CUTTING HYPERPLANES FOR DIVIDE-AND-CONQUER

Bernard Chazelle

CS-TR-335-91

June 1991

Cutting Hyperplanes for Divide-and-Conquer

Bernard Chazelle
Department of Computer Science
Princeton University
Princeton, NJ 08544

Abstract: Given n hyperplanes in E^d, a $(1/r)$-*cutting* is a collection of simplices which together cover E^d and such that the interior of each simplex intersects at most n/r hyperplanes. We present an algorithm for computing a $(1/r)$-cutting of (asymptotically) minimum size in $O(nr^{d-1})$ time. If, as is the case in practice, the lists of cutting hyperplanes must be explicitly provided, then the algorithm is optimal. Our result bridges a gap in a recent algorithm of Matoušek by extending its performance to all values of r; the previous bound was restricted to $r \leq n^{1-\delta}$, for any fixed $\delta > 0$. To attain our goal, we show that optimal cuttings can be refined by composition. This is interesting in its own right, because it leads to the improvement and the simplification of a number of geometric algorithms, e.g., point location among hyperplanes, counting segment intersections, Hopcroft's line/point incidence problem, linear programming in fixed dimension. One of the main tools used in the cutting construction is a proof that ε-approximations can be used to estimate how many vertices of a hyperplane arrangement fall inside a given simplex. In a different development, to be reported elsewhere, we have used this lemma to derive an optimal deterministic algorithm for computing convex hulls in higher dimensions. Thus, we suspect that the lemma will have other useful applications in computational geometry.

The author wishes to acknowledge the National Science Foundation for supporting this research in part under Grant CCR–9002352.

1. Introduction

Let H be a set of n hyperplanes in E^d: A $(1/r)$-*cutting* for H is any collection of (possibly unbounded) d-dimensional closed simplices which together cover E^d and such that the interior of each simplex intersects at most n/r hyperplanes [22]. Spurred by the works of Clarkson [8,9] and Haussler and Welzl [18], cuttings have proven enormously successful for solving all kinds of geometric problems; see, e.g., [4,8,9,11,12,15,26]. The reason for this success is that cuttings play a role analogous to separators in graph algorithms: they set the grounds for efficient divide-and-conquer schemes. Of course, the *size* of a cutting (i.e., its cardinality) is always a critical parameter. A size of $O(r^d \log^d r)$ is achievable by a straightforward probabilistic algorithm. Actually, with a little more effort, the size can be reduced to $O(r^d)$, which is optimal [5]. If, for each simplex, the collection of intersecting hyperplanes is to be explicitly computed, then trivially $\Omega(nr^{d-1})$ is a lower bound on the complexity of computing a $(1/r)$-cutting. While this bound can be easily attained using randomization [5] the search for an optimal deterministic algorithm has been more difficult. An efficient algorithm for computing optimal cuttings in two dimensions was given by Matoušek [19] and then improved by Agarwal [1]. Chazelle and Friedman showed that optimal cuttings are computable in polynomial time in any dimension [5]. Recently, Matoušek [20,21,22] came close to putting the whole question to rest by exhibiting several optimal algorithms for computing a minimum-size $(1/r)$-cutting, provided that $r < n^{1-\delta}$, for any fixed $\delta > 0$. (The term minimum is to be understood in the asymptotic sense.) We bridge this gap by relaxing the restriction on r. Specifically, we show how to compute a $(1/r)$-cutting of size $O(r^d)$ in $O(nr^{d-1})$ time, for any value of $r \leq n$.

Perhaps more interesting than the result itself is the method used to achieve it. The reason why Matoušek's bound does not extend to large values of r is that standard cuttings do not "compose" very well. Indeed, suppose that we attempt to refine a cutting by taking each simplex in turn and applying Matoušek's algorithm to the set of hyperplanes intersecting its interior. This produces a $(1/r^2)$-cutting for H of size $O(r^{2d})$. But the big-oh notation conceals the fact that the constant in the original $O(r^d)$ space bound gets squared in the process. Because it is greater than one, further iteration of this composition process might end up causing big trouble. Composing cuttings is a very popular and useful algorithmic paradigm, however, so it is regrettable that it should suffer from such a serious flaw. If r is a constant, for example, composing cuttings (as in [8,9]) adds undesirable multiplicative factors of the form n^ε to the running times.

