Technische Universität Graz   Institut für Mathematik
Christian-Doppler-Laboratorium „Diskrete Optimierung“

Three-Clustering of Points in the Plane

Johann Hagauer
Günter Rote

Report 259   July 1996   CDLDO-21

Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria

Version 2a

This version has been extended by a new final section about applications to 4-clusterings, 5-clusterings, etc., which was not present in the first version of this paper.

This paper will appear in *Computational Geometry, Theory and Applications* **6** (1996). The first version appeared in *Algorithms — ESA '93, Proc. First Annual European Symposium on Algorithms*, Bad Honnef, Germany, September 1993, ed. T. Lengauer, Lecture Notes in Computer Science **726**, Springer-Verlag, 1993, pp. 192–199.

Title page processed by TEX on October 2, 1996

Three-Clustering of Points in the Plane

Johann Hagauer* and Günter Rote†

October 2, 1996

Abstract

Given n points in the plane, we partition them into three classes such that the maximum distance between two points in the same class is minimized. The algorithm takes $O(n^2 \log^2 n)$ time.
Keywords: geometric clustering; diameter; constraint propagation

1 Introduction

Overview and Statement of Results. In the classical area of cluster analysis, a given set of items is to be classified into groups (so-called clusters), such that "similar" items belong to the same group and "different" items go into different groups.

A specific case of these problems deals with *geometric* clustering problems, where the items can be represented as points in the plane (or some higher-dimensional space). In these problems the number k of clusters is fixed. Typical clustering criteria are the diameter of a cluster (the maximum distance between two points) or the radius of the smallest enclosing ball. Some previous results include an $O(n \log n)$ algorithm for finding a 2-clustering of a planar point set which minimizes the maximum diameter (Asano, Bhattacharya, Keil, and Yao 1988) and an $O(n \log^2 n / \log \log n)$ time algorithm for finding a 2-clustering which minimizes the sum of the two diameters (Hershberger 1992). The related k-center problem, where the points have to be covered by k circles whose maximum radius is to be minimized, is more difficult: the 2-center problem has only recently been solved in less than quadratic time, by an algorithm of Sharir (1996) which takes $O(n \log^9 n)$ time.

In the present paper we focus on 3-clustering. We present an $O(n^2 \log^2 n)$ algorithm for finding a 3-clustering which minimizes the maximum diameter. These results were presented at the First Annual European Symposium on Algorithms (Hagauer and Rote 1993).

*Institut für Grundlagen der Informationsverarbeitung, Technische Universität Graz, Klosterwiesgasse 32/II, A-8010 Graz, Austria.

†Institut für Mathematik, Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria; electronic mail: rote@opt.math.tu-graz.ac.at

Problem Setting. Capoyleas, Rote, and Woeginger (1991) have shown that any two clusters in an optimal k-clustering are *linearly separable*, for a wide variety of clustering criteria, including the maximum diameter criterion. Since there are only $O(n^2)$ ways to partition an n-point set by a line, this immediately implies polynomial k-clustering algorithms for any fixed value of k. Three clusters can be pairwise separated from each other by three lines. The straightforward application of this fact leads to an $O(n^7 \log n)$ time algorithm which checks $O(n^6)$ possibilities. A less straightforward approach considers all $O(n^4)$ ways in which a cluster can be separated from the rest by two rays meeting at a common point, and solving a two-clustering problem for the remaining points by the algorithm of Asano et al. (1988). This would still lead to a complexity of $O(n^5 \log n)$ for finding a 3-clustering which minimizes the maximum diameter. We will use a more direct approach.

Overview of the algorithm. The maximum diameter can only be one of the $\binom{n}{2}$ distances between the n given points. By a binary search among these values we can therefore reduce the optimization problem to the decision problem of testing the existence of a 3-clustering with a given upper bound δ on the maximum diameter. The cluster which contains the leftmost point P is denoted by A. We test every possible choice for the rightmost point of A separately. This requires some insight into the structure of optimal clusterings and is described in the sequel.

