Ray Shooting in Convex Polytopes*

Otfried Schwarzkopf
Utrecht University, Department of Computer Science,
P.O. Box 80·089, 3508 TB Utrecht, the Netherlands

Abstract

Let H be a set of n halfspaces in $\mathbf{E}^d$ (where the dimension $d \geq 4$ is fixed), and $\mathcal{P}(H)$ the convex polytope defined as their intersection. We show that $\mathcal{P}(H)$ can be preprocessed in time and space $\mathcal{O}(n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor - \varepsilon})$ (for any fixed $\varepsilon > 0$) so that ray shooting queries with rays starting in $\mathcal{P}(H)$ can be answered in time $\mathcal{O}(\log n)$. This improves previous bounds by obtaining optimal query time and by improving the product $Q(n)S(n)^{1/\lfloor d/2 \rfloor}$ ($Q(n)$ and $S(n)$ denoting query time and storage of a data structure for this problem) to $\mathcal{O}(n(\log n)^{\varepsilon})$ for an arbitrarily small $\varepsilon > 0$. By a well known lifting transformation, the results imply the same bounds for nearest (or furthest) neighbor queries in space of one dimension lower, which is an improvement in itself. We furthermore show that a structure for the ray shooting problem can be dynamically maintained under a sequence of random *insertions, using the history of the maintenance process for the polytope as a point location data structure. The expected update time is $\mathcal{O}(m^{\lfloor d/2 \rfloor - 1}(\log m)^{O(1)})$, the query time for ray shooting queries is $\mathcal{O}(\log^2 m)$ with high probability, where m is the current number of points.*

*This research was supported by the ESPRIT II Basic Research Action of the European Community under contract No. 3075 (project ALCOM). It was done while the author was employed at Freie Universität Berlin. Part of it was done during the First Utrecht Workshop on Computational Geometry, supported by the Netherlands' Organization for Scientific Research (NWO).

8th Annual Computational Geometry, 6/92, Berlin, Germany
©1992 ACM 89791-518-6/92/0006/0286 ..... $1.50

1 Introduction

We consider the following problem: Given a polytope $\mathcal{P}(H)$ defined as the intersection of a set H of n halfspaces, preprocess them into a data structure such that queries of the form: *What is the facet intersected by a query ray starting within $\mathcal{P}(H)$?* can be answered efficiently.

The most efficient previous solution to the d-dimensional problem is due to Agarwal and Matoušek [AM92]. They have shown how to answer ray shooting queries by applying Meggido's parametric search to a data structure for queries of the form: "does a query point lie in the polytope $\mathcal{P}(H)$?" Using a data structure for the latter problem by Clarkson [Cla88] with $\mathcal{O}(n^{\lfloor d/2 \rfloor + \varepsilon})$ storage (for an arbitrarily small but fixed constant $\varepsilon > 0$) and $\mathcal{O}(\log n)$ query time, they thus obtain a solution with the same storage and query time $\mathcal{O}(\log^2 n)$ for ray shooting queries. Unfortunately, this logarithmic overhead between the basis algorithm (the structure by Clarkson) and the resulting solution to the ray shooting problem seems to be intrinsic to the approach using parametric search. In order to avoid that bottleneck and to improve the query time to optimal $\mathcal{O}(\log n)$, we discuss a variation of Clarkson's structure to solve ray shooting queries directly without recurring to parametric search. We thus obtain a solution with $\mathcal{O}(n^{\lfloor d/2 \rfloor} \operatorname{polylog} n)$ storage and $\mathcal{O}(\log n)$ query time. Since the complexity of the intersection of n halfspaces in d dimensions can be $\Theta(n^{\lfloor d/2 \rfloor})$, it is generally believed that this is close to optimal. A popular conjecture suspects a lower bound of $\Omega(n)$ on the product $Q(n) \cdot S(n)^{1/\lfloor d/2 \rfloor}$, where $Q(n)$ and $S(n)$ denote query time and storage of a data structure for the problem. (There is, however, nothing even approaching a proof of this conjecture, but compare with some recent results by Brönnimann and Chazelle [BC92] who prove an $\Omega((n/\log n)^{1-\frac{d-1}{d(d+1)}}/M^{1/d})$ lower bound for the query time of a halfspace range reporting data struc-

ture with storage M).

It seems therefore interesting whether we can further improve the storage of the data structure. We show that the idea of bootstrapping by applying the previous structure to a large random sample of H used by Chazelle and Friedman in [CF92] (for point location in hyperplane arrangements) can in fact be used to improve the storage of our solution for ray shooting queries to $\mathcal{O}(n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor - \varepsilon})$. This obtains a $Q(n)S(n)^{1/\lfloor d/2 \rfloor}$ product of $\mathcal{O}(n(\log n)^{\varepsilon})$ for an arbitrarily small $\varepsilon > 0$, which improves the previous bounds using parametric search on Clarkson's ($\mathcal{O}(n^{1+\varepsilon})$) and Mulmuley's solution ([Mul91b] proves a $\mathcal{O}(n(\log n)^{O(1)})$ solution).

Note that by a well known lifting transformation (see e.g. [Ede87]), nearest (and furthest) neighbor queries can be solved by shooting a vertical ray in the upper envelope of a set of n hyperplanes (which is a polytope) in a space of one dimension more. We thus obtain a structure for the post office problem, which is an improvement to previous results in itself. Even in three-dimensional space, for instance, the best previous solution with space $\mathcal{O}(n^2)$ had query time $\mathcal{O}(\log^2 n)$, see [Cha85, AESW90]. We improve this query time to $\mathcal{O}(\log n)$.

We then turn our attention to a dynamic version of ray shooting queries. Note that we do *not* intend to make the structure discussed before dynamic (a recent result by Agarwal and Matoušek [AM91] shows that a version of Clarkson's structure with $\mathcal{O}(n^{\lfloor d/2 \rfloor + \varepsilon})$ storage can—with considerable effort—be made dynamic, and in fact, the ideas mentioned above to solve ray shooting queries directly without requiring parametric search can be applied to their dynamic solution as well). Instead, we consider the problem of maintaining the intersection of halfspaces in dimension d under a model of *random* insertions and deletions (in a nutshell, it is assumed that at every insertion step, every halfspace currently not present is equally likely to be inserted, and at a deletion step, every halfspace currently present has the same probability of being deleted). Several authors [Mul91c, CMS92, Sch91] studied this problem before and presented (three different) ways of maintaining the intersection polytope efficiently under such a random model. For randomized incremental algorithms, it is often the case that the history of the dynamic maintenance process can be used as a point location structure for the current structure, see e.g. [BDS+90, GKS92, BDT90, DMT92, Mul88, Sei91b, CMS92], [Mul91c] and [Sch91] also considered deletions. For the case of maintaining halfspace intersections, however, such a data structure has remained elusive for dimension $d \geq 5$. ([Mul91b] gives a structure for the problem of deciding whether a point q lies in the polytope, assuming a random sequence of insertions and deletions, but it does not use the history as a search structure). As Mulmuley [Mul91c] observed, the problem is the retriangulation of the regions adjacent to a newly created facet. We show in Section 4 that Chazelle and Friedman's notion of an *antenna* can be used to cope with this problem. We consider the semi-dynamic case, where halfspaces are only allowed to be inserted. We obtain a data structure for d-dimensional ray shooting queries capable of insertions of halfspaces in a model of random insertions. It is possible to obtain similar results for deletions of halfspaces. For reasons of space and motivation, however, we do not pursue that any further.