One possible interpretation of this problem is that composing cuttings is a "nonstable" computing process. If the size of a cutting is cr^d, then c can be regarded as a multiplicative error term. The nonstable part of the composition process is that it amplifies the error and does not keep it bounded. The remedy is to have the algorithm rely not so closely on the cutting of the previous generation but on some global quantity *independent* of the composition process itself, such as the number of vertices inside each simplex. Computing such quantities efficiently is hopeless, so one must devise a scheme for *estimating* them. Of course, this still means having to deal with errors, but those errors can be made to be additive and not multiplicative, which is good enough for our purposes. We are thus led to develop a new kind of cuttings, which enjoy all the properties of the old ones, and in addition, allow themselves to be refined by composition. Besides leading to an optimal construction algorithm this ability to be refined gives our cuttings greater versatility. We use them to derive:

1. A very simple optimal algorithm for locating a point in a collection of n hyperplanes. The algorithm has $O(\log n)$ query time and requires $O(n^d)$ preprocessing time and space. This improves the preprocessing time of Chazelle and Friedman's solution [6]. (In fairness the solution in [6] also supports unidirectional ray-shooting, which this one does not.) If the query asks only whether a point lies on one of the hyperplanes, then the preprocessing can be reduced to $O(n^d/(\log n)^{d-1})$.
2. A solution to Hopcroft's problem (i.e., detecting any incidence between n lines and n points) in $O(n^{4/3}(\log n)^{1/3})$ time and linear space, which slightly improves on the complexity of a solution by Agarwal [2]. Also, the algorithm generalizes immediately to higher dimensions.
3. An algorithm for counting the number of intersections among n segments in $O(n^{4/3}(\log n)^{1/3})$ time and linear space, which improves on the solutions of Guibas et al. [17] and Agarwal [2].
4. A straightforward linear-time algorithm for linear programming with a fixed number of variables, which greatly simplifies the methods of [7,13,24]. This result, which was obtained independently by Matoušek [23], is derived by direct derandomization of a probabilistic algorithm of Clarkson [10].

The theory of ε-nets is used for two purposes in the cutting construction: one is to provide separation properties, and the other is to build efficient estimators. In particular, we prove a lemma which roughly says that ε-approximations can be used to efficiently estimate how many vertices in a hyperplane arrangement lie inside a given simplex. Interestingly, ε-nets are too weak to provide any kind of meaningful approximation for that quantity. (As an exercise, the reader should try to illustrate this on an example in two dimensions.) In a different development, to be reported elsewhere, we have used the same lemma to derive an optimal deterministic algorithm for computing convex hulls in higher dimensions [3]. Thus, we suspect that the lemma will have other useful applications in computational geometry.

2. Refining Cuttings by Composition

The main idea here is to use a finer measure of the effectiveness of a cutting. The intuition is that in the composition process second-generation cuttings are only useful within the confines of the (first-generation) simplices for which they are defined. Thus, similarly to what Agarwal did in his construction of two-dimensional cuttings [1], we wish to keep their sizes in line with the complexity of the (partial) arrangements inside the first-generation simplices. What makes our task harder is that in higher dimensions none of the clever geometric tricks used in [1] work any more, and completely general techniques must be developed. The key tool we shall use is that, given an arrangement of hyperplanes, an ε-approximation can be used to estimate the number of vertices inside any query simplex. Interestingly, this not possible using ε-nets. This is a consequence of the fact that unlike ε-nets, ε-approximations provide estimations from below as well as above.

For simplicity, we shall assume that the set H of n hyperplanes in E^d is in general position. This can be relaxed without any difficulty by using the perturbation techniques of [16,30]. We begin with two definitions adapted from [29]. Let $R \subseteq H$ be a subset of ρ hyperplanes; given a line segment e

in E^d, let R_e (resp. H_e) denote the number of hyperplanes of R (resp. H) that intersect the relative interior of e. Given a (closed) simplex s, we also use the notation R_s (resp. H_s) to refer to the number of arrangement vertices created by H (resp. R) in the relative interior of s. We say that R is a $(1/r)$-*approximation* for H if, for any e, the densities in R and H of the hyperplanes crossing e differ by less than $1/r$, i.e.,

$$\left|\frac{R_e}{\rho} - \frac{H_e}{n}\right| < \frac{1}{r}.$$

Whereas a $(1/r)$-approximation acts as an estimator, a $(1/r)$-net acts as a threshold function: R is a $(1/r)$-net if, for any segment e, the inequality $H_e > n/r$ implies that $R_e > 0$. A beautiful result of Matoušek [20] says that, if r is a constant, then it is possible to compute a $(1/r)$-approximation as well as a $(1/r)$-net in linear time. We need to strengthen the notion of a $(1/r)$-net a little by requiring that the facial complexity of the portion of the arrangement that it forms within a given simplex is not more than expected. We say that a subset $R \subseteq H$ is a *strong* $(1/r)$-*net for* (H, s) if:

(i) For any segment e, $H_e > n/r$ implies $R_e > 0$;

(ii) $R_s \leq 4(\rho/n)^d H_s$.

