Journal of Classification 13:231-248 (1996)

The Weighted Sum of Split and Diameter Clustering

Y. Wang

University of Toronto

H. Yan

The Chinese University of Hong Kong

C. Sriskandarajah

University of Toronto

This research was supported in part by the National Science and Engineering Research Council of Canada (Grant OGP 0104900). The authors wish to thank two anonymous referees, whose detailed comments on earlier drafts improved the paper.

Authors' Addresses: Y. Wang, Faculty of Management, University of Toronto, 246 Bloor St. W., Toronto M5S 1V4, Canada; H. Yan, Department of Systems Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong; C. Sriskandarajah, Department of Industrial Engineering, University of Toronto, 4 Taddle Creek Road, Toronto M5S 1A4, Canada.

Abstract: In this paper, we propose a bicriterion objective function for clustering a given set of N entities, which minimizes $[\alpha d - (1 - \alpha)s]$, where $0 \leq \alpha \leq 1$, and d and s are the diameter and the split of the clustering, respectively. When $\alpha = 1$, the problem reduces to minimum diameter clustering, and when $\alpha = 0$, maximum split clustering. We show that this objective provides an effective way to compromise between the two often conflicting criteria. While the problem is NP-hard in general, a polynomial algorithm with the worst-case time complexity $O(N^2)$ is devised to solve the bipartition version. This algorithm actually gives all the Pareto optimal bipartitions with respect to diameter and split, and it can be extended to yield an efficient divisive hierarchical scheme. An extension of the approach to the objective $[\alpha(d_1 + d_2) - 2(1 - \alpha)s]$ is also proposed, where d_1 and d_2 are diameters of the two clusters of a bipartition.

Keywords: Diameter; Split; Bicriteria clustering; Complexity; Polynomial algorithm; Divisive hierarchical clustering.

1. Introduction

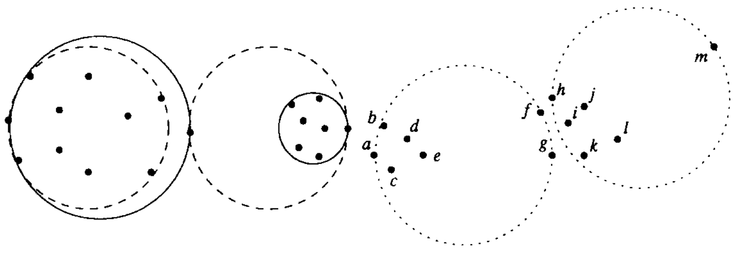
Clustering problems arise regularly in many areas such as group technology, data reorganization, pattern recognition, biology, etc. The application of clustering analysis is based on the assumption that homogeneous clusters actually exist in the data. The objective of cluster analysis is to partition entities of a given set into homogeneous and/or well separated classes. Separation and homogeneity can be expressed in many different ways. Typically, measures of dissimilarities can be defined for any two individual elements in the data set. Based on the dissimilarities, two criteria are widely used: split and diameter as proposed by Delattre and Hansen (1980). The split of two clusters, say A and B, is defined as the minimum dissimilarity between two entities, one from A and the other from B. The diameter of a cluster is defined as the maximum dissimilarity between two entities in the same cluster. The former measures separation of two clusters, and the latter measures homogeneity of a cluster. Similarly, the split of a partition is defined as the minimum of all splits between any two clusters in the partition, and the diameter of a partition is defined as the maximum of diameters of all the clusters in the partition.

An ideal partition should then have minimum diameter and maximum split. Unfortunately, the two criteria are often conflicting. Minimum diameter clustering often suffers from the dissection effect (Cormack 1971), i.e., quite similar entities may end up in different groups in order to keep the diameter small. On the other hand, maximum split clustering suffers from the chain effect (Johnson 1967), i.e., very different entities are clustered in the same group.

Example (a) in Figure 1 is taken from Hansen and Jaumard (1987). It illustrates a typical case of the dissection effect. The solid circles represent a

	b	c	d	e	f	g	h	i	j	k	l	m	
a	1.2	0.9	1.4	2.0	6.8	7.0	7.3	7.7	8.5	8.2	9.5	14.0	a
b		1.7	1.1	2.0	6.3	6.7	6.7	7.3	7.9	7.9	9.2	13.3	b
c			1.4	1.4	6.4	6.3	6.9	7.2	7.9	7.6	8.9	13.5	c
d				0.9	5.4	5.8	5.9	6.3	7.2	6.9	8.2	12.6	d
e					5.0	5.1	5.5	5.8	6.6	6.3	7.6	12.2	e
f						1.6	0.8	1.0	1.6	2.2	3.0	7.2	f
g							2.2	1.4	2.3	1.2	2.6	7.6	g
h								1.2	1.3	2.6	2.3	6.7	h
i									0.9	1.4	2.0	6.5	i
j										1.9	1.8	5.6	j
k											1.4	6.6	k
l												5.3	l

Table 1. Data for Example (b) of Figure 1



(a) dissection effect (b) chain and dissection effects

Figure 1. Examples of partitions that suffer the dissection and chaineffects.

natural bipartition of the data set. But if the min-diameter criterion were used, a partition shown as the dashed circles would be inevitable. Example (b) further demonstrates that when objectives focusing on either one of the two criteria are used, unsatisfactory partitions may occur. The distance matrix for this example is shown in Table 1.