2 Definitions

Given a set H of n hyperplanes in d-dimensional space $\mathbf{E}^d$, we denote by $\mathcal{A}(H)$ the arrangement of the hyperplanes, and by $\mathcal{P}(H)$ the closure of the cell of $\mathcal{A}(H)$ containing a certain fixed point o (the origin of the coordinate system, say). For $h \in H$, we define h^+ as the open halfspace bounded by h and containing o and h^- as the open halfspace not containing o. $\mathcal{P}(H)$ is a d-dimensional polytope, defined as the closure of the intersection of the n halfspaces $\{h^+ \mid h \in H\}$. Its complexity is thus at most $\mathcal{O}(n^{\lfloor d/2 \rfloor})$ by the upper bound theorem, see e.g. [Ede87]. We will assume throughout this paper that the given hyperplanes are in general position, i.e. no $d+1$ hyperplanes intersect in a common point. Let us say that a hyperplane h is *hit* by a ray ρ if the intersection of ρ and h is a point, i.e. ρ intersects h but is not contained in h. We can thus define $\psi(\rho, H)$ as the *first* hyperplane $h \in H$ hit by ρ. We resolve ambiguities arising if ρ passes through a lower dimensional face of $\mathcal{A}(H)$ by imposing some arbitrary order on H, such as the lexicographic order with respect to some unique representation of the hyperplanes.

Definition 1 *Given a set H of n hyperplanes and its polytope $\mathcal{P}(H)$. A* ray shooting query *is defined as follows: Given a ray ρ with origin p such that $p \in \mathcal{P}(H)$, find the first hyperplane $h \in H$ hit by ρ.*

Note that we do not consider hyperplanes *containing* ρ. This is essential since we will have to use rays ρ which are contained in hyperplanes bounding $\mathcal{P}(H)$. Another, more formal definition we could have used is: *Given a ray ρ with origin p such that $p \in \mathcal{P}(H)$, find the point $q \in \mathcal{P}(H)$ with $\rho \cap \mathcal{P}(H) = pq$, and a hyperplane containing q but not p.*

We furthermore need the following concept, which is a modified version of a notion due to Chazelle and Friedman [CF92]. Again we are given a set H of hyperplanes and its arrangement $\mathcal{A}(H)$. Given a subset

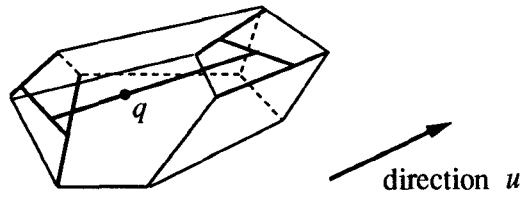

Figure 1: Example for an antenna in three dimensions

$K \subset H$ with $|K| < d$ and a direction vector u, we define another direction vector $w_K(u)$ as follows: We let f denote the $(d - |K|)$-dimensional flat $f := \bigcap_{h \in K} h$, and take the first vector w in the list

$$(u, (1,0,0,\ldots,0), (0,1,0,\ldots,0), \ldots, (0,0,0,\ldots,1)),$$

that is not normal to f (w exists since $|K| < d$) and define $w_K(u)$ as the direction vector lying in f which maximizes $w \cdot w_K(u)$. Imagine walking around in f. The direction $w_K(u)$ is the direction in which we proceed as much in direction w as possible in the flat f at all.

Definition 2 *Given a set H of hyperplanes in general position, a point q, a subset $K \subset H$ with $q \in \bigcap_{h \in K} h$, and a direction vector u, the* antenna $ant_u(q, K)$ *of q with respect to K and u is defined as a binary tree of depth $d - |K|$ whose edges are labeled with elements of H and whose nodes correspond to points in $\mathbb{E}^d$. If $|K| = d$, $ant_u(q, K)$ consists of a single leaf corresponding to the vertex $\bigcap_{h \in K} h$; otherwise we consider the rays ρ' and ρ'' with origin q and directions $w_K(u)$ and $-w_K(u)$ resp. Let $h' = \psi(\rho', H)$ and $h'' := \psi(\rho'', H)$ be the first hyperplanes hit by ρ' and ρ'' resp., and let $q' := \rho' \cap h'$ and $q'' := \rho'' \cap h''$ be the points of intersection. Then $ant_u(q, K)$ is defined as the binary tree consisting of a root (corresponding to the point q), two outgoing edges labeled with h' and h'', and the two subtrees $ant_u(q', K \cup \{h'\})$ and $ant_u(q'', K \cup \{h''\})$.*

The *edges* of $ant(\rho, H)$ are the line segments joining the points corresponding to two adjacent nodes (i.e. parent and child) in the tree $ant_u(q, K)$. In Figure 1 an example of an antenna in three dimensions is given, showing the edges of $ant_u(q, \emptyset)$. The figure should explain the name of the construct.

Observation 3 *Given the situation of Definition 2. Then $ant_u(q, K)$ has 2^k leaves corresponding to vertices of $\mathcal{A}(H)$ that are not necessarily distinct, where $k := d - |K|$; and q lies in the convex hull of these vertices. The vertex corresponding to a leaf v of the tree $ant_u(q, K)$ can be found as the intersection of the hyperplanes in K with those labeling the path from the root of $ant_u(q, K)$ to v; v is adjacent to the face of $\mathcal{A}(H)$ containing q. Furthermore, the antenna $ant_u(q, K)$ can be computed by at most 2^{k+1} ray shooting queries in $\mathcal{A}(H)$.*

Finally, we define the antenna of a ray ρ with respect to a set H of hyperplanes as $ant(\rho, H) := ant_u(p, \emptyset)$, where p is the origin of ρ and where u is the direction of ρ.