Lemma 2.1 (Matoušek, [20]). *Given a collection H of n hyperplanes in E^d, it is possible to compute a $(1/r)$-approximation for H of size $O(r^2 \log r)$ in time $nr^{O(1)}$.*

Lemma 2.2. *Given a collection H of n hyperplanes in E^d and a simplex s, it is possible to compute a strong $(1/r)$-net for (H, s) of size $O(r \log n)$ in time polynomial in n.*

Proof: The lemma is not as strong as it could be, but it is easier to prove that way and still powerful enough for our purposes. To begin with, we show that a random sample R of size $\rho = \min\{\lceil (2d+1) r \log n \rceil, n\}$ constitutes a strong $(1/r)$-net for (H, s) with nonzero probability. † We can obviously assume that $\rho < n$. The expected value E of R_s is

$$\binom{n-d}{\rho-d} H_s \Big/ \binom{n}{\rho} \leq (\rho/n)^d H_s,$$

therefore by Markov's inequality,

$$\text{Prob}\left[R_s > 4(\rho/n)^d H_s\right] < \frac{E}{4(\rho/n)^d H_s} \leq 1/4.$$

Next, pick a point inside each cell of the arrangement formed by H and let S be the set of all segments connecting pairs of these points. It suffices to ensure that for each $e \in S$, $H_e > n/r$ implies $R_e > 0$. The probability p_e of e failing that test is

$$\binom{n-H_e}{\rho} \Big/ \binom{n}{\rho} < \left(1 - \frac{1}{r}\right)^{\rho} < e^{-\rho/r} < 1/n^{2d+1}.$$

Since the number of segments in S is $O(n^{2d})$, for n large enough, we have

$$\frac{E}{4(\rho/n)^d H_s} + \sum_{e \in S} p_e < \frac{1}{2},$$

and therefore a random R is a strong $(1/r)$-net for (H, s) with probability greater than $1/2$.

† All logarithms are to the base 2.

To convert this existence proof into a polynomial-time algorithm, we use Raghavan and Spencer's method of conditional probabilities [25,28] (see also [5,20] for previous applications of the method to cuttings). We pick sample elements one at a time, always making sure that the sample can be completed into a strong $(1/r)$-net. Let R' be a partial pick; the next sample element h is chosen in $H \setminus R'$ so as to minimize the value of

$$\frac{E(R' \cup \{h\})}{4(\rho/n)^d H_s} + \sum_{e \in S} p_e(R' \cup \{h\}),$$

where the argument $R' \cup \{h\}$ indicates that the corresponding expectation or probability is conditioned upon having picked $R' \cup \{h\}$ as part of the sample.

Each $p_e()$ can be expressed by means of binomial coefficients and thus can be computed effectively. Note that as in [5,20] we can limit the size of the arithmetic operations by rounding off the calculations appropriately, as long as the final absolute error is less than $1/2$. Because the conditional probability that a given vertex is to be eventually created by R depends solely on how many of its defining hyperplanes have already been selected, it is equally easy to evaluate $E()$ in polynomial time. When R' reaches size ρ, all the conditions for a strong $(1/r)$-net are met and we can set $R = R'$. ∎

We use the two previous lemmas to compute an optimal $(1/r)$-cutting for H. Let $r_0 \leq r$ be a constant large enough; the implication on r of this assumption is justifiable in light of Matoušek's optimal results for small r. For $k = 1, \ldots, \lceil \log_{r_0} r \rceil$, we compute a $(1/r_0^k)$-cutting C_k for H by successive refinement. The last C_k is a $(1/r)$-cutting for H.

For $k = 1$, we simply apply Matoušek's algorithm, which produces a $(1/r_0)$-cutting of size $O(r_0^d)$ in $O(n)$ time. For subsequent values of k, we refine C_{k-1} into C_k by cutting up its simplices into small pieces one-by-one. Let s be a simplex of C_{k-1} and let $H(s)$ be the set of hyperplanes of H intersecting its interior. We assume by induction that $H(s)$ is explicitly available for each s. If $|H(s)| \leq n/r_0^k$, then s stands as such. Otherwise, calling on Lemmas 2.1 and 2.2, we first compute a $(1/2d\rho_0)$-approximation A for $H(s)$ and then a strong $(1/2d\rho_0)$-net R for (A, s), where

$$\rho_0 = r_0^k |H(s)|/n.$$

Finally, we compute the arrangement formed by R and the $d+1$ hyperplanes bounding s and form its *canonical triangulation* [9]: this is obtained by first triangulating recursively the $(d-1)$-dimensional cross-section of the arrangement made by each hyperplane, and then for each cell of the arrangement, lifting all the k-simplices on its boundary $(k = 0, \ldots, d-1)$ toward a chosen vertex (except for the simplices decomposing the faces incident upon the vertex in question). The set of d-dimensional simplices inside s constitutes the contribution of that simplex to C_k. We repeat the procedure for each s in C_{k-1}.