For example, to minimize the diameter of a bipartition, entities *f* and *g* would be assigned to a group including all the entities on their left, resulting in a diameter 7 and a split 0.8; to maximize the split, however, we would select a partition with *m* in a single entity cluster and all others in another cluster, producing a split 5.3 and a diameter 9.5.

Remark. The examples shown in Figure 1 have natural partitions that are easily recognized by sight, and the undesired partition in (b) results from the choice of the number of clusters which is equal to two. However, this does not exclude seeking better alternatives. In fact, these examples clearly demonstrate the tendency that if we focus on either criterion, the other one will be weakened. While this does not pose serious difficulty for small problems, real-world data sets are so large that one cannot determine in advance which criterion and how many clusters will produce the best results. Moreover, in many cases, dissimilarities cannot be represented by *Euclidean* distances, which makes a visual inspection of the ``goodness'' of a partition impossible.

Trying to resolve the conflict, Hansen and Jaumard (1987) propose using the minimum sum of diameters criterion to overcome the drawbacks shown in Figure 1(a), and they give an $O(N^3 \log N)$ algorithm, where N is the size of the data set. But this criterion may not deal well with the situation shown in Figure 1(b): in this case, it would yield a bipartition the same as the one maximizing split. For the M-clustering problem, i.e., the problem of partitioning the data set into M clusters, $2 \leq M \leq N$, Delattre and Hansen (1980) suggest searching for the Pareto efficient solutions for a bicriterion objective.[1] Their algorithm is enumerative, and hence its time complexity is exponential. Thus, finding an effective way to compromise between homogeneity and separation of a partition is an important subject in the clustering literature.

Another difficulty comes from the complexity of clustering problems. While the maximum split problem can be solved in $O(N^2)$ time for any M (Gower and Ross 1969; Hubert 1977; Delattre and Hansen 1980), the minimum diameter can only be solved in $O(N^2)$ time for $M = 2$ (Rao 1971; Zahn 1971). For $M > 2$, it becomes NP-hard (Brucker 1978; Hansen and Delattre 1978). For $M \geq 3$, the problem of minimizing diameter is NP-hard (see Brucker 1978; Hansen and Delattre 1978; and Welch 1983). In dealing with M-clustering problems, two strategies are commonly used, hierarchical clustering methods and non-hierarchical clustering methods. Hierarchical strategies branch into agglomerative hierarchical schemes and divisive hierarchical schemes. The former start with N clusters, each entity being a cluster, and then merge two clusters into one at each iteration. The latter work in the opposite way, starting with all the entities clustered together and recursively dividing one cluster into two. We shall discuss the divisive hierarchical approach in this paper.

The efficiency of a divisive hierarchical scheme depends on the bipartition algorithm that is invoked at each iteration. In this paper, we show that partitioning problems with the following objective can be solved efficiently for the bipartition case, and that such a bicriterion objective indeed provides an effective compromise:

$$\min\,[\alpha d - (1 - \alpha)s]\,,$$

where $0 \leq \alpha \leq 1$, and d and s are the diameter and split of the partition, respectively. This objective is a generalization of min d and max s.

1. A partition is Pareto efficient if no other partition into at most the same number of clusters is better in both criteria.

Such an objective function leads to a satisfactory compromise for the two examples shown in Figure 1. The partition represented by the solid circles in Example 1(a) can be obtained with an $\alpha \leq 0.75$. With $0.215 < \alpha < 0.875$ in Example 1(b), partition $\{a,b,c,d,e\}$, $\{f,g,h,i,j,k,l,m\}$ provides the optimal solution with a diameter 7.6 and a split 5.0. These examples also demonstrate the insensitivity of the choice of α.

The weighted sum type of objective is extensively used in multiple criteria decision making. Quite often, two or more conflicting objectives are desired at the same time. For instance, in portfolio analysis, return and risk usually move in the same direction, but one is to be maximized while the other is to be minimized. In location theory, center and median are two measures that reflect two different interests. Various ways of combining two objectives have been studied (see Berman, Einav and Handler 1990; Hansen 1979; and for a comprehensive text on *multiple criteria decision making*, Zeleny, 1982).

The advantages of the weighted sum type are immediate: it is easily understood, and the decision maker can control the outcome by choosing different weighting schemes. The disadvantage is that sometimes it is not clear which criterion should receive the greater weight. To overcome this, we derive a minimal set of dominant partitions in the sense that for any bipartition P not in the set, there is one P' in the set such that P' is preferable to P for any given α. We have developed an efficient algorithm to accomplish this task.

This paper is organized as follows. First the problem is formulated and notations are specified in the next section. In Section 3, we present the major contribution of this paper: definition of the dominance class and the algorithm that derives the dominance class. An example is provided in Section 4. The result is then extended to the following objective in Section 5:

$$\min_{P_2} [\alpha(d_1 + d_2) - 2(1 - \alpha)s] ,$$

where $P_2 = \{V_1, V_2\}$ is a bipartition, d_1 and d_2 are diameters of V_1 and V_2, respectively, and s is the split of P_2. While this problem can be solved in $O(N^4 \log N)$ time, an efficient approximation algorithm of complexity $O(N^3)$ is also developed.

