Computing a largest empty anchored cylinder, and related problems

Frank Follert[*†] Elmar Schömer[*] Jürgen Sellen[*] Michiel Smid[‡]
Christian Thiel[‡]

Abstract

Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a positive weight $w(p)$. We consider the problem of computing a ray R emanating from the origin (resp. a line l through the origin) such that $\min_{p \in S} w(p) \cdot d(p, R)$ (resp. $\min_{p \in S} w(p) \cdot d(p, l)$) is maximal. If all weights are one, this corresponds to computing a silo emanating from the origin (resp. a cylinder whose axis contains the origin) that does not contain any point of S and whose radius is maximal. For $d = 2$, we show how to solve these problems in $O(n \log n)$ time, which is optimal in the algebraic computation tree model. For $d = 3$, we give algorithms that are based on the parametric search technique and run in $O(n \log^5 n)$ time. The previous best known algorithms for these three-dimensional problems had almost quadratic running time. In the final part of the paper, we consider some related problems.

1 Introduction

Geometric optimization problems in low-dimensional spaces have received great attention. See e.g. [1, 3, 7, 9]. Such problems often occur in practical situations. Consider the following example from the field of neurosurgery: A surgeon wants to remove tissue samples from the brain of a patient for diagnosis purposes. This is done by inserting a probe through a small hole in the skullcap of the patient. In order to minimize the exposure to danger, the point of entry has to be chosen in such a way that the trajectory of the probe stays away from certain brain areas. If we model this trajectory as a ray, and the brain areas we want to avoid by weighted points in three-dimensional space, then we want to find a ray R emanating from the position at which we want to remove the tissue sample such that the minimal weighted distance from any of the points to R is maximal.

We denote the Euclidean distance between a point p and the origin by $\|p\|$. Also, the Euclidean distance between two points p and q is denoted by $d(p, q)$. If p is a point in $\mathbb{R}^d$, and R is a closed subset of $\mathbb{R}^d$, then the distance between p and R is defined as $d(p, R) := \min\{d(p, q) : q \in R\}$. Finally, we define an *anchored ray* as a ray that emanates from the origin. The above mentioned optimization problem is the three-dimensional version of the following problem.

Problem 1 *Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a weight $w(p)$, which is a positive real number. Compute an anchored ray R for which* $\min_{p \in S} w(p) \cdot d(p, R)$ *is maximal.*

[*]Universität des Saarlandes, Fachbereich 14, Informatik, Lehrstuhl Prof. Hotz, Im Stadtwald, D-66041 Saarbrücken, Germany. E-mail: {follert,schoemer,sellen}@cs.uni-sb.de.

[†]This author was supported by a Graduiertenkolleg Fellowship from DFG, Germany.

[‡]Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. E-mail: {michiel,thiel}@mpi-sb.mpg.de. These authors were supported by the ESPRIT Basic Research Actions Program, under contract No. 7141 (project ALCOM II).

We get an obvious generalization if we ask for a line through the origin instead of an anchored ray:

Problem 2 *Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a weight $w(p)$, which is a positive real number. Compute a line l through the origin for which $\min_{p\in S} w(p) \cdot d(p,l)$ is maximal.*

Let R be any ray, and let $\delta \geq 0$. The set of all points in $\mathbb{R}^d$ that are at distance at most δ from R is called a *silo* with *axis* R and *radius* δ.

If each point of S has weight one, then Problem 1 asks for the silo whose axis starts in the origin, that does not contain any point of S in its interior, and that has maximal radius. Also, in this case, Problem 2 asks for the cylinder of maximal radius whose axis contains the origin and that does not contain any point of S in its interior.

Problem 1 and the application described above were posed by Prof. Hotz, and appeared for the first time in Follert's Master Thesis [3]. He shows how to solve this problem in $O(n\alpha(n)\log n)$ time when $d = 2$, and in $O(n^{2+\epsilon})$ expected time when $d = 3$. Here, $\alpha(n)$ denotes the inverse of Ackermann's function, and ϵ is an arbitrarily small positive constant.

Follert also considers Problem 2. For $d = 2$, he shows how to solve this problem in $O(n \log n)$ time. Moreover, he reduces problem Max-Gap-on-a-Circle to Problem 2. (See also Lee and Wu [9].) Hence, Problem 2 has time complexity $\Omega(n \log n)$ in the algebraic computation tree model. For $d = 3$, Follert gives an algorithm that solves Problem 2 in $O(n\lambda_6(n)\log n)$ time, where $\lambda_6(n)$ is the maximal length of any Davenport-Schinzel sequence of order six over an alphabet of size n. It is known that $\lambda_6(n)$ is slightly superlinear. (See Agarwal *et al.* [2].) Hence, Follert's algorithm has almost quadratic running time.

1.1 Our contribution

In Section 2, we prove some preliminary results. First, we show that we can assume w.l.o.g. that all points have weight one, i.e., it suffices to consider the unit-weight versions of Problems 1 and 2. (This observation appears already in [3, 9].) Then we show that the time complexity of Problem 2 is bounded above by that of Problem 1.

In Section 3, we consider the two-dimensional version of Problem 1. We give an extremely simple algorithm that solves this problem in $O(n \log n)$ time. This algorithm uses the lower envelope of some appropriately chosen curves. A careful analysis shows that this lower envelope has linear combinatorial complexity.

The results of Section 2 imply that the two-dimensional version of Problem 2 can also be solved in $O(n \log n)$ time. Since Follert [3] proved an $\Omega(n \log n)$ lower bound for this problem, it follows that our algorithms for solving the planar versions of Problems 1 and 2 are optimal in the algebraic computation tree model.