To see that C_k is, indeed, a $(1/r_0^k)$-cutting is immediate. It suffices to show that the interior of none of the new simplices s_0 created within s intersects more than $n/r_0^k = |H(s)|/\rho_0$ hyperplanes of $H(s)$. First, observe that no segment e in the interior of s_0 can intersect more than $|H(s)|/(d\rho_0)$ hyperplanes. Indeed, if that were to be the case, then e would intersect more than

$$\frac{(|H(s)|/d\rho_0)|A|}{|H(s)|} - \frac{|A|}{2d\rho_0} = \frac{|A|}{2d\rho_0}$$

hyperplanes of A, and hence, at least one hyperplane of R, which is impossible. Now, any vertex of s_0 is defined by d hyperplanes of H (not necessarily bounding s_0) that avoid the interior of s_0. Therefore, by general position, any hyperplane of H meeting the interior of s_0 must avoid its vertices, and hence, must meet one of d line segments in the interior of s_0 infinitesimally close to, say, the d edges incident upon a given vertex of s_0. This implies that at most $d \times |H(s)|/(d\rho_0)$ hyperplanes can meet the interior of s_0 and therefore that C_k is a $(1/r_0^k)$-cutting.

To estimate the size of C_k is more delicate. We can check that $\rho_0 \leq r_0$, and therefore, $|C_k| = O\big(|C_{k-1}|(r_0 \log r_0)^d\big)$, but this upper bound is too crude to do us any good. We must show that using the strong net R helps keep C_k small when H_s, the number of vertices formed by H in the interior of s, is small. To compute H_s can be very expensive, however, but it can be approximated fairly accurately on the basis A_s alone. This yields an upper bound on R_s, and hence, $|C_k|$. For convenience, we shall use the notation $|H(s)| = \nu$, $|A| = \alpha$, and $|R| = \rho$. Note that $\nu \leq n/r_0^{k-1}$, $\alpha = O(\rho_0^2 \log \rho_0)$, and $\rho = O(\rho_0 \log \rho_0)$.

Lemma 2.3. $\Big|H_s - (\nu/\alpha)^d A_s\Big| \leq \nu^d/\rho_0$ *and* $R_s \leq 4(\rho/\nu)^d H_s + 4\rho^d/\rho_0$.

Proof: Let F be a k-flat equal to E^d if $k = d$, or else formed as a $(d-k)$-wise intersection of hyperplanes in A, and assume that F intersects the interior of s. Let $H(F)$ (resp. $A(F)$) denote the number of vertices of the k-dimensional arrangement induced in F by H (resp. A) that lie in the relative interior of $s \cap F$. Note that $H_s = H(E^d)$ and $A_s = A(E^d)$. We prove by induction that for $k = 1, \ldots, d$,

$$\Big|H(F) - \Big(\frac{\nu}{\alpha}\Big)^k A(F)\Big| \leq \frac{\nu^k}{\rho_0}. \tag{2.1}$$

The case $k = 1$ is easy. The set $s \cap F$ is a line segment. We have

$$\Big|\frac{A(F)}{\alpha} - \frac{H(F)}{\nu}\Big| \leq \frac{1}{2d\rho_0},$$

therefore $|\nu A(F)/\alpha - H(F)| \leq \nu/(2d\rho_0) \leq \nu/\rho_0$.

Assume now that $k > 1$. Let F_i $(1 \leq i \leq \alpha - d + k)$ be the intersection of F with a hyperplane of A (not used by F), and let L_j $(1 \leq j \leq \binom{\nu-d+k}{k-1})$ be the line obtained by intersecting F with a $(k-1)$-wise intersection of hyperplanes in $H(s)$ (not used by F). Since $\sum_j H(L_j) = kH(F)$ and $\sum_j A(L_j) = \sum_i H(F_i)$, summing together the inequalities

$$\Big|H(L_j) - \Big(\frac{\nu}{\alpha}\Big) A(L_j)\Big| \leq \frac{\nu}{2d\rho_0}$$

yields

$$\Big|kH(F) - \Big(\frac{\nu}{\alpha}\Big) \sum_i H(F_i))\Big| \leq \frac{\nu}{2d\rho_0}\binom{\nu - d + k}{k-1}.$$

By induction, we know from (2.1) that

$$\left|H(F_i) - \left(\frac{\nu}{\alpha}\right)^{k-1} A(F_i)\right| \le \frac{\nu^{k-1}}{\rho_0},$$

and therefore

$$\left|kH(F) - \left(\frac{\nu}{\alpha}\right)^{k} \sum_i A(F_i)\right| \le \frac{\nu^k(\alpha - d + k)}{\alpha\rho_0} + \frac{\nu}{2d\rho_0}\binom{\nu - d + k}{k-1}.$$

Since $\sum_i A(F_i) = kA(F)$, we finally derive

$$\left|H(F) - \left(\frac{\nu}{\alpha}\right)^{k} A(F)\right| \le \frac{\nu^k(\alpha - d + k)}{k\alpha\rho_0} + \frac{\nu}{2kd\rho_0}\binom{\nu - d + k}{k-1} \le \frac{\nu^k}{\rho_0},$$

which establishes (2.1) and the first part of the lemma. Because R is a strong $(1/2d\rho_0)$-net for A, we have $R_s \le 4(\rho/\alpha)^d A_s$, and the second part of the lemma follows. ▮

The first part of the lemma is a very useful tool, and it is worthwhile to restate it in a slightly more general form. Note that the fact that all the ν hyperplanes happen to meet the interior of s is never used in the proof. Therefore, the lemma still holds for any simplex s. We can also generalize the result to a simplex of arbitrary dimension within the ambient space.

