Algorithms for Diametral Pairs and Convex Hulls that are Optimal, Randomized, and Incremental

Kenneth L. Clarkson and Peter W. Shor
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Abstract

We give a simple algorithmic technique for building geometric structures. The technique is randomized and incremental. As an application, we give an algorithm of this kind for computing the intersection of a set of halfspaces in three dimensions. (This intersection problem is linear-time equivalent to the computation of the convex hull of a point set.) The algorithm requires $O(n \log n)$ expected time, where the expectation is over the random behavior of the algorithm. A similar algorithm can be used to determine 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 problem of determining the diameter to the problem of computing spherical intersections, 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 $O(n\sqrt{n} \log n)$[Agg].

1 Introduction

1.1 The problems and results

One simple and obvious way of building a geometric structure, such as the intersection of a set of halfspaces, is to determine the structure incrementally, adding the halfspaces one by one and maintaining the resulting intersection.

Such an incremental approach gives an optimal algorithm for constructing an arrangement of hyperplanes[EOS86]. 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[Che86] 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 online, as some simple information is maintained for the objects that are not yet added. Our results assume some nondegeneracy conditions for convenience; for example, we assume that input point sets in E^3 have no four points coplanar.

Our general theorem should have many applications, as discussed briefly in §5. We will consider here two applications, for geometric problems set in three dimensions.

As one of these applications, we show that our approach gives an optimal Las Vegas algorithm for computing the convex hull of a set of points in E^3. The algorithm can be viewed as a variant of the "beneath-beyond" method. [Ede87] The divide-and-conquer algorithm of Preparata and Hong [PH77] yields the convex hull of a set of n points in E^3 in optimal $O(n \log n)$ time, in the worst case. While our algorithm does not give an improvement in asymptotic complexity, it is simpler than the divide-and-conquer algorithm.

Our convex hull algorithm will be presented as an algorithm for an equivalent problem, that of determining the intersection of a set of halfspaces. As another application of our randomized incremental technique, we will also show that the intersection of

a set of unit balls in E^3 can be computed in a similar way. Such an intersection of unit balls, or of balls of some given common radius, will be called a *spherical intersection*. In general, given $S \subset E^d$, the r-spherical intersection of S, or $I_r(S)$, is the intersection of the closed balls of radius r 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 [Hep56]. Like polytopes, spherical intersections have duals, which were introduced as *α-hulls* in [EKS83].

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 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 in S. A *diametral pair* of S is a pair of points in S 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 [PS85]. 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"[PS85] in computational geometry. In this paper, we give a randomized algorithm requiring $O(n \log n)$ expected time. The best algorithms previously known have worst-case time bounds no better than $O(n\sqrt{n} \log n)$ [Agg].

The main idea for our diameter algorithm is a randomized reduction from the diameter problem to the spherical intersection problem.

1.2 Contents of the paper

In §2, the diameter problem is reduced to a series of spherical intersection problems, using a simple randomizing procedure. In §3, incremental procedures are given for convex hulls and for spherical intersections. Modified versions of these algorithms, requiring $O(n)$ space in the worst case, are also given. A time bound for a quite general version of these procedures is proven in §4. Some concluding remarks are made in §5.

2 The diameter problem

In this section, 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 $r = D_p$ for some $p \in S$. Any point $q \in S$ that is in the interior of $I_r(S)$ 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 $I_r(S)$, then $D_S \geq D_q > D_p$. Thus, if there are no points of S outside $I_r(S)$, then $D_S = D_p$, and if there are any such points, only those points can possibly be in diametral pairs.

Suppose the points of S could be listed in nondecreasing order of their D_p values. Then if a point $p \in S$ 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. Suppose $I_r(S)$ can be computed in $O(n \log n)$ time, and also $O(n \log n)$ time suffices to determine the points of S outside $I_r(S)$. Then t_n expected time suffices 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 $t_n = O(n \log n)$.