2 Preliminaries

The given set of n points in the plane will be denoted by P. We denote the coordinates of a point u by $u = (u_x, u_y)$. For ease of exposition we assume that no two given points have the same x- or y-coordinate. (This can be achieved by an appropriate rotation.) The line segment joining two points u and v is denoted by $\overline{uv}$, and $d(u,v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2}$ denotes their Euclidean distance. Two sets of points in the plane are said to be *linearly separable* if they can be strictly separated by a straight line. For a point set S, $\operatorname{diam}(S)$ denotes its diameter, i. e., the maximum distance between two points in S. Our algorithm relies on the the following result of Capoyleas, Rote, and Woeginger (1991).

Proposition 1 *Consider the k-clustering problem of minimizing the maximum diameter. For every point set in the plane, there exists an optimal k-clustering such that each pair of clusters is linearly separable.* □

Our approach will be to specify a parameter δ and test for the existence of a 3-partitioning such that each cluster has diameter at most δ. We need another easy lemma:

Proposition 2 (Capoyleas, Rote, and Woeginger 1991, proposition 1.) *Let δ be a positive real number, let u and v be two points in the plane at distance less than or equal to δ. Let C_1 and C_2 be the circles with radius δ centered at u and v, let D denote the points in the vertical strip between u and v. Then the part of the region $C_1 \cap C_2 \cap D$ that lies above the line segment $\overline{uv}$ has diameter $\leq \delta$.* □

3 Geometric Properties of a Solution

The algorithm to be described searches for all possible linearly separable solutions A, B, C in the following way. The point a with minimum x-coordinate is placed in A. Each point $a' \in P$ satisfying $d(a, a') \leq \delta$ is tested as a candidate for being the rightmost point in A. That is, we will only allow a point u to be assigned to A if its x-coordinate $u_x < a'_x$. For every point a' we will test in $O(n \log n)$ time whether there is a solution or not. This yields an overall time complexity of $O(n^2 \log n)$ for the decision problem and of $O(n^2 \log^2 n)$ for the optimization problem.

The pair (a, a') gives rise to the following partition of the point set $P - \{a, a'\}$, see figure 1.

$$\mathrm{NORTH} := \{\, u \in P \mid u_x < a'_x \text{ and } u \text{ is above the segment } \overline{aa'}\,\}.$$
$$\mathrm{SOUTH} := \{\, u \in P \mid u_x < a'_x \text{ and } u \text{ is below the segment } \overline{aa'}\,\}.$$
$$\mathrm{EAST} := \{\, u \in P \mid u_x > a'_x \,\}.$$
$$A_{\text{cand}} := \{\, u \in P \mid d(a, u) \leq \delta,\ d(a', u) \leq \delta \,\}.$$

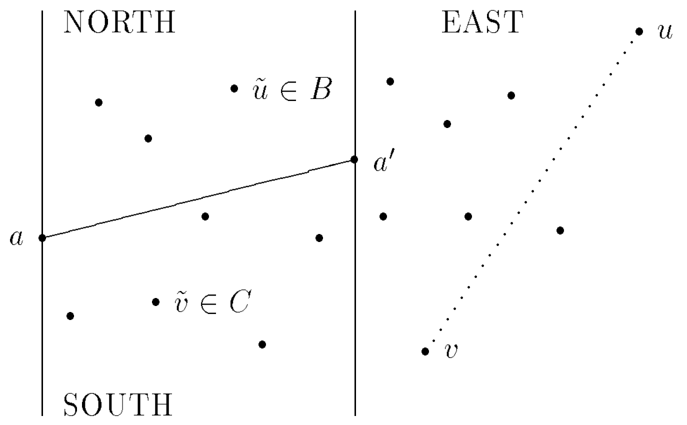

Figure 1: The partition into NORTH, SOUTH, and EAST.

The following lemma is an immediate consequence of proposition 2.