Lemma 2.4. *Let H be a set of n hyperplanes in E^d and let A be an (already computed) $(1/r)$-approximation for H. Given any relatively open simplex s of dimension j, the number of vertices of the arrangement of H that fall within s can be estimated in time $r^{O(1)}$ with an absolute error of at most $2jn^j/r$.*

Proof: The case $j = d$ is a mere restatement of Lemma 2.3. If $j < d$, the key observation is that the cross-section of A by the j-flat F spanned by s is a $(1/r)$-approximation for the arrangement in F induced by H. ▮

We are now in a position to show that the size of the last C_k is $O(r^d)$ and that it can be found in $O(nr^{d-1})$ time. We need to derive an upper bound on the number of simplices that $s \in C_{k-1}$ can contribute to C_k. The facial complexity of the portion of the arrangement of R within s is $O(\rho^{d-1} + R_s)$. To see this, observe that up to within a constant factor every feature can be accounted for by the facial structure along the boundary of s, which is of size $O(\rho^{d-1})$, and by the R_s vertices created by R inside s. (To see this, consider the argument prior to triangulation, and then observe that triangulating does not multiply the facial complexity by more than a constant factor.) It follows from Lemma 2.3 that the triangulation of s consists of a number of simplices at most proportional to

$$(\rho_0 \log \rho_0)^{d-1} + \left(\frac{\rho_0 \log \rho_0}{|H(s)|}\right)^d H_s + \rho_0^{d-1}(\log \rho_0)^d.$$

Since $\rho_0 \le r_0$ we have $\rho_0(\log\rho_0)/|H(s)| \le r_0^k(\log r_0)/n$. Now, because $\sum_{s\in C_{k-1}} H_s \le \binom{n}{d}$ and $|C_1| = O(r_0^d)$, we find that for some constant $c > 0$ (independent of r_0), we have $|C_1| \le cr_0^d$, and for $k > 1$,

$$|C_k| \le c\left(\frac{r_0^k \log r_0}{n}\right)^d n^d + cr_0^{d-1}(\log r_0)^d |C_{k-1}|.$$

We easily prove by induction that if r_0 is a large enough constant, $|C_k| \le r_0^{(k+1)d}$. This implies that the last C_k is of size $O(r^d)$, as desired.

What is the time needed to compute all the C_k's? To obtain C_1 takes $O(n)$ time. Regarding C_k, by virtue of Lemmas 2.1 and 2.2, for each s it takes $O(\nu)$ time to compute A and constant time to get R and triangulate its clipped arrangement. The set of crossing hyperplanes is also obtained within the same amount of time, since R has constant size. Consequently, every time a simplex is created it incurs a cost proportional to the number of hyperplanes intersecting its interior. The running time is therefore on the order of

$$\sum_{1\le k\le \lceil \log_{r_0} r\rceil} \left(\frac{n}{r_0^k}\right)|C_k| \le nr_0^{k(d-1)+d} = O(nr^{d-1}),$$

which completes the proof. We have thus achieved our goal.

Note that the proof would not work if the size of the strong net was a little bigger, e.g., ρ_0^2, or if the facial structure of the boundary of s was, say, of complexity $\Omega(\rho^d)$.

Theorem 2.5. *Given a collection H of n hyperplanes in E^d, for any $r \le n$ it is possible to compute a $(1/r)$-cutting for H of size $O(r^d)$ in time $O(nr^{d-1})$.*

3. Applications of the Cutting Construction

One of the most interesting features of Theorem 2.5 is that it is achieved by refining cuttings of constant size. This allows us to simplify many of the applications for which cuttings have been used, and also improve their space and time complexity. All these applications are very similar in spirit. We use cuttings to break up a problem into a well-balanced set of subproblems, which we solve recursively by divide-and-conquer. Since the cuttings have constant size, the divide part is particularly simple. Moreover, by evaluating the recursion tree in a depth-first search manner, we can keep the storage linear. Also, by stopping the recursion shortly before it bottoms out, and then finishing the work naively, we can produce small additional savings. Many applications of cuttings have been given in the literature. Because most of them bear a certain family resemblance, we will discuss only three fairly representative cases and briefly sketch their solutions.

