RANDOMIZED GEOMETRIC ALGORITHMS

KENNETH L. CLARKSON
AT&T Bell Laboratories, 600 Mountain Avenue
Murray Hill, NJ 07974 USA

1 Introduction

Randomization has found many applications in computational and combinatorial geometry in recent years. For a variety of geometric problems, the resulting algorithms are simpler than those previously known, or asymptotically faster, or both. Randomization helps in building geometric subdivisions, and in building data structures to quickly answer questions about such subdivisions. Randomization gives a general way to "divide and conquer" geometric problems, and can be used in the setting of parallel as well as serial computation. There are several facts of combinatorial geometry that have simple probabilistic proofs. This paper will explore some of these ideas and applications. We will also briefly discuss connections with the theory of "learning" geometric concepts.

It is important to note that the main focus here is *not* on random input, such as a collection of points distributed uniformly and independently in a region. Combinatorial and algorithmic questions concerning such point sets, or problems in *stochastic geometry*[68], have an enormous literature. Here we will mainly look at algorithms that use a source of random numbers, and consider their performance for arbitrary input. We generally look at *Las Vegas* algorithms, whose output is guaranteed to be correct, and not *Monte Carlo* algorithms, whose output may be incorrect. (Monte Carlo algorithms generally return an estimate of some quantity, with a performance guarantee that the estimate is accurate within some range with high probability.)

It is worth mentioning at the outset that there is nothing magical about randomization: several algorithms described here have deterministic competitors, or can be "derandomized" to yield deterministic versions. (See the works of Matoušek, P. Agarwal, and Chazelle. [18, 14, 49, 50, 2]) While the resulting algorithms are more complicated, they often satisfy the same asymptotic performance guarantees as the randomized algorithms, without the need for random bits. We hope the reader will see, however, how useful a "random" point of view can be.

While the emphasis here is on theorems and proofs, several of the algorithms discussed have been implemented. (For example, the algorithms of §4,§7.1,§4.4

have been implemented, and the resulting programs seem to be quite acceptable in complexity and efficiency. Fortune's survey gives some rough results for one randomized incremental algorithm.[39]) We also emphasize general techniques, and not the specifics of applications; the closing remarks of §8 give some references for these.

1.1 Outline

This paper will consider two main ways to apply randomization: a randomized incremental approach to building geometric structures, and a randomized divide-and-conquer technique for geometric computations. The divide-and-conquer scheme is to use a random subset of the input to create small subproblems, solve them, and then merge the solutions to obtain a solution for the original problem. If the input comprises n sites (points, for example), and the random subset has size r, the subproblems will have size about n/r. The number of subproblems will be bounded by some function $m(r) > r$.

When divide-and-conquer is applied to a computational problem, ideally the problem is partitioned into equally-sized subproblems, so that no subproblem has size larger than n/r and the subproblem sizes add up to the input size. Much of this survey discusses how closely this ideal situation can be approached in a geometric setting.

Section 2 discusses the use of randomized divide-and-conquer for answering closest-point queries. It shows how to obtain $O(r)$ subproblems of size $O(\log r)n/r$;[1] in many instances, this is within $O(\log r)$ of the divide-and-conquer ideal: the total of the subproblem sizes is $O(n \log r)$. The section also discusses the Vapnik-Chervonenkis (VC) dimension, a combinatorial property closely linked to range query algorithms, PAC-learnability of geometric concepts, and statistics.

Section 3 provides sharper bounds for the analysis of algorithms using the probabilistic approach of §2; we see that on average, the subproblems have size $O(n/r)$, within a constant factor of the ideal. As an application, we describe a nearly optimal algorithm for finding the trapezoidal diagram of a simple polygon; this is a subdivision of the interior of the polygon into simple regions. Finally, the section mentions the generalization of the approach to finding the trapezoidal subdivision of the plane induced by lines, or by line segments that cross.

Section 4 describes an incremental approach to building geometric structures, as applied to convex hulls: maintaining the hull of a subset as more and more points are included in that subset. The section also discusses a simple approach to planar point location.

Section 5 gives some results of combinatorial geometry. One result is an upper bound on the number of incidences between a set of points and a set of

[1] Remember that $O(\log r)n/r$, $O((\log r)n/r)$, and $O(1)(\log r)n/r$ all denote the same set of functions. This survey will use the variations of this notation that seem clearest.

lines in the plane. The relevance of this result to the *Hough transform* technique of computer vision is noted, as are Monte Carlo algorithms for that technique. The section also gives results on ($\leq k$)-*regions*, bounds that are helpful in the following section.

Section 6 gives still more careful results on probabilistic divide-and-conquer; here we see that the average of the *squares* of the subproblem sizes is $O(n/r)^2$, so that subproblems are small quite uniformly. The section also gives a construction of *cuttings*, giving subproblems that are *all* of size $O(n/r)$.

Section 7 describes two applications of *reweighting* techniques: linear programming in small dimension, and a data structure for triangular range queries.

Section 8 gives some references for applications.

Appendix A summarizes some simple facts of probability and geometry, and some useful inequalities; these are applied many times throughout. Appendix B gives a list of the common features of various geometric problems considered in this paper. These features allow similar arguments to be applied in proving expected bounds for algorithms.

2 Probabilistic divide-and-conquer

2.1 Closest-point queries

Just how can randomization help in geometric computations? Let's look at *closest-point problems.* Let S be a set of n points in the plane; we'll call the points of S *sites* for convenience. One closest-point problem:

> Construct a data structure for S so that given a query point p, a site closest to p in Euclidean distance can be found quickly.

Of course, there is the following obvious approach, which we'll dignify as **Algorithm O**; its "data structure" is just a list of sites.

- compute the distance to p from each site;
- return a site for which the computed distance is smallest.

This algorithm is simple, and often the best approach, but can be too slow if S is large and we need to answer many queries.

To beat the obvious approach, we will use a subset R of S to limit the possibilities for the closest site, following a simple observation: if we know the site q of R nearest to a query point p, then we know that the closest site in S to p must be at least as close as q. In terms of the "locus" approach to geometric problems [34], we have the open disk $C_R(p)$, which is bounded by a circle with center p and passing through q. Either q itself is a closest site, or a closest site to p is in the (open) disk $C_R(p)$.

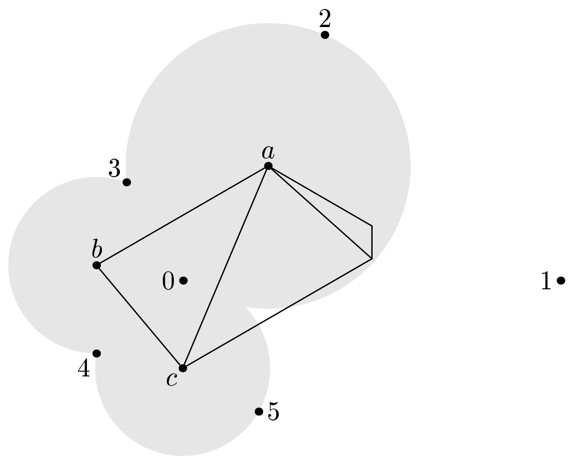

Figure 1: A triangle T with vertices a, b and c, and $C_R(T)$.

Now for a region T in the plane, let $C_R(T)$ be the union of the disks $C_R(s)$ for all $s \in T$. If the query point p falls in T, a closest site to p must be in $C_R(T)$, or in R.

Suppose Δ is a subdivision of the plane into simple regions. Suppose that for Δ and $R \subset S$, we have lists of the "candidate" set of sites $S \cap C_R(T)$ for all regions $T \in \Delta$. (We ignore here the details of this preprocessing.) We are ready to answer queries, as follows:

- for a query point p, find the region $T \in \Delta$ containing p;
- apply Algorithm **O** to p and the candidate sites $S \cap C_R(T)$;
- apply Algorithm **O** to p and sites of R;
- pick the closer of the sites in R and $S \cap C_R(T)$.

If the number of regions in Δ, the number of candidate sites for p, and the number of sites in R are all small relative to the size of S, we have answered the query faster than Algorithm **O**.

How can we choose R and Δ to get a speedup, for any query point? Conveniently, there is a subdivision $\Delta(\mathcal{V}(R))$ into triangular regions with the property that for $T \in \Delta(\mathcal{V}(R))$, the region $C_R(T)$ is the union of the open disks $C_R(v)$, v a vertex of T. Moreover, these three disks are *Delaunay* disks of R: they contain no sites of R, and each has a bounding circle that passes through three sites of R. (Here we ignore unbounded regions, but similar conditions apply to them as well. For those interested, the subdivision $\Delta(\mathcal{V}(R))$ is a triangulation of the Voronoi diagram of R: see Fortune's survey.[39]) Figure 1 gives an example with

R comprising the six numbered sites. The figure shows three triangular regions of $\Delta(\mathcal{V}(R))$, including a region T with vertices a, b, and c. The shaded area is $C_R(T)$. Note that $C_R(T) = C_R(a) \cup C_R(b) \cup C_R(c)$.

If R is a subset of size r, then $\Delta(\mathcal{V}(R))$ will have $O(r)$ regions; a value for r is discussed below.

The randomization at last comes in: we choose R to be a random subset of S, where all subsets of size r are equally likely to be chosen. With this choice of R, we're very likely to get candidate sets that are all small.

Why is this statement plausible? Suppose some arbitrary open disk D contains a large proportion of the sites. If R is a large random subset, we are very likely to have some site of R in D, and so D is unlikely to be a Delaunay disk of R. Roughly, the fact that Delaunay disks of R contain no sites of R is evidence that they contain few sites of S.

Here is a more precise statement.

Lemma 2.1 *Let S be a set of points in the plane, and R a random subset of S of size r. With probability at least $1 - O(1/n^2)$, every candidate set $S \cap C_R(T)$ for $T \in \Delta(\mathcal{V}(R))$ contains $O(\log n)n/r$ sites. More precisely, if $r > 5$ then with probability at least $1 - e^{-K/2+3\ln r}$, the random subset R is* good*: every Delaunay disk of R contains no more than Kn/r sites.*

To apply this lemma to obtain a good subset R, choose r large enough that $8 \ln r < r$ and $K = 8 \ln r$. Then choose R at random and check if every Delaunay disk of R contains no more than Kn/r sites. If R is not good, we try again. Each attempt fails with probability less than $1/r$, so in k attempts it fails with probability only $1/r^k$.

If we take $r = \sqrt{n}$, queries can be answered in $O(\sqrt{n} \log n)$ time, since the number of regions in $\Delta(\mathcal{V}(R))$ is $O(r)$. We could also make r a sufficiently large constant, and recursively apply this scheme to the candidate sets of $\Delta(\mathcal{V}(R))$ to obtain a search tree, with a search time that is $O(\log n)$.

Proof:[of Lemma 2.1] Two observations make the proof, as discussed below:

- the number of disks that are potentially Delaunay disks of a subset is not too large (only polynomially large);
- the probability that a given disk that contains more than Kn/r sites will be a Delaunay disk of a random subset is exponentially small.

More specifically, let $\mathcal{F}_S$ be the set of open disks bounded by circles inscribed on three sites. (A circle is *inscribed* on three points if it passes through them. There can be only one inscribed circle if the points are distinct.) There are at most $\binom{n}{3} = n!/3!(n-3)! = n(n-1)(n-2)/6$ such disks. Note that the Delaunay disks of any $R \subset S$ are contained in $\mathcal{F}_S$, since Delaunay disks are inscribed on three sites of R.

Suppose some open disk $D \in \mathcal{F}_S$ contains j sites. What is the probability that D will be a Delaunay disk of R? There are $\binom{n}{r} = n!/r!(n-r)!$ choices for R, but D will be a Delaunay disk if and only if the three sites on bounding circle of D are chosen, and if none of the j sites in D are chosen. That is, there are only $\binom{n-j-3}{r-3}$ ways to pick R so that D is a Delaunay disk of R, and so the probability is

$$\binom{n-j-3}{r-3} \Big/ \binom{n}{r}. \tag{1}$$

Using some standard manipulations that we omit (but which are very similar to (9) of §6), probability (1) is no more than $(1-j/(n-3))^{r-3}\binom{r}{3}\Big/\binom{n}{3}$; since $1+x < e^x$ for all x, this is no more than

$$e^{-j(r-3)/(n-3)}\binom{r}{3}\Big/\binom{n}{3}. \tag{2}$$

(It is instructive to consider the corresponding probability for a sample (subset) "taken with replacement." Here each of the r sample sites $\hat{R} = \{x_1 \ldots, x_r\}$ is chosen independently at random, with each site in S having equal probability to be x_i for each i. The probability is about r/n that a given site is chosen to be in the sample $\hat{R}$, so the probability that the three sites defining a given disk $D \in \mathcal{F}_S$ are chosen is about $(r/n)^3$. The probability that no sites in D are chosen is $(1-j/n)^r$, since the chance that no site in D is chosen as x_1 is $1-j/n$, and the same must hold true for each of the independently chosen x_i.)

Now to put the two observations together: we want to bound the probability that the random subset R is not good. This is the event that some disk $D \in \mathcal{F}_S$ containing at least $j \geq Kn/r$ sites is a Delaunay disk of R. The probability of the union of a set of events is no more than the sum of the probabilities of those events. From (2), the probability of one of the events is less than $e^{-j(r-3)/(n-3)}\binom{r}{3}/\binom{n}{3}$.

The number of events is no more than the number of disks in $\mathcal{F}_S$ containing more than Kn/r sites, which is no more than $\binom{n}{3}$. Thus the probability that R is not good is no more than

$$\binom{n}{3}e^{-j(r-3)/(n-3)}\binom{r}{3}\Big/\binom{n}{3} = e^{-j(r-3)/(n-3)}\binom{r}{3} < e^{-K/2+3\ln r},$$

for $j > Kn/r$. ■

This approach to closest-point problems is due to Clarkson.[24]

The proof of Lemma 2.1 didn't depend on anything particular to closest-point queries, or really on anything geometric. For example, we can generalize this argument to the more general ϵ-transversal problem for range spaces. A *range space* (U, X) is an underlying set U and a collection X of subsets of U. The members of X are the ranges; sometimes we refer only to X and leave U

implicit. An *ϵ-transversal* is a subset $R \subset U$ that has nonempty intersection with every range $x \in X$ with $\epsilon|X| > |U|$. (Here $|X|$ and $|U|$ denote the sizes of these sets.) For given ϵ, how small is an ϵ-transversal of X? Conversely, given r, for how small an ϵ is there an ϵ-transversal of size r? The proof above suggests the following lemma, whose proof is omitted.

Lemma 2.2 *If (U, X) is a range space and $K > 0$, then with probability at least $1 - e^{-K/2+\ln|X|}$ a random subset of U of size r is a K/r-transversal.*

This lemma implies the first statement of Lemma 2.1: the underlying set is S and the ranges are the subsets of the form $S \cap D$ for all $D \in \mathcal{F}_S$. A K/r-transversal R has the property that no Delaunay disk of R contains more than Kn/r sites.

An ϵ-transversal is sometimes called a *cover* of the ranges it meets, or a *puncture set*. The lemma, and related bounds, yield several combinatorial results proved using the "probabilistic method," as noted by Spencer.[76, 38] Even in the one-dimensional case, Lemma 2.1 has interesting implications; Reischuk used a special case to analyze a parallel algorithm for sorting.[67] The use of a random subset as a cover is also discussed in the context of network problems.[6] Another application, to estimating the Hough transform, is mentioned in §5.1.