Lemma 3

$$\mathrm{diam}(A_{\text{cand}} \cap \mathrm{NORTH}) \leq \delta$$
$$\mathrm{diam}(A_{\text{cand}} \cap \mathrm{SOUTH}) \leq \delta \qquad \square$$

We call a partition A, B, C of P an (a, a')-*solution* if $a \in A$ is the leftmost point in P, a' the rightmost point in A, $\mathrm{diam}(A) \leq \delta$, $\mathrm{diam}(B) \leq \delta$, and $\mathrm{diam}(C) \leq \delta$, and A, B, and C are linearly separable. In order to find an (a, a')-solution A, B, C we consider three different cases.

Case 1: NORTH $\subseteq A$.
Case 2: SOUTH $\subseteq A$.
Case 3: NORTH $\not\subseteq A$ and SOUTH $\not\subseteq A$.

Case 1 can be treated in the following way. We define $A := \{a, a'\} \cup \text{NORTH} \cup M$, where

$$M := \{\, u \in \text{SOUTH} \cap A_{\text{cand}} \mid d(u,v) \leq \delta \text{ for all } v \in \text{NORTH} \,\} \ .$$

By lemma 3, $\text{diam}(\text{SOUTH} \cap A_{\text{cand}}) \leq \delta$ which implies $\text{diam}(M) \leq \delta$. Therefore placing any point $u \in \text{SOUTH} \cap A_{\text{cand}}$ in A does not impose any restriction for other points in $\text{SOUTH} \cap A_{\text{cand}}$. Each of these points can be assigned to A independently of the others. Therefore the maximal feasible set A is uniquely defined. We are left with a two-clustering problem for the set of points $P - A$, which can be solved in $O(n \log n)$ time by the algorithm of Asano et al. (1988). Case 2 is treated analogously.

3.1 Case 3: The Initial Classification of Points

Case 3 turns out to be the most difficult case. Since we are looking for separable solutions, it is clear that neither B nor C may contain a point both from NORTH and from SOUTH. Therefore we assume w. l. o. g. $B \cap \text{SOUTH} = \emptyset$ and $C \cap \text{NORTH} = \emptyset$.

Consider now the set EAST. All points from EAST will be placed either in B or C. The following lemma allows us to make this decision for point pairs which have large distance.

Lemma 4 *Let A, B, C be an (a, a')-solution with $B \cap \text{NORTH} \neq \emptyset$ and $C \cap \text{SOUTH} \neq \emptyset$. Then for each pair of points $u, v \in \text{EAST}$ with distance $d(u,v) > \delta$ the following holds: If $u_y > v_y$ then $u \in B$ and $v \in C$.*

Proof. By assumption there are points $\tilde{u} \in \text{NORTH} \cap B$ and $\tilde{v} \in \text{SOUTH} \cap C$ (see figure 1). Assume w. l. o. g. $u_x > v_x$. If $\tilde{v}_y < v_y$ then $d(\tilde{v}, u) > d(v, u) > \delta$. Therefore u must be in B. Otherwise, $\tilde{v}_y \geq v_y$, and the segment $\overline{\tilde{u}v}$ crosses either the segment $\overline{\tilde{v}u}$ or the segment $\overline{aa'}$. Either possibility would contradict $u \in C$ and $v \in B$ together with the separability assumption. $\square$

The above lemma and the previous discussion justifies the following initial classification of points:

$A_0 := \{a, a'\}$.
$B_0 := \{\, u \in \text{NORTH} \mid d(u,a) > \delta \text{ or } d(u,a') > \delta \,\} \cup$
$\qquad \{\, u \in \text{EAST} \mid \text{there exists a point } v \in \text{EAST}, d(u,v) > \delta \text{ and } u_y > v_y \,\}$.
$C_0 := \{\, u \in \text{SOUTH} \mid d(u,a) > \delta \text{ or } d(u,a') > \delta \,\} \cup$
$\qquad \{\, u \in \text{EAST} \mid \text{there exists a point } v \in \text{EAST}, d(u,v) > \delta \text{ and } u_y < v_y \,\}$.
$AB_{\text{cand}} := \text{NORTH} - B_0$.
$CA_{\text{cand}} := \text{SOUTH} - C_0$.
$BC_{\text{cand}} := \text{EAST} - B_0 - C_0$.

