Journal of Classification 1:7-24 (1984)

Efficient Algorithms for Agglomerative Hierarchical Clustering Methods

William H. E. Day
Memorial University of Newfoundland

Herbert Edelsbrunner
Technische Universität Graz

Abstract: Whenever n objects are characterized by a matrix of pairwise dissimilarities, they may be clustered by any of a number of sequential, agglomerative, hierarchical, nonoverlapping (SAHN) clustering methods. These SAHN clustering methods are defined by a paradigmatic algorithm that usually requires $O(n^3)$ time, in the worst case, to cluster the objects. An improved algorithm (Anderberg 1973), while still requiring $O(n^3)$ worst-case time, can reasonably be expected to exhibit $O(n^2)$ expected behavior. By contrast, we describe a SAHN clustering algorithm that requires $O(n^2 \log n)$ time in the worst case. When SAHN clustering methods exhibit reasonable space distortion properties, further improvements are possible. We adapt a SAHN clustering algorithm, based on the efficient construction of nearest neighbor chains, to obtain a reasonably general SAHN clustering algorithm that requires in the worst case $O(n^2)$ time and space.

Whenever n objects are characterized by k-tuples of real numbers, they may be clustered by any of a family of centroid SAHN clustering methods. These methods are based on a geometric model in which clusters are represented by points in k-dimensional real space and points being agglomerated are replaced by a single (centroid) point. For this model, we have solved a class of special packing problems involving point-symmetric convex objects and have exploited it to design an efficient centroid clustering algorithm. Specifically, we describe a centroid SAHN clustering algorithm that requires $O(n^2)$ time, in the worst case, for fixed k and for a family of dissimilarity measures including the Manhattan, Euclidean, Chebychev and all other Minkowski metrics.

Keywords: Algorithm complexity; Algorithm design; Centroid clustering method; Geometric model; SAHN clustering method.

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada and by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung.

Authors' addresses: William H. E. Day, Department of Computer Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1C 5S7. Herbert Edelsbrunner, Institute für Informationsverarbeitung, Technische Universität Graz, Schiessstattgasse 4a, A-8010 Graz, Austria.

1. Introduction

Agglomerative hierarchical clustering methods occupy a prominent position in the science of classification, and for this reason most standard references devote considerable space to their explication and evaluation (Anderberg 1973; Cormack 1971; Everitt 1980; Hartigan 1975; Sneath and Sokal 1973). We follow Sneath and Sokal (1973, pp.202-209, 216) in summarizing generic characteristics of the clustering methods we wish to investigate. They are *agglomerative* methods: starting with a set of n objects to be clustered, they group these objects into successively fewer than n sets, arriving eventually at a single set containing all n objects. They are *hierarchical nonoverlapping* methods that specify a sequence P_0 ,..., P_w of partitions of the objects in which P_0 is the disjoint partition, P_w is the conjoint partition, and P_i is a refinement (in the usual sense) of P_j for all $0 \leqslant i < j \leqslant w$. They are *sequential* methods since the same algorithm is used iteratively to generate P_{i+1} from P_i for all $0 \leqslant i < w$. They are *pair-group* methods: at each iteration exactly two clusters are agglomerated into a single cluster. Pair-group methods are ill-defined in a technical sense if they fail to describe completely the criterion to select clusters for agglomeration when ties exist. Sneath and Sokal (1973) use the acronym SAHN to designate clustering methods that are sequential, agglomerative, hierarchical and nonoverlapping.

SAHN clustering methods require for their operation a quantitative specification of dissimilarity between each pair of objects in the set of objects being clustered. This information is usually provided in one of two forms. In the first, which Anderberg (1973, p.134) calls the *stored matrix* approach, a nonnegative real-valued matrix $\mathbb{D}=[d(x,y)]$ is provided in which $d(x,y)$ is a quantitative measure of dissimilarity between objects x and y. In the second, which Anderberg (1973, p.145) calls the *stored data* approach, each object x is described by a k-tuple $x=(x_1, \ldots, x_k)$ of real numbers, x_i being the score pertaining to the *i-th* variable or character, and a computational rule is specified to calculate from two k-tuples the dissimilarity between the corresponding objects. The *Minkowski* metrics are familiar examples of such rules, where for fixed $p \geqslant 1$ and for any objects x and y,

$$L_p(x,y) = \left\{ \sum_{i=1}^{k} \mid x_i - y_i \mid^p \right\}^{\frac{1}{p}} .$$

This family includes the L_1 or *Manhattan* metric, the L_2 or *Euclidean* metric, and in the limit the L_∞ or *Chebychev* metric for which $L_\infty(x,y) = \max\{ \mid x_i - y_i \mid : 1 \leqslant i \leqslant k \}$.

TABLE 1
Paradigmatic SAHN Clustering Algorithm

Algorithm	Time Complexity
begin	
for m = n **downto** 2 **do**	
begin	
1. Search dissimilarity matrix D for a closest pair (i,j) of clusters.	$O(m^2)$
2. Replace clusters i and j by an agglomerated cluster h.	O(1)
3. Update D to reflect deletion of i and j and to exhibit revised dissimilarities between h and all remaining clusters.	O(m)
end	
Output the hierarchy of agglomerated clusters.	
end	

NOTE: Initially each of the n objects to be clustered is in a cluster by itself. In step 1 of each loop iteration, the dissimilarity matrix D has a row and a column for each of the m remaining clusters. The number of clusters decreases by one for steps 2 and 3. When step 3 completes, the revised matrix D has a row and a column for each of the (m-1) remaining clusters.