In §3, it is shown that $I_r(S)$ can be computed in $O(n \log n)$ expected time. The set $S \setminus I_r(S)$ can be computed in $O(n \log n)$ time, using an algorithm for point location in a planar subdivision[PS85]. To do this, define a "stereographic projection" for $I_r(S)$ as follows: pick a point p on the boundary of $I_r(S)$, and a sphere Z determining the face of $I_r(S)$ containing p. Let p' be the point antipodal to p on Z, and H the tangent 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. The set of points that are the image under F of the edges of $I_r(S)$ naturally induce 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 $F(s)$. The region of the subdivision that contains $s' = F(s)$ corresponds to a face of $I_r(S)$, and s is in $I_r(S)$ if and only if the line segment $\overline{ps}$ does not pass through the boundary of that face.

3 Convex hulls and spherical intersections

3.1 The algorithms

In this section, we give an algorithm 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 halfspaces in E^3. The necessary modifications to compute spherical intersections are then shown.

The halfspace intersection algorithm is randomized and incremental. It adds the halfspaces one by one in random order to a set R, maintaining $\mathcal{P}(R)$ as each halfspace is added. The incidence graph of $\mathcal{P}(R)$ is maintained (see e.g. [Ede87, §8.2]), giving the incidence relations between the vertices, edges, and faces of $\mathcal{P}(R)$.

To make the algorithm faster, some additional information is used: a *conflict graph* is maintained, 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 as linked lists of conflicts, with L_H the edges of $\mathcal{P}(R)$ not contained in H, and with a similar list L_e for each edge e of $\mathcal{P}(R)$.

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 exactly those *not* in L_H. Some edges in L_H have both vertices not in H. Such edges can be discarded. The remaining edges in L_H are cut by the bounding plane of H. A face F 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 face gives rise to a new face F' of $\mathcal{P}(R')$, bounded by edges not in L_H, and by new edges $e_1' = e_1 \cap H$, $e_2' = e_2 \cap H$, and e_{12}, which is the intersection of F 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. Thus the polytope $\mathcal{P}(R')$, and its incidence graph, is 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 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. The motivation for the algorithm is that when adding the rth halfspace, the expected number of such halfspaces is $O(n/r)$, so the expected work overall is $O(n \sum_{1 \le r \le n} 1/r) = O(n \log n)$.

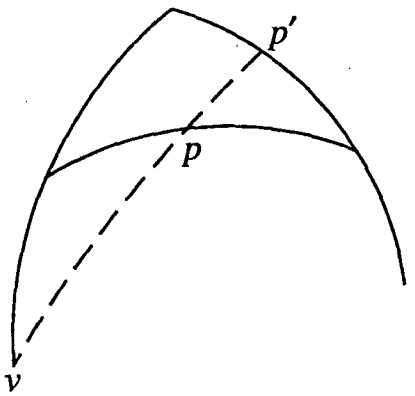

Figure 1: A vertex v and a point p on new edge e.

The algorithm for spherical intersection is very much the same, except that instead of adding halfspaces one by one, we add closed balls of a given radius. The geometric fact necessary to the correctness of this algorithm is given in the following lemma.

Lemma 3.1 *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.*

Proof. Suppose the edges involved are on an old face F and a new face F', with B the newly added ball, so that $F' = F \cap B$. Two of the new edges are subsets of deleted edges, so it is only necessary to show the containment for the new edge e that is contained in the intersection of F and the sphere bounding B.

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, then z is no farther than 1 from all points on the great circle arc connecting x and y. This fact is easily proven.

Now suppose v is a vertex of a deleted edge, where v is also a vertex of a new edge, so v is within B. (See Figure 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 is on the conflict list for the edge containing v or the edge containing p'. □

3.2 Variants using linear space

The following section shows that the algorithms we have given require $O(n \log n)$ expected time. Before this analysis, we consider the space used by the algorithms, and give variants of our algorithms. The variants require $O(n)$ space in the worst case, with the same time bound as the original algorithms. For convenience, the discussion will focus on halfspace intersections, although the same ideas apply to spherical intersections.

