Lower Bounds for Intersection Searching and Fractional Cascading in Higher Dimension*

Bernard Chazelle[†] Ding Liu[‡]

Abstract

Given an n-edge convex subdivision of the plane, is it possible to report its k intersections with a query line segment in $O(k+\text{ polylog}(n))$ time, using subquadratic storage? If the query is a plane and the input is a polytope with n vertices, can one achieve $O(k+\text{ polylog}(n))$ time with subcubic storage? Does any convex polytope have a boundary dominant Dobkin-Kirkpatrick hierarchy? Can fractional cascading be generalized to planar maps instead of linear lists? We prove that the answer to all of these questions is no, and we derive near-optimal solutions to these classical problems.

1 Introduction

Fractional cascading is a general technique for speeding up lookup queries in catalogs associated with the nodes of a graph [14]. Specifically, suppose that G is a bounded-degree graph, where each node v is associated with a catalog C_v (which is just a sorted list of numbers). The successor of x in C_v is defined as $\min\{y \in C_v \cup \{\infty\} \,|\, y \geq x\}$. Given a subset of k nodes in G whose induced subgraph is connected, it is easy, by repeated binary search, to compute the successors of any query x in the k catalogs in time $O(k \log n)$, where n is the combined size of all the catalogs. Fractional cascading reduces the query time to $O(k + \log n)$ while increasing the storage by only a constant factor. The technique has found numerous applications in

*A preliminary version of this paper appeared in Proc. 33rd Annual ACM Symp. Theory of Comput. (2001), 322–329.

[†]NEC Research Institute and Department of Computer Science, Princeton University
[‡]Department of Computer Science, Princeton University

computational geometry (from which it originated), but also in constraint databases [6], IP routing [7], packet classification [21], data mining [26], geographical information systems [3], etc. This versatility has motivated the design of all sorts of variants: dynamic, probabilistic, parallel, external-memory fractional cascading [4, 3, 16, 18, 24, 28, 27].

An outstanding open problem has been the two-dimensional generalization of fractional cascading: what if we replaced the catalogs at the nodes by planar subdivisions? The question is motivated by more than mere curiosity. Indeed, the wide applicability of the technique stems from the fact that many data structures for multidimensional searching [5, 23] consist of a skeleton graph whose nodes are themselves repositories of auxiliary (often recursively defined) data structures. As long as linear lists reside at the bottom of the hierarchy, fractional cascading can be called into action. In some cases, however, the bottom layer consists of planar maps (ie, 2D catalogs) and the current technology fails.

2D fractional cascading would immediately lead to improved algorithms for nearest neighbor searching in E^3, intersection search for query segments in planar line arrangements and convex subdivisions, intersection search for query planes and polygons in polytopes, fixed-directional ray shooting in 3D, not to mention numerous instances of range searching. This paper dashes all such hopes. We show that not only generalizing FC to 2D catalogs is impossible but that no pointer machine solutions can provide the sort of logarithmic speed-up associated with fractional cascading.

Furthermore, the counterexamples are hardly pathological: in fact, catalogs consisting of simple parallel strips are enough to break the whole fractional cascading scheme apart. This can seem rather surprising in view of previous results suggesting that such generalizations might, indeed, be possible. For example, navigation among geodesic triangles as described in [13, 19] can be naturally viewed as an instance of 2D fractional cascading. So, it appears that the catalog graphs of geodesically triangulated polygons are more exceptional than previously thought.

We also resolve a question that had been open since the mid-eighties. The Dobkin-Kirkpatrick hierarchy [17] is a simplicial complex subdivision of an n-vertex convex polytope with the property that no line can intersect more than $O(\log n)$ simplices. It is an essential tool for all sorts of polyhedral operations, mesh simplification, etc. If the DK hierarchy is *boundary dominant*, meaning that, up to a logarithmic additive term, the number of simplices intersected by any given plane is at most proportional to the boundary size of the intersection, then it can also be used for intersection searching or silhouette computation. For example, we could use such a hierarchy to compute, in optimal time, the shadow of a polytope cast by a single light source. Constructing a DK hierarchy is highly nondeterministic: Among the (usually) exponentially many possible realizations, could it be that at least one of them is boundary dominant? We prove that the answer is no. Of anecdotal interest, we should mention that this purely combinatorial question is resolved by using algorithmically inspired arguments.

Here is a summary of our results. All complexity results hold in the standard pointer machine model [29].

1. There is a graph with 2D catalogs attached to its nodes such that to perform the same point location query at the k nodes of a connected subgraph in time $O(k+ \text{polylog}(n))$ requires storage $\tilde{\Omega}(n^2)$, where n is the combined size of all the catalogs.[1]

2. For any n large enough, there is a convex planar subdivision with n vertices such that to compute all k edges intersected by a query line in $O(k+ \text{polylog}(n))$ time requires storage $\tilde{\Omega}(n^2)$. On the other hand, $O(n^2)$ storage is sufficient to achieve $O(k + \log n)$ query time (even if the query is a line segment). The same lower bound holds for intersecting a query line with a simple polygon with n vertices.