Many SAHN clustering methods can be defined algorithmically simply by providing a precise specification of step 3 in the paradigmatic algorithm shown in Table 1. Johnson (1967), Wishart (1969), Anderberg (1973, p.133) and others use this approach to define SAHN clustering methods; but the idea is not without disadvantages. Jardine and Sibson (1971, p.42) warn against confusing clustering algorithms with clustering methods. Rohlf (1982) describes cases in which proposals for new clustering methods simply turn out to be new algorithms for the single linkage SAHN clustering method.

Lance and Williams (1966, 1967) define the class of *combinatorial* SAHN clustering methods by specifying a generalized recurrence formula to perform the updating required by step 3 in Table 1. Suppose clusters i and j are being replaced by the agglomerated cluster h. Lance and Williams propose that the revised dissimilarity between h and any other cluster k be defined by the recurrence

$$d(h,k) = \alpha_i d(i,k)+\alpha_j d(j,k)+\beta d(i,j)+\gamma \mid d(i,k)-d(j,k) \mid \qquad (1)$$

where parameter values α_i, α_j, β and γ characterize the particular combinatorial clustering method. Table 2 exhibits these parameter values for eight well-known combinatorial SAHN clustering methods. This formulation of combinatorial clustering methods has stimulated research in several directions. Milligan (1979) and Batagelj (1981) derive conditions on parameter values that are necessary and sufficient for a combinatorial clustering method to have its partitions generated at monotonically increasing dissimilarity values. Hubert and Schultz (1975) and DuBien and Warde (1979) classify combinatorial clustering methods by the type and degree of object space distortion (Lance and Williams 1967) caused by the corresponding combinatorial algorithm.

In order to evaluate time and space complexities of clustering algorithms, we measure problem size by the number n of objects to be clustered, and we describe an algorithm's *time complexity* (respectively, *expected time complexity*) by a function $f(n)$ expressing, for each n, the largest (respectively, average) amount of time the algorithm needs to solve any problem instance of size n. Space complexity is defined analogously. In describing the asymptotic behavior of such positive valued functions, we say that $f(n)$ is $0(g(n))$ whenever there exists a positive constant c such that $f(n) \leqslant c \cdot g(n)$ for all large positive n; $f(n)$ is $\Omega(g(n))$ if $g(n)$ is $0(f(n))$. For further information concerning the analysis of algorithm complexity, the reader can consult the classic textbook by Aho, Hopcroft and Ullman (1974) or the fine survey by Weide (1977).

For combinatorial SAHN clustering methods, the Table 1 algorithm is specified in enough detail to enable us to estimate its time and space complexities. Let m denote the number of clusters remaining at each execution of the loop. In any reasonable implementation of this algorithm, step 1 searches $\binom{m}{2}$ matrix entries and therefore requires $0(m^2)$ time. Since step 1 dominates each loop execution, it follows that both expected and worst-case time complexities for this algorithm are $0(n^3)$. We adopt the convention that an algorithm's input cannot be destroyed; consequently the space complexity for any reasonable implementation of this combinatorial SAHN algorithm is both $\Omega(n^2)$ and $0(n^2)$.

TABLE 2

Parameter Values for Combinatorial SAHN Clustering Methods

Clustering Method	α_i	α_j	β	γ
Single linkage (Nearest neighbor)	1/2	1/2	0	-1/2
Complete linkage (Furthest neighbor)	1/2	1/2	0	1/2
Group Average (UPGMA)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Weighted Average (WPGMA)	1/2	1/2	0	0
Unweighted Centroid (UPGMC)	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{-n_i n_j}{(n_i+n_j)^2}$	0
Weighted Centroid (WPGMC)	1/2	1/2	-1/4	0
Minimum Variance (Ward)	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	$\frac{-n_k}{n_i+n_j+n_k}$	0
Flexible (Lance and Williams)	$\frac{1-\beta}{2}$	$\frac{1-\beta}{2}$	$\beta<1$	0

NOTE: n_i is the number of objects in cluster i.

Anderberg (1973, pp.135-136) modifies the design of the Table 1 combinatorial SAHN algorithm in order to improve its expected behavior (see also Hartigan 1975, pp.216-218). The basic idea is to improve the expected time complexity of step 1 (the execution bottleneck) by retaining information from one iteration to the next. Anderberg introduces an n-tuple $P = (P_1, \ldots, P_n)$ of indices that identify for each cluster i a nearest neighboring cluster P_i: in an iteration where **D** has a row and column for each of m remaining clusters, each P_i for $1 \leqslant i \leqslant m$ satisfies the condition that $d(i,P_i) = \min\{d(i,j) : 1 \leqslant j \leqslant m, i \neq j\}$. Table 3 exhibits Anderberg's modified algorithm. At each iteration, P simplifies the step 1 search for a closest pair (i,j) of clusters; but now whenever $P_k = i$ or $P_k = j$, step 4

