The Visual Computer

Clustering methods for geometric objects and applications to design problems

F. Dehne and H. Noltemeier

Informatik I, University of Würzburg,
Am Hubland, D-8700 Würzburg,
Federal Republic of Germany

Clustering of geometric objects is a very familiar and important problem in many different areas of applications as well as in the theoretical foundation of some modern fields of computer science. This paper describes how design problems, especially the design of an assembly line, can be transformed into a clustering problem. In order to solve the problem for large sizes of input data we introduce a structure, called Voronoi Tree, which applied to our real world data (assembly line design) did not only reduce the time to get a feasible design of an assembly line dramatically, but additionally increased the value of the design by more than 30% (in comparison with standard design methods). In addition to this we introduce a clustering method which is of interest for those applications which can be transformed to planar clustering problems. In this particular case it is possible to compute an (hierarchically) optimized clustering with resp. to a large class of clustering measures in time $O(n\,n^{1/2}\log^3 n + U_F(n)\,n\,n^{1/2} + P_F(n))$ [n: number of points; $U_F(n)$, $P_F(n)$ dependent on the chosen clustering measure].

Key words: Computational geometry – Clustering – Layout problems

Clustering of geometric objects like
- sets of points in a wide variety of different spaces
- sets of edges or polygons
- sets of polyhedrons or even more complex 'geometric objects' like 'frames' etc.

is a very familiar and important problem in many different areas of applications as well as in the theoretical foundation of some modern fields of computer science (i.e. artificial intelligence). By clustering we mean partitioning and/or aggregation of sets of 'geometric objects' with respect to some given criteria (constraints, objective functions etc.) which may be very simple in some circumstances but can be very complicated in other cases which we will sketch later on, too.

To give a first very special but common example of aggregation and partitioning methods remember the well known convex hull operation (Preparata and Hong 1977). Figure 1 illustrates a set of points in the plane (a). A minimal convex aggregation leads to the convex hull (b) of the point set whereas a more sophisticated (parameterized) modification generates the shape hull (c) which in some cases gives us more insight into the structure of the point set (Edelsbrunner et al. 1981).

Very recently (Dehne 1985) significant advances were done in the area of clustering more complex objects (scenes of edges, polygons, polyhedrons, disjoint sets of compact objects in general, etc.).

In general clustering of geometric objects is an important tool for
- pattern recognition
- pattern matching
- image processing
- image understanding
- motion analysis

as well as for numerous layout and design problems:
- VLSI placement problems
- board design
- design of assembly lines and manufacturing processes in general et al.

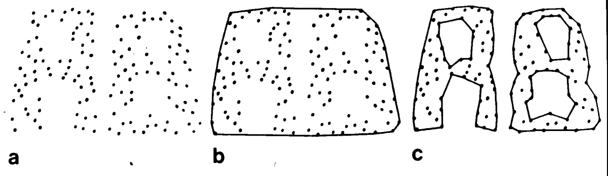


Fig. 1

In the following we will illustrate our ideas by a (simplified) example of an assembly line design. Section 2 will introduce the design problem, which will be transformed into a geometric framework in Sect. 3. To finally solve the problem we present a new structure called Voronoi tree in Sect. 4 and summarize our experimental results.
In the second part of our paper (Sect. 5) we will briefly introduce one more clustering method which is particularly important for those applications (some are listed above) which can be transformed into planar clustering problems. This clustering method has the additional advantage that it generates an (hierarchically) optimized clustering with resp. to any given (efficiently computable) clustering measure.

2. A (simplified) design problem

In order to give the reader an idea how clustering methods can support the solution of *design problems* let us try to illustrate our methods by an example of an assembly line design.
Consider that a wide spectrum of products ('groups') has to be assembled on an assembly line and that each product consists of a multiset of parts ('elements') which is a subset of one large set of elements.
Let $E=\{e_1,\dots,e_m\}$ denote the set of all elements and $G=\{g_1,\dots,g_n\}$ the set of groups (types of products). We assume that the production data is given by a 'Gozinto graph' (which element belongs to which group) as illustrated in Fig. 2 with some integer values m_{ij} assigned to the arcs (which may represent the number of occurences of element e_i in group g_j) and some nonnegative weights h_j assigned to the groups (which may represent the frequency of production of g_j in the actual period). Sometimes additional "weights" of the elements have to be taken into account (which may represent the characteristic of the element with respect to robot operation, etc.).