Delaunay disks have the useful property that every disk containing no sites of R is contained in the union of three Delaunay disks of R. (Strictly speaking, this holds only if we include as "Delaunay disks" the empty halfplanes of R, as defined in §4.1.) Thus if R is good in the sense of Lemma 2.1, *any* disk containing no sites of R contains few sites of S. An ϵ-transversal R is thus a 3ϵ-transversal of the range space with underlying set S and with ranges of the form $S \cap D$, where D is *any* open disk. (More commonly ϵ-transversals are called ϵ-nets in papers on geometric algorithms, but this sometimes makes for confusion with the earlier idea of ϵ-nets in metric spaces, and so we follow Blumer *et al.*[8] in using a different term.)

Next we'll consider ϵ-transversals in greater generality, and see a general condition that implies that random subsets are ϵ-transversals with high probability.

2.2 The Vapnik-Chervonenkis dimension

The *Vapnik-Chervonenkis dimension*[79, 80] is a property of range spaces; it has a fundamental relation to some questions about algorithms, learning, and statistics. In this subsection, we'll briefly note some of these relations.

Call a range space (U, X) *polynomial* of degree k if $|X| \le |U|^k$. Lemma 2.2 immediately implies that if (U, X) is polynomial of degree k, then a random subset $R \subset U$ of size r is a K/r-net with probability $1 - e^{-K/2+k\ln|U|}$, and so (U, X) has $3(\ln|U|)k/r$-transversals of size r.

If S is a set of n sites in the plane, the range space $(S, \mathcal{F}_S)$ of §2.1 is polynomial of degree 3. Again, this follows from the fact that any disk in $\mathcal{F}_S$ is "defined by" three sites: any such disk is bounded by a circle inscribed on three sites of S. Thus Lemma 2.1 can be generalized to range spaces where the ranges are defined by some small number b of elements of U. We'll say that such range spaces have *specification complexity* b; they are polynomial of degree b and so have small ϵ-transversals. Appendix B gives several more examples.

There are more general conditions, however, for range spaces to have small ϵ-transversals. If (U, X) is a range space, for $U' \subset U$ define the *subspace induced by* U' as the range space (U', X') with

$$X' = \{x \cap U' \mid x \in X.\}$$

If U is infinite, and there is some constant k' so that every subspace induced by $U' \subset U$ with finite $|U'| > k'$ is polynomial of degree k, say that (U, X) has *polynomial discrimination* of degree k. That is, the ranges in X don't break up any subset of U into too many pieces. It follows from the paragraph before last that if (U, X) has polynomial discrimination of degree k, then any subspace has small ϵ-transversals.

The range spaces with polynomial discrimination are precisely those with bounded *Vapnik-Chervonenkis dimension*, or VC dimension for short. The VC dimension of (U, X) is the maximum size of a subset $U'' \subset U$ such that the subspace induced by U'' is *complete*: its ranges comprise all subsets of U''. If no such maximum exists, the VC dimension is unbounded, or infinite.

Many classes of geometric regions have bounded VC dimension: disks in the plane, balls in E^d, halfspaces in E^d, regions bounded by polynomial curves of bounded degree, axis-oriented rectangular regions, and many others. (See Blumer *et al.*[8] for references.)

We quote without proof the following theorem, which gives explicitly and sharply the relation between polynomial discrimination and VC dimension. (This result is due to Sauer and to Perles and Shelah;[69, 75] see also Bollabás's text[11], where the term *trace* is used for what we call the VC dimension.)

Theorem 2.3 *Let (U', X') be range space with $|U'|$ finite. If $|X'| > \sum_{0 \leq i < k} \binom{|U'|}{i}$, then for some U'' of size at least k, the subspace of (U', X') induced by U'' is complete.*

If (U, X) has bounded VC dimension d, then so does any subspace (U', X'), and from the theorem, $|X'| < |U'|^d$ if $|U'|$ is finite, and so (U, X) has polynomial discrimination of degree d. Thus if a range space has bounded VC dimension, all its subspaces have small ϵ-transversals.

2.2.1 Range queries

A key algorithmic problem associated with range spaces is that of processing *range queries*:

Given a set $S \subset U$ of size n, build a data structure so that for any range $A \in X$, the set $A \cap S$ can be found quickly.

Here the query range A is given by some compact description: a halfspace might be given as the equation of its bounding plane. Sometimes the answer can be reported as the union of some predefined subsets, so that the reporting cost is independent of the answer size.

If X has unbounded VC dimension, then any subset of S might be an answer to a query; the time required to look up one answer out of the 2^n possible is $\Omega(n)$ by any reasonable cost measure, so there is no way to answer a query faster than by looking at each element of S and checking for containment in A.

On the other hand, if X has bounded VC dimension, then as Haussler and Welzl showed in pioneering work, queries can be answered more quickly than $\Omega(n)$, even if only $O(n)$ storage is allowed.[42] More recently, Welzl has shown that an answer can be expressed as the union of $O(n^{1-1/d})$ subsets of S, though an algorithm to find those subsets quickly for a given query is unknown in general.[82, 19] Section 7.2 discusses an approach to range queries that applies in a geometric setting.

2.2.2 PAC-learnability of geometric concepts

The VC dimension is intimately related to the general problem of *learning* in range spaces:

Suppose (U, X) is a range space, and a random subset $R = \{x_1, \ldots, x_r\}$ is chosen from U. Given $R \cap A$ for some unknown range $A \in X$, find a "good approximation" $A(R)$ to A: the hypothesis $A(R)$ attempts to account for the "concept" A.

In Figure 2, for example, the underlying set U is the set of points in a square, and the ranges are rectangular regions bounded by horizontal and vertical line segments. The range A is shaded, and the subset R is chosen uniformly at random from square U. The points in $A \cap R$ are shown with $+$'s, the points in $R \backslash A$ are shown with $-$'s, and the hypothesis $A(R)$, with its boundary rectangle shown, agrees with A on R. (Since there are uncountably many points in the square, we are not really examining a random subset of a finite set, and the proper setting is a probability distribution on U; to keep comfortably finite, imagine that U is the set of grid points $\{(i/m, j/m) \mid 1 \leq i, j \leq m\}$ for some very large m.)

One measure of the badness of $A(R)$ is the probability of misclassification of a random element of U: the probability that random x_{r+1} is in the symmetric difference $A(R) \oplus A$. A range space (U, X) is said to be PAC-learnable only if there is always a range $A(R)$ so that with high probability, this misclassification probability is small. (PAC stands for "Probably Approximately Correct.")

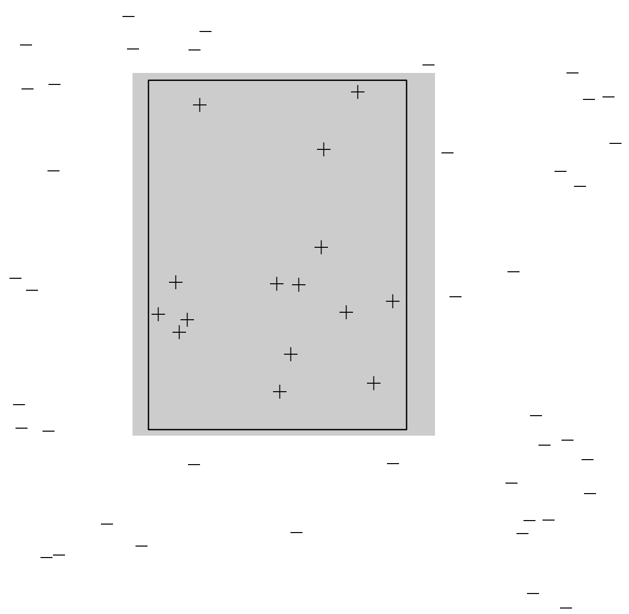

Figure 2: Sample points, the concept range A, and hypothesis $A(R)$.

A range space has bounded VC dimension if and only if it is PAC-learnable.[8] If the VC dimension d is bounded, a satisfactory hypothesis $A(R)$ is any range agreeing with the data, so that $A(R) \cap R = A \cap R$. If (U, X) has bounded VC dimension, then (U, X') does, where $X' = \{x \oplus y \mid x, y \in X\}$. If r is large enough, then with high probability R is an ϵ-transversal for X', with $\epsilon = O(\log r)d/r$. Hence the probability that random x_{r+1} is in $A(R) \oplus A$ is less than ϵ, since otherwise $R \cap (A(R) \oplus A)$ is not empty.

2.2.3 Conclusions on the VC dimension

In this discussion of the Vapnik-Chervonenkis dimension, we have seen that the property of bounded VC dimension is necessary and sufficient for a range space to have polynomial discrimination, ϵ-transversals, efficient range query structures, and PAC-learnability.

The bounds on the size of an ϵ-transversal given above can be sharpened;[42, 8] the best bound known is that a range space of VC dimension d has an ϵ-transversal of size $(d/\epsilon)\log(1/\epsilon)(1 + o(1))$ as $1/\epsilon \rightarrow \infty$.[47]

Pach and Wöginger have shown that there are range spaces of dimension d with ϵ-transversals of size $\Omega((1/\epsilon)\log(1/\epsilon))$.[62] Thus the $\log(1/\epsilon)$ factor is required in the ϵ-transversal size bounds for general range spaces. In contrast, point sets in E^3 have ϵ-transversals with respect to halfspaces of size $O(1/\epsilon)$, as do points in the plane with respect to disks, as can be shown using arguments similar to those of §6.2.[51] Moreover, the results of §3 and §6 show that the logarithmic factor is not needed for divide-and-conquer applications with range spaces of bounded specification complexity. Thus while the combinatorial condition of bounded VC dimension is elegant and fundamental, there remain geometric problems for which it is insufficient to yield the best results.

3 Sharper expected bounds for divide-and-conquer

3.1 Trapezoidal diagrams of simple polygons

We have seen one example of a divide-and-conquer approach to a geometric problem using a random subset: we get some $m(r)$ subproblems of size $n_1, \ldots, n_{m(r)}$, for some function $m(r)$ of the subset size r. In §2.1, the $m(r) = O(r)$ subproblems were all small when R was a good subset, with $n_i = O(\log r)n/r$ for all i, where n_i is the number of sites in the ith Delaunay disk of R. For some purposes, a uniform bound on the subproblem sizes is critical, but for others it is enough that the subproblems are small on average, so that the total $\sum_i n_i$ of the sizes of all the subproblems is small. This section shows that sharper expected bounds apply for this total than are implied by the uniform bounds alone. Specifically, while the high-probability bounds imply that the total of the subproblem sizes $\sum_{1 \leq i \leq m(r)} n_i$ is no more than $m(r)O(\log r)n/r$, we can get expected bounds on this sum like $m(r)O(1)n/r$.

With the logarithmic factor removed, optimal or near-optimal algorithms can be obtained for a variety of problems. The problem of computing *trapezoidal diagrams* of simple polygons is one example. Let S be a set of line segments that do not cross. The trapezoidal diagram of S is the planar subdivision $\mathcal{T}(S)$ with regions bounded by the segments, and also by all *visibility* segments. Figure 3 shows a trapezoidal diagram of five noncrossing segments. A visibility segment v is a vertical segment whose endpoints meet S, but otherwise does not meet S, and with one endpoint of v also an endpoint of a segment of S. We can think of the trapezoidal diagram as a collection of simple *cells*, or regions, that are often trapezoids. These cells also have adjacency relations that must be represented in an implementation: for now, two cells will be considered adjacent only if their boundaries share a common visibility segment. A representation of a trapezoidal diagram should include this visibility information. Note that each cell has at most four neighbors (under some non-degeneracy assumptions.)

The boundary of a simple polygon is a closed polygonal chain: a collection S of segments meeting at their endpoints, and only there. Many algorithmic problems involving simple polygons were long known to require $O(n)$ time for a polygon with n sides, given the trapezoidal diagram of the polygon. Algorithms were also long known that require $O(n \log n)$ time to find the trapezoidal diagram of a set of n noncrossing segments, and in particular a simple polygon.

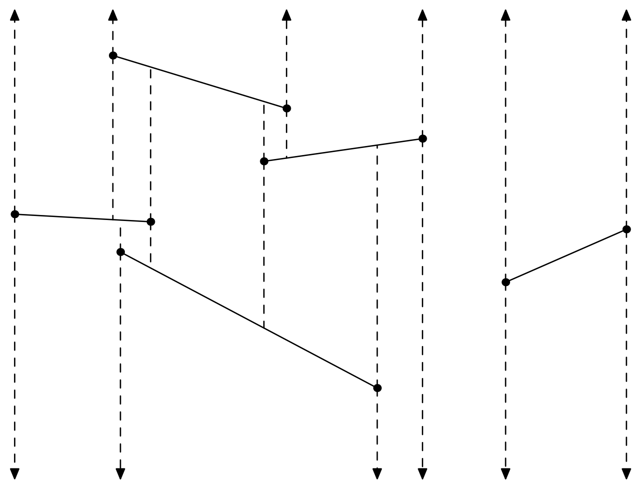

Figure 3: A set of five segments, and its trapezoidal diagram.

An outstanding open question was whether the diagram could itself be found in $O(n)$ time, given the polygon as a list of connected segments. Recently, Chazelle found such a linear-time algorithm.[15] While Chazelle's algorithm is optimal in the worst case, here we'll look at a slightly superlinear randomized algorithm that is much simpler than Chazelle's. (The algorithm is due to Clarkson, Cole, and Tarjan[26]; Seidel has given a similar algorithm.[72] The first algorithm with comparable performance is due to Clarkson, Tarjan and Van Wyk.[30]) The idea is to use randomized divide-and-conquer, with the general framework as follows:

- Compute $\mathcal{T}(R)$ for random $R \subset S$ of size r;
- insert the segments of S into $\mathcal{T}(R)$: that is, find the cells that meet each segment, and so the set of segments S_T that meet each cell $T \in \mathcal{T}(R)$;
- for each cell $T \in \mathcal{T}(R)$, compute the subdiagram $T \cap \mathcal{T}(S_T)$;
- merge appropriate cells of the subdiagrams to build cells of $\mathcal{T}(S)$.

Figure 4 shows a simple polygon and the trapezoidal diagram of a subset R comprising two segments, shown in bold.

Why does this approach produce the desired output? A visibility segment found in a subdiagram is a visibility segment of $\mathcal{T}(S)$, and no other visibility segments need be found, since such segments do not cross the segments of R. On the other hand, visibility segments of $\mathcal{T}(R)$ are not necessarily visibility segments of $\mathcal{T}(S)$, so we need to merge together some cells of subdiagrams in

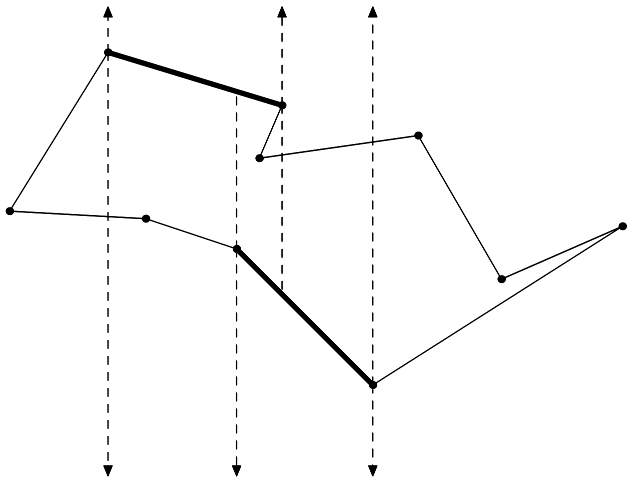

Figure 4: Using $\mathcal{T}(R)$ for divide-and-conquer.

the last step. If T and T' are cells of $\mathcal{T}(R)$ that share a visibility segment v, the cells of the subdiagrams of T and T' to merge are those that meet v. That is, there are (sorted) lists of cells along v in T and in T' that must be matched up and merged.

