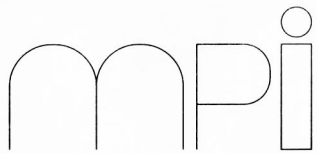
MAX-PLANCK-INSTITUT FÜR INFORMATIK

An $O(n \log n)$ algorithm for finding a k-point subset with minimal L_∞-diameter

Michiel Smid

MPI-I-93-116 April 1993



Im Stadtwald
66123 Saarbrücken
Germany

An $O(n \log n)$ algorithm for finding a k-point subset with minimal L_∞-diameter

Michiel Smid

MPI-I-93-116 April 1993

An $O(n \log n)$ algorithm for finding a k-point subset with minimal L_∞-diameter*

Michiel Smid
Max-Planck-Institut für Informatik
W-6600 Saarbrücken, Germany
michiel@mpi-sb.mpg.de

April 5, 1993

Abstract

Let S be a set of n points in d-space, let R be an axes-parallel hyper-rectangle and let $1 \leq k \leq n$ be an integer. An algorithm is given that decides if R can be translated such that it contains at least k points of S. After a presorting step, this algorithm runs in $O(n)$ time, with a constant factor that is doubly-exponential in d. Two applications are given. First, a translate of R containing the maximal number of points can be computed in $O(n \log n)$ time. Second, a k-point subset of S with minimal L_∞-diameter can be computed, also in $O(n \log n)$ time. Using known dynamization techniques, the latter result gives improved dynamic data structures for maintaining such a k-point subset.

1 Introduction

In statistical clustering and pattern recognition, the following problem often arises: Given a set S of n points in d-dimensional space and an integer k, find a subset of S of size k that minimizes some proximity measure. Measures that have been considered are the perimeter of the convex hull of the k points, its diameter and the perimeter of its smallest enclosing rectangle or square. See Dobkin et al. [4], Aggarwal et al. [1], Smid [9], Eppstein and Erickson [5], and Datta et al. [3].

In this paper, we take the L_∞-diameter as the proximity measure. That is, we want to find k points that have minimal L_∞-diameter among all k-point subsets. Note that this is equivalent to finding a smallest d-dimensional axes-parallel cube that contains at least k points.

We summarize the results that were known for this problem. In [1], the first efficient algorithm for the planar version occurs. It uses k-th order Voronoi diagrams in the L_∞-metric. The running time of this algorithm is bounded by $O(k^2 n \log n)$ and it uses $O(kn)$

*This work was supported by the ESPRIT Basic Research Actions Program, under contract No. 7141 (project ALCOM II).

space. In [9], a simple sweep algorithm is given that solves the problem in $O(n \log n + kn \log^2 k)$ time using $O(n)$ space.

This running time was subsequently improved in [5]. In fact, their algorithm is the first efficient one that solves the d-dimensional version of the problem for fixed $d \geq 2$. Starting by computing for each point its $O(k)$ nearest neighbors leads to an $O(nk \log n + nk^{d/2-1} \log^2 k)$ time algorithm that uses $O(nk + k^{d/2})$ space. Using techniques that were designed for closest pair problems, this result was improved in [3]: There, an algorithm is given with $O(n \log n + nk^{d/2-1} \log^2 k)$ running time using $O(n + k^{d/2})$ space. Note that for the planar case, the time resp. space bounds of the latter algorithm are $O(n \log n + n \log^2 k)$ resp. $O(n)$. This seems to be the status of the problem.

In this paper, we improve these results drastically: We give an algorithm that, given any set S of n points in $\mathbb{R}^d$ and any integer $1 \leq k \leq n$, computes a k-point subset with minimal L_∞-diameter, in $O(n \log n)$ time using $O(n)$ space. That is, the complexity of the problem does not depend on k. The constant factor, however, is $2^{O(d^2 2^{2d})}$. This makes the algorithm unsuited for practical applications.

Our algorithm is based on an efficient solution to the following problem: Given a d-dimensional axes-parallel rectangle, decide if it can be translated such that it contains at least k points of S. The solution in [5] also depends on a solution to this problem. There, the problem is solved using a variant of the algorithm of Overmars and Yap [8] that computes the volume of the union of a set of axes-parallel rectangles. This variant takes $O(n^{d/2} \log n)$ time.

We show that the problem of deciding if the rectangle can be translated such that it contains at least k points can be solved in $O(n \log n)$ time. In fact, after a presorting step, the algorithm takes only $O(n)$ time. In this algorithm, the doubly-exponential dependence on d occurs.

Note that this result immediately gives an $O(n \log n)$ time algorithm for translating the rectangle such that it contains the maximal number of points. This result seems to be new. The related problem for a planar circle can be solved in $O(n^2)$ time, see Chazelle and Lee [2].

This paper is organized as follows. In Section 2, we recall degraded grids. Degraded grids have basically the same properties as standard grids. They can be constructed, however, without using the floor function. In this way, our algorithms fall inside the algebraic decision tree model.

In Sections 3 and 4, we consider the problem of deciding if a hyper-rectangle R can be translated such that it contains at least k points. First, in Section 3, we consider the restricted case where all points of S lie in a rectangle that has sides of lengths twice the side-lengths of R. We give a recursive algorithm for this problem with running time $O(n)$. Then, in Section 4, degraded grids are used to reduce the general problem of translating R to the restricted version of Section 3.

In Section 5, we solve the problem of finding a k-point subset with minimal L_∞-diameter. As mentioned already, this is equivalent to finding a smallest d-dimensional axes-parallel cube that contains at least k points of S. The algorithm makes a binary search in the set of all $d\binom{n}{2}$ differences $|p_i - q_i|$, where $p, q \in S$ and $1 \leq i \leq d$. To test one such difference, the algorithm of Section 4 is used. As in [7], the set of candidate differences is maintained implicitly. In this way, the algorithm uses only $O(n)$ space.

In Section 6, known dynamization results of [3] are used to obtain improved data

structures that maintain a k-point subset with minimal L_∞-diameter if points are inserted and deleted. In Section 7, we give some concluding remarks.

2 Preliminaries: Degraded grids

Later in this paper, we want an algorithm that decides if a d-dimensional axes-parallel rectangle R can be translated such that it contains at least k points of a given set of n points. A possible solution is to construct a grid having a size that is equal to the size of R, and solve the problem for subsets that are contained in neighboring cells of this grid. Using this approach, however, we need the floor-function to find the grid cell that contains a given point. Hence, the algorithm falls outside the algebraic decision tree model.