A_0, B_0, and C_0 will be called the *initial sets*, the other three sets the *candidate sets*. We will use the generic terms X_0 and XY_{cand} to refer to any of the respective initial or candidate sets. Note that if B_0 and C_0 are not disjoint, then case 3 cannot lead to a solution, and we may stop immediately.

Lemma 5 *The diameter of all candidate sets is at most δ.*

Proof. The inequalities $\mathrm{diam}(AB_{\mathrm{cand}}) \leq \delta$ and $\mathrm{diam}(CA_{\mathrm{cand}}) \leq \delta$ follow from lemma 3. The relation $\mathrm{diam}(BC_{\mathrm{cand}}) \leq \delta$ follows from the definitions of B_0, C_0, and BC_{cand}. □

3.2 Propagation of Constraints

Every (a, a')-solution A, B, C must satisfy $C \cap AB_{\mathrm{cand}} = \emptyset$, $B \cap CA_{\mathrm{cand}} = \emptyset$, and $A \cap BC_{\mathrm{cand}} = \emptyset$. So far, the placement of points of the initial sets is fixed, whereas the placement of points in candidate sets is not yet fixed. Some point in an initial set may force certain points in a candidate set to be assigned to A, B, or C. Such constraints imposed by the initial assignment may propagate. We describe this situation by two directed graphs $G_1 := (V, E_1)$ and $G_2 := (V, E_2)$ with vertex set $V = P$. There is an arc (u, v) in E_1 if $d(u, v) > \delta$ and one of the following five conditions holds:

1. $u \in B_0$, $v \in AB_{\mathrm{cand}}$
2. $u \in C_0$, $v \in BC_{\mathrm{cand}}$
3. $u \in AB_{\mathrm{cand}}$, $v \in CA_{\mathrm{cand}}$
4. $u \in CA_{\mathrm{cand}}$, $v \in BC_{\mathrm{cand}}$
5. $u \in BC_{\mathrm{cand}}$, $v \in AB_{\mathrm{cand}}$

Similarly, there is an arc (u, v) in E_2 if $d(u, v) > \delta$ and one of the following conditions holds:

1. $u \in B_0$, $v \in BC_{\mathrm{cand}}$
2. $u \in C_0$, $v \in CA_{\mathrm{cand}}$
3. $u \in AB_{\mathrm{cand}}$, $v \in BC_{\mathrm{cand}}$
4. $u \in BC_{\mathrm{cand}}$, $v \in CA_{\mathrm{cand}}$
5. $u \in CA_{\mathrm{cand}}$, $v \in AB_{\mathrm{cand}}$

Figure 2 gives a schematic representation of these graphs. The two graphs correspond to "counter-clockwise" propagation ($A \to C \to B \to A$) and to "clockwise" propagation of constraints ($A \to B \to C \to A$), respectively. (The constraints imposed by the initial set $A_0 = \{a, a'\}$ have already explicitly been taken care of in the definition of B_0 and C_0.) Note that every pair of points $u, v \in AB_{\mathrm{cand}} \cup BC_{\mathrm{cand}} \cup CA_{\mathrm{cand}}$ with $d(u, v) > \delta$ gives rise to two arcs, one in each graph.

Let Forced_1 denote the set of points in $AB_{\mathrm{cand}} \cup CA_{\mathrm{cand}} \cup BC_{\mathrm{cand}}$ which are reachable by some path in G_1 starting at a vertex in $B_0 \cup C_0$, and Forced_2 the analogous set for G_2.