TABLE 3

Combinatorial SAHN Clustering Algorithm (Anderberg, 1973)

Algorithm		Time Complexity
begin		
0.	Initialize n-tuple P to identify a nearest neighbour of each cluster.	
	for m = n downto 2 do	
	begin	
1.	Search D with P to identify a closest pair (i,j) of clusters.	O(m)
2.	Replace clusters i and j by an agglomerated cluster h.	O(1)
3.	Update D to reflect deletion of i and j and to exhibit revised dissimilarities between h and all remaining clusters.	O(m)
4.	Update P to reflect deletion of i and j and inclusion of h.	$O(m^2)$
	end	
	Output the hierarchy of agglomerated clusters.	
end		

must update P_k by a process requiring $O(m)$ time. Since at each iteration the number of such updates lies between zero and $(m-2)$, the time complexity of the Table 3 algorithm is both $\Omega(n^2)$ and $O(n^3)$. If, as Anderberg speculates, step 4 averages a constant number of updates per iteration, then the Table 3 combinatorial SAHN algorithm exhibits $O(n^2)$ expected time complexity.

The Table 1 combinatorial SAHN algorithm can be altered to improve its worst-case behavior. In section 2 we describe a new algorithm that makes extensive use of priority queues to identify cluster nearest neighbors.

The time complexity for this algorithm is $0(n^2 \log n)$; consequently its time complexity is asymptotically superior to those for the combinatorial SAHN algorithms of Tables 1 and 3. Further improvements are possible for combinatorial SAHN clustering methods that exhibit reasonable degrees of object space distortion (Lance and Williams 1967). In section 2 we adapt an algorithm based on nearest neighbor chains (Benzécri 1982; Juan 1982; Murtagh 1983) to obtain a reasonably general combinatorial SAHN clustering algorithm with $0(n^2)$ time and space complexities.

Further improvements in the complexities of SAHN clustering methods lie in either (or both) of two general categories: they exploit characteristics of particular clustering methods; or they exploit characteristics of data descriptions of the objects being clustered. Sibson (1973) describes a single linkage SAHN algorithm based on the efficient extension of a hierarchical clustering of m objects to one of $(m+1)$ objects; the algorithm requires $0(n^2)$ time and $0(n)$ space. Defays (1977) uses this approach to obtain a complete linkage SAHN algorithm with the same asymptotic behavior. Gower and Ross (1969) establish that single linkage SAHN algorithms can be based on algorithms to construct minimum spanning trees (as in Ross 1969). Rohlf (1973) uses this approach to obtain a single linkage SAHN algorithm requiring $0(n^2)$ time and $0(n)$ space. The single linkage SAHN clustering method has, of course, been studied extensively; Rohlf (1982) gives an illuminating classification and evaluation of eleven different algorithms for this clustering method.

Considerable attention has been given to the interpretation of objects in the stored data approach as points in a k-dimensional real space. Murtagh (1983) gives a fine presentation of SAHN clustering algorithms based on this approach. Rohlf (1978) attempts to avoid completely the computation of interpoint dissimilarities that are irrelevant to the agglomeration process; his single linkage SAHN algorithm exhibits $0(n \log \log n)$ expected time complexity for favorable data sets. Rohlf (1977) extends this approach to obtain unweighted and weighted centroid SAHN algorithms with similar asymptotic behavior. For points in two-dimensional real space, there exist efficient single-linkage SAHN algorithms based on minimum spanning tree construction: algorithms requiring $0(n \log n)$ time and $0(n)$ space are described by Shamos and Hoey (1975) if the Euclidean metric is used to measure interpoint dissimilarity, and by Hwang (1979) if the Chebychev metric is used. Murtagh (1983) describes a minimum variance SAHN algorithm (Ward 1963) based on the efficient construction of nearest neighbor chains; the algorithm requires $0(n^2)$ time and $0(n)$ space. This approach also can be used to provide $0(n^2)$ approximation algorithms for the unweighted and weighted centroid SAHN clustering methods.

Anderberg (1973, p.146) describes a variation of the Table 3 algorithm that applies when objects in the stored data approach are treated as points in a k-dimensional real space, and when a *centroid* strategy is used in which

TABLE 4
Centroid SAHN Clustering Algorithm (Anderberg, 1973).

Algorithm	Time Complexity
begin	
0. Initialize the n-tuple P to identify a nearest neighbor of each cluster.	
for m = n **downto** 2 **do**	
begin	
1. Use P to identify a closest pair (i,j) of clusters.	O(km)
2. Replace clusters i and j by an agglomerated cluster h.	O(k)
3. Update P to reflect deletion of i and j and inclusion of h.	O(α km)
end	
Output the hierarchy of agglomerated clusters.	
end	

NOTE: Each object or cluster, including the agglomerated cluster h, is represented by a point in a k-dimensional space. The constant α is an upper bound on the maximum number of updates required of elements in P. Section 4 describes the assumptions being made for this model of centroid SAHN clustering methods.

points being agglomerated are replaced by a single point. Table 4 exhibits Anderberg's centroid algorithm. When k is fixed, step 3 becomes the execution bottleneck, and a straightforward analysis establishes that the Table 4 algorithm has $O(n^3)$ time complexity. However, in a related development, we solve in section 3 a class of special geometric packing problems involving point-symmetric convex objects. In section 4 we exploit these results to establish general conditions under which the Table 4 centroid SAHN algorithm requires $O(n^2)$ time and $O(n)$ space. In particular, the unweighted and weighted centroid SAHN clustering methods have algorithms requiring $O(n^2)$ time and $O(n)$ space when they use any of a family of dissimilarity

measures including the Manhattan, Euclidean, Chebychev and all other Minkowski metrics.

