices in X. If $|X|(|X|-1)/2$ is even, we understand that the median is the average

of the shortest-path distances of the $|X|(|X|-1)/4$th and the $|X|(|X|-1)/4+1$th vertex pairs. We call $\mathrm{MD}(X)$ the *median diameter* of X.

For disjoint subsets $X, Y \subset P$, we define

$$d_5(X,Y) = \mathrm{MD}(X \cup Y), \tag{15}$$

and call it the *median-pair distance* between X and Y.

Because X and Y contain $\mathrm{O}(n)$ elements and $\overline{N}$ is planar, the shortest-path distances for all pairs of vertices can be computed in $\mathrm{O}(n^2 \log n)$ time. It is known that the median of m elements can be found in $\mathrm{O}(m)$ [Aho et al. 1974]. Hence $d_5(X,Y)$ can be computed in $t(n) = \mathrm{O}(n^2 \log n)$ time. From Theorem 1, the hierarchical structure of clusters with respect to the median-pair distance can be obtained in $\mathrm{O}(n^4 \log n)$ time.

This time complexity can be decreased if we compute and store the shortest-path distances between all pairs of vertices at the beginning. That is, we can construct the next algorithm.

Algorithm 4 (Clustering with the median-pair distance)
Input: Network $N = (V, L)$, and event vertex set $P = \{v_1, v_2, \ldots, v_n\}$.
Output: Hierarchical structure of clusters with respect to the median-pair distance.
Procedure:

1. Initialize the cluster set as $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$.
2. For all $v \in V$, set $\mathrm{MD}(\{v\}) = 0$.
3. Construct the extended network $\overline{N} = (V \cup P, \overline{L})$, and compute the shortest-path distances $D(v,w)$ for all pairs $v, w \in V$, and set $d_5(\{v\}, \{w\}) = D(v,w)$.
4. Store all the adjacent pairs of clusters in Q.
5. Choose and delete from Q the adjacent pair (X,Y) of clusters with the minimum distance, and do the following.
   5.1. Merge X and Y into $X \cup Y$, and put $C_i = (C_{i-1} \backslash \{X, Y\}) \cup \{X \cup Y\}$.
   5.2. Compute the median diameter by $\mathrm{MD}(X \cup Y) = d_5(X,Y)$.
   5.3. For all $Z \in C_{i-1} \backslash \{X, Y\}$, delete the pairs (X,Z) and (Y,Z) from Q, insert new pairs $(X \cup Y, Z)$ to Q, and compute $d_5(X \cup Y, Z)$.
6. If Q is empty, stop; otherwise go to Step 5. □

The structure of this algorithm is almost the same as Algorithm 3; the only differences are (i) AD and d_4 are replaced with MD and d_5, and
(ii) the method of computing the distances in Step 5.3.
Computation of $d_5(X \cup Y, Z)$ in Step 5.3 can be done in $\mathrm{O}(n^2)$ time for each Z, because the shortest-path distances have already been computed in Step 3. Hence, the time complexity of Algorithm 4 is also $\mathrm{O}(n^4)$. This time complexity does not decrease even if we assume that $\overline{N}$ is pure.

10 Radius Distance

In this and the next section, we consider network counterparts of the method used in the centroid method in ordinary cluster analysis [Sokal and Sneath 1963, Lance and Williams 1967].

For subset X of the set of event vertices and a vertex $v \in X$, we define $r(X, v)$ by

$$r(X, v) = \max_{v' \in X} D(v, v'), \tag{16}$$

that is, $r(X, v)$ is the shortest-path distance from v to the farthest vertex. We call $r(X, v)$ the *radius* of X with respect to v.

A vertex with the minimum radius is called a *central vertex* of X. In other words, $v \in P$ is a central vertex of X if and only if

$$r(X, v) = \min_{v' \in X} r(X, v') \tag{17}$$

holds. The radius $r(X, v)$ with respect to a central vertex v is called the *radius* of X, and is denoted by $\mathrm{Radi}(X)$.

Note that the central vertex is not necessarily unique. For example, if $X \subset P$ consists of three vertices and all pairs of vertices are connected by equal-length links, all of the vertices are central vertices. On the other hand, the radius $\mathrm{Radi}(X)$ of X is always unique.

For each $v \in X$, $r(X, v)$ can be computed in $\mathrm{O}(n \log n)$ time by first computing the shortest-path distances from v to all the other vertices by the Dijkstra's algorithm, and next taking their maximum. Because $\mathrm{Radi}(X)$ can be computed by taking the minimum of the radii $r(X, v)$ for all $v \in X$, $\mathrm{Radi}(X)$ can be obtained in $\mathrm{O}(n^2 \log n)$ time.

