Discrete Comput Geom 4:387-421 (1989)

Applications of Random Sampling in Computational Geometry, II

Kenneth L. Clarkson and Peter W. Shor

AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

Abstract. We use random sampling for several new geometric algorithms. The algorithms are "Las Vegas," and their expected bounds are with respect to the random behavior of the algorithms. These algorithms follow from new general results giving sharp bounds for the use of random subsets in geometric algorithms. These bounds show that random subsets can be used optimally for divide-and-conquer, and also give bounds for a simple, general technique for building geometric structures incrementally. One new algorithm reports all the intersecting pairs of a set of line segments in the plane, and requires $O(A+n \log n)$ expected time, where A is the number of intersecting pairs reported. The algorithm requires $O(n)$ space in the worst case. Another algorithm computes the convex hull of n points in E^d in $O(n \log n)$ expected time for $d=3$, and $O(n^{\lfloor d/2 \rfloor})$ expected time for $d>3$. The algorithm also gives fast expected times for random input points. Another algorithm computes the diameter of a set of n points in E^3 in $O(n \log n)$ expected time, and on the way computes the intersection of n unit balls in E^3. We show that $O(n \log A)$ expected time suffices to compute the convex hull of n points in E^3, where A is the number of input points on the surface of the hull. Algorithms for halfspace range reporting are also given. In addition, we give asymptotically tight bounds for $(\leq k)$-sets, which are certain halfspace partitions of point sets, and give a simple proof of Lee's bounds for high-order Voronoi diagrams.

1. Introduction

In recent years, random sampling has seen increasing use in discrete and computational geometry, with applications in proximity problems, point location, and range queries [10], [11], [28]. These applications have largely used random sampling for divide-and-conquer, to split problems into subproblems each guaranteed to be small. In this paper we use random sampling in a similar way, with the additional observation that the total of the sizes of the subproblems is small on the average. This fact gives improved resource bounds for a variety of randomized algorithms.

A key application of this sharper average-case bound is a general result implying that a simple, general technique for computing geometric structures yields asymptotically optimal algorithms for several fundamental problems. This method is a small change to one of the simplest ways of building a geometric structure, the incremental approach: for example, for determining the intersection of a set of halfspaces, this approach adds the halfspaces one by one and maintains the resulting intersections.

Such an incremental approach gives an optimal algorithm for constructing an arrangement of hyperplanes [23]. In general, we have a set of objects, not necessarily halfspaces or hyperplanes, that determine a structure, and we add the objects one by one, maintaining the resulting structure. One variant of this incremental approach, a simple way to randomize the process, is to add the objects in random order. Chew [9] used this approach for building Voronoi diagrams of the vertices of convex polygons. In this paper we prove a general theorem regarding a version of this randomized and incremental technique. We should note that although our technique is incremental, it is not on-line, as some simple information is maintained for the objects that are not yet added.

Some general terminology and assumptions: in this paper the dimension d is generally considered to be fixed. The expected resource bounds shown are "Las Vegas," and the expectations are with respect to the random behavior of the algorithms, unless otherwise indicated. The parameter A is generally used to denote the size of the answer to a computation. The inputs to the algorithms are assumed to be nondegenerate, so an input set of line segments has no three intersecting at the same point, an input set of points in E^d has no $d+1$ coplanar, and so on. This is no great loss of generality, as usually small tie-breaking perturbations can be appropriately applied, and the answer sizes A as defined are unchanged. Recently systematic methods have been developed to apply such perturbations "formally," that is, to break ties in an arbitrary but consistent way, so as to simulate nondegeneracy with degenerate input [22], [44].

1.1. *Problems, Results, and Related Work*

This paper is a combination of papers [14] and [16], with a sharper proof of the main divide-and-conquer theorem (here Theorem 3.6). The results can be viewed as an improvement to those of [11], and this fact suggested the title of this paper. The results in this paper have been used in an algorithm for triangulating simple polygons [17], for small-dimensional linear and integer programming [13], for an optimal parallel algorithm for Voronoi diagrams [39], and in various combinatorial results on arrangements [15].

Algorithms for Trapezoidal Diagrams and Line-Segment Intersections. For S, a set of n line segments in the plane, what are the pairs of intersecting segments of S? This computational problem has received much attention, culminating in the recent algorithm of Chazelle and Edelsbrunner requiring $O(A+n\log n)$ time in the worst case to report the A intersecting pairs [5]. Their algorithm requires (moderately) sophisticated data structures and many sophisticated algorithmic techniques, and $\Omega(n+A)$ space. This paper gives three Las Vegas algorithms for this problem. Two of the algorithms incrementally build the *trapezoidal diagram* of S (defined below), adding line segments in random order. As a by-product, the intersecting pairs of S are found. The algorithms require $O(A+n \log n)$ expected time; one requires expected $O(A+n \log n)$ space, and another requires $O(n+A)$ space in the worst case. Mulmuley [35] has independently found a similar algorithm, with the same time bound and $O(n+A)$ worst-case space bound. Another algorithm given here builds on these algorithms, and requires the same time but $O(n)$ space in the worst case. Reif and Sen [38] applied randomization to obtain parallel algorithms for related problems.

The trapezoidal diagram (or "vertical visibility map"), denoted $\mathscr{T}(S)$, is defined as follows: for every point p that is either an endpoint of a segment in S, or an intersection point of two segments in S, extend a vertical segment from p to the first segment of S above p, and to the first segment of S below p. If no such segment is "visible" to p above it, then extend a vertical ray above p, and similarly below. The resulting vertical segments, together with the segments in S, form a subdivision of the plane into simple regions that are generally trapezoids. We call this subdivision the trapezoidal diagram. (We call these regions trapezoids even though some are only degenerately so, and we may also call them cells.)

Convex Hulls. We give a Las Vegas algorithm for computing the convex hull of n points in E^3. The algorithm requires $O(n \log A)$ expected time for any set of points in E^3, where A is the number of points of S on the surface of the hull. Kirkpatrick and Seidel obtained a deterministic algorithm for planar convex hulls with the same time bound [31]. We also give a Las Vegas incremental algorithm requiring $O(n \log n)$ expected time for $d=3$ and $O(n^{\lfloor d/2 \rfloor})$ expected time for $d>3$. This improves known results for odd dimensions [36], [40], [41], [20]. For independently identically distributed points, the algorithm requires $O(n)\sum_{1 \le r \le n} f(r)/r^2$ expected time, where $f(r)$ is the expected size of the convex hull of r such points. (Here $f(r)$ must be nondecreasing.) The algorithm is not complicated.

Spherical Intersections and Diametral Pairs. We give a Las Vegas algorithm for determining the intersection of a set of unit balls in E^3, the problem of *spherical intersection.* This problem arises in the computation of the *diameter* of a point set in E^3. For a set S of n points, the diameter of S is the greatest distance between two points in S. We give a randomized reduction from the diameter problem to the spherical intersection problem, resulting in a Las Vegas algorithm for the diameter requiring $O(n \log n)$ expected time. The best algorithms previously known for this problem have worst-case time bounds no better than $O(n\sqrt{n} \log n)$ [2].

Tight Bounds for ($\le k$)-sets. Let $S \subset E^d$ contain n points. A set $S' \subset S$ with $|S'|=j$ is a *j-set* of S if there is a hyperplane that separates S' from the rest of S. A j-set is a $(\le k)$-set if $j \le k$. Let $g_k(S)$ be the number of $(\le k)$-sets, and let $g_{k,d}(n)$ be the maximum value of $g_k(S)$ over all n-point sets $S \subset E^d$.

This paper shows that

$$g_{k,d}(n) = \Theta(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil}),$$

as $n/k \to \infty$, for fixed d. The proof technique for the combinatorial bound can also be applied to give $(\leq k)$-set bounds for independently identically distributed points. For example, if the convex hull of such a set of points has $f(n)$ expected facets, then the expected number of $(\leq k)$-sets is $O(k^d f(n/k))$. The proof technique employed for the improved bounds is an instance of a "probabilistic method" [24]. The $(\leq k)$-set bounds are a corollary of more general results that are intimately related with the probabilistic results for the complexity analysis of our algorithms.

As a by-product of our techniques, we give an alternative derivation of a bound for the complexity of higher-order Voronoi diagrams.

The concept of a k-set is a generalization of the concept of a convex hull facet, which can be viewed as a d-set. The new bound is a generalization of the known upper bound $O(n^{\lfloor d/2 \rfloor})$ for the number of facets of a convex polytope with n vertices. Indeed, the new bound follows from this polytope upper bound. Our bound is within a small constant factor of the tight bounds known for the plane [25], [3], [42], and it improves previous results for $d=3$ [18], [7], [11]; apparently no interesting bounds were known before for higher dimensions. The proof of the bound is also considerably simpler than those given for the earlier, weaker bounds.

Improved Bounds for Range Reporting. The halfspace range-reporting problem is this: for a set S of n points, build a data structure so that given some query halfspace, the points of S in the halfspace can be reported quickly. The new bound for $(\leq k)$-sets is applied in this paper to sharpen the analysis of the algorithm of [7] for halfspace range reporting. It is also used to analyze two new algorithms for that problem. One algorithm is shown to require expected $O(n^{\lfloor d/2 \rfloor + \varepsilon})$ preprocessing time, and in the worst case $O(n^{\lfloor d/2 \rfloor + \varepsilon})$ storage. The resulting query time is $O(A + \log n)$, where A is the size of the answer to the query. These resource bounds apply for any fixed $\varepsilon > 0$, and the constant factors in the bounds depend on d and ε. Another algorithm requires $O(n)$ storage, $O(n \log n)$ expected preprocessing time, and allows queries to be answered in $O(A + n^{1+\varepsilon-\gamma})$ time, where $\gamma = 1/(1+(d-1)\lfloor d/2 \rfloor)$. The algorithm is a variant of that of Haussler and Welzl [28]. Their query time is $O(n^{1+\varepsilon-\gamma'})$, where $\gamma' = 1/(1+d(d-1))$. (This is independent of the answer size, however.)

These results do not improve the algorithm of [6] for halfplane queries; that algorithm requires $O(n)$ storage, $O(n \log n)$ preprocessing, and $O(A + \log n)$ query time. See also [43] and [8] for recent related results.

1.2. Outline of the Paper

The remainder of this section gives an informal discussion of the ideas in this paper. The next section gives a description of the formal framework used in the theorems, and the main lemma for the rest of the paper. This lemma is then applied in Section 3 to give a general theorem that implies the asymptotically tight bound for $(\leq k)$-sets. We also prove a general theorem for probabilistic divide-and-conquer in Section 3, and a general result on randomized incremental construction of geometric structures. In Section 4 we apply these results to trapezoidal diagrams, convex hulls, spherical intersections and diameter, and halfspace range queries. The final section gives some concluding remarks.

1.3. The Ideas

This section gives a low-rigor general discussion of the ideas in this paper. These ideas begin with some observations about random samples of point sets, and the information that we can get from such samples.

Random Samples. Assume S is a set of points in the plane. Assume $R \subset S$ of size r is chosen at random, with all subsets of size r equally likely. Let e be an edge of the convex hull of R, and let l be the straight line containing e. The points in the halfplane bounded by l, and not containing R, are said to be *beyond* the edge e. Then with probability $1 - 1/n^{\Omega(1)}$, for every edge e of the convex hull of R, the number of points of S that are beyond e is $O(\log r)n/r$. That is, with high probability the convex hull of a random subset splits a point set in small pieces. Intuitively, the fact that an edge e has no points of the random sample R beyond it is good evidence that e has few points of S beyond it. This kind of tail estimate has been the basis of several previous applications of random sampling to computational geometry [11], [28].

In addition to the tail estimate, another fact holds for R and S: let S_e be the set of points in S that are beyond e, and let Z be the set of edges of the convex hull of R. Then the expected value of $\sum_{e \in Z} |S_e|$ is $O(n)$. In contrast, the tail estimate says only that $\sum_{e \in Z} |S_e|$ is $O(n \log r)$. (This is with high probability, though.) This observation is a key new idea in this paper. Moreover, $\sum_{e \in Z} |S_e|$ behaves roughly as a sum of Poisson random variables, so that $\sum_{e \in Z} |S_e|^c$ is $(n/r)^c O(r)$, for nonnegative integer c.