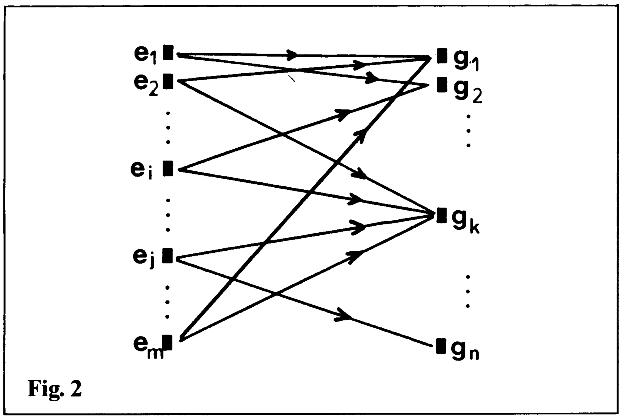


Fig. 2

Our 'real world' data consisted of about $m = 15{,}000$ elements, $n = 2{,}500$ groups, and some hundred thousand edges in the Gozinto graph (our experiences and results are summarized in Sect. 4).
The design problem roughly is to distribute the elements on 'tables' (manufacturing units which can be handled by a person, a team, or a robot) along the assembly line such that 'highly parallel' manufacturing is possible and the 'output' (in some precise sense) is maximized (obeying limited amount of investment, cost of inventory, etc.).
More abstractly we have to find a suitable family of subsets of E (the set of all elements) with each subset representing those elements of E which are 'stored' (and 'handled') on one table.
In this paper we will restrict our attention to the simplest case that every element is placed on exactly one table, thus, we have to partition E into disjoint subsets.
In the following we will briefly demonstrate how to transform this problem into a more "geometric" framework.

3. Transforming the design problem into a geometric framework

Let $E=\{e_1,\dots,e_n\}$ be a nonempty finite set (of elements), $G:=\{g_1,\dots,g_n\}$ $(g_i \subset E)$ a family of subsets of E, and $G_i:=\{g_j \in G / e_i \in g_j\}$ the set of all subsets which contain e_i, with $|G_i|$ denoting the cardinality of G_i.
With this we can state the following

Theorem 3.1. *The mapping* d: $E^2 \rightarrow R_+$ *defined by* $d(e_i,e_j) := |G_i| + |G_j| - 2\,|G_i \cap G_j|$ *generates a* quasi metric *on* E.

This means that (for all $e, e', e'' \in E$):

- $d(e,e)=0$
- $d(e,e')=d(e',e)$ (symmetry)
- $d(e,e'') \leq d(e,e')+d(e',e'')$ (triangle inequality).

However, it may happen in quasi metric spaces that there are two different elements $e, e' \in E$ with $d(e,e')=0$, since, if $e \in g_k \Leftrightarrow e' \in g_k$ for all $g_k \in G$, then $d(e,e')=0$.

Theorem 3.2. *If* h: $G \to R_+$ *is any nonnegative real valued function then the mapping* d_h: $E^2 \to R_+$ *defined by*

$$d_h(e,e') := \sum_{e \in g} h(g) + \sum_{e' \in g} h(g) - 2 \sum_{e, e' \in g} h(g)$$

generates a quasi metric *on E.*

Remarks

1. h has to be chosen adequately with respect to the specific problem and may f.e. regard to the frequency of production, characteristic of robot operation, etc.
2. Given additionally a nonnegative edge-weight w: $E \times G \to R_+$ then the analogous mapping

$$d_{w,h}(e,e') := \sum_{e \in g} w(e,g)\,h(g) + \sum_{e' \in g} w(e',g)\,h(g) - \sum_{e, e' \in g} [(w(e,g)+w(e',g))\,h(g)]$$

in general does not fullfill the triangle inequality even for $h \equiv 1$ (see Noltemeier (1985) on how to avoid this problem).

4. Using Voronoi trees to solve clustering problems in quasi metric spaces

This section will present a new data structure called Voronoi tree to support the solution of proximity problems in general quasi-metric spaces with efficiently computable distance functions.

