Cutting Hyperplanes for Divide-and-Conquer*

Bernard Chazelle

Department of Computer Science, Princeton University,
Princeton, NJ 08544, USA

Abstract. Given n hyperplanes in E^d, a $(1/r)$-*cutting* is a collection of simplices with disjoint interiors, which together cover E^d and such that the interior of each simplex intersects at most n/r hyperplanes. We present a deterministic algorithm for computing a $(1/r)$-cutting of $O(r^d)$ size in $O(nr^{d-1})$ time. If we require the incidences between the hyperplanes and the simplices of the cutting to be provided, then the algorithm is optimal. Our method is based on a hierarchical construction of cuttings, which also provides a simple optimal data structure for locating a point in an arrangement of hyperplanes. We mention several other applications of our result, e.g., counting segment intersections, Hopcroft's line/point incidence problem, linear programming in fixed dimension.

1. Introduction

Let H be a set of n hyperplanes in E^d. A $(1/r)$-*cutting* for H is a collection of (possibly unbounded) d-dimensional closed simplices with disjoint interiors, which together cover E^d and such that the interior of each simplex intersects at most n/r hyperplanes [19]. Spurred by the works of Clarkson [8], [9] and Haussler and Welzl [17], cuttings have proven very successful in solving all kinds of geometric problems; see, e.g., [4], [8], [9], [11], [14], and [16]. 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. Predictably, the cardinality of a cutting (which we call its *size*) is a critical parameter. A simple probabilistic argument shows that a size of $O(r^d \log^d r)$ is achievable. Actually, with a little more effort, the size can be reduced to $O(r^d)$, which is optimal [5]. If, for each simplex of the cutting, we need to keep a list of all the hyperplanes crossing it (which often

is the case in practice), then $\Omega(nr^{d-1})$ can be easily shown to be 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 $O(r^2)$-size cuttings in two dimensions was given by Matoušek [18] and then improved by Agarwal [1]. Chazelle and Friedman [5] showed that $O(r^d)$-size cuttings are computable in polynomial time in any dimension d. Recently, Matoušek [20], [21], [19] came close to putting the whole question to rest by exhibiting several efficient algorithms for computing $(1/r)$-cuttings of size $O(r^d)$, provided that $r < n^{1-\delta}$, for any fixed $\delta > 0$. We bridge this gap by relaxing the restriction on r. Specifically, we show how to compute a $(1/r)$-cutting for H of size $O(r^d)$ in time $O(nr^{d-1})$, for any value of $r \leq n$. The running time is optimal if we require the lists of simplex/hyperplane incidences to be part of the output. If we do not, then a result of Matoušek [21] shows how to compute a $(1/r)$-cutting in $O(n \log r)$ time, for any $r \leq n^{1/(2d-1)}$. Extending his result to all values of r remains open.

Interestingly, our method provides a new, simpler proof of the existence of an $O(r^d)$-size cutting. 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 $O(r^{2d})$. However, 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 is risky. For example, it might lead to extra multiplicative factors of the form n^ε (for arbitrarily small fixed $\varepsilon > 0$) in the running times of cutting-based algorithms, e.g., [4], [8], and [9].

One possible interpretation of this problem is that composing cuttings is a "nonstable" computing process. If the size of a cutting is cr^d, for $c > 1$, then the factor c can be regarded as a multiplicative error term. The nonstable part of the composition process is that it amplifies the error. 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 we 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. This leads to a kind of cutting which can be refined by composition. Besides providing a simpler, more efficient cutting construction method, our approach can be used to derive:

1. A simple optimal algorithm for locating a point in an arrangement 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]. (In a recent development following this result, Matoušek [23] has further improved the time bound for the line/point incidence problem to $O(n^{4/3}2^{O(\log^* n)})$, by using a more complicated argument.) Our algorithm generalizes easily 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. [16] and Agarwal [2].

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 the latter category we prove a lemma which says, roughly, that ε-approximations can be used efficiently to estimate how many vertices of a hyperplane arrangement lie inside a query simplex. Interestingly, ε-nets are too weak to provide any kind of meaningful approximation for that quantity. In a different development, to be reported elsewhere, we have used the same vertex-count estimation lemma in the design of an optimal deterministic algorithm for computing convex hulls in any fixed dimension [3]. Thus, we suspect that the lemma will have other useful applications in computational geometry.