For disjoint subsets $X, Y \subset P$, let $d_6(X, Y)$ be defined by

$$d_6(X, Y) = \mathrm{Radi}(X \cup Y), \tag{18}$$

and call it the *radius distance* between X and Y.

Because $d_6(X, Y)$ can be computed in $\mathrm{O}(n^2 \log n)$ time, from Theorem 1, the hierarchical structure of the clusters can be computed in $\mathrm{O}(n^4 \log n)$ time.

However, this time complexity is reduced by computing the shortest-path distances between all pairs of vertices in X at the beginning of the procedure, as shown in the next algorithm.

Algorithm 5 (Clustering with respect to the radius distance)
Input: Network $N = (V, L)$, and the event vertex set $P = \{v_1, v_2, \ldots, v_n\}$.
Output: Hierarchical structure of clusters.
Procedure:
1. Initialize the cluster set as $C_0 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$.
2. Construct the extended network $\overline{N} = (V \cup P, \overline{L})$, and compute the shortest-path distances between all pairs of vertices.

3. Compute the radius distances for all adjacent pairs of event vertices and store them in priority queue Q.
4. Find and delete pair (X, Y) with the minimum distance from Q, and do the following.
   4.1. Merge X and Y into $X \cup Y$, and update the cluster set by $C_i \leftarrow (C_{i-1} \backslash \{X, Y\}) \cup \{X \cup Y\}$.
   4.2. For each $Z \in C_{i-1} \backslash \{X, Y\}$, delete (X, Z) and (Y, Z) from Q.
   4.3. For each $Z \in C_{i-1} \backslash \{X, Y\}$, if $X \cup Y$ and Z are adjacent, compute $d_6(X \cup Y, Z)$ and insert them to Q.
5. If Q is empty, stop; otherwise go to Step 4. □

Suppose that we have already obtained the shortest-path distances between all the vertices in X, and that $X, Y \subset P$ are two clusters chosen in some step of the hierarchical clustering. To compute $\mathrm{Radi}(X \cup Y)$, we first obtain $r(X \cup Y, v)$ for all v, which requires $\mathrm{O}(n^2)$ time, and next take their minimum, which requires $\mathrm{O}(n)$ time. Thus, $\mathrm{Radi}(X \cup Y)$ can be computed in $\mathrm{O}(n^2)$ time. Hence, Step 4.3 in Algorithm 5 can be executed in $\mathrm{O}(n^3)$ time. Step 4 is repeated $\mathrm{O}(n)$ times, and therefore Algorithm 5 requires $\mathrm{O}(n^4)$ time.

11 Generalized Radius Distance

For subset X of the set of event vertices and a point p on a link of the extended network $\overline{N} = (V \cup P, \overline{L})$, we define $r(X, p)$ by

$$r(X, p) = \max_{v \in X} D(p, v), \tag{19}$$

that is, $r(X, p)$ is the shortest-path distance from p to the farthest vertex. We call $r(X, p)$ the *generalized radius* of X with respect to p.

A point p is called a *generalized central point* of X if $r(X, p)$ is the minimum over all points on $\overline{N}$. The generalized radius $r(X, p)$ with respect to a generalized central vertex is called the *generalized radius* of X, and is denoted $\mathrm{GRadi}(X)$.

We should note that the generalized central point is different from the midpoint of the shortest-path connecting the farthest pair of vertices, and consequently the generalized radius of X is not always half the diameter of X. Consider the network shown in Fig. 7, which has four vertices v_1, v_2, v_3, v_4 and they are connected by links with lengths shown in the figure. The farthest pair is v_1 and v_3, and the midpoint is at v_2 and their shortest-path distance is 6. Thus, the diameter is 6. On the other hand, the generalized central point is p^* in the figure, which we reach if we move from v_2 by 0.5 toward v_4, and hence the generalized radius is 3.5.

Like the central vertex, the generalized central point of X is not unique. It can be found in the following way. For each vertex $v \in X$, let us define function $f_v(p)$ as the shortest-path distance from v to point p on the extended network $\overline{N}$.