3. For any n large enough, there is a (nonconvex) polytope P in $\mathbf{R}^3$ with n vertices such that, given a query plane π, to compute its intersection $P \cap \pi$ in time $O(k+ \text{polylog}(n))$, where k is the number of vertices in $P \cap \pi$, requires storage $\Omega(n^{3-\varepsilon})$, where ε is an arbitrarily small positive constant. On the other hand, $O(n^3)$ storage is sufficient to achieve $O(k + \log n)$ query time.

4. For any n large enough, there exists a convex polytope with n vertices that admits of no boundary dominant DK hierarchy.

Remarks: In all cases, the lower bounds remain true, up to a factor of n^ε, if we replace the polylog term in the query time by a function of the form n^δ, for small enough $\delta, \varepsilon > 0$. Our third result also holds for convex polytopes but the lower bound then drops to $\tilde{\Omega}(n^2)$. (We have been unable to find a matching upper bound for that restricted version of the problem.)

How do our results compare to previous work? The upper bounds are mostly applications of known techniques; in particular, duality, filtering

[1] The tilde notation hides a polylogarithmic factor.

search, navigation in arrangements. The lower bounds, on the other hand, should come as more of a surprise. Although not particularly difficult technically, they are rather counter-intuitive. Pointer machine lower bounds exist for all sorts of problems, including union-find [25, 29] and range searching [10, 15]; see surveys [1, 22]. However, we are not aware of previous lower bounds for intersection searching or 2D fractional cascading. Our proofs rely on a general volume argument developed in [15] and a Heilbronn-type result proven in [9]. The rest is new and self-contained.

2 Preliminaries

Intersection searching refers to the following type of problem: Given a set S of geometric "objects" (eg, edges of a planar subdivision, facets of a polytope), we wish to preprocess S into a data structure so that, given any query q from a predefined class (line segment, hyperplane), the set of all objects in S intersected by q can be reported efficiently. It is understood that the set S is fixed once for all, while queries are presented and answered on-line. Note that this definition need not make any reference to geometry; in fact it is often useful to view intersection searching abstractly by simply assuming a relation between each q and a subset of S (the "intersected" objects). In the pointer machine framework [29], a data structure for intersection searching is modeled as a directed graph G with bounded outdegree. The graph G is referred to as a *search structure for* S. Some of the nodes are assigned objects of S (not necessarily injectively). Given a query q, the algorithm navigates through the graph G, beginning at some start node, until the subgraph traversed contains at least one node assigned to each object of S intersected by q. For lower bound purposes, it suffices to count the number of nodes visited as a conservative estimate of the query time. (This also covers randomized query-answering algorithms as well.)

As in [10, 15], we first give a general lemma on the size of any pointer machine data structure for efficient intersection searching; then we establish the desired storage lower bound by exhibiting a set of "hard" queries.

Definition 2.1 *A search structure G for set S is (α, ω)-*effective*, with α a positive constant and ω an additive overhead, if for any query q, we have $|G(q)| \leq \alpha(k+\omega)$. Here $G(q)$ is the set of nodes visited in G while answering query q, and k is the output size, ie, the number of objects in S intersected by q.*

This definition expresses our focus on search structures that answer queries in time linear in the output size, aside from an additive overhead of $\omega = \omega(|S|)$. Of course, we can also handle arbitrary multiplicative overheads by setting $\alpha = \alpha(|S|)$.

Definition 2.2 *A collection of queries $Q = \{q_i\}$ is (m, ω)-favorable for S, with $m > 1$, if Q satisfies the following relevance and independence conditions.*

1. Relevance*: $|S \cap q_i| \geq \omega$, for any query $q_i \in Q$.*

2. Independence*: $|S \cap q_{i_1} \cap \cdots \cap q_{i_m}| = O(1)$, for all possible $i_1 < \cdots < i_m$.*

Here $S \cap q_i$ denotes the set of objects in S intersected by q_i, and $S \cap q_{i_1} \cap \cdots \cap q_{i_m}$ has a similar meaning.

The *relevance* condition means that each query intersects enough objects so make the output size dominate the additive overhead. The *independence* condition gets to the heart of the matter: It is a Zarankiewicz-type condition stating that the bipartite graph induced by objects and queries should be free of large complete subgraphs. The motivation for these definitions comes from a general volume argument [10, 15] about pointer machines. Adapted to our purposes it states that:

Lemma 2.3 *Given a search structure G for S and a set Q of queries, assume that G is (α, ω)-effective for some constant α, and Q is (m, ω)-favorable for S. For any ω large enough, the size of G is $\Omega(|Q|\ \omega/m)$.*

Lemma 2.3 is the main vehicle for proving lower bounds on the storage required by a search structure with a given query time. Complexity questions are in this way reduced to purely combinatorial ones, involving the existence of "favorable" sets of queries.

The rest of this paper is organized as follows. In Section 3 we establish lower bounds on the complexity of intersecting a planar subdivision (resp. convex polytope) with a query line segment (resp. query plane). These results give us the stepping stone from which we can conclude that 2D fractional cascading is impossible (Section 4), and that not all convex polytopes admit of a boundary dominant Dobkin-Kirkpatrick hierarchy (Section 5). We establish the $\Omega(n^{3-\varepsilon})$ lower bound for (nonconvex) polytopes in Section 6, and also provide a nearly matching upper bound.