3 The static ray shooting problem

In this section we assume that a set H of n hyperplanes and its polytope $\mathcal{P}(H)$ are given in advance, and we wish to preprocess them into a data structure which allows for fast ray shooting queries. Agarwal and Matoušek [AM92] have given such a structure with preprocessing time and storage $\mathcal{O}(n^{\lfloor d/2 \rfloor + \varepsilon})$ (for an arbitrarily small, but fixed $\varepsilon > 0$) and query time $\mathcal{O}(\log^2 n)$ by applying parametric search to a data structure by Clarkson. We first improve upon this solution by giving a data structure with query time $\mathcal{O}(\log n)$ and storage and preprocessing time $\mathcal{O}(n^{\lfloor d/2 \rfloor}(\log n)^{O(1)})$. We then further improve the storage requirement of this structure slightly by doing as Chazelle and Friedman [CF92] and bootstrapping the first solution. While [CF92] needs four such bootstrapping steps for the final solution, we employ somewhat more powerful tools to do so in a single step. This approach should work for the problem considered in [CF92] as well.

We first need several definitions, partly borrowed from [Mat91b]. For a simplex Δ, let H_Δ denote the set of hyperplanes $h \in H$ intersecting Δ. Note that our simplices can be unbounded. In fact, we define a simplex as the intersection of $d + 1$ closed halfspaces.

A $(1/r)$-cutting for $\mathcal{P}(H)$ is a collection Ξ of simplices with disjoint interiors, such that $|H_\Delta| \leq n/r$ for every $\Delta \in \Xi$, and such that $\mathcal{P}(H)$ is covered by the union of the simplices and that every simplex Δ contains a point of $\mathcal{P}(H)$. The *size* of a cutting is the number of its simplices. We then have the following result:

Lemma 4 (Restricted version of the Shallow cutting lemma, [Mat91b]) *Let H be a collection of n hyperplanes in general position in $\mathbb{E}^d$, and $r < n$ a parameter. There exists a $(1/r)$-cutting Ξ for $\mathcal{P}(H)$ of size $\mathcal{O}(r^{\lfloor d/2 \rfloor})$.*

We then have the following observation[1]:

Lemma 5 *Given a set H of hyperplanes in $\mathbb{E}^d$ and a set Ξ of simplices such that $\mathcal{P}(H)$ is covered by the union of the simplices and that every $\Delta \in \Xi$ intersects $\mathcal{P}(H)$. Let ρ be a ray with origin p in $\mathcal{P}(H)$ and let Δ*

[1] The author is indebted to Jirka Matoušek for pointing out this observation which somewhat simplifies the algorithm

be the last *simplex in Ξ intersected by ρ (with respect to the natural order of simplices along the ray ρ). Then $h := \psi(\rho, H)$ intersects Δ.*

Proof: There is a simplex Δ intersected by ρ since ρ starts in $\mathcal{P}(H)$ and $\mathcal{P}(H)$ is covered by Ξ. Consequently, there also exists a last simplex Δ intersected by ρ. Let $h := \psi(\rho, H)$ be the first hyperplane in H hit by ρ and let $q := \rho \cap h$ be the point of intersection. If $q \in \Delta$, h clearly intersects Δ. Otherwise Δ contains a point x on ρ but outside $\mathcal{P}(H)$. Since $q \in h$ and $p \in h^+$ we must have $x \in h^-$. Since Δ intersects $\mathcal{P}(H)$, there is a point $y \in \Delta$ that lies in the polytope $\mathcal{P}(H)$ and thus in h^+. This implies that xy intersects h, so h intersects Δ. □

We will need the following data structure as a secondary structure.

Lemma 6 *Given a set Ξ of m simplices with disjoint interior in $\mathbf{E}^d$. We can build in time $\mathcal{O}(m^{8d+2})$ a data structure of size $\mathcal{O}(m^{8d-6+\varepsilon})$ (for an arbitrarily small constant $\varepsilon > 0$) that permits queries of the form "Given a query ray ρ, find the last simplex Δ in Ξ intersected by ρ" with query time $\mathcal{O}(\log m)$.*

Proof: We apply a standard locus approach. We consider the $(2d-1)$-dimensional space of all rays. The loci corresponding to rays with the same last simplex induce a subdivision of this space. We can do point location in this subdivision using the following result by Chazelle et al. [CEGS89].

Lemma 7 ([CEGS89]) *Given N polynomials f_i, $1 \leq i \leq N$, of bounded degree in D variables (where D is also considered a constant). For a D-dimensional variable $x := (x_1, x_2, \ldots, x_D)$, let $sign_i(x)$ be the sign of $f_i(x)$, and $sign(x) := (sign_1(x), \ldots, sign_N(x))$ be the sign vector of x with respect to the family (f_i). Assume we are given a label for every possible value of $sign(x)$.*

There is a data structure with storage $\mathcal{O}(N^{2D-3+\varepsilon})$, for an arbitrarily small constant $\varepsilon > 0$, that can be computed in time $\mathcal{O}(N^{2D+1})$ and that permits, given a D-dimensional variable x, to find the label associated with $sign(x)$ in time $\mathcal{O}(\log N)$.

We represent the query ray ρ by its origin p and by a direction vector u. We thus have a $D := 2d$ dimensional[2] representation for ρ. Consider now a fixed simplex Δ. We compute a constant number of polynomials in the $D = 2d$ variables representing ρ to test whether ρ intersects Δ. We first observe that ρ intersects Δ if either the origin p of ρ lies in Δ, or if some facet f of Δ is visible from p and the direction u lies in the convex cone with apex p spanned by f. The first case can easily be tested by $d+1$ linear inequalities (polynomials of degree one). For second case we first identify the facets f visible from p using the same inequalities as in the first case.

We now have a facet f of Δ visible from p and wish to test whether u lies in the cone spanned by f. This can be done by checking the orientation of u with respect to the d hyperplanes that pass through p and the d ridges ($(d-2)$-faces) of f. For every ridge, we choose $d-1$ affinely independent points $q_1, q_2, \ldots, q_{d-1}$. The orientation of u with respect to the hyperplane spanned by p and $q_1, q_2, \ldots, q_{d-1}$ can be tested by computing the orientation of the simplex spanned by u and the $d-1$ vectors $q_1 - p, q_2 - p, \ldots, q_{d-1} - p$. This orientation can be found by computing the determinant of the matrix whose rows are these d vectors, which corresponds to the evaluation of a polynomial of degree d.

In this fashion we have determined a set of $\mathcal{O}(m)$ polynomials. The signs of these polynomials for the variable representing a given ray ρ give us complete information as to which simplices are intersected by ρ, and, if a simplex Δ is intersected, through which facet f of Δ the ray ρ enters the simplex Δ. It remains to determine the *last* simplex intersected by ρ.