Why is this kind of bound useful in computing convex hulls? One approach is to take a large random subset $R \subset S$, recursively compute the convex hull of R, and then determine the result of adding the remaining points $S \backslash R$. Roughly speaking, the changes owing to the remaining points can be expected to be "local," and require a small amount of work per point. Indeed, for a given point $p \in S \backslash R$, the number of such changes is proportional to the number of edges of the hull of R that p is beyond. The total of these changes is therefore expected $O(n)$, as in the above discussion.

Instead of algorithms for convex hulls of point sets, we describe algorithms for determining the intersection of a set of halfspaces. This problem is linear-time equivalent to the convex-hull problem via well-known duality mappings. The above average-case result translates as follows to halfplane intersections: assume S is a set of n halfplanes, and $R \subset S$ is a random subset of S of size r. Let S_e

denote the set of halfplanes in S that do not contain line segment e. Then the expected value of $\sum_{e\in Z}|S_e|$ is $O(n)$, where Z is the set of edges bounding the intersection of the halfplanes in R. Intuitively, because all the halfspaces of R contain e, it is likely that most of the halfspaces of S do as well.

Randomized Incremental Construction. Similar observations are also useful in applying divide-and-conquer to other intersection problems, such as determining the number of intersecting pairs in a set of n line segments in the plane, or finding the intersection of a set of unit balls in E^3. For these problems, however, we give a simple alternative to divide-and-conquer, a technique we call *randomized incremental construction.*

To motivate this technique we give a way to speed up insertion sort. Recall that insertion sort constructs a sorted list of values from an unsorted list by building up the sorted list one value at a time; at each step, an item from the unsorted list is put into its place in the sorted list. Each step of insertion sort is time-consuming because a large proportion of the sorted list may be examined at each step. One way to speed up the sorting is to remember, for each value not yet inserted, its location in the current sorted list, and conversely to keep a list of all uninserted values that go in a particular location in the current sorted list. Inserting a number in the sorted list now is easy; if c goes between a and b on the sorted list, we use our stored information to put it there; we must also look at the list of uninserted values between a and b, and decide for each uninserted value in the list whether that value goes between a and c, or between c and b.

This "sped-up" insertion sort is just a version of quicksort [30]. The time required for insertion is now dominated by the time for the partitioning step; this time is proportional to the number of uninserted values between a and b, when we insert c between them. Suppose we insert numbers in random order. Then at step r, the inserted values will be fairly evenly distributed among the whole set, so the number of values to partition will be about n/r on the average. The whole sorting process then takes expected time proportional to $n\sum_{1\le r\le n} 1/r = O(n \log n)$.

Our technique of randomized incremental construction is a similar transformation from an "insertion"-like algorithm to a "quick" one. For finding the convex polygon that is the intersection of a set S of halfplanes, we determine the polygon incrementally, randomly choosing a halfplane and slicing off the portion of the current polygon that is not contained in that halfplane. As with insertion sort, this technique is slow if we just examine the edges of the current polygon to determine which must be removed; instead, we remember for each edge those uninserted halfplanes that do not contain it. The work of inserting a halfplane is now dominated by the work in updating this edge-halfplane *conflict* information.

When we insert halfplanes in random order, the expected time required to add a halfplane is $O(n/r)$: at each step, the set R of inserted halfplanes is a random subset of S, so that the facts about random subsets discussed above can be applied. The algorithm requires optimal $O(n \log n)$ expected time.

Bounding ($\leq k$)-sets. For the combinatorial problem of bounding the number of ($\leq k$)-sets, it is helpful to use a kind of converse to the above relation between the convex hulls of point sets. Let $S \subset E^2$ and let R be a random subset of S of size r. Consider two points $a, b \in S$; they define some line l that has $S' \subset S$ on one side of it. With probability roughly $(r/n)^2$, a and b will be in R. If no points of S' are chosen for R, a and b will be vertices of the convex hull of R. If $|S'| < n/r$, this will occur with probability at least about $(1-|S'|/n)^r \approx 1/e$. That is, with probability proportional to $(r/n)^2$, the pair of points a and b with $|S'| < n/r$ contribute an edge to the hull of R. However, the convex hull of R has at most r edges, so if z is the number of such pairs of points, we know $C(r/n)^2 z \leq r$, for a constant C, so $z \leq O(n)n/r$. Put another way, if $k = n/r$, the number of pairs of points of S with k or fewer on one side is $O(nk)$. The number of such pairs is roughly the same as the number of ($\leq k$)-sets, so we have bounded that quantity. In short, the bound on the complexity of the convex hull of the random subset R of size n/k implies a bound on the number of ($\leq k$)-sets of S.

2. The Formal Framework and Main Lemma

The ideas in this paper can be applied to a variety of geometric structures. To aid and to show this generality, a formal and abstract framework for geometric computations is useful. This framework is similar to that in [11].

Let S be a set of n subsets of E^d; the elements of S are also called *objects*. The set S is the input to a geometric computation, and so could be a set of points (singleton sets), line segments in the plane, halfspaces, or balls. Let $\mathcal{F}$ be a set of subsets of E^d, which we term *ranges*, as in range query, or *regions*.

In applications the ranges are defined by the objects in some way. When computing convex hulls of points in the plane, the objects are points, and the ranges are halfplanes, so the ranges defined by the objects are those open halfplanes that are bounded by lines through pairs of the points. The notion of "defined" is formalized as follows: for integer b, let $S^{(b)}$ be the collection of subsets of S with b or fewer elements, and let δ be a relation between $\mathcal{F}$ and $S^{(b)}$. A range $F \in \mathcal{F}$ is defined by $X \in S^{(b)}$ if $F\delta X$, that is, if the δ relation holds between X and F, so the set of ranges $\mathcal{F}_S$ defined by the objects in S is

$$\mathcal{F}_S = \{F \in \mathcal{F} \mid F\delta X, X \in S^{(b)}\}.$$

In problems of construction, the desired computation is this: determine all ranges $F \in \mathcal{F}_S$ such that $F \cap s$ is empty for all objects $s \in S$. For convex hulls, this is the set of all open halfplanes in $\mathcal{F}_S$ that contain none of the input points. For Voronoi diagrams in the plane, the ranges are open disks or halfspaces, the objects are points (singletons), and the δ relation has $F\delta X$ for $X \in S^{(3)}$ when F is the open disk bounded by the circle through the points in X, and $F\delta X$ for $X \in S^{(2)}$ when F is an open halfspace bounded by the line through the points in X. The empty members of $\mathcal{F}_S$ are Delaunay disks or empty halfspaces. For trapezoidal diagrams of line segments in the plane, the objects are line segments, the ranges are open "trapezoids," and the parameter b is four. The relation δ is defined as follows: $F\delta X$ for $X \in S^{(4)}$, when F is a (generalized) trapezoid in $\mathcal{T}(X)$.

While the desired ranges in construction problems have no intersections with S, it will also be useful to consider ranges that do meet objects in S. For $F \in \mathcal{F}$ and $R \subseteq S$, let $F \wedge R$ denote the members of R that have nonempty intersection with F. Similarly, for $s \in S$, let $s \wedge \mathcal{F}$ denote the members of $\mathcal{F}$ having nonempty intersection with s. Let $|F \wedge R|$ denote the number of objects in $F \wedge R$, and let $|F|$ denote $|F \wedge S|$. For a given integer j, let $\mathcal{F}_S^j$ denote the set of $F \in \mathcal{F}$ with $|F| = j$. In construction problems, the set $\mathcal{F}_S^0$ is desired. Note that with S and $\mathcal{F}$ as for convex hulls, as mentioned above, the set $\mathcal{F}_S^j$ is closely related to the set of j-sets of S, and $|\mathcal{F}_S^j|$ is the number of such sets.

For $R \subset S$, the sets $\mathcal{F}_R$ and $\mathcal{F}_R^j$ are defined analogously: $\mathcal{F}_R$ is the collection of all ranges F such that $F \delta X$ for some $X \in R^{(b)}$, and $F \in \mathcal{F}_R^j$ if $|F \wedge R| = j$ and $F \in \mathcal{F}_R$.

Since we are mainly interested in $\mathcal{F}_R^0$, for $R \subset S$, we assume that if $F \delta X$ for $X \in S^{(b)}$, then the set $F \wedge X$ is empty.

Ranges in $\mathcal{F}_S$ are defined by b or fewer elements of S. For a given $F \in \mathcal{F}_S$, let i_F denote the size of a smallest set defining F. That is, such a set $X \in S^{(b)}$ has $F \delta X$, and $|X|$ is no larger than for any such set. For each $F \in \mathcal{F}_S$, pick some such X with $F \delta X$ and $|X| = i_F$, and call it X_F.

When there is only one such possible X_F for each $F \in \mathcal{F}_S$, we say that δ is *functional*. (Most of the theorems hold if the number of minimal X with $F \delta X$ is $O(1)$, not exactly 1.) For $R \subset S$ we have $F \in \mathcal{F}_R$ if and only if $X_F \subset R$. In general, the δ relation is functional when the input S is nondegenerate.

It is also convenient here to define $T_c(R)$ as

$$\sum_{F \in \mathcal{F}_R^0} \binom{|F|}{c},$$

so that $T_0(R) = |\mathcal{F}_R^0|$ and $T_1(R) = \sum_{F \in \mathcal{F}_R^0} |F|$. For integer r, let $T_c(r)$ denote the expected value $ET_c(R)$ for random $R \subset S$ of size r, with all subsets of size r equally likely.

With this machinery, here is the main lemma.

Lemma 2.1. *Let $R \subset S$ be a random subset of size r, and $c \geq 0$. Then*

$$\sum_{F \in \mathcal{F}_S} \binom{|F|}{c} \binom{n - i_F - |F|}{r - i_F - c} \leq \binom{n}{r} E|\mathcal{F}_R^c|,$$

with equality if δ is functional. Also

$$\sum_{F \in \mathcal{F}_S} \binom{|F|}{c} \binom{n - b - |F|}{r - b - c} \leq \binom{n}{r} E|\mathcal{F}_R^c|.$$

Proof. For $F \in \mathcal{F}_S$, let I_F be the indicator function for the event $F \in \mathcal{F}_R^c$, so $I_F = 1$ when $F \in \mathcal{F}_R^c$ and 0 otherwise. Then $|\mathcal{F}_R^c| = \sum_{F \in \mathcal{F}_S} I_F$, so $E|\mathcal{F}_R^c|$ is

$$E\left[\sum_{F \in \mathcal{F}_S} I_F\right] = \sum_{F \in \mathcal{F}_S} EI_F = \sum_{F \in \mathcal{F}_S} \operatorname{Prob}\{F \in \mathcal{F}_R^c\}$$
$$\geq \sum_{F \in \mathcal{F}_S} \binom{|F|}{c} \binom{n - i_F - |F|}{r - i_F - c} \Big/ \binom{n}{r}.$$

The inequality here follows from the observation that, for $F \in \mathcal{F}_S$, $F \in \mathcal{F}_R^c$ if $|F \wedge R| = c$ and $X_F \subset R$. The number of subsets of size r that satisfy these conditions, divided by the total number of subsets of that size, gives the probability of this event. The number of such subsets is

$$\binom{|F|}{c}\binom{n-i_F-|F|}{r-i_F-c},$$

choosing c elements of R from among those meeting F, and choosing $r-i_F-c$ elements of R from among the $n-i_F-|F|$ elements of S that neither meet F nor are in X_F. When δ is functional, $F \in \mathcal{F}_R^c$ only if $X_F \subset R$ and $|F \wedge R| = c$, so the above holds with equality.

The second inequality of the lemma statement follows from

$$\binom{n-b-|F|}{r-b-c} \le \binom{n-i_F-|F|}{r-i_F-c} \quad \text{for } i_F \le b$$

(and for the relevant values of n, r, c, and $|F|$). □

3. The Probabilistic Theorems

3.1. *Improved Bounds for ($\le k$)-sets*

Rather than prove an upper bound for ($\le k$)-sets only, we prove a much more general result, from which the ($\le k$)-set bound will follow as a corollary.

Theorem 3.1. *With the notation of Section* 2, *for* $k > 1$

$$\sum_{0 \le j \le k} |\mathcal{F}_S^j| \le 4k^b T_0(\lfloor n/k \rfloor)(1 + O(k/n)),$$

as $k/n \to 0$.