3 Planar Subdivisions and Convex Polytopes

Theorem 3.1 *Given a planar subdivision with n vertices, to compute all k edges intersected by a query line in $O(k+\omega)$ time requires $\Omega(n^2/\omega^2)$ storage.*

Theorem 3.2 *Given an n-vertex convex polytope in $\mathbf{R}^3$, to compute all k edges intersected by a query plane in $O(k+\omega)$ time requires $\Omega(n^2/\omega^2)$ storage.*

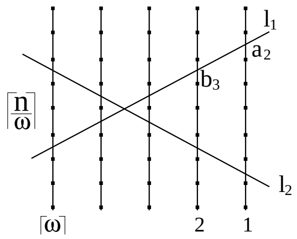

Figure 1: *Towards a hard convex subdivision.*

The lower bound for planar subdivisions is essentially optimal: an $O(k+\log n)$ query time solution using $O(n^2)$ storage is described in [8]. Note that the lower bound also holds for the problem of intersecting a query line with a simple polygon, since any n-vertex planar subdivision can be easily transformed into a simple $O(n)$-gon (by essentially duplicating each edge and creating a polygon of very small area). For convex polytopes the complexity gap is still large. The best solution with $O(k+\text{ polylog}(n))$ query time requires $O(n^3)$ storage and is obtained by applying the solution for the nonconvex case described in Section 6.

Both theorems use the same basic starting construction. Take $\lceil\omega\rceil$ congruent vertical segments evenly distributed horizontally, labeled $1,\ldots,\lceil\omega\rceil$, from right to left (Fig. 1). Next, decompose each segment into $t=\lceil n/\omega\rceil$ subsegments of the same length, bringing the total number of edges to $\Theta(n)$. Let a_i (resp. b_i) denote the midpoint of the i-th subsegment on segment 1 (resp. 2), counting top down. Of the t^2 lines passing through pairs (a_i,b_j), any one that intersects segment $\lceil\omega\rceil$ is called *hitting*. For example, line l_1 in Fig. 1 is such a hitting line. A few simple observations:

Fact 3.3 *Every hitting line intersects all vertical segments 1 to $\lceil\omega\rceil$, and every intersection point is the midpoint of some subsegment.*

To bound the number of hitting lines, observe that any point on segment 1 can join, via a hitting line, at least $\lceil n/\omega \rceil / (\lceil \omega \rceil - 1)$ points on segment 2; and hence,

Fact 3.4 *The number of hitting lines is at least* n^2/ω^3.

Let the *canonical subset* of a hitting line refer to the set of subsegments that intersect it. By Fact 3.3, intersections take place at midpoints, and so two lines that intersect the same subsegments pass through the same two points and therefore are identical.

Fact 3.5 *Each canonical subset is of size* $\lceil \omega \rceil$ *and the intersection of any two of them is of size at most* 1.

Now let S (resp. Q) be the set of subsegments (resp. hitting lines). Fact 3.5 implies that Q is $(2, \omega)$-favorable for S. We are now ready to build a full-fledged convex subdivision around S. Thicken the vertical segments, turn them into thin ladder-like concave strips, and add horizontal rungs to ensure the convexity of their decompositions (Fig. 2). Finally, cap the top and bottom parts of the ladders with two suitable convex pieces. We redefine S to be the left subsegments of the ladders, and check that, by keeping the ladders thin enough, Q still is $(2, \omega)$-favorable for S: we then are ready to apply Lemma 2.3 to derive a lower bound. The only problem is that the intersection search problem defined by Q and S is not, properly speaking, a "line-intersects-subdivision" problem since the output contains edges outside of S. However, by keeping the ladders thin enough, we see that each set of intersected edges in S is a subset at least half the size of the actual set of intersected edges. And so the lower bound for the (S, Q) problem also applies to the "line-intersects-subdivision." Theorem 3.1 follows from the fact that the number n of vertices in the resulting subdivision is $O(|S|)$. $\Box$

The reader might wonder why we did not choose to augment S by including in it all the edges of the subdivision, and then check again that the relevance and independence conditions still hold. The reason is that although augmenting S works in this case, it does not in the more complex configurations discussed below. So, we prefer to use a reduction argument, where the actual geometric problem is reduced to an abstract intersection searching problem specified by a map $Q \mapsto 2^S$.

It is easy to modify the construction used in Theorem 3.1 to derive Theorem 3.2. The modifications are sufficiently straightforward to dispense

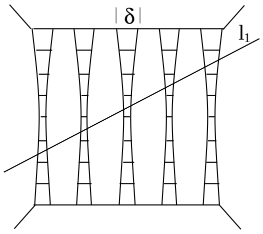

Figure 2: *Completing the convex subdivision.*

with a formal explanation. The idea is to raise the planar subdivision to form the "roof" of a shed-like polytope in $\mathbf{R}^3$. Each hitting line is replaced by a plane passing through that line and perpendicular to the ground, then the set of planes is favorable for the box. A brief sketch follows:

We begin with the polytope of Fig. 3, whose xy-projection reproduces the configuration of segments in Fig. 1. Next, we replace each rectangle on the roof by a convex polygon with $2(\lceil n/\omega \rceil + 1)$ edges, as shown in Fig. 4. (The bounding box is drawn only for illustration purposes.)