2. A Combinatorial SAHN Clustering Algorithm

In the Table 3 combinatorial SAHN algorithm, step 4 is an execution bottleneck: it requires $0(m)$ cluster nearest neighbor updates, and each requires $0(m)$ time. Two strategies are available to improve the algorithm's time complexity: obtain an improved bound on the required number of nearest neighbor updates (an approach used in section 4 for centroid SAHN clustering methods); or obtain an improved bound on the time required for each update (the approach used here). A straightforward update implementation searches $0(m)$ intercluster dissimilarities to find a nearest neighbor in $0(m)$ time. However, if intercluster dissimilarities are maintained in a priority queue, a standard priority queue operation finds a nearest neighbor in just $0(\log m)$ time. The algorithm we propose is based on this observation.

A *priority queue* is a data structure for a finite set of elements. Associated with each element is a label selected (in our case) from the nonnegative real numbers. The elements in a priority queue are available in an order determined by their labels. Priority queues are manipulated with three operations. INSERT(a,b,Q) associates label b with element a and adds a to priority queue Q. DELETE(a,Q) removes element a from priority queue Q. MIN(Q) identifies an element with least label in priority queue Q. A priority queue of m elements can be implemented as a *heap* data structure requiring $O(m)$ time for its initialization; each subsequent MIN requires O(1) time, while subsequent INSERT and DELETE operations each require $O(\log m)$ time. For details the reader can consult Aho, Hopcroft and Ullman (1974, pp.87-92, 148-152) or any standard data structure text.

Table 5 exhibits a combinatorial SAHN clustering algorithm in which a priority queue is associated with each cluster. These priority queues are manipulated in steps 0, 1 and 4. Since each of the n priority queues in step 0 requires $O(n)$ initialization time, step 0 requires $0(n^2)$ time. Consider the iteration where m clusters remain. Step 1 is accomplished by m MIN's and so requires $0(m)$ time. Step 4 requires two DELETE's and one INSERT to update each of $0(m)$ priority queues in $0(\log m)$ time; it also requires $0(m)$ time to initialize a priority queue for the agglomerated cluster. Since step 4 is an execution bottleneck requiring $0(m \log m)$ time, the entire algorithm has $0(n^2 \log n)$ time complexity. Since each of the n priority queues requires $0(n)$ space, the algorithm has $0(n^2)$ space complexity.

The combinatorial SAHN clustering algorithm in Table 5 is valuable because it is insensitive to distortions of the object space that occur for methods defined by particular values of α_i, α_j, β and γ. Further improvements in time complexity are possible for combinatorial SAHN clustering methods that satisfy a special requirement. A clustering method is said to

TABLE 5
Combinatorial SAHN Clustering Algorithm Based on Priority Queues.

Algorithm	Time Complexity
begin	
0. Construct for each cluster a priority queue of its (n-1) neighbors ordered by inter-cluster dissimilarity.	
for m = n **downto** 2 **do**	
begin	
1. Interrogate the m priority queues to identify a closest pair (i,j) of clusters.	O(m)
2. Replace clusters i and j by an agglomerated cluster h.	O(1)
3. Update D to reflect deletion of i and j and to exhibit revised dissimilarities between h and all remaining clusters.	O(m)
4. Update the priority queues to reflect deletion of i and j and inclusion of h.	O(m log m)
end	
Output the hierarchy of agglomerated clusters.	
end	

satisfy the *reducibility property* (Bruynooghe 1978) if, when clusters i and j are agglomerated to obtain cluster h, $d(h,k) \geqslant \min\{d(i,k),d(j,k)\}$ for every other cluster k. Murtagh (1983) describes a stored data SAHN clustering algorithm (see also Benzécri (1982) and Juan (1982)) requiring that the clustering method satisfy the reducibility property and that intercluster distances be computable in constant time. The algorithm is based on an efficient construction of chains of nearest neighboring clusters; it requires $O(n^2)$ time and $O(n)$ space. When combinatorial SAHN clustering methods

satisfy the reducibility property, Murtagh's algorithm can easily be adapted to them by modifying it to maintain, using equation (1), a matrix of inter-cluster distances. This modification of Murtagh's algorithm requires $0(n^2)$ time and space; its time complexity is superior to those of algorithms presently available for the group average (UPGMA), weighted average (WPGMA) and flexible combinatorial SAHN clustering methods.

3. Packing Problems in k Dimensions

This section is devoted to the development of geometric preliminaries which are exploited in section 4 in the design of efficient centroid SAHN clustering algorithms. In particular, a class of special packing problems involving point-symmetric convex objects is considered.