How much space can our halfspace intersection algorithm require? Certainly no more space than time, so $O(n \log n)$ expected space follows from our time bound. Can we do better than this? It would seem that we can: as part of the time analysis below, it is shown that at any given step, the expected size of the conflict graph is $O(n)$. Thus it would seem that no more than $O(n)$ space is expected overall. However, this is only an expectation, and furthermore, we are considering the space used not just at any given step, but throughout the algorithm. Indeed, it can be shown that our algorithm requires $\Omega(n \log \log n)$ expected space. It is an interesting open question whether this bound is tight. However, a simple variant of our algorithm has a worst-case space bound of $O(n)$, so the question is not as pressing as it might be.

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 the $\mathcal{P}(R)$ 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 Randomized incremental construction

In this section, we give a generalized framework for our convex hull and spherical intersection algorithms, and prove an expected-time bound for algorithms within that framework. The $O(n \log n)$ expected time for our algorithms will then follow readily.

The algorithmic technique we describe will be called *randomized incremental construction*. The formal framework will follow that in [Cla88], summarized as follows: let S be a set of n subsets of E^d, called *objects*, and let $\mathcal{F}$ be another collection of subsets of E^d, called *regions*. The objects "define" the regions in some way, which we formalize as a relation δ between $\mathcal{F}$ and $\cup_{i \le b} S^i$, where b is an integer parameter of an application. An i-tuple $X \in S^i$ defines $F \in \mathcal{F}$ if and only if $F \delta X$, that is, (F, X) is in the relation δ. The set of all regions defined by the objects in S is denoted $\mathcal{F}_S$. The set $\mathcal{F}_S^j$ is that of all $F \in \mathcal{F}_S$ such that F has nonempty intersection with j objects of S. The construction problem associated with an instance of this framework is to determine, given an input set S, the set $\mathcal{F}_S^0$ of empty regions.

In the halfspace intersection algorithm, the objects are the complements of the halfspaces, and the regions are the edges of the intersections, with $b = 4$.

A randomized incremental construction algorithm 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 regions $\mathcal{F}_R^0$ are maintained, and updated as each object is added. To make this algorithm faster, a *conflict graph* is maintained between the objects in $S \setminus R$ 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 a point s to R, and to update $\mathcal{F}_R^0$ and the conflict graph, is linearly proportional to the number of objects that conflict with regions that conflict with s. Plainly, the update time is at least as long as this. In

many instances, this linear time suffices.

To prove the theorem, we will need to make a nondegeneracy assumption, that the relation δ is a function up to permutation: that is, for any $F \in \mathcal{F}_S$, there is an i-element subset $T_F \subset S$ such that $F \in \mathcal{F}_R$ for $R \subset S$ if and only if $T_F \subset R$. For halfspace intersections, this is implied by the assumption that the bounding planes of the halfspaces are nondegenerate, that is, no four intersect at a common point.

Theorem 4.1 Randomized incremental construction. *In an instance of the general incremental construction algorithm, suppose the update and nondegeneracy conditions hold. Suppose that there is a nondecreasing function $m(r)$ so that for R a random subset of S of size r, $m(r) \geq E[|\mathcal{F}_R^0|]$. Then the expected time required by the instance is within a constant factor of $m(n) + n\sum_{1\leq r\leq n/2} m(r)/r^2$.*

Proof. For $r \leq n/2$, we will show that the expected time required to add object $s \in S$ to a subset R of size r is proportional to $(n/r)^2/(n-r)m(r)$. Thus to add objects up to $r \approx n/2$, the expected time required is proportional to $n\sum_{1\leq r\leq n/2} m(r)/r^2$. For $r > n/2$, we will show that the time required to add an object is proportional to $m(r)/n$, so that $m(n)$ bounds the expected time to add all the objects for $n/2 < r \leq n$. This gives the desired bound.