The next lemma gives several bounds for the sizes of the sets S_T. The general idea here, analogous to that for Lemma 2.1, is that a cell $T \in \mathcal{T}(R)$ meets no segments of R, and this is good evidence that it meets few segments of S. (We regard T as open, and we are not interested in segments that meet its boundary. Appendix B may help to fill out the details of the analogy here.)

Lemma 3.1 *Given a set S of n line segments, let $R \subset S$ be a random subset of S of size r, with all subsets of size r equally likely. Then:*

(i) *With probability* $1 - 1/n$, $\max_{T \in \mathcal{T}(S)} |S_T| = O(\log n)n/r$.

(ii) *The expected value of* $\sum_{T \in \mathcal{T}(S)} |S_T|$ *is* $O(n)$.

(iii) *The expected value of* $\sum_{T \in \mathcal{T}(S)} |S_T| \log |S_T|$ *is* $O(n) \log(n/r)$.

Proof: The tail estimate (i) can be proven with the same kind of argument as for Lemma 2.1: for a set of segments S, let $\mathcal{F}_S$ denote the set of regions that are cells of some diagram $\mathcal{T}(A)$, where $A \subset S$ has four or fewer segments. For any $R \subset S$, the cells of $\mathcal{T}(R)$ are in $\mathcal{F}_S$, which has size $O(n^4)$. For each region $T \in \mathcal{F}_S$ there is a set of segments that meet it: this set would be S_T if $T \in \mathcal{T}(R)$. If any segment in this set is in R, then T in not in $\mathcal{T}(R)$. Thus we want a K/r-transversal R of the range space (S, X), where $X = \{S_T \mid T \in \mathcal{F}_S\}$, and the number of ranges in X is $O(n^4)$. Lemma 2.2 applies, and this gives us the tail estimate (i). Part (ii) is given in §3.2, and part (iii) is proven at the beginning of §6. ■

Such results for general probabilistic divide-and-conquer were first proven by Clarkson and Shor.[29] See also the work of Reif and Sen.[65]

As a first step in applying this approach, suppose $r = |R| = n/\log n$. Computing $\mathcal{T}(R)$ then requires $O(n)$ time, using an algorithm needing $O(r \log r) = O(n)$ time. Each subdiagram can be computed in $O(|S_T| \log |S_T|)$ time, which from Lemma 3.1(iii) requires $O(n \log\log n)$ expected time overall. The insertion step, finding the sets S_T, can be done by walking along the bounding segments S, and simultaneously through $\mathcal{T}(R)$, recording intersections of segments and cells as the walk proceeds. It is not hard to show that the time required by the walk is proportional to the number of such intersections recorded, which by Lemma 3.1(ii) is $O(n)$. Finally, the merge step requires $O(|S_T|)$ for each T, since it amounts to merging two sorted lists; thus it requires $O(n)$ expected time overall. So with $r = n/\log n$, we have an algorithm requiring $O(n \log\log n)$ expected time to compute the trapezoidal diagram. We used the connectivity information among the segments of S to divide the problem into many small subproblems, and got leverage to reduce a factor of $\log n$ to $\log\log n$.

To get an algorithm that is even faster (at least asymptotically), we build a sequence of diagrams of random subsets

$$S^1 \subset S^2 \subset \cdots \subset S^{\log^* n} = S,$$

where $\log^* n$ is the smallest number k so that $\log^{(k)} n < 1$, and $\log^{(k)} n = \log\log^{(k-1)} n$ is the kth iterate of the logarithm function. Each subset S^i has size

$$r_i \approx n/\log^{(i)} n;$$

to find such a nested sequence of random subsets, imagine the segments of S to be randomly permuted, and take S^i as the first r_i segments in the permuted order. (For more on this, see Appendix A.)

We use $\mathcal{T}(S^{i-1})$ to compute $\mathcal{T}(S^i)$, in the same way that $\mathcal{T}(R)$ was used above to find $\mathcal{T}(S)$. The only difference in this computation is that in the insertion phase, we walk S to find the intersections with cells of $\mathcal{T}(S^{i-1})$, and then throw away most of the intersection information we get that way: we retain such information only for segments of S^i. The time required for the walk is still expected $O(n)$, however.

Since S^i is a bit smaller than S, so too are the sets $S^i_T = S_T \cap S^i$ for $T \in \mathcal{T}(S^{i-1})$; as a result, the computation of the subdiagrams is a bit faster. Since S^{i-1} is a random subset of S^i, Lemma 3.1 can be applied, with S^i taking on the role of S, and S^{i-1} the role of R. The result is that computing all the subdiagrams requires $O(n)$ total expected time.

Thus each phase of computing the diagram of a subset S^i requires expected $O(n)$ time. The total expected time required by the algorithm is $O(n \log^* n)$.

3.2 Proving the expected bounds

For now we'll only prove Lemma 3.1(ii).

Proof:[of Lemma 3.1(ii)] We want to show that a typical segment of $S \backslash R$ meets $O(1)$ cells. Since there are $n-r$ segments in $S \backslash R$, this implies the bound. To find such a typical segment, take a segment s at random from $S \backslash R$. Now the cells of $\mathcal{T}(R)$ that s meets are those in $\mathcal{T}(R) \backslash \mathcal{T}(R')$, where $R' = R \cup \{s\}$ and we view the diagrams as sets of cells. That is, we are interested in counting the number of cells that s "destroys" when added to R.

There is a helpful preservation of randomness here: since R is a random subset of S and s is a random element of $S \backslash R$, we know that R' is a random subset of S and s is a random element of R'. So we can think of these random choices taking place "backwards": first pick random $R' \subset S$, and then pick random $s \in R'$.

For finite sets A and B,

$$|A| + |B \backslash A| = A \cup B = |B| + |A \backslash B|,$$

so

$$|\mathcal{T}(R) \backslash \mathcal{T}(R')| = |\mathcal{T}(R)| - |\mathcal{T}(R')| + |\mathcal{T}(R') \backslash \mathcal{T}(R)|. \tag{3}$$

For a given $R' \subset S$, consider the expected values of these quantities for random $s \in R'$: by the linearity of expectation, we have

$$E|\mathcal{T}(R) \backslash \mathcal{T}(R')| = E|\mathcal{T}(R)| - |\mathcal{T}(R')| + E|\mathcal{T}(R') \backslash \mathcal{T}(R)|.$$

Here the expectation is with respect to the choice of s, which does not affect R'.

Consider the last quantity, $E|\mathcal{T}(R') \backslash \mathcal{T}(R)|$, the expected number of cells of $\mathcal{T}(R')$ deleted when random s is removed from R'. A cell T is removed from $\mathcal{T}(R')$ if s is one of the segments defining it; that is, s contains an upper or lower boundary segment of T, or an endpoint of s gives a visibility segment bounding T. At most four segments define T, so the probability that s is one of them is at most $4/(r+1)$. We have

$$E|\mathcal{T}(R') \backslash \mathcal{T}(R)| \leq 4|\mathcal{T}(R')|/(r+1).$$

Now we take expectations with respect to the random choice of R', to obtain

$$|E\mathcal{T}(R) \backslash \mathcal{T}(R')| \leq E|\mathcal{T}(R)| - E|\mathcal{T}(R')| + 4E|\mathcal{T}(R')|/(r+1).$$

(Again, with the random choice of R' and $s \in R'$, the subset R is a random subset of S.) We have reduced Lemma 3.1(ii) to the problem of accurately estimating the expected complexity of the trapezoidal diagram of a random subset of a given size. It is enough to consider sets of segments that don't meet at all, even at endpoints: if we perturb segments a tiny bit so that they no longer meet at endpoints, there is a new cell/segment intersection for every old one. (We are only looking at intersections of segments with *open* regions.) When S is a set of line segments with no two intersecting, the number of cells in $\mathcal{T}(R)$ for R any subset of S of size r is $3r+1$, as the industrious reader can show. This gives us

$$\begin{aligned}|E\mathcal{T}(R)\setminus\mathcal{T}(R')| &\leq (3r+1)-(3(r+1)+1)+4(3(r+1)+1)/(r+1)\\ &= 9+12/(r+1).\end{aligned}$$

So the expected number of cells met by a random member of $S\setminus R$ is $9+12/(r+1)$, and the expected total number of cell/segment intersections is at most $n-r$ times this. ∎

This analysis is based on that of Clarkson, Mehlhorn, and Seidel.[28] Seidel first showed the usefulness of the "backwards" style of analysis.[73]

By similar arguments applied to the visibility segments (rather than cells), it can be shown that the expected number of cell/segment intersections is (exactly) $5(n-r)$, and this holds for boundary segments of a simple polygon as well.

3.3 General segments and lines

While we have assumed that input line segments do not cross, trapezoidal diagrams can also be defined for those that do. Simply include visibility segments extending from intersection points, as well as from endpoints. The result is a subdivision into simple "trapezoidal" regions, as before, with similar results for divide-and-conquer using random subsets. If a set S of n segments has I pairs of intersecting segments, then the number of cells is $O(I+n)$, using the planarity of the graph of edges of the diagram and Euler's relation.

It is useful here to notice that the expected size of the trapezoidal diagram of a random subset is a bit smaller than you might think: a random subset R of size r has expected $I\binom{r}{2}/\binom{n}{2}$ intersecting pairs. Briefly, a given intersecting pair is in $\binom{n-2}{r-2}$ subsets, and so has $\binom{n-2}{r-2}/\binom{n}{r}=\binom{r}{2}/\binom{n}{2}$ probability of appearing in R. The expected number appearing in R is the sum of the probabilities that a given one appears and so is $I\binom{r}{2}/\binom{n}{2}$.

This fact implies, for example, that trapezoidal diagrams can be computed in $O(I+n\log n)$ expected time, as shown by Mulmuley and by Clarkson and Shor.[55, 58, 29] These algorithms use randomized incremental approaches akin to those discussed in the next section. (Chazelle and Edelsbrunner gave a deterministic algorithm with the same complexity.[16]) The existence of an algorithm with complexity linearly dependent on I was a longstanding open question.

Similarly, we can define trapezoidal diagrams for *arrangements* of lines, which are the subdivision of the plane induced by a set of lines. This subdivision is a collection of convex polygonal regions,and we carry over the terminology of vertices and edges in the obvious way. The visibility segments have the effect of partitioning the polygonal cells into trapezoids or triangles.

We can apply divide-and-conquer to build a search tree for locating points in a trapezoidal diagram by building the trapezoidal diagram of a random subset of a size r which is constant but “sufficiently large”; each cell will correspond to a child node of the root of the search tree. Now build the search tree recursively for the lines (or segments) meeting each cell. Each cell meets $O(\log n)n/r$ lines with high probability. (The proof is almost identical to that of Lemma 3.1(i).) Hence this construction will succeed, with high probability, in producing a search tree whose leaves correspond to cells contained in at most one cell of the trapezoidal diagram of the whole set. The space required is $O(n^{2+\epsilon})$ for trapezoidal diagrams of lines, where ϵ is a constant whose value decreases in r.

There are better ways to locate points in a trapezoidal diagram (see for example §4.4); however, this scheme can be generalized to higher dimensions. Several applications apply this kind of divide-and-conquer to higher dimensional problems.

4 Randomized incremental algorithms

While the divide-and-conquer technique was the main topic earlier, here we'll look at the different approach of building a geometric structure incrementally, making changes as needed.

For example, if we have the trapezoidal diagram $\mathcal{T}(S)$ of some set of segments S, it would be convenient given some new segment s to make only the necessary changes in $\mathcal{T}(S)$ to get $\mathcal{T}(S \cup \{s\})$. Similarly we might want to maintain the Voronoi diagram of a set of points, or its convex hull.

For some geometric structures, such as trapezoidal diagrams and planar Voronoi diagrams, the change in structure when including one new site is $\Omega(n)$ in the worst case, matching the complexity of the whole diagram up to a constant factor. Thus these update problems are not very interesting in a worst-case setting.

However, suppose that the updates are randomly chosen from some set; that is, we maintain $\mathcal{T}(R)$ for random $R \subset S$, and then update by inserting into R a random element of $S \setminus R$, or deleting a random element from R. These update operations preserve the randomness of R. Here (as we'll show) the insertions can be done fast enough that we have optimal algorithms for computing $\mathcal{T}(S)$.

If we are given the set S and want to insert all of its members, yielding $\mathcal{T}(S)$, we can ensure that these randomness assumptions are satisfied, since we simply repeatedly insert random elements of S. Put another way, in this *off-line* setting, we can pick a random permutation of S and insert into R based on that permutation.

The randomness assumptions are also satisfied if S is a collection of *random* points that are distributed independently and identically. Here the randomness is in the data, and we can compute $\mathcal{T}(R)$ on-line, without knowing S beforehand.

4.1 Convex hulls

This subsection gives a randomized incremental algorithm for maintaining convex hulls of point sets.

Before discussing the algorithm, a few words about convex hulls. (For more discussion and some motivation for computing them, see Fortune's survey.[39]) The convex hull $P = \mathcal{C}(S)$ of a set of sites (points) S is the "smallest" convex set that contains S: it is a subset of any convex set containing S. A set of points is convex if any line segment with endpoints in the set is contained in the set. In particular, halfspaces are convex, and $\mathcal{C}(S)$ is the intersection of all halfspaces containing S. We'll assume the sites (the points of S) are in "general position," so that no $d+1$ are coplanar when $S \subset E^d$. With sites in three-dimensional space, so $d = 3$, the boundary of $\mathcal{C}(S)$ is a collection of triangles; each triangle, called a *facet*, has an associated *supporting* plane containing it; the *supporting halfspace* of a facet is bounded by its supporting plane and contains $\mathcal{C}(S)$. In fact, $\mathcal{C}(S)$ is the intersection of the supporting halfspaces of its facets. The vertices of each facet are sites.

We'll call the open complement of a supporting halfspace of a facet an *empty halfspace* of S. Note that each empty halfspace of $S \subset E^3$ is "specified" by three sites contained in its bounding plane, the vertices of the corresponding facet. As the name should suggest, each empty halfspace contains no sites of S.

As in previous sections for Delaunay balls and cells in trapezoidal diagrams, useful bounds hold for the number of sites (of S) in the empty halfspaces of a random subset R of S. For example, on average such an empty halfspace contains $O(n/r)$ sites, for $|R| = r$, and with high probability every empty halfspace contains $O(\log r)n/r$ sites.

While we'll sometimes confuse $\mathcal{C}(R)$ with its associated set of empty halfspaces, a computer representation of $\mathcal{C}(R)$ should include the adjacency relations between facets: each facet will know the facets with which it shares all but one vertex.

Figure 5 shows a set of sites in the plane ($d = 2$), the boundary facets (edges) of its convex hull, and the line that bounds both a supporting halfspace (halfplane) and its corresponding empty halfspace.

Now consider the problem of maintaining the convex hull in three dimensions under insertions. Given a new site $s \in S \backslash R$, how is $\mathcal{C}(R')$ different from R, where $R' = R \cup \{s\}$? Some empty halfspaces of R are not empty halfspaces of R', since they contain s, and so some facets of $\mathcal{C}(R)$ are not facets of $\mathcal{C}(R')$. These facets are said to be *visible* to s (and s is visible to them), since line segments between s and such a facet are not "blocked" by $\mathcal{C}(R)$. On the other hand, R' may have some facets with s as a vertex, and so some empty halfspaces not in $\mathcal{C}(R)$.