We will analyse some structural properties and report experimental results showing that Voronoi trees are a proper and very efficient tool for the representation of proximity properties and generation of suitable clusterings.

Only very few papers on clustering in such general spaces have been published yet.

Let E be an arbitrary space, d: $E^2 \to R_+$ a mapping which induces a quasi metric on E (which is computable efficiently). We are given a finite set $S = \{e_1, \ldots, e_n\}$ of "points" of E and we first will represent this set S by a binary tree (called 'bisector tree' or 'bs-tree' for the remaining of this paper) in the following way (see Kalantari and McDonald 1983):

(i) each node of the actual tree T_i (representing the actual set $S_i := \{e_1, \ldots, e_i\}$, $0 \leq i \leq n$) contains at least one (p_L) and at most two elements (p_L and p_R) of S_i.

(ii) to insert a new element e_{i+1} into T_i (yielding T_{i+1}) start at the root node of T_i as follows:
  - if the actual node v contains only one element then insert e_{i+1} into v
  - if (otherwise) v contains two elements p_L and p_R then recursively insert e_{i+1} into the left subtree (which has root p_L) if $d(e_{i+1}, p_L) < d(e_{i+1}, p_R)$

    or into the right subtree (which has root p_R) if $d(e_{i+1}, p_R) < d(e_{i+1}, p_L)$ and arbitrarily if the distances are equal.

To support nearest neighbor queries each node v of the bs-tree T storing two elements p_L, p_R with left subtree T_L and right subtree T_R additionally stores the following two values:

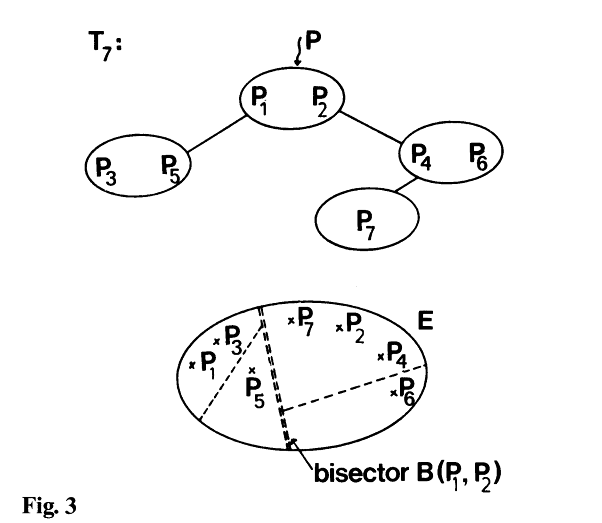


Fig. 3

LRADIUS(v): $=\max\{d(p, p_L)/p$ is stored in $T_L\}$ (left radius of v)

RRADIUS(v): $=\max\{d(p, p_R)/p$ is stored in $T_R\}$ (right radius of v).

Thus, given a query point $P \in E$ and a bs-tree representing a set $S \subset E$ in order to search for the nearest neighbor of P in S, the search process can prune the left [right] subtree $T_L[T_R]$ of an actual node v if

(*) $d(P, P_L) - \mathrm{LRADIUS}(v) \geqq \mathrm{DACTUAL}$
$[d(P, P_R) - \mathrm{RRADIUS}(v) \geqq \mathrm{DACTUAL}]$

holds with DACTUAL denoting the distance of P to its actual nearest neighbor in S (see Kalantari and McDonald 1983).
However, we have to state the following remarks:

(R1) A son may have larger radia than his father ('eccentric son').

(R2) There are cases in which a bs-tree has to be searched exhaustively even if we consider only balanced bs-trees (having logarithmic height).

(It is easy to construct examples for such cases even in E^2 and is left to the reader.)
To overcome some disadvantages of bs-trees let us introduce 'Voronoi trees'.