An example is shown in Fig. 8, where solid lines represent the links in the network and broken lines represent the values of $f_v(p)$ as the height at each point on a link. We classify the

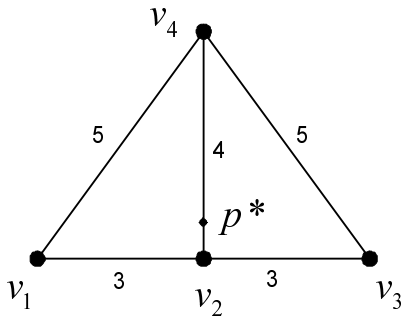

Figure 7. Generalized central point, and the midpoint of the farthest pair of vertices.

links into two classes, one consisting of links along the shortest path (which are represented by thick solid lines in Fig. 8), and the other consisting of other links (which are represented by thin solid lines in Fig. 8). The function $f_v(p)$ is linear on a link in the first class; indeed $f_v(p)$ is obtained by the linear interpolation of the shortest-path distances at the two end vertices of the link. Because $f_v(p)$ is the distance from v to p, the slope of this function is the same along all the links in the first class. On a link belonging to the second class, on the other hand, the function $f_v(p)$ consists of two pieces of linear functions, forming a local maximum, as the points p_1 and p_2 in Fig. 8. The peak is symmetric in the sense that the slope in both side of the peak is the same.

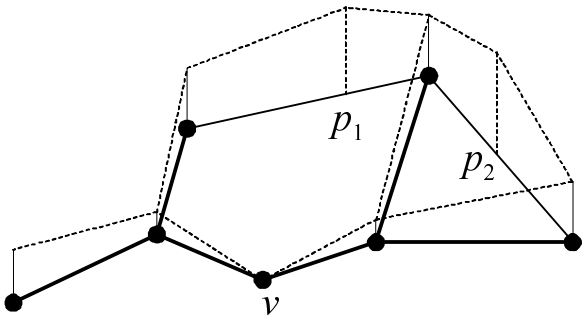

Figure 8. Shortest-path distance function.

Let $F(p)$ be the upper envelope of $f_v(p)$ for all the vertices in X, that is,

$$F(p) = \max_{v \in X} f_v(p). \tag{20}$$

$F(p)$ represents the shortest-path distance from p to the farthest vertex in X. Let p^* be the point such that $F(p^*)$ admits the minimum value of $F(p)$ over all points on $\overline{N}$. Then, the point p^* is the generalized central point.

The function $f_v(p)$ can be constructed in $O(n \log n)$ time, first by computing the shortest-path distances from v to all the vertices in X, using Dijkstra's algorithm, and then by interpolating the shortest-path distance to an arbitrary point on $\overline{N}$. For each link in $\overline{L}$, $f_v(p)$ may have at most one peak point, and hence $f_v(p)$ over all links in $\overline{N}$ is represented by a piecewise linear function with $O(n)$ pieces.

To find the generalized central point of X, we construct the upper envelope $F(p)$ defined in eq. (20), and take the minimum. Each $f_v(p)$ may have one peak on each link, and hence the upper envelope can have $O(n)$ peaks on one edge.

Figure 9 shows an example of the upper envelope with $O(n)$ peaks on one link. This network is a cycle consisting of n links $l_1, l_2, \ldots, l_n$; l_1 and l_3 have almost the same length, l_2 is very long, and all the other links are very short. Then, as shown in Fig. 9(a), $f_v(p)$ for each vertex v at the bottom, has a peak on the link l_2. If we overlap the functions $f_v(p)$ for all vertices v at the bottom, we have many peaks on l_2 as shown in Fig. 9(b), and thus the upper envelope contains $O(n)$ peaks and valleys on the link l_2.

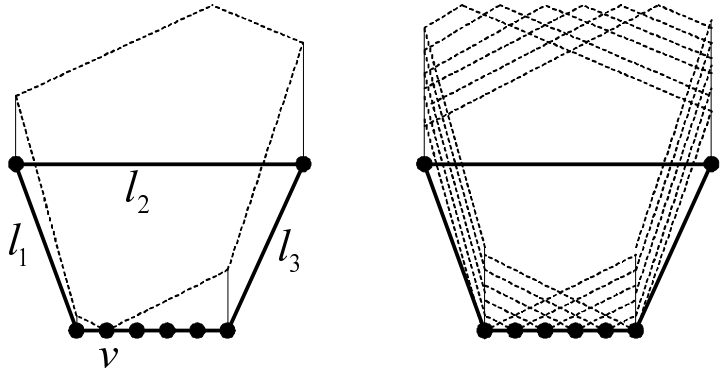

Figure 9. Link with $O(n)$ peaks and valleys in the upper envelope of the shortest-path distances.

Because the upper envelope can contain $O(n)$ peaks and valleys on each link, the total number of peaks and valleys over the whole network can be as large as $O(n^2)$. This bound is tight, as shown in Fig. 10. The network in Fig. 10 is obtained from that in Fig. 9 by first replacing the two end vertices of the link l_2 by sequences of many vertices connected with short edges, and next connecting new left and right vertices by noncrossing very long edges.