Proof. Let $R \subset S$ be a random subset of size $r = \lfloor n/k \rfloor$, with $k > 1$. From Lemma 2.1 with $c = 0$, we know that $T_0(r)$ is

$$\begin{aligned} E|\mathcal{F}_R^0| &\ge \sum_{j \ge 0} \binom{n-b-j}{r-b} |\mathcal{F}_S^j| \Big/ \binom{n}{r} \\ &\ge \sum_{0 \le j \le k} \binom{n-b-j}{r-b} |\mathcal{F}_S^j| \Big/ \binom{n}{r} \\ &\ge \sum_{0 \le j \le k} |\mathcal{F}_S^j| / (4k^b)(1 + O(k/n)). \end{aligned}$$

The last inequality is easily proven using Stirling's formula. (Hint: reduce to within $1+O(k/n)$ of

$$\frac{1}{k^b}\left(1+\frac{kr}{n(n-k-r)}\right)^{n-k-r}\left(1-\frac{k}{n}\right)^r\left(1-\frac{r}{n}\right)^k$$

using Stirling's formula, then observe that the product of the middle two terms is $1+O(k/n)$ and the last term is bounded below by $\frac{1}{4}$.) □

The following theorem is of particular interest in the applications in the next section.

Theorem 3.2. *With the notation of Section* 2,

$$|\mathcal{F}_S^1| \le ebT_0(\lfloor bn/(b+1)\rfloor)(1+O(1/b)+O(b/n))$$

and

$$|\mathcal{F}_S^2| \le (eb/2)^2 T_0(\lfloor bn/(b+2)\rfloor)(1+O(1/b)+O(b/n)).$$

Proof. Let $R\subset S$ be a random subset of size $r=\lfloor bn/(b+1)\rfloor$. From Lemma 2.1 with $c=0$,

$$E|\mathcal{F}_R^0| \ge \sum_{j\ge 0}\binom{n-b-j}{r-b}|\mathcal{F}_S^j|\Big/\binom{n}{r}$$
$$\ge \binom{n-b-1}{r-b}|\mathcal{F}_S^1|\Big/\binom{n}{r},$$

and so

$$|\mathcal{F}_S^1| \le E|\mathcal{F}_R^0|\frac{n-b}{n-r}\binom{n}{b}\Big/\binom{r}{b}.$$

The first bound of the theorem readily follows, and the second bound is similarly proven. □

Theorem 3.1 gives a bound on the quantity $\hat{g}_{k,d}(n)$, defined as follows: assume $S\subset E^d$ is a set of n points, $\mathcal{F}$ is the set of halfspaces in E^d, and $\mathcal{F}_S$ is the set of halfspaces bounded by hyperplanes that are affine hulls of points in S. Then $\hat{g}_{k,d}(n)$ is the maximum, over all such S, of $\sum_{0\le j\le k}|\mathcal{F}_S^j|$. A member of $\mathcal{F}_S^j$ is called a *j-facet* of S.

Corollary 3.3.

$$g_{k,d}(n)=O(n^{\lfloor d/2\rfloor}k^{\lceil d/2\rceil}),$$

as $n/k\to\infty$.

Proof. It is easy to show, using the results of Section 3.2 of [20], that $g_{k,d}(n) = O(\hat{g}_{k,d}(n))$.

We can assume that S is nondegenerate, so that no $d+1$ points of S are on a common hyperplane. This is no loss of generality, as $g_k(S)$ attains its maximum when S is nondegenerate [20]. To apply the theorem to $\hat{g}_{k,d}(n)$, S is a set of points in E^d (or more precisely, a collection of singleton sets of points of E^d). The set of ranges $\mathcal{F}$ is the set of open halfspaces of E^d, and with $b=d$, the δ relation is defined as follows: for $X \in S^{(b)}$, let $F\delta X$ when $|X| = b$ and F is bounded by the affine hull of the points in X. The upper bound for $\hat{g}_{k,d}(n)$ follows, using the upper bound $O(r^{\lfloor d/2 \rfloor})$ for $|\mathcal{F}_R^0|$, here the number of facets of a polytope with r vertices [20, Section 6.2.4]. □

Lemma 3.4.

$$g_{k,d}(n) = \Omega(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil}),$$

as $n/k \to \infty$.

Proof. Omitted. Cyclic polytopes [20] realize the bound, as can be shown using the techniques of the theorem, or constructively [19]. □

Although bounds on $|\mathcal{F}_S^j|$, for given j, seem to be difficult to obtain, the following result is of interest.

Theorem 3.5. *Let $S \subset E^3$ be a set of n points that are the vertices of a convex polytope. Assume S is nondegenerate, that is, has no four points coplanar. Then S has $2(j+1)(n-j-2)$ j-facets.*

Using well-known relations between Voronoi diagrams in the plane and convex hulls of point sets in E^3, this result gives a sharp bound on the number of vertices of order j Voronoi diagrams. This is an alternate proof of the bound of D. T. Lee for this quantity. This result is stated as Corollary 13.35 of [20], and given yet another proof.

Proof. Assume $R \subset S$ is random, of size r with $4 \le r \le n$. Since S forms the vertices of a convex polytope, so does $R \subset S$. Since S is nondegenerate, so is R. Therefore $|\mathcal{F}_R^0|$, the number of facets of the convex hull of R, is $2(r-2)$ [20, Theorem 6.11]. With the nondegeneracy condition, the δ relation here is functional, and by Lemma 2.1 we have, for $4 \le r \le n$,

$$\sum_{j \ge 0} \binom{n-j-3}{r-3} |\mathcal{F}_S^j| = \binom{n}{r} E|\mathcal{F}_R^0| = 2(r-2)\binom{n}{r}.$$

The same expression holds for $r=3$, since any set of three points of S defines two j-facets. The values $|\mathcal{F}_S^j|=2(j+1)(n-j-2)$, for $0\le j\le n-3$, satisfy these equations, since

$$\sum_{0\le j\le n-3}\binom{n-j-3}{r-3}(n-j-2)2(j+1)=2(r-2)\sum_{0\le j\le n-3}\binom{n-j-2}{r-2}\binom{j+1}{1}$$
$$=2(r-2)\binom{n}{r},$$

using well-known binomial coefficient identities (see, e.g., [26]). The matrix associated with this linear system has a determinant with absolute value 1, using expansion by minors, and using the facts that

$$\binom{n-j-3}{r-3}=1 \quad \text{for } r=n-j\ge 0, \qquad \binom{n-j-3}{r-3}=0 \quad \text{for } r>n-j\ge 0.$$

Thus the given solution is unique. □

In the nondegenerate case but with S not necessarily the vertex set of a polytope, we can express $|\mathcal{F}_S^j|$ as

$$|\mathcal{F}_S^j|=\sum_{3\le r\le n}\binom{r-3}{n-3-j}\binom{n}{r}(-1)^{j+r+n}T_0(r).$$

If only this implied something interesting about $|\mathcal{F}_S^j|$!

3.2. *Probabilistic Divide-and-Conquer*

This section gives results implying that random sampling can be used effectively for a divide-and-conquer approach to geometric problems. A corollary is also given that combines the results of [11] with the main theorem.

To state the main theorem needed, some terminology and notation in addition to that in Section 2 is useful:

For nonnegative integers k and c, let $k^{\underline{c}}$ denote the "falling power" $c!\binom{k}{c}$. Recall that a function W from the reals to the reals is concave when

$$W(\alpha x+(1-\alpha)y)\ge \alpha W(x)+(1-\alpha)W(y),$$

for all x, y and α with $0\le\alpha\le 1$. (That is, when these values are defined.) Note that x^β is a concave function of x, for $0\le\beta\le 1$, as is the logarithm function.

For $R\subset S$, nonnegative integer c, and function W from the nonnegative reals to the nonnegative reals, let $T_{W,c}(R)$ denote

$$\sum_{F\in\mathcal{F}_R^0} W\left(\binom{|F|}{c}\right).$$

That is, $T_{W,c}(R)$ is the total work done for $\mathcal{F}_R^0$ when $W\left(\binom{|F|}{c}\right)$ work is done for the $|F|$ objects of S meeting $F \in \mathcal{F}_R^0$. The earlier notation $T_c(R)$ is the case $W(j) = j$. Finally, let

$$\tau_0(r) = \max_{1 \le z \le r} T_0(z).$$

Theorem 3.6. *With the terminology of Section* 2 *and above, assume the relation δ is functional, the function W is concave, and c is a nonnegative integer. Assume R is a random subset of S of size r. Then*

$$ET_{W,c}(R) \le W\left(\frac{(n-r+c)^{\underline{c}}}{(r-b)^{\underline{c}}} K_{c,b}\right) E|\mathcal{F}_R^0|,$$

where $K_{c,b} = E|\mathcal{F}_R^c| / E|\mathcal{F}_R^0|$.

Proof. Let I_F be the indicator function for the event $F \in \mathcal{F}_R^0$, so $I_F = 1$ where $F \in \mathcal{F}_R^0$ and 0 otherwise. Then $ET_{W,c}(R)$ is

$$\begin{aligned} E\left[\sum_{F \in \mathcal{F}_S} I_F W\left(\binom{|F|}{c}\right)\right] &= \sum_{F \in \mathcal{F}_S} EI_F W\left(\binom{|F|}{c}\right) \\ &= \sum_{F \in \mathcal{F}_S} \text{Prob}\{F \in \mathcal{F}_R^0\} W\left(\binom{|F|}{c}\right) \\ &\le E|\mathcal{F}_R^0| W\left[\sum_{F \in \mathcal{F}_S} \text{Prob}\{F \in \mathcal{F}_R^0\} \binom{|F|}{c} \Big/ E|\mathcal{F}_R^0|\right], \end{aligned}$$

using the concavity of W and the fact that $E|\mathcal{F}_R^0| = \sum_{F \in \mathcal{F}_S} \text{Prob}\{F \in \mathcal{F}_R^0\}$.

Now using Lemma 2.1 and the assumption that δ is functional, the sum in the last expression above is bounded:

$$\begin{aligned} \sum_{F \in \mathcal{F}_S} \text{Prob}\{F \in \mathcal{F}_R^0\} \binom{|F|}{c} &= \sum_{F \in \mathcal{F}_S} \binom{|F|}{c} \binom{n - i_F - |F|}{r - i_F} \Big/ \binom{n}{r} \\ &\le \frac{(n-r+c)^{\underline{c}}}{(r-b)^{\underline{c}}} \sum_{F \in \mathcal{F}_S} \binom{|F|}{c} \binom{n - i_F - |F|}{r - i_F - c} \Big/ \binom{n}{r} \\ &= E|\mathcal{F}_R^c| \frac{(n-r+c)^{\underline{c}}}{(r-b)^{\underline{c}}}. \end{aligned}$$

The theorem follows. □

In many applications we are only interested in the cases $c = 1$ or $c = 2$, with $W(j) = j$.

Theorem 3.7. *With the terminology of Section 2 and above, assume the relation δ is functional. Then*

$$T_1(r) \le \frac{n-r+1}{r-b} \tau_0(r) eb\left(1 + O\left(\frac{1}{b}\right) + O\left(\frac{b}{r}\right)\right)$$

and

$$T_2(r) \le \frac{(n-r+2)^2}{(r-b)^2} \tau_0(r) \left(\frac{eb}{2}\right)^2 \left(1 + O\left(\frac{1}{b}\right) + O\left(\frac{b}{r}\right)\right),$$

as $b, r \to \infty$.

In some cases, the constant factor for $T_1(r)$ is exactly b.

Proof. The result follows from Theorem 3.6; for the $c = 1$ case, we need to bound $E|\mathcal{F}_R^1|$ above by $\tau_0(r) eb(1 + O(1/b) + O(b/r))$. From Theorem 3.2, we have

$$|\mathcal{F}_R^1| \le T_0(\lfloor br/(b+1) \rfloor) eb(1 + O(1/b) + O(b/r)).$$

Note that $T_0(\lfloor br/(b+1) \rfloor)$ here refers to random subsets of R; there is no ambiguity, since random subsets of R are random subsets of S. The result follows by definition of $\tau_0(r)$. The $c = 2$ case follows similarly. □

As noted above, the function x^β is concave in x for $0 \le \beta \le 1$, as is $\log(1+x)$. Thus when j^β work is done for a range of $\mathcal{F}_R^0$ meeting j objects of S, the average work per range of $\mathcal{F}_R^0$ is $O(n/r)^\beta$, and similarly if $\log(1+j)$ work is done, the average work is $O(\log(n/r))$.

The following is a combination of the above theorem and Corollary 4.2 of [11].