Figure 6 shows a planar hull and a new site s that results in the deletion of several visible edges and the creation of two new ones.

There are two problems to be solved to obtain $\mathcal{C}(R')$: the *search* problem, to find the facets of $\mathcal{C}(R)$ visible to s, and the *update* problem, to delete visible

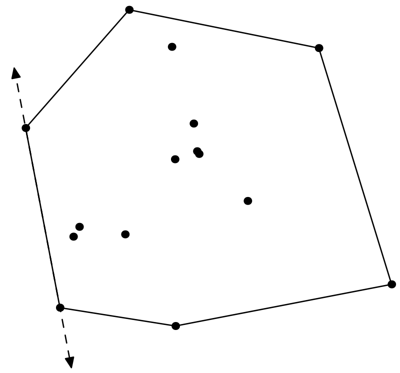

Figure 5: A set of sites in the plane and its hull.

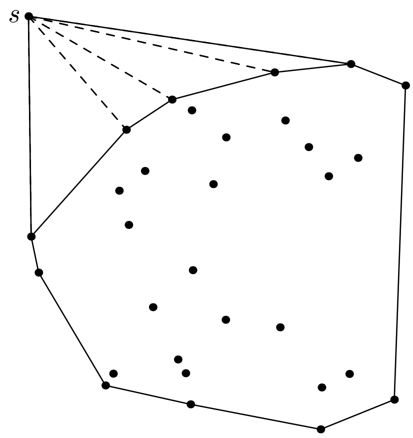

Figure 6: The effect of a new site s.

facets and determine the new facets with s as a vertex (*incident* to s). Note if s is inside the current hull, there will not be *any* facets visible to s, and no updating at all need be done.

4.1.1 Updates

While searching is the more expensive of the two operations, let's look at updating first, in the setting $d = 3$. As noted, in the worst case updating can take $\Omega(n)$ time: a good fraction of the facets of $\mathcal{C}(R)$ might be visible to s, and need to be deleted. However, when R is a random subset of S, then as in the proof of Lemma 3.1(ii), when $d = 3$ on average a point $s \in S \backslash R$ is visible to a constant number of facets of $\mathcal{C}(R)$.

The number of new facets incident to s is also expected to be constant. Moreover, each new facet incident to s has an edge (not containing s) that is contained in a facet visible to s, and also in another facet not visible to s; such *horizon* edges give all and only the new facets. Since each visible facet is adjacent to at most three facets that aren't visible, the time needed to create the new facets is proportional to the number of old facets.

It will be convenient for updating and searching to maintain a triangulation $\mathcal{H}$ of the convex hull: a collection of tetrahedra whose vertices are sites, whose union is the hull, and for which the intersection of two tetrahedra in the collection is either empty or a common face of the two. (Here a face is a vertex, edge, or triangular facet.)

The triangulation $\mathcal{H}$ will be maintained for insertions by including, for each visible facet, the tetrahedron with that facet, and also a vertex that is the inserted site. (If s is in the hull, no updating of the triangulation need be done.) Call the visible facet the *base* facet of the tetrahedron, and the inserted site its *peak* vertex. The facets of $\mathcal{C}(R') \backslash \mathcal{C}(R)$, incident to s, will be among the facets of the new tetrahedra.

We'll maintain adjacencies among these tetrahedra, so that each tetrahedron knows the four others with which it shares a facet. We can represent adjacencies for current facets by including tetrahedra with current "base" facets and with a "dummy" peak vertex that may later be replaced by a site visible to the base facet.

When fewer than $d+1$ sites have been inserted, we simply maintain R. When R has $d+1$ sites, we initialize $\mathcal{H}$ with an *origin* tetrahedron whose vertices are all sites, and with $d+1$ tetrahedra adjacent to the origin tetrahedron, which all have the dummy site as a peak vertex. It will be convenient to have the convention that the origin tetrahedron contains the coordinate origin, and has a peak vertex and base facet, though that facet was never a hull facet. Unlike every other base facet, the base facet of the origin tetrahedron will be (by convention) visible to sites on the *same* side of the facet as the origin.

4.1.2 Search

As with many known ways to search for the visible facets, here we'll look at a way to use visibility of old, deleted facets to find current visible ones. That is, after a site visible to a facet has been inserted, that facet is marked "deleted," but lives on as a base facet of a tetrahedron of $\mathcal{H}$, to help with searches for other sites.

How can deleted facets help with searches? First we must extend the notion of visibility to deleted facets: call a facet visible to a site, and vice versa, if that relation held when the facet was created. In other words, a facet is visible to a site if the site is in the halfspace that was the empty halfspace for the facet when it was created.

With this extended idea of visibility, our search procedure will find all visible facets, deleted as well as current: to visit a tetrahedron, check if its base facet is visible, and if so, visit tetrahedra adjacent to it. Start with the origin tetrahedron and the neighbor with which it shares a base facet.

Figure 7 shows the triangulation $\mathcal{H}$ for a set of 400 sites uniformly distributed in a disk. Not all the sites are shown, and the number of vertices of the triangulation is much smaller than 400, since most sites were in the hull when inserted. Base facets are shown as solid segments, and many have small triangles pointing toward their associated peak vertex. The remaining segments are dashed. (Note that many edges are base facets of one triangle they bound, though not the other.) The shaded triangles are some of those visited when searching for the site s.

Why does this procedure work? One argument is as follows: draw a line segment from a point in the origin tetrahedron to the new site. The base facet of every tetrahedron that meets that segment is visible to the new site, and the tetrahedra along the segment form a series of adjacent pairs, so the outermost tetrahedron, with a visible and current base facet, would be visited by the search procedure. Given that one current visible facet, the remaining such facets can be found using the tetrahedron adjacency information, since (as can be shown) the visible current facets form a connected set under the adjacency relation. (More exactly, in the undirected graph whose nodes are tetrahedra and with edges between adjacent tetrahedra, the visible current facets form a connected subgraph.)

4.2 Analysis of insertions

The time for inserting a new site is dominated, up to a constant factor, by the search time. This is proportional to the number of facets, deleted and current, visible to the new site. The space required by the triangulation is proportional to the total number of facets, deleted and current. We'll analyze the expected sizes of these quantities when the sites are inserted in the order $x_1, x_2, \ldots, x_n$, and this is a random permutation of S. This implies x_i is a random element

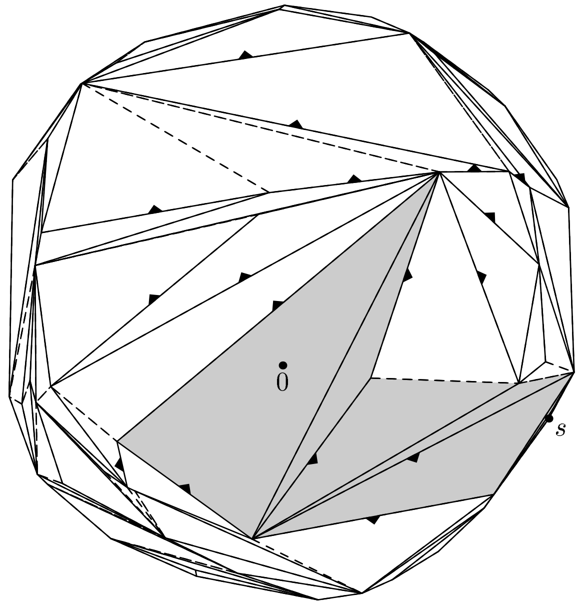

Figure 7: Some triangles visited in searching for a facet seen by site s.

of $S \backslash R_{i-1}$, inserted into $R_{i-1} = \{x_1, \ldots, x_{i-1}\}$, which is a random subset of S. We'll let $\mathcal{H}_i$ denote the triangulation $\mathcal{H}$ after i insertions.

The analysis is not much different when the dimension d is not equal to 3, so we'll put it in terms of d. The analysis uses the fact that facets have d vertices: in the terms of §2.2, they have a "specification complexity" of d. In d dimensions, the convex hull of $m+1 \leq d+1$ points is called an *m-simplex*. (Generally a d-simplex is called a simplex, omitting the d.) A 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron. With the general position assumption, the facets are $(d-1)$-simplices, and the triangulation comprises d-simplices.

The analysis will express the expected time and space requirements in terms of f_j, the expected number of facets of $\mathcal{C}(R_j)$. When $d = 3$ we have $f_j < 2r - 4$, and in general $f_j = O(j^{\lfloor d/2 \rfloor})$.[52, 12] We'll say that $f_d \equiv 2$ for convenience, and $f_i \equiv 0$ for $i < d$.

Theorem 4.1 *Let p_r be the expected size of the triangulation $\mathcal{H}_r$ of $\mathcal{C}(R_r)$. Then $p_r = \sum_{j \leq r} d f_j / j$.*

Proof: After $d+1$ points have been added, the triangulation $\mathcal{H}_{d+1}$ has $d+2$ simplices, as discussed above. For $j \geq d+2$, the base facets of $\mathcal{H}_j$ that are not base facets of $\mathcal{H}_{j-1}$ are those incident to x_j and facets of the hull of R_j.

Since each facet is incident to d vertices, the expected number of vertex-facet incidences is df_j, since R_j is a random subset of S. Since x_j is a random element of R_j, it is incident to an expected df_j/j facets. ■

In $d = 3$, $p_r = O(r)$, and in general the triangulation size is within a constant factor of the output size for "well-behaved" values f_j. As an example of a point set that is *not* well-behaved in this sense, pick a set with $f_j = \Omega(j)$, and then include the vertices of a tetrahedron containing the set. While the output has only four facets, the triangulation will have $\Omega(r)$ tetrahedra. On the other hand, for many natural probability distributions random data will have $f_j = O(j)$, or even $f_j = \log^{O(1)} j$, and the triangulations will be about the "right size."

Theorem 4.2 *The expected number of base facets of $\mathcal{H}_{r-1}$ that see x_r is*

$$-df_r/r + \sum_{j \leq r} d(d-1)f_j/j(j-1).$$

When $d = 3$ the search time is expected $O(1)\sum_{1 \leq j \leq r} 1/j = O(\log r)$. This means the expected total time to build $\mathcal{C}(S)$ is $O(n \log n)$, which is optimal for arbitrary point sets.

The proof shifts the analysis from estimating the number of facets visible to x_r, to one of estimating the number that would be been incident to x_r (with x_r as a vertex) if it had been inserted first. The latter is easy to estimate.

Proof: Let $\mathcal{H} = \mathcal{H}_{r-1} = \mathcal{H}(x_1, \ldots, x_{r-1})$ and $\mathcal{H}' = \mathcal{H}(x_r, x_1, \ldots, x_{r-1})$, that is, in $\mathcal{H}'$ we "pretend" that x_r was put in first. (Note that each simplex of a triangulation has exactly one corresponding base facet, so we'll confuse $\mathcal{H}$ and $\mathcal{H}'$ with the corresponding sets of base facets.)

Let X be the number of base facets of $\mathcal{H}_{r-1}$ that see x_r. Now $X = |\mathcal{H} \backslash \mathcal{H}'|$ since $\mathcal{H} \backslash \mathcal{H}'$ is the set of base facets in $\mathcal{H}$ which see x_r, and hence are not in $\mathcal{H}'$. We have

$$X = |\mathcal{H} \backslash \mathcal{H}'| = |\mathcal{H}| - |\mathcal{H}'| + |\mathcal{H}' \backslash \mathcal{H}|.$$

(Compare with (3) of §3.2.) By the linearity of expectation, and using Theorem 4.1,

$$EX = E|\mathcal{H} \backslash \mathcal{H}'| = p_{r-1} - p_r + E|\mathcal{H}' \backslash \mathcal{H}|, \tag{4}$$

averaging over the random choices of x_1 through x_r, and noting that $(x_r, x_1, \ldots, x_{r-1})$ is a random permutation of R_r.

It remains to estimate $E|\mathcal{H}' \backslash \mathcal{H}|$. What base facets are in $\mathcal{H}' \backslash \mathcal{H}$? This is a set of base facets incident to x_r: the collection of facets of the hull with x_r as a vertex, over the insertion sequence $x_r, x_1, \ldots, x_{r-1}$. To count these base facets, we count the expected number that appear when x_j is inserted, for $j \leq r$. That is, letting $R'_j = R_j \cup \{x_r\}$, for each base facet F of $\mathcal{H}'$, either F is a facet of$\mathcal{C}(R'_{d-1})$, or there is exactly one $j \geq d$ such that F is a facet of $\mathcal{C}(R'_j)$ and x_j is incident to F. A base facet in $\mathcal{H}' \backslash \mathcal{H}$ is also incident to x_r, and so for given j the number of base facets we count is the expected number of facets of $\mathcal{C}(R'_j)$

incident to x_r and x_j. Since each facet of $\mathcal{C}(R'_j)$ is incident to $\binom{d}{2}$ ordered pairs of vertices, the expected number of facets incident to a random pair of vertices in R'_j like x_r and x_j is $\binom{d}{2}E|\mathcal{C}(R'_j)|/\binom{j+1}{2}$. Hence,

$$E|\mathcal{H}'\backslash\mathcal{H}| = f_d + \sum_{d\leq j<r}\binom{d}{2}E|\mathcal{C}(R'_j)|/\binom{j+1}{2}.$$

Putting this expression for $E|\mathcal{H}'\backslash\mathcal{H}|$ with (4),

$$EX = p_{r-1} - p_r + f_d + \sum_{d<j\leq r-1} d(d-1)E|\mathcal{C}(R'_j)|/(j+1)j$$

Since R'_j is a random subset of S, $E|\mathcal{C}(R'_j)| = f_{j+1}$, and a bit of rearranging gives the result. ■

4.2.1 References

The above algorithm and analysis follows Clarkson, Mehlhorn, and Seidel[28], who also discussed tail estimates for the space required by the triangulation. Boissannatt and Teillaud gave an on-line algorithm for Delaunay triangulations;[10] Clarkson and Shor gave a similar off-line algorithm and analyzed its performance.[29] Independently Mulmuley described and analyzed an off-line randomized incremental algorithm for trapezoidal diagrams[55]. Subsequently Boissannat and others, and Guibas and others, gave on-line algorithms and analyzed them,[9, 40] and recently several papers have given on-line algorithms that allow deletion from R, with fast expected times if the deleted point is a random element of R.[71, 61, 59, 60, 28, 32] Klein has described a (greatly) generalized planar Voronoi diagram;[46] it can be computed in optimal expected time with a randomized incremental algorithm.[46, 54]

4.3 Other search schemes

4.3.1 Off-line algorithms

The approach of building a structure to help search is not the only way to find visible facets if the algorithm is off-line and looks at all of S at once. For example, we might maintain the visibility relation between all facets of $\mathcal{C}(R)$ and all sites in $S\backslash R$. With this information, the search problem is trivial. Notice that the expected total number of visible facet-site pairs can be estimated: the algorithm at any moment has the information that might be used with R for divide-and-conquer. The algorithm must maintain the visibility relation as facets in the hull are deleted and created. The work to eliminate a facet is proportional to the number of sites that see it; that is, we pay $O(1)$ for every facet that a site sees before the site is inserted. In other words, the total work for building $\mathcal{C}(S)$ by this algorithm is the same, up to a constant, as the total work for building $\mathcal{C}(S)$ by the on-line algorithm.