We introduce our notation and prove the vertex-count estimation lemma in Section 2. Then we turn to the cutting construction proper in Section 3, and conclude with applications in Section 4. We also include a result which, although unrelated to cuttings, illustrates some of the same ideas: this is a straightforward linear-time algorithm for linear programming with a fixed number of variables, which greatly simplifies the methods of [7], [12], and [24]. This result, which was obtained independently by Matoušek [22], is derived by direct derandomization of a probabilistic algorithm of Clarkson [10].

2. The Vertex-Count Estimator

Let H be a collection of n hyperplanes in E^d. For simplicity, we assume that the set H is in general position. Given a polytope (or a flat) P of any dimension between 1 and d, let $H_{|P}$ be the set of hyperplanes of H that intersect the relative interior of P but do not contain P. We denote by $v(H; P)$ the number of vertices in the portion of the arrangement of H in the relative interior of P. If P is of dimension $k < d$, the k-dimensional arrangement formed by H over the affine hull of P is understood. So, in particular, if P is a line segment, $v(H; P) = |H_{|P}|$.

The next definition is adapted from [28]. We say that a subset R of H is a $(1/r)$-*approximation* for H if, for any line segment e, the densities in R and H of the hyperplanes crossing e differ by less than $1/r$, or, formally,

$$\left| \frac{|H_{|e|}|}{|H|} - \frac{|R_{|e|}|}{|R|} \right| < \frac{1}{r}.$$

We show that, given any simplex s, a $(1/r)$-approximation can be used as an accurate estimator for $v(H; s)$. This is because the densities of the vertices formed by H and R that fall inside s differ by less than $1/r$. The following lemma can be interpreted as generalizing the characteristic property of a $(1/r)$-approximation.

Lemma 2.1 (Vertex-Count Estimation). *Let R be a $(1/r)$-approximation for a finite set H of hyperplanes in E^d. For any d-dimensional simplex s, we have*

$$\left| \frac{v(H; s)}{|H|^d} - \frac{v(R; s)}{|R|^d} \right| < \frac{1}{r}.$$

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 R. We prove by induction that, for $k = 1, \ldots, d$,

$$\left| \frac{v(H_{|F}; s \cap F)}{|H|^k} - \frac{v(R_{|F}; s \cap F)}{|R|^k} \right| < \frac{1}{r}. \tag{2.1}$$

The lemma follows by setting $k = d$. The case $k = 1$ follows directly from the definition of a $(1/r)$-approximation, so assume that $k > 1$. Let F_i ($1 \le i \le |R| - d + k$) be the intersection of F with a hyperplane of R (not containing F), and let L_j, for

$$1 \le j \le \binom{|H| - d + k}{k - 1},$$

be the line obtained by intersecting F with a $(k - 1)$-wise intersection of hyperplanes in H (not including those containing F). Since

$$\sum_j v(H_{|L_j}; s \cap L_j) = k v(H_{|F}; s \cap F)$$

and

$$\sum_j v(R_{|L_j}; s \cap L_j) = \sum_i v(H_{|F_i}; s \cap F_i),$$

summing together the inequalities obtained from the fact that R is a $(1/r)$-approximation,

$$\left| \frac{v(H_{|L_j}; s \cap L_j)}{|H|} - \frac{v(R_{|L_j}; s \cap L_j)}{|R|} \right| < \frac{1}{r},$$

yields

$$\left| \frac{k v(H_{|F}; s \cap F)}{|H|} - \frac{1}{|R|} \sum_i v(H_{|F_i}; s \cap F_i) \right| < \frac{1}{r} \binom{|H| - d + k}{k - 1}.$$

By induction, we know from (2.1) that

$$\left| \frac{v(H_{|F_i}; s \cap F_i)}{|H|^{k-1}} - \frac{v(R_{|F_i}; s \cap F_i)}{|R|^{k-1}} \right| < \frac{1}{r},$$

and therefore

$$\left| \frac{kv(H_{|F}; s \cap F)}{|H|^k} - \frac{1}{|R|^k} \sum_i v(R_{|F_i}; s \cap F_i) \right| < \frac{|R| - d + k}{r|R|} + \frac{1}{r|H|^{k-1}} \binom{|H| - d + k}{k - 1}.$$