Let (R^k,d) denote the k -dimensional real space together with the dissimilarity function $d : R^k \times R^k \rightarrow R_0^+$. We require that

(G1) $d(x,y) = d(0,y-x)$ with 0 the origin of R^k;

(G2) $d(\lambda x,\lambda y) = \lambda \cdot d(x,y)$ for any nonnegative real number λ;

(G3) the *unit ball* $B(0,1) = \{x : d(0,x) \leqslant 1\}$ is symmetric at 0, convex and with proper volume.

Recall that a set B in R^k is *convex* if for any points x and y in B, the line segment connecting x and y is contained in B; B has *proper volume* if it contains a ball $B(x,\epsilon)$ for some positive real ϵ. Conditions (G1) and (G2) are required to move and to change the size of point configurations in (R^k,d). With respect to (G3), observe that $B(0,1)$ is symmetric at 0 if and only if d is symmetric (i.e., $d(x,y) = d(y,x)$), while $B(0,1)$ is convex if and only if d satisfies the triangle inequality (i.e., $d(x,z) \leqslant d(x,y) + d(y,z)$). The requirement that $B(0,1)$ have proper volume avoids degenerate and uninteresting cases. Well-known examples of dissimilarity functions satisfying (G1) - (G3) include the Manhattan, Euclidean, Chebychev and all other Minkowski metrics.

Centroid SAHN clustering algorithms use finite sets of points in (R^k,d) and identify pairs of points which are close in a sense peculiar to the algorithm involved. Consider algorithms that store for each point a closest point. During algorithm execution, points are deleted and points are inserted into the set. These insertions and deletions make necessary the adjustment of closest points. We now establish a geometric result which shows that efficient adjustment is possible.

Let P denote a finite set of sufficiently many points in (R^k,d). A point x in P is called a *nearest neighbor of y in P* if $d(y,x) \leqslant d(y,z)$ for all $z \neq y$ in P. We call $A_P(x) = \{y \in P : x \text{ is a nearest neighbor of } y\}$ the *attracted set of x in P* and $| A_P(x) |$ (i.e., the cardinality of the attracted set)

the *attractive power of x in P*. The remainder of this section concerns how large the attractive power of a point in P can be. It seems intuitively plausible that the attractive power of a point has an upper bound which depends on k and which is independent of the cardinality of P. This fact will be made concrete and an optimal upper bound, following from geometric packing results, will be established.

Let $P_x = \{x\} \cup A_P(x)$ denote the subset of P including with x its attracted points. Without loss of generality let $d(x,y) = 1$ for y in P_x that maximizes the distance to x. Define $x' = 0$ and $z' = (z-x)/d(x,z)$ for each z in $P_x - \{x\}$.

Lemma 3.1 *$d(u',v') \geq 1$ for any points u and v in $P_x - \{x\}$.*

Proof. Assume $d(u',v')<1$ for some particular points u and v in $P_x - \{x\}$. If $d(x,u) = d(x,v)$ then $d(u,v) = d(u',v')\cdot d(x,v) < d(x,v)$, a contradiction of the assumption that u and v are in $P_x - \{x\} = A_P(x)$. Hence, assume without loss of generality that $d(x,u)<d(x,v)$. Define $u'' = x + (u-x)\cdot d(x,v)/d(x,u)$; see also Figure 1. Then $d(x,u'') = d(x,v)$ and thus $d(u'',v)=d(u',v')\cdot d(x,v)<d(x,v)$. But then by convexity of $B(v,d(v,x))$ it follows that $d(u,v)<d(v,x)$, a contradiction. ●

Theorem 3.2 *Let P denote a set of points in (R^k,d) such that $P=\{x\} \cup A_P(x)$. Then there exists a set Q in (R^k,d) such that*

(i) *$Q=\{y\} \cup A_Q(y)$, and $d(y,u)=1$ for any u in $A_Q(y)$;*

(ii) *$|Q|=|P|$;*

(iii) *$d(u,v) \geq 1$ for any two points u and v in $Q-\{y\}$.*

Proof. We construct Q from P. Define $x' = 0$ and $z' = (z-x)/d(x,z)$ for each z in $P - \{x\}$. Now define $Q = \{u' : u \in P\}$. Lemma 3.1 implies condition (iii) for $y = x'$, and by construction $d(y,u') = 1$ for any u' in $Q - \{y\}$; thus condition (i) holds. Condition (ii) is true as no two points in Q are identical. ●

Thus, for developing an upper bound on the attractive power of a point we can restrict ourselves to sets Q satisfying the conditions of Theorem 3.2. For each point u in Q let $B(u) = B(u,1/2)$ be called the *ball of u*. Clearly, no two balls overlap but some may touch; in fact $B(y)$ touches each of the other balls of points in Q. Observe that the balls are translates of each other and, in particular, that each ball is a translate of $B(y)$. Observe also that for each set Q there is such a configuration of balls, and for each such configuration of balls the set Q of centers satisfies the conditions of