In this section, we recall degraded grids. These have basically the same properties as standard grids. We can build and search in a degraded grid, however, without using the floor-function. Degraded grids were introduced in [7]. We use a somewhat simpler notion that was introduced in [3].

A d-dimensional axes-parallel rectangle of the form

$$[a_1 : b_1] \times [a_2 : b_2] \times \ldots \times [a_d : b_d],$$

where a_i and b_i, $1 \leq i \leq d$, are real numbers is called a *box*. If $b_i = a_i + \delta$ for all i, then the box is called a *δ-box*.

Definition 1 *Let S be a set of n real numbers and let δ be a positive real number. Let $a_1, a_2, \ldots, a_l$ be a sequence of real numbers such that*

1. *for all $1 \leq j < l$, $a_{j+1} \geq a_j + \delta$,*
2. *for all $p \in S$, $a_1 \leq p < a_l$,*
3. *for all $1 \leq j < l$, if there is a point $p \in S$ such that $a_j \leq p < a_{j+1}$, then $a_{j+1} = a_j + \delta$.*

The collection of intervals $[a_j : a_{j+1}]$, $1 \leq j < l$, is called a one-dimensional degraded δ-grid for S. *If p is a real number, then p is said to be* contained *in the interval $[a_j : a_{j+1}]$ if $a_j \leq p < a_{j+1}$.*

Constructing a one-dimensional degraded δ-grid: Sort the elements of S. Let $p^{(1)} \leq p^{(2)} \leq \ldots \leq p^{(n)}$ be the sorted sequence. Let $a_1 := p^{(1)}$. Let $j \geq 1$, and assume that $a_1, \ldots, a_j$ are defined already.

If there is an element of S that lies in the half-open interval $[a_j : a_j + \delta)$, then we set $a_{j+1} := a_j + \delta$. Otherwise, we set a_{j+1} to the value of the smallest element in S that is larger than a_j. The construction stops if we have visited all elements of S.

Definition 2 *Let S be a set of n points in d-space and let δ be a positive real number. For $1 \leq i \leq d$, let S_i be the set of i-th coordinates of the points in S. Let*

$$[a_{ij} : a_{i,j+1}], \quad 1 \leq j < l_i,$$

be a one-dimensional degraded δ-grid for the set S_i. The collection of d-dimensional boxes

$$\prod_{i=1}^{d}[a_{ij_i} : a_{i,j_i+1}], \text{ where } 1 \leq j_i < l_i,$$

is called a d-dimensional degraded δ-grid for S. *A point $p = (p_1, p_2, \ldots, p_d) \in \mathbb{R}^d$ is said to be* contained *in the box $\prod_{i=1}^{d}[a_{ij_i} : a_{i,j_i+1}]$ if $a_{ij_i} \leq p_i < a_{i,j_i+1}$ for all $1 \leq i \leq d$.*

See Figure 1 for an example. The following lemma follows immediately.

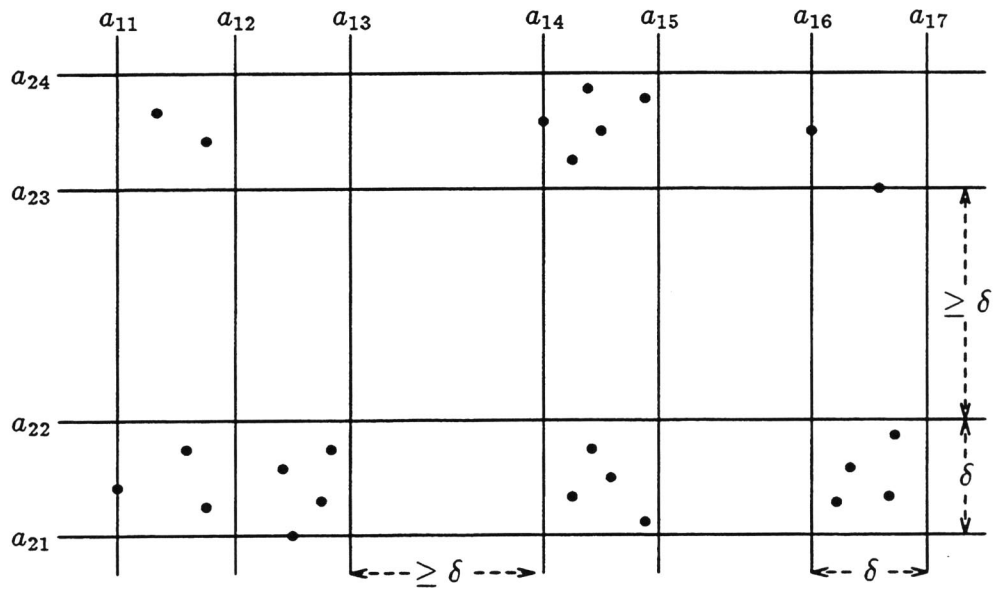


Figure 1: Example of a degraded δ-grid.

Lemma 1 *Let $p \in \mathbb{R}^d$ and let B be the box in the degraded δ-grid for S that contains p. All points of S that are within L_∞-distance δ from p are contained in B and in the $3^d - 1$ boxes that surround B.*

Constructing a d-dimensional degraded δ-grid: Assume the points of S are stored in an array $\mathcal{S}$. For each $1 \leq i \leq d$, sort the elements of S_i. Give each element in S_i a pointer to its occurrence in $\mathcal{S}$.

For each $1 \leq i \leq d$, construct a one-dimensional degraded δ-grid $[a_{ij} : a_{i,j+1}]$, $1 \leq j < l_i$, for the set S_i using the algorithm given above. During this construction, for each j and each element p_i—which denotes the i-th coordinate of point p—such that $a_{ij} \leq p_i < a_{i,j+1}$, follow the pointer to $\mathcal{S}$. Store the numbers a_{ij} and j with the occurrence of p in $\mathcal{S}$.

At the end, each point in $\mathcal{S}$ stores with it two vectors of length d. If point p has vectors $(b_1, b_2, \ldots, b_d)$ and $(j_1, j_2, \ldots, j_d)$, then p is contained in the δ-box with "lower-left" corner $(b_1, b_2, \ldots, b_d)$. This δ-box is part of the j_i-th δ-slab along the i-th axis.