To this end, we add another set of $\mathcal{O}(m^2)$ polynomials. Assume that ρ intersects two simplices Δ_1 and Δ_2, and enters them through their facets f_1 and f_2, resp. Let h_1 and h_2 be the hyperplanes containing f_1 and f_2. We can then test which of the two simplices is intersected first by testing which of the two intersections of ρ with h_1 and h_2 lies closer to p. But this can again be tested by determining the orientation of u with respect to the hyperplane through p and the $(d-2)$-flat $h_1 \cap h_2$. This can be dealt with as above. In this fashion we can add a polynomial for every pair of facets of different simplices.

The sign sequence of the resulting set of polynomials determines uniquely which simplex is the last one intersected by ρ. We can therefore appeal to Lemma 7, with parameters $N \in \mathcal{O}(m^2)$ and $D = 2d$, to obtain a search structure with the time bounds claimed above. Given a ray ρ, this structure returns the labels of the last simplex Δ intersected by ρ. □

We can now start to describe our data structure. We are given a set H of n halfspaces in general position, and the special point o defining $\mathcal{P}(H)$. If n is smaller than some (large) constant, we simply store the list of hyperplanes H. Otherwise, we take a cutting Ξ according to Lemma 4 with parameter $r := n^\alpha$, where $\alpha > 0$ is a (small) constant (depending on d) to be determined later. Furthermore, we build a secondary structure according to Lemma 6 for the set Ξ. By choosing α

[2] This is admittedly not optimal.

small enough, the storage of this secondary structure is bounded by $\mathcal{O}(n^{\lfloor d/2 \rfloor})$. We then identify for every simplex $\Delta \in \Xi$ the set H_Δ of hyperplanes intersecting Δ, and recursively construct the same structure for H_Δ.

The storage requirement $S(n)$ follows the recurrence: $S(n) \in \mathcal{O}(1)$ for $n \in \mathcal{O}(1)$, and

$$S(n) = c_1 r^{\lfloor d/2 \rfloor} S(n/r) + \mathcal{O}(n^{\lfloor d/2 \rfloor})$$

for some constant $c_1 > 0$ and $r = n^\alpha$, which solves to $S(n) \in \mathcal{O}(n^{\lfloor d/2 \rfloor} \log^c n)$ for some constant $c > 0$.

To answer a ray shooting query with query ray ρ, we first use the secondary data structure for the collection Ξ to find the last simplex $\Delta \in \Xi$ intersected by the query ray ρ. We then continue in the structure associated with Δ, until we reach a point where we explicitly stored all hyperplanes, which we can simply test sequentially.

The correctness of this query algorithm relies on two facts: There always is a last simplex Δ intersected by ρ, and the hyperplane $h = \psi(\rho, H)$ which is the solution to the ray shooting query intersects this simplex Δ. These claims follow directly from Lemma 5.

The query time follows a recursion of the form $Q(n) = Q(n^{1-\alpha}) + \mathcal{O}(\log n)$, which gives $Q(n) = \mathcal{O}(\log n)$.

Let us now turn to the preprocessing time. The main problem is the construction of a shallow cutting according to Lemma 4. Inspection of the proof of Lemma 4 in [Mat91b] shows that it can be computed by a rather simple randomized algorithm in time $\mathcal{O}(nr^{\lfloor d/2 \rfloor})$, but Matoušek [Mat92] actually shows that it can even be computed by a deterministic algorithm within time $\mathcal{O}(n \log r)$ for our choice of r. Identification of the sets H_Δ can trivially be done in time $\mathcal{O}(nr^{\lfloor d/2 \rfloor})$ by testing every hyperplane with every simplex of the cutting.

The time to construct the secondary structure on Ξ according to Lemma 6 is polynomial in $r = n^\alpha$, and for sufficiently small $\alpha > 0$ all this is bounded by $\mathcal{O}(n^{\lfloor d/2 \rfloor})$ again. Thus we get the same recursion as for the space requirement and find that the preprocessing time is also bounded by $\mathcal{O}(n^{\lfloor d/2 \rfloor} \log^c n)$.

Theorem 8 *Given a set H of n hyperplanes in general position of $\mathbb{E}^d$, $d \geq 4$. It can be preprocessed with preprocessing time and storage $\mathcal{O}(n^{\lfloor d/2 \rfloor}(\log n)^{O(1)})$ such that ray shooting queries for $\mathcal{P}(H)$ can be answered with query time $\mathcal{O}(\log n)$.*

In [AM91], Agarwal and Matoušek give a dynamic version of the classical structure for containment in a polytope by Clarkson [Cla88]. They show that a set of n halfspaces H can be stored in $\mathcal{O}(n^{\lfloor d/2 \rfloor + \varepsilon})$ space, for an arbitrarily small constant $\varepsilon > 0$, such that queries of the form "Does a given point q lie in the intersection $\mathcal{P}(H)$ of the halfspaces?" can be answered in time $\mathcal{O}(\log n)$, and such that halfspaces can be added to or removed from the set in amortized time $\mathcal{O}(n^{\lfloor d/2 \rfloor - 1 + \varepsilon})$. The trick presented in this section allows us to convert their structure into one which permits ray shooting queries on $\mathcal{P}(H)$ with the same time bounds.

We will now show that the idea of bootstrapping used by Chazelle and Friedman in [CF92] can be used to improve the storage of our solution for ray shooting queries to $\mathcal{O}(n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor - \varepsilon})$. While [CF92] needs four bootstrapping steps, we employ somewhat more powerful tools do obtain our result with a single such step. This idea could be used for the point location in hyperplane arrangements as well, and, in fact, the time bound given in [CF92] can be slightly improved by following exactly the same ideas as presented here.

We need a result by [AM92] which gives a solution for the ray shooting problem with a tradeoff between space and query time, and a lemma on random sampling following easily from results by Clarkson [CS89, Cla87].