In Section 4, we consider the three-dimensional version of Problem 1. The appropriate technique to apply seems to be Megiddo's parametric search [10]. We show that this is indeed true. In particular, we show that it suffices to design sequential and parallel algorithms for the following problem: Given a set of n disks on the unit sphere, decide whether these disks cover the sphere. Then, Megiddo's technique immediately solves Problem 1. Our algorithms for solving the sphere cover problem are based on the following topological fact: The boundary of the union of the n disks has combinatorial complexity $O(n)$ and can be computed by a divide-and-conquer algorithm. (See Kedem *et al.* [8].) The overall algorithm for solving Problem 1 has running time $O(n \log^5 n)$. Another solution based on the parametric search technique, which requires only running time $O(n \log^4 n)$, is proposed in [4]. In contrast to this approach, which uses additional geometric properties, our solution uses only topological information and can be easier modified for other obstacles.

By the results of Section 2, the three-dimensional version of Problem 2 can be solved within the same time bound. Compared with the previous almost quadratic time bounds of [3], these are drastic improvements.

In Section 5, we consider some related problems. In particular, the dual of the three-dimensional version of Problem 1, which asks for an anchored ray R for which $\max_{p \in S} w(p) \cdot d(p, R)$ is minimal, can be solved in $O(n \log^5 n)$ time using basically the same approach as in Section 4. We also discuss the dual of the three-dimensional version of Problem 2, which seems to be much more difficult. Finally, for $d = 3$, we show how to compute a plane H through the origin such that $\max_{p \in S} w(p) \cdot d(p, H)$ is minimal, in $O(n \log n)$ time. It was proved in [9] that the planar version of the latter problem has an $\Omega(n \log n)$ lower bound in the algebraic computation tree model. Hence, our algorithm is optimal in this model.

2 Some preliminary results

Let S be a set of points in $\mathbb{R}^d$. If S contains the origin, then any anchored ray R (resp. any line l through the origin) is a solution to Problem 1 (resp. 2). Therefore, from now on, we assume that set S does not contain the origin.

Lemma 1 *Let $p = (p_1, p_2, \ldots, p_d)$ be a point in $\mathbb{R}^d$, let w be a positive real number, and let R be an anchored ray in $\mathbb{R}^d$. Let $p' := (wp_1, wp_2, \ldots, wp_d)$. Then $w \cdot d(p, R) = d(p', R)$.*

Corollary 1 *Let $T(n)$ denote the complexity of the unit-weight version of Problem 1. Then the weighted version of Problem 1 has complexity $O(T(n))$.*

Lemma 2 *Let $T(n)$ be the complexity of Problem 1. Then the complexity of Problem 2 is bounded by $O(T(2n))$.*

Proof: Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a positive weight $w(p)$. We want to compute a line l through the origin for which $\min_{p \in S} w(p) \cdot d(p, l)$ is maximal. Let $S' := S \cup -S$, where $-S := \{(-p_1, -p_2, \ldots, -p_d) : (p_1, p_2, \ldots, p_d) \in S\}$. We give each point in $-S$ the weight of the corresponding point of S. Let R^* be the anchored ray such that $\min_{p \in S'} w(p) \cdot d(p, R^*)$ is maximal. Let l^* be the line that supports R^*. Then, l^* is a solution to Problem 2 for the set S. ■

3 Problem 1: the two-dimensional case

Let S be a set of n points in the plane, and let each point p of S have a positive weight $w(p)$. We want to compute an anchored ray R such that $\min_{p \in S} w(p) \cdot d(p, R)$ is maximal. By Corollary 1, we can assume w.l.o.g. that $w(p) = 1$ for all points p. Define

$$\delta^* := \max\{\min_{p \in S} d(p, R) : R \text{ is an anchored ray}\}.$$

Let δ_l^* (resp. δ_r^*) denote the analogous quantity where we only consider anchored rays that lie on or to the left (resp. right) of the y-axis. It is clear that $\delta^* = \max(\delta_l^*, \delta_r^*)$. We show how to compute δ_r^*. The value δ_l^* can be computed in a symmetric way.

Let $\delta_{min} := \min\{\|p\| : p \in S\}$. For each $\delta \geq 0$ and each point p of S, let D_p^δ denote the disk with center p and radius δ. For $0 \leq \delta \leq \delta_{min}$ and $p \in S$, let C_p^δ denote the cone consisting of all anchored rays that intersect or touch the disk D_p^δ. (Since $\delta \leq \delta_{min}$, D_p^δ does not contain the origin. Therefore, C_p^δ really is a cone.) Note that C_p^δ has the origin as its apex.

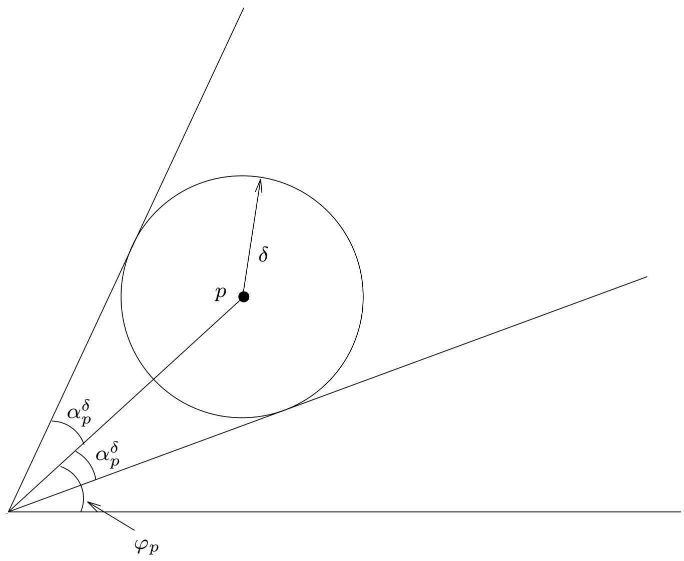

Figure 1: Illustration of the angles φ_p and α_p^δ.