Let us change the number of vertices in such a way that the lower part contains $n/2$ vertices, the left and right upper parts both contain $n/4$ vertices, and hence the network contains n vertices. Then, there are $O(n)$ very long edges, each of which contains $O(n)$ peaks and valleys, and hence the total number of peaks and valleys of the upper envelope is proportional to n^2.

The upper envelope on a link is constructed in the following way. Suppose that X contains

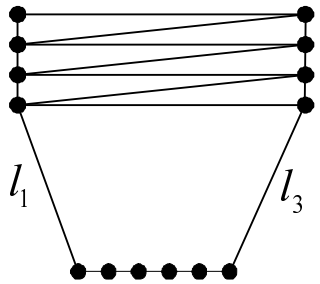

Figure 10. Network in which the upper envelope of the shortest-path distances contains $O(n^2)$ peaks and valleys.

k vertices $v_1, v_2, \ldots, v_k$. Let $F_i(p)$ represent the upper envelope of the first k functions:

$$F_i(p) = \max_{1 \le j \le i} f_{v_j}(p). \tag{21}$$

We start with $F_1(p) = f_{v_1}(p)$, and construct $F_j(p)$ by $F_j(p) \leftarrow \max\{F_{j-1}(p), f_{v_j}(p)\}$ one by one. For each j, we locate the points of intersection between $F_{j-1}(p)$ and $f_{v_j}(p)$. There are at most two points of intersection because the slopes of the functions are the same. This can be done by a binary search in $O(\log n)$ time. We repeat this for $j = 2, 3, \ldots, k$ $(\le n)$ and hence the upper envelope $F(p)$ on one link can be constructed in $O(n \log n)$ time. Therefore, the time to construct the upper envelope for the whole network is $O(n^2 \log n)$.

As we have already seen, the shortest-path distance $f_v(p)$ can be constructed in $O(n \log n)$ time for each $v \in X$. The generalized radius of X is obtained by taking the minimum of $F(P)$. Thus in total, the generalized radius $\mathrm{GRadi}(X)$ can be computed in $O(n^2 \log n)$ time.

For disjoint subsets $X, Y \subset P$, let $d_7(X, Y)$ be defined by

$$d_7(X, Y) = \mathrm{GRadi}(X \cup Y), \tag{22}$$

and call it the *generalized radius distance* between X and Y.

Because $\mathrm{GRadi}(X \cup Y)$ can be obtained in $O(n^2 \log n)$ time, the distance $d_7(X, Y)$ can also be found in $t(n) = O(n^2 \log n)$ time. Consequently, from Theorem 1, the hierarchical structure of clusters can be constructed in $O(n^4 \log n)$ time.

It seems that this time complexity does not decrease even if we compute the shortest-path distances for all pairs of vertices at the beginning, because the computation of the generalized radius for each subset of P still requires $O(n^2 \log n)$ time.

12 Remark on Ward's Method

One of the most common methods in general cluster analysis is Ward's method [Ward 1963, Ward and Hook 1963, Whishart 1969]. In this method, the distance between two clusters is

defined as the increase of the variance on merging the associated two clusters. Readers might expect that Ward's method can be extended to the network cluster analysis.

However, this seems difficult. Recall that the variance is defined as the sum of the squared distances from the mean to the individual data points. The counterpart of the mean for a network seems to be the central vertex or the generalized central point. However, neither the central vertex nor the generalized central point are unique, as we have already seen. Consequently, a concept similar to the variance cannot be defined in a straightforward manner.

One might think that we can define the variance by checking all the central vertices or the generalized central points and by taking the minimum over all the associated variances. However, this approach is far from practical, because there are $O(n)$ central vertices or generalized central points in the worst case, and consequently the computational cost will be very high; indeed the total time complexity will be of $O(n^5 \log n)$ if we compute the variance with respect to the generalized central point. The extension of Ward's method to the network is therefore an open problem for future.

13 Concluding Remarks

We have considered algorithms for hierarchical cluster analysis on networks for various distances between clusters. The distances we have considered are the closest-pair distance, the farthest-pair distance, the diameter distance, the average-pair distance, the median-pair distance, radius distance, and the generalized radius distance.