Lemma 9 (Agarwal and Matoušek [AM92]) *Given a set H of n hyperplanes $\mathbb{E}^d$ ($d \geq 4$), and a parameter M ($n \leq M \leq n^{\lfloor d/2 \rfloor}$), there is a data structure for ray shooting queries with space and preprocessing time $\mathcal{O}(M^{1+\varepsilon})$ and query time $\mathcal{O}(n/M^{1/\lfloor d/2 \rfloor} \log^3 n)$, where $\varepsilon > 0$ is an arbitrarily small constant.*

Lemma 10 *Given a set H of n hyperplanes and a parameter $\beta > 0$. When taking a random sample R of r hyperplanes from H, with probability at least $1/2$ the following is true: For all vertices v of $\mathcal{P}(R)$, the number n_v of hyperplanes $h \in H$ with $v \in h^-$ is bounded by $\mathcal{O}(n \log r/r)$, and the sum $\sum_{v \in \mathcal{P}(R)} n_v^\beta$ is bounded by $\mathcal{O}(r^{\lfloor d/2 \rfloor}(n/r)^\beta)$.*

Proof: The expected value of the sum $\sum_{v \in \mathcal{P}(R)} n_v^\beta$ is in $\mathcal{O}(r^{\lfloor d/2 \rfloor}(n/r)^\beta)$ by Theorem 3.6 in [CS89]. The probability that the sum exceeds four times its mean is bounded by $1/4$. On the other hand, Corollary 4.2 in [Cla87] proves that the probability that any n_v exceeds $cn \log r/r$ is at most $1/4$, for a suitable constant c. Together, these results prove the claim. □

The improved structure can now be derived as follows: We take a random sample R of H of size $r = n/\log^k n$ for some parameter k (which is determined later), and build a structure according to Theorem 8 for $\mathcal{P}(R)$. Furthermore, for every vertex v of $\mathcal{P}(R)$ we identify the set H_v of hyperplanes $h \in H$ with $v \in h^-$. Let $n_v = |H_v|$. We then construct for every vertex $v \in \mathcal{P}(R)$ a secondary structure according to Lemma 9 with storage $M = \mathcal{O}(n_v^{\beta+\varepsilon})$ and query time $\mathcal{O}(n_v^{1-\beta/\lfloor d/2 \rfloor} \log^3 n_v)$ ($1 \leq \beta \leq \lfloor d/2 \rfloor$ is yet another constant to be chosen later). By Lemma 10 we can choose R such that the

total storage for the secondary structures is

$$\sum_{v \in \mathcal{P}(R)} n_v^{\beta+\varepsilon} = \mathcal{O}(r^{\lfloor d/2 \rfloor}(n/r)^{\beta+\varepsilon}) =$$
$$= \mathcal{O}(n^{\lfloor d/2 \rfloor}(\log n)^{k\beta+k\varepsilon-k\lfloor d/2 \rfloor})$$

and such that $n_v \in \mathcal{O}(\log^{k+1} n)$ for all vertices $v \in \mathcal{P}(R)$.

To perform a query with a ray ρ with origin p in $\mathcal{P}(H)$, we first use the structure of Theorem 8 to find the antenna of ρ with respect to R. By Observation 3, this can be done by repeated ray shooting queries on $\mathcal{P}(R)$. Finally, we query the secondary structures associated with the at most 2^d vertices appearing as leaves in $ant(\rho, R)$ with the ray ρ, and output the first hyperplane hit by all the subproblems.

It remains to show that the first hyperplane $h = \psi(\rho, H)$ in H hit by ρ appears in H_v for some vertex v of $ant(\rho, R)$. Let $h' = \psi(\rho, H')$ denote the first hyperplane hit by ρ in $\mathcal{P}(R)$. If the solution hyperplane h is not identical to h', it must separate the origin p of ρ and the point q where ρ intersects h'. Since q lies in the convex hull of the $v \in ant(\rho, R)$, h must separate p from at least one of the vertices v. And since h cannot separate p from o (p being in $\mathcal{P}(H)$), h must separate this vertex v from o as well.

Since $n_v \in \mathcal{O}(\log^{k+1} n)$, the query time of our combined structure is bounded by

$$\mathcal{O}(\log n + (\log n)^{(k+1)(1-\beta/\lfloor d/2 \rfloor)}(\log\log n)^3).$$

We choose β such that $(k+1)(\lfloor d/2 \rfloor - \beta) = \lfloor d/2 \rfloor - \delta$ for some small constant $\delta > 0$. This implies that the query time is bounded by $\mathcal{O}(\log n)$, and that the storage is bounded by

$$\mathcal{O}\big(r^{\lfloor d/2 \rfloor}\log^c r + n^{\lfloor d/2 \rfloor}(\log n)^{k\beta+k\varepsilon-k\lfloor d/2 \rfloor}\big)$$
$$= \mathcal{O}\big(n^{\lfloor d/2 \rfloor}(\log n)^{c-k\lfloor d/2 \rfloor} + n^{\lfloor d/2 \rfloor}/(\log n)^{(k+1)(\lfloor d/2 \rfloor-\beta)-k\varepsilon-(\lfloor d/2 \rfloor-\beta)}\big)$$
$$= \mathcal{O}\big(n^{\lfloor d/2 \rfloor}(\log n)^{c-k\lfloor d/2 \rfloor} + n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor-\delta-k\varepsilon-\lfloor d/2 \rfloor/(k+1)}\big)$$

Choosing k sufficiently large, and δ and ε sufficiently small, we can bound this by $\mathcal{O}(n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor-\varepsilon'})$ for an arbitrarily small $\varepsilon' > 0$.

Let us turn to the cost of preprocessing. The problem is to find a sample R which conforms to the requirements of Lemma 10. Those can be checked once we know the number of halfspaces in the lists H_v for every vertex $v \in \mathcal{P}(R)$ (these halfspaces have to be computed anyway). We just take a random sample R of the appropriate size, and construct its polytope $\mathcal{P}(R)$. Then we sequentially take all n halfspaces h from H and find all $v \in \mathcal{P}(R)$ which are in h^-. It is well known that this can be done in time linear in the number of such vertices once one such vertex is known (see e.g. [CS89]). On the other hand, we get such a vertex (a *conflict* of h) by linear programming on R in time $\mathcal{O}(r)$ (see [Sei91a]), so the total time for this step is $\mathcal{O}(nr)$ plus the total number of vertices found. We stop this algorithm if it tries to report more than $\mathcal{O}(nr^{\lfloor d/2 \rfloor-1})$ such vertices (because then the sample R is bad), otherwise we check whether it fulfills the requirements of Lemma 10. If not, we discard R and take a new random sample. By Lemma 10, the expected number of such trials is constant, so we can find R in time $\mathcal{O}(nr^{\lfloor d/2 \rfloor-1})$ (and identify the lists H_v in the same time). The secondary structures can be computed in time linear in their storage (see [AM92]), so we have again the same recursion as for the space. We have thus proven

Theorem 11 *Given a set H of n hyperplanes in $\mathbb{E}^d$, $d \geq 4$, ray shooting queries on $\mathcal{P}(H)$ can be done with storage $S(n) = \mathcal{O}(n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor-\varepsilon})$ and query time $Q(n) = \mathcal{O}(\log n)$, where $\varepsilon > 0$ is an arbitrarily small constant.*

The post office problem in $\mathbb{E}^d$ (which calls for performing nearest-neighbor queries on a set of points) can be transformed into the problem of shooting a vertical ray on the upper envelope of a set of n hyperplanes in $\mathbb{E}^{d+1}$ by a well-known lifting map, see e.g. [Ede87]. We thus obtain the following corollary.