Observation 1 *Using these notations, we have*

1. δ_r^* *is the maximal value of* δ, $0 \le \delta \le \delta_{min}$, *such that there is an anchored ray in the halfplane* $x \ge 0$ *that does not intersect the interior of any disk* D_p^δ, $p \in S$.
2. $0 \le \delta_r^* \le \delta_{min}$.
3. δ_r^* *is the minimum of* δ_{min} *and the minimal value of* δ, $0 \le \delta \le \delta_{min}$, *such that the cones* C_p^δ, $p \in S$, *cover the halfplane* $x \ge 0$.

Let $0 \le \delta \le \delta_{min}$ and let $p \in S$. Consider the intersection of the cone C_p^δ with the halfplane $x \ge 0$. Let $I_p(\delta)$ be the interval of slopes spanned by all anchored rays that lie in this intersection. We represent each slope by the angle between the ray and the positive x-axis. Hence, $I_p(\delta) \subseteq [-\pi/2, \pi/2]$. We can easily write down this interval explicitly:

Let p have coordinates (p_1, p_2), and let φ_p, $-\pi < \varphi_p \le \pi$, be the angle between the vector $\vec{p}$ and the positive x-axis. Then, $\sin \varphi_p = p_2/\|p\|$. Also, for $0 \le \delta \le \delta_{min}$, let α_p^δ be the angle between $\vec{p}$ and an anchored ray that is tangent to the disk D_p^δ. (There are two such tangents, but both define the same angle.) Then, $0 \le \alpha_p^\delta \le \pi/2$ and $\sin \alpha_p^\delta = \delta/\|p\|$. (See Figure 1.) If $p_1 \ge 0$, then

$$I_p(\delta) = \begin{cases} [\varphi_p - \alpha_p^\delta, \varphi_p + \alpha_p^\delta] & \text{if } 0 \le \delta \le \delta_{min} \text{ and } \delta \le p_1, \\ [\varphi_p - \alpha_p^\delta, \pi/2] & \text{if } p_1 \le \delta \le \delta_{min} \text{ and } p_2 \ge 0, \\ [-\pi/2, \varphi_p + \alpha_p^\delta] & \text{if } p_1 \le \delta \le \delta_{min} \text{ and } p_2 \le 0. \end{cases}$$

If $p_1 \le 0$, then

$$I_p(\delta) = \begin{cases} \emptyset & \text{if } 0 \le \delta \le \delta_{min} \text{ and } \delta \le -p_1, \\ [\varphi_p - \alpha_p^\delta, \pi/2] & \text{if } -p_1 \le \delta \le \delta_{min} \text{ and } p_2 \ge 0, \\ [-\pi/2, \varphi_p + \alpha_p^\delta] & \text{if } -p_1 \le \delta \le \delta_{min} \text{ and } p_2 \le 0. \end{cases}$$

It is clear that the cones C_p^δ, $p \in S$, cover the halfplane $x \ge 0$ if and only if the intervals $I_p(\delta)$, $p \in S$, cover $[-\pi/2, \pi/2]$. Hence, δ_r^* is the minimum of (1) δ_{min}, and (2) the minimal value of δ, $0 \le \delta \le \delta_{min}$, such that the intervals $I_p(\delta)$ cover $[-\pi/2, \pi/2]$.

Using the intervals $I_p(\delta)$ has the disadvantage that we need non-algebraic functions. In order to stay within the algebraic computation tree model, our algorithm works with the intervals

$$J_p(\delta) := \sin(I_p(\delta)) = \{\sin\gamma : \gamma \in I_p(\delta)\}.$$

Note that $I_p(\delta) \subseteq [-\pi/2, \pi/2]$ and that the function $\sin(\cdot)$ is increasing on $[-\pi/2, \pi/2]$. Using the relations $\sin\varphi_p = p_2/\|p\|$, $\cos\varphi_p = p_1/\|p\|$, $\sin\alpha_p^\delta = \delta/\|p\|$, $\cos\alpha_p^\delta = \sqrt{p_1^2+p_2^2-\delta^2}/\|p\|$, and $\sin(x+y) = \sin x\cos y + \cos x \sin y$, we get the following expressions for $J_p(\delta)$. If $p_1 \geq 0$, then

$$J_p(\delta) = \begin{cases} \left[\frac{p_2\sqrt{p_1^2+p_2^2-\delta^2}-p_1\delta}{\|p\|^2}, \frac{p_2\sqrt{p_1^2+p_2^2-\delta^2}+p_1\delta}{\|p\|^2}\right] & \text{if } 0 \leq \delta \leq \delta_{min} \text{ and } \delta \leq p_1, \\ \left[\frac{p_2\sqrt{p_1^2+p_2^2-\delta^2}-p_1\delta}{\|p\|^2}, 1\right] & \text{if } p_1 \leq \delta \leq \delta_{min} \text{ and } p_2 \geq 0, \\ \left[-1, \frac{p_2\sqrt{p_1^2+p_2^2-\delta^2}+p_1\delta}{\|p\|^2}\right] & \text{if } p_1 \leq \delta \leq \delta_{min} \text{ and } p_2 \leq 0. \end{cases}$$

If $p_1 \leq 0$, then

$$J_p(\delta) = \begin{cases} \emptyset & \text{if } 0 \leq \delta \leq \delta_{min} \text{ and } \delta \leq -p_1, \\ \left[\frac{p_2\sqrt{p_1^2+p_2^2-\delta^2}-p_1\delta}{\|p\|^2}, 1\right] & \text{if } -p_1 \leq \delta \leq \delta_{min} \text{ and } p_2 \geq 0, \\ \left[-1, \frac{p_2\sqrt{p_1^2+p_2^2-\delta^2}+p_1\delta}{\|p\|^2}\right] & \text{if } -p_1 \leq \delta \leq \delta_{min} \text{ and } p_2 \leq 0. \end{cases}$$

The value of δ_r^* is equal to the minimum of δ_{min} and the minimal value of δ, $0 \leq \delta \leq \delta_{min}$, such that the intervals $J_p(\delta)$ cover $[-1, 1]$. For $p \in S$, let

$$R_p := \{(x, \delta) : 0 \leq \delta \leq \delta_{min}, x \in J_p(\delta)\}.$$

The region R_p is contained in the rectangle $[-1, 1] \times [0, \delta_{min}]$.