For each individual distance, we first construct a method for computing the distance between two clusters independently, consider its computational time, and evaluate the time complexity of the general algorithm for the hierarchical clustering. Next, we tried to improve the time complexity by utilizing the relations among distances between all possible pairs of clusters. We summarize our results in Table 1, where the leftmost column shows the distance type between clusters, the next column shows the time complexity of the direct method, in which the distances are computed when they become necessary. The right two columns show the time complexity of the improved methods, in which the common parts of the computation of the distances is done at the beginning. The second-last column is for a general network, and the rightmost column is for a pure network in which all the vertices in the original network are objects to be clustered.

We understand that this is the first systematic study of the time complexity of hierarchical cluster analysis on networks. However, this is just the start of studies in this area, and many problems remain to be solved in future.

The first group of future problems includes the design of more efficient algorithms for bottleneck procedures. In particular, the median-pair distances, the radius distances, and the generalized radius distances require much time for computation even if we try to compute them in total. As a result, the cluster analysis for these distances costs $O(n^4)$ or more time, and hence cannot be used for large networks. For practical use, more efficient methods are necessary. If

Table 1. Summary of the time complexity of the hierarchical cluster analysis in a network.

Distance	Direct method (Algorithm 1)	Improved method	
		General network	Pure network
Closest-pair	$O(n^4 \log n)$	$O(n^2 \log n)$	$O(n \log n)$
Farthest-pair	$O(n^4 \log n)$	$O(n^2 \log n)$ (Algorithm 2)	$O(n^2 \log n)$
Diameter	$O(n^4 \log n)$	$O(n^2 \log n)$ (Algorithm 2)	$O(n^2 \log n)$
Average pair	$O(n^4 \log n)$	$O(n^2 \log n)$ (Algorithm 3)	$O(n^2 \log n)$
Median pair	$O(n^4 \log n)$	$O(n^4)$ (Algorithm 4)	$O(n^4)$
Radius	$O(n^4 \log n)$	$O(n^4)$ (Algorithm 5)	$O(n^4)$
Generalized radius	$O(n^4 \log n)$	—	—

they are difficult, we must also consider approximation strategies instead of exact computation of the cluster structure.

We have not yet succeeded in generalizing the concept of the variance to the network, and consequently have not constructed a Ward-like method. To generalize the variance, we require the counterpart of the mean for the network. It seems that the concepts of the central vertex and the generalized central point are near to the mean, but the uniqueness is not guaranteed and hence they cannot be used to define something like the variance.

Properties (i.e., merits and demerits) of the clusterings based on individual distances should also be studied from both the theoretical and experimental points of view. We want to provide some standard for users to select the distances that are suitable for their own purposes for cluster analyses in practical situations.

For practical users, and also to acquire our own experimental experience, we must implement the algorithms in computer software. This is one of our main future tasks.

Acknowledgments

This study was partly supported by the Grant-in-Aid for Scientific Research (B) No. 20300098 and (B) No. 20360044 of the Japanese Society for Promotion of Science.

References

[Aho et al. 1974] A. V. Aho, J. E. Hopcroft and J. D. Ullman: *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Massachusetts, 1974.

[Anderberg 1973] M. R. Anderberg: *Cluster Analysis for Applications*. Academic Press, New York, 1973.

[Cormen et al. 2001] T. H. Cormen, C. E. Leisersen, R. L. Rivert and C. Stein: *Introduction to Algorithms*, 2nd Edition, MIT Press, Cambridge, Massachusetts, 2001.

[Gower 1967] J. C. Gower: A comparison of some methods of cluster analysis. *Biometrics*, **23** (1967), 623–637.

[Gower and Ross 1969] J. C. Gower and G. J. S. Ross: Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, **18** (1969), 54–64.

[Johnson 1967] S. C. Johnson: Hierarchical clustering schemes. *Psychometrika*, **32** (1967), 241–254.

[Lance and Williams 1967] G. N. Lance and W. T. Williams: A general theory of classificatory sorting strategies. 1. Hierarchical systems. *Computer Journal*, **8** (1967), 246–249.

[Sokal and Sneath 1963] R. R. Sokal and P. H. Sneath: *Principles of Numerical Taxonomy*. San Francisco: Freeman, 1963, 183–185.

[Ward 1963] J. H. Ward, Jr.: Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, **58** (1963), 236–244.

[Ward and Hook 1963] J. H. Ward, Jr. and M. E. Hook: Application of an hierarchical grouping procedure to a problem of grouping profiles. *Education and Psychological Measurement*, **23** (1963), 69–82.

[Whishart 1969] D. Whishart: An algorithm for hierarchical classifications. *Biometrics*, **22** (1969), 165–170.

[Zahn 1971] C. T. Zahn: Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, **C-20** (1971), 68–86.