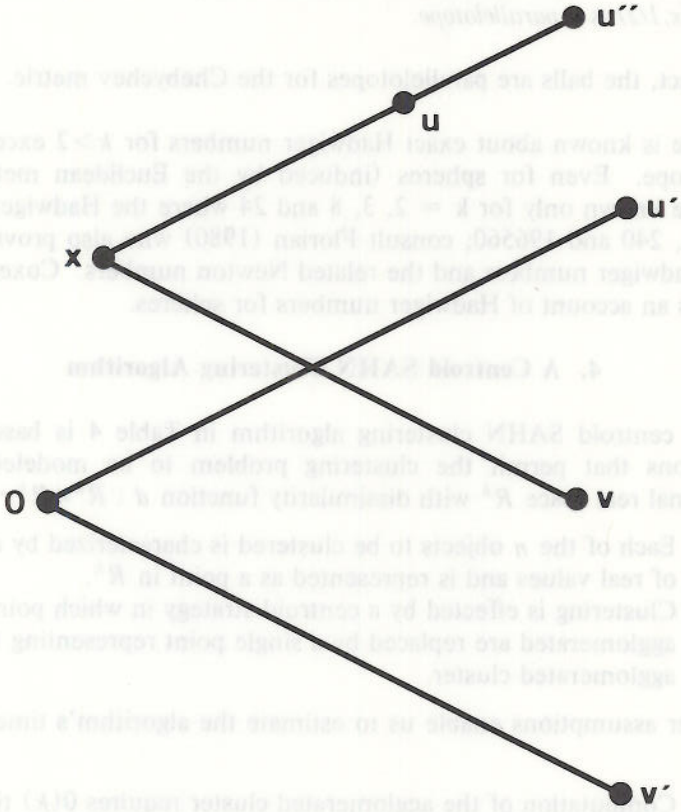

Figure 1: Illustration, in the Euclidean Plane, Relevant to the Proof of Lemma 3.1.

Theorem 3.2. Thus the problem of finding an upper bound on the attractive power of a point in (R^k, d) is equivalent to finding the maximum number of nonoverlapping translates of $B(y)$ such that each of them touches $B(y)$. The latter number is well-known in discrete geometry as the *Hadwiger number H(B(y)) of B(y).*

Hadwiger and Debrunner (1955) first mentioned the problem of finding (bounds for) Hadwiger numbers. Since then an extensive literature has been devoted to the problem. We will cite those results having the most important implications for the attractive power of points.

Proposition 3.3 *(Hadwiger 1957; Groemer 1961; Grünbaum 1961). Let B denote a convex body with proper volume in* R^k. *Then* $H(B) \leqslant 3^k - 1$, *and* $H(B) = 3^k - 1$ *if and only if B is a parallelotope.*

Corollary 3.4 *Let P denote a finite set of points in (R^k,d). Then $|A_P(x)| \leqslant 3^k-1$ for any point x in P, and the upper bound is achieved if and only if $B(x,1/2)$ is a parallelotope.*

In fact, the balls are parallelotopes for the Chebychev metric.

Little is known about exact Hadwiger numbers for $k>2$ except for the parallelotope. Even for spheres (induced by the Euclidean metric) exact values are known only for k = 2, 3, 8 and 24 where the Hadwiger numbers are 6, 12, 240 and 196560; consult Florian (1980) who also provides a survey of Hadwiger numbers and the related Newton numbers. Coxeter (1963) also gives an account of Hadwiger numbers for spheres.

4. A Centroid SAHN Clustering Algorithm

The centroid SAHN clustering algorithm in Table 4 is based on two assumptions that permit the clustering problem to be modeled in a k-dimensional real space R^k with dissimilarity function $d : R^k \times R^k \to R_0^+$.

(A1) Each of the n objects to be clustered is characterized by a k-tuple of real values and is represented as a point in R^k.
(A2) Clustering is effected by a centroid strategy in which points being agglomerated are replaced by a single point representing the agglomerated cluster.

Two other assumptions enable us to estimate the algorithm's time complexity.

(A3) Computation of the agglomerated cluster requires $0(k)$ time.
(A4) Computation of $d(x,y)$ requires $0(k)$ time for any points x and y in R^k.

Step 0 is accomplished in $0(kn^2)$ time by computing $0(n^2)$ dissimilarities. Consider the iteration when m points remain. Step 1 is accomplished in $0(km)$ time by computing $0(m)$ dissimilarities. Step 2 clearly requires $0(k)$ time to replace points i and j by the agglomerated point h. Step 3 requires several types of updates of P. The dissimilarities between h and the remaining points must be calculated in order to initialize P_h and to update P_x whenever h becomes the new nearest neighbor of any point x. These updates require $O(km)$ time. Now let α denote the maximum number of points in R^k that have either i or j as a nearest neighbor. Each point counted by x represents a possible update of P that can be accomplished in $O(km)$ time by computing $O(m)$ dissimilarities. Since step 3 is dominated by updates of this latter type, it requires $0(\alpha km)$ time. Since step 3 is an execution bottleneck at each iteration, the complete algorithm has $0(\alpha kn^2)$ time complexity.

Three final assumptions about d enable us to bound α in terms of k.