2. Problem Formulation

Given a set $V = \{v_1, v_2, \ldots, v_N\}$ with $N = |V|$ entities, define a real function on $V \times V$, $f: (v_k, v_l) \rightarrow d_{kl}$, $k = 1,2,\ldots,N$, $l = 1,2,\ldots,N$, that satisfies $d_{kl} \geq 0$, $d_{kk} = 0$, and $d_{kl} = d_{lk}$ for all k,l. An M-clustering of V is defined as a partition of V into M nonempty disjoint subsets (clusters):

$P_M = \{V_1, V_2, \ldots, V_M\}$. Each cluster V_j has two associated measurements: a split and a diameter defined by

$$s(V_j) = \min_{v_k \in V_j, v_l \notin V_j} d_{kl} ,$$

$$d(V_j) = \max_{v_k, v_l \in V_j} d_{kl} ,$$

respectively. Similarly, a split and a diameter are also defined for the partition P_M:

$$s(P_M) = \min_j s(V_j) ,$$

$$d(P_M) = \max_j d(V_j) .$$

Two commonly used clustering objectives are $\min d(P_M)$ and $\max s(P_M)$. Hereafter, we shall refer to $\min d(P_2)$ as the MinDiameter problem and $\max s(P_2)$ as the MaxSplit problem.

Define a valued graph $G = (V,E)$, where $E = \{(v_k,v_l) \mid v_k,v_l \in V$, for all $k < l\}$ with the weight on (v_k,v_l) being d_{kl}. Then MinDiameter and MaxSplit are solved by constructing a maximum spanning tree and a minimum spanning tree of G, respectively. Recall that a *spanning tree* T of G is a subgraph T of G which contains all the vertices of G, is connected and has no cycle. A *maximum spanning tree* (*minimum spanning tree*) is a spanning tree such that the sum of the weights of its edges is the maximum (minimum).

Theorem 1. (Hansen and Jaumard 1987) *Let T denote a maximum spanning tree of G and (v_p,v_q) an edge of $G \backslash T$ which has the maximum weight subject to the condition that its union with T forms an odd cycle. Then for all partitions $\{V_1,V_2\}$ of V, either $d(\{V_1,V_2\}) = d_{pq}$ or $d(\{V_1,V_2\}) = d_{kl}$ with $(v_k,v_l) \in T$ and $d_{kl} \geq d_{pq}$.*

Note that a tree admits a unique bicoloration of its vertices. The following corollary, obtained independently by Guénoche (1989) and Monma and Suri (1991), indicates how a minimum diameter bipartition can be obtained.

Corollary 1. *The partition $\{V_1,V_2\}$ induced by a bicoloration of the vertices of a maximum spanning tree T of G has minimum diameter.*

The following theorem gives an indication on how to construct a maximum split bipartition (which is in fact given by the well-known single-linkage algorithm, Gower and Ross 1969).

Theorem 2. (Delattre and Hansen 1980) *The number of distinct values of the split $s(P_M)$ for all partitions P_M of V is at most $N-1$. Moreover, these values are equal to the weights of the edges of the minimum spanning tree of G.*

Let T_1 denote a minimum spanning tree of G and T_2 a maximum spanning tree of G. Thus the longest edge in $G \backslash T_2$ that closes an odd cycle in T_2 defines the minimum diameter of all partitions P_2 of V:

$$d^* = \min d(P_2) . \tag{1}$$

The longest edge in T_1 defines the maximum split of all partitions P_2 of V:

$$s^* = \max s(P_2) . \tag{2}$$

As discussed in Section 1, either criterion alone has serious drawbacks. It is natural then to consider both criteria simultaneously. Here we use the weighted sum of diameter and split of a partition as the objective. First define for any partition P:

$$g(\alpha,P) = \alpha d(P) - (1-\alpha)s(P) , \tag{3}$$

where $0 \le \alpha \le 1$. Then the weighted sum of diameter and split problem (WSDS) can be written as

$$g(\alpha) = \min_{P \in \mathbf{P}} g(\alpha,P) , \tag{4}$$

where **P** is the collection of all admissible partitions.

For $M > 2$, the problem is clearly NP-hard. We shall show that, for $M = 2$, the problem can be solved in $O(N^2)$ time, the same complexity as for solving either the MinDiameter or MaxSplit problem.

In the remainder, we will only consider the bipartition version of the WSDS problem, i.e., unless otherwise stated, $M = 2$.

3. The Dominance Class

Since the choice of α in (4) is rather subjective, instead of solving the problem for a given α, we focus on obtaining a set of bipartitions that dominate those not in the set regardless of the choice of α.

Let us now take a closer look at the problem. Let the edges of the minimum spanning tree T_1 be $E_1 = \{e_1^s, e_2^s, \ldots, e_{N-1}^s\}$. The elements of E_1 are indexed such that $f(e_1^s) \le f(e_2^s) \le \ldots \le f(e_{N-1}^s)$.