Lemma 6 *There is an (a, a')-solution A, B, C with $B \cap \mathrm{NORTH} \neq \emptyset$ and $C \cap \mathrm{SOUTH} \neq \emptyset$ (corresponding to case 3) if and only if*

(i) $\mathrm{diam}(B_0) \leq \delta$,

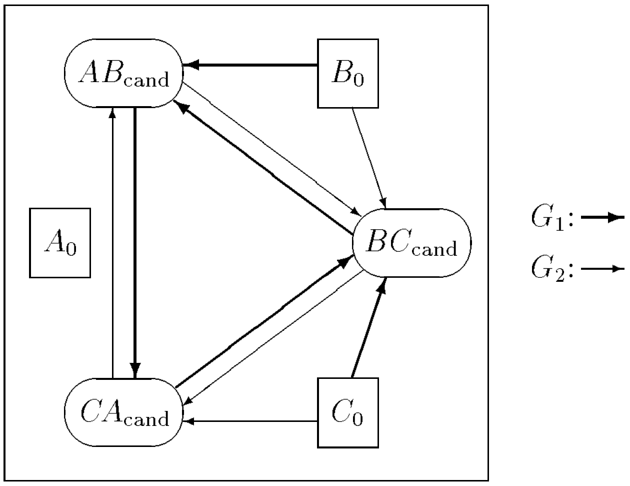

Figure 2: Schematic representation of the constraint propagation graphs G_1 and G_2. G_1 is shown with thick arcs and G_2 with thin arcs.

(ii) $\mathrm{diam}(C_0) \leq \delta$, *and*
(iii) *the sets* Forced_1 *and* Forced_2 *are disjoint.*

Proof. Assume first that the conditions hold. We construct a solution by the following rules. Initially $A = A_0$, $B = B_0$, and $C = C_0$.

Rule 1: If $u \in \mathit{Forced}_1$ then u is assigned to A if $u \in AB_{\text{cand}}$,
u is assigned to C if $u \in CA_{\text{cand}}$,
u is assigned to B if $u \in BC_{\text{cand}}$.

Rule 2: If $u \in \mathit{Forced}_2$ then u is assigned to B if $u \in AB_{\text{cand}}$,
u is assigned to A if $u \in CA_{\text{cand}}$,
u is assigned to C if $u \in BC_{\text{cand}}$.

Rule 3: If $u \in (AB_{\text{cand}} \cup CA_{\text{cand}} \cup BC_{\text{cand}}) - (\mathit{Forced}_1 \cup \mathit{Forced}_2)$ then classify u as in rule 1.

Rules 1 and 2 express the necessary assignment for the sets A, B, or C that follow from propagating assignments through the graphs, as can be seen by induction on the path length leading to a point $u \in \mathit{Forced}_1 \cup \mathit{Forced}_2$. Rule 3 is an arbitrary decision. The points that fall under this rule could either be handled (consistently) as in rule 1 or as in rule 2. A concise statement of the result of applying the three rules is as follows:

$$A = A_0 \cup (CA_{\text{cand}} \cap \mathit{Forced}_2) \cup (AB_{\text{cand}} - \mathit{Forced}_2)$$
$$B = B_0 \cup (AB_{\text{cand}} \cap \mathit{Forced}_2) \cup (BC_{\text{cand}} - \mathit{Forced}_2)$$
$$C = C_0 \cup (BC_{\text{cand}} \cap \mathit{Forced}_2) \cup (CA_{\text{cand}} - \mathit{Forced}_2)$$

As every point is placed by exactly one rule, the sets A, B and C form a partition of P. To show that we have a valid solution we have to prove that the diameters do not exceed δ. Consider two points u, v with $d(u,v) > \delta$. If both are in initial sets, they cannot be in the same initial set, by conditions (i) and (ii). Therefore they end up in different sets A, B, or C. If at least one of the two points is in a candidate set, lemma 5 implies that they are not in the same candidate set. Therefore one of the arcs (u, v) and (v, u) is either in E_1 or in E_2. Therefore u and v are either both in $Forced_1$ or in $Forced_2$ or in neither set. This means that they are assigned to different sets A, B, or C by rule 1, by rule 2, or by rule 3, respectively.