These vectors implicitly define the degraded δ-grid. Note that each j_i is an integer in the range from 1 to n. Hence, we can sort the vectors $(j_1, j_2, \ldots, j_d)$ in $O(n)$ time

by using radix-sort. This gives the non-empty boxes of the degraded grid, sorted in lexicographical order.

Consider a non-empty box B of the degraded δ-grid. Let $(b_1, b_2, \ldots, b_d)$ be the lower-left corner of B. If the box B' with lower-left corner $(b_1, b_2, \ldots, b_d + \delta)$ is non-empty, then it occurs next to B in the sorted list of boxes. That is, if we are in B, then we can access B' in constant time.

We extend the algorithm: For each $1 \leq i < d$, do the following: Sort the vectors $(j_1, \ldots, j_{i-1}, j_{i+1}, \ldots j_d, j_i)$ lexicographically. Then, given box B, we can access the box with lower-left corner

$$(b_1, \ldots, b_{i-1}, b_i + \delta, b_{i+1}, \ldots, b_d),$$

provided it is non-empty, in constant time.

We summarize the result:

Theorem 1 *Let S be a set of n points in d-space and let δ be a positive real number. Assume the points of S are stored in an array $\mathcal{S}$. Moreover, assume that for each $1 \leq i \leq d$, the elements of S_i are sorted, and each element of this set contains a pointer to its occurrence in $\mathcal{S}$.*

1. *We can construct a d-dimensional degraded δ-grid for S in $O(n)$ time using $O(n)$ space.*
2. *Given a query point $q \in \mathbb{R}^d$, we can in $O(\log n)$ time decide if the box of the degraded grid that contains q is empty or not. If this box is non-empty, we can access it in $O(\log n)$ time.*
3. *Given a non-empty box B of the degraded grid and an integer $1 \leq i \leq d$, we can access the box that is immediately to the right of B along the i-th dimension—provided it is non-empty—in constant time.*

Remark: We have defined degraded grids for which the value of δ is the same for each dimension. Of course, we can also define degraded grids having a slab-width of δ_i along the i-th dimension, $1 \leq i \leq d$. It is clear that all results of this section still hold.

3 Translating a box such that it contains at least k points: a restricted case

Let $a_1, a_2, \ldots, a_d$ be positive real numbers, let $S \subseteq \mathbb{R}^d$ be a set of n points that are contained in the box $B := \prod_{i=1}^{d}[-a_i : a_i]$, and let k be a positive integer. We want an algorithm that decides if the box $\prod_{i=1}^{d}[0 : a_i]$ can be translated such that it contains at least k points of S. It is allowed that the translate of this box intersects the boundary of B.

For simplicity, we only treat the case that all a_i's are equal, say $a_i = 1$ for all $1 \leq i \leq d$. Treating the general case can be done in exactly the same way. Since this does not give any more insight into the algorithm, we leave the details to the reader.

So our problem is as follows. Let S be a set of n points that are contained in the 2-box $[-1 : 1]^d$, and let k be a positive integer. We want an algorithm that decides if

there is a 1-box that contains at least k points of S. If such a 1-box exists, we want to compute it.

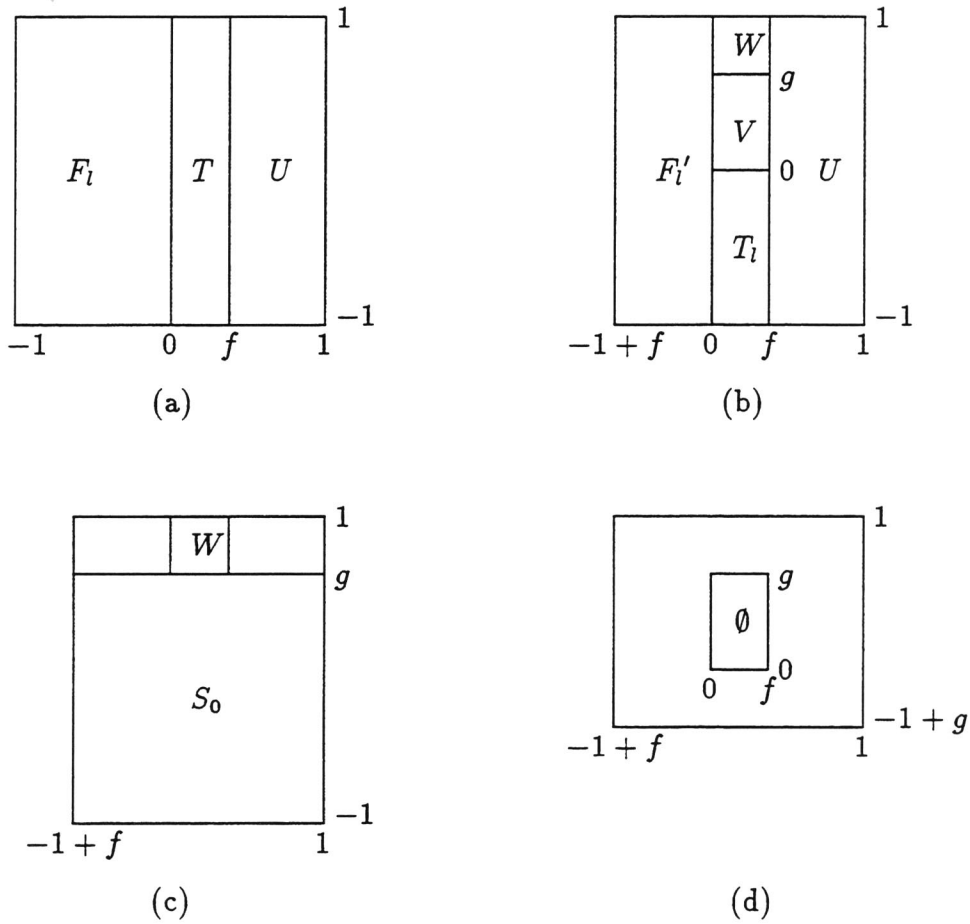


Figure 2: (a) $F_r = T \cup U$ and $|T| = |U| \geq n/4$; (b) $T_r = V \cup W$ and $|V| = |W| \geq n/16$; (c) $|S_0| \leq 15n/16$; (d) is there a 1-box that contains at least $k - |V|$ points of $S'' \setminus V$?

We first give an intuitive description of the algorithm for the planar case. Refer to Figure 2(a). Partition S into sets F_l resp. F_r consisting of those points that lie to the left resp. right of the y-axis. Assume w.l.o.g. that $|F_r| \geq n/2$. Compute the median f of the x-coordinates of the points in F_r. Then, partition F_r into sets T resp. U consisting of those points having an x-coordinate less than resp. at least equal to f.