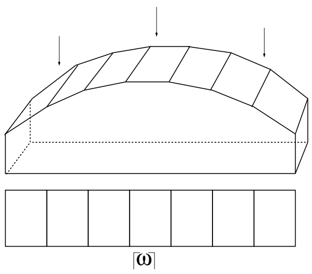

Figure 3: *A "house" and its projection.*

We patch the gaps between consecutive polygons on the roof by taking the convex hull (Fig. 5). Note that the polygonal chains $p_1, p_2, \ldots$ and $q_1, q_2, \ldots$ both lie in a plane but are not coplanar. Our construction is a lifted version of the hard convex subdivision of Fig. 2. The floor and vertical walls of the shed add only a constant number of edges to the intersection with any hitting plane, and so the same lower bound argument applies, leading

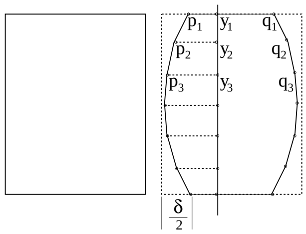

Figure 4: *Carving out the details.*

to Theorem 3.2.

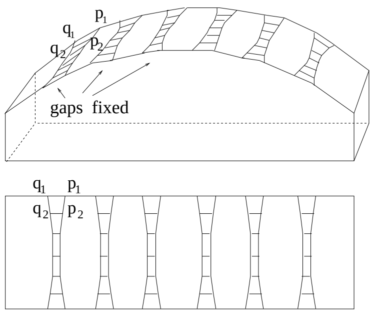

Figure 5: *Gaps fixed by taking the convex hull.*

4 Fractional Cascading in Higher Dimension

We use the results of the previous section to show that 2D fractional cascading is impossible. Recall that fractional cascading is a general technique for speeding up lookup queries in catalog graphs [14]. For our purposes, a catalog graph G is a connected bounded-degree graph, where each node v is associated with a catalog C_v (which is just a sorted list of numbers). The successor of x in C_v is defined as $\min\{y \in C_v \cup \{\infty\} \,|\, y \geq x\}$. A query is specified by a key x and a connected subgraph H of G. Its answer is the list of successors of x in the catalogs associated with the nodes of H.

Fractional cascading is a scheme that allows a query to be answered in time $O(k + \log n)$, using $O(n)$ storage, where k is the number of nodes in H and n is the combined size of all the catalogs. This improves upon the naive $O(k \log n)$ query time solution.

The construction of Fig. 1 tells the whole story: The graph G is a simple path whose nodes are associated with the $\lceil \omega \rceil$ vertical segments. The catalog at a node consists of the lines dual to the midpoints of the $\lceil n/\omega \rceil$ subsegments. We use the duality:

$$(a, b) \mapsto y = ax + b \quad \text{and} \quad y = ax + b \mapsto (-a, b)$$

because it respects above/below relationships. In this way, a catalog appears as a collection of parallel strips, ie, a planar convex subdivision (Fig. 6). A hitting query line dualizes to a query point, and the line/subsegment intersections identify which of the strips contain the dual point. Thus, 2D fractional cascading is seen to suffer from exactly the same lower bound as intersection searching for line/planar subdivision (Theorem 3.1). □

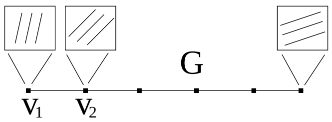

Figure 6: *A catalog graph in which 2D fractional cascading is impossible.*

Theorem 4.1 *There is a graph with planar subdivisions attached to its nodes such that to perform the same point location query at the k nodes of a connected subgraph in time $O(k+$ polylog$(n))$ requires storage $\tilde{\Omega}(n^2)$, where n is the combined size of all the catalogs.*

5 The DK Hierarchy

Let P be an n-vertex convex polytope in $\mathbf{R}^3$ whose vertex set is $V(P)$. A sequence of convex polytopes, $\mathcal{H}(P) = P_0, \ldots, P_k$, is called a (polyhedral) *hierarchy* if: (i) $P_0 = P$ and P_k is a simplex; (ii) $P_{i+1} \subset P_i$ and

$V(P_{i+1}) \subset V(P_i)$ for $0 \leq i < k$; (iii) $V(P_i) \setminus V(P_{i+1})$ forms an independent set with respect to the facial graph of ∂P_i. The *size* of $\mathcal{H}(P)$ is defined as $\sum_{i=0}^{k} |V(P_i)|$, its *height* is k, and its *degree* is:

$$\max_{0 \leq i \leq k} \#\text{edges of } P_i \text{ incident to a vertex of } V(P_i) \setminus V(P_{i+1}).$$

The hierarchy is called *Dobkin-Kirkpatrick* (or DK for short) if its size is $O(n)$, its degree is $O(1)$, and its height is $O(\log n)$. It is well known [17] that any polytope P admits of a DK hierarchy. In fact, a polytope will typically have an exponential number of distinct DK hierarchies. Of particular interest are the *boundary dominant* hierarchies: These have the additional property that, given any plane π,

$$\sum_{i=0}^{k} |P_i \cap \pi| = O(|P \cap \pi| + \log n),$$

where $|\,\text{polygon}\,|$ denotes the number of vertices in the polygon. Intuitively, it means that any planar cross-section should essentially be like the 2D equivalent of a DK hierarchy: a collection of concentric layers with a fraction of the complexity on the outermost layer.