Consider a bipartition P^0 induced by bicoloring the vertices of the maximum spanning tree T_2, which has the optimal diameter $d^0 = d^*$ and a split $s^0 = f(e_i^s)$ for some i. If $s^0 = s^*$, then P^0 is optimal. But this is rarely

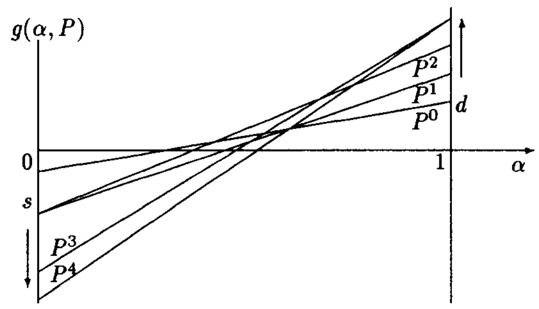


Figure 2. Illustration of Algorithm WSDS.

the case. (Recall that $s^* = f(e^s_{N-1})$. Thus if $s^0 < s^*$, we must have $i < N - 1$.) To improve the objective, we can impose the condition that the resulting partition has a split $s > f(e^s_i)$. Although this may lead to an increased diameter, the weighted sum may be reduced. Equivalently, the condition stipulates that edge e^s_i and all other edges with weight less than or equal to $f(e^s_i)$ are collapsed so that their vertices are merged and forced into the same cluster. This constraint may or may not increase the diameter. To update the diameter, we consider two cases. In the first case, the collapsed edge does not belong to T_2, and hence a cycle C is formed in T_2 after the merge. To find the minimum diameter with this restriction, we can update T_2 by simply cutting C at the shortest edge. A local recoloration of the updated T_2 will define a new partition P^i. According to Theorem 1, if C (after two vertices are merged) is an even cycle, the diameter is still d^0; if C is an odd cycle, and the shortest edge in C has a weight greater than d^0, the diameter will become equal to the weight of the edge. In the second case, the collapsed edge belongs to T_2, and no cycle is formed. In this case, if the weight of the collapsed edge is greater than d^0, it becomes the new diameter; otherwise, the diameter remains unchanged. In any case, we have $d^i \geq d^0$ and $s^i \geq s^0$, where $d^i = d(P^i)$ and $s^i = s(P^i)$, and we can compare P^0 and P^i to determine which one is preferable.

Let us write

$$h(\alpha, e^s_i) = g(\alpha, P^i) = \alpha d(P^i) - (1 - \alpha)s(P^i)\,, \tag{5}$$

where P^i is the bipartition such that $d(P^i) = \min_{s(P_2) \geq f(e^s_i)} d(P_2)$. Thus, for a given α, the WSDS problem is decomposed to subproblems $\{h(\alpha, e) : e \in E_1\}$

(Theorem 2), and we have

$$g(\alpha) = \min_{e \in E_1} h(\alpha, e). \tag{6}$$

We are now ready to develop an algorithm that progressively increases s, and in each iteration, finds a partition with the best diameter. When α is arbitrary, we are interested in deriving a dominance class which is defined as follows.

Dominance class. *Let* **P** *be the collection of all bipartitions of V. Let* $g(\alpha,P)$ *be a measure of satisfaction of* $P \in \mathbf{P}$ *for* $0 \leq \alpha \leq 1$. *If* $g(\alpha,P') \leq g(\alpha,P'')$ *for* $P', P'' \in \mathbf{P}$ *for all* $\alpha \in [0,1]$, *we say* P' *dominates* P''. *A dominance class* $\mathbf{D} \subset \mathbf{P}$ *is a set of partitions such that each element in* $\mathbf{P} \setminus \mathbf{D}$ *is dominated by at least one partition in* **D**, *and no pair of partitions in* **D** *dominates each other.*

```
Algorithm WSDS
   inputs: $G = (V,E,f)$;
   outputs: $\mathbf{D}$.
1      begin
2         construct the minimum spanning tree $T_1$;
3         arrange the edges of $T_1$ in the non-decreasing
            order of their weights;
            set $s^* = f(e^s_{N-1})$;
4         construct the maximum spanning tree $T_2$,
            and find the longest edge in $G \setminus T_2$
            that forms an odd cycle with edges in $T_2$,
            say, $(v_1,v_2)$;
            set $d = f(v_1,v_2)$;
5         bicolor $T_2$ and find a bipartition $P^0$,
            compute $s_0 = s(P^0)$;
            stack $P^0$ in $\mathbf{D}$;
6         if $s_0 = s^*$, stop;
7         for $i = 1$ to $N-2$ do
8            merge $(T_2,u,v)$, where $u,v$ are
               vertices of $e^s_i$;
9            if $f(e^s_{i+1}) > f(e^s_i)$, then do
10              locally recolor $T_2$ and find bipartition $P^i$;
11              compute split $s_i = s(P^i)$;
12              if $s_i > s_0$, set $s_0 = s_i$, update $\mathbf{D}$;
13           endif
14        endfor
15     end
```

```
Procedure merge $(T,u,v)$
    begin
        if $u$ and $v$ are already merged, end;
        merge $u,v$;
        if $(u,v) \in T$,
            delete $(u,v)$ from $T$, and rebuild $T$;
            if $f(u,v) > d$, let $d = f(u,v)$;
        else
            identify cycle $C$ in $T$, and
                find the edge of minimum weight in $C$
                  and denote it $(v_1,v_2)$;
            if $C$ is an odd cycle and $f(v_1,v_2) > d$,
              let $d = f(v_1,v_2)$;
            delete $(v_1,v_2)$ from $T$, and rebuild $T$;
        endif
    end
```

We now describe our algorithm {WSDS} that finds such a dominance class. Procedure merge (T,u,v) takes a maximum spanning tree T, merges the two vertices u and v, finds the cycle formed by the merging, and updates the maximum spanning tree by cutting the cycle at the shortest edge. Algorithm WSDS takes the graph, computes the minimum and the maximum spanning trees (lines 2-5), and in the mean time, identifies the optimal diameter d^*. It then enumerates lines 8-13 for each edge in T_1, from the shortest to the longest, and finds a minimum diameter partition given that the edge under consideration has to be collapsed. In each loop, either both the split and the diameter remain unchanged, or at least one of them increases. When the split increases, a new dominant partition appears, and **D** is updated by adding the new partition and removing those dominated by it; otherwise, the new partition is dominated and ignored. Figure 2 illustrates the process.