Corollary 3.8. *With the terminology of Section 2 and above, assume the relation δ is functional. Assume $|\mathcal{F}_S| \le K\binom{n}{b}$ for some constant K. Then, for any $q \ge 0$, there exists $z_{r,q} = O(b \log r) + q + \ln K$ as $r \to \infty$ such that with probability $3/4 - e^{-q}$, both of these conditions hold:*

$$\sum_{F \in \mathcal{F}_R^0} |F| \le O(n/r) \tau_0(r)$$

and

$$\max_{F \in \mathcal{F}_R^0} |F| \le z_{r,q} n/r.$$

Proof. For the first condition we use Theorem 3.7 with $c=1$. By Markov's inequality, the probability that $T_1(R)$ exceeds four times its mean is no more than $1/4$. For the second condition, we use Corollary 4.2 of [11], which implies that the probability that the second condition fails is at most e^{-q}. For completeness, we prove the second bound here: assume P_k is the probability that $\max_{F \in \mathcal{F}_R^0} |F| \geq k$. Then

$$P_k = \operatorname{Prob}\left\{ \bigvee_{\substack{F \in \mathcal{F}_S \\ |F| \geq k}} F \in \mathcal{F}_R^0 \right\} \leq \sum_{\substack{F \in \mathcal{F}_S \\ |F| \geq k}} \operatorname{Prob}\{F \in \mathcal{F}_R^0\}.$$

As in Lemma 2.1 with $c=0$,

$$\operatorname{Prob}\{F \in \mathcal{F}_R^0\} \binom{n}{r} \leq \binom{n-|F|-b}{r-b} \leq \binom{n-k-b}{r-b}$$

for $|F| \geq k$, so $P_k \binom{n}{r}$ is no more than

$$\begin{aligned}\sum_{F \in \mathcal{F}_S} \binom{n-k-b}{r-b} &\leq K \binom{n}{b} \binom{n-k-b}{r-b} \\ &\leq K \binom{r}{b} \binom{n-k}{r}.\end{aligned}$$

For $k = \lceil zn/r \rceil$, we use Stirling's approximation, the bound $\binom{a}{b} \leq (ae/b)^b$, and the bound $1+x \leq e^x$ to obtain $P_k \leq K(er)^b \sqrt{z} e^{-z}$, which implies that an appropriate value $z_{r,q}$ can be chosen to satisfy the conditions of the corollary. □

3.3. *Randomized Incremental Construction*

In this section we prove an expected-time bound for randomized incremental construction, within the formal framework of Section 2. The construction problem associated with an instance of this framework is to find $\mathcal{F}_S^0$ given an input set S.

As discussed in Section 1.3, randomized incremental construction solves this problem as follows: the objects in S are added one by one in random order to a set R. As these objects are added, the set of regions $\mathcal{F}_R^0$ is maintained, and updated as each object is added. To make this algorithm faster, a *conflict graph* is maintained between the objects in $S \backslash R$ and the regions in $\mathcal{F}_R^0$. This graph has an edge between each object and region that have nonempty intersection, so that the object prevents the region from being in $\mathcal{F}_S^0$. When an object s is added to R, the regions adjacent to s in the conflict graph are deleted from $\mathcal{F}_R^0$, and new regions are added that result from the presence of s. The following theorem gives a time bound for instances of this general algorithm in which an *update condition* holds. The update condition is this: the time to add an object s to R, and to update $\mathscr{F}_R^0$ and the conflict graph, is linearly proportional to the number of objects that conflict with regions that conflict with s. That is, the work is proportional to $\sum_{F \in s \wedge \mathscr{F}_R^0} |F|$.

Plainly, the update time is at least as long as this. In many instances, this linear time suffices. Put another way, the update condition says that the work performed at a step is linear in the number of conflict-graph edges deleted at that step. Since no work is done for an edge except when creating or deleting it, with the update condition the total work is proportional to the number of edges created, which is also the number of edges deleted.

Theorem 3.9 (Randomized Incremental Construction). *In an instance of the general incremental construction algorithm, assume the update condition holds and δ is functional. Then the expected time required by the instance is $O(n)\sum_{1 \le r \le n} \tau_0(r)/r^2$.*

Proof. It is enough to show that the expected time required to add object $s \in S$ at step $r+1$ is $O(\tau_0(r))(n-r)/r^2$. This fact is a consequence of Theorem 3.7. By the update condition, the time required is proportional to

$$\sum_{F \in \mathscr{F}_S} |F| I_F,$$

where $I_F = 1$ when both F is in $\mathscr{F}_R^0$ and F meets s, and $I_F = 0$ otherwise. Here R is the current (random) set of r objects.

The expected value of this quantity is

$$\sum_{F \in \mathscr{F}_S} |F| \operatorname{Prob}\{F \in \mathscr{F}_R^0, s \in F \wedge S\},$$

or

$$\sum_{F \in \mathscr{F}_S} \frac{|F|^2}{n-r} \operatorname{Prob}\{F \in \mathscr{F}_R^0\} = \frac{T_2(r)}{n-r},$$

from the proof of Theorem 3.6. By Theorem 3.7, this quantity is on the order of $O(\tau_0(r))(n-r)/r^2$.

This completes the proof. Here is another proof: by the remarks above, we may bound the work of the algorithm by bounding the number of conflict graph edges created in the course of the algorithm. A range $F \in \mathscr{F}_S$ contributes $|F|$ edges to the number created at step $r+1$ if $F \notin \mathscr{F}_R^0$ but $F \in \mathscr{F}_{R'}^0$, where $R' = R \cup \{s\}$. This occurs if and only if $|F \wedge R'| = 0$, $X_F \subset R'$, and $s \in X_F$, so the expected number of edges created is

$$\sum_{F \in \mathscr{F}_S} |F| \operatorname{Prob}\{F \in \mathscr{F}_{R'}^0, s \in X_F\}.$$

Since R' is a random subset of S, s is a random element of R'. Given that $F \in \mathcal{F}^0_{R'}$, the probability that $s \in X_F$ is no more than $b/(r+1)$. Hence the expected number of edges created is

$$\sum_{F \in \mathcal{F}_S} |F| \frac{b}{r+1} \operatorname{Prob}\{F \in \mathcal{F}^0_{R'}\} = \frac{bET_1(R)}{r+1},$$

from the proof of Theorem 3.6. By Theorem 3.7 this is $O(\tau_0(r))(n-r)/r^2$. □

4. Algorithmic Applications

4.1. *Line-Segment Intersections*

In this section three algorithms are given for the problem of constructing $\mathcal{T}(S)$. All three require $O(A + n \log n)$ expected time. The first algorithm is an instance of the randomized incremental construction technique. The second algorithm is a refinement of the first, requiring $O(n + A)$ space in the worst case. The third algorithm requires only $O(n)$ space in the worst case. (To be precise, the third algorithm can only be said to compute line-segment intersections and vertical visibility information; it does not compute the complete trapezoidal diagram.)

For convenience of presentation, we assume that the line segments are nondegenerate, so that no three intersect at the same point, and no endpoints or intersection points are covertical (on the same vertical line). These conditions can be simulated using appropriate small perturbations, so little loss of generality is implied. (Some loss is entailed, however: we might prefer a measure of A that counts the number of intersection points, not the number of intersecting pairs. We can go further in this direction than is reported here.)

One important issue here is the representation of adjacency information for trapezoidal diagrams. For the first algorithm, we assume some limitations on the adjacency information recorded for each trapezoid. We assume that a trapezoid "knows" only its upper and lower bounding line segments, its corner points, and the trapezoids with which it shares vertical boundaries. We do not assume that a trapezoid knows all of the (possibly very many) trapezoids with which it shares upper and lower bounding segments. (Such information is of course readily obtained for any given diagram.) That is, the first algorithm will not need to preserve such information as the diagram is built. For the second and third algorithms, we assume that the diagram is represented as a planar subdivision in one of the standard ways [34], [27].

For the first algorithm, the line segments are added in random order, one by one, to a set R. The diagram $\mathcal{T}(R)$ is maintained as R grows. In addition, a conflict graph is maintained, as discussed in Section 3.3. Here the conflicts are between line segments in $S \backslash R$ and the interiors of trapezoids in $\mathcal{T}(R)$. There is a conflict between a segment and a trapezoid (interior) when they intersect. The conflict graph can be represented by lists: for each segment s the set $s \wedge \mathcal{F}^0_R$ is maintained as a list, and for each cell F of $\mathcal{T}(R)$, the set of conflicting segments

$F \wedge S$ is maintained as a list. When adding a segment s, the cells that must be deleted are found by examining $s \wedge \mathcal{T}(R)$. (Here we identify $\mathcal{T}(R)$ with the set of cells in it.) These cells are then split up by s, each into at most four pieces. Some of these pieces will merge together for new cells, since s will reduce the visibility of some points, and so shrink some vertical bounding edges.

The edges in the conflict graph to the new cells that result from s can be found by examining the lists $F \wedge S$, for the cells $F \in s \wedge \mathcal{T}(R)$. To satisfy the update condition of Theorem 3.9, we need to show that these new edges can be found in time linear in the total of the lengths of the lists $F \wedge S$ for $F \in s \wedge \mathcal{T}(R)$. The only nontrivial problem here is that when some pieces of deleted cells merge to make a new cell F, we must have a given segment conflicting with F represented only once in the list for $F \wedge S$. We can maintain this nonredundancy as follows. (We are sketchy here since the next algorithm described is superior in theory and probably in practice.) Maintain the conflict lists for segments in left-to-right order of intersection. Use this order for s to merge pieces of deleted cells in left-to-right order. Determine the conflicts for each piece. When constructing a conflict list for a new cell, examine the conflicts for a given piece, and for segment s' conflicting with that piece, walk from the current piece to the remaining ones that will merge for the new cell, deleting s' from the conflicts for those pieces. (We assume that appropriate cross-pointers between lists are maintained.)

We are ready to apply Theorem 3.9 to the analysis of this algorithm. To estimate $\tau_0(r)$, we have the following theorem. We use $|\mathcal{T}(R)|$ to refer to the number of cells of $\mathcal{T}(R)$.

Lemma 4.1. *Assume S is a set of n nondegenerate line segments in the plane, with A intersections. Let R be a random subset of S of size r. The number of cells in $\mathcal{T}(R)$ is no more than $O(r+A')$, where A' is the number of intersecting pairs of segments of R. The expected value of A' is $Ar(r-1)/n(n-1)$. Therefore $E|\mathcal{T}(R)| = O(r+Ar^2/n^2)$.*

Proof. The first statement is an obvious consequence of the fact that $\mathcal{T}(R)$ is a planar map.

Let Z_S be the set of intersection points of S, and Z_R the set of intersection points of R. For $z \in Z_S$, let $I_z = 1$ when $z \in Z_R$, and 0 otherwise. Then $A' = \sum_{z \in Z_S} I_z$, so

$$\begin{aligned} EA' &= \sum_{z \in Z_S} EI_z = \sum_{z \in Z_S} \operatorname{Prob}\{z \in Z_R\} \\ &= A\binom{r}{2} \Big/ \binom{n}{2}, \end{aligned}$$

since $z \in Z_R$ if and only if the two line segments that meet at z are both in R. □

Theorem 4.2. *For a set S of n line segments having A intersecting pairs, the trapezoidal diagram $\mathcal{T}(S)$ can be computed in $O(A+n\log n)$ expected time.*

Proof. As discussed in Section 2, $\mathcal{T}(S)$ corresponds to $\mathcal{F}_S^0$, with $b=4$ and δ defined so that $F\delta X$ for $X \in S^{(4)}$ when F is a cell in $\mathcal{T}(X)$. The nondegeneracy conditions imply that δ is functional. From the lemma above, $\tau_0(r) = O(r + Ar^2/n^2)$, and with the update condition satisfied we have that the expected time required by the algorithm is proportional to

$$n \sum_{1 \le r \le n} O(r + Ar^2/n^2)/r^2 = O(A + n \log n). \qquad \square$$

The space bound for this algorithm is certainly $O(n \log n + A)$ on the average. Moreover, at step r, the conflict graph has $O(n + Ar/n)$ edges. However, a simple example shows that the conflict graph can grow to $\Omega(n \log\log n)$ edges over the course of the algorithm, so we do not have expected space proportional to the $O(n+A)$ output size. However, we can achieve $O(n+A)$ worst-case space by a simple change as in Mulmuley's algorithm [35] and similar to that described in [16] (and below) for convex hulls. This change is to store only part of the conflict graph, only the conflicts between line-segment endpoints and trapezoids. That is, for each cell there is a list of line segments whose left endpoints are in the cell. When adding a segment, we need to traverse the current diagram along the segment to determine all of the cells conflicting with that segment. Such endpoint conflicts are readily updated, and plainly only $O(n+A)$ storage is necessary.