Since

$$\sum_i v(R_{|F_i}; s \cap F_i) = kv(R_{|F}; s \cap F),$$

we finally derive

$$\left| \frac{v(H_{|F}; s \cap F)}{|H|^k} - \frac{v(R_{|F}; s \cap F)}{|R|^k} \right| < \frac{|R| - d + k}{rk|R|} + \frac{1}{rk|H|^{k-1}} \binom{|H| - d + k}{k - 1} \leq \frac{2}{rk} \leq \frac{1}{r},$$

which establishes (2.1). $\qquad\qquad\square$

It is useful to restate the lemma in a slightly more general form. Consider a simplex s of any dimension $j \leq d$ and assume that s is not contained in any $h \in H$. If $j < d$, then the cross-sections of the hyperplanes of R by the affine hull F of s form a $(1/r)$-approximation for the j-dimensional arrangement in F induced by H. Note that by our general position assumption, at most d hyperplanes of H can contain s, and, therefore, what we just said remains basically true (modulo a lower-order term).

Corollary 2.2. *Let H be a set of n hyperplanes in E^d for which we have available a $(1/r)$-approximation for H of size $r^{O(1)}$. Given an arbitrary simplex s of dimension $j \leq d$, let $\mathscr{A}(H, s)$ denote the j-dimensional arrangement induced by H over the affine hull of s. Then the number of vertices of $\mathscr{A}(H, s)$ within the relative interior of s can be estimated in time $r^{O(1)}$ with an absolute error less than n^j/r.*

3. Refining Cuttings by Composition

The main idea here is to use a finer measure for 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.

Once again, we assume that the set H of n hyperplanes in E^d is in general position. This can be relaxed without difficulty by using the perturbation techniques of [15] and [29]. A subset R of H is called a $(1/r)$-net for H if, for any line segment e, the inequality $|H_{|e}| > n/r$ implies that $|R_{|e}| > 0$. A $(1/r)$-net plays the same intersection detection role as a $(1/r)$-approximation, but it is not nearly as powerful: in particular, it cannot be used as a vertex-count estimator. A beautiful result of Matoušek [20] says that, if r is a constant, 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 d-dimensional simplex s is not too large. We say that a $(1/r)$-net R is *sparse* for s if

$$\frac{v(R;s)}{v(H;s)} \leq 4\left(\frac{|R|}{|H|}\right)^d.$$

Lemma 3.1 [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 3.2. *Given a collection H of n hyperplanes in E^d and a d-dimensional simplex s, it is possible to compute a $(1/r)$-net sparse for 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 $(1/r)$-net sparse for s with nonzero probability.[1] We can obviously assume that $\rho < n$. The expected value $E[v(R;s)]$ is

$$v(H;s)\binom{n-d}{\rho-d}\bigg/\binom{n}{\rho} \leq \left(\frac{\rho}{n}\right)^d v(H;s),$$

therefore, by Markov's inequality,

$$\text{Prob}\left[v(R;s) > 4\left(\frac{\rho}{n}\right)^d v(H;s)\right] < \frac{E[v(R;s)]}{4(\rho/n)^d v(H;s)} \leq 1/4.$$

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

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

---

[1] All logarithms are to the base 2.

Since the number of segments in S is $O(n^{2d})$, for n large enough, we have $\sum_{e \in S} p_e < 1/4$, and, therefore,

$$\text{Prob}\left[v(R; s) > 4\left(\frac{\rho}{n}\right)^d v(H; s)\right] + \sum_{e \in S} p_e < 1/2.$$

It follows that a random R is a $(1/r)$-net sparse for s with probability greater than $1/2$.

To convert this existence proof into a polynomial-time algorithm, we use Raghavan's and Spencer's method of conditional probabilities [25], [27] (see also [5] and [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 $(1/r)$-net sparse for s. Let R' be a partial pick; the next sample element h is chosen in $H \backslash R'$ so as to minimize the value of

$$\frac{E[v(R; s) | R' \cup \{h\}]}{4(\rho/n)^d v(H; s)} + \sum_{e \in S} (p_e | R' \cup \{h\}),$$

where the expectation of $v(R; s)$ and the probabilities p_e are conditioned upon including $R' \cup \{h\}$ as part of the sample.