Corollary 12 *Given an n-point set P in $\mathbb{E}^{d-1}$, nearest-neighbor queries on P can be solved with storage $S(n) = \mathcal{O}(n^{\lfloor d/2 \rfloor}/(\log n)^{\lfloor d/2 \rfloor-\varepsilon})$ and query time $Q(n) = \mathcal{O}(\log n)$, where $\varepsilon > 0$ is an arbitrarily small constant.*

This corollary is an improvement in itself. Even in three dimensions, the best previous solution to the post office problem with space $\mathcal{O}(n^2)$ needed query time $(\log^2)$, see e.g. Chazelle [Cha85] and Agarwal et al. [AESW90].

Remark: From a practical point of view, it was not a good idea to use the antenna concept in the algorithm of Theorem 11, because it introduces an exponential dependence on d, which can be avoided by replacing the antenna of ρ by a suitable simplex of the bottom-vertex triangulation of $\mathcal{P}(R)$. Since we will really need the antenna concept in the next section, it seemed permissible from a theoretical point of view to present the result this way, making some additional definitions unnecessary. And then, Theorem 11 is of theoretical interest only anyway.

4 Semi-dynamic ray shooting

As mentioned before, we do not aim at making the structures presented in the previous section dynamic. (But compare with the recent structure by Agarwal and Matoušek [AM91]).

We prefer to concentrate on the question whether the different randomized incremental algorithms to compute the intersection of halfspaces in d-space (using a sequence of random insertions) can be augmented to permit ray shooting queries using the history of the insertion process as a data structure.

Let us thus assume that we are given a set H of hyperplanes, a random permutation $\{h_1, h_2, \ldots, h_n\}$ of H, and we wish to maintain $\mathcal{P}(\{h_1, \ldots, h_r\})$ while adding the hyperplanes in that random order. For brevity, we put $H_r := \{h_1, \ldots, h_r\}$.

Assume we have $\mathcal{P}(H_{r-1})$ and we wish to add h_r. A vertex v of $\mathcal{P}(H_{r-1})$ which is deleted by the addition of h_r can be found by solving a linear program in time $\mathcal{O}(r)$ (an especially simple algorithm for doing this has been given by Seidel [Sei91a]). Then a graph search can be used to identify the portion of $\mathcal{P}(H_{r-1})$ that gets removed by h_r (because it lies in h_r^-) and it can be replaced by the new facet (in h_r) in time linear in the structural change (the number of vertices that are removed or created when going from $\mathcal{P}(H_{r-1})$ to $\mathcal{P}(H_r)$), see Clarkson and Shor [CS89]. Since the expected structural change in any stage is bounded by $\mathcal{O}(r^{\lfloor d/2 \rfloor - 1})$ ([CS89]), we can update the intersection polytope $\mathcal{P}(H_r)$ in expected time $\mathcal{O}(n_r^{\lfloor d/2 \rfloor - 1})$ per step (still assuming $d \geq 4$).

As mentioned before, the problems arise when we wish to maintain a point location structure at the same time. We want to use the history of the maintenance process as a point location data structure, an idea that by now is nearly part of the folklore in the field, see e.g. [BDS+90, GKS92, BDT90, DMT92, Mul88, Sei91b, Mul91c, CMS92]. The problem with this approach is the choice of a suitable triangulation of the polytope: When the "region"[3] containing the query point q in $\mathcal{P}(H_{r-1})$ is destroyed by the insertion of h_r, we have to find the appropriate region in $\mathcal{P}(H_r)$. It is possible to do this when the region "belongs" to the facet of the newly inserted hyperplane h_r, but this seems rather difficult and gives rise to nasty complications when the new region belongs to one of the facets adjacent to h_r (a facet which has been truncated by the insertion of h_r and which had therefore been repartitioned into regions). The problem is that then we have to build a point location structure not for the whole facet, but only for the portion of the "triangulation" that has been changed between $\mathcal{P}(H_{r-1})$ and $\mathcal{P}(H_r)$. This can be done for up to four dimensions, but for higher dimensions a solution to this problem has been elusive. In this section we first describe how to use Chazelle and Friedman's ([CF92]) concept of an *antenna* defined in Definition 2 to circumvent this problem. Using the antenna of a query point, we can step from $\mathcal{P}(H_{r-1})$ to $\mathcal{P}(H_r)$ using only location operations in the newly created facet (that of h_r).

What is our data structure actually? When we have to remove some vertices of $\mathcal{P}(H_{r-1})$, we do not really delete them, but just mark them as "old" and attach a pointer to them, indicating the hyperplane h_r whose insertion made the vertex disappear. Furthermore, at every insertion, we build a static data structure for ray shooting queries on the $(d-1)$-dimensional polytope $\mathcal{P}(H_r) \cap h_r$ (the facet supported by h_r). (We will discuss later how to find such a structure).

To perform a ray shooting query with a query ray ρ in $\mathcal{P}(H_m)$, we trace this history data structure while maintaining $ant(\rho, H_r)$. (If we can find this $ant(\rho, H_m)$ for the current H_m, that solves the ray shooting query). To do so, we find $ant(\rho, H_1)$ in $\mathcal{P}(H_1)$, and observe that $ant(\rho, H_r)$ is different from $ant(\rho, H_{r-1})$ exactly if at least one of the vertices in $ant(\rho, H_{r-1})$ is removed by the insertion of h_r (by the same argument as in the proof of Theorem 11). Since a vertex which is removed from $\mathcal{P}(H_{r-1})$ has a pointer to h_r, we can thus jump from one stage changing $ant(\rho, H_r)$ to the next, skipping all stages that leave $ant(\rho, H_r)$ unchanged. By backwards analysis, it is easy to see that the expected number of stages changing $ant(\rho, H_r)$ is $\mathcal{O}(\log r)$ (in fact, it is $\mathcal{O}(\log r)$ even with high probability, see e.g. [Sch91]).