It is folk knowledge that the existence of a *boundary dominant* DK hierarchy would carry with it all sorts of algorithmic benefits: For example, we could use such a hierarchy to compute, in optimal time, the shadow of a polytope cast by a single light source. We dash all such hopes by exhibiting polytopes with no boundary dominant DK hierarchies. Of anecdotal interest, we should mention that this purely combinatorial question is resolved by using algorithmically inspired arguments.

A remarkable feature of a DK hierarchy is that it lends itself naturally to navigation along piecewise linear curves and surfaces. Specifically, the pockets formed by $P_i \setminus P_{i+1}$ can be triangulated so as to turn the whole polytope P into a simplicial cell complex $\mathcal{C}$ (with the standard glueing properties one expects of a cell complex). Given any plane π, one can find, in $O(\log n)$ time, a starting point in $P \cap \pi$, and then explore every simplex of $\mathcal{C}$ that intersects π in a breadth first search traversal that takes constant time per simplex [11]. Suppose now that the DK hierarchy is boundary dominant. Because the pockets are each of constant size, the number of intersected simplices would be $O(|P \cap \pi|)$. So, we could compute the intersection between a query plane and a convex polytope in $O(k + \log n)$ time, using only $O(n)$ storage. This would stand in contradiction with Theorem 3.2. $\square$

Theorem 5.1 *For any n large enough, there exists a convex polytope with n vertices that admits of no boundary dominant DK hierarchy.*

6 Nonconvex Polytopes

We consider the problem of intersecting a (possibly nonconvex) n-vertex polytope in $\mathbf{R}^3$ with a query plane. We prove a lower bound first; then we give a nearly matching upper bound for the case where $\omega = O(\log n)$.

Theorem 6.1 *Given an n-vertex polytope in $\mathbf{R}^3$, to compute all k edges intersected by a query plane in $O(k+\omega)$ time requires $\Omega(n^{3-\varepsilon}/\omega^3)$ storage, where ε is an arbitrarily small positive constant.*

Note that this implies our earlier claim, the $\Omega(n^{3-\varepsilon})$ lower bound when $\omega(n)$ is polylogarithmic or even of the form n^δ, for any constant $\delta > 0$ with $\varepsilon = \varepsilon(\delta)$. From now on, ε denotes an arbitrarily small positive constant. Since the storage is at least linear in n, we may assume with no loss of generality that $\omega/n^{(2-\varepsilon)/3}$ is small enough. Our proof relies on the construction of a "hard" polytope P and a set Q of query planes that is $(\Theta(\log n), \omega)$-favorable for some *designated* edges of P that constitute the set S. For notational convenience, we use n as a parameter that differs from the number of vertices of P by at most a constant factor. As usual, we ensure that of all the edges intersected by any query plane of Q, at least a fixed fraction of them are designated. To achieve a query time of $O(|P \cap \pi| + \omega)$, we know from Lemma 2.3 that the size of the data structure should be $\Omega(|Q|\omega/\log n)$.

The input polytope P is carved out of the unit cube $\mathcal{C} = [0,1]^3$ in several "carving" steps. Let $\varepsilon_0 > 0$ be a small enough absolute constant (ie, with no dependence on ε). To begin with, we choose $\lceil \omega \rceil$ random points in the subsquare $|2y-1| \le \varepsilon_0$, $|2z-1| \le \varepsilon_0$ of the face $x = 0$ of $\mathcal{C}$, independently and uniformly, and join them to the face $x = 1$ by x-parallel segments. Next, decompose each such unit-length segment s into $\lceil 6n/\omega \rceil$ congruent subsegments. For technical reasons, we need to perform a random shift of this decomposition: For each s, pick a random real α uniformly between $-\frac{1}{2}\lceil 6n/\omega \rceil^{-1}$ and $\frac{1}{2}\lceil 6n/\omega \rceil^{-1}$ and move each of the $\lceil 6n/\omega \rceil - 1$ interior endpoints along s by α (Fig. 7). Note that each interior endpoint in a given s is shifted by the same amount, but that the $\lceil \omega \rceil$ random shifts are independent.