Each conditional probability p_e can be expressed by means of binomial coefficients and thus can be computed effectively. Note that as in [5] and [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 $\frac{1}{2}$. 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 the conditional expectations of $v(R; s)$ in polynomial time. When R' reaches size ρ, all the conditions for a sparse $(1/r)$-net are met and we can set $R = R'$.   □

We use the two previous lemmas to compute a $(1/r)$-cutting for H of size $O(r^d)$. Because a $(1/r)$-cutting is also a $(1/r')$-cutting for any $r' < r$, we can assume that r exceeds some appropriate constant r_0. For $k = 0, \ldots, \lceil \log_{r_0} r \rceil$, 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 $k = 0$, we simply view E^d as one unbounded simplex, which forms C_0. For $k > 0$, 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}; we assume that $H_{|s}$ is explicitly available for each s. We apply the following procedure for each s. If $|H_{|s}| \le n/r_0^k$, then s stands as such. Otherwise, calling on Lemmas 3.1 and 3.2, we compute first a $(1/2d\rho_0)$-approximation A for $H_{|s}$ and then a $(1/2d\rho_0)$-net R for A that is sparse for s, where

$$\rho_0 = \frac{r_0^k}{n} |H_{|s}|.$$

We can ensure that $|A| = O(\rho_0^2 \log \rho_0)$ and $|R| = O(\rho_0 \log \rho_0)$. Finally, we compute the arrangement formed by R and the $d + 1$ hyperplanes bounding s and form its

canonical triangulation [5], [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 $(k = 0, \ldots, d - 2)$ toward a chosen vertex (except for the simplices partitioning 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 $d \times |H_{|s}|/(d\rho_0) = n/r_0^k$ 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 subtle. We can check that $\rho_0 \leq r_0$, and, therefore, $|C_k| = O(|C_{k-1}|(r_0 \log r_0)^d)$, but this upper bound is too crude to do us any good. We can approximate $v(H; s)$ fairly accurately on the basis of $v(A; s)$ alone (Lemma 2.1). This, in turn, yields an upper bound on $v(R; s)$, and, hence, on $|C_k|$. We have

$$\left| \frac{v(H_{|s}; s)}{|H_{|s}|^d} - \frac{v(A; s)}{|A|^d} \right| < \frac{1}{2d\rho_0},$$

and, because of the sparsity of R,

$$\frac{v(R; s)}{v(A; s)} \leq 4\left(\frac{|R|}{|A|}\right)^d.$$

This implies

$$v(R; s) < 4\left(\frac{|R|}{|H_{|s}|}\right)^d v(H_{|s}; s) + \frac{2|R|^d}{d\rho_0}. \tag{3.1}$$

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. Consider the portion of the

arrangement of R within s. Every face is incident upon at least one vertex in the interior of s or on the boundary of s. Since, prior to triangulation, no vertex is incident upon more than a constant number of faces, the facial complexity of the portion of the arrangement of R within s is $O(v(R; s) + |R|^{d-1})$. Note that triangulating the portion multiplies the facial complexity by at most a constant factor. It follows from (3.1) that the triangulation of s consists of a number of simplices at most proportional to

$$(\rho_0 \log \rho_0)^{d-1} + \left(\frac{\rho_0 \log \rho_0}{|H_{|s}|}\right)^d v(H_{|s}; s) + \rho_0^{d-1}(\log \rho_0)^d.$$

Since $\rho_0 \leq r_0$ we have

$$\frac{\rho_0 \log \rho_0}{|H_{|s}|} = \frac{r_0^k \log \rho_0}{n} \leq \frac{r_0^k \log r_0}{n}.$$