Observation 2 *δ_r^* is the minimum of (1) δ_{min}, and (2) the minimal value of δ, $0 \leq \delta \leq \delta_{min}$, such that the horizontal segment with endpoints $(-1, \delta)$ and $(1, \delta)$ is completely contained in $\bigcup_{p\in S} R_p$.*

Let l_p be the lower envelope of R_p. Then, l_p is the graph of a continuous function on a subinterval of $[-1, 1]$. Finally, let L be the lower envelope of the graphs l_p, $p \in S$, and the line segment with endpoints $(-1, \delta_{min})$ and $(1, \delta_{min})$.

Observation 3 *δ_r^* is the y-coordinate of a highest vertex of L.*

We now analyze the lower envelope L. Let B_l, B_r, B_t and B_b be the left, right, top and bottom side of the rectangle $[-1, 1] \times [0, \delta_{min}]$, respectively.

Let $p = (p_1, p_2)$ be a point of S, and consider the graph l_p. If $p_1 \geq 0$, then l_p consists of a decreasing part l_p^- that has $(p_2/\|p\|, 0)$ as its lowest and rightmost endpoint, and an increasing part l_p^+ that has $(p_2/\|p\|, 0)$ as its lowest and leftmost endpoint. Moreover, l_p^- (resp. l_p^+) has its leftmost (resp. rightmost) endpoint on B_l or B_t (resp. B_r or B_t). If $p_1 \leq 0$ and $p_2 \geq 0$, then l_p is decreasing from some point on B_t to some point on B_r. Finally, if $p_1 \leq 0$ and $p_2 \leq 0$, then l_p is increasing from some point on B_l to some point on B_t.

Lemma 3 *Let $p = (p_1, p_2)$ and $q = (q_1, q_2)$ be two distinct points of S. The graphs l_p and l_q intersect at most twice.*

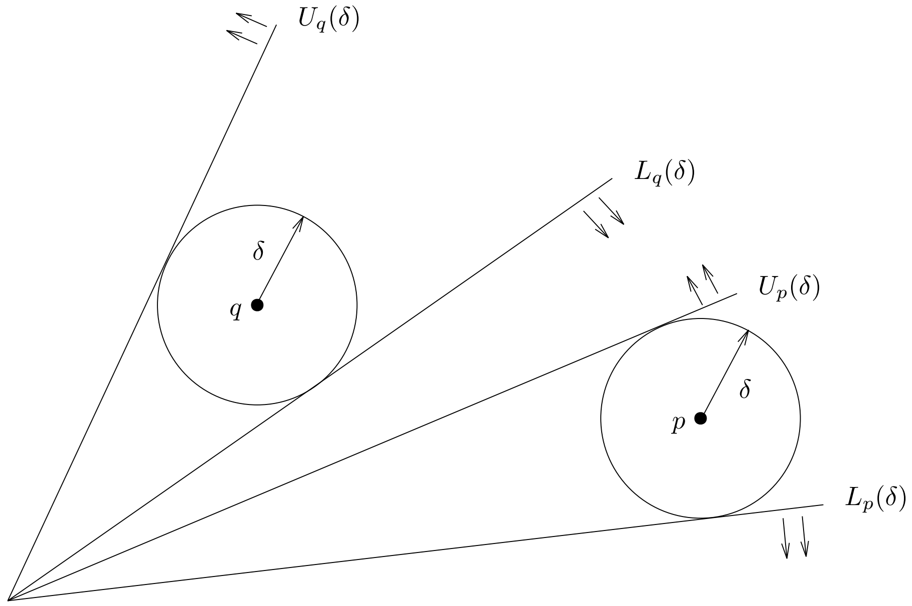

Figure 2: Growing δ from 0 to δ_{min}.

Proof: We give a geometric explanation for this claim. In the full paper, we give a rigorous proof. Assume that p_1 and q_1 are both positive and that $\varphi_q > \varphi_p$. For $0 \leq \delta \leq \delta_{min}$, let $U_p(\delta)$ (resp. $L_p(\delta)$) be the anchored ray that is upper (resp. lower) tangent to the disk D_p^δ. Define $U_q(\delta)$ and $L_q(\delta)$ analogously.

Intersections of l_p and l_q are in one-to-one correspondence with values of δ such that $\{U_p(\delta), L_p(\delta)\} \cap \{U_q(\delta), L_q(\delta)\} \neq \emptyset$.