A *Voronoi tree (VT)* which represents S is a ternary tree with each node representing at least two and at most three points of S. To insert a new point into a *VT* we proceed in essentially the same way as described above but with the following difference:

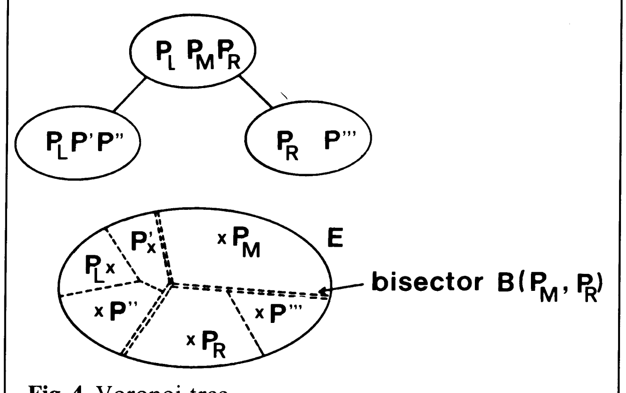


Fig. 4. Voronoi-tree

(**) If a new leaf v has to be created in order to store a new point P and Q is P's nearest neighbor of the (three) points stored in the father node of v, then Q (redundantly) has to be stored in v, too (see Fig. 4).

With this we can prove the following

Theorem 4.1

1. *Voronoi trees do not have eccentric sons.*
2. *A Voronoi tree is* transitive *in the following sense:*
   $d(P, father(P)) = \min\{d(P, P')/P' \in ancestors(P)\}$ (*with* $ancestors(P)$ *denoting the set of all elements stored in those nodes which are on the path from the root to* P).
3. *If we search for nearest neighbors of points which are stored in the root node, then this search is possible in time* $O(h)$ *with* h *denoting the height of the tree.*

Proof. Part 1 follows immediately from the definition of Voronoi trees where we have to distinguish LRADIUS(v), MRADIUS(v), and RRADIUS(v), respectively. Part 2 and 3 are a consequence of (**).

Remark. The term 'Voronoi tree' is chosen due to the fact that all triples of bisectors exactly determine the vertices of all order k Voronoi diagrams (Dehne 1983; Shamos and Hoey 1975).

Now, we will present *experimental results* to demonstrate that Voronoi trees can be efficiently used to represent proximity properties in quasi metric spaces (in general no linear spaces) and especially that *VTs* are an efficient tool for the solution of clustering problems.
Our test data were taken from two very different areas:

1. Randomly generated points in the unit cube of the d-dimensional real space R^d with L_p-norm (with $d = 2, 3, 4, 5$ and $p = 1, 2, \infty$)
2. Real production data in order to support the design of an assembly line (see Sect. 2):
   A set of 15,000 elements and three different quasi metrices which were induced by a given Gozinto graph and some other superponing effects alternatively gave the basis for our field experiments.

In each case the Voronoi tree was constructed (with a wide range of different parameters and number of points in the first environment) and the computation time, storage requirement, heights of the *VTs*, scattering of the heights of the leafs, etc. was analysed and compared with experimental results from bs-trees and ternary bs-trees (bs-trees with at most 3 points stored in each node but without condition (**), $k=3$ in Kalantari and McDonald 1983).

Additionally, the *VT* could be modified by choosing some rule how to place a copy of a father Q in his son node with respect to condition (**), i.e. fixed place, cyclic placement during the construction process, randomly chosen, or "at the same place as in the father node" (hereditary property).

We furthermore studied how the order in which the points are inserted into the *VT* influences the final result, especially some (dynamic) self-organizing principles.

The following figures illustrate a small but typical sample from our test series.

Similar to this series (with relatively small sample set) our test series in the second environment showed that the time required to construct a Voronoi tree is much smaller than the time needed to build a bs-tree or ternary bs-tree.

This result is due to the fact that Voronoi trees showed to be balanced much better than bs-trees which was expectable from Theorem 4.1.

The following table states some experiences with our large scale assembly line design problem (Sect. 2) and examplifies our general experimental results:

Gozinto-Graph (see Dehne and Noltemeier 1985c; Noltemeier 1985) containing

14,457	elements
2,259	groups
54,735	edges (density $D=0.00168$)