We recursively solve the problem for the set $F_l \cup T$. Note that this set has size at most $3n/4$. Clearly, if there is a 1-box that contains at least k points of $F_l \cup T$, we are done. So, assume there is no such 1-box. Then, if there is a 1-box containing at least k points of S, this box must contain points of U. Moreover, such a 1-box cannot contain any points of F_l that have an x-coordinate less than $-1+f$. Therefore, we remove these points from S. Let S' be the resulting set and let $F_l' = S' \cap F_l$.

We proceed as follows. Refer to Figure 2(b). Consider the set T. Partition this set into sets T_l resp. T_r consisting of those points that lie below resp. above the x-axis. Assume w.l.o.g. that $|T_r| \geq |T|/2 \geq n/8$. Compute the median g of the y-coordinates

of the points in T_r. Partition T_r into sets V resp. W consisting of those points having a y-coordinate less than resp. at least equal to g.

Let S_0 be the set of all points in S' that have a y-coordinate less than g. (See Figure 2(c).) We recursively solve the problem for the set S_0. (Hence, in this recursive call, we start partitioning S_0 by x-coordinate.) Since $|W| \geq n/16$, the set S_0 has size at most $15n/16$. Again, if there is a 1-box that contains at least k points of S_0, we are done. Assume there is no such 1-box. At this point, we know that if there is a 1-box containing at least k points of S, this box must contain *all* points of V. Moreover, such a 1-box cannot contain any points of S_0 that have a y-coordinate less than $-1+g$. Therefore, we remove these points from S'. Let S'' be the resulting set. Then, it suffices to solve the problem of deciding if there is a 1-box containing at least $k - |V|$ points of $S'' \setminus V$. (See Figure 2(d).) We solve this problem using the same algorithm recursively. Note that since $|V| \geq n/16$, the set $S'' \setminus V$ has size at most $15n/16$. Moreover, it is clear that we may assume that $n \geq k$. Therefore, $k - |V| \leq k - n/16 \leq 15k/16$. That is, we have reduced both parameters n and k by a constant factor.

To summarize, let $P(n,k)$ denote the problem of deciding if there is a 1-box that contains at least k points of a set $S \subseteq [-1:1]^d$ of size n. Then, in $O(n)$ time, our algorithm reduces $P(n,k)$ to problems $P(3n/4,k)$, $P(15n/16,k)$ and $P(15n/16,15k/16)$. At first sight, this does not lead to an efficient algorithm.

We may assume, however, that $n < 4k$: If $n \geq 4k$, then one of the 4 quadrants of the plane contains at least k points of S. Clearly, in this case, there is a 1-box that contains at least k points of S. That is, if $n \geq 4k$, we can solve our problem in a trivial way and there are no recursive calls.

So assume that $n < 4k$. In the first two recursive calls, the value of k remains the same, whereas the size of the point set decreases. After a constant number of such iterations, the point set will have size less than k. But then the algorithm can stop because there is no 1-box that contains at least k points of this small set. As we will see, this observation causes the running time of the entire algorithm to be $O(n)$.

This concludes the intuitive description of the algorithm. The formal algorithm is given in Figure 3. In this algorithm, the coordinates of a point p are denoted by $p_1, p_2, \ldots, p_d$. In Step 4, local variables occur that satisfy the following

Invariant:

1. for all $1 \leq j \leq d$: $-1 \leq l_j \leq 0 \leq r_j \leq 1$, and $l_j = 0$ or $r_j = 0$,
2. $0 \leq i \leq d$,
3. $F^{(i)} \subseteq S^{(i)} \subseteq S$, and $S^{(i)} \subseteq \prod_{j=1}^{d}[-1+r_j : 1+l_j]$,
4. for all $1 \leq j \leq i$, the j-th coordinates of the points in $F^{(i)}$ are contained in the interval $[l_j : r_j]$,
5. if there is a 1-box that contains at least k points of S, then there is such a 1-box that
   (a) contains at least k points of $S^{(i)}$, and that
   (b) contains the box $\prod_{j=1}^{d}[l_j : r_j]$.

Algorithm *Place*(S, n, k)

Comments: S is a set of n points in $[-1:1]^d$ and k is a positive integer. The algorithm outputs a 1-box that contains at least k points of S, if such a box exists. Otherwise, it outputs NO.

Step 1. If $n < 2^{2d+1}$, solve the problem by brute force. If a 1-box is found that contains at least k points of S, output this box and stop. Otherwise, output NO and stop. If $n \geq 2^{2d+1}$, go to Step 2.

Step 2. If $n < k$, output NO and stop. Otherwise, go to Step 3.

Step 3. If $n \geq 2^d k$, find a d-dimensional quadrant of $\mathbb{R}^d$ that contains at least k points of S. Output the 1-box that lies in this quadrant and that has a corner at the origin, and stop. Otherwise, if $n < 2^d k$, go to Step 4.

Step 4. (* We know that $k \leq n < 2^d k$. *)

begin
$l_1 := \ldots := l_d := r_1 := \ldots := r_d := 0$; $F^{(0)} := S$; $S^{(0)} := S$; $i := 0$;
(* Invariant holds. *)
while $i \neq d$
do $F_l := \{p \in F^{(i)} : p_{i+1} \leq 0\}$; $F_r := \{p \in F^{(i)} : p_{i+1} \geq 0\}$;
(* $F_l \cap F_r$ may be non-empty. *)
if $|F_l| \geq |F^{(i)}|/2$ **then** $h := l$ **else** $h := r$ **fi**;
$f :=$ median of $\{p_{i+1} : p \in F_h\}$;
$T := \{p \in F_h : |p_{i+1}| \leq |f|\}$; $U := \{p \in F_h : |p_{i+1}| \geq |f|\}$;
(* Ties are broken such that $T \cap U = \emptyset$, $|T| = \lceil |F_h|/2 \rceil$ and $|U| = \lfloor |F_h|/2 \rfloor$. *)
if $h = l$
then $S_0^{(i)} := \{p \in S^{(i)} \setminus U : p_{i+1} \geq f\}$
else $S_0^{(i)} := \{p \in S^{(i)} \setminus U : p_{i+1} \leq f\}$
fi;
$n_0^{(i)} := |S_0^{(i)}|$;
Place$(S_0^{(i)}, n_0^{(i)}, k)$;
if $h = l$
then $l_{i+1} := f$; $r_{i+1} := 0$; $S^{(i+1)} := \{p \in S^{(i)} : p_{i+1} \leq 1 + l_{i+1}\}$
else $r_{i+1} := f$; $l_{i+1} := 0$; $S^{(i+1)} := \{p \in S^{(i)} : p_{i+1} \geq -1 + r_{i+1}\}$
fi;
$F^{(i+1)} := T$;
$i := i + 1$
(* Invariant holds. *)
od;
$S^{(d+1)} := S^{(d)} \setminus F^{(d)}$; $n' := |S^{(d+1)}|$; $k' := k - |F^{(d)}|$;
Place$(S^{(d+1)}, n', k')$
end

Figure 3: Translating a 1-box such that it contains at least k points.