Consider what happens when we grow δ from 0 to δ_{min}. Initially, $U_p(\delta) = L_p(\delta)$ and $U_q(\delta) = L_q(\delta)$. If δ increases, then the tangents $U_p(\delta)$ and $L_p(\delta)$ move in opposite directions. Similarly, the tangents $U_q(\delta)$ and $L_q(\delta)$ move in opposite directions. (See Figure 2.) Clearly, there is exactly one δ_0 such that $L_q(\delta_0) = U_p(\delta_0)$. This corresponds to an intersection of l_q^- and l_p^+. Also, for $\delta < \delta_0$, there are no intersections between l_p and l_q. Now we grow δ further, from δ_0 to the next "time" δ_1 at which $\{U_p(\delta_1), L_p(\delta_1)\} \cap \{U_q(\delta_1), L_q(\delta_1)\} \neq \emptyset$. (If there is no such time, then the graphs l_p and l_q intersect exactly once, and we are done.) Then, $U_p(\delta_1) = U_q(\delta_1)$ or $L_p(\delta_1) = L_q(\delta_1)$. Assume w.l.o.g. that at time δ_1, $U_p(\delta_1) = U_q(\delta_1)$. This corresponds to the second intersection between l_p and l_q; more precisely, an intersection between l_p^+ and l_q^+. Note that then $U_p(\delta)$ must move faster than $U_q(\delta)$. Hence, for $\delta > \delta_1$, these two tangents never coincide any more. That is, l_p^+ and l_q^+ intersect only once. Now look at $L_p(\delta)$ and $L_q(\delta)$: Since $L_p(\delta)$ and $U_p(\delta)$ (resp. $L_q(\delta)$ and $U_q(\delta)$) move at the same, but opposite, velocities, $L_q(\delta)$ will never overtake $L_p(\delta)$. That is, l_p^- and l_q^- do not intersect. ∎

Lemma 4 *The lower envelope L consists of $O(n)$ vertices.*

Proof: We will show that the names of the points that correspond to the edges of L, when we traverse L from left to right, form a Davenport-Schinzel sequence of order two. This will prove the claim. (See e.g. [6].) Hence, we must show that for any pair p and q of distinct points of S, this sequence of names does not contain a subsequence of the form $p \ldots q \ldots p \ldots q$. But this follows from the fact that l_p and l_q intersect at most twice, and from the restrictions on the endpoint of these graphs. ∎

Now we are ready to give the algorithm for computing δ_r^* and a corresponding ray R^*.

1. Compute the graphs l_p, $p \in S$.
2. Compute the lower envelope L of the graphs l_p, $p \in S$, and the horizontal segment with endpoints $(-1, \delta_{min})$ and $(1, \delta_{min})$.
3. Walk along L and find a highest vertex on it. Let this vertex have coordinates (a, δ).
4. Output δ and the anchored ray $R := \{(x, ax/\sqrt{1-a^2}) : x \geq 0\}$.

To prove the correctness of this algorithm, consider the vertex (a, δ) that is found in Step 3. Observation 3 implies that $\delta = \delta_r^*$. Let φ be the angle such that $-\pi/2 \leq \varphi \leq \pi/2$ and $\sin\varphi = a$. Let R^* be the anchored ray that makes an angle of φ with the positive x-axis. Then $\delta = \min_{p \in S} d(p, R^*)$. It is easy to see that $R = R^*$.

Next we analyze the running time of our algorithm. Step 1 takes $O(n)$ time. The lower envelope L can be computed by a divide-and-conquer algorithm. (See e.g. [6].) Since L has linear size, this algorithm, and hence Step 2, takes $O(n \log n)$ time. Step 3 takes $O(n)$ time, and Step 4 takes $O(1)$ time. We have proved the following result.

Theorem 1 *Let S be a set of n points in the plane, and let each point p of S have a positive weight $w(p)$. In $O(n \log n)$ time, we can compute an anchored ray R^* for which $\min_{p \in S} w(p) \cdot d(p, R^*)$ is maximal.*

Corollary 2 *Let S be a set of n points in the plane, and let each point p of S have a positive weight $w(p)$. In $O(n \log n)$ time, we can compute a line l^* through the origin for which $\min_{p \in S} w(p) \cdot d(p, l^*)$ is maximal.*

The results of Theorem 1 and Corollary 2 are optimal in the algebraic computation tree model. Follert [3] proves an $\Omega(n \log n)$ lower bound for Problem 2. It follows from Lemma 2 that this lower bound holds for Problem 1 as well.

4 Problem 1: the three-dimensional case

We briefly recall Megiddo's parametric search technique [10]. Suppose we are given a fixed set of n data items, such as points in $\mathbb{R}^3$. Let $\mathcal{P}(t)$ be a decision problem whose value depends on the n data items and a real parameter t. Assume that $\mathcal{P}$ is monotone, meaning that if $\mathcal{P}(t_0)$ is true for some t_0, then $\mathcal{P}(t)$ is also true for all $t < t_0$. Our aim is to find the maximal value of t for which $\mathcal{P}(t)$ is true. We denote this value by t^*.

Assume we have a sequential algorithm A_s that, given the n data items and t, decides if $\mathcal{P}(t)$ is true or not. The control flow of this algorithm is governed by comparisons, each of which involves testing the sign of some low-degree polynomial in t. Let T_s and C_s denote the running time and the number of comparisons made by algorithm A_s, respectively. Note that by running A_s on input t, we can decide if $t \leq t^*$ or $t > t^*$: we have $t \leq t^*$ iff $\mathcal{P}(t)$ is true.

The parametric search technique simulates A_s on the unknown value t^*. Whenever A_s reaches a branching point that depends on a comparison operation, the comparison can be reduced to testing the sign of a suitable low-degree polynomial $f(t)$ at $t = t^*$. The algorithm computes the roots of this polynomial and checks each root a—by running A_s on input a—to see if it is less than or equal to t^*. In this way, the algorithm identifies two successive roots between which t^* must lie and thus determines the sign of $f(t^*)$. Hence, we get an interval I that contains t^*. Also the comparison now being resolved, the execution can proceed. As we proceed through the execution, each comparison that we resolve results in constraining I further and we get a sequence of progressively smaller intervals each known to contain t^*. The simulation will run to completion and we are left with an interval I that contains t^*. It can be shown that for any real number $r \in I$, $\mathcal{P}(r)$ is true. Therefore, t^* must be the right endpoint of I.