3.1. Point Location. This is not an application per se. It is the mere observation that the hierarchical sequence of cuttings C_k *is* a data structure for point location. The problem is to preprocess a collection H of n hyperplanes in E^d so that given a query point the face of H that encloses it can be determined very fast. Set $r = n$ in the construction. By tracing the query point from cell to cell across C_1, C_2, etc., we eventually land in a simplex entirely contained in a cell, at which point we can easily find the desired answer. Adding the necessary pointers to turn this nested hierarchy of cuttings into a point location structure is routine. We outline the procedure: First, compute the full arrangement in $O(n^d)$ time [14]; then at stage k, keep the clipped portions of the arrangement within each simplex s of C_k. To process stage k, we need not look at the whole arrangement within s but only at the vicinity of the boundaries of the simplices. Recalling that $|C_k| \le r_0^{(k+1)d}$, with a bit of care we can ensure that the total preprocessing time is on the order of

$$\sum_k r_0^{(k+1)d}\left(\frac{n}{r_0^k}\right)^{d-1} = O(n^d).$$

This improves the preprocessing time of Chazelle and Friedman's solution [6]. (In fairness the solution in [6] also supports undirectional ray-shooting, which this one does not.) The storage requirement is $O(n^d)$ and the query time is $O(\log n)$.

Theorem 3.1. *Given a collection H of n hyperplanes in E^d, we can preprocess it for point location in $O(n^d)$ time and space, so that given a query point, the face of the arrangement enclosing the point can be found in $O(\log n)$ time.*

Note that by setting $r = n/\log n$ and pursuing the search naively at the bottom of the hierarchy, we can detect whether the query point lies on any of the input hyperplanes in $O(\log n)$ time. This reduces the preprocessing to $O(n^d/(\log n)^{d-1})$.

Theorem 3.2. *Given a collection H of n hyperplanes in E^d, we can preprocess it in $O(n^d/(\log n)^{d-1})$ time and space, so that given a query point, whether the point lies on at least one of the hyperplanes can be checked in $O(\log n)$ time.*

3.2. Hopcroft's Problem. The problem is to decide whether, among n lines and n points in the plane, at least one of the lines passes through at least one point. An earlier randomized expected complexity bound for this problem given by Edelsbrunner et al. [15] was later improved and made deterministic by Agarwal [2]. His solution requires $O(n^{4/3}(\log n)^{1.78})$ time. Our algorithm is also deterministic and improves the time to $O(n^{4/3}(\log n)^{1/3})$. It also generalizes to any dimension d, where the problem is to detect incidence between n hyperplanes and n points. (One can also tailor the solution to the case of n hyperplanes and $m \ne n$ points, and to allow the explicit reporting of all incidences.) Our improvement is due to a combination of two factors: using optimal cuttings and replacing standard point location by the improved incidence detection method of Theorem 3.2.

Theorem 3.3. *Detecting any incidence between n hyperplanes and n points in d-space can be done deterministically in $O(n^{2d/(d+1)}(\log n)^{1/(d+1)})$ time and linear space.*

Proof: In $O(nr^{d-1} + n \log n)$ time, we compute a $(1/r)$-cutting for the hyperplanes (for some well chosen r) by applying the cutting refinement method, and we use the associated point location structure to locate all the n points in their enclosing simplices. If any incidence can be discovered at this time, we stop. Else, we break up the subset of points within each simplex into groups of size roughly n/r^d or less. Next, we apply the dual version of Theorem 3.2 to each group and we query each group with respect to each hyperplane cutting its enclosing simplex. The total number of queries is $O(nr^{d-1})$. Setting $r^{d^2-1} = n^{d-1}/(\log n)^d$ gives a running time of $O(nr^{d-1} \log n + n^d r^{d-d^2}/(\log n)^{d-1})$, which is $O(n^{2d/(d+1)}(\log n)^{1/(d+1)})$. To make the storage requirement linear, we refine cuttings in a depth-first search fashion. The storage will be dominated by the space needed to store a single point location structure, which is easily seen to be $O(n)$. ∎

3.3. Counting Segment Intersections. Given n line segments in the plane, we wish to count how many pairwise intersections they form. A randomized algorithm by Guibas et al. [17] solves the problem in $O(n^{4/3+\varepsilon})$ expected time, for any $\varepsilon > 0$, and linear space. Agarwal [2] found a deterministic solution requiring $O(n^{4/3} \log^{1.78} n)$ time and using $O(n^{4/3}/\log^{2.56} n)$ space. We slightly improve upon it.