Lemma 2 *Let $0 \leq i < d$. If the while-loop of Step 4 makes at least $i+1$ iterations, then*

1. $|F^{(i)}| \geq n/2^{2i}$, *and*
2. $|S_0^{(i)}| < 1 + (1 - 1/2^{2i+2})n$.

If the while-loop makes d iterations, then $|S^{(d+1)}| \leq (1 - 1/2^{2d})n$ and $k' \leq (1 - 1/2^{2d})k$.

Proof: The proof of 1 follows by induction on i, because

$$|F^{(i+1)}| = |T| = \lceil |F_h|/2 \rceil \geq |F_h|/2 \geq |F^{(i)}|/4.$$

To prove 2, note that

$$|U| = \lfloor |F_h|/2 \rfloor \geq \lfloor |F^{(i)}|/4 \rfloor \geq \lfloor n/2^{2i+2} \rfloor > n/2^{2i+2} - 1.$$

Combining this with the fact that $U \subseteq S^{(i)}$ yields

$$|S_0^{(i)}| \leq |S^{(i)} \setminus U| = |S^{(i)}| - |U| < n - (n/2^{2i+2} - 1).$$

Assume the while-loop makes d iterations. Then

$$|S^{(d+1)}| = |S^{(d)}| - |F^{(d)}| \leq (1 - 1/2^{2d})n,$$

and

$$k' = k - |F^{(d)}| \leq k - n/2^{2d} \leq (1 - 1/2^{2d})k.$$

This proves the lemma. ■

Lemma 3 *Algorithm Place is correct.*

Proof: We prove the claim by induction on n. If $n < 2^{2d+1}$, the algorithm is clearly correct. Let $n \geq 2^{2d+1}$ and assume the algorithm is correct for sets of size less than n. Let S be a set of size n. If $n < k$ or $n \geq 2^d k$, then $Place(S, n, k)$ gives a correct output. So it remains to consider the case where $k \leq n < 2^d k$. By carefully inspecting the algorithm, it follows that during the while-loop of Step 4, the invariant is correctly maintained.

We prove that $Place(S, n, k)$ outputs a 1-box containing at least k points of S if and only if such a 1-box exists.

First assume there is no 1-box that contains at least k points of S. Then, for each subset S' of S, there is no 1-box that contains at least k points of S'. Therefore, by the induction hypothesis, each recursive call $Place(S_0^{(i)}, n_0^{(i)}, k)$, for $0 \leq i < d$, outputs NO. (Since $n \geq 2^{2d+1}$, Lemma 2 implies that $n_0^{(i)} < n$. Hence, the induction hypothesis can be applied.) We claim that the recursive call $Place(S^{(d+1)}, n', k')$ also outputs NO.

Assume the contrary. Then, by the induction hypothesis—which can be applied since $n' < n$—this recursive call outputs a 1-box B that contains at least k' points of $S^{(d+1)}$. The invariant implies that

$$S^{(d+1)} = S^{(d)} \setminus F^{(d)} \subseteq \prod_{j=1}^{d} [-1 + r_j : 1 + l_j],$$

$$F^{(d)} \subseteq \prod_{j=1}^{d} [l_j : r_j],$$

and

$$-1 + r_j \leq l_j \leq r_j \leq 1 + l_j.$$

Let $b_1, b_2, \ldots, b_d$ be real numbers such that

$$B = \prod_{j=1}^{d} [b_j : b_j + 1].$$

Let $1 \leq j \leq d$ and consider the j-th interval $[b_j : b_j + 1]$ of B. We consider three cases:
Case 1: $b_j < -1 + r_j$. Then we translate B along the j-th axis to the "right" until the left endpoint of its j-th interval lies at $-1 + r_j$. Note that then the j-th interval of B contains the interval $[l_j : r_j]$. Moreover, B still contains at least k' points of $S^{(d+1)}$.
Case 2: $-1 + r_j \leq b_j \leq l_j$. Then the j-th interval of B contains the interval $[l_j : r_j]$.
Case 3: $b_j > l_j$. Then we translate B along the j-th axis to the "left" until the right endpoint of its j-th interval lies at $1 + l_j$. Then, the j-th interval of B contains $[l_j : r_j]$. Moreover, B still contains at least k' points of $S^{(d+1)}$.

These three cases prove that we may assume that B is positioned such that

$$\prod_{j=1}^{d} [l_j : r_j] \subseteq B.$$

Therefore, B contains all points of $F^{(d)}$. Hence, B contains at least $k' + |F^{(d)}| = k$ points of the set $S^{(d+1)} \cup F^{(d)} \subseteq S$. This is a contradiction.

We have proved that the algorithm outputs NO, if there is no 1-box that contains at least k points of S. To prove the converse, assume there is a 1-box that contains at least k points of S. If one of the recursive calls of the while-loop finds such a 1-box, we are done. So assume no such 1-box is found during the while-loop. Consider what happens immediately after the while-loop. By the invariant, there is a 1-box that contains at least k points of $S^{(d)}$ and that contains $\prod_{j=1}^{d} [l_j : r_j]$. Moreover, $F^{(d)} \subseteq S^{(d)}$ and $F^{(d)} \subseteq \prod_{j=1}^{d} [l_j : r_j]$. Hence, there is a 1-box containing at least $k - |F^{(d)}| = k'$ points of $S^{(d)} \setminus F^{(d)} = S^{(d+1)}$. Since $|S^{(d+1)}| < n$, the induction hypothesis implies that the recursive call $Place(S^{(d+1)}, n', k')$ finds such a 1-box. Clearly, this 1-box contains at least k points of S. This completes the proof. ■

We now analyze the complexity of the algorithm. Let $T(n, k)$ denote the worst-case running time of $Place(S, n, k)$. Then, $T(n, k) = O(1)$ if $n < 2^{2d+1}$ or $n < k$. If $n \geq 2^d k$ and $n \geq 2^{2d+1}$, then $T(n, k) = O(n)$.