It remains to show how to update $ant(\rho, H_r)$ if it actually changes between $\mathcal{P}(H_{r-1})$ and $\mathcal{P}(H_r)$. Recalling Definition 2, we can find $ant(\rho, H_r)$ by shooting from q in two directions until we hit the first hyperplanes, then shooting from the two thus obtained points in four directions etc. The important observation is that, when shooting from some point q', the first hyperplane hit is either the same as in $\mathcal{P}(H_{r-1})$ (which we know since it labels the appropriate edge in $ant(\rho, H_{r-1})$ or the newly inserted hyperplane h_r. We thus only have to check whether the hyperplane which was hit in H_{r-1} is now covered by h_r. Since this can be done in constant time, we can in constant time traverse the antenna tree $ant(\rho, H_{r-1})$ and determine a set of nodes where we have hit the new hyperplane h_r. At every such node we have to replace the subtree by some lower dimensional antenna which lies completely in h_r and can thus be obtained by the structure for ray shooting queries on $\mathcal{P}(H_r) \cap h_r$. This implies the following theorem.

[3] We have to be a bit vague in this paragraph. Readers without experience in history-oriented randomized incremental algorithms are advised to skip to the next paragraph.

Theorem 13 *Let $T(r)$ be the expected time (and space) to build a structure for ray shooting queries on $\mathcal{P}(H_r) \cap h_r$ when adding a hyperplane h_r to $\mathcal{P}(H_{r-1})$, and let $Q(r)$ be its query time.*

Then $\mathcal{P}(H_r)$ can be maintained under random insertions of hyperplanes with $\mathcal{O}(m^{\lfloor d/2 \rfloor} + mT(m))$ expected storage, $\mathcal{O}(m^{\lfloor d/2 \rfloor - 1} + T(m))$ expected update time, and $\mathcal{O}(\log m \cdot Q(m))$ query time (with high probability) for ray shooting queries on $\mathcal{P}(H_r)$, where m is the current number of hyperplanes present.

It remains to find a structure for ray shooting queries on $\mathcal{P}(H_r) \cap h_r$. If the dimension d is even, this is easy: We just take the $(d-1)$-dimensional version of Theorem 11. Since for d even we have $\lfloor (d-1)/2 \rfloor = \lfloor d/2 \rfloor - 1$, we thus have

Corollary 14 *For even dimension $d \geq 4$, we can maintain the intersection of halfspaces under random insertions of halfspaces with $\mathcal{O}(m^{\lfloor d/2 \rfloor})$ expected storage, $\mathcal{O}(m^{\lfloor d/2 \rfloor - 1})$ expected update time, and $\mathcal{O}(\log^2 m)$ query time (with high probability), where m is the current number of halfspaces.*

The problem becomes considerably more involved if the dimension is odd, because then the complexity of the facet $\mathcal{P}(H_r) \cap h_r$ can be as bad as $\Theta(n^{\lfloor d/2 \rfloor})$. However, we can observe that if we fix H_r, h_r is still a *random* facet of $\mathcal{P}(H_r)$, and its *expected* complexity is only $\mathcal{O}(n_i^{\lfloor d/2 \rfloor - 1})$. This will help us to find a better solution for this case. We prove the following theorem.

Theorem 15 *Given a set H of n hyperplanes in $\mathbb{E}^d$. A $(d-1)$-dimensional data structure for ray shooting queries on a random facet of $\mathcal{P}(H)$ can be build with expected storage $\mathcal{O}(n^{\lfloor d/2 \rfloor - 1}(\log n)^{O(1)})$ (where the expectancy is with respect to the random choice of the facet) and query time $\mathcal{O}(\log n)$.*

In fact, we will use the structure of Theorem 8, with a slight modification. The trick is to think of the problem as if it was actually a problem in d dimensions, i.e. we build a structure for ray shooting queries on $\mathcal{P}(H)$—*but with the restriction that the query ray must lie in the hyperplane h!*

We thus take a cutting Ξ for $\mathcal{P}(H)$ according to Lemma 4, with parameter $r = n^\alpha$ (for some sufficiently small $\alpha > 0$ to be determined later), and construct a secondary structure for the simplices of Ξ as in Theorem 8. For every $\Delta \in \Xi$ we identify the set H_Δ of halfspaces whose bounding hyperplanes intersect Δ. Furthermore, we identify all simplices of Ξ intersected by h (the hyperplane determining the $(d-1)$-dimensional space we intend to work in) and denote this collection by Ξ_h. We then recursively construct for every $\Delta \in \Xi_h$ the same structure for H_Δ.

Since a query ray must lie in h, the last simplex intersected by the ray ρ is also intersected by h. We have thus constructed the recursive structure for this Δ, and can perform a query as in the full structure of Theorem 8, taking query time $\mathcal{O}(\log n)$.

Denote the storage required by the structure for set H and chosen halfspace $h \in H$ by $S(H,h)$, and denote the expected storage for a random facet of the polytope $\mathcal{P}(H)$ by $S(H)$ (where the expectation is over the choice of $h \in H$, so $S(n) = \sum_{h \in H} S(H,h)/n$). The maximum value of $S(H)$ for any set of n halfspaces is denoted by $S(n)$.

The crucial observation is that $\Delta \in \Xi_h$ exactly if $h \in H_\Delta$. We thus have (choosing $\alpha > 0$ small enough so that the storage for the secondary structure for Ξ is in $\mathcal{O}(n^{\lfloor d/2 \rfloor - 1})$)

$$\begin{aligned}
S(n) &= \frac{1}{n} \sum_{h \in H} S(H,h) \\
&= \frac{1}{n} \sum_{h \in H} \sum_{\Delta \in \Xi_h} S(H_\Delta, h) + \mathcal{O}(n^{\lfloor d/2 \rfloor - 1}) \\
&\leq \frac{1}{n} \sum_{\Delta \in \Xi} \sum_{h \in H_\Delta} S(H_\Delta, h) + \mathcal{O}(n^{\lfloor d/2 \rfloor - 1}) \\
&\leq \frac{1}{n} \sum_{\Delta \in \Xi} |H_\Delta| \, S(H_\Delta) + \mathcal{O}(n^{\lfloor d/2 \rfloor - 1}) \\
&\leq \frac{1}{n} c_1 r^{\lfloor d/2 \rfloor} \frac{n}{r} S(n/r) + \mathcal{O}(n^{\lfloor d/2 \rfloor - 1}) \\
&= c_1 r^{\lfloor d/2 \rfloor - 1} S(n/r) + \mathcal{O}(n^{\lfloor d/2 \rfloor - 1})
\end{aligned}$$