Since A_s makes at most C_s comparisons during its execution, the entire simulation and, hence, the computation of t^* take $O(C_s T_s)$ time. To speed up this algorithm, Megiddo replaces A_s by a parallel algorithm A_p that uses P processors and runs in T_p parallel time. At each parallel step, let A_p make a maximum of W_p independent comparisons. Then our algorithm simulates A_p sequentially, again at the unknown value t^*. At each parallel step, we get at most W_p low-degree polynomials in t. We compute the roots of all of them and do a binary search among them using repeated median finding to make the probes for t^*. For each probe, we run the sequential algorithm A_s. In this way, we get the correct sign of each polynomial in t^*, and our algorithm can simulate the next parallel step of A_p.

For the simulation of each parallel step, we spend $O(W_p)$ time for median finding. Hence, the entire simulation of this step takes time $O(W_p + T_s \log W_p)$. As a result, the entire algorithm computes t^* in time $O(W_p T_p + T_s T_p \log W_p)$. Since $W_p \leq P$, the running time is bounded by $O(PT_p + T_s T_p \log P)$.

4.1 Applying the parametric search technique

Let S be a set of n points in $\mathbb{R}^3$. Each point p of S has a positive weight $w(p)$. Define

$$\delta^* := \max\{\min_{p \in S} w(p) \cdot d(p, R) : R \text{ is an anchored ray}\}.$$

Our goal is to compute δ^* together with the corresponding ray R^*. We saw already that we may assume w.l.o.g. that $w(p) = 1$ for all points p.

In order to apply the parametric search technique, we have to solve the following decision problem $\mathcal{P}(\delta)$: Given the set S and the real number $\delta \geq 0$, is there an anchored ray R such that $\min_{p \in S} d(p, R) \geq \delta$. It is clear that $\mathcal{P}$ is monotone, and δ^* is the maximal δ for which $\mathcal{P}(\delta)$ is true.

We reformulate the decision problem $\mathcal{P}(\delta)$ in the following way. Let $\delta \geq 0$. For each point p of S, let B_p^δ denote the ball with center p and radius δ. Then $\mathcal{P}(\delta)$ is true if and only if there is an anchored ray R that does not intersect the interior of any of these balls.

Let $\delta_{min} := \min\{\|p\| : p \in S\}$. Then $\mathcal{P}(\delta)$ is clearly false for $\delta > \delta_{min}$.

For $0 \leq \delta \leq \delta_{min}$, let C_p^δ denote the circular cone consisting of all anchored rays that intersect or touch the ball B_p^δ. This cone intersects the unit sphere—i.e., the surface of the ball of radius one centered at the origin—in a disk. We denote this disk by D_p^δ.

Let $0 \leq \delta \leq \delta_{min}$. It is clear that $\mathcal{P}(\delta)$ is true if and only if there is a point x on the unit sphere that is not contained in the interior of any of these n disks. If there is such a point x, then the ray R that starts in the origin and contains x satisfies $\min_{p \in S} d(p, R) \geq \delta$. In other words, $\mathcal{P}(\delta)$ is true if and only if the interiors of these n disks do not cover the unit sphere.

In the next two subsections, we give sequential and parallel algorithms that decide the latter condition.

4.1.1 A sequential algorithm that decides the covering problem

Let $D_1, D_2, \ldots, D_n$ be a set of n disks on the unit sphere. We want to decide if the interiors of these disks cover the unit sphere. Clearly, we can use the arrangement of the disks for deciding this. This arrangement, however, may have size $\Omega(n^2)$.

Let I_i (resp. γ_i) denote the interior (resp. boundary) of D_i, $1 \leq i \leq n$, and let $I := \bigcup_{i=1}^{n} I_i$. (Note that there may be $i \neq j$ such that $\gamma_i = \gamma_j$.) We denote the closure of I by $cl(I)$. The boundary B of I is equal to

$$B = cl(I) \setminus I = \left(\bigcup_{i=1}^{n} D_i \right) \setminus \left(\bigcup_{i=1}^{n} I_i \right).$$

The interiors of the disks $D_1, D_2, \ldots, D_n$ cover the unit sphere if and only if B is empty. Hence, our problem can be solved by computing the boundary B rather than the entire arrangement of the n disks.

The boundary B is a planar graph on the unit sphere. Each edge of this graph is part of a circle γ_i for some i, and each vertex is an intersection point of at least two distinct circles. We choose an arbitrary point p_i on each circle γ_i, $1 \leq i \leq n$, with the restriction that $p_i = p_j$ if $\gamma_i = \gamma_j$. Then, if γ_i does not intersect any other circle, it forms an edge of B with both endpoints equal to p_i. Note that B can have isolated vertices: If three circles intersect in one point x, and there is an arbitrarily small disk α (not equal to any of $D_1, D_2, \ldots, D_n$) centered at x such that $\alpha \setminus \{x\}$ is contained in the union of the interiors of these three circles, then x is a vertex of B, and x is not incident to any edge of B.

Lemma 5 ([8]) *The boundary B is a planar graph on the unit sphere, and, if $n \geq 3$, it contains at most $6n - 12$ vertices.*

In [8], an algorithm is given that computes the boundary of the union of n regions in the plane, each of which is bounded by a simple closed Jordan curve. This algorithm follows the divide-and-conquer paradigm, and the merge step is implemented by using a plane sweep algorithm of Ottmann, Widmayer and Wood [11] for computing the boundary of the union of superimposed polygonal planar regions. This plane sweep algorithm also works if the edges of the planar regions are curved. We can easily modify this algorithm such that it computes the boundary B:

Consider the disks $D_1, D_2, \ldots, D_n$. Recursively compute the boundary B_1 (resp. B_2) of the union of the interiors of $D_1, D_2, \ldots, D_{n/2}$ (resp. $D_{n/2+1}, D_{n/2+2}, \ldots, D_n$). Note that B_1 and B_2 are planar graphs on the unit sphere. Let l and r be the points on the unit sphere with minimal and maximal y-coordinate, respectively. Using the algorithm of [11], we compute the boundary B from B_1 and B_2 by sweeping a circular arc with endpoints l and r around the unit sphere. Let b_1 and b_2 denote the number of edges of B_1 and B_2, respectively, and let t denote the number of intersections between B_1 and B_2. Then this sweep algorithm runs in time $O((b_1 + b_2 + t) \log(b_1 + b_2))$. It follows from Lemma 5 that $b_1 + b_2 = O(n)$. Since each intersection point between B_1 and B_2 is a vertex of B, Lemma 5 also implies that $t = O(n)$. Hence, the entire sweep algorithm runs in time $O(n \log n)$. This shows that the entire divide-and-conquer algorithm for computing the boundary B takes $O(n \log^2 n)$ time. The interiors of the input disks $D_1, D_2, \ldots, D_n$ cover the unit sphere if and only if the graph B is empty. If B is not empty, then any vertex of B is a point on the unit sphere that is not contained in the interior of any disk. We have proved the following result.

Lemma 6 *Let $D_1, D_2, \ldots, D_n$ be a set of disks on the unit sphere. In $O(n \log^2 n)$ time, we can decide if the union of the interiors of these disks cover the unit sphere. If this is not the case, then the algorithm finds a point on the unit sphere that is not contained in the interior of any disk.*

4.1.2 A parallel algorithm that decides the covering problem

Now we give a parallel algorithm for computing the boundary B. Consider again the disks $D_1, D_2, \ldots, D_n$. The algorithm uses n processors. The first (resp. last) $n/2$ processors compute the boundary B_1 (resp. B_2) of the union of the interiors of $D_1, D_2, \ldots, D_{n/2}$ (resp. $D_{n/2+1}, D_{n/2+2}, \ldots, D_n$). It remains to describe the merge step. That is, given B_1 and B_2, how to compute the boundary B of the union of the interiors of the n input disks.

Rüb [12] gives a parallel algorithm based on a segment tree, that computes the intersections among red and blue curved segments in the plane. The interiors of the red (resp. blue) segments are assumed to be pairwise disjoint. Also, each segment is assumed to be x-monotone, meaning that any vertical line intersects a segment at most once. Finally, it is assumed that each red-blue pair of segments intersect at most a constant number of times. If n denotes the total number of red and blue segments, and t denotes the total number of intersection points among the red-blue pairs of segments, then Rüb's algorithm runs on a CREW-PRAM in time $O(\log n + t/n)$ using n processors.

This algorithm can be used to compute the boundary B from B_1 and B_2: In our case, the slabs that define the segment tree are bounded by circular arcs on the unit sphere with two fixed diametral endpoints. In order to guarantee that each curved edge of B_1 and B_2 is monotone, we cut each of them into at most two parts. Note that, by Lemma 5, $t = O(n)$. Hence, using Rüb's algorithm, we compute all intersections of B_1 and B_2 in $O(\log n)$ time using n processors.

Then, for each edge e of B_1, we sort the intersection points on this edge. This gives the arrangement A of the union B_1 and B_2. Given this arrangement, we compute the boundary B by removing the appropriate vertices and edges from A. All this can be done in $O(\log n)$ time using n processors.

Hence, the entire merge step of our parallel divide-and-conquer algorithm takes $O(\log n)$ time and uses n processors. This proves:

Lemma 7 *Let $D_1, D_2, \ldots, D_n$ be a set of disks on the unit sphere. There is a CREW-PRAM algorithm that decides if the union of the interiors of these disks cover the unit sphere. If this is not the case, then the algorithm finds a point on the unit sphere that is not contained in the interior of any disk. The algorithm takes $O(\log^2 n)$ time and uses n processors.*

Lemmas 6 and 7, and the parametric search technique immediately provide a solution for Problem 1:

Theorem 2 *Let S be a set of n points in $\mathbb{R}^3$, and let each point p of S have a positive weight $w(p)$. In $O(n \log^5 n)$ time, we can compute an anchored ray R^* for which $\min_{p \in S} w(p) \cdot d(p, R^*)$ is maximal.*

Corollary 3 *Let S be a set of n points in $\mathbb{R}^3$, and let each point p of S have a positive weight $w(p)$. In $O(n \log^5 n)$ time, we can compute a line l^* through the origin for which $\min_{p \in S} w(p) \cdot d(p, l^*)$ is maximal.*

5 Some related problems

Problem 3 *Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a weight $w(p)$, which is a positive real number. Compute an anchored ray R for which $\max_{p \in S} w(p) \cdot d(p, R)$ is minimal.*

As before, we can assume w.l.o.g. that all points have weight one. In [9], Lee and Wu show how to solve this problem in $O(n \log n)$ time when $d = 2$. We show how to solve it for $d = 3$. Let B_p^δ denote the ball with center p and radius δ. Then we want to compute the minimal real number $\delta \geq 0$ such that there is an anchored ray that intersects all balls B_p^δ, $p \in S$. We find this minimal δ using the parametric search technique.

Let $\delta \geq 0$. We need sequential and parallel algorithms for deciding if there is an anchored ray that intersects all balls B_p^δ, $p \in S$. Clearly, we do not have to consider those balls that contain the origin. Using the same approach as in Section 4, we arrive at the following problem: Given a set of at most n disks on the unit sphere, decide if their intersection is empty. This intersection has combinatorial complexity $O(n)$. Moreover, it can be computed by basically the same approaches as in Sections 4.1.1 and 4.1.2.

Theorem 3 *Let S be a set of n points in $\mathbb{R}^3$, and let each point p of S have a positive weight $w(p)$. In $O(n \log^5 n)$ time, we can compute an anchored ray R for which $\max_{p \in S} w(p) \cdot d(p, R)$ is minimal.*

Problem 4 *Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a weight $w(p)$, which is a positive real number. Compute a line l through the origin for which $\max_{p \in S} w(p) \cdot d(p, l)$ is minimal.*

For $d = 2$, this problem can be solved in $O(n \log n)$ time, which is optimal in the algebraic computation tree model. See [9]. The three-dimensional version seems to be much harder. Follert [3] solves this problem in $O(n \lambda_6(n) \log n)$ time.