Since Algorithm WSDS enumerates explicitly the split candidate values, and for each enumeration, an optimal diameter is obtained subject to the split constraint, the algorithm locates all the dominant bipartitions. Also since computing a minimum (maximum) spanning tree takes $O(N^2)$ time, bicoloring T_2 takes $O(N)$ time, and each line between lines 8 and 13 takes at most $O(N)$ time and is executed for $O(N)$ times, the algorithm has a time complexity of $O(N^2)$. This is stated in the following theorem. The proof can be found in the Appendix.

Theorem 3. *Algorithm* WSDS *finds the dominance class* **D** *in* $O(N^2)$ *time.*

Note that each partition in **D** defines a function of α:

$$g(\alpha,P^i) = \alpha d(P^i) - (1-\alpha)s(P^i).$$

For any pair of partitions P' and P'' in **D**, $g(\alpha,P')$ and $g(\alpha,P'')$ intersect each other at some value of $\alpha \in (0,1)$. Thus the following function is well defined:

$$g^*(\alpha) = \min_{P\in\mathbf{D}} [\alpha d(P) - (1-\alpha)s(P)]. \qquad (7)$$

Clearly, $g^*(\alpha)$ is a concave, piecewise linear and increasing function. Note, however, the difference between the dominance class **D** and the elements in **D** that define $g^*(\alpha)$. The latter form only a subset of **D**. For instance, in Figure 2, P^0,P^1 and P^4 constitute the dominance class, but only P^0 and P^4 define the function $g^*(\alpha)$.

Given function $g^*(\alpha)$, a bipartition satisfying our objective can easily be determined by choosing a value of α that best suits the compromise.

Remark. Although our dominance class is defined with respect to the *weighted sum* of diameter and split, our algorithm actually finds the Pareto-efficient set; that is, no bipartition not belonging to **D** has both a smaller diameter and a greater split than any bipartition in **D**, and any two elements in **D** are equivalent in the sense that if one has a smaller diameter than the other, it must also have a smaller split than the other. Such a set can also be used with any other utility functions defined with respect to diameter and split by decision makers (presumably, a greater split and a smaller diameter are always preferable).

Based on this result, we propose using a divisive hierarchical scheme that recursively selects the cluster with the largest diameter and optimally partitions it. With the algorithm we developed, we can achieve a complexity of $O(MN^2)$ for a hierarchical M-clustering method. This, to our knowledge, represents the first polynomial algorithm for solving *multiple criteria* clustering problems in the literature.[2]

Divisive hierarchical schemes offer effective approximations for the general problem. Guénoche, Hansen and Jaumard (1991) have provided strong arguments in favor of such schemes.

2. There exist efficient hierarchical algorithms, however, for single criterion M-clustering problems. For example, Guénoche, Hansen and Jaumard (1991) give an efficient algorithm for a divisive method based on minimum diameter bipartitions.

	2	3	4	5	6	7	8	9	10	11	12	13
1	•.682	.597	.532	.679	.665	.696	.668	.674	.884	.692	.686	.511
2		.683	.770	.715	.766	.843	.843	.805	.943	.850	.855	.761
3			.695	.753	.732	.777	.618	.816	1.075	.909	.860	.679
4				.773	.673	.665	.609	.675	.901	.890	.840	.673
5					.378	.344	.422	.277	.689	.656	.785	.656
6						.278	.473	.286	.797	.647	.909	.691
7							.381	.315	.754	.768	.819	.655
8								.468	.715	.700	.729	.605
9									.830	.720	.887	.720
10										.516	.415	.592
11											.572	.469
12												.448

Table 2. Data for the Example of Section 4

Merge	P^i	d	s	V_1	V_2
	P^0	0.819 (7,12)	0.278 (6,7)	1,2,3,4,5,6,9	7,8,10,11,12,13
(5,9)	P^1	0.819 (7,12)	0.278 (6,7)	1,2,3,4,5,6,9	7,8,10,11,12,13
(6,7)	P^2	0.843 (2,7)	0.381 (7,8)	1,2,3,4,5,6,7,9	8,10,11,12,13
(6,9)	P^3	0.843 (2,7)	0.381 (7,8)	1,2,3,4,5,6,7,9	8,10,11,12,13
(7,8)	P^4	0.843 (2,7)	0.511 (1,13)	1,2,3,4,5,6,7,8,9	10,11,12,13
(10,12)	P^5	0.843 (2,7)	0.511 (1,13)	1,2,3,4,5,6,7,8,9	10,11,12,13
(12,13)	P^6	0.843 (2,7)	0.511 (1,13)	1,2,3,4,5,6,7,8,9	10,11,12,13
(11,13)	P^7	0.843 (2,7)	0.511 (1,13)	1,2,3,4,5,6,7,8,9	10,11,12,13
(1,13)	P^8	0.884 (6,12)	0.532 (1,4)	2,3,4,5,6,7,8,9	1,10,11,12,13
(1,4)	P^9	0.901 (4,10)	0.597 (1,3)	2,3,5,6,7,8,9,	1,4,10,11,12,13
(1,3)	P^{10}	1.075 (3,10)	0.605 (8,13)	2,5,6,7,8,9	1,3,4,10,11,12,13
(8,13)	P^{11}	1.075 (3,10)	0.682 (1,2)	2	1,3,4,5,6,7,8,9,10,11,12,13