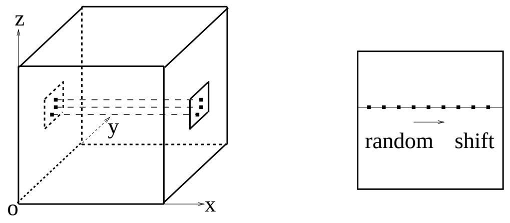

Figure 7: *Carving P out of a cube.*

The next step is to turn this configuration of edges into a bona fide polytope. First, make each horizontal segment into a cylinder with a tiny square base, say, of side length $1/n^2$. Next, attach these cylinders to the face $x = 0$ but *not* to the face $x = 1$; let it come very near the latter but not touch it. What happens to the (randomly shifted) decompositions? They naturally partition the boundary of each cylinder into rectangles. The polytope P consists of the unit cube with the protrusions through $x = 0$ formed by the cylinders (Fig. 8). Each shifted endpoint gives rise to four vertices of P: all the x-parallel edges that are incident upon any of them are called *designated* and form the set S. Note that many faces are coplanar in this construction but it is routine to perturb the polytope P to make it simplicial and in general position. Trivally,

Fact 6.2 *The number of vertices of P and the number of designated edges are both in $\Theta(n)$. Furthermore, no designated edge is of length larger than $\omega/2n$.*

Next, we define a large set Q of query planes that satisfy both the relevance and independence conditions (Definition 2.2). Choose t random points $q_1, \ldots, q_t$ uniformly in the square $1 - \varepsilon_0 \leq x, y \leq 1$ on the face $z = 1$, where $t = \lceil n^{2-\varepsilon}/\omega^3 \rceil$. Next, along each segment Oq_i, add the points $(3j\omega/n)q_i$, for all integers $n/6\omega < j < n/5\omega$. This defines a set $\{q\}$ of $\Theta(tn/\omega)$ points, which in turn specifies the set Q of query planes, each defined by the equation $\pi_q : \langle p, q \rangle = \|q\|_2^2$ (Fig. 9).

Let S_q be the slab consisting of all points at distance at most ω/n from the plane π_q. By using a Heilbronn-type argument [9], we can prove along the lines of [15]:

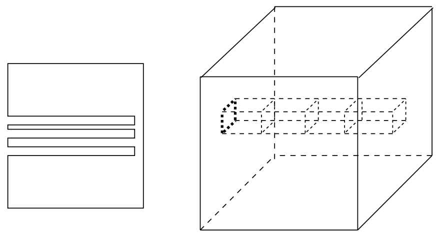

Figure 8: *Turning transversals into protrusions.*

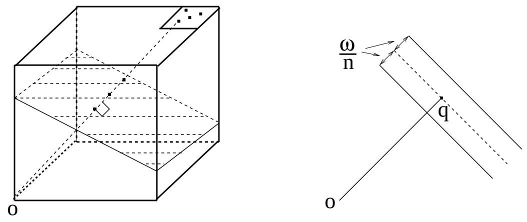

Figure 9: *The query set Q.*

Lemma 6.3 *With probability $1-o(1)$, any subset of at least $c\log n$ planes in Q, for fixed c large enough, contains three planes $\pi_{\hat{q}_1}, \pi_{\hat{q}_2}, \pi_{\hat{q}_3}$ such that the volume of the polyhedron $S_{\hat{q}_1} \cap S_{\hat{q}_2} \cap S_{\hat{q}_3}$ is $O(1/n^{1+\varepsilon})$.*

Proof: We may restrict ourselves to a subset $\mathcal{S}$ of size $c\log t$ planes, since $\log t = O(\log n)$. Recall that each plane is normal to some segment Oq_i where q_i is a point on the face $z=1$. Slabs normal to the same Oq_i do not intersect, and so we may assume without loss of generality that the planes of $\mathcal{S}$ are normal to distinct segments $Oq_1, Oq_2, \ldots, Oq_{|\mathcal{S}|}$. Because the q_i's are chosen randomly in an ε_0-by-ε_0 square at $z=1$, they have the following Heilbronn-type property (proven in [9]). For c large enough, with probability $1-o(1)$, the convex hull of $\{\, q_1, \ldots, q_{|\mathcal{S}|} \,\}$ is a polygon of area $\Omega(\varepsilon_0^2(\log t)/t)$. A triangulation of this convex hull produces at least one triangle of area $\Omega(\varepsilon_0^2/(ct))$, which we relabel $q_1q_2q_3$. We now show that the corresponding planes $\pi_{\hat{q}_1}, \pi_{\hat{q}_2}, \pi_{\hat{q}_3}$ satisfy the desired condition:

$$\operatorname{vol} \bigcap_{i=1}^{3} S_{\hat{q}_i} = O(1/n^{1+\varepsilon}). \tag{1}$$