Assume that $n \geq 2^{2d+1}$ and $k \leq n < 2^d k$. Let $\alpha = 1 - 1/2^{2d+1}$. Then, for some constant c,

$$T(n, k) \leq cn + \sum_{i=0}^{d-1} T(|S_0^{(i)}|, k) + T(|S^{(d+1)}|, k').$$

Since $n \geq 2^{2d+1}$, Lemma 2 implies that $|S_0^{(i)}| \leq \alpha n$, $|S^{(d+1)}| \leq \alpha n$, and $k' \leq \alpha k$. Therefore,

$$T(n, k) \leq cn + d \cdot T(\alpha n, k) + T(\alpha n, \alpha k).$$

Applying this recurrence L times yields

$$T(n,k) \leq \sum_{l=0}^{L-1} \sum_{i} c\binom{l}{i} d^{l-i}\alpha^l n + \sum_{i} \binom{L}{i} d^{L-i} \cdot T(\alpha^L n, \alpha^i k).$$

We bound the values of i in these summations. Let z be such that $\alpha^{z-1}n \geq k$ and $\alpha^z n < k$. Since $n < 2^d k$, we have

$$z < 1 + d/\log(1/\alpha) \sim 1 + d2^{2d+1} \ln 2.$$

Let $l \geq z$ and let $i \leq l - z$. Then $\alpha^l n \leq \alpha^{i+z} n < \alpha^i k$. That is, for $l \geq z$, the recursion stops at terms $T(\alpha^l n, \alpha^{l-z} k)$. Moreover, for $i < l - z$, there are no terms $T(\alpha^l n, \alpha^i k)$. Therefore,

$$\begin{aligned} T(n,k) &\leq \sum_{l=0}^{L-1} \sum_{i=\max(0,l-z)}^{l} c\binom{l}{i} d^{l-i}\alpha^l n + \sum_{i=\max(0,L-z)}^{L} \binom{L}{i} d^{L-i} \cdot T(\alpha^L n, \alpha^i k) \\ &\leq \sum_{l=0}^{\infty} \sum_{i=0}^{z} c\binom{l}{i} d^i \alpha^l n. \end{aligned}$$

We use the following identity (see [6, page 199]) to compute the latter double summation:

$$\sum_{l=0}^{\infty} \binom{l}{i} \alpha^l = \frac{\alpha^i}{(1-\alpha)^{i+1}}.$$

By a straightforward calculation, we get

$$\begin{aligned} \sum_{l=0}^{\infty} \sum_{i=0}^{z} \binom{l}{i} d^i \alpha^l &= \sum_{i=0}^{z} d^i \sum_{l=0}^{\infty} \binom{l}{i} \alpha^l \\ &= \sum_{i=0}^{z} d^i \frac{\alpha^i}{(1-\alpha)^{i+1}} \\ &\leq \frac{1}{1-\alpha} \sum_{i=0}^{z} \left(\frac{d}{1-\alpha}\right)^i \\ &= 2^{2d+1} \sum_{i=0}^{z} \left(d\,2^{2d+1}\right)^i \\ &\leq 2^{2d+2} \left(d\,2^{2d+1}\right)^z \\ &= 2^{O(d^2 2^{2d})}. \end{aligned}$$

This proves that

$$T(n,k) \leq cn\, 2^{O(d^2 2^{2d})}.$$

We have proved:

Theorem 2 *Let $a_1, a_2, \ldots, a_d$ be positive real numbers, let $S \subseteq \mathbb{R}^d$ be a set of n points that are contained in the box $B := \prod_{i=1}^{d}[-a_i : a_i]$, and let k be a positive integer. In $O(n)$ time and using $O(n)$ space we can find a translate of the box $\prod_{i=1}^{d}[0 : a_i]$ that contains at least k points of S, if such a translate exists.*

4 Translating a box such that it contains at least k points: the general case

Let $a_1, a_2, \ldots, a_d$ be positive real numbers, let S be a set of n points in $\mathbb{R}^d$, and let k be a positive integer. In this section, we consider the problem of finding a translate of the box $\prod_{i=1}^{d}[0 : a_i]$ that contains at least k points of S. As in the previous section, we only treat the case that all a_i's are equal to one. That is, we want an algorithm that decides if there is a 1-box that contains at least k points of S. If such a 1-box exists, we want to compute it. We reduce this problem to that of the previous section.

Assume we already made the presorting step of Section 2. That is, the points of S are stored in an array $\mathcal{S}$. Moreover, for each $1 \leq i \leq d$, there is a list containing the points of S sorted by their i-th coordinates, and each element in this list contains a pointer to its occurrence in $\mathcal{S}$. The following algorithm solves our problem.

Step 1. Construct a degraded 1-grid for S. With each non-empty box of this grid, store a list consisting of all points of S that are contained in it.

Step 2. For each non-empty box B of the degraded 1-grid, do the following: Let $(b_1, b_2, \ldots, b_d)$ be the lower-left corner of B. Find the at most 2^d non-empty boxes of the degraded grid whose interior overlap the 2-box

$$[b_1 : b_1 + 2] \times [b_2 : b_2 + 2] \times \ldots \times [b_d : b_d + 2].$$

These boxes are found by using the neighbor pointers that are stored with B. (See Theorem 1.) Let S_B be the set of those points of S that are contained in these at most 2^d boxes. Use the algorithm of Section 3 to decide if there is a 1-box that contains at least k points of S_B.

Theorem 3 *Let $a_1, a_2, \ldots, a_d$ be positive real numbers, let S be a set of n points in $\mathbb{R}^d$ and let k be a positive integer. After the presorting step, we can in $O(n)$ time find a translate of the box $\prod_{i=1}^{d}[0 : a_i]$ that contains at least k points of S, if such a translate exists.*

Proof: Assume for simplicity that all a_i's are equal to one. Consider Step 2 of the algorithm. By the definition of degraded grid, each non-empty box of the 1-grid is a 1-box. Therefore, the set S_B is contained in a 2-box and, hence, the algorithm of Section 3 can be called on this set. The correctness of the entire algorithm follows easily.