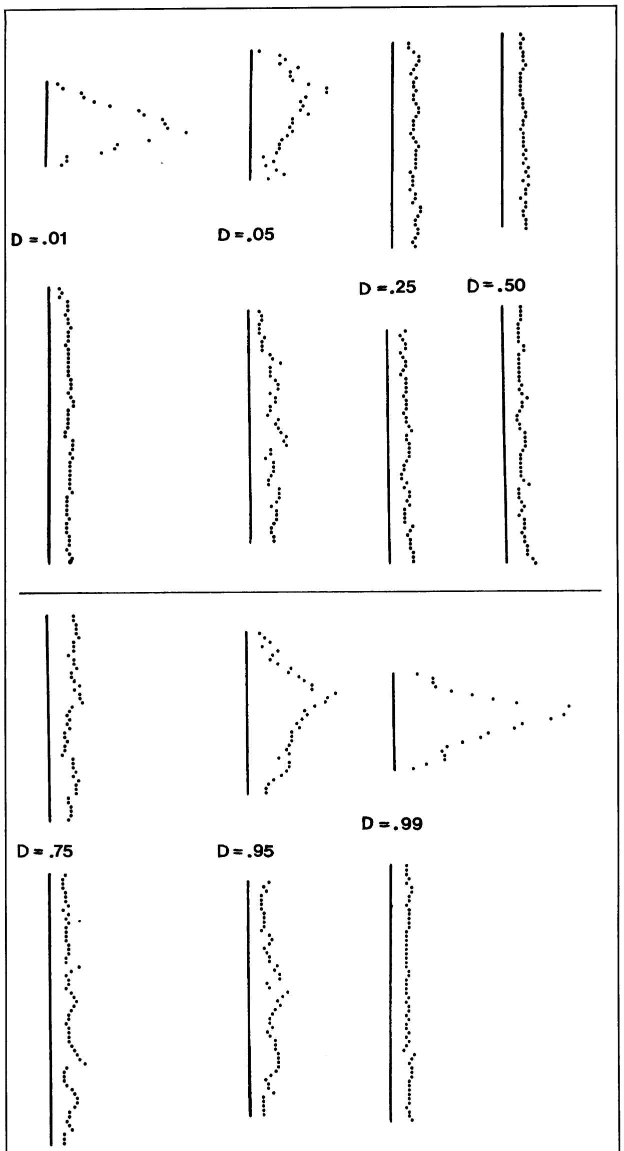


Fig. 6. Comparison of the height of bs-trees *(upper diagrams)* and Voronoi-trees *(lower diagrams)*. D: density of edges in the Gozinto-graph

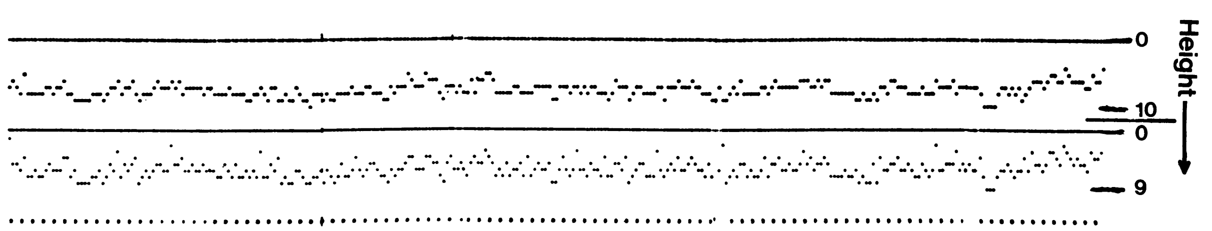


Fig. 5. Voronoi-tree (representing 1,000 points from R^2, L_1-metric): Heights of leafs *(upper part)*, heights of first occurrence of the father corresponding to the leaf *(lower part)*

	ms-tree	Voronoi tree*
Construction time in CPU sec	513	35
Number of leafs	1,512	4,273
Minimum height of leafs	5	5
Maximum height of leafs	1,211	16
Average height of leafs	427.27	10.24

* with hereditary property

It is obvious that subtrees of a Voronoi tree represent subsets of points with certain proximity properties (with resp. to the chosen quasi metric of course).
How good these proximity properties are represented by Voronoi trees is exemplified in Fig. 7. (Here by a *cluster* we mean an arbitrary subset of the set S which is represented by VT. A *clustering* is a disjoint partition and cover of S. The *diameter* of a cluster is defined to be the maximal distance of points in the cluster. There are usually some additional constraints such as upper and lower bounds on the size or number of clusters. See Dubes and Jain (1980) for more details about clustering.)
Applied to our real world problem (assembly line design) the use of Voronoi trees did not only reduce the time to get a feasible design of an assembly line dramatically, but additionally increased the value of the design by more than 30% (in comparison with standard design methods).