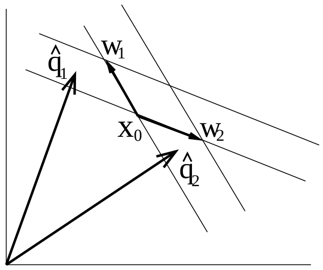

Figure 10: *Intersection parallelotope.*

The polyhedron $S_{\hat{q}_1} \cap S_{\hat{q}_2} \cap S_{\hat{q}_3}$ is spanned by w_1, w_2, w_3 where,

$$\langle w_j, \hat{q}_i \rangle = \begin{cases} (2\omega/n)\|\hat{q}_i\|_2 & \text{if } i = j, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $S_{\hat{q}_1} \cap S_{\hat{q}_2} \cap S_{\hat{q}_3}$ is the set of linear combinations $x_0 + \sum_i \alpha_i w_i$, for $0 \leq \alpha_i \leq 1$, where x_0 is the unique point satisfying

$$\langle x_0, \hat{q}_i \rangle = \Big(\|\hat{q}_i\|_2 - \frac{\omega}{n} \Big) \|\hat{q}_i\|_2$$

for $1 \leq i \leq 3$ (Fig. 10). Denote by $[w]$ the 3-by-3 matrix (w_1, w_2, w_3) and define the matrices $[q]$ and $[\hat{q}]$ similarly. Finally, let Λ be the diagonal matrix with $\Lambda_{ii} = \|\hat{q}_i\|_2$. It follows from $[w]^T [\hat{q}] = (2\omega/n)\Lambda$ that

$$\det [w] \det [\hat{q}] = \Big(\frac{2\omega}{n}\Big)^3 \prod_{i=1}^{3} \|\hat{q}_i\|_2$$

Because $\prod_i \|\hat{q}_i\|_2 (\det [q]) = \prod_i \|q_i\|_2 (\det [\hat{q}])$, and each $\|q_i\|_2$ is $\Theta(1)$,

$$\det [w] \det [q] = \Big(\frac{2\omega}{n}\Big)^3 \prod_{i=1}^{3} \|q_i\|_2 = \Theta\Big(\frac{\omega^3}{n^3}\Big)$$

The determinant of $[q]$ is, in absolute value, at least proportional to the area of triangle $q_1 q_2 q_3$, which is $\Omega(1/t)$. By our choice of $t = \lceil n^{2-\varepsilon}/\omega^3 \rceil$, $|\det[w]| = O(t\omega^3/n^3) = O(1/n^{1+\varepsilon})$. Note that vol $\bigcap_i S_{\hat{q}_i} = |\det[w]|$, which proves (1). $\Box$

Lemma 6.3 shows that, for any subset $K \subseteq Q$ of size at least $c \log n$, the volume of $\bigcap\{ S_q \mid \pi_q \in K \}$ is itself in $O(1/n^{1+\varepsilon})$. We now verify that with high enough probability the relevance and independence conditions hold.

Lemma 6.4 *Any plane in Q intersects at least $\lceil \omega \rceil$ designated edges.*

Proof: It suffices to show that, given any y, z with $|2y - 1| \leq \varepsilon_0$ and $|2z - 1| \leq \varepsilon_0$, the points $p_0 = (0, y, z)$ and $p_1 = (1, y, z)$ lie cleanly on opposite sides of any plane π_q of Q. (The adjective "cleanly" refers to the fact that this must remain true after turning segments into cylinders and perturbing P.) Recall that $q = (\beta u, \beta v, \beta)$, where

$$1 - \varepsilon_0 \leq u, v \leq 1 \qquad \text{and} \qquad \frac{1}{2} < \beta < \frac{3}{5}\,.$$

We easily verify that we have the two desired inequalities:

$$\langle p_0, q \rangle - \|q\|_2^2 = \beta v y + \beta z - \beta^2 (u^2 + v^2 + 1) = -\frac{\beta}{2} + O(\varepsilon_0) < 0.$$

$$\langle p_1, q \rangle - \|q\|_2^2 = \beta u + \beta v y + \beta z - \beta^2 (u^2 + v^2 + 1) = \frac{\beta}{5} - O(\varepsilon_0) > 0.$$

Lemma 6.5 *With probability $1 - o(1)$, given any set of at least $c \log n$ planes in Q, at most a constant number of designated edges intersect all of them.*

Proof: Let K be a set of at least $c \log n$ planes in Q and let K^* be the (interior of the) polyhedron $\mathcal{C} \cap \bigcap\{ S_q \mid \pi_q \in K \}$. Recall that to build P we choose random points on the face $x = 0$ of $\mathcal{C}$. Let $(0, y, z)$ be one of them and let $\ell = \ell(y, z)$ be the line passing through it parallel to the x-axis. The construction of P proceeds by choosing points on ℓ at regular intervals of length $\lambda = \lceil 6n/\omega \rceil^{-1}$ and shifting the $\lceil 6n/\omega \rceil - 1$ endpoints by a fixed, random amount between $-\lambda/2$ and $\lambda/2$. At the end of this process, let $N(K^*, \ell)$ denote the number of endpoints in $K^* \cap \ell$. A simple calculation shows that the expected value of $N(K^*, \ell)$ (over the random shift) is at most $|K^* \cap \ell|/\lambda$, where $|K^* \cap \ell|$ designates the length of the corresponding segment (we say "at most" and not "equal to" for the rare case where K^* intersects ℓ very near the boundary of $\mathcal{C}$). It follows that

$$\begin{aligned} \operatorname{vol} K^* &= \int_{[0,1]^2} |K^* \cap \ell(y,z)| \, dy \, dz \\ &\geq \lambda \int_{[0,1]^2} \mathbf{E}\, N(K^*, \ell(y,z)) \, dy \, dz \, . \end{aligned}$$