Table 3. Iterations of Algorithm WSDS

$0 \le \alpha < 0.328$	$0.328 \le \alpha < 0.597$	$0.597 \le \alpha < 0.907$	$0.907 \le \alpha < 1$
P^{11}	P^9	P^4	P^0
$1.757\alpha - 0.682$	$1.498\alpha - 0.597$	$1.354\alpha - 0.511$	$1.097\alpha - 0.278$

Table 4. Function $g^*(\alpha)$ Defined for the Psychological test Data

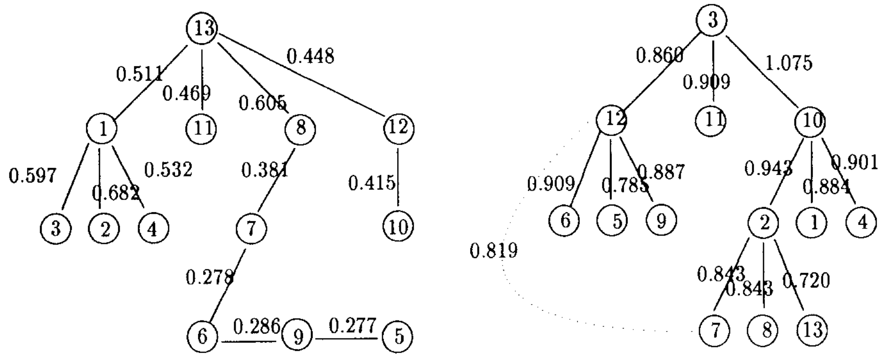


Figure 3. The minimum spanning tree (left) and the maximum spanning tree (right).

4. An example

In this section, we demonstrate Algorithm WSDS with an example. The data are taken from a real case of 13 psychological tests (Harman 1967). These data have been used in the clustering literature for illustrations, e.g., Delattre and Hansen (1980). The dissimilarity matrix is shown in Table 2.

The minimum spanning tree T_1 and the maximum spanning tree T_2 corresponding to the data are shown in Figure 3. The maximum split clustering yields a partition of $V_1 = \{2\}$ and $V_2 = \{1,3,4,5,6,7,8,9,10,11,12,13\}$ with a split 0.682 and a diameter 1.075. The minimum diameter clustering gives a partition of $V_1 = \{1,2,3,4,5,6,9\}$ and $V_2 = \{7,8,10,11,12,13\}$ with a split 0.278 and a diameter 0.819. We will use this example to demonstrate Algorithm WSDS.

The detailed iterations of executing Algorithm WSDS for this example are listed in Table 3. By bicoloring T_2, a bipartition P^0 is determined, which is shown in the first row of Table 3. From this initial partition, we begin the algorithm by merging entities 5 and 9 into a supernode (5,9) because they correspond to the smallest edge in T_1. Since both entities 5 and 9 are in the same cluster V_1^0, merging them does not change the initial partition. In the next iteration, entities 6 and 7 are merged; this merge forms an odd cycle which has to be broken at the shortest edge (2,7), and thus forces 7 into V_1. The new partition becomes $V_1^2 = \{1,2,3,4,5,6,7,9\}$ and $V_2^2 = \{8,10,11,12,13\}$ with a split 0.381 and a diameter 0.843. In the same manner, Algorithm WSDS ends when all entities in T_1 are merged. The dominance class consists of P^0,P^4,P^8,P^9 and P^{11} (P^2 is dominated by P^4, P^{10} is dominated by P^{11}). Among them, P^0,P^4,P^9 and P^{11} define a function $g^*(\alpha)$ which is shown in Table 4.

5. An extension

The approach discussed in the previous sections and Algorithm WSDS can be adapted to incorporate the following objective function:

$$g(\alpha,P_2) = \alpha[d(V_1) + d(V_2)] - 2(1-\alpha)s(P_2)\,. \tag{8}$$

It is done simply by replacing line 10 of Algorithm WSDS with a procedure that finds an optimal solution to

$$\min_{P_2} [d(V_1) + d(V_2)]\,. \tag{9}$$

Hansen and Jaumard (1987) propose a procedure to solve the min-sum-of-diameter problem. They reduce the problem to the determination of the consistency of a quadratic Boolean equation and obtain an $O(N^3 \log N)$ algorithm. Applying their procedure in Algorithm WSDS and using the algorithm in a divisive hierarchical scheme for solving the M-partition problem, we would obtain an algorithm with a worst-case complexity of at least $O(MN^4 \log N)$. For large N, this is quite slow. In this section, we propose an efficient approximation algorithm that utilizes the information contained in the spanning trees and yields an $O(N^2)$ time complexity for solving the min-sum-of-diameter problem. A divisive hierarchical scheme embedding this algorithm will therefore have an $O(MN^3)$ overall complexity.