5. Optimized hierarchical clustering of planar point sets using generalized Voronoi diagrams

In this Section we will briefly introduce one more clustering method which is particularly important for those applications which can be transformed into planar clustering problems. This clustering method has the additional adventage that it generates an (hierarchically) optimized clustering with resp. to any given (efficiently computable) clustering measure.

5.1. Basic definitions and properties

Several clustering methodologies (e.g., our above method, see also Dubes and Jain (1980) for other methods) select cluster centers from S assigning the remaining points to their nearest cluster center (consult Dubes and Jain (1980) for more details).
We extend this to the following

Definition 5.1

(a) A *cluster* $S_i \subseteq S$ is called "*centralized*", if there exists a *center* $x \in R^2$ with S_i being the set of s_i nearest neighbors of x with respect to S. (Let $s_i := |S_i|$ for the remaining of this paper.)

(b) A *C-clustering* $(S_1, \ldots, S_C)$ of S is called *centralized*, if all S_i $(1 \leqq i \leqq C)$ are centralized.

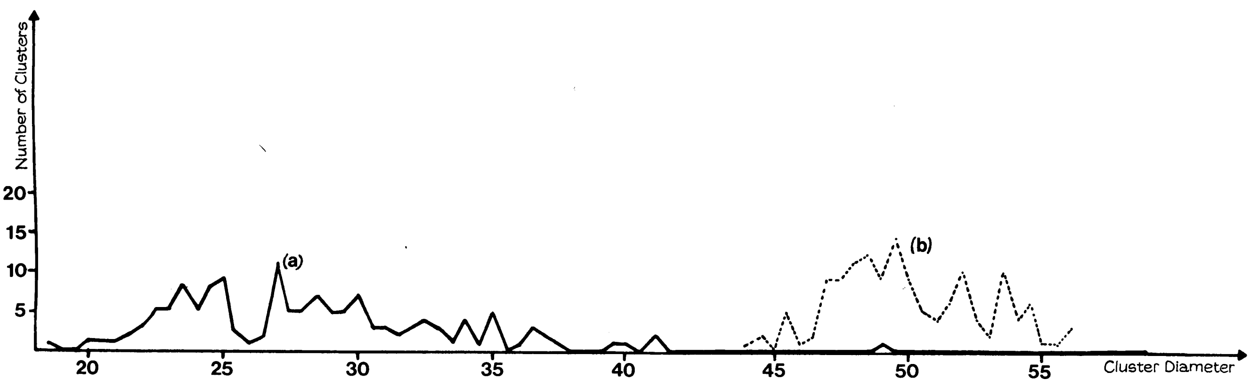


Fig. 7. Distribution of the diameters of clusters induced by Voronoi-trees (*a*) and randomly generated (*b*). Sample set: 5,000 points in E^5, cluster size between 30 and 50

(c) A *C-clustering* $(S_1, \ldots, S_C)$ of S is called "*balanced*", if for all $1 \leqq i < j \leqq C$: $|s_i - s_j| \leqq 1$ (This is the most interesting case in practice).
Let $v_k(S_i, S)$ be the order k Voronoi polygon of some $S_i \subseteq S$ $(k = s_i)$ and $V_k(S)$ be the order k Voronoi diagram of S (Shamos and Hoey 1975; Lee 1981). We shall call S_i the "*label*" of the Voronoi polygon $v_k(S_i, S)$. Using the notations of Shamos and Hoey (1975), Lee (1981) and Dehne (1983)

Lemma 5.1

5.1.1. $S_i \subseteq S$ is a centralized cluster *if and only if S_i is the label of some Voronoi polygon $v_k(S_i, S) \neq \{\ \}$.*

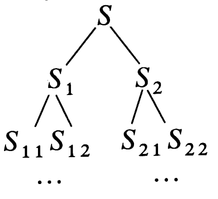
5.1.2. $(S_1, \ldots, S_C)$ is a centralized C-clustering *if and only if all S_i $(1 \leqq i \leqq C)$ are labels of some Voronoi polygon of some Voronoi diagram $V_k(S)$ and S is the disjoint union of $S_1, \ldots, S_k$*

5.1.3. $(S_1, \ldots, S_C)$ is a balanced centralized C-clustering *of S if and only if all S_i $(1 \leqq i \leqq C)$ are labels of some Voronoi polygon of $V_{\lfloor n/C \rfloor}(S)$ or $V_{\lceil n/C \rceil}(S)$ and S is the disjoint union of $S_1, \ldots, S_C$.*

Thus, a centralized C-clustering is a selection of disjoint labels of Voronoi polygons. This leads to the idea, to use the geometric properties of Voronoi diagrams for the design of clustering methodologies.