The relation between these on-line and off-line versions is the same as between quicksort and binary search trees: indeed, sorting can be reduced to a special case of trapezoidal diagram construction and planar convex hull construction; the off-line algorithm is a clumsy version of quicksort, and the on-line algorithm is a clumsy version of building a binary search tree.

The convex hull algorithm only needs one visible current facet to find the rest quickly; thus an off-line algorithm need only maintain for each site $s \in S \backslash R$ one visible facet v_s of the current hull. This gives an approach that is space-efficient in the worst case, unlike either the algorithm based on triangulations or the above off-line algorithms. There is an updating problem for this information: when v_s is deleted because some site t is inserted into R that also sees v_s, some facet seen by s and not by t must be found. This can be done by walking among the facets of the hull, starting at v_s, until one seen by s and not t is found. (If no such facet exists, s is inside the hull, and it can be disregarded.)

These two off-line algorithms were given by Clarkson and Shor.[29]

4.3.2 Avoiding search

Several algorithms use a randomized incremental approach, but obtain improved performance by avoiding the work that a general search procedure requires: the visibility information is obtained by other, faster means. (This may be true in a practical as well as theoretical setting: searching may be faster in practice with bucketing or quadtrees.)

For example, Seidel notes that the problem of finding a facet of $\mathcal{C}(R)$ visible to a given point can be reduced to small-dimensional linear programming, for which several $O(r)$ algorithms have been proposed.[73] (See §7.1.) This approach is faster for worst-case sites in $d > 3$ dimensions than the triangulation-searching method given here, and needs little space. (On the other hand, since for many distributions of random input we have $f_j = O(j)$, the expected search time of the above scheme is $O(\log r)$. On the other, other hand, Seidel's scheme is a bit simpler.)

In one of the earliest randomized incremental algorithms with provably good performance, Chew gave an algorithm for the Voronoi diagram of points that are vertices of a convex polygon.[20] That is, the points are in *convex position*, and their order along the polygon is given as input. Chew shows that with a small amount of auxiliary processing, the searches can be done in $O(1)$ time, giving a randomized incremental algorithm requiring $O(n)$ expected time.

Here is a sketch of his ideas, assuming some knowledge of Voronoi diagrams: the analog of site-facet visibility for Voronoi diagrams is site-Delaunay disk intersection. It turns out that two sites that are successive vertices of a polygon are Delaunay neighbors, and so have two incident Delaunay disks in common. This holds also for successive sites in a subset R, and so when a site is inserted between two successive vertices, the search need only examine the two disks incident to those vertices. Thus search is trivial if the neighbors of the site in R are known. This information can be obtained by running the insertions backward, maintaining polygon adjacencies as sites are deleted. A similar trick is helpful for maintaining convex hulls in higher dimension under deletions, when the sites are in convex position.[28]

The algorithm of §3.1 for trapezoidal diagrams of simple polygons can be viewed as using the connectivity information about segments to avoid search. Devillier has extended these observations to speed the computation of medial axis transforms and Delaunay triangulation for sites with a known minimum spanning tree.[31]

4.4 Trapezoidal diagrams and planar point location

Many of the above schemes have analogs for randomized incremental construction of trapezoidal diagrams; here the analog of facet-site visibility is cell-segment intersection. (Appendix B discusses this analogy in more detail.) An algorithm might maintain all such intersections; alternatively, for space efficiency, only the location of one endpoint in each uninserted segment (in $S \setminus R$) could be maintained. The remaining intersections could then be found by walking through the current diagram when inserting a segment.

An analog of the use of a triangulation for on-line maintenance is to keep old trapezoids around and use them for search. A cell that meets a new segment s is contained in the union of two new cells, each with s containing their upper or lower boundaries. Searching is aided by providing "influence" pointers from the deleted cell to each of the two new ones. If a point (say, a segment endpoint) is in that cell, then it is in one of the new ones as well. Searching entails tracing through the influence pointers from an initial cell to the current one containing the point. Searching is a means of finding the *location* of the point in the current diagram.

The time required is $O(1)$ per cell visited in this search process. As for convex hull searching, the expected search time for randomly chosen endpoints of segments in $S \setminus R$ is $O(\log r)$; this bound also holds for the expected number of cells, deleted and current, that meet a random uninserted segment.

What if we seek the location of an *arbitrary* point? Previous analysis techniques in this paper tell us little about this case. (At least, they say nothing directly, and don't provide the best bound.) However, bounds on the search time do hold for any given point: as discussed below, with probability $1 - 1/n^A$, for any fixed A, the search time is no more than $B \log n$ for some constant B. Here B is increasing in A. We'd like a bound that holds for *all* points, however. Fortunately there are only $n^{O(1)}$ distinct classes of points: two points are equivalent here if the same sequence of cells is visited for them while searching. Thus the probability that search time exceeds $B \log n$ for *any* point is $n^{O(1)}/n^A$. By making A and B sufficiently large constants, we have $O(\log n)$ search time for all points, with high probability.

Thus the search data structure for randomized incremental construction of trapezoidal diagrams solves the *planar point location* problem, with high probability: given a subdivision of the plane by noncrossing line segments, build a data structure so that given a point, its location in the subdivision can be found quickly. Moreover, the data structure can be maintained under random insertions and deletions with rapid execution of these operations on average.[28, 61, 71]

Given a triangulation of the plane, with its adjacency information, the data structure can be found in $O(n)$ expected time: by starting with the triangulation and maintaining the trapezoidal diagram of its segments under random deletions, we avoid searching but obtain the cells and pointers that an incremental algorithm would produce with random insertions; the cells to delete when deleting a segment can be found in constant time per cell, since we have the cell adjacencies, and finding the extensions of the visibility edges incident to a deleted segment is equivalent to merging sorted lists. Therefore the $O(n)$ cells in the data structure require $O(1)$ time per cell.

Theorem 4.3 *For any positive A, there is a constant B so that for any given point, the number of cells visited while searching through influence pointers is less than $B \log n$ with probability $1 - n^A$.*

Proof:[Sketch] Consider the construction of the data structure by deleting the segments of the triangulation in random order. The cell containing a given point changes only when one of the segments defining it is deleted. This occurs with probability at most $4/r$ when r segments remain. Thus the number of cells visited is the sum of n random variables, with variable $y_r = 1$ with probability $4/r$, and $y_r = 0$ otherwise. The expected value of this sum is $4H_n = \ln n + O(1)$, where $H_n = \sum_{1 \leq r \leq n} 1/r$. By arguments as for the Chebyshev bounds for binomial variates, the probability that the sum exceeds c times its mean is no more than $e^{H_n(1+c\ln(c/e))}$. These observations are due to Seidel.[72] (See also Reif and Sen's work.[65] Clarkson, Mehlhorn, and Seidel also give a proof.[28] ■

5 Some combinatorial bounds

In this section, we'll look at some questions of *combinatorial geometry*, which are counting problems regarding geometric objects. These problems can be attacked with some of the tools we have discussed; conversely, the answers to these problems can help us analyze some geometric algorithms.

5.1 Incidences

Given a set S of m points and a set L of n lines, how many incidences can there be between them? How large is

$$I(S, L) \equiv \{(p, l) \mid p \in S, l \in L, p \in l\},$$

the set of incident pairs? Let's look at some extreme cases. Suppose S comprises a single point. Then $|I(S, L)| \leq n$, and can be that large, if all the lines meet at that point. Suppose S contains at least n^2 points. Each line of L can contain no more than $n-1$ points of S that are also incident to other lines of L; therefore $|I(S, L)| \leq m + n(n-1)$. In the former case, the single point of S was incident to n lines. In the latter, an average point of S is incident to $1 + n(n-1)/m \leq 2$ lines. What happens in between? We seek a general bound on the quantity $I(m, n)$, the maximum size of $I(S, L)$ over all m-point sets S and n-line sets L. It is worth noting that $I(m, n) = I(n, m)$, using a duality map, since the dual line of a point of S and the dual point of a line of L are incident just when the point and line are. (For more on duality, see Appendix A, or for example Edelsbrunner's text.[34])

First we'll show that $I(m, n) < m + n\sqrt{m}$. Number the points of S from 1 to m. Let k_i denote the number of lines incident to point i. We can bound $\sum_i \binom{k_i}{2}$: the quantity $\binom{k_i}{2} = k_i(k_i - 1)/2$ counts the number of *pairs* of lines incident to point i, so the sum counts the number of incidences of points with pairs of lines. Since two distinct lines meet at no more than one point, a given pair contribute only one incidence; thus $\sum_i \binom{k_i}{2}$ is no more than $\binom{n}{2}$, or more loosely,

$$\sum_i (k_i - 1)^2 \leq n^2. \tag{5}$$

We can now invoke the Cauchy-Buniakowskii-Schwartz inequality, which implies

$$m \sum_i (k_i - 1)^2 \geq \left(\sum_i (k_i - 1) \right)^2 .$$

(That is, the dot product of the vectors $(k_1 - 1, \ldots, k_m - 1)$ and $(1, \ldots, 1)$ is no more than the product of the Euclidean norms of the two.) Hence from (5),

$$\sum_i (k_i - 1) \leq n\sqrt{m},$$

or $\sum_i k_i \leq m + n\sqrt{m}$.

Using projective duality, as discussed in Appendix A, we also have $I(m, n) = O(n + m\sqrt{n})$.

This kind of bound is due to Canham.[13] The bound fits our extreme cases $m = 1$ and $m = n^2$, but is it best possible? We'll prove the following theorem.

Theorem 5.1 *The maximum number of incidences between a set of m points and a set of n lines in the plane is $O((mn)^{2/3} + m + n)$.*

Proof: To get a better bound, divide-and-conquer the combinatorial problem: choose a random $R \subset L$ of size r, and consider the trapezoidal diagram $\mathcal{T}(R)$. A point of S is either incident to a line of R, in an open cell of $\mathcal{T}(R)$, or on a visibility segment, incident to no lines of L.

Let S' denote the points of S incident to lines of R. The number of incidences involving points of S' is no more than $m + 2nr$: such a point is either incident to a line of R and no others, or to a line of R and a line of L. Each line of L can contain no more than r points incident to a line of R. Hence the number of incidences involving the points of S' is no more than

$$m + 2nr. \tag{6}$$

Each point of $S \backslash S'$ is in some open cell of $\mathcal{T}(R)$. If cell $C \in \mathcal{T}(R)$ contains m_C points of S and meets n_C lines of L, then by the above Canham bound, the number of incidences between these points and lines is no more than $n_C + m_C\sqrt{n_C}$. That is, "charge" each point of $S \cap C$ a number of incidences $\sqrt{n_C}$, and the total charges will bound the number of incidences in C, up to n_C. Put another way, charge each point i of S with $\sqrt{n_i}$, the square root of the number of lines meeting the cell containing that point. The total of the charges $\sum_i \sqrt{n_i}$ will then bound the number of incidences involving points of $S \backslash S'$, up to $\sum_{C \in \mathcal{T}(R)} n_C$. The expected value of the latter sum is $O(n)$, using Theorem 3.1.

The next paragraph shows that the expectation of $\sum_i \sqrt{n_i} = O(m)\sqrt{n/r}$, so the expected number of incidences involving points in open cells of $\mathcal{T}(R)$ is $O(m)\sqrt{n/r} + O(n)$. (Since this expectation holds for a random subset, there must exist a subset for which the bound holds.) With (6), the total number of incidences is no more than

$$O(m)\sqrt{n/r} + O(n) + m + 2nr.$$

Choosing $r = (m^2/n)^{1/3}$ to balance these terms, this quantity is $O((mn)^{2/3} + m + n)$, as claimed.

To complete the proof, it suffices to show that for each point i, the expected value of $\sqrt{n_i}$ is $O(\sqrt{n/r})$. Again using the Cauchy-Buniakowskii-Schwartz inequality, one can show that $E\sqrt{n_i} \le \sqrt{En_i}$, and it is enough to bound En_i by $O(n/r)$. As in the proof of Theorem 3.1, consider the probability that a random line $l \in L \backslash R$ meets the cell $C \in \mathcal{T}(R)$ that contains p. This is the probability that the cell D of $\mathcal{T}(R')$ that contains p is *not* C. Here $R' = R \cup \{l\}$, and R' is a random subset of L. Now $D \in \mathcal{T}(R')$ is not in $\mathcal{T}(R)$ only if l is chosen from among the four (or fewer) lines defining the boundary of D. The probability of this is no more than $4/(r+1)$, and so the probability that random $l \in L \backslash R$ meets C is no more than $4/(r+1)$, and the expected number of lines of $L \backslash R$ meeting C is no more than $4(n-r)/(r+1)$, as needed. ∎

Besides the mathematical interest of these questions, why should anyone care about them? Here is one possible motivation: given a set S of m sites (points), we may want to find patterns in the sites. One sort of pattern is collinearity: many sites fall on some common straight line. For example, suppose we have the image of a black object against a white background. Many schemes are available to check the neighborhood of a point in the image to decide if it is on or near the black/white image boundary. Such schemes are not entirely reliable, though, so some way is needed to find a large collection of boundary points suggesting a boundary line. This kind of problem is closely related to the *Hough transform*, an approach computer vision researchers have used for finding straight boundary lines, or visual structure of similar kinds. Our bounds show that no more than $O(m^2/t^3)$ lines can each be incident to t points in a set of m points; this gives a bound on the size of the output of a structure-finding algorithm.

On the algorithmic side, it is worth noting that several computer vision researchers have suggested *Monte Carlo* algorithms for the Hough transform.[7, 44, 83] If a collinear subset of the sites has size t, then with high probability a random subset of $O(\log m)m/t$ sites will contain at least two sites of that subset, as Lemma 2.2 implies. Thus with high probability, the set of lines containing at least t sites is contained in the set of lines defined by pairs of sites in the random subset. This gives an algorithm for finding lines containing large collinear subsets; the algorithm is only *Monte Carlo*, however: as the above bound shows, no more than a $O(\log m)^2/t$ fraction of the reported lines will contain t sites, and some such lines may not be reported. However, fast randomized algorithms for this problem can be obtained using the techniques of this survey.

The approach of this subsection can be extended to other questions about arrangements, for example, the *many faces* question: what is the maximum possible total number of vertices of a set of m cells in an arrangement of n lines? This quantity $K(m,n)$ obeys a bound asymptotically similar to that of $I(m,n)$.

The discussion here follows the results in a series of papers[27, 36, 37, 35] which also discuss similar bounds for the total complexity of collections of faces in arrangements of curves and surfaces, and related algorithmic questions.

5.2 Bounds on $(\leq k)$-regions

A convex polygon in the plane with n vertices has n edges, and hence n empty halfplanes (as defined in §5). That is, there are n lines with the property that each line contains two vertices of the polygon, and the vertices not on the line are contained in a halfplane bounded by the line. A natural generalization of this question: how many distinct open halfplanes have the property that their bounding line contains two vertices, and they contain no more than k vertices, for some k? Such halfplanes are "within k" of being empty.

We might ask a similar question about Delaunay disks: how many disks can be within k sites of being Delaunay disks of a set of sites in the plane? Such *k-disks* are open disks bounded by circles inscribed on three sites, and contain no more than k sites.

As yet another example, consider trapezoidal diagrams of noncrossing segments: while these diagrams have $O(n)$ cells, how many k-cells do they have? Such a region is within k segments of being a cell: it is a cell of some diagram $\mathcal{T}(S')$ where $S' \subset S$ and has at least $n-k$ segments. That is, such a region T is a cell of some diagram $\mathcal{T}(A)$ for $A \subset S$ of size no more than four, and no more than k segments of S meet T.