On the other hand, failure of any of the conditions of the lemma would lead to a contradiction: Conditions (i) and (ii) are necessary by lemma 4. If (iii) does not hold, rules 1 and 2 lead to incompatible conclusions about the assignment of a point $u \in Forced_1 \cup Forced_2$. By the above discussion this makes a solution impossible. □

4 The Tripartition Algorithm

The algorithm follows the outline in the previous section. We have to test all points $a' \in P$ as possible rightmost points of A. For each a' we determine the sets NORTH, SOUTH, and EAST in linear time, and we test cases 1, 2 and 3 of section 3 (following lemma 3). Let us first look at case 3: After the determination of initial and candidate sets we call the procedures *rule1* and *rule2* below, which in turn rely on the procedure *distribute*. These procedures explore the graphs G_1 and G_2 as they construct them, assigning the points to the sets A, B, or C and propagating constraints as soon as possible.

procedure *rule1*
    $C_{\text{forced}} := C_0$;
    $distribute(C, B)$;
    $B_{\text{forced}} := B_{\text{forced}} \cup B_0$;
    while $B_{\text{forced}} \neq \emptyset$ **do**
        $distribute(B, A)$;
        $distribute(A, C)$;
        $distribute(C, B)$;
end procedure

procedure *rule2*
    $B_{\text{forced}} := B_0$;
    $distribute(B, C)$;
    $C_{\text{forced}} := C_{\text{forced}} \cup C_0$;
    while $C_{\text{forced}} \neq \emptyset$ **do**
        $distribute(C, A)$;
        $distribute(A, B)$;
        $distribute(B, C)$;
end procedure

procedure $distribute(X, Y)$
    $Y_{\text{forced}} := \emptyset$;
    for all points $u \in X_{\text{forced}}$ **do**
        $X := X \cup \{u\}$;
        $New := \{\, v \in XY_{\text{cand}} \mid d(u, v) > \delta \,\}$;
        $Y_{\text{forced}} := Y_{\text{forced}} \cup New$;
        $XY_{\text{cand}} := XY_{\text{cand}} - New$;
end procedure

(A literal interpretation of the procedure *distribute* refers to sets such as BA_{cand}. This is the same as the set AB_{cand}, and in general XY_{cand} and YX_{cand} are identical.)

Rule 3 just assigns the remaining points as follows: $A := A \cup AB_{\text{cand}}$, $B := B \cup BC_{\text{cand}}$, and $C := C \cup CA_{\text{cand}}$. We finally compute the diameters of A, B, and C in $O(n \log n)$ time to check that we really have a solution.

In order to implement the procedure *distribute* efficiently we use the circular hull introduced by Hershberger and Suri (1991). Let δ be fixed. The *circular hull* of a set of points U is the common intersection of all disks of radius δ containing U. (Circular hulls are also known as α-hulls, see Edelsbrunner, Kirkpatrick, and Seidel 1983.) Circular hulls can be constructed in $O(n \log n)$ time. The data structure of Hershberger and Suri (1991) supports the following two main operations:

1. Given a query point u, determine in time $O(\log n)$ a point $v \in U$ such that $d(u, v) > \delta$, if such a point exists.
2. Delete a point from U and update the circular hull in an amortized cost of $O(\log n)$.

The procedures *rule1*, *rule2*, and *distribute* are implemented in the following way: First the circular hulls for the candidate sets XY_{cand} are constructed. For a point $u \in X_{\text{forced}}$ in the procedure *distribute* we can use the circular hull of XY_{cand} to check whether a point $v \in XY_{\text{cand}}$ exists with $d(u, v) > \delta$. Such a point v is deleted from the circular hull of the point set XY_{cand} and inserted in Y_{forced}. Then u is repeatedly used as a query to the circular hull until no more points v are found. The point u is finally removed from XY_{cand} and assigned to X, and it will never again act as query to some circular hull. Therefore the number of queries is bounded by the number of deletions plus the number of points in the initial sets. Since no point is inserted in a circular hull, the overall complexity for deletions and queries is $O(n \log n)$. Thus the time bound for the procedures *rule1* and *rule2* is $O(n \log n)$.