Consider the problem of bipartition given that an edge (u,v) defines the diameter for V_1 while $d(V_2)$ is minimized:

$$\min d_2 = d(V_2) \tag{10}$$

$$s.t. \quad u,v \in V_1, \text{ and} \tag{11}$$

$$d_1 = d(V_1) = f(u,v)\,. \tag{12}$$

Without loss of generality, we assume that $d_1 \geq d_2$. According to Theorem 1, we only need to consider as the candidates for d_1 the edges of the maximum spanning tree T_2 that satisfy $d_{kl} \geq d_{pq}$ and the edge (v_p,v_q) as defined in the theorem. Let E_d be the collection of such edges excluding (v_p,v_q). We propose a bicoloration procedure **minsum** to derive the partition $\{V_1,V_2\}$.

```
Procedure minsum ($T,E_d,v_p,v_q$)

  inputs: $T$, a tree;
      $E_d$, a collection of candidate edges;
  output: bipartition $\{V_1,V_2\}$ with $V_1$
      being red and $V_2$ blue;
    begin
```

```
        let $d_1 = f(v_p,v_q)$, $d_2 = 0$;
        color $v_p$ red, depth_first_search $(v_p,v_q)$;
        let $\Sigma d = d_1 + d_2$, store $\{V_1,V_2\}$
          and reset color;
        let $d_2 = 0$, color $v_q$ red,
          depth_first_search $(v_p,v_q)$;
        if $d_1 + d_2 < \Sigma d$, let $\Sigma d = d_1 + d_2$,
          store $\{V_1,V_2\}$ and reset color;
        while $E_d$ is not empty;
            take $e \in E_d$, let the two vertices
              of $e$ be $u,v$;
            color $u,v$ red, delete $\{u,v\}$ from $E_d$;
            let $d_1 = f(u,v)$, $d_2 = 0$;
            if $d_1 \geq \Sigma d$, skip to endwhile;
                depth_first_search $(u,v)$;
                depth_first_search $(v,u)$;
                if $d_1 + d_2 < \Sigma d$,
                let $\Sigma d = d_1 + d_2$,
                store $\{V_1,V_2\}$ and reset color;
        endwhile;
    end

Procedure **depth_first_search** $(u,v)$
    begin
        if $u$ is red and $f(u,v) > d_1$, color $v$ blue;
            update $d_2$;
        otherwise, color $v$ red;
        for each $w$ adjacent to $v$;
            if $w$ is not colored,
              depth_first_search $(v,w)$;
        endfor
    end
```

This is a myopic type algorithm and proceeds as follows: it first partitions the vertices such that $d_1 = f(v_p,v_q)$. Then, it enumerates the edges in E_d, the set of candidate diameters for d_1. For each edge $e \in E_d$, it colors its vertices red (i.e., assigns them to V_1), sets the diameter equal to its weight, and then visits other vertices by a depth-first-search algorithm; the subsequent vertices are colored red if doing so does not incur a diameter greater than d_1, and blue otherwise. The partition with the least sum of diameters is retained.

Although this algorithm does not guarantee an optimal solution, it is an efficient algorithm, and for large data sets, the gain in speed may well compensate the loss in optimality. When this algorithm is applied to the example given in Hansen and Jaumard (1987), the optimal solution is obtained immediately: the partition with $d_1 = f(v_p,v_q)$ and $d_2 = 1$ is optimal and thus the while-loop, i.e., all the edges in E_d, is skipped.

6. Conclusion

We have studied the problem of data clustering in this paper. Recognizing that either of the two much used criteria, maximum split and minimum diameter, has drawbacks if used alone, we have proposed using the weighted sum of both as the clustering criterion. An algorithm of complexity $O(N^2)$ has been developed to derive the Pareto-efficient set on which an optimal solution can be obtained for any given weighting scheme.

We further extended the criterion to the weighted average of the sum of the diameters and the split of a 2-partition, and devised an efficient approximation algorithm. The merit of our algorithms is that, when they are applied in a divisive hierarchical scheme, the quality of an M-clustering can be improved with manageable computational requirements, and the preference for stressing split or diameter can easily be adjusted.

We should point out, however, that the application of the dominance class in a divisive hierarchical method is not at all straightforward. Different strategies have to be developed in using our algorithms. Although it is not our intention to discuss the full issue here, at least three directions can be pursued: (*i*) fix a value for α and use it at all steps; (*ii*) choosing different values for α as the divisive scheme proceeds; (*iii*) derive the dominance class at each step for arbitrary α. For the first two directions, various criteria need to be developed in choosing the value for α. For the third direction, a manageable size for the problem has to be maintained. For example, some partitions in the dominance class at each step can be discarded. These should all be interesting topics for future research.[3]

Appendix

We prove Theorem 3 in this appendix. The following lemma is a well-known (and easy to prove) property of minimum spanning trees.

3. The authors thank an anonymous referee for inspiring the above comments.

Lemma 1. *Suppose edge $e \notin T_1$ and subgraph C of G is a cycle formed by adding e to T_1. Then*

$$f(e) = \max\{f(u,v) \mid (u,v) \in C\}$$

Proof of Theorem3 We only need to show that lines 8-13 of Algorithm WSDS produce a bipartition P^i that satisfies

$$d(P^i) = \min_{s(P_2) \geq f(e_i^s)} d(P_2). \tag{13}$$

First for $i = 0$, we have $s(P^0) \geq f(e_1^s)$ since $f(e_1^s)$ is the minimum, and $d(P^0) = d^*$. Let $P^1 = P^0$. Therefore, we obtain the first partition satisfying (13) at the beginning of the algorithm.