In all these cases, we can use the bounds on the number of *0-regions*, which in the example above are, respectively, empty halfplanes, trapezoidal diagram cells, and Delaunay disks. With these bounds we can bound the number of ($\leq k$)-regions, which are "within k" of being 0-regions. For these structures we will show a bound of the form $f_{n/k}k^b$; here $f_{n/k}$ is the number of 0-regions of a random subset of size n/k, and $f_{n/k} = O(n/k)$ for the above structures. The parameter b is the "specification complexity" of 2, 4, and 3 for the examples of halfplanes, trapezoidal diagram cells, and Delaunay disks, respectively. Thus bounds of the form $O(n)k^{b-1}$, with b as given, hold for the number of ($\leq k$)-regions.

Before looking at proof techniques for this collection of problems, it is worth noting that the ($\leq k$)-halfplanes question is a special case of the many-faces problem for arrangements, mentioned at the end of the last subsection. Consider the ($\leq k$)-halfplanes problem, when stated in terms of the projective dual: we have a set of lines, the dual lines of the vertices, and look at the halfplanes that are bounded by these lines and contain the origin. The intersection of these halfplanes is a polygon that the dual of the original polygon. A line segment between the origin and a vertex of this dual polytope does not cross the dual lines, and we know that there are no more than n vertices of the arrangement of these lines with this property. For how many of the vertices of the arrangement does the corresponding line segment cross no more than k dual lines? Each such vertex corresponds to a ($\leq k$)-region of the original polygon. Thus we are bounding the complexity of a particular family of cells of the arrangement of the dual lines: those whose "crossing distance" from the origin is no more than k.

As an example, here is a proof of bounds for ($\leq k$)-halfplanes of point sets. We want to bound the maximum, over all sets S of n sites (points), of the number of ($\leq k$)-halfplanes of the sites. These open halfplanes contain no more than k sites, and are bounded by lines that contain two sites. (We will assume that no three points of the set are collinear.)

The basic idea for proving the bounds is to use the relationship of these halfplanes to the 0-halfplanes (empty halfplanes) of subsets. That is, we know that any subset of size r has no more than r associated 0-halfplanes, each halfplane corresponding to an edge of the convex hull of the subset. We will show that a

($\leq k$)-halfplane of S has a relatively high likelihood of being an empty halfplane of a random subset R of size $r = \lfloor n/k \rfloor$. Since there are at most r such empty halfplanes, we have a bound on the number of ($\leq k$)-halfplanes.

More precisely, suppose R is a random subset of S of size r, with all subsets of that size equally likely. We can write the expected number of empty halfplanes of R in the following way: let $\mathcal{F}_S$ denote the set of j-halfplanes of S, over all j (not only the set of ($\leq k$)-halfplanes); that is, $\mathcal{F}_S$ is the set of open halfplanes bounded by a line containing two points of S. For $H \in \mathcal{F}_S$, let I_H be a random variable that is 1 when H is an empty halfplane of R, and zero otherwise. Then the number of empty halfplanes of R is

$$\sum_{H \in \mathcal{F}_S} I_H,$$

and the expectation of this quantity is the expected number of 0-halfplanes of R. By the linearity of expectation, this is

$$\sum_{H \in \mathcal{F}_S} EI_H.$$

The expectation EI_H is the probability that H is an empty halfplane of R. Since R has at most r empty halfplanes, we know that

$$\sum_{H \in \mathcal{F}_S} \mathrm{Prob}\{H \in \mathcal{F}_R^0\} \leq r, \tag{7}$$

where $\mathcal{F}_R^0$ denotes the set of empty halfspaces of R.

Next we find an expression for the probability that some j-halfplane H is an empty halfplane of R. This event requires two conditions:

(i) The two sites on the line bounding H must be in R;

(ii) The j sites in H must *not* be in R.

Two points in R are fixed by condition (i) above, and so from among the $n-j-2$ sites that aren't in H or on its boundary, any $r-2$ can be chosen. That is, the number of subsets of S of size r that satisfy these conditions is $\binom{n-j-2}{r-2}$. Since there are $\binom{n}{r}$ subsets of S with size r, the probability that H appears as an empty halfspace of R is

$$\frac{\binom{r}{2}}{\binom{n}{2}} \frac{\binom{n-j-2}{r-2}}{\binom{n-2}{r-2}}. \tag{8}$$

The factor $\binom{r}{2}/\binom{n}{2}$ is the probability that the two sites on the boundary of H appear in R. The other factor expresses the probability that the j sites in H

are not in R, and we can bound it from below:

$$\begin{aligned}\frac{(n-j-2)!(r-2)!(n-r)!}{(n-j-r)!(r-2)!(n-2)!} &= \frac{(n-r)(n-r-1)\cdots(n-r+1-j)}{(n-2)(n-3)\cdots(n-j-1)} \\ &= \left(1-\frac{r-2}{n-2}\right)\left(1-\frac{r-2}{n-3}\right)\cdots\left(1-\frac{r-2}{n-j-1}\right) \\ &\geq \left(1-\frac{r-2}{n-j-1}\right)^j\end{aligned}$$

For $j \leq k$, this is at least $(1-(r-2)/(n-k-1))^k$, which for $r = \lceil n/(k+1) \rceil$ is at least

$$(1-1/(k+1))^k = \frac{1}{(1-1/k)^k} \geq 1/e,$$

using $1+x \leq e^x$. Substituting $1/e$ for the second term in (8), the probability that a j-halfplane H is an empty halfplane of R is at least $\binom{r}{2}/e\binom{n}{2}$.

Putting this bound with (7), if M is the number of ($\leq k$)-halfplanes of S, then

$$\frac{M\binom{r}{2}}{e\binom{n}{2}} \leq r,$$

or

$$M \leq \frac{en(n-1)}{r-1} \leq \frac{en(n-1)}{n/(k+1)-1} < 3n(k+1)$$

for $k < n/11$.

Theorem 5.2 *Any set of n points in the plane in general position (no three collinear) has no more than $3n(k+1)$ ($\leq k$)-halfplanes, for $k < n/11$.*

We can prove similar results for $k \geq n/11$. This bound is not tight: the maximum number of ($\leq k$)-halfplanes is $n(k+1)$, and this bound is tight[4]. However, our proof is easily generalized: the only thing here specific to planar convex hulls and halfplanes was the fact that two points specify a halfplane, via its bounding line. In other instances of this general problem, we simply substitute a different number b of specifiers, yielding a different power of k in the bound, and use the appropriate bound for the number of 0-regions.

For example, in higher dimensions, we have ($\leq k$)-halfspaces, where the 0-halfspaces are bounded by planes containing facets of the convex hull of the points. From the bound $O(n^{\lfloor d/2 \rfloor})$ for the number of facets of the convex hull of n points in d dimensions, we obtain $O((n/k)^{\lfloor d/2 \rfloor} k^d) = O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$ as a bound for the number of ($\leq k$)-halfspaces.

These results were given in excruciating generality by Clarkson and Shor;[29]; subsequently they were sharpened a bit by Mulmuley.[56, 57] The lower bound argument given here for (8) uses some arguments of Sharir[74], who gave several applications of such bounds.

6 Still sharper bounds for divide-and-conquer

This section bounds the distribution of subproblem sizes more carefully, using analogs of the bounds for ($\leq k$)-halfplanes. For example, with divide-and-conquer using a random subset, an average subproblem has a *squared* size that is $O(n/r)^2$. These bounds hold because the number of subproblems of size in/r falls off (almost) exponentially in i: about $1/e$ of the subproblems are of size less than about n/r, $1/e^2$ are of size less than $2n/r$ and greater than n/r, and so on. Here is a more precise statement of this claim.

Lemma 6.1 *Let S be a set of n noncrossing line segments in the plane, and R a random subset of S of size r. There is a constant K so that the expected number of j-cells of S which satisfy $i = \lfloor j(r-4)/(n-4) \rfloor$ and appear in $\mathcal{T}(R)$ is no more than $Kr(i+1)^3 e^{-i}$.*

Proof: Let $\mathcal{T}_i$ be the set of j-cells of S with $\lfloor j(r-4)/(n-4) \rfloor = i$. We want bound the expected number of cells of $\mathcal{T}_i$ that appear in $\mathcal{T}(R)$. We'll prove this only for the cells defined by exactly four segments; the proofs for those cells defined by fewer segments are analogous.

The probability that a given j-cell T of S appears as a 0-cell of R is

$$\frac{\binom{r}{4}}{\binom{n}{4}} \frac{\binom{n-j-4}{r-4}}{\binom{n-4}{r-4}},$$

similarly to (8); the factor on the left is the probability that the four segments defining T appear in R, and is no more than $(r/n)^4$. The factor on the right reflects the probability that the j segments meeting T are *not* in R. We bound the latter from above:

$$\begin{aligned}
\frac{(n-j-4)!(r-4)!(n-r)!}{(n-j-r)!(r-4)!(n-4)!} &= \frac{(n-j-4)(n-j-5)\cdots(n-j-3-r)}{(n-4)(n-3)\cdots(n-r-3)} \\
&= \left(1-\frac{j}{n-4}\right)\left(1-\frac{j}{n-5}\right)\cdots\left(1-\frac{j}{n-r-3}\right) \\
&\leq \left(1-\frac{j}{n-4}\right)^{r-4} < e^{-j(r-4)/(n-4)} \qquad (9)
\end{aligned}$$

Hence the probability that $T \in \mathcal{T}_i$ is a cell in $\mathcal{T}(R)$ is no more than $(r/n)^4 e^{-i+1}$. As in §5.2, the number of j-cells with $j \leq (i+1)(n-4)/(r-4)$ is $O(n((i+1)n/r)^3)$,so the expected number of cells of $\mathcal{T}_i$ in $\mathcal{T}(R)$ is

$$(r/n)^4 e^{-i+1} O(n((i+1)n/r)^3) = O(r)(i+1)^3 e^{-i}.$$

∎

As an application of this lemma, here is a proof of one of the results needed for the analysis of the algorithm of §3.1.

Proof:[of Lemma 3.1(iii).] We need to show that the expected value of $\sum_{T\in\mathcal{T}(S)} |S_T|\log|S_T|$ is $O(n)\log(n/r)$. A j-cell with $i = \lfloor jr/n \rfloor$ contributes no more than $((i+1)n/r)\log((i+1)n/r)$ to the sum; the expected number of such cells is no more than $Kr(i+1)^3e^{-i}$, by the lemma, and so the expected value of the sum is no more than

$$\sum_{i\geq 0}((i+1)n/r)\log((i+1)n/r)Kri^3e^{-i} = Kn\sum_{i\geq 0} e^{-i}i^3(i+1)\log((i+1)n/r),$$

which is $O(n)\log(n/r)$. ∎

Such a proof was given in generality by Clarkson[23].

6.1 Bounds for the sum of squares

As given below, similar bounds hold for trapezoidal diagrams of lines. The following subsection uses the sharper bounds to show how to make *cuttings*: subdivisions of a collection of lines into $O(r^2)$ trapezoidal (or triangular) cells so that for *every* cell, the number of segments meeting it is $O(n/r)$.

The following lemma has a proof analogous to the previous one.

Lemma 6.2 *There is a constant K so that, for a set S of n lines in the plane and R a random subset of S, the expected number of j-cells of S which satisfy $i = \lfloor jr/n \rfloor$ and appear in $\mathcal{T}(R)$ is no more than $Kr^2(i+1)^2e^{-i}$.*

This lemma readily implies the following theorem, for which we'll also supply another proof.

Theorem 6.3 *For a set S of n lines in the plane and R a random subset of S, the expectation of $\sum_{T\in\mathcal{T}(R)} |S_T|^2$ is $O(n^2)$.*

Proof: The previous lemma implies the bound; here is another proof giving a smaller constant factor. Let $\mathcal{F}_S$ be the set of all possible cells in $\mathcal{T}(R)$: the set of cells that are in $\mathcal{T}(A)$, for some $A \subset S$ with $|A| \leq 4$. Let I_T be an indicator function for the event that $T \in \mathcal{F}_S$ is in $\mathcal{T}(R)$. Then

$$\sum_{T\in\mathcal{T}(R)} \binom{|S_T|}{2} = \sum_{T\in\mathcal{F}(S)} I_T\binom{|S_T|}{2},$$

and using the linearity of expectation and the fact that $EI_T = \text{Prob}\{T \in \mathcal{T}(R)\}$,

$$E\sum_{T\in\mathcal{T}(R)} \binom{|S_T|}{2} = \sum_{T\in\mathcal{F}_S} \text{Prob}\{T \in \mathcal{T}(R)\}\binom{|S_T|}{2}.$$

Letting i_T denote the number of segments that define T, the probability that T is in $\mathcal{T}(R)$ is

$$\binom{n-|S_T|-i_T}{r-i_T}\Big/\binom{n}{r},$$

since out of the $\binom{n}{r}$ subsets, a subset must be picked that includes the $i_T \leq 4$ segments defining T, and for the remaining $r - i_T$ segments, there are only $n - |S_T| - i_T$ to choose from, since the $|S_T|$ meeting T cannot be picked. Thus the quantity to estimate is

$$\sum_{T \in \mathcal{F}_S} \binom{|S_T|}{2} \binom{n - |S_T| - i_T}{r - i_T} \Big/ \binom{n}{r},$$

which is no more than

$$2\frac{n^2}{r^2} \sum_{T \in \mathcal{F}_S} \binom{|S_T|}{2} \binom{n - |S_T| - i_T}{r - i_T - 2} \Big/ \binom{n}{r},$$

for large r and $n - r$. The trick here is that the summand is the probability that the cell T is a 2-cell of R. For T to be a 2-cell of R, the segments defining T and two of the segments meeting it must be in R; the remaining $r - i_T - 2$ segments of R must be chosen from the segments that neither meet T nor define it. The summand is the probability of this event. The sum is the expected number of 2-cells of R. The number of 2-cells of any set of lines of size r is $O(r^2)$, using a proof analogous to that of Theorem 5.2. The sum obeys this bound, and the theorem follows. ■

This proof follows that of Clarkson and Shor.[29]

6.2 Uniform bounds and cuttings

This section shows the existence of optimal *cuttings*: given a set S of n lines and a parameter r, a $(1/r)$-*cutting* is a subdivision into simple cells so that every cell meets no more than n/r lines. The methods of §2 imply only that the trapezoidal diagram of a random subset of S of size $O(r \log n)$ is a $(1/r)$-cutting with high probability; the number of cells is $O(r \log n)^2$. The methods of §3 show that, on average, a cell of a trapezoidal diagram of a random subset of $O(r)$ lines meets n/r lines, but still some cells may meet more.

Here we'll show that a $(1/r)$-cutting of size $O(r^2)$ can be built; this is asymptotically optimal. The idea is to start with the trapezoidal diagram of a random subset and recursively "repair" the cells which meet too many lines. By the results of the last subsection, the sizes of the repairs will be small. Such repairs have some resemblance to deletion methods of the "probabilistic method."[77]

Theorem 6.4 *There is a constant C so that given a set S of n lines and a parameter m, there is a $(1/m)$-cutting of S of size no more than Cm^2.*

Proof: Choose a random subset R of S of size $r = m\alpha\sqrt{C}$ and build its trapezoidal diagram, choosing α small enough that $\mathcal{T}(R)$ has no more than $Cm^2/4$ cells. For every cell T meeting $j > n/m$ lines, recursively subdivide T with a $1/\lceil jm/n \rceil$-cutting. This completes the construction.