Theorem 3.4. *The number of intersections among n segments in the plane can be determined in $O(n^{4/3}(\log n)^{1/3})$ time and linear space.*

Proof: The algorithm is similar to Agarwal's. We begin with a special case of the problem, where n segments intersect a triangle Δ and m of them have at least one endpoint in Δ: how many intersections fall inside Δ? The intersections among *long* segments, i.e., those crossing Δ through and through, can be counted in $O(n \log n)$ time [2]. The number of intersections between long segments and short (i.e., non-long) ones can be found by dualizing the problem and counting the number of point/double wedge inclusions. More precisely, we are given a line arrangement formed by m double wedges, and for each point obtained as the dual of the line supporting a long segment, we must count how many double wedges enclose it. By going back to the same idea used in the proof of Theorem 3.2, we construct a cutting of $O(m^2/\log^2 m)$ triangles, each of which is crossed by at most $\log m$ lines bounding double wedges. In $O(m^2/\log m)$ time, we can find all these triangles, and for each of them, (i) set up a list of the double wedges partially overlapping it, and (ii) count how many double wedges completely enclose the triangles. In this way, we can compute the short-long count in $O(n \log m)$ time. To count short-short intersections we apply, say, Agarwal's method, which takes $O(n^{4/3}(\log n)^{1.78}) = O(m^{10/7})$ time (to use a more convenient bound). To summarize, the entire computation takes $O(n \log n + m^2/\log m)$ time and space. We can improve on it by partitioning the short segments into groups of at most $\sqrt{n} \log n$ and breaking up the short-long counts accordingly. This gives a solution requiring $O(n \log n + m\sqrt{n} + m^{10/7})$ time and $O(n \log n)$ space.

To solve our original problem, we specialize the setting used in the proof of Theorem 3.3 to the case $d = 2$. In $O(nr + n \log n)$ time, we compute a $(1/r)$-cutting for the lines supporting the segments (for some well chosen r) and we use its associated point location structure to locate all the $2n$ segment endpoints in their enclosing triangles. Next, we compute the intersection counts within each of the $O(r^2)$ triangles, which requires a total of

$$O\Big(n\sqrt{\frac{n}{r}} + nr\log n + \sum_i m_i^{10/7}\Big)$$

time, where the m_i's sum up to $2n$ and none of them exceeds n/r. Setting $r = n^{1/3}/(\log n)^{2/3}$ gives a running time of $O\big(n^{4/3}(\log n)^{1/3}\big)$. Again, we can use depth-first search to keep the storage linear. Not too surprisingly, this is the same complexity as for our solution to Hopcroft's problem in two dimensions. ▮

3.4. Linear Programming. We look at the problem of optimizing a linear function over a set of n linear constraints in d-space. A remarkable result of Megiddo [24] states that if d is a constant, then the problem can be solved in linear time. The dependency on d is rather steep, however. From doubly exponential, Clarkson [7] and Dyer [13] independently reduced this dependency to a multiplicative factor of roughly 3^{d^2}. Still smaller dependencies can be achieved by allowing randomization: d^2 plus additive term (Clarkson [10]) and $d!$ (Seidel [27]). Unlike its deterministic counterparts, the probabilistic algorithm of Clarkson [10] is extremely simple. As it turns out, a straightforward variant of it can be immediately derandomized by using Matoušek's algorithm for computing constant-size ε-nets. (Note that we do not even need to use cuttings here.) This leads to a remarkably simple linear-time algorithm for linear programming when the dimension is fixed. Matoušek [23] has derived a similar linear programming algorithm independently, and has communicated to us that the dependency of its running time on d is about $d!$, which is comparable to that of Seidel's algorithm [27].

We assume that the reader is familiar with Clarkson's algorithm. Our variant is only a very slight modification of it, so we only give a rough sketch. Clarkson's idea is to take a random sample R of the constraints and solve the problem recursively. If the answer satisfies all the constraints, then we are done. Else, we can easily argue that the set V_1 of non-satisfied constraints contains at least one of the constraints defining the desired answer x^*. At this point, Clarkson resamples, but we don't really need to do that in our case. We simply solve the problem recursively on $R \cup V_1$. Again, if no constraints get in the way of the answer, we are done. Otherwise, we argue that the new set V_2 of non-satisfied constraints must now contain another constraint defining x^* (different from the previous one). So, again we solve the problem recursively on $R \cup V_1 \cup V_2$, and we iterate in this fashion. After d stages the set of non-satisfied constraints V_d, if nonempty, must be such that $R \cup V_1 \cup \cdots \cup V_d$ contains all the constraints defining x^*, so one final recursive call will produce the solution.

The two differences from Clarkson's algorithm are: (i) we begin with a sample of constant size, and (ii) we do not resample at every stage. If, instead of a random sample, we use a constant-size ε-net for the range space induced by the action of line segments on hyperplanes, we will have the property that no subproblem solved recursively exceeds size n/c, for some arbitrarily large constant c. Thus, the running time $T(n)$ follows a recurrence of the form $T(n) = O(1)$, for $n = O(1)$, and $T(n) = (d+1)T(n/c) + O(n)$, for larger n. Setting c large enough gives $T(n) = O(n)$.

It remains a very interesting open problem whether the quadratic dependency on d of Clarkson's probabilistic algorithm can be achieved deterministically. Note that this precludes computing ε-nets or cuttings, since these bring along exponential dependency on d.

Acknowledgments: I wish to thank Pankaj K. Agarwal and Jirka Matoušek for several helpful discussions.