Now suppose that equation (13) is satisfied for the first $i = k - 1$ iterations, $k = 1, \ldots, N - 1$, and that no two vertices connected with weights less than $f(e_k^s)$ are now separated in two clusters. The current minimum diameter partition has a split $s \geq f(e_k^s)$. Subsequent partitions will be dominated by it unless a greater split is produced. Let u_k, v_k be the vertices of e_k^s. Procedure **merge** puts u_k and v_k in the same cluster. If there is no other edge with the weight $f(e_k^s)$, we may now obtain a new dominant partition with a split $s \geq f(e_{k+1}^s)$. If there is another edge e with weight $f(e_k^s)$, then according to Theorem 2 and Lemma 1, either $e \in T_1$, which will be, if not already, collapsed, or the addition of e to T_1 forms a cycle in which the maximum weight is $f(e_k^s)$. Thus either all the vertices in the cycle are already merged into the same supernode, or the noncollapsed edges in the cycle all have the same weight $f(e_k^s)$, and their vertices will be merged in subsequent iterations. A new dominant partition will be determined after all the edges in the cycle are merged.

Since it is the maximum spanning tree that determines the new dominant partition, if we can show that the maximum spanning tree produced by Procedure **merge** is not affected by the order in which the edges of the same weight are merged, then by induction, the theorem is proved. Suppose there exists an i such that $f(e_i^s) = f(e_{i+1}^s)$. Let us add e_i^s and e_{i+1}^s to T_2. If no cycle or only one cycle is formed, then the order of collapsing the two edges will not affect the construction of the maximum spanning tree. If two cycles are formed, let them be C_1 and C_2. Let $A = C_1 \cap C_2$. If $A = \varnothing$, the order is again irrelevant. Otherwise, let e_1 be the minimum weight edge in $C_1 \backslash e_i^s$, and e_2 the minimum weight edge in $C_2 \backslash e_{i+1}^s$. There are then three possible cases: neither e_1 nor e_2 belongs to A; one of them belongs to A; they both belong to A. Checking all the cases confirms that the procedure results in the same maximum spanning tree regardless of the order of the merging. ∎

References

BERMAN, O., EINAV D., and HANDLER, G. (1990), ''The Constrained Bottleneck Problem in Networks,'' *Operations Research, 38*, 178-181.

BRUCKER, P. (1978), ''On the Complexity of Clustering Problems,'' in *Optimization and Operations Research, Lecture Notes in Economics and Mathematical Systems, 157*, Eds., M. Beckmann and H. Kunizi, Heidelberg: Springer-Verlag, 45-54.

CORMACK, R. M. (1971), ''A Review of Classification,'' *Journal of the Royal Statistical Society, Series A, 134*, 321-367.

DELATTRE, M., and HANSEN, P. (1980), ''Bicriterion Cluster Analysis,'' *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-2*, 277-291.

GOWER, J. C., and ROSS, G. J. S. (1969), ''Minimum Spanning Trees and Single-linkage Cluster Analysis,'' *Applied Statistics, 18*, 54-64.

GUÉNOCHE, A. (1989), ''Partitions with Minimum Diameter,'' Conference of the I.F.C.S., Charlottesville.

GUÉNOCHE, A., HANSEN, P., and JAUMARD, B. (1991), ''Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion,'' *Journal of Classification, 8*, 5-30.

HANSEN, P. (1979), ''Bicriterion Path Problems,'' in *Multiple Criteria Decision Making: Theory and Applications*, Eds., G. Fandel and T. Gal, New York: Springer-Verlag, 109-127.

HANSEN, P., and DELATTRE, M. (1978), ''Complete-link Cluster Analysis by Graph Coloring,'' *Journal of the American Statistical Association, 73*, 397-403.

HANSEN, P., and JAUMARD, B. (1987), ''Minimum Sum of Diameters Clustering,'' *Journal of Classification, 4*, 215-226.

HARMAN, H. H. (1967) *Modern Factor Analysis*, Chicago: University of Chicago Press.

HUBERT, L. J. (1977), ''Data Analysis Implications of Some Concepts Related to the Cuts of a Graph,'' *Journal of Mathematical Psychology, 15*, 199-208.

JOHNSON, S. C. (1967), ''Hierarchical Clustering Schemes,'' *Psychometrika, 32*, 241-254.

MONMA, C., and SURI, S. (1991), ''Partitioning Points and Graphs to Minimize the Maximum or the Sum of Diameters,'' in *Graph Theory, Combinatorics and Applications*, Eds., Y. Alari, G. Chartrand, O.R. Oellerman and A.J. Schwenk, New York: Wiley, 899-912.

RAO, M. R. (1971), ''Cluster Analysis and Mathematical Programming,'' *Journal of the American Statistical Association, 66*, 622-626.

WELCH, J. W. (1983), ''Algorithmic Complexity: Three NP-hard Problems in Computational Statistics,'' *Journal of Statistical Computation and Simulation, 15*, 17-25.

ZAHN, C. T. (1971), ''Graph-theoretical Methods for Detecting and Describing Gestalt Clusters,'' *IEEE Transactions on Computers, C-20*, 68-86.

ZELENY, M. (1982) *Multiple criteria decision making*, New York: McGraw-Hill.