The expected size of the resulting cutting is no more than

$$Cm^2/4 + \sum_{i>\alpha\sqrt{C}} (C(i+1)^2)Kr^2(i+1)^2 e^{-i+1} = Cm^2/4 + Cr^2 \sum_{i>\alpha\sqrt{C}} K(i+1)^4 e^{-i+1},$$

using Lemma 6.2. For sufficiently large C, the sum is less than $1/4\alpha\sqrt{C}$, so the expected size of the cutting is no more than $Cm^2/2$. By Markov's inequality, the probability that the cutting will have size greater than Cm^2 is no more than $1/2$, so with probability $1/2$ the construction succeeds in obtaining a cutting with no more than Cm^2 cells. Repeat until success, requiring two expected attempts. ■

In the literature, the cells of a cutting are triangles, so as a last step here, split the cells into triangles.

This construction follows ideas of Chazelle and Friedman.[18] Similar ideas shows that cuttings into simplices in higher dimensions can be found. A more recent construction, and a "derandomized" version, is described by Chazelle[14], who uses it for a worst-case optimal algorithm for computing convex hulls in higher dimensions.

7 Reweighting methods

This section gives two applications of a *reweighting* technique. The idea is to assign weights to input sites, so that the weight reflects the "importance" of the site in some way. By choosing random subsets where the probability of choosing a site is proportional to its weight, we can divide into subproblems with low weight, and hence importance.

In early applications, Welzl applied this approach to building range query data structures, and Littlestone applied it to learning.[82, 48]

We'll always use integral weights; in fact, the weights will be powers of 2. The weight of site s will be denoted by $w(s)$, and the total weight of a set of sites Q will be $w(Q) = \sum_{s\in Q} w(s)$. To take a random subset R using the weights, for each site s we conceptually make ordered pairs $(s,1),(s,2)\ldots,(s,w(s))$, yielding a resulting (unweighted) set S'. Now take a random subset R' of S' of size r, and then project: put in R all sites s with $(s,i) \in R'$, for some i. With this scheme, weighted versions of the divide-and-conquer theorems can be shown to follow.

7.1 Linear programming

In this section we'll look at an algorithm for the following problem:

> Given a set S of n points in d dimensions, and a line l, find the lower endpoint of $l \cap \mathcal{C}(S)$, the intersection of l with the convex hull of S.

Here "lower" means having the smallest x_1 coordinate. Perhaps surprisingly, this problem is linear programming, from the "column space geometry" point of view. (In a review article, Stone and Tovey look at simplex and interior point methods from this perspective.[78])

The algorithm will find the facet $T^*(S)$ of $\mathcal{C}(S)$ that meets l at its lower endpoint. The algorithm is interesting only when $n \gg d$. Here we can view d as a constant, and try to minimize the dependence of the running time on n.

The basis of the algorithm is the following simple observation:

> If $R \subset S$, the set $V(R)$ of points of $S \backslash R$ that see $T^*(R)$ must include at least one vertex of $T^*(S)$, unless $T^*(R) = T^*(S)$.

That is, at least one vertex of $T^*(S)$ must be in the empty halfspace of $T^*(R)$, below it. Otherwise, all vertices of $T^*(S)$ would be contained in the supporting halfspace H of $T^*(R)$, above it, and so would their convex hull, which is $T^*(S)$. This would imply that $l \cap T^*(S)$ is above $T^*(R)$, a contradiction.

To use this observation, we give all elements of S weights, that are all one at first. To mark the importance of $V(R)$, or at least one member of it, we double the weights of its elements when $w(V(R)) < 2dw(S)/r$. Here $r = 4d^2$ is the size of R. Naturally, we'll make R a random subset of S, using the weighted sampling scheme discussed above; this implies that the expected weight $w(V(R))$ is no more than $dw(S)/r$, as shown below.

In short, the main iteration of the algorithm is: take sample R, compute $T^*(R)$, and double the weights of elements of $V(R)$ when $w(V(R)) < 2dw(S)/r$. This is iterated until the current set $V(R)$ is empty, and we have $T^*(S)$. This is the entire algorithm, but for the problem of computing $T^*(R)$. Since $r = 4d^2$ and we're assuming $n \gg d$, we can use any algorithm we like, including computing the entire convex hull of R. Let $L(n, d)$ denote the time taken by this "brute force" algorithm to solve the problem for n sites in set S in d dimensions. From the upper bound theorem, we have $L(r, d) = L(4d^2, d)$ proportional to $\binom{4d^2}{d} < d^{O(d)}$.

Theorem 7.1 *The algorithm terminates in expected $O(d \log n)$ iterations, and so has expected running time $O(d^2 n \log n) + d^{O(d)}$.*

Proof: We'll show that in the unweighted case the expected size of $V(R)$ is $d(n-r)/(r+1)$. A similar bound $d(w(S) - r)/(r+1)$ holds for the weighted case, though that won't be proven here.

The proof is analogous to the last paragraph of the proof of Theorem 5.1. It is enough to show that a random element of $S \backslash R$ sees $T^*(R)$, and so is in $V(R)$, with probability $d/(r+1)$. If $s \in S \backslash R$ sees $T^*(R)$, then $T^*(R') \neq T^*(R)$, where $R' = R \cup \{s\}$. This occurs only if s is a vertex of $T^*(R')$. Since s is a random element of R', this occurs with probability $d/(r+1)$, as desired.

How many iterations will the algorithm need to "converge" to $T^*(S)$? First we bound the expected number of iterations, versus the number of "successful" ones when $w(V(R)) < 2dw(S)/r$ and the weights of elements of $V(R)$ are doubled. Since the expected weight $Ew(V(R)) < dw(S)/r$, the probability that $V(R)$ is doubled is at least $1/2$ for each iteration by Markov's inequality, and so the expected number of iterations is twice the number of successful ones.

After a successful iteration, the weight $w(S)$ increases by a factor no more than $1 + 2d/r = 1 + 1/2d$, and so is no more than $(1 + 1/2d)^k < e^{k/2d}$ after k successful iterations. On the other hand, by the critical observation above, some vertex of $T^*(S)$ is in $V(R)$, and so has its multiplicity doubled after a successful iteration, so after k successful iterations, the total weight $w(T^*(S))$ is $\sum_{1\leq j\leq d} 2^{i_j}$, for some integers $i_j \geq 1$ with $\sum_{1\leq j\leq d} i_j = k$. The sum is minimized when all $i_j = k/d$, so $w(T^*(S)) > d2^{k/d}$. Since $w(S)/w(T^*(S)) > 1$, the number k of successful iterations must satisfy $(ne^{k/2d})/(d2^{k/d}) > 1$, which implies $k = O(d \log n)$. ■

This algorithm is due to Clarkson;[22] an earlier, recursive, algorithm requiring $O(n)$ expected time can be combined with it, and the approach can be applied to integer programming. Adler and Shamir have shown that these algorithms can be applied to general convex programming.[1] Chazelle has noted that the recursive algorithm can be derandomized, giving a deterministic algorithm requiring $d^{O(d)}n$ time.[14] The first $O(n)$-time algorithms for $d = 3$ are due to Megiddo, and to Dyer.[53, 33]

Seidel gave a different randomized algorithm[73], requiring $O(d!n)$ expected time; recently Sharir and Welzl found a variant of Seidel's algorithm requiring time subexponential in d. Their algorithm is a randomized instance of the simplex algorithm.[81] Kalai was the first to find a subexponential simplex algorithm.[43] Problem instances have long been known for which versions of the simplex algorithm require at least 2^d operations.[45] These results cast new light on the complexity of the simplex algorithm, and on the possibility that linear programming problems can be solved in "strongly polynomial" time, with $(nd)^{O(1)}$ operations, and with the number of operations independent of the size of the numbers specifying a problem instance. (For more on these issues, see for example Schrijver's textbook.[70])

7.2 Triangular partitions

This subsection gives a data structure for answering halfplane queries: given a set S of n points, the resulting data structure requires $O(n)$ space; given a halfplane H, the data structure can be used to find the size of $S \cap H$ in $\sqrt{n}(\log n)^{O(1)}$ time. The data structure uses a *triangular partition* of S: a collection Π of pairs (S_i, t_i), with $1 \leq i \leq m$, where t_i is a triangular region and $S_i \subset S \cap t_i$. Also $|S_i| < 2n/m$ and the S_i form a partition of S. That is, Π splits S into roughly equally sized pieces, and each piece is contained in a triangle. An important property of such a partition is its *crossing number*: the maximum, over all lines, of the number of triangles of Π cut by the line. Matoušek, extending ideas of Welzl, proved the following remarkable fact:[50]

Theorem 7.2 *For any parameter $r \leq n$ and set of points S, there is a triangular partition Π of S of size $m = O(r)$ and with crossing number $O(\sqrt{r})$.*

To apply this to halfplane queries, build a *partition tree* by recording the size of S_i for each t_i, and recursively applying the construction to the sets S_i. The total size of the resulting tree is clearly $O(n)$. To answer a query H, check for each i if $t_i \subset H$; if so, add $|S_i|$ to the answer. If t_i is crossed by the boundary line of H, recursively find $|S_i \cap H|$ and add that quantity to the answer. By choosing $r = \sqrt{n}$, the query time is $\sqrt{n}(\log n)^{O(1)}$.

The proof of the theorem will need two lemmas. The first lemma says that for the purpose of building a triangular partition, we can use a small *test set* of lines to check the crossing number property of a partition.

Lemma 7.3 *Given a set S of n points in the plane, there is a set Q of lines such that for every triangular partition Π of S satisfying $|S_i| \geq n/r$ for all i, the following holds: if κ_Q is the crossing number of Π relative to Q, the crossing number of Π is no more than $3\kappa_Q + O(\sqrt{r})$.*

Proof:[sketch] Construct a cutting (§6.2) of the dual lines of the points of S. A suitable set Q is the dual of the vertices of the triangles of that cutting. We omit the proof that Q satisfies the conditions of the lemma. ■

The next lemma says that S has a large subset with a triangular partition.

Lemma 7.4 *Given a set S of n points and parameter $r \leq n$, there is a subset W of S at least half the size of S, so that W has a triangular partition Π with sets W_i of size $\lfloor n/r \rfloor$ for all i, and with crossing number $O(\sqrt{r})$.*

Given this lemma, a triangular partition for the whole set S, as promised by Theorem 7.2, is built as follows: apply the lemma to $S^0 = S$ with parameter $r_0 = r$, yielding a triangular partition for subset W^0. Then apply the lemma to $S^1 = S \backslash W^0$ with parameter $r_1 = r/2$, and so on $j = O(\log r)$ times, until S^j has fewer than n/r points. A single triangle containing S^j, and all the triangles built along the way, completes the construction.

Proof: [of Lemma 7.4.] The partition is built triangle-by-triangle. To make the first pair (S_1, t_1), build a $1/t$-cutting of the test set Q for S, where t is chosen so that the cutting has no more than $r/2$ triangles; by Theorem 6.4, a value of $t = \Theta(\sqrt{r})$ will suffice. Since S has n points, some triangle of the cutting contains at least $\lfloor n/r \rfloor$ points. Shrink that triangle until it contains exactly that many points; those points and the triangle are S_1 and t_1 for the cutting.

So far, so good: each line of the test set cuts only one triangle, because there *is* only one triangle. The application of reweighting here is to make sure that the lines cutting that triangle are less likely to cut future triangles. To do this, we double the weight of the lines meeting t_1, and repeat the above procedure with $S \backslash S_1$. The fact that, by construction, no more than $O(\sqrt{r})$ lines of Q cut

t_1, means that we are not greatly altering the conditions of the next cutting for the lines that do not meet t_1.

In the general step, we have pairs $(S_1, t_1), \ldots, (S_{i-1}, t_{i-1})$. If the total of the size of these S_j's is more than $n/2$, we're done. Otherwise, we want to find (S_i, t_i).

Each line $l \in Q$ has a weight $w_i(l) = 2^{\kappa_{i-1}(l)}$, where $\kappa_{i-1}(l)$ is the number of triangles $t_1, \ldots, t_{i-1}$ crossed by l. Using a variant of the cutting construction for these weighted lines, we get a $(1/t)$-cutting for Q with these weights, again so that the cutting has $r/2$ triangles, and some triangle t contains at least $\lfloor n/r \rfloor$ points of $S \setminus \cup_{1 \leq j < i} S_j$. We shrink that triangle until it contains exactly $\lfloor n/r \rfloor$ points, and call the result t_i. The weights of the lines cutting t_i are doubled, and the construction of (S_i, t_i) is complete.

The triangle t_i is cut by lines of weight no more than $w_i(Q)/t$. Therefore doubling makes $w_{i+1}(Q)$ no more than $(1 + 1/t)w_i(Q)$.

When the construction is done we have $w_{m+1}(Q) = \sum_{l \in Q} 2^{\kappa(l)}$, where $\kappa(l)$ is the crossing number of l for the triangular partition. By construction, $w_{m+1}(Q) \leq (1 + 1/t)^m w_1(Q) < e^{m/t} r$. Hence for any $l \in Q$, $2^{\kappa(l)} < e^{m/t} r$, or $\kappa(l) < m/t + \lg r$; since $m < 2r$ and $t = \Theta(\sqrt{r})$, we have $\kappa(l) = O(\sqrt{r})$, and by Lemma 7.3, this bound extends to all lines. ■

The results here follow Matoušek,[49] who gave a construction that was applicable in higher dimensions, and which he made deterministic.

8 Closing Remarks

The reader may now be convinced of the wide-ranging use of randomization in computational geometry. If not, here is a list (not exhaustive, with apologies) of some other application areas: parallel algorithms;[5, 66, 41, 26] geometric minimum spanning trees;[3] ray shooting;[2, 64] line orientation queries;[17] levels in arrangements and high-order Voronoi diagrams;[57, 21] the planar traveling salesman problem;[25] and binary space partition trees.[63]

Appendix: Some probability, geometry, and notation

For A and B finite sets, let the set difference $A \setminus B = \{a \in A \mid a \notin B\}$, and the symmetric difference $A \oplus B = (A \setminus B) \cup (B \setminus A)$. The size of a set A is denoted $|A|$.

Expectations. Let X be a random variable taking a finite set A of real values. The expectation of X is $EX \equiv \sum_{x \in A} x \text{Prob}\{X = x\}$. If X and Y have expectations EX and EY, then the expectation of the random variable $X + Y$ is $EX + EY$. If X takes on only nonnegative values, then *Markov's inequality* is that $\text{Prob}\{X > kEX\}$, the probability that X is greater than kEX, is no more than $1/k$, since

$$k\text{Prob}\{X > kEX\} = k \sum_{x \in A, x > kEX} \text{Prob}\{X = x\} < \sum_{x \in A, x > kEX} x\text{Prob}\{X = x\} < EX.$$

Random permutations. We need some simple facts about random permutations. We can label a set S with n elements by the numbers 1 through n, and so consider here only permutations of the set $\mathbf{n} = \{1, 2, \ldots, n\}$. If $x = (x_1, x_2, \ldots, x_n)$ is a random permutation of $\mathbf{n}$, and $y = (y_1, y_2, \ldots, y_n)$ is a fixed permutation of $\mathbf{n}$, then the product $xy = (y_{x_1}, y_{x_2}, \ldots, y_{x_n})$ is also a random permutation: exactly one x yields a given product $z = xy$, so the probability of x and of z is $1/n!$.

If x is a random permutation, then the set $R_j(x) = \{x_1, x_2 \ldots, x_j\}$ is a random subset of $\mathbf{n}$. To see this, suppose E and F are two subsets of $\mathbf{n}$ of size j. Then there is a permutation $y = (y_1, y_2, \ldots, y_n)$ such that $y_i \in F$ if and only if $i \in E$. Thus a permutation x has $R_j(x) = E$ if and only if $R_j(xy) = F$. Thus all subsets of $\mathbf{n}$ of size j are equally likely.