Unlike the first algorithm, we must prove a time bound for the traversal of the diagram by a newly added segment s. Such a traversal would walk around the boundaries of the cells that intersect s. A difficulty here is that the upper or lower boundaries of a cell F may be split into many edges, when F shares those boundaries with many cells. The following lemma shows that examining these boundary edges is not expensive on the average. Call the portion of a cell's boundary that is contained in input line segments that cell's *segment boundary* (as opposed to the vertical boundary edges of the cell).

Lemma 4.3. *The expected number of edges that bound trapezoids conflicting with added segment s at step r is $O(1 + Ar/n^2)$.*

From this lemma we have a total expected time $O(n+A)$ for determining conflicts over the course of the algorithm.

Proof. By the nondegeneracy assumption, every trapezoid shares its vertical boundaries with $O(1)$ other trapezoids, so we are mainly concerned with edges on segment boundaries. We bound, equivalently, the expected number of cell corners that appear on a trapezoid's segment boundary. We apply Theorem 3.7, but using a set $\mathcal{F}_S$ and defining relation δ that is different from that used above. Here $b=6$, and the region of interest defined by a set $X \in S^{(6)}$ consists of the interior of a trapezoidal diagram cell defined by four or fewer of the segments of X, together with a vertical edge with an endpoint on the cell's segment boundary. The other endpoint of the vertical edge is either the endpoint of a line segment in X, or the intersection point of two segments in X. That is, $\mathcal{F}_S^0$ consists of pairs of trapezoids Q and vertical edges e, where e is a boundary of a trapezoid sharing an upper or lower boundary with Q. To prove the lemma, we want to bound the expected number of ranges in $\mathcal{F}_R^0$ at step $r+1$ for which the trapezoid of the range meets s. The expected value of this quantity is no more than

$$\sum_{F\in\mathcal{F}_S} \operatorname{Prob}\{F\in\mathcal{F}_R^0,\, s\in\mathcal{F}\wedge S\},$$

or

$$\sum_{F\in\mathcal{F}_S} \frac{|F|}{n-r}\operatorname{Prob}\{F\in\mathcal{F}_R^0\} = \frac{1}{n-r}\,\frac{n-r}{r}\,O(\tau_0(r))$$

by Theorems 3.7 and 3.6 and their proofs. The lemma follows, since $|\mathcal{F}_R^0|$ is no more than twice the number of vertical edges in $\mathcal{T}(R)$; from Lemma 4.1 and planarity of $\mathcal{T}(R)$, we have $E|\mathcal{F}_R^0| = O(r+Ar^2/n^2)$, which also bounds $\tau_0(r)$, and so gives the lemma. □

To improve the space bound further, we use one of the above algorithms as a major step in a third algorithm. The third algorithm uses random sampling of line segments to apply divide-and-conquer to the problem. The algorithm is recursive, but has a recursion depth of only 2.

The idea is to find all intersecting pairs of S by finding all intersecting pairs of $F\wedge S$, for $F\in\mathcal{T}(R)$. (Also including the intersecting pairs in R.) To show that this approach is helpful, we use the following corollary.

Corollary 4.4. *Assume S is a set of n line segments in the plane, with A pairs of these segments intersecting. Let R be a random subset of S of size r. There exist constants $K_{\max}$ and K_{tot} such that, with probability at most* $1/4$,

$$K_{\text{tot}}(n+Ar/n) \le T_1(R) = \sum_{F\in\mathcal{T}(R)} |F|,$$

and, with probability at most $1/n^{10}$,

$$K_{\max}(n/r)\log n \le \max_{F\in\mathcal{T}(R)} |F|.$$

Proof. The proof follows that of Corollary 3.8. □

If the second inequality does not hold, the sample R is said to be "good." We use good samples for divide-and-conquer. Let N denote the size of the top-level input set, and use n for the input size at the general step of the recursion. The scheme of the algorithm is to find a good sample of size $r=\sqrt{n}$, compute the sets $F\wedge S$, and recur. The recursion stops when an input set has no more than $\sqrt{N}$ segments; since the input size n is N at the top, $O(\sqrt{N}\log N)$ at the next, and then $O(N^{1/4}\log^{3/2} N)$, the recursion depth is 2.

The key question here is: how can we find the sets $F \wedge S$ using only $O(n)$ storage? Our algorithm employs two methods to do this; both methods begin by choosing R at random. Method I then simply checks all segments in $S \backslash R$ with all cells in $\mathcal{T}(R)$, tallying up the values $|F|$, checking if R is a good sample. When R is found not to be a good sample, another is chosen, repeating until a good sample is found.

Method II also chooses a random sample $R \subset S$ of size r, and builds $\mathcal{T}(R)$. However, as in the second basic method above, while $\mathcal{T}(R)$ is built, the locations of the endpoints of $S \backslash R$ are maintained. The sets $F \wedge S$ are then found by walking along each segment in $S \backslash R$, finding conflicts as in the second basic algorithm. If at any time, either R is found not to be a good sample, or the total number of conflicts found exceeds $2K_{\text{tot}}n$, method II restarts with another sample.

These two methods are run concurrently until one stops with a good sample. The algorithm then recurs. (If method I succeeded, the sets $F \wedge S$ are generated again for each $F \in \mathcal{T}(R)$ in turn.)

Before proving bounds for this algorithm, we make the following simple observation.

Lemma 4.5. *For X a nonnegative random variable and event B, we have $E[X|B] \le EX/\text{Prob}\{B\}$.*

Proof. Easy. □

Theorem 4.6. *Assume S is a set of N nondegenerate line segments in the plane, with A pairs of these segments intersecting. The pairs of line segments of S that intersect can be computed in $O(N)$ space and $O(A+N \log N)$ expected time.*

Proof. The space bound is obvious. For the time bound, we first show bounds for methods I and II to divide the problem. We bound the work for method I when $A > n\sqrt{n}$, and for method II when $A \le n\sqrt{n}$. (Here A represents the number of intersecting pairs in the current subproblem.)

The expected time required by method I is $O(A+nr)$, since the time needed to check all segments against all $F \in \mathcal{T}(R)$ is proportional to $n(r+A')$, where A' is the number of intersection points of R. (The expected time $O(A+n \log n)$ needed by the second algorithm to compute $\mathcal{T}(R)$ is dominated by $O(A+nr)$, remembering that $r=\sqrt{n}$.) From Lemma 4.1, $EA' = O(Ar^2/n^2) = O(A/n)$. We must use, however, the expected value of A', conditioned on R being a good sample. By Lemma 4.5, the bound $EA' = O(A/n)$ still holds with this condition. Therefore, method I requires $O(nr+A)$ to check a given sample, which is $O(A)$ for $A > nr$. Since sample R is bad with probability no more than $1/n^{10}$, we have that the expected time for method I to find a good sample is $O(nr+A)(1+1/n^{10}+1/n^{20}+\cdots) = O(nr+A)$.

The expected time for method II to construct $\mathcal{T}(R)$ is $O(A+n \log n)$; this includes the work to maintain the locations of endpoints of $S \backslash R$. We need to bound the time required to construct the sets $F \wedge S$ by walking along the segments of $S \backslash R$. To do this we use the same relation δ as in Lemma 4.3. The work for

$F \in \mathcal{F}_R^0$ is $|F|$, so, by Theorem 3.7, the total work to build the sets $F \wedge S$ is $O(n/r)\tau_0(r)$, or $O(n/r)O(r+Ar^2/n^2) = O(n+Ar/n)$. For $A \le nr$, this is $O(n)$. Thus the expected work to find the sets $F \wedge S$, for any given random subset R, is $O(A+n \log n)$.

If $A \le nr$, then by Lemma 4.4, with probability 3/4, the total $\sum_{F \in \mathcal{T}(R)} |F|$ is no more than $K_{\text{tot}}(n+Ar/n) \le 2K_{\text{tot}}n$. Combining this with the condition that R is a good sample, the probability that method II will need to restart with a new sample is no more than $P = 1/4 + 1/n^{10}$. By Lemma 4.5, the expected time to construct the sets $F \wedge S$, given that sample R does not require method II to restart, is $O(A+n \log n)/(1-P) = O(A+n \log n)$. Thus method II requires $O(A+n \log n)(1+P+P^2+\cdots) = O(A+n \log n)$ expected time.

Since we run methods I and II concurrently until one succeeds, the total time to run them is no more than twice the minimum time of the two, and so is $O(A+n \log n)$ for all values of A. Now a two-level induction completes the proof of the theorem: the base case is obvious, and the work at the top level is

$$O(A+n \log n) + \sum_{F \in \mathcal{T}(R)} A_F + |F| \log|F| \le O(A+n \log n) + O(\log n) \sum_{F \in \mathcal{T}(R)} |F|,$$

where A_F is the number of intersections in cell F. Finally, the sum $T_1(R)$ above is $O(n)$ if method II succeeded, and its expected value is $O(Ar/n)$ even when conditioned on the success of method I, by Lemma 4.5. Since the induction has only two levels, multiplication by constant factors at each level does not alter the asymptotic bound, and so the theorem follows. □

4.2. Convex-Hull Algorithms

In this section we give algorithms for computing the convex hull of a set of sites in E^3, or, equivalently, of determining the intersection $\mathcal{P}(S)$ of a set S of n closed halfspaces in E^3. We assume that the bounding planes of the halfspaces are nondegenerate, so that no four intersect at a common point. We also extend these results to higher dimensions.

To fit in our framework, we use the open halfspaces $\bar{H}$, the complements of $H \in S$, and refer to this collection as $\bar{S}$. The intersection $\mathcal{P}(S)$ is the set of points not in $\bigcup_{\bar{H} \in \bar{S}} \bar{H}$.

One algorithm we give is randomized and incremental. As in Section 1.3, it adds the halfspaces one by one in random order to a set R, maintaining $\mathcal{P}(R)$ as each halfspace is added. To make the algorithm faster, some additional information is used: we maintain a conflict graph, which is a bipartite graph on the halfspaces of S and the edges of $\mathcal{P}(R)$, with a graph edge between a halfspace $H \in S$ and an edge $e \in \mathcal{P}(R)$ when e is not contained in H. This graph can be represented by linked "conflict lists" that give the sets $e \wedge \bar{S}$ for $e \in \mathcal{P}(R)$, and $\bar{H} \wedge \mathcal{P}(R)$ for $H \in S \backslash R$, where we identify $\mathcal{P}(R)$ with its set of edges.

In the general step of the algorithm, a halfspace H is added to R, making the new set $R' = R \cup \{H\}$. At this step, the edges of $\mathcal{P}(R)$ that are retained in $\mathcal{P}(R')$

are those *not* in $\bar{H} \wedge \mathcal{P}(R)$. Some edges in $\bar{H} \wedge \mathcal{P}(R)$ have both vertices not in H. Such edges can be discarded. The remaining edges in $\bar{H} \wedge \mathcal{P}(R)$ are cut by the bounding plane of H. A facet G of $\mathcal{P}(R)$ that is cut by the bounding plane of H is incident to two such edges e_1 and e_2. Such a facet gives rise to a new facet G' of $\mathcal{P}(R')$, bounded by edges of G not in $\bar{H} \wedge \mathcal{P}(R)$, and by new edges $e_1' = e_1 \cap H$, $e_2' = e_2 \cap H$, and e_{12}, which is the intersection of G with the bounding plane of H. The edge e_{12} is also incident to the face of $\mathcal{P}(R')$ that is the intersection of $\mathcal{P}(R)$ with the bounding plane of H.

To update the representation of $\mathcal{P}(R')$ to reflect these changes, we must find, for each edge e_1 cut by the bounding plane of H, the faces F incident to e_1 and the edge e_2 incident to F that is also cut. This is easily done by walking around F from e_1 to e_2, staying outside of H, so that the edges traversed are in L_H. In this way, the polytope $\mathcal{P}(R')$ and the changes in incidence relations are obtainable in time proportional to the number of edges in L_H.