After presorting, the degraded 1-grid can be constructed in $O(n)$ time. By Theorem 2, we spend for each non-empty box B of this grid $O(|S_B|)$ time in Step 2. Each point of S occurs in at most 2^d subsets S_B. Hence, the total time spent in Step 2 is bounded by $O(n)$. This proves that—after the presorting step—the algorithm takes $O(n)$ time. ∎

Corollary 1 *Let $a_1, a_2, \ldots, a_d$ be positive real numbers and let S be a set of n points in $\mathbb{R}^d$. In $O(n \log n)$ time and using $O(n)$ space, we can find a translate of the box $\prod_{i=1}^{d}[0 : a_i]$ that contains the maximal number of points of S.*

Proof: In $O(n \log n)$ time, perform the presorting step. Then, do a binary search in the set $\{1, 2, \ldots, n\}$. For a test value k, decide in $O(n)$ time if the box can be translated such that it contains at least k points of S. ∎

5 Finding k points with minimal L_∞-diameter

Let S be a set of n points in $\mathbb{R}^d$ and let $1 \leq k \leq n$ be an integer. We want an algorithm that finds a k-point subset of S with minimal L_∞-diameter. This is equivalent to finding a smallest d-dimensional axes-parallel cube that contains at least k points of S. Let δ^* be the side-length of such a smallest cube. Clearly, δ^* is equal to the L_∞-distance between two points of S. Therefore, we could make a binary search for δ^* in the set of $\binom{n}{2}$ L_∞-distances defined by S. It is not clear, however, how to maintain this set efficiently. Instead, as in [3, 7], we make a binary search in the larger set consisting of all differences $|p_i - q_i|$, where p and q are points of S and $1 \leq i \leq d$. Of course, we maintain the candidate differences implicitly. The following lemma is clear.

Lemma 4 *Let δ be a real number. There exists a δ-box that contains at least k points of S, if and only if $\delta \geq \delta^*$.*

The algorithm maintains the following information:

- Arrays $A_1, \ldots, A_d$ of length n, where A_i contains the points of S sorted w.r.t. their i-th coordinates. For each $1 \leq i \leq d$, each point in A_i contains a pointer to its occurrence in A_1.
- For each $1 \leq i \leq d$ and $1 \leq j < n$, we store with $A_i[j]$ an interval $[l_{ij} : h_{ij}]$, where l_{ij} and h_{ij} are integers, such that $j < l_{ij} \leq h_{ij} + 1 \leq n + 1$.
- A real number δ' and a δ'-box B.

We define the set of *candidate differences* as follows. Let $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ be two distinct points in S, and let $1 \leq i \leq d$. Moreover, let j and j' be such that $A_i[j] = p$ and $A_i[j'] = q$. Assume w.l.o.g. that $j < j'$. Then $|q_i - p_i|$ is a candidate difference iff $l_{ij} \leq j' \leq h_{ij}$. Hence, the total number of candidate differences is equal to

$$\sum_{i=1}^{d} \sum_{j=1}^{n-1} (h_{ij} - l_{ij} + 1).$$

The algorithm makes a sequence of iterations. In each iteration, this summation is decreased by a factor of at least one fourth. During the iteration, we maintain the following

Invariant:

1. $\delta' \geq \delta^*$ and B is a δ'-box that contains at least k points of S, and
2. $\delta^* = \delta'$, or δ^* is contained in the set of candidate differences.

Initialization: Build the arrays $A_1, \ldots, A_d$. For each $1 \leq i \leq d$ and $1 \leq j < n$, store with $A_i[j]$ the interval $[l_{ij} : h_{ij}] = [j + 1 : n]$. Initialize $\delta' := \infty$ and $B := \mathbb{R}^d$.

Now, the algorithm starts with the

While-loop: Repeat Steps 1, 2 and 3 until the set of candidate differences is empty:

Step 1. For each $1 \leq i \leq d$ and $1 \leq j < n$, such that $l_{ij} \leq h_{ij}$, take the pair

$$A_i[\lfloor (l_{ij} + h_{ij})/2 \rfloor] \quad \text{and} \quad A_i[j]$$

and take the (positive) difference of their i-th coordinates. Give this difference *weight* $h_{ij} - l_{ij} + 1$. This gives a sequence of at most $d(n-1)$ weighted differences.

Step 2. Compute a weighted median δ of these weighted differences.

Step 3. Use the algorithm of Section 4 to decide if there is a δ-box that contains at least k points of S.

3.1 If there is such a δ-box, then for each pair

$$A_i[\lfloor (l_{ij} + h_{ij})/2 \rfloor] \quad \text{and} \quad A_i[j]$$

selected in Step 1 whose i-th coordinates have difference at least δ, set $h_{ij} := \lfloor (l_{ij} + h_{ij})/2 \rfloor - 1$.

Moreover, if $\delta < \delta'$, then set $\delta' := \delta$ and set B to the δ-box that was found.

3.2 If such a δ-box does not exist, then for each pair

$$A_i[\lfloor (l_{ij} + h_{ij})/2 \rfloor] \quad \text{and} \quad A_i[j]$$

selected in Step 1 whose i-th coordinates have difference at most δ, set $l_{ij} := \lfloor (l_{ij} + h_{ij})/2 \rfloor + 1$.

After the while-loop: Walk along the points of S and select k points that are contained in the δ'-box B. Output δ', B and these k points.

Lemma 5 *During the while-loop, the invariant is correctly maintained.*

Proof: After the initialization, the total number of candidate differences is equal to

$$\sum_{i=1}^{d} \sum_{j=1}^{n-1} (n-j) = d\binom{n}{2},$$

i.e., the set of candidate differences equals the set of all $d\binom{n}{2}$ differences $|p_i - q_i|$. Therefore, the invariant holds initially. Consider one iteration.

First assume that Case 3.1 applies. Then, Lemma 4 implies that $\delta \geq \delta^\star$. The algorithm sets $\delta' := \min(\delta', \delta)$ and it removes differences $|p_u - q_u|$ from the set of candidate differences that are at least equal to δ. It is clear that the new value of δ' is still at least equal to $\delta^\star$, i.e., after the iteration, item 1 of the invariant still holds.

To prove that item 2 also holds, first assume that immediately before the iteration, $\delta^\star = \delta'$. Then, after this iteration, we still have $\delta^\star = \delta'$ and, hence, item 2 still holds.