Similar arguments show that $x' = (x_1, x_2 \ldots, x_j)$ is a random permutation of the set $R_j(x)$, and that x_j is a random element of $R_j(x)$ and $\{x_j, x_{j+1}, \ldots, x_n\}$.

Projective duality. A duality mapping is a one-to-one correspondence between points and lines, so that, among other things, incidence is preserved: a point p is on line l if and only if its dual point is on the dual line of p. One important form of duality is *projective*: let (w, x, y), with $w \geq 0$, be a representation of points in the plane by *homogeneous coordinates*, so that (w, x, y) and (w', x', y') represent the same point in E^2 if and only if $w'x = wx'$ and $w'y = wy'$. (No point has the representation $(0, 0, 0)$.) For any lines l, there is a triple $a = (a_w, a_x, a_y)$ so that l comprises all points whose homogeneous coordinates $p = (w, x, y)$ satisfy $a \cdot p \equiv a_w w + a_x x + a_y y = 0$. The dual point $\mathcal{D}(l)$ of l is has representation $\mathcal{D}(l) = a = (a_w, a_x, a_y)$, and the dual line $\mathcal{D}(p)$ of a point $p = (w, x, y)$ is the line comprising all points (w', x', y') satisfying $(w, x, y) \cdot (w', x', y') = 0$. Thus point p with homogeneous coordinates (w, x, y) is on l if and only if $\mathcal{D}(l)$ is on $\mathcal{D}(p)$, since both conditions are equivalent to $a \cdot p = 0$. Moreover, the mapping $\mathcal{D}()$ is *orientation preserving*: p is on the same side of l as the origin if and only $\mathcal{D}(l) = (a_w, a_x, a_y)$ is on the same side of $\mathcal{D}(p)$ as the origin, since both conditions are equivalent to $a \cdot p \geq 0$.

Appendix B

This paper repeatedly uses analogies among different geometric structures; this appendix explicitly gives the roles played by the elements of each structure.

In the following list, S is a set of *sites* and $\mathcal{F}_S$ is a set of regions "defined" by those sites. (Each region in $\mathcal{F}_S$ is defined by b or fewer elements of S, where b is the specification complexity of the particular problem, as discussed in §2.2.

Hence we have $\mathcal{F}_S$ in one-to-one correspondence with a multiset whose elements are subsets of S of size b or less.)

In each domain, there is a *conflict* relation between the sites and the regions, and for a subset R of S we are generally interested in the set $\mathcal{F}_R^0$ of all regions $F \in \mathcal{F}_S$ such that the sites defining F are in R, but the sites conflicting with it are not.

(The reader may now wish to look at the list below for several instances of these general notions.)

Sometimes, as in §5.2, we may be interested in the set of regions $F_R^k \subset \mathcal{F}_S$, the set of all regions $F \in \mathcal{F}_S$ such that the sites defining F are in R and exactly k sites conflicting with it are not. When we use a random subset R for divide-and-conquer, the regions $\mathcal{F}_R^0$ each give a subproblem, and the subproblem size is k for a region $F \in \mathcal{F}_S^k$.

For each of these geometric structures, an associated range space has the set of sites S as the underlying set, and for each region $F \in \mathcal{F}_S$, there is a range S_F comprising all sites conflicting with F. If $|S_F| = k$ then $F \in \mathcal{F}_S^k$. Lemma 2.2 can be applied to this range space to show the existence of ϵ-transversals.

We give below for each geometric structure the main sections discussing it, the name and description of the sites S, the regions $\mathcal{F}_S$, the conflict relation between sites and regions and its name, and the name of the set $\mathcal{F}_R^0$.

- Voronoi diagrams in the plane, closest point problems; §2.1.

  S: Points in the plane E^2.

  $\mathcal{F}_S$: The set of open disks bounded by circles inscribed on 3 sites.

  conflict: $s \in S$ and $D \in \mathcal{F}_S$ have $s \in D$.

  $\mathcal{F}_R^0$: The Delaunay disks of R.

- Trapezoidal diagrams; §3.1,§3.3.

  S: Line segments or lines in the plane.

  $\mathcal{F}_S$: Open cells of every diagram $\mathcal{T}(A)$ where $A \subset S$ with $|A| \leq 4$.

  conflict: $s \in S$ and $T \in \mathcal{F}_S$ have nonempty intersection (they meet).

  $\mathcal{F}_R^0$: Cells of the trapezoidal diagram of R.

- The cell of the trapezoidal diagram containing a given point p; §5.1.

  S: Lines in the plane.

  $\mathcal{F}_S$: Open cells of every diagram $\mathcal{T}(A)$ where $A \subset S$ with $|A| \leq 4$, such that the cell T has $p \in T$.

  conflict: $s \in S$ and $T \in \mathcal{F}_S$ have nonempty intersection (they meet).

  $\mathcal{F}_R^0$: The cell of $\mathcal{T}(R)$ containing p.

- Convex hulls; §4.1.

  S: Points in E^d.

  $\mathcal{F}_S$: Open halfspaces bounded by planes that contain d sites.

  conflict: $s \in S$ and $H \in \mathcal{F}_S$ have $s \in H$. The site s sees the facet with empty halfspace H.

  $\mathcal{F}_R^0$: Empty halfspaces of R.

- Linear programming; §7.1.

  S: Points in E^d.

  $\mathcal{F}_S$: Open halfspaces bounded by planes that contain d sites, such that the convex hull of the d sites meets l.

  conflict: $s \in S$ and $H \in \mathcal{F}_S$ have $s \in H$.

  $\mathcal{F}_R^0$: The empty halfspace of the facet $T^*(R)$ that meets l. (The lower of the two such facets.)

Acknowledgements

The author is very grateful to Jon Bentley, Steve Fortune, John Hobby, and Margaret Wright for their vigorous yet tactful comments. Thanks also to Jon for suggesting Appendix B, and John for MetaPost help.

References

1. I. Adler and R. Shamir. A randomization scheme for speeding up algorithms for linear and convex quadratic programming problems with a high constraints-to-variables ratio. Technical Report 21-90, Rutgers Univ., May 1990. To appear in *Math. Programming.*

2. P. K. Agarwal. *Intersection and Decomposition Algorithms for Planar Arrangements.* Cambridge University Press, New York, 1991.

3. P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete and Computational Geometry*, pages 407–422, 1991.

4. N. Alon and E. Győri. The number of small semispaces of a finite set of points in the plane. *J. Combin. Theory Ser. A*, 41:154–157, 1986.

5. N. Alon and N. Megiddo. Parallel linear programming in fixed dimension almost surely in constant time. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 574–582, 1990.

6. B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive routing. In *Proc. 21st Annual SIGACT Symp.*, pages 479–489, 1989.

7. J. R. Bergen and H. Shvaytser. A probabilistic algorithm for computing Hough transforms. *J. Algorithms*, 4:639–656, 1991.

8. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, pages 929–965, 1989.

9. J.D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry*, to appear. Available as Technical Report INRIA 1285. Abstract published in IMACS 91, Dublin.

10. J.D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoretical Computer Science*, to appear. Available as Technical Report INRIA 1140.

11. B. Bollabás. *Combinatorics: Set Systems, Hypergraphs, Families of vectors, and Combinatorial Probability.* Cambridge University Press, New York, 1986.

12. A. Brøndsted. *An Introduction to Convex Polytopes.* Springer-Verlag, Berlin, 1983.

13. R. J. Canham. A theorem on arrangements of lines in the plane. *Israel J. Math.*, 7:393–397, 1969.

14. B. Chazelle. An optimal convex hull algorithms and new results on cuttings. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 29–38, 1991.

15. B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, pages 485–524, 1991.

16. B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 590–600, 1988.

17. B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Lines in space: Combinatorics, algorithms and applications. In *Proc. 21st Annual SIGACT Symp.*, pages 382–393, 1989.

18. B. Chazelle and J. Friedman. A deterministic view of randomization and its use in geometry. *Combinatorica*, 10:229–249, 1990.

19. B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete and Computational Geometry*, 4, 1989.

20. L. P. Chew. Building Voronoi diagrams of convex polygons in linear expected time. Unpublished manuscript.

21. K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete and Computational Geometry*, 2:195–222, 1987.

22. K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 452–456, 1988. Revised version: Las Vegas algorithms for linear and integer programming when the dimension is small (preprint).

23. K. L. Clarkson. Random sampling in computational geometry, II. *Proc. Fourth ACM Symp. on Comp. Geometry*, pages 1–11, 1988.

24. K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17:830–847, 1988.

25. K. L. Clarkson. Approximation algorithms for planar traveling salesman tours and minimum-length triangulations. In *Proc. 2nd ACM-SIAM Symp. Discrete Algorithms*, pages 17–23, 1991.

26. K. L. Clarkson, R. Cole, and R. E. Tarjan. Randomized parallel algorithms for trapezoidal diagrams. In *Proc. Seventh ACM Symp. on Comp. Geometry*, pages 152–161, 1991.

27. K. L. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and surfaces. *Discrete and Computational Geometry*, 5:99–160, 1990.

28. K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, 1992.

29. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4:387–421, 1989.

30. K. L. Clarkson, R. E. Tarjan, and C. J. Van Wyk. A fast Las Vegas algorithm for triangulating a simple polygon. *Proc. Fourth ACM Symp. on Comp. Geometry*, 1988.

31. O. Devillers. Randomization yields simple $O(n \log^\star n)$ algorithms for difficult $\omega(n)$ problems. *International Journal on Computational Geometry and Aplications*, to appear. Full paper available as Technical Report INRIA 1412. Abstract published in the Third Canadian Conference on Computational Geometry 1991 in Vancouver.

32. O. Devillers, S. Meiser, and M. Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. In *WADS 91*, volume LNCS 519. Springer Verlag, 1991. Full version available as Technical Report INRIA 1349.

33. M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM Journal on Computing*, 13:31–45, 1984.

34. H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, New York, 1987.

35. H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, M. Sharir, J. Snoeyink, and E. Welzl. Implicitly representing arrangements of lines or segments. *Discrete and Computational Geometry*, pages 433–466, 1989.

36. H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and of segments. *Discrete and Computational Geometry*, pages 161–196, 1990.

37. H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity of many cells in arrangements of planes and related problems. *Discrete and Computational Geometry*, pages 197–216, 1990.

38. P. Erdős and J. Spencer. *Probabilistic Methods in Combinatorics.* Academic Press, New York, 1974.

39. S. J. Fortune. Voronoi diagrams and Delaunay triangulations. *This volume.*

40. L.J. Guibas, D.E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Proc. of ICALP*, pages 414 – 431, 1990. To appear in *Algorithmica.*

41. T. Hagerup, H. Jung, and E. Welzl. Efficient parallel computation of arrangements of hyperplanes in d dimensions. In *Proceedings of the Second Symposium on Parallel Algorithms and Architectures*, pages 290–297, 1990.

42. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete and Computational Geometry*, 2:127–151, 1987.

43. G. Kalai. A subexponential randomized simplex algorithm. Extended abstract, 1991.

44. N. Kiryati, Y. Eldar, and A. M. Bruckstein. A probabilistic Hough transform. *Pattern Recognition*, 24, 1991.

45. V. Klee and G. J. Minty. How good is the simplex algorithm? In *Inequalities III*, pages 159–175. Academic Press, 1972.

46. R. Klein. Abstract Voronoi diagrams and their applications. In *Computational Geometry and its Applications (CG '89)*, pages 148–157. LNCS 344, 1988.

47. J. Komlós. Manuscript, 1990.

48. N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 68–77, 1987.

49. J. Matoušek. Cutting hyperplane arrangements. *Discrete and Computational Geometry*, pages 385–406, 1991.

50. J. Matoušek. Efficient partition trees. *Proc. Seventh ACM Symp. on Comp. Geometry*, pages 1–9, 1991.

51. J. Matoušek. Reporting points in halfspaces. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 207–215, 1991.

52. P. McMullen. The maximum number of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

53. N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31:114–127, 1984.

54. K. Mehlhorn, St. Meiser, and C. Ò'Dunlaing. On the construction of abstract Voronoi diagrams. *Discrete and Computational Geometry*, 6:211 – 224, 1991.

55. K. Mulmuley. A fast planar point location algorithm: part I. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 580–589, 1988. Full version to appear, *J. Symb. Logic.*

56. K. Mulmuley. On obstructions in relation to a fixed viewpoint. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 592–597, 1989.

57. K. Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete and Computational Geometry*, 6:307–338, 1990.

58. K. Mulmuley. A fast planar point location algorithm, II. *Journal of the ACM*, 38:74–103, 1991.

59. K. Mulmuley. Randomized multidimensional search trees: Dynamic sampling. In *Proc. Seventh ACM Symp. on Comp. Geometry*, 1991.

60. K. Mulmuley. Randomized multidimensional search trees: Further results in dynamic sampling. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 216 – 227, 1991.

61. K. Mulmuley. Randomized multidimensional search trees: Lazy balancing and dynamic shuffling. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 180 – 194, 1991.

62. J. Pach and G. Wöginger. Some new bounds for ϵ-nets. In *Proc. Sixth ACM Symp. on Comp. Geometry*, pages 10–15, 1990.

63. M. S. Paterson and F. F. Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete and Computational Geometry*, pages 485–504, 1990.

64. M. Pellegrini. Stabbing and ray shooting in 3 dimensional space. In *Proc. Seventh ACM Symp. on Comp. Geometry*, pages 177–186, 1990.

65. J. Reif and S. Sen. Optimal parallel algorithms for computational geometry. In *Proc. 16th International Conference on Parallel Processing*, 1987.

66. J. Reif and S. Sen. Polling: A new randomized sampling technique for computational geometry. In *Proc. 21st Annual SIGACT Symp.*, 1989.

67. R. Reischuk. A fast probabilistic parallel sorting algorithm. In *Proc. 22nd IEEE Symp. on Foundations of Computer Science*, pages 212–219, 1981.

68. L. A. Santalo. *Integral Geometry and Geometric Probability.* Addison-Wesley, Reading, 1976.

69. N. Sauer. On the density of families of sets. *J. Combinatorial Theory, (A)*, 13, 1972.

70. A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

71. O. Schwarzkopf. Dynamic maintenance of geometric structures made easy. *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 197 – 206, 1991.

72. R. 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.

73. R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete and Computational Geometry*, pages 423–433, 1991.

74. M. Sharir. The k-set problem for arrangements of curves and surfaces. *Discrete and Computational Geometry*, to appear.

75. S. Shelah. A combinatorial problem; stability and order for models and theories of infinitary languages. *Pacific J. Math.*, 41, 1972.

76. J. Spencer. Puncture sets. *J. Combinatorial Theory A*, 17:329–336, 1974.

77. J. Spencer. *Ten Lectures on the Probabilistic Method.* SIAM, Philadelphia, 1987.

78. R. E. Stone and C. A. Tovey. The simplex and projective scaling methods as iteratively reweighted least squares methods. *SIAM Review*, 33:220–237, 1991.

79. V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, New York, 1982.

80. V. N. Vapnik and A. Y. A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Th. Prob. and its Appl.*, 16, 1971.

81. E. Welzl. Personal communication.

82. E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. Fourth ACM Symp. on Comp. Geometry*, pages 23–33, 1988.

83. L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough transform (RHT). *Pattern Recognition Lett.*, 5, 1990.