Now, because

$$\sum_{s \in C_{k-1}} v(H_{|s}; s) \leq \binom{n}{d},$$

we find that, for some constant $c > 0$ (independent of r_0) and $k > 0$,

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

Since $|C_0| = 1$, we easily prove by induction that if r_0 is a large enough constant, $|C_k| \leq r_0^{(k+1)d}$. 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? By virtue of Lemmas 3.1 and 3.2, for each s it takes $O(|H_{|s}|)$ 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, in order to pay for its subsequent refinement, we should charge it a cost proportional to the number of hyperplanes intersecting its interior. The running time is therefore on the order of

$$\sum_{0 \leq k \leq \lceil \log_{r_0} r \rceil} \left(\frac{n}{r_0^k}\right) |C_k| \leq n r_0^{k(d-1)+d} = O(nr^{d-1}),$$

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

Note that the proof would not work if the facial complexity along the boundary of s was, say, on the order of ρ_0^d, or if the size of R was a little bigger, e.g., ρ_0^2. The latter is the reason why we could not use ε-approximations directly but had to resort to sparse ε-nets.

Theorem 3.3. *Given a collection H of n hyperplanes in E^d, for any $r \leq n$ it is possible to compute a $(1/r)$-cutting for H of size $O(r^d)$ in time $O(nr^{d-1})$.*

4. Applications of the Cutting Construction

One of the most interesting features of Theorem 3.3 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. Since the cuttings have constant size, breaking up the problems is particularly easy. 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 discuss only four fairly representative cases and briefly sketch their solutions.

4.1. Point Location

This is not an application *per se*. It is the mere observation that the hierarchical sequence of cuttings C_k is by itself 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 finding any cell of the arrangement of H whose closure contains the point can be done efficiently. Set $r = n$ in the construction. By tracing the query point from cell to cell across C_0, C_1, C_2, etc., we eventually land in a simplex entirely contained in a cell. This simplex belongs to the last cutting. All we need to ensure is that the simplex in question is labeled with the name of the cell containing it. This is easy to do in $O(n^d)$ preprocessing and is left as an exercise. The storage requirement is $O(n^d)$ and the query time is $O(\log n)$. This solution is much simpler than the one given in [6] and the preprocessing is faster. (In fairness the solution in [6] also supports unidirectional ray-shooting, which this one does not.)

Theorem 4.1. *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 any point can be located 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 still detect whether a 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 4.2. *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 it lies on at least one of the hyperplanes can be determined in $O(\log n)$ time.*

4.2. 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.* [14] 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 further 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. (We can also tailor the solution to the case of n hyperplanes and $m \neq n$ points, and 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 4.2.

Theorem 4.3. *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 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 4.2 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. Aside from the storage for the input, the space requirement is dominated by the storage of a single point-location structure, which is easily seen to be $O(n)$. ☐

Remark. In a recent development following this result, Matoušek [23] has further improved the time bound for the line/point incidence problem to $O(n^{4/3}2^{O(\log^* n)})$, by using a more complicated argument.

4.3. 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.* [16] 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 4.4. *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., nonlong) 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 4.2, 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 triangle (using the hierarchy of cuttings).

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 $O(m^{4/3}(\log m)^{1.78})$ time. 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 triangle Δ into $\lceil m/(\sqrt{n} \log n) \rceil$ subtriangles, each containing at most $2\sqrt{n} \log n$ endpoints, and solving the problem separately in each subtriangle. This gives a solution requiring $O(n \log n + m\sqrt{n})$ time and $O(m + n \log n)$ space.

To solve our original problem, we specialize the setting used in the proof of Theorem 4.3 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 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. The total running time is

$$O\left(nr \log n + \sqrt{\frac{n}{r}} \sum_i m_i \right),$$

where the m_i's sum up to $2n$. Setting $r = n^{1/3}/(\log n)^{2/3}$ gives a running time of $O(n^{4/3}(\log n)^{1/3})$. 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.   □

4.4. 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 [12] independently reduced this dependency to a multiplicative factor of roughly 3^{d^2}. Still smaller dependencies can be achieved by allowing randomization [10], [26]. 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 deterministic algorithm for linear programming when the dimension is fixed. (Matoušek [22] has derived a similar linear programming algorithm independently.)