Lemma 4.1 *If d satisfies (G1) - (G3), then $\alpha \leqslant 2\ (3^k-2)$.*

Proof. Immediate from Corollary 3.4. ●

It follows from Lemma 4.1 that when (A1) - (A4), (G1) - (G3) hold and when k is fixed by the design of the clustering experiment, the Table 4 centroid SAHN clustering algorithm has $0(n^2)$ time complexity and $0(n)$ space complexity.

5. Conclusion

We have described efficient algorithms for two important classes of SAHN clustering methods. For problems using the stored matrix approach, we describe in section 2 a general combinatorial SAHN clustering algorithm with $0(n^2 \log n)$ time complexity and $0(n^2)$ space complexity, and a reasonably general combinatorial SAHN clustering algorithm with $0(n^2)$ time and space complexities. To our knowledge, this latter algorithm exhibits a time complexity superior to that of any algorithm previously proposed for the group average (UPGMA), weighted average (WPGMA) and flexible (Lance and Williams 1967) combinatorial SAHN clustering methods. For problems using the stored data approach and any of a large family of dissimilarity measures, we describe in section 4 a general centroid SAHN clustering algorithm with $0(n^2)$ time complexity and $0(n)$ space complexity. Centroid methods such as unweighted centroid (UPGMC) and weighted centroid (WPGMC) now join single linkage, complete linkage and minimum variance (Ward 1963) in the list of SAHN clustering methods with algorithms requiring $0(n^2)$ time and $0(n)$ space.

Three research topics are suggested by this paper.

Our results in section 4 suggest that further improvements to centroid SAHN clustering methods may be possible using the stored data approach. For example, it is of interest to know whether algorithms exist that run in asymptotically less than $0(n^2)$ time when each object is specified by two real numbers. Methods from computational geometry (a field in computer science concerned with algorithmic aspects of geometric problems) are probably relevant; Edelsbrunner and van Leeuwen (1983) give an extensive bibliography in this field.

It is desirable to understand properties of clusters and classifications generated by SAHN clustering methods. Such properties may be of particular interest if clusters and intercluster dissimilarities have geometric interpretations. For example, in two dimensions Shamos and Hoey (1975) exploit

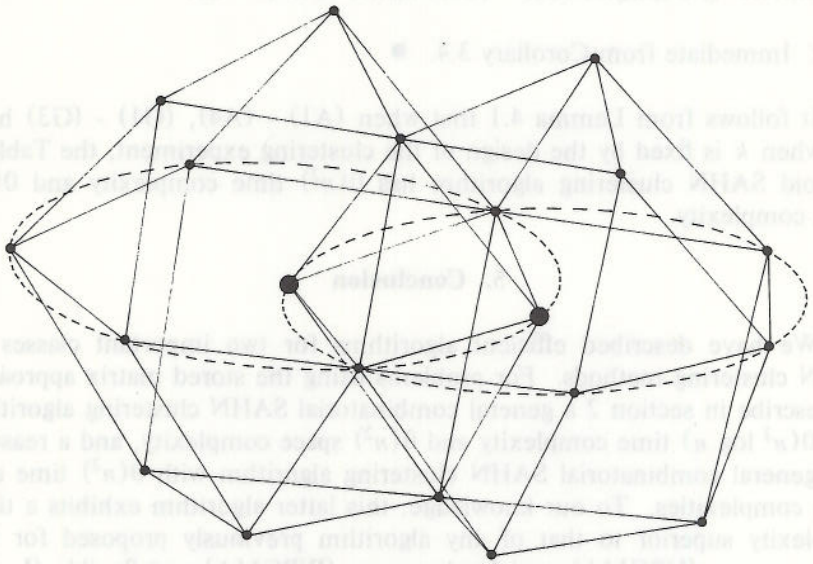

Figure 2: Packing Euclidean Balls Around Two Touching Balls in Three Dimensions.

methods of computational geometry and the well-known relationship between minimum spanning trees and single linkage SAHN classifications to obtain an $0(n \log n)$ single linkage SAHN clustering algorithm.

Our results in sections 3 and 4 suggest a purely geometric packing problem in real space. Let B be a point-symmetric convex body with proper volume in R^k, and let B' be a translate of B that touches B. Give bounds for the maximum number α of nonoverlapping translates of B that touch at least one of B and B' but do not overlap either. Lemma 4.1 gives an easy upper bound that follows from the Hadwiger number of B. It is rather easy to see that $\alpha = 2$ for B an interval in R^1, and that $\alpha = 8$ for B a disk in R^2. We conjecture that $\alpha = 18$ for B a Euclidean ball in R^3; Figure 2 depicts in three dimensions a configuration of centers of twenty balls that realize this bound.

References

AHO, A.V., HOPCROFT, J.E., and ULLMAN, J.D. (1974), *The Design and Analysis of Computer Algorithms*, Reading, Massachusetts: Addison-Wesley.

ANDERBERG, M.R. (1973), *Cluster Analysis for Applications*, New York: Academic.

BATAGELJ, V. (1981), "Note on Ultrametric Hierarchical Clustering Algorithms," *Psychometrika*, 46, 351-352.

BENZECRI, J.P. (1982), "Construction d'une Classification Ascendante Hiérarchique par la Recherche en Chaîne des Voisins Réciproques," *Les Cahiers de l'Analyse des Données*, VII, 209-218.