REFERENCES

1. Agarwal, P.K. *Partitioning arrangements of lines I: An efficient deterministic algorithm,* Disc. Comput. Geom. 5 (1990), 449–483.
2. Agarwal, P.K. *Partitioning arrangements of lines II: Applications,* Disc. Comput. Geom. 5 (1990), 533–573.
3. Chazelle, B. *An optimal convex hull algorithm for point sets in any fixed dimension,* Tech. Rep. CS–TR–336–91, Princeton University, 1991.
4. Chazelle, B., Edelsbrunner, H., Guibas, J.J., Sharir, M. *A singly-exponential stratification scheme for real semi-algebraic varieties and its applications,* Proc. 16th International Colloquium on Automata, Languages and Programming, 179–192, 1989. Also, to appear in Theoret. Comput. Sci.
5. Chazelle, B., Friedman, J. *A deterministic view of random sampling and its use in geometry,* Combinatorica 10 (1990), 229–249.
6. Chazelle, B., Friedman, J. *Point location among hyperplanes and unidirectional ray-shooting,* Tech. Rep., CS–TR–333–91, June 1991, Princeton University.
7. Clarkson, K.L. *Linear programming in* $O(n3^{d^2})$ *time,* Inf. Proc. Lett. 22 (1986), 21–24.
8. Clarkson, K.L. *New applications of random sampling in computational geometry,* Disc. Comput. Geom. 2 (1987), 195–222.
9. Clarkson, K.L. *A randomized algorithm for closest-point queries,* SIAM J. Comput. 17 (1988), 830–847.
10. Clarkson, K.L. *A Las Vegas algorithm for linear programming when the dimension is small,* Proc. 29th Ann. Symp. Foundat. Comput. Sci. (1988), 452–456.
11. Clarkson, K.L., Shor, P.W. *Applications of random sampling in computational geometry, II*, Disc. Comput. Geom. 4 (1989), 387–421.
12. Clarkson, K., Tarjan, R.E., Van Wyk, C.J. *A fast Las Vegas algorithm for triangulating a simple polygon,* Disc. Comput. Geom. 4 (1989), 432–432.
13. Dyer, M.E. *On a multidimensional search technique and its application to the Euclidean one-centre problem,* SIAM J. Comput. 15 (1986), 725–738.
14. Edelsbrunner, H. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, Germany, 1987.
15. Edelsbrunner, H., Guibas, L.J., Herschberger, J., Seidel, R., Sharir, M., Snoeyink, J., Welzl, E. *Implicitly representing arrangements of lines or segments,* Disc. Comput. Geom. 4 (1989), 433–466.
16. Edelsbrunner, H., Mücke, P. *Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms,* ACM Trans. Comput. Graphics 9 (1990), 66–104.
17. Guibas, L.J., Overmars, M., Sharir, M. *Counting and reporting intersections in arrangements of line segments,* Tech. Rep. 434, Dept. Comput. Sci., New York Univ., (1989).

18. Haussler, D., Welzl, E. *Epsilon-nets and simplex range queries,* Disc. Comput. Geom. 2, (1987), 127–151.

19. Matoušek, J. *Construction of ε-nets,* Disc. Comput. Geom. 5 (1990), 427–448.

20. Matoušek, J. *Approximations and optimal geometric divide-and-conquer,* Proc. 23rd Ann. ACM Symp. Theory of Comput. (1991), 505–511.

21. Matoušek, J. *Efficient partition trees,* KAM Series (tech. report) 90-175, Charles University, 1990.

22. Matoušek, J. *Cutting hyperplane arrangements,* to appear in Disc. Comput. Geom., 1991. Also, in Proc. 6th Ann. ACM Symp. Comput. Geom. (1990), 1–9.

23. Matoušek, J. Private communication, 1991.

24. Megiddo, N. *Linear programming in linear time when the dimension is fixed,* J. ACM 31 (1984), 114–127.

25. Raghavan, P. *Probabilistic construction of deterministic algorithms: approximating packing integer programs*, Proc. 27th Ann. IEEE Symp. on Foundat. of Comput. Sci. (1986), 10–18.

26. Reif, J.H., Sen, S. *Optimal randomized parallel algorithms for computational geometry*, Proc. 16th Internat. Conf. Parallel Processing, St. Charles, IL, 1987. Full version, Duke Univ. Tech. Rept., CS–88–01, 1988.

27. Seidel, R. *Linear programming and convex hulls made easy,* Proc. 6th Ann. ACM Symp. Comput. Geom. (1990), 211–215.

28. Spencer, J. *Ten lectures on the probabilistic method*, CBMS-NSF, SIAM, 1987.

29. Vapnik, V. N., Chervonenkis, A. Ya. *On the uniform convergence of relative frequencies of events to their probabilities,* Theory Probab. Appl. 16 (1971), 264–280.

30. Yap, C.K. *A geometric consistency theorem for a symbolic perturbation scheme,* Proc. 4th Ann. ACM Symp. Comput. Geom. (1988), 134–142.