5.2. Divisive hierarchical clustering of planar point sets

Using our above definitions a (C-nested) *divisive hierarchical clustering* is a nested sequence of C-clusterings (which we will call clustering steps) successively decomposing S into smaller subsets as demonstrated in Fig. 8.
(S_1, S_2), (S_{11}, S_{12}), (S_{21}, S_{22}) is a 2-clustering of S, S_1, S_2 respectively



We shall call a divisive hierarchical clustering centralized (balanced), if all clustering steps are centralized (balanced).

This Section will demonstrate the relationships between order k Voronoi diagrams and 2-nested centralized divisive hierarchical clustering.

Definition 5.2. Two disjoint Voronoi polygons vp_1 and vp_2 are "*opposite*" to each other, if there are two nonparallel straight lines g and g' each containing two disjoint rays r_1, r_2 and r'_1, r'_2, respectively, with $r_1, r'_1 \subseteq vp_1$ and $r_2, r'_2 \subseteq vp_2$. (Note that opposite Voronoi polygons are always open.)
With this definition we prove the following lemmata:

Lemma 5.2. *Let a, b be two positive integers with $a + b \leqq n$, $a = |S_1|$, $b = |S_2|$ and $v_a(S_1, S)$, $v_b(S_2, S)$ two nonempty Voronoi polygons which are opposite, then S_1 and S_2 are disjoint.*

Lemma 5.3. *Let a, b be two positive integers with $a + b = n$, $a = |S_1|$, $b = |S_2|$ and $v_a(S_1, S)$, $v_b(S_2, S)$ two Voronoi polygons with S being the disjoint union of S_1 and S_2, then $v_a(S_1, S)$ and $v_b(S_2, S)$ are open and opposite.*

Summarizing this, we have

Theorem 5.1. *The set of all centralized 2-clusterings (S_1, S_2) of S with $|S_1| = a$ and $|S_2| = b$ is exactly the set of all pairs of labels of opposite Voronoi polygons $v_a(S_1, S)$ and $v_b(S_2, S)$ of $V_a(S)$ and $V_b(S)$ respectively.*

Because every $S_1 \subseteq S$ has exactly one complement $S_2 = S - S_1$, it follows immediately, that every open order k Voronoi polygon $v_k(S_1, S)$ has exactly one opposite order $n - k$ Voronoi polygon, thus the four bounding rays of these two polygons having pairwise exactly opposite direction.
This is an interesting property of order k Voronoi diagrams, which appears to be new.
Consider the problem of constructing an *optimal* centralized 2-clustering (S_1, S_2) of S with respect to some *clustering measure* $f(S_1, S_2) \in R$ and $|S_1| = k$, $|S_2| = n - k$. We assume a given algorithm F, which is able to compute $f(S_1, S_2)$ in time $P_F(n)$ and exchange exactly one element of S_1 and S_2, respectively, in $U_F(n)$ steps (eventually using heraditary properties). The following steps are appropriate to solve the problem:

1. Compute all open order k (and $n-k$) Voronoi polygons sorted by the angle of their bounding rays (respectively). (There are $O(n k^{1/2})$ such polygons; see Theorem 5.1, Lemma 5.4 and Edelsbrunner and Welzl 1982a).
2. Follow exactly one revolution of a rotating line pointing at the current pair of opposite Voronoi polygons and select the optimal one with respect to f, computing $O(n k^{1/2})$ updates using F.