It is easy to see that the halfspaces in the conflict lists for the three edges e_1', e_2', and e_{12} are contained in the conflict lists of e_1 and e_2. (Any halfspace that contains these two edges also contains their convex hull, which includes the new edges.) These lists are searched to find the conflict lists for the three new edges. The conflict lists of all new edges in $\mathcal{P}(R)$ are found in this way.

This is the entire algorithm, assuming appropriate initialization. A moment's thought shows that when adding a halfspace $H \in S$, the work performed is proportional to the total number of halfspaces in the conflict lists of the edges in the conflict list of H. By Theorem 3.9 we have

Theorem 4.7. *Given a nondegenerate set S of n halfspaces in E^3, the randomized incremental algorithm computes $\mathcal{P}(S)$ in $O(n \log n)$ expected time.*

Proof. We apply Theorem 3.9 with $b = 4$; the objects are the halfspaces in $\bar{S}$, the ranges $\mathcal{F}$ are line segments in E^3, and e is defined by a set X of halfspaces if e is an edge of the intersection of the closed complements of the halfspaces in X. As discussed above, the update condition holds, and $\tau_0(r) = O(r)$. □

We also have the following extension.

Theorem 4.8. *Given a nondegenerate set S of n halfspaces in E^d, a randomized incremental algorithm computes $\mathcal{P}(S)$ in $O(n^{\lfloor d/2 \rfloor})$ expected time.*

Proof. The algorithm is similar to that above; we maintain $\mathcal{P}(R)$ as halfspaces are added to R. The *incidence graph* of $\mathcal{P}(R)$ is maintained, as in the beneath-beyond method [20], and the conflict lists of edges (one-dimensional faces) are maintained. To find conflict lists for new edges (corresponding to edge e_{12} above), we must find pairs of edges e_1 and e_2 on the same 2-face (polygonal region) and cut by the bounding hyperplane of an added halfspace. To do this, put each edge cut by the bounding hyperplane into a list L_G for each incident 2-face G, and maintain a list of cut 2-faces. Having done this for all conflicting edges, the desired pairs are those in the two-element lists L_G for cut 2-faces G.

The application of Theorem 3.9 is similar to that for the previous theorem, with $b=d+1$. □

A Linear-Space Variant. A simple example, where $\mathcal{P}(S)$ is the dual of a cyclic polytope [20], shows that the above algorithm requires $\Omega(n \log\log n)$ expected space. We next give a variant algorithm for E^3 that requires $O(n)$ space in the worst case, with the same $O(n \log n)$ expected-time bound.

The variant is as follows: rather than maintain the entire conflict graph, it is enough to maintain, for each halfspace H not yet added, a single edge $C(H)$ with which it conflicts. When adding H, we can quickly determine all the edges with which it conflicts, by searching the edges of $\mathcal{P}(R)$ starting at $C(H)$. (Note that the set of conflicting edges gives a connected subgraph of the 1-skeleton of $\mathcal{P}(R)$.) By our nondegeneracy assumption, each vertex of the polytope is incident to three edges, so this searching requires constant time per conflicting edge.

This variant has a slightly different update problem: suppose a halfspace H is to be added, after which edge e will no longer be present in the intersection. If $e=C(H')$ for some halfspace H', we must find some edge e' in $\mathcal{P}(R')$ that conflicts with H'. To do this, we search the edges of $\mathcal{P}(R)$ starting at e, maintaining the condition that the edges we examine conflict with H'. At some point, we find an edge that conflicts with H' and also either does not conflict with H, or is cut by the bounding plane of H. In the former case, we have an edge of $\mathcal{P}(R')$ that conflicts with H', and we can reset $C(H')$. In the latter case, we are led to an edge of $\mathcal{P}(R')$ that is not in $\mathcal{P}(R)$, and that may conflict with H'. If the new edge conflicts, we are done. Otherwise, we continue searching the edges that conflict with H'. If we never find an edge of $\mathcal{P}(R')$ that conflicts with H', we may ignore H' in later processing. Otherwise, we have updated $C(H')$, and have done no more work than the original algorithm.

4.2.1. An Output-Sensitive Algorithm. This section gives an output-sensitive algorithm for computing $\mathcal{P}(S)$. We assume that a point p_* in the intersection is known; such a point can be found in linear time using linear programming [33], [13].

The main idea of the algorithm is to filter out quickly those halfspaces in S that contain $\mathcal{P}(S)$ in their interiors. Such halfspaces are *redundant*, and removing them gives a set of *irredundant* halfspaces S' with $\mathcal{P}(S)=\mathcal{P}(S')$. Furthermore, if the descriptive complexity of $\mathcal{P}(S)$ is A, then there are certainly no more than A halfspaces in S'. (Dually, we are quickly removing points that are inside the hull.)

The algorithm is based on an expected relation between the intersection $\mathcal{P}(R)$ of random $R \subset S$, and the intersection $\mathcal{P}(S)$. Assume $\mathcal{P}(R)$ is split up into simple pieces as follows: take some arbitrary (fixed) plane h, and cut each face of $\mathcal{P}(R)$ into trapezoids using the translates of h that pass through the vertices of the face. Decompose $\mathcal{P}(R)$ into a set of simple regions $\Delta(R)$, each region consisting of the convex closure of p_* with a trapezoid from the cutting of the faces. For $F \in \Delta(R)$, the set $F \wedge \bar{S}$ corresponds to the set of halfspaces of S that do not contain F entirely in their interiors, and $|F|$ denotes $|F \wedge \bar{S}|$. The following lemma is a corollary of Theorem 3.7.

Lemma 4.9. *The expected value of* $\sum_{F \in \Delta(R)} |F|$ *is* $O(n)$.

Proof. The objects are open halfspaces, the complements of the input halfspaces. The parameter b is five: one halfspace determines the face containing a trapezoid, two more halfspaces determine the two edges on that face that determine the trapezoid, and two more determine the vertices of the trapezoid. The ranges are polyhedra with p_* as one vertex and up to five sides. Nondegeneracy implies that δ is functional. Also $\tau_0(r) = O(r)$: the trapezoidal subdivision of the surface of $\mathcal{P}(R)$ forms a planar graph. □

Note that $\mathcal{P}(S)$ consists of the union of the regions $F \cap \mathcal{P}(S)$, over all $F \in \Delta(R)$. The halfspaces of S that contribute to a nontrivial region $F \cap \mathcal{P}(S)$ are the complements of those in $F \wedge \bar{S}$.

Since every irredundant halfspace determines a vertex of $\mathcal{P}(S)$, we need not consider all the regions $F \in \Delta(R)$, only those that contain at least one vertex of $\mathcal{P}(S)$. Call this set of regions $\Delta^*(R)$. Certainly $\Delta^*(R)$ contains at most A regions. The following lemma is a corollary of Theorem 3.7.

Lemma 4.10. *The expected value of* $\sum_{F \in \Delta^*(R)} |F|$ *is* $O(n/r)A$, *for sample size* r.

Proof. As in the previous lemma, where the δ relation is defined only for ranges that contain a vertex of $\mathcal{P}(S)$. Plainly $\tau_0(r) \leq A$. □

Now suppose the sample size r is at least A^2. Then $\sum_{F \in \Delta^*(R)} |F|$ is $O(n/A)$ on the average. This observation provides a fast means of filtering out redundant halfspaces, making two assumptions: we can obtain an estimate of A, and we have a fast means of determining $\Delta^*(R)$ and the sets $F \wedge \bar{S}$ for all $F \in \Delta^*(R)$. We next consider these two problems.

The regions $F \in \Delta(R)$ and the corresponding $F \wedge \bar{S}$ can be readily obtained in $O(n \log r)$ expected time using the randomized incremental algorithm given above. To determine $\Delta^*(R)$ from $\Delta(R)$, we must have a fast means of determining the regions F that contain no vertices of $\mathcal{P}(S)$ or, conversely, the regions that contain only parts of faces or edges. This is done as follows: let t be a triangular face of a region $F \in \Delta(R)$, with p_* a vertex of t. Then, the polygon $P_t = t \cap \mathcal{P}(S)$ can be determined using the algorithm of Kirkpatrick and Seidel [31] in time on the order of $|F| \log A_t$, where A_t is the number of sides of P_t. All but two of the sides of P_t correspond to faces of $\mathcal{P}(S)$, so that the total time to compute all such polygons is expected $O(n \log A')$, where A' is the total number of faces of $\mathcal{P}(S)$ identified. If a region $F \in \Delta(R)$ contains no vertices of $\mathcal{P}(S)$, the polygons corresponding to faces of F completely determine the structure of $F \cap \mathcal{P}(S)$, and this can be verified or disproven in $O(|F|)$ time. Thus the regions in $\Delta^*(R)$ and their corresponding halfspaces can be found in $O(n)(\log r + \log A')$ time.

Now to consider the problem of estimating A, or, rather, of using only lower bounds for A. To do this, determine $\Delta^*(R)$ for a sequence of sample sizes, using at each step an estimate A^* of A. Initially, the A^* value is some constant, say 10. In the general step, we have an estimate A^* and a set of halfspaces S still being considered. If $A^* > |S|$, then compute $\mathcal{P}(S)$ using the randomized incremental algorithm. Otherwise, we compute $\Delta^*(R)$ and the sets $F \wedge \bar{S}$ with $r = |R| = A^*$, and include in S those halfspaces with complements in $\bigcup_{F \in \Delta^*(R)} F \wedge \bar{S}$. Suppose that at least half of the current halfspaces are eliminated in this way. Then the current value of A^* is retained for another iteration. Otherwise, assign $A^* \leftarrow (A^* A')^2$.

Theorem 4.11. *Given a set S of n halfspaces in E^3, where $\mathcal{P}(S)$ has A vertices, the intersection $\mathcal{P}(S)$ can be computed in $O(n \log A)$ time.*

Proof. The time needed for the general step of the algorithm is dominated by the time to compute $\mathcal{P}(R)$ and the sets $F \wedge \bar{S}$ for $F \in \Delta^*(R)$. As noted above, this is expected $O(n)(\log r + \log A')$, where $r = A^*$ and A' is a lower bound on A. Let A_i^* denote the ith value assigned to A^*. During a period that $A^* = A_i^*$, the number of halfspaces is cut in half at each step except the last one, so the expected time to perform the steps during that period is within a constant factor of

$$(n + n/2 + n/4 + \cdots) \log A_i^* A' = 2n \log A_i^* A',$$

where A' here denotes the largest value assumed by that variable during the period. Since $\log A_{i+1}^* \geq 2 \log A_i^* A'$, the total work done before $A^* = A_i^*$ is expected $O(n \log A_i^*)$. Let $i_0 = \max\{i | A_i^* \leq A^2\}$. Then the expected work for A_i^* with $i \leq i_0$ is $O(n \log A)$.

We must still bound the work for A_i^* with $i > i_0$. By Lemma 4.10, the expected proportion of halfspaces eliminated is $1 - O(A/A_i^*)$; by Markov's inequality, the probability that the number of halfspaces is not cut in half is bounded above by a quantity proportional to A/A_i^*. That is, the probability that we will perform the general step with $A^* = A_i^*$, and then immediately reassign A^* as A_{i+1}^*, is no more than $O(A/A_i^*)$. The work done before A^* is *not* reassigned is therefore bounded above by $\sum_{i \geq i_0} (A/A_i^*) O(n \log A_{i+1}^*)$. This rapidly converging sum is dominated by its leading term, and, since $A_i^* > A^2$ for $i > i_0$, we have $O(n)$ expected work before the number of halfspaces is cut in half. The total work for $i > i_0$ is $O(n + n/2 + n/4 + \cdots) = O(n)$, which completes the proof. □

4.3. *Spherical Intersections and Diameter*

In this section we give an algorithm to compute the intersection of a set of fixed-radius balls in E^3, and apply this algorithm to the problem of computing the diameter of a point set in E^3.

Such an intersection is called a *spherical intersection.* In general, given $S \subset E^d$, the ρ-spherical intersection of S, or $\mathcal{I}_\rho(S)$, is the intersection of the closed balls of radius ρ that have centers at the sites of S. Spherical intersections have many properties in common with convex polytopes, which are the intersections of sets of closed halfspaces. Like polytopes, spherical intersections are convex, and have vertices, edges, and faces, that in E^3 naturally define a planar graph. That graph has $O(n)$ descriptive complexity [29]. Like polytopes, spherical intersections have duals, which were introduced as *α-hulls* in [21].