Circular hulls can also be used to carry out the remaining operations in $O(n \log n)$ time: the computation of B_0 and C_0 and the treatment of cases 1 and 2. (These tasks can also be solved in $O(n \log n)$ time with furthest-point Voronoi diagrams, a more standard data structure.)

Therefore our tripartition algorithm takes $O(n^2 \log n)$ time, for fixed δ.

Theorem 7 *Given a set of n points in the plane and a real number δ, we can determine in $O(n^2 \log n)$ time whether there is a partition of P into sets A, B, C with diameters at most δ.* □

Theorem 8 *Given a set of n points in the plane, we can in $O(n^2 \log^2 n)$ time construct a partition of P into sets A, B, C such that the largest of the three diameters is as small as possible.*

Proof. We carry out a binary search on the $\binom{n}{2}$ distances occurring in a point set of cardinality n, using the *tripartition* algorithm. □

5 Applications to More Than Three Clusters

By the separability result of Capoyleas, Rote, and Woeginger (1991), the k-clustering problem can be solved in polynomial time for every fixed value of k. We will now show that our 3-clustering algorithm can be used as a subroutine to speed up k-clustering algorithms for $k \geq 4$.

In the case of 4-clusterings, there is always an optimal clustering in which two of the clusters can be separated from the remaining two by a polygonal chain with at most four straight pieces (see Edelsbrunner, Robison, and Shen 1990, lemmas 1 and 2, or Capoyleas et al. 1991, lemma 8). Accordingly, Capoyleas et al. proposed to divide the problem into two 2-clustering problems, trying all $O(n^8)$ ways to separate the point set into two parts using a polygonal chain with four straight pieces. Each of the 2-clustering problems can be solved in $O(n \log n)$ time by the method of Asano et al. (1988), yielding an overall complexity of $O(n^9 \log n)$. Using our 3-clustering algorithm, this can be speeded up to $O(n^8 \log^2 n)$: we know that each cluster can be separated from the remaining three by a polygonal chain with at most *three* straight pieces. This gives $O(n^6)$ choices, and for each choice, an associated 3-clustering problem must be solved.

For 5-clusterings, two clusters can be separated from the remaining three by 5 straight pieces, and this can be done in $O(n^{10})$ possible ways, which leads to a complexity of $O(n^{12} \log^2 n)$. For 6-clusterings, three clusters can be separated from the remaining three by 6 straight pieces, yielding a total complexity of $O(n^{14} \log^2 n)$. Analogously, problems with a larger number of clusters can be treated, but already for 4-clusterings, there is ample space for improvement.

References

Te. Asano, B. Bhattacharya, M. Keil, and F. Yao: Clustering algorithms based on minimum and maximum spanning trees. In: *Proc. 4th Ann. Symp. Comput. Geometry* (1988), pp. 252–257.

V. Capoyleas, G. Rote, and G. Woeginger: Geometric clusterings. *J. Algorithms* **12** (1991), 341–356.

H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel: On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory* **IT-29** (1983), 551–559.

H. Edelsbrunner, A. D. Robison, and X. Shen: Covering convex sets with non-overlapping polygons. *Discrete Math.* **81** (1990), 153–164.

J. Hagauer and G. Rote: Three-clustering of points in the plane. In: *Algorithms — ESA '93, Proc. First Annual European Symposium on Algorithms*, Bad Honnef, Germany, September 1993, ed. T. Lengauer, Lecture Notes in Computer Science **726**, Springer-Verlag, 1993, pp. 192–199.

J. Hershberger: Minimizing the sum of diameters efficiently. *Comput. Geom. Theory Appl.* **2** (1992), 111–118.

J. Hershberger and S. Suri: Finding Tailored Partitions. *J. Algorithms* **12** (1991), 431–463.

M. Sharir: A near-linear algorithm for the planar 2-center problem. In: *Proc. 12th Ann. Sympos. Comput. Geometry* (1996), pp. 106–112; to appear in *Discrete Comput. Geom.*