We assume that the reader is familiar with Clarkson's algorithm. As explained in [10] we can assume that the system is feasible. Our variant is only a slight modification of it, so we only give a rough sketch. We take a random sample R of the constraints and solve the problem directly (say, by using the simplex method). If the answer satisfies all the constraints, then we are done. Else, we can easily argue that the set V_1 of unsatisfied constraints contains at least one of the constraints defining the desired answer x^*. At this point, Clarkson resamples, but we do not 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 unsatisfied 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 unsatisfied 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 have the property that no $|V_i|$ exceeds n/c, for some arbitrarily large constant c. Thus, the running time $T(n)$ of the deterministic algorithm follows a recurrence of the form $T(n) = O(1)$, for $n = O(1)$, and

$$T(n) = T\left(b + \frac{n}{c}\right) + T\left(b + \frac{2n}{c}\right) + \cdots + T\left(b + \frac{dn}{c}\right) + O(n),$$

where b is a constant dependent on c. Setting c large enough gives $T(n) = O(n)$.

References

1. Agarwal, P. K. Partitioning arrangements of lines, I: An efficient deterministic algorithm, *Discrete Comput. Geom.* **5** (1990), 449–483.
2. Agarwal, P. K. Partitioning arrangements of lines, II: Applications, *Discrete Comput. Geom.* **5** (1990), 533–573.
3. Chazelle, B. An optimal convex hull algorithm in any fixed dimension, *Discrete Comput. Geom.*, to appear.
4. Chazelle, B., Edelsbrunner, H., Guibas, J. J., Sharir, M. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications, *Theoret. Comput. Sci.* **84** (1991), 77–105.
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, Technical Report CS-TR-333-91, Princeton University, June 1991.
7. Clarkson, K. L. Linear programming in $O(n3^{d^2})$ time, *Inform. Process. Lett.* **22** (1986), 21–24.
8. Clarkson, K. L. New applications of random sampling in computational geometry, *Discrete 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 Annu. Symp. on Foundations of Computer Science*, 1988, pp. 452–456.
11. Clarkson, K. L., Shor, P. W. Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* **4** (1989), 387–421.
12. Dyer, M. E. Linear time algorithms for two- and three-variable linear programs, *SIAM J. Comput.* **13** (1984), 31–45.
13. Edelsbrunner, H. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.
14. Edelsbrunner, H., Guibas, L. J., Hershberger, J., Seidel, R., Sharir, M., Snoeyink, J., Welzl, E. Implicitly representing arrangements of lines or segments, *Discrete Comput. Geom.* **4** (1989), 433–466.
15. 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.
16. Guibas, L. J., Overmars, M., Sharir, M. Counting and reporting intersections in arrangements of line segments, Technical Report 434, Dept. Computer Science, New York University, 1989.
17. Haussler, D., Welzl, E. Epsilon-nets and simplex range queries, *Discrete Comput. Geom.* **2** (1987), 127–151.
18. Matoušek, J. Construction of ε-nets, *Discrete Comput. Geom.* **5** (1990), 427–448.
19. Matoušek, J. Cutting hyperplane arrangements, *Discrete Comput. Geom.* **6** (1991), 385–406.
20. Matoušek, J. Approximations and optimal geometric divide-and-conquer, *Proc. 23rd Annu. ACM Symp. on Theory of Computing*, 1991, pp. 505–511.
21. Matoušek, J. Efficient partition trees, *Proc. 7th Annu. ACM Symp. on Computational Geometry*, 1991, pp. 1–9.
22. Matoušek, J. Private communication, 1991.
23. Matoušek, J. Range searching with efficient hierarchical cuttings, Technical Report KAM Series, Dept. Applied Mathematics, Charles University, 1992.
24. Megiddo, N. Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.* **31** (1984), 114–127.
25. Raghavan, P. Probabilistic construction of deterministic algorithms: approximating packing integer programs, *J. Comput. System Sci.* **37** (1988), 130–143.
26. Seidel, R. Linear programming and convex hulls made easy, *Proc. 6th Annu. ACM Symp. on Computational Geometry*, 1990, pp. 211–215.
27. Spencer, J. *Ten Lectures on the Probabilistic Method*, CBMS–NSF, SIAM, Philadelphia, PA, 1987.
28. 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.
29. Yap, C. K. A geometric consistency theorem for a symbolic perturbation scheme, *Proc. 4th Annu. ACM Symp. on Computational Geometry*, 1988, pp. 134–142.