This recursion solves to $S(n) = \mathcal{O}(n^{\lfloor d/2 \rfloor - 1} \log^c n)$ for some constant $c > 0$. For the preprocessing time we have a slight problem: To get the same recursion as above for the space, we have to be able to build the cutting and to identify the sets H_Δ in time $\mathcal{O}(n^{\lfloor d/2 \rfloor - 1})$. As we have seen before that this can be done in time $\mathcal{O}(nr^{\lfloor d/2 \rfloor})$, this is no problem for dimension d at least 6. For dimension 5 ($d = 4$ is even and therefore harmless anyway), however, $\mathcal{O}(nr^{\lfloor d/2 \rfloor})$ is not good enough a bound. Let us consider this case. As mentioned above, the cutting Ξ itself can be computed in time $\mathcal{O}(n \log r) = \mathcal{O}(n \log n)$. It remains to identify the sets H_Δ. We build in time $\mathcal{O}(n \log n)$ a data structure storing the set H, such that for a simplex Δ we can report in time $\mathcal{O}(n^{4/5}(\log n)^{O(1)} + |H_\Delta|)$ the set H_Δ of halfspaces whose bounded hyperplanes intersect Δ. This can be done by using a partition tree for simplex range searching, see [Mat91a] for details. Now we observe that we only have to determine the sets H_Δ for the simplices $\Delta \in \Xi_h$, and that we can easily determine Ξ_h from Ξ in time $\mathcal{O}(r^{\lfloor d/2 \rfloor})$. Using the structure built before, we can therefore identify the sets H_Δ for all $\Delta \in \Xi_h$ in

time $\mathcal{O}(|\Xi_h|(n/r))$. Since

$$\frac{1}{n}\sum_{h\in H}|\Xi_h| = \frac{1}{n}\sum_{\Delta\in\Xi}|H_\Delta| \leq \mathcal{O}(\frac{1}{n}\cdot r^2\cdot\frac{n}{r}) = \mathcal{O}(r),$$

the average size of Ξ_h is $\mathcal{O}(r)$. This implies that the average time to identify the sets H_Δ is only $\mathcal{O}(r(n/r)) = \mathcal{O}(n)$. Thus we get the same recursion as above, replacing the additive term by $\mathcal{O}(n\log n)$, which does not influence the result. □

Corollary 16 *For any dimension d, we can maintain the intersection of halfspaces under random insertions of halfspaces with $\mathcal{O}(m^{\lfloor d/2\rfloor}(\log m)^{O(1)})$ expected storage, $\mathcal{O}(m^{\lfloor d/2\rfloor-1}(\log m)^{O(1)})$ expected update time, and $\mathcal{O}(\log^2 m)$ query time (with high probability) for ray shooting queries, where m is the current number of halfspaces.*

It is possible to tackle the ray shooting problem on convex polytopes in the presence of deletions of halfspaces by going further along the lines demonstrated in this section. A treatment of this problem is omitted for reasons of space. Furthermore, the author has not found any way of dealing with deletions of hyperplanes without paying a high price in the complexity of the resulting data structure. The main justification for considering algorithms that perform well *on the average* only, taken over all possible sequences of updates, was that we hoped to find structures that are considerably simpler (compared, for example, to the structure by Agarwal and Matoušek [AM91] cited before). This is already somewhat dubious for the structure of Corollary 16. It is for this reason as well that we do not pursue our ideas any further from this point.

Acknowledgments: Many thanks go to Jirka Matoušek and Emo Welzl for lots of helpful discussions. Furthermore, I would like to thank all participants of the First Utrecht Workshop on Computational Geometry, especially Leo Guibas, for helpful discussions and remarks. I would also like to thank Pankaj Agarwal for his comments on previous versions of this work.

References

[AESW90] Pankaj K. Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. In *Proc. 6^{th} Ann. ACM Symp. Computational Geometry*, pages 203–210, 1990.

[AM91] Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. Tech. Report CS-1991-43, Duke University 1991, 1991.

[AM92] Pankaj K. Agarwal and Jiří Matoušek. Ray shooting and parametric search. In *24th Symp. on Theory of Computing*, 1992. To appear. Also published as Tech. Report CS-1991-22, Duke University 1991.

[BC92] Hervé Brönnimann and Bernard Chazelle. How hard is halfspace range searching? These proceedings.

[BDS+90] Jean-Daniel Boissonnat, Olivier Devillers, René Schott, Monique Teillaud, and Mariette Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete & Computational Geometry*, 1990. To appear. Available as Technical Report INRIA 1285.

[BDT90] Jean-Daniel Boissonnat, Olivier Devillers, and Monique Teillaud. A randomized incremental algorithm for constructing higher order Voronoi diagrams. *Algorithmica*, 1990. (to appear), an abstract has been published in the 2^{nd} *Canadian Conference on Computational Geometry*, 1990.

[CEGS89] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. In *Proc. 16th International Colloquium on Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 179–192, 1989.

[CF92] Bernard Chazelle and Joel Friedman. Point location among hyperplanes and vertical ray shooting. *Computational Geometry: Theory and Applications*, 1992. To appear.

[Cha85] Bernard Chazelle. How to search in history. *Information and Control*, 64:77–99, 1985.

[Cla87] Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

[Cla88] Kenneth L. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17:830–847, 1988.

[CMS92] Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental construction. In *Symp. on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 463–474. Springer-Verlag, 1992.

[CS89] Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[DMT92] Olivier Devillers, Stefan Meiser; and Monique Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Computational Geometry: Theory and Applications*, 1992. To appear. Available as Technical Report INRIA 1349. Abstract published in LNCS 519 (WADS'91, august 1991).

[Ede87] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[GKS92] Leo J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992.

[Mat91a] Jiří Matoušek. Efficient partition trees. In *Proc. 7th Symp. on Computational Geometry*, pages 1–9, 1991.

[Mat91b] Jiří Matoušek. Reporting points in halfspaces. In *32nd Symp. Foundations of Computer Science*, pages 207–215, 1991.

[Mat92] Jiří Matoušek. Personal communications, February 1992.

[Mul88] Ketan Mulmuley. A fast planar partition algorithm, I. In *29th Symp. Foundations of Computer Science*, pages 580–589, 1988.

[Mul91a] Ketan Mulmuley. Randomized, multidimensional search trees: dynamic sampling. In *Proc. 7th Symp. on Computational Geometry*, pages 121–131, 1991.

[Mul91b] Ketan Mulmuley. Randomized multidimensional search trees: Further results in dynamic sampling. In *32nd Symp. Foundations of Computer Science*, pages 216–227, 1991.

[Mul91c] Ketan Mulmuley. Randomized multidimensional search trees: Lazy and dynamic shuffling. In *32nd Symp. Foundations of Computer Science*, pages 180–196, 1991.

[Sch91] Otfried Schwarzkopf. Dynamic maintenance of geometric structures made easy. In *32nd Symp. Foundations of Computer Science*, pages 197–206, 1991.

[Sei91a] Raimund Seidel. Low dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6(5):423–434, 1991.

[Sei91b] Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, 1:51–64, 1991.