It remains to consider the case that before the iteration, $\delta^\star$ is one of the candidate differences. If $\delta > \delta^\star$, then this remains true, because we only remove differences that are larger than $\delta^\star$. Otherwise, if $\delta = \delta^\star$, we have $\delta' = \delta^\star$ after the iteration, i.e., item 2 still holds.

This proves that if Case 3.1 applies, the invariant holds after the iteration.

If Case 3.2 applies, then Lemma 4 implies that $\delta < \delta^*$. Hence, we can remove differences $|p_u - q_u|$ from the set of candidate differences that are at most equal to δ, without invalidating the invariant. ∎

Lemma 6 *The while-loop makes at most* $1 + \log_{4/3}(dn^2) = O(\log n)$ *iterations.*

Proof: At the start of the while-loop, the set of candidate differences has size $d\binom{n}{2}$. In each iteration, the size of this set is decreased by a factor of at least one fourth. (See [7] for a precise proof of this.) ∎

Theorem 4 *Let S be a set of n points in $\mathbb{R}^d$ and let $1 \leq k \leq n$ be an integer. In $O(n \log n)$ time and using $O(n)$ space, we can compute a k-point subset of S that has minimal L_∞-diameter.*

Proof: Consider the real number δ', the δ'-box B and the k points that are reported after the while-loop. Since at that moment the set of candidate differences is empty, the invariant implies that $\delta' = \delta^*$. Moreover, the k points are contained in B. This proves the correctness of the algorithm. The initialization of the algorithm takes $O(n \log n)$ time. By Theorem 3 and since a weighted median can be computed in linear time, one iteration takes $O(n)$ time. Therefore, the while-loop takes $O(n \log n)$ time. After the while-loop, we need $O(n)$ time to find k points that are contained in B. This proves that the entire algorithm has a running time of $O(n \log n)$. It is clear that it uses linear space. ∎

Remark: Consider the real number δ' that is reported by the algorithm. We search for this number in the set of all possible differences $|p_i - q_i|$. Since $\delta' = \delta^*$, however, it cannot be any difference; it is an L_∞-distance between two points of S.

6 Dynamic solutions

We combine Theorem 4 with the results of [3] to obtain dynamic data structures that maintain a k-point subset with minimal L_∞-diameter, if points are inserted and deleted in S. We recall the results of [3]:

Let $\mathcal{A}$ be *any* algorithm that, given a set of n points and an integer k, computes a k-point subset with minimal L_∞-diameter. Let $t(n,k)$ resp. $s(n,k)$ be the running time resp. space complexity of this algorithm. Then, there exist data structures that maintain such a k-point subset under insertions and deletions of points.

If only insertions are allowed, then there is a constant c such that the data structure has size $O(n + s(ck,k))$ and an insertion time of $O(\log n + t(ck,k))$. If both insertions and deletions are allowed then—again for some constant c—the space bound is $O(n \log^d(n/k) + s(ck,k))$ and the amortized update time bound is

$$O(\log n \log^{d-1}(n/k) + \log^d(n/k) \log\log n + t(ck,k) \log^d(n/k)).$$

If we take for $\mathcal{A}$ the algorithm of Theorem 4, then we get:

Corollary 2 *There is a data structure that maintains a k-point subset with minimal L_∞-diameter, if points are inserted. This data structure has size $O(n)$ and an insertion time of $O(\log n + k \log k)$.*

Corollary 3 *There is a data structure that maintains a k-point subset with minimal L_∞-diameter, if points are inserted and deleted. This data structure has size $O(n \log^d(n/k))$ and an amortized update time of*

$$O(\log^d(n/k) \log\log n + k \log k \log^d(n/k)).$$

Proof: Note that for all k and n such that $1 \leq k \leq n$, $\log n + \log(n/k) \log\log n + k \log k \log(n/k) = O(\log(n/k) \log\log n + k \log k \log(n/k))$. ■

7 Concluding remarks

We have solved the problem of finding a k-point subset with minimal L_∞-diameter or, equivalently, finding a smallest d-dimensional axes-parallel cube that contains at least k points. Although the running time is $O(n \log n)$, the dependence on the dimension is much too high: The constant factor is $2^{O(d^2 2^{2d})}$. It would be interesting to improve this dependence on d.

Another open problem is to consider other measures. For example, find a smallest d-dimensional box—w.r.t. the sum of its side-lengths—that contains at least k points. Note that our algorithm for translating a box such that it contains at least k points can be applied. It is not clear, however, how to efficiently generalize the algorithm of Section 5 to arbitrary boxes.

A related open problem is, given a figure C, translate it such that it contains the maximal number of points. We solved this problem in $O(n \log n)$ time for the case where C is a box. What is the complexity for other figures C? In the planar case, if C is a disk, this problem can be solved in $O(n^2)$ time. See Chazelle and Lee [2]. Can this be improved?

References

[1] A. Aggarwal, H. Imai, N. Katoh and S. Suri. *Finding k points with minimum diameter and related problems.* J. Algorithms **12** (1991), pp. 38-56.

[2] B. Chazelle and D.T. Lee. *On a circle placement problem.* Computing **1** (1986), pp. 1-16.

[3] A. Datta, H.P. Lenhof, C. Schwarz and M. Smid. *Static and dynamic algorithms for k-point clustering problems.* Report MPI-I-93-108, Max-Planck-Institut für Informatik, Saarbrücken, 1993.

[4] D.P. Dobkin, R.L. Drysdale and L.J. Guibas. *Finding smallest polygons.* In: F.P. Preparata (ed.), Advances in Computing Research, Vol. 1, Computational Geometry, J.A.I. Press, London, 1983, pp. 181-214.

[5] D. Eppstein and J. Erickson. *Iterated nearest neighbors and finding minimal polytopes.* Proc. 4th Annual ACM-SIAM Symp. on Discrete Algorithms, 1993, pp. 64-73.

[6] R.L. Graham, D.E. Knuth and O. Patashnik. *Concrete Mathematics.* Addison-Wesley, Reading, MA, 1989.

[7] H.P. Lenhof and M. Smid. *Enumerating the k closest pairs optimally.* Proc. 33rd Annual IEEE Symp. Foundations of Computer Science, 1992, pp. 380-386.

[8] M.H. Overmars and C.K. Yap. *New upper bounds in Klee's measure problem.* SIAM J. Comput. **20** (1991), pp. 1034-1045.

[9] M. Smid. *Finding k points with a smallest enclosing square.* Report MPI-I-92-152, Max-Planck-Institut für Informatik, Saarbrücken, 1992.