Unfortunately, spherical intersections do not share with convex polytopes some properties helpful for algorithms. In particular, the divide-and-conquer technique of Preparata and Hong [36] does not seem to lead to a fast algorithm for computing spherical intersections. Our simple algorithm for spherical intersections, requiring $O(n \log n)$ expected time, is asymptotically faster than any previous algorithm.

The spherical intersection problem arises in a classic problem of computational geometry, the diameter problem. Let $S \subset E^d$ contain n sites (points). The *diameter* D_S is the largest distance between a pair of sites. A *diametral pair* of S is a pair of sites that realize the diameter. The problem of determining the diameter (and all diametral pairs) of a point set in the plane has long been known to require $\Theta(n \log n)$ time [37]. In E^3, the number of diametral pairs of n sites is known to be $O(n)$, as an easy consequence of the fact that the D_S-spherical intersection has $O(n)$ descriptive complexity. This suggests that the diameter problem in E^3 should not be too much harder than for E^2. However, obtaining an algorithm for E^3 with complexity close to $O(n \log n)$ "has been a source of frustration to a great many workers" [37] in computational geometry. Our algorithm requiring $O(n \log n)$ expected time improves on the best algorithms previously known, that have worst-case time bounds no better than $O(n\sqrt{n} \log n)$ [2].

Our algorithm for spherical intersection is very similar to the incremental algorithm in Section 4.2, except that instead of adding halfspaces one by one, we add closed balls of radius ρ. The objects of Section 2 are the complements of these balls, since we want the edges defined by the balls that are contained in all the balls. The geometric fact necessary to the correctness of this algorithm is given in the following lemma.

Lemma 4.12. *In the spherical intersection algorithm, the balls in the conflict lists of the three new edges bounding a face are contained in the conflict lists of the deleted edges bounding that face.*

The proof is expressed in terms of unit balls, but obviously holds for balls of any given radius.

Proof. Assume the edges involved are on an old face F and a new face F', with B the newly added ball, so that $F' = F \cap B$. Then two edges e_1 and e_2 that are cut by the sphere bounding B give two new edges e_1' and e_2', and another new edge e_{12} which is the intersection of F with the bounding sphere of B. Since $e_1' \subset e_1$ and $e_2' \subset e_2$, so it is only necessary to show the containment for e_{12}.

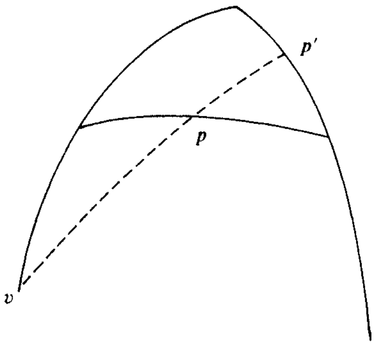

Fig. 1. A vertex v and a point p on new edge e.

The lemma is proven using the following fact: for points x, y on a unit sphere, if point z is no farther than 1 from x and y (in straight-line distance), then z is no farther than 1 from all points on the (minor) great circle arc connecting x and y. This fact is easily proven.

Now assume v is a vertex of a deleted edge and also a vertex of a new edge, so v is within B. (See Fig. 1.) For a point p on e, the great circle arc between p and v is contained in F', using the geometric fact. If this arc is continued past p, it will reach a point p' on a deleted edge. Again using the geometric fact, any ball that contains v and p' will contain p. Thus if p is outside any ball, then either v or p' is, and that ball's complement is on the conflict list for the edge containing v or the edge containing p'. □

This lemma implies that the update condition for randomized incremental construction of spherical intersections is satisfied. To make the appropriate δ relation functional, we require nondegeneracy, which means here that no four input points are on the same sphere of radius ρ.

Theorem 4.13. *Given a nondegenerate set S of n points in E^3, their spherical intersection $\mathcal{I}_\rho(S)$ can be computed in $O(n \log n)$ expected time.*

Proof. With the above lemma, the update condition holds; the objects are in a set $\bar{S}$, the set of complements of closed balls of radius ρ about the points S; the δ relation has $F\delta X$ for $X \in \bar{S}^{(4)}$ when F is an edge in $\mathcal{I}_\rho(X')$, where X' is the set of points corresponding to the regions in X. Nondegeneracy implies that δ is functional. As noted above, the results of [29] imply that $\tau_0(r) = O(r)$. □

Next we give a reduction from the diameter problem to the spherical intersection problem. The idea is this: let D_p denote the farthest distance from point p to a site in S. Let $\rho = D_p$ for some $p \in S$. Any point $q \in S$ that is in the interior of $\mathcal{I}_\rho(S)$ is closer than D_p to all points in S. The point q has $D_q < D_p \leq D_S$, and so q is not in any diametral pair. On the other hand, if $q \in S$ is outside $\mathcal{I}_\rho(S)$, then $D_S \geq D_q > D_p$. Thus, if there are no points of S outside $\mathcal{I}_\rho(S)$, then $D_S = D_p$, and if there are any such points, only those points can possibly be in diametral pairs.

Based on these observations, we have an algorithm. Perform the following loop: choose $p \in S$ at random. Compute D_p and the intersection $\mathscr{I}_\rho(S)$ for $\rho = D_p$. Find the points of S outside $\mathscr{I}_\rho(S)$; if there are none, we have the diametral pairs using $\mathscr{I}_\rho(S)$. If there are points outside $\mathscr{I}_\rho(S)$, assign $S \leftarrow S \backslash \mathscr{I}_\rho(S)$.

To find the set $S \backslash \mathscr{I}_\rho(S)$, we use an algorithm for point location in a planar subdivision (see [37] and [20]). To do this, define a "stereographic projection" for $\mathscr{I}_\rho(S)$ as follows: pick a point p on the boundary of $\mathscr{I}_\rho(S)$, and a sphere Z determining the face of $\mathscr{I}_\rho(S)$ containing p. Let h denote the tangent plane to Z at p. Let p' be the point antipodal to p on Z, and let h' be the tangent plane to Z at p'. Define a function $F(x)$ from E^3 to h', by mapping a point x to the intersection point with h' of the ray from p passing through x. (If x is on the other side of h from Z or on h, then $F(x)$ is undefined, and $x \notin \mathscr{I}_\rho(S)$; this is checked for all $s \in S$ in constant time per point.) The set of points that are the image under F of the edges of $\mathscr{I}_\rho(S)$ naturally induces a subdivision of H. Now in $O(n \log n)$ time, build a data structure for determining the location of a point on that subdivision. For each point $s \in S$, determine in $O(\log n)$ time the location of $F(s)$ (if $F(s)$ is defined). The region of the subdivision that contains $F(s)$ corresponds to a face of $\mathscr{I}_\rho(S)$, and $s \in \mathscr{I}_\rho(S)$ if and only if the line segment $\overline{ps}$ does not pass through the boundary of that face. We can also determine if s is on a face of $\mathscr{I}_\rho(S)$. If so, and $\rho = D_S$, then s forms a diametral pair with the site corresponding to the face.

We have an optimal time bound for the algorithm:

Theorem 4.14. *Given a set $S \subset E^3$ of n nondegenerate points, the diametral pairs of S can be determined in $O(n \log n)$ expected time.*

Proof. Suppose the points of S could be listed in nonincreasing order of their D_p values. Then when $p \in S$ is chosen at random, with probability $1/n$ its rank in that list is k, for $1 \leq k \leq n$, so that at most $k-1$ points of S need be considered in any further computation of the diameter. From Theorem 4.3, $\mathscr{I}_\rho(S)$ can be computed in $O(n \log n)$ expected time, and $O(n \log n)$ time suffices to find $S \backslash \mathscr{I}_\rho(S)$, using an optimal algorithm for point location. Thus t_n expected time is enough to determine the diameter of S, where

$$t_n \leq O(n \log n) + \sum_{1 \leq k < n} t_k / n.$$

It is readily verified that $t_n = O(n \log n)$. □

4.4. *Algorithms for Halfspace Range Queries*

In this section the new $(\leq k)$-set bound gives an improved storage bound for a deterministic algorithm for halfspace range search in E^3. Two new randomized algorithms for range search in E^d are given and analyzed. For convenience we assume that the input points are in general position, that is, no $d+1$ are coplanar.

Given n points $S \subset E^d$, the halfspace range-search problem is to build a data structure for S so that given a query halfspace h^*, the set of points $h^* \cap S$ can be reported quickly. In [7] Chazelle and Preparata show that in the case $d=3$, a data structure requiring $O(n(\log n)^8(\log\log n)^4)$ storage can be constructed that allows queries to be answered in $O(A+\log n)$ time, where A is the number of points in the answer to the query. Theorem 1 of that paper implies that if $\hat{g}_{k,3}(n)=O(nk^\beta)$, then the storage required by their data structure is $O(n(\log n)^{2(\beta-1)}(\log\log n)^{\beta-1})$. The bound given here implies that $O(n(\log n)^2\log\log n)$ storage is sufficient.

The upper bound on $\hat{g}_{k,d}(n)$ gives a bound on a randomized algorithm for halfspace range search in E^d. This algorithm is conveniently described by putting the range-query problem into a dual form, using the transform $\mathscr{D}$ described in Section 3.1 of [20], to which the reader is referred for background. Given point $p=(\pi_1,\ldots,\pi_d)$, $\mathscr{D}(p)$ is the hyperplane of points $x=(x_1,\ldots,x_d)$ satisfying $x_d=2\pi_1x_1+\cdots+2\pi_{d-1}x_{d-1}-\pi_d$. Thus $\mathscr{D}$ maps points in E^d to nonvertical hyperplanes in E^d. (Here "nonvertical" means that the hyperplane does not contain a vertical line. A vertical line is a translate of the x_d-axis.) We also have $\mathscr{D}$ map nonvertical hyperplanes to points, so $\mathscr{D}(h)$ for hyperplane h is the point p such that $\mathscr{D}(p)=h$. Under this duality, incidence and order are preserved, so that point p is on plane h if and only if $\mathscr{D}(h)\in\mathscr{D}(p)$, and also p is in the halfspace h^+ if and only if $\mathscr{D}(h)\in\mathscr{D}(p)^+$. In this setting, a point set S gives rise to an arrangement of hyperplanes $\mathscr{A}$ by the duality transform. Given a query halfspace h^-, the answer to the query is the set of all hyperplanes $\mathscr{D}(p)$ in $\mathscr{A}$ such that $\mathscr{D}(h)$ is below them, that is, $\mathscr{D}(h)\in\mathscr{D}(p)^-$.

We are interested in the set of all points that are below no more than k hyperplanes of $\mathscr{A}$, for some k. These points correspond to the set of all query halfspaces h^- whose answer set has no more than k points. The lower surface of this set of points is called a k-level. It is not too hard to see that the number of vertices of cells above a k-level is bounded by $\hat{g}_{k,d}(S)$. This value asymptotically bounds the total complexity of the cells of $\mathscr{A}$ above the k-level.

The main idea for the range-search algorithm is the following generalization and restatement of Lemma 5.4 of [11].

Lemma 4.15. *Assume $S\subset E^d$ in general position, with $|S|=n$. Let $R\subset S$ be a random subset of size r. Then there is an integer $j_*=O(\log r/\log\log r)$ and a value $\alpha_*=O(\log r/r)$, such that, with probability at least $1/2$, the following holds: the j_*-level of R is below the $n/(r-d^2)$-level of S, and strictly above the α_*n-level of S.*

Proof. (Sketch.) Consider any simplex T whose vertices are those of a given polytope in the j_*-level of R. The simplex T has a j_*/r proportion of the halfplanes of R above it. This is good evidence that the proportion of hyperplanes of S above T is more than $1/r$ and smaller than $(\log r)/r$. This can be made precise by appealing to Corollaries 4.3 and 4.4 of [11], in the same way as done in the proof of Lemma 5.4 of that paper. □

Let us assume that r is constant (though "sufficiently large"). A given random subset can be tested for satisfying the conditions of Lemma 4.15 in $O(\hat{g}_{j_*,d}(r)n)$

time, which is $O(n)$ for fixed r. Thus, by repeatedly sampling, a suitable sample can be found in two trials, on the average.