A *symmetric slab* is defined as the region between two parallel planes in $\mathbb{R}^3$ that are at the same distance from the origin. If we intersect a symmetric slab with the unit sphere, then we get a *symmetric slab on the unit sphere*. A natural approach to solve the three-dimensional version of Problem 4 is to use the parametric search technique. Then we have to design sequential and parallel algorithms for the following decision problem: Given a set of n symmetric slabs on the unit sphere, do they cover the unit sphere.

This decision problem resembles the following problem: Given a circle C and a set of n slabs, both in the plane, decide whether these slabs cover C. Gajentaan and Overmars [5] proved that this problem is n^2-hard, which indicates that it is probably very hard to find a subquadratic algorithm for it.

Open problem 1 *Decide if the problem "Given a set of n symmetric slabs on the unit sphere, do they cover the unit sphere", is n^2-hard, or if it can be solved in subquadratic time. Note that if this problem is n^2-hard, that then also the three-dimensional version of Problem 4 is n^2-hard.*

Problem 5 *Let S be a set of n points in $\mathbb{R}^d$, and let each point p of S have a weight $w(p)$, which is a positive real number. Compute a hyperplane H through the origin for which $\max_{p \in S} w(p) \cdot d(p, H)$ is minimal.*

Lee and Wu [9] proved an $\Omega(n \log n)$ lower bound for the planar version of this problem. We show how to solve the three-dimensional version of Problem 5 in $O(n \log n)$ time. We can assume w.l.o.g. that all points of S have weight one.

Our problem is equivalent to that of computing the symmetric slab of minimal width that contains all points of S. Let $S' := S \cup -S$, where $-S := \{(-p_1, -p_2, -p_3) : (p_1, p_2, p_3) \in S\}$. For any plane H through the origin, we have $H = -H$. Therefore, $d(p, H) = d(-p, -H) = d(-p, H)$. As a result, it suffices to solve our problem for the set S'. Since this set is symmetric w.r.t. the origin, the width of the minimal symmetric slab containing S' is equal to the width of S', which is defined as the minimal width of *any* slab containing this set.

The best known algorithm for computing the width of an arbitrary set of n points in $\mathbb{R}^3$ has running time $O(n^{17/11+\epsilon})$, where ϵ is an arbitrarily small positive constant. (See Agarwal *et al.* [1].) In our case, however, the set of points has a special form.

Houle and Toussaint [7] observed that the width of a set of points in $\mathbb{R}^3$ is the minimum distance between parallel planes of support passing through either an antipodal vertex-face pair or an antipodal edge-edge pair of the convex hull of the set.

It is not difficult to see that in order to compute the width of our set S', we only have to consider parallel planes of support passing through an antipodal vertex-face pair of the convex hull of S', and take the minimum distance between any such pair of planes. This minimum distance can be computed in $O(n \log n)$ time. (See [7].)

Theorem 4 *Let S be a set of n points in $\mathbb{R}^3$, and let each point p of S have a positive weight $w(p)$. In $O(n \log n)$ time, we can compute a plane H through the origin for which $\max_{p \in S} w(p) \cdot d(p, H)$ is minimal. This is optimal in the algebraic computation tree model.*

Acknowledgement

The authors thank Prof. Hotz for posing the problems, and the application to neurosurgery, that were considered in this paper. They also thank Christine Rüb and Stefan Schirra for several helpful discussions.

References

[1] P.K. Agarwal, B. Aronov, and M. Sharir. *Computing enveloped in four dimensions with applications.* Proc. 10th Annual ACM Conf. on Comp. Geom., 1994, pp. 348–358.

[2] P.K. Agarwal, M. Sharir, and P. Shor. *Sharp upper and lower bounds for the length of general Davenport-Schinzel sequences.* J. Combin. Theory, Ser. A **52** (1989), pp. 228–274.

[3] F. Follert. *Lageoptimierung nach dem Maximin-Kriterium.* Master's Thesis. Department of Computer Science, Universität des Saarlandes, Saarbrücken, 1994.

[4] F. Follert. *Maxmin location of an anchored ray in* 3*-space and related problems,* submitted to CCCG95

[5] A. Gajentaan and M.H. Overmars. *n^2-Hard problems in computational geometry.* Tech. Rep. RUU-CS-93-15, Department of Computer Science, University of Utrecht, 1993.

[6] L. Guibas and M. Sharir. *Combinatorics and algorithms of arrangements.* In: New Trends in Discrete and Computational Geometry, Ed. J. Pach. Springer-Verlag, Berlin, 1993, pp. 9–36.

[7] M.E. Houle and G.T. Toussaint. *Computing the width of a set.* IEEE Trans. Pattern Anal. Mach. Intell., PAMI-10 (1988), pp. 761–765.

[8] K. Kedem, R. Livne, J. Pach, and M. Sharir. *On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles.* Discrete Comput. Geom. **1** (1986), pp. 59–71.

[9] D.T. Lee and Y.F. Wu. *Geometric complexity of some location problems.* Algorithmica **1** (1986), pp. 193–211.

[10] N. Megiddo. *Applying parallel computation algorithms in the design of serial algorithms.* J. ACM **30** (1983), pp. 852–865.

[11] T. Ottmann, P. Widmayer, and D. Wood. *A fast algorithm for Boolean mask operations.* Comput. Vision Graph. Image Process. **30** (1985), pp. 249–286.

[12] C. Rüb. *Computing intersections and arrangements for red-blue curve segments in parallel.* Proc. 4th Canadian Conf. on Comp. Geom., 1992, pp. 115–120.