BRUYNOOGHE, M. (1978), "Classification Ascendante Hiérarchique des Grands Ensembles de Données: un Algorithme Rapide Fondé sur la Construction des Voisinages Réductibles," *Les Cahiers de l'Analyse des Données*, III, 7-33.

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

COXETER, H.S.M. (1963), "An Upper Bound for the Number of Equal Nonoverlapping Spheres that can Touch Another of the Same Size," in *Proceedings of the Symposium in Pure Mathematics VII, Convexity*, ed. V. Klee, Providence, Rhode Island: American Mathematical Society, 53-71.

DEFAYS, D. (1977), "An Efficient Algorithm for a Complete Link Method," *Computer Journal*, 20, 364-366.

DUBIEN, J.L., and WARDE, W.D. (1979), "A Mathematical Comparison of the Members of an Infinite Family of Agglomerative Clustering Algorithms," *Canadian Journal of Statistics*, 7, 29-38.

EDELSBRUNNER, H., and VAN LEEUWEN, J. (1983), "Multidimensional Data Structures and Algorithms - a Bibliography," Report F 104, Institute für Informationsverarbeitung, Technische Universität Graz, Graz, Austria.

EVERITT, B. (1980), *Cluster Analysis* (Second Edition), London: Heinemann.

FLORIAN, A. (1980), "Newtonsche und Hadwigersche Zahlen," Report 145, Institut für Mathematik, Universität Salzburg, Salzburg, Austria.

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

GROEMER, H. (1961), "Abschätzungen für die Anzahl der konvexen Körper, die einen konvexen Körper berühren," *Monatshefte für Mathematik*, 65, 74-81.

GRUNBAUM, B. (1961), "On a Conjecture of H. Hadwiger," *Pacific Journal of Mathematics*, 11, 215-219.

HADWIGER, H. (1957), "Uber Treffanzahlen bei translationsgleichen Eikörpern," *Archiv der Mathematik*, 8, 212-213.

HADWIGER, H., and DEBRUNNER, H. (1955), *Kombinatorische Geometrie in der Ebene*, Genève: Monographies de l'Enseignement Mathématique.

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

HUBERT, L., and SCHULTZ, J. (1975), "Hierarchical Clustering and the Concept of Space Distortion," *British Journal of Mathematical and Statistical Psychology*, 28, 121-133.

HWANG, F.K. (1979), "An 0(n log n) Algorithm for Rectilinear Minimal Spanning Tree," *Journal of the Association for Computing Machinery*, 26, 177-182.

JARDINE, N., and SIBSON, R. (1971), *Mathematical Taxonomy*, London: John Wiley.

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

JUAN, J. (1982), "Programme de Classification Hiérarchique par l'Algorithme de la Recherche en Chaîne des Voisins Réciproques," *Les Cahiers de l'Analyse des Données*, VII, 219-225.

LANCE, G.N., and WILLIAMS, W.T. (1966), "A Generalized Sorting Strategy for Computer Classifications," *Nature*, 212, 218.

LANCE, G.N., and WILLIAMS, W.T. (1967), "A General Theory of Classificatory Sorting Strategies. 1. Hierarchical Systems," *Computer Journal*, 9, 373-380.

MILLIGAN, G.W. (1979), "Ultrametric Hierarchical Clustering Algorithms," *Psychometrika*, 44, 343-346.

MURTAGH, F. (1983), "A Survey of Recent Advances in Hierarchical Clustering Algorithms," *Computer Journal*, 26, 354-359.

ROHLF, F.J. (1973), "Algorithm 76. Hierarchical Clustering Using the Minimum Spanning Tree," *Computer Journal*, 16, 93-95.

ROHLF, F.J. (1977), "Computational Efficiency of Agglomerative Clustering Algorithms," Report RC 6831, IBM T.J. Watson Research Center, Yorktown Heights, New York.

ROHLF, F.J. (1978), "A Probabilistic Minimum Spanning Tree Algorithm," *Information Processing Letters*, 7, 44-48.

ROHLF, F.J. (1982), "Single-link Clustering Algorithms," in *Handbook of Statistics, Vol. 2*, eds. P.R. Krishnaiah and L.N. Kanal, Amsterdam and New York: North Holland, 267-284.

ROSS, G.J.S. (1969), "Algorithm AS 15. Single Linkage Cluster Analysis," *Applied Statistics*, 18, 106-110.

SHAMOS, M.I., and HOEY, D. (1975), "Closest-point Problems," *Sixteenth Symposium on Foundations of Computer Science*, New York: Institute of Electrical and Electronics Engineers, 151-162.

SIBSON, R. (1973), "SLINK: an Optimally Efficient Algorithm for the Single-link Cluster Method," *Computer Journal*, 16, 30-34.

SNEATH, P.H.A., and SOKAL, R.R. (1973), *Numerical Taxonomy*, San Francisco: W.H. Freeman.

WARD, Jr., J.H. (1963), "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association*, 58, 236-244.

WEIDE, B. (1977), "A Survey of Analysis Techniques for Discrete Algorithms," *Computing Surveys*, 9, 291-313.

WISHART, D. (1969), "256 Note: An Algorithm for Hierarchical Classifications," *Biometrics*, 25, 165-170.