The proof is very similar to the proof of Theorem 3.2 of [Cla88], to which the reader is referred for background and notation.

By the update condition, the time required is proportional to

$$\sum_{\substack{j\geq 0\\ F\in\mathcal{F}_S^j}} jI_F,$$

where $I_F = 1$ when $F \in \mathcal{F}_R^0$ meets s, and $I_F = 0$ otherwise.

The expected value of this quantity is

$$\sum_{\substack{j\geq 0\\ F\in\mathcal{F}_S^j}} j\text{Prob}\{F \in \mathcal{F}_R^0, s \in F\},$$

or

$$\sum_{\substack{j\geq 0\\ F\in\mathcal{F}_S^j}} \frac{j^2}{n-r}\text{Prob}\{F \in \mathcal{F}_R^0\}$$

The result now follows for $r \leq n/2$, since from Theorem 3.2 of [Cla88], this quantity is

$$E[T_w(R)]/(n-r) \leq 2E[T_w(R)]/n,$$

where $w(j) = j^2$.

Now the case $r > n/2$. As in the proof of Theorem 3.2 of [Cla88], we can assume that the relation δ is defined only for $X \in S^b$, and not for $X \in S^i$ with $i < b$. Application of the proof b times then gives the general result.

We know that because δ is a function up to permutation, the last sum above is no more than

$$\sum_{j\geq 0} \frac{j^2}{n-r} p_j(n,r,b)|\mathcal{F}_S^j|,$$

where $p(j;n,r,b) = \binom{n-j-b}{r-b}/\binom{n}{r}$. This is

$$\sum_{j\geq 0} \frac{j^2}{n-r} \rho_j p_j(n,r_j,b)|\mathcal{F}_S^j|,$$

where $r_j = \lfloor (r-b)/(j+1)\rfloor + b$ and $\rho_j = p_j(n,r,b)/p_j(n,r_j,b)$. We have

$$m(r) \geq m(r_j) \geq E[|\mathcal{F}_{R_j}^0|] \geq p_j(n,r_j,b)|\mathcal{F}_S^j|,$$

where $R_j \subset S$ is random and $|R_j| = r_j$, and it is easy to show that

$$\rho_j \leq O(1)(1+j)^{b+1/2}\left(\frac{n-r}{n-r_j}\right)^j,$$

as $r \to \infty$. Plugging in these two upper bounds, the expected work to add s is no more than

$$\sum_{j\geq 0} \frac{j^2}{n-r} O(m(r))(1+j)^{b+1/2}\left(\frac{n-r}{n-r_j}\right)^j,$$

which is no more than

$$\frac{O(m(r))}{(n-r_1)}\sum_{j\geq 0} j^2(1+j)^{b+1/2}\left(\frac{n-r}{n-r_j}\right)^{j-1}.$$

The bound $O(m(n)/n)$ follows by noting that the sum converges to a constant. This implies that the time to add all the objects, for $n/2 < r \leq n$, is $O(m(n))$, and the theorem follows. □

5 Concluding remarks

The randomized incremental construction technique may well see many other applications. In [Cla88], it is used to obtain an optimal algorithm for line segment intersection. It readily extends to obtain optimal algorithms for convex hulls in higher dimensions, and for Voronoi diagrams of line segments in the plane.

References

[Agg] A. Aggarwal. personal communication.

[Che86] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. unpublished manuscript, 1986.

[Cla88] K. L. Clarkson. Random sampling in computational geometry, II. *This proceedings*, 1988.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, New York, 1987.

[EKS83] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29:551–559, 1983.

[EOS86] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15:341–363, 1986.

[Hep56] A. Heppes. Beweis einer Vermutung von A. Vázsonyi. *Acta Math. Acad. Sci. Hungar.*, 7:463–466, 1956.

[PH77] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20:87–93, 1977.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.