Suppose that for a query halfspace h^-, the point $\mathcal{D}(h)$ is below the j_*-level of R, where R is now a suitable sample. (We assume that the query halfspace contains the $-\infty$ point of the x_d-axis. Symmetric processing must be done for positive halfspaces.) Then $\mathcal{D}(h)$ is below the $n/(r-d^2)$-level of $\mathcal{A}$, and the query has answer size $A=\Omega(n)$. Here sophistication does not pay, and linear search through S determines the answer in $O(n)=O(A)$ time. On the other hand, suppose $\mathcal{D}(h)$ is above or on the j_*-level of R. Then recursively search a data structure for S, which is built as follows: triangulate the polyhedral cells of the j_*-level of R. By the results of [11], this yields $O(\hat{g}_{j_*,d}(r))$ simplices, as $r\to\infty$. (The triangulation here involves simple pieces that are simplices when bounded; the unbounded pieces can be viewed as simplices with vertices "at infinity.") For any simplex T, let $S(T)$ be the set of hyperplanes of $\mathcal{A}$ that are above any point of T. Recursively build a search structure for $S(T)$, for all such simplices T. To answer a query when $\mathcal{D}(h)$ is above the j_*-level of R, search the data structure for the simplex containing the vertical downward projection of $\mathcal{D}(h)$.

A necessary observation here is that since R satisfies the conditions of Lemma 4.15, each simplex T on the j_*-level of R has vertices that have no more than $\alpha_* n$ hyperplanes above them. Any hyperplane above a point in T is also above some vertex of T, so $|S(T)|$ is no more than $d\alpha_* n$.

That fact implies that our data structure has a query time of $O(A+\log n)$, and a storage bound $B(n)$ satisfying

$$\begin{aligned} B(n) &\le O(n)+O(\hat{g}_{j_*,d}(r))B(d\alpha_* n) \\ &\le O(n)+O(r^{\lfloor d/2\rfloor} j_*^{\lceil d/2\rceil})B(O(\log r)n/r), \end{aligned}$$

as $r\to\infty$, using Theorem 3.1. The given bound $O(n^{\lfloor d/2\rfloor+\varepsilon})$ follows. The expected time required by the algorithm to build the data structure satisfies the same bound.

Lemma 4.15 and the upper bound on $\hat{g}_{k,d}(n)$ can be applied in another way to obtain an algorithm for halfspace range queries that requires less storage and preprocessing, at the cost of a longer query time. Consider the arrangement $\mathcal{A}'_{j_*}$ of hyperplanes defined by the j_*-facets of R. These hyperplanes are the duals of the vertices of the cells of $\mathcal{A}$ that are on the j_*-level of $\mathcal{A}$ (or, symmetrically, on the $(n-j_*)$-level). It will be convenient later to include in $\mathcal{A}'_{j_*}$ some additional hyperplanes: these hyperplanes are vertical, contain $d-1$ points of S, and have j_* points of S on one side. Such hyperplanes correspond dually to the "endpoints" at infinity of unbounded edges of $\mathcal{A}$ on the j_*-level (or on the $(n-j_*)$-level).

The cells of $\mathcal{A}'_{j_*}$ induce a partition of S. For each cell $\mathcal{C}$, we recursively build a data structure for the points $\mathcal{C}\cap S$.

In answering a query h^-, as above we check if $\mathcal{D}(h)$ is below the j_*-level of R. If so, the answer size is more than $n/(r-d^2)$, so a naive algorithm should be used. If $\mathcal{D}(h)$ is above the j_*-level of R, the answer is smaller and the data structure must be used. The cells of the arrangement $\mathcal{A}'_{j_*}$ are examined. Some do not intersect the query hyperplane, and so contribute either all or none of the points they contain to the answer. The remaining cells, that do meet the query hyperplane, must be examined recursively.

From previous analysis [28], [4], there are two key properties of this algorithm that imply a bound on the query time. The first is that the number of cells cut by a given query hyperplane is $O\left(\binom{g_{l_*,d}(r)}{d-1}\right)$, the complexity of the subdivision of the query hyperplane by the hyperplanes of $\mathscr{A}'_{j_*}$. (It is easy to show that the number of vertical hyperplanes in $\mathscr{A}'_{l_*}$ is no more than $g_{l_*,d}(r)$.)

The second property is that the total number of points in the cells examined for a query is no more than $d\alpha_* n$, when the dual point $\mathscr{D}(h)$ is above the j_*-level of R. (Otherwise, no recursive call is made, and the work is proportional to the answer size.) To show this, consider the vertical projection x of $\mathscr{D}(h)$ onto the j_*-level of R. The hyperplanes above $\mathscr{D}(h)$ are also above x, so $h^- \subset \mathscr{D}(x)^-$. The projection x is contained in a $(d-1)$-simplex T on the j_*-level. (We are using the generalized sense of "simplex" mentioned above.) Suppose T is bounded. Any halfspace containing x contains at least one vertex of T, so $\mathscr{D}(x)^-$ is contained in U, the union of the halfspaces $\mathscr{D}(v)^-$ for v a vertex of T. This implies $h^- \subset U$. Since the duals of the vertices of T are j_*-facets of R, U is a union of cells of A'_{j_*}. Hence the total number of points in the cells examined for the query is no more than $d\alpha_* n$.

Suppose the simplex containing x is unbounded. Then x is in a (generalized) simplex on the j_*-level of R, and so is the convex closure of no more than $d-1$ vertices and unbounded edges. The edge endpoints at infinity correspond dually to the vertical hyperplanes added to A'_{l_*}, and the points are dual to j_*-facet hyperplanes. Let U be the union of the (negative) halfspaces bounded by these d hyperplanes. Then $h^- \subset \mathscr{D}(x)^- \subset U$, and again the total number of points in the cells examined for the query is no more than $d\alpha_* n$.

With these two facts, by [28] and [4] the resulting query time is $O(A+n^{\beta})$, where $\beta = 1 - 1/(1+B)$, and

$$B = \frac{O\left(\log\binom{r^{\lfloor d/2\rfloor} j_*^{\lceil d/2\rceil}}{d-1}\right)}{-\log(d+1)\alpha_*},$$

so $\beta = 1-\gamma+\varepsilon$, where $\gamma = 1/(1+(d-1)\lfloor d/2\rfloor)$, and $\varepsilon > 0$ is independent of n and decreasing in r. The preprocessing time is expected $O(n \log n)$ and the storage is $O(n)$, as is easy to verify.

5. Concluding Remarks

One natural question regarding these results: can deterministic algorithms do the same things? For example, Theorem 3.7 guarantees the existence of subsets that are good for divide-and-conquer; can a deterministic algorithm find such subsets? The work of [12] says yes. However, these algorithms are expensive, requiring $\Omega(n^b)$ time. The recent algorithms of Matoušek and Agarwal are faster, but seem to be more specific: they apply to arrangements of lines in the plane [32], [1]. The results in these papers are stronger than those here, and they show how to find ranges all with $O(n/r)$ conflicts, rather than $O(n/r)$ on average. How general, and fast, can these results be made?

Another natural question concerns problems for which $\tau_0(n) \gg A$. The problems of convex-hull computation for $d > 3$, visibility graph construction, and hidden surface elimination are all in this category. Here our results do not readily imply output-sensitive algorithms. Is there some way to make effective use of randomization for these problems?

As one more application of random sampling, the ideas of this paper can readily be used to obtain an algorithm for point location in planar subdivisions that requires $O(n \log n)$ expected preprocessing, $O(n)$ space, and $O(\log n)$ query time.

The results of Section 3.2 can be extended to some "degenerate" cases where δ is not functional, so that the work done in the convex-hull algorithm is $O(n \log A')$, where A' is the number of extreme points in the output polytope. This extension will be reported elsewhere.

Acknowledgments

We thank John Hershberger for finding an error in the ancestor of Theorem 3.7. Thanks to Micha Sharir for helpful comments, and for observing the need for nondegeneracy in Theorem 3.7. Ketan Mulmuley observed that Lemma 4.3 is needed, and made other helpful comments. Conversations with Joe O'Rourke led to the incremental versions of our algorithms. It is a pleasure to thank two trenchant referees, and also John Hobby and Ellen Feinberg (née Silverberg), for helpful comments.

References

1. P. K. Agarwal. An efficient deterministic algorithm for partitioning arrangements of lines and its applications. In *Proceedings of the Fifth Symposium on Computational Geometry*, pages 11-22, 1989.
2. A. Aggarwal. Personal communication.
3. N. Alon and E. Györi. The number of small semispaces of a finite set of points in the plane. *J. Combin. Theory Ser. A*, 41:154-157, 1986.
4. N. Alon, D. Haussler, and E. Welzl. Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension. In *Proceedings of the Third Symposium on Computational Geometry*, pages 331-340, 1987.
5. B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 590-600, 1988.
6. B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76-90. 1985.
7. B. Chazelle and F. P. Preparata. Halfspace range search: an algorithmic application of k-sets. *Discrete Comput. Geom.*, 1:83-93, 1986.
8. B. Chazelle and E. Welzl. Range searching and VC-dimension: a characterization of efficiency. Technical Report B-88-09, Freie Universität Berlin, Institut für Mathematik III, Arnimallee 2-6, 1000 Berlin 33, 1989.

9. L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Unpublished manuscript, 1986.
10. K. L. Clarkson. A probabilistic algorithm for the post office problem. In *Proceedings of the 17th Annual SIGACT Symposium*, pages 175-184, 1985.
11. K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195-222, 1987.
12. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in computational geometry. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 539-549, 1988.
13. K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proceedings of the 29th Annual IEEE Symposium 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).
14. K. L. Clarkson. Applications of random sampling in computational geometry, II. In *Proceedings of the Fourth Symposium on Computational Geometry*, pages 1-11, 1988.
15. K. L. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and surfaces. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 568-579, 1988.
16. K. L. Clarkson and P. W. Shor. Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental. In *Proceedings of the Fourth Symposium on Computational Geometry*, pages 12-17, 1988.
17. K. L. Clarkson, R. E. Tarjan, and C. J. Van Wyk. A fast Las Vegas algorithm for triangulating a simple polygon. *Discrete Comput. Geom.*, this issue, pages 423-432.
18. R. Cole, M. Sharir, and C. Yap. On k-hulls and related problems. *SIAM J. Comput.*, 16:61-77, 1987.
19. H. Edelsbrunner. Personal communication.
20. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, 1987.
21. H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, 29:551-559, 1983.
22. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. In *Proceedings of the Fourth Symposium on Computational Geometry*, pages 118-113, 1988.
23. H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341-363, 1986.
24. P. Erdös and J. Spencer. *Probabilistic Methods in Combinatorics*. Academic Press, New York, 1974.
25. J. E. Goodman and R. E. Pollack. On the number of k-subsets of a set of n points in the plane. *J. Combin. Theory Ser. A*, 36:101-104, 1984.
26. D. H. Greene and D. E. Knuth. *Mathematics for the Analysis of Algorithms*. Birkhäuser, Boston, 1981.
27. L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graphics*, 4:75-123, 1985.
28. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127-151, 1987.
29. A. Heppes. Beweis einer Vermutung von A. Vázsonyi. *Acta Math. Acad. Sci. Hungar.*, 7:463-466, 1956.
30. C. A. R. Hoare. Quicksort. *Comput. J.*, 5:10-15, 1962.
31. D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15:287-299, 1986.
32. J. Matoušek. Construction of ε-nets. In *Proceedings of the Fifth Symposium on Computational Geometry*, pages 1-10, 1989.
33. N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31:114-127, 1984.
34. D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7:217-236, 1978.

35. K. Mulmuley. A fast planar partition algorithm: part I. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 580-589, 1988.
36. F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Comm. ACM*, 20:87-93, 1977.
37. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.
38. J. Reif and S. Sen. Optimal parallel algorithms for computational geometry. In *Proceedings of the 16th International Conference on Parallel Processing*, 1987.
39. J. Reif and S. Sen. Polling: a new randomized sampling technique for computational geometry. In *Proceedings of the 21st Annual SIGACT Symposium*, pages 394-404, 1989.
40. R. Seidel. A convex hull algorithm optimal for point sets in even dimensions. Technical Report 81/14, Dept. Computer Science, University of British Columbia, 1981.
41. R. Seidel. Constructing higher dimensional convex hulls at logarithmic cost per face. In *Proceedings of the 18th Annual SIGACT Symposium*, pages 404-413, 1986.
42. E. Welzl. More on k-sets of finite sets in the plane. *Discrete Comput. Geom.*, 1:95-100, 1986.
43. E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proceedings of the Fourth Symposium on Computational Geometry*, pages 23-33, 1988.
44. C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. In *Proceedings of the Fourth Symposium on Computational Geometry*, pages 134-142, 1988.

Received August 12, 1988, and in revised form March 2, 1989.