From Edelsbrunner (private communication), Edelsbrunner and Welzl (1982a, b) we know

Lemma 5.4. *The open polygons of $V_k(S)$ can be computed in time $O(n k^{1/2} \log^2 n)$.*

Since we obtain a centralized divisive hierarchical clustering by a successive application of this procedure we get

Theorem 5.2

(a) *An optimal centralized 2-clustering can be constructed in $O(n n^{1/2} \log^2 n + U_F(n) n n^{1/2} + P_F(n))$ steps.*

(b) *An optimal centralized divisive hierarchical 2-clustering can be constructed in $O(n n^{1/2} \log^3 n + U_F(n) n n^{1/2} + P_F(n))$ steps.*

Thus, allowing cluster centers to be arbitrary points of R^2 gives us the possibility to apply the geometric structure of order k Voronoi diagrams as an interesting tool for solving clustering problems for planar point sets.

References

Day WHE, Edelsbrunner H Efficient algorithms for agglomerative hierarchical clustering methods. Report F122, Institut für Informationsverarbeitung, TU Graz, Graz, Austria

Dubes R, Jain AK (1980) Clustering methodologies in exploratory data analysis. In: Yovits MC (ed) Adv Comput. 19:113–228

Dehne F (1983) An $O(n^4)$ algorithm to construct all Voronoi diagrams for K nearest neighbor searching in the euclidean plane. Proceedings of the 10th International Colloquium on Automata, Languages and Programming (ICALP '83), Barcelona, Spain. Lecture Notes in Comput Sci, no 154

Dehne F (1986) Optical clustering. The Visual Computer 2:39–43

Dehne F, Noltemeier H (1985a) Clustering geometric objects and applications to layout problems. Proc Comput Graph. Springer, Tokyo

Dehne F, Noltemeier H (1985b) A computational geometry approach to clustering problems. Proceedings of the 1st ACM Siggraph Symposium on Computational Geometry, Baltimore, MD, USA

Dehne, F, Noltemeier H (1985c) Voronoi trees and clustering problems. Rep Inf I, Würzburg

Dehne F, Noltemeier H (1985d) Clustering geometric objects and applications to layout problems. In: Kunii TL (ed) Computer graphics – visual technology and art, Springer, Tokyo

Edelsbrunner H (private communication)

Edelsbrunner H, Kirkpatrick DG, Seidel R (1981) On the shape of a set of points in the euclidean plane. Rep F71, Institut für Informationsverarbeitung, TU Graz, Graz, Austria

Edelsbrunner H, O'Rouke J, Seidel R (1983) Constructing arrangements of lines and hyperplanes with applications. Rep F123

Edelsbrunner H, Welzl E (1982a) On the number of line-separations of a finite set in the plane. Rep F97

Edelsbrunner H, Welzl E (1982b) Halfplanar range estimation. Rep F98

Edelsbrunner H, Welzl E (1983) Halfplanar range search in linear space and $O(n^{0.695})$ query time. Rep F111

Kalantari, I, McDonald G (1983) A data structure and an algorithm for the nearest point problem. IEEE Transactions on Software Engineering, vol. SE-9, no 5

Lee DT (1981) An approach to finding the K-nearest neighbors in the euclidean plane. Rep Dept Electric Engin Comput Sci, Northwestern Univ, Evanston, Il. 60201, USA

Murtagh F (1983) Expected-time complexity results for hierarchic clustering algorithms which use cluster centers. Inf Process Lett 16:237–241

Noltemeier, H (1985) Distances in hypergraphs. Report, Inf. I, Würzburg

Overmars MH, Van Leeuwen J (1981) Maintenance of configurations in the plane. J Comput Syst Sci, vol 23:166–204

Page RL (1974) A minimum spanning tree clustering method. Commun ACM 17:321–324

Preparata FP, Hong SJ (1977) Convex hulls of finite sets of points in two and three dimensions. Comm ACM 20:87–93

Rohlf FJ (1973) Hierarchical clustering using the minimum spanning tree. Comput J 16:93–95

Schrader R (1983) Approximations of clustering and subgraph problems on trees. Discrete Appl Math 6

Shamos MI, Hoey D (1975) Closest point problems. Proc. 16th Ann. IEEE Symp Found Comp Sci

Willard DE (1982) Polygon retrieval, SIAM. J Comput 11:149–165

Yao FF (1983) A 3-space partition and its application (Extended Abstract), Proc 15th ACM Symp Theory Comp