Restricting the integration domain to the ε_0-by-ε_0 squares to which each ℓ is actually adjacent and identifying $\mathbf{E}\, N(K^*, \ell(y,z))$ with the conditional expectation of $N(K^*, \ell)$ given $\ell = \ell(y,z)$, we find that

$$\begin{aligned} \operatorname{vol} K^* &\geq \lambda \int_{[(1-\varepsilon_0)/2,(1+\varepsilon_0)/2]^2} \mathbf{E}\, N(K^*, \ell(y,z)) \, dy \, dz \\ &= \Omega(\varepsilon_0^2 \lambda)\, \mathbf{E}_{(y,z)}\, \mathbf{E}\, [\, N(K^*, \ell) \,|\, \ell = \ell(y,z) \,] \\ &= \Omega(\varepsilon_0^2 \lambda)\, \mathbf{E}\, N(K^*, \ell) \, , \end{aligned}$$

where ℓ is a random x-parallel segment of the type used in the construction (ie, with endpoints in the tiny squares at $x = 0, 1$). It follows that

$$\operatorname{Prob}[\, N(K^*, \ell) > 0 \,] = O\Big(\frac{\operatorname{vol} K^*}{\varepsilon_0^2 \lambda} \Big) \, .$$

On the other hand, a simple geometric argument shows that $|S_q \cap \ell| = (2\omega/n)/\cos\theta$, where θ is the angle between Oq and the x axis.

$$\cos\theta = q_x / \|q\|_2 = \Theta(1),$$

and so

$$|S_q \cap \ell| = \Theta(\lambda)$$

and, therefore, any given ℓ can contribute only $O(1)$ such endpoints. By Lemma 6.3, it follows that a given ℓ contributes either no endpoint to K^*, or if it does, the number of endpoints is $O(1)$ and this event happens with probability $O(\operatorname{vol} K^*)/\varepsilon_0^2\lambda = O(\varepsilon_0^{-2})/\omega n^{\varepsilon}$. The choice of ℓ is repeated $\lceil \omega \rceil$ times independently, so the expected number is $O(n^{-\varepsilon})$. By a Chernoff-type estimate (Theorem A.1.12 [2]), the probability that the number of endpoints in K^* exceeds $n^{-\varepsilon}\Delta$ is at most $O(1/\Delta)^{O(n^{-\varepsilon}\Delta)}$. Setting $\Delta = bn^{\varepsilon}$, for some large enough constant $b = b(\varepsilon)$, we find that $O(1)$ endpoints lie in K^* with probability at most n^{-10}.

The size of Q is $O(n^3)$, which bounds by $O(n^9)$ the number of triplets of slabs that can be formed in Lemma 6.3. We can thus ensure that with high probability no K^* contains more than a constant number of endpoints. By Fact 6.2, we know that the designated edges of P are all of length at most $\omega/2n$ and so at most $O(1)$ of them can intersect any set of query planes of size at least $c \log n$. (The 2 in the denominator is an overly generous slack factor to account for the distortions resulting from turning ℓ into a cylinder.)

We observed earlier that, because of Lemma 2.3, to achieve a query time of $O(|P \cap \pi| + \omega)$ requires $\Omega(|Q|\omega/\log n)$ storage. By Lemmas 6.4 and 6.5, we now have a polytope P and a set Q of query planes that is $(\Theta(\log n), \omega)$-favorable for the $\Theta(n)$ designated edges of P. The storage requirement is $\Omega(|Q|\omega/\log n) = \Omega(n^{3-\varepsilon}/\omega^3 \log n)$, which, after suitably readjusting ε, proves Theorem 6.1. $\Box$

We prove a nearly matching upper bound for the case where $\omega = O(\log n)$.

Theorem 6.6 *Given an n-vertex polytope in $\mathbf{R}^3$, there is a data structure of size $O(n^3)$ that allows us to compute all k edges intersected by a query plane in $O(k + \log n)$ time.*

Proof: Actually, our solution is more general than that: the input can be any set of line segments in $\mathbf{R}^3$. We begin with an $O(n^4)$ solution, which we improve to $O(n^3)$ in a second stage. We dualize the problem by transforming the endpoints of the input segments into $2n$ planes, using the polarity $ax + by + cz = 1$. A line segment is dualized into a double wedge. We can preprocess the arrangement of $2n$ planes for fast point location [12] so that, given a point (here, the dual of the query plane), the convex cell that contains it can be found in $O(\log n)$ time. If each cell keeps a list of the double wedges that enclose it, then an intersection query can be answered in time $O(k + \log n)$, using $O(n^4)$ storage.

We use filtering search to reduce the storage to $O(n^3)$. Think of the cells as forming the nodes of a graph, with an edge joining two nodes whose corresponding cells share a facet. By doubling each edge, we can form an Eulerian tour that visits all the nodes. Now observe that the wedge lists in the $O(n^4)$ solution have a great deal of "coherence." Indeed, fix a double wedge and observe the nodes of the tour in whose lists it appears: these nodes form intervals along the tour, whose endpoints correspond to an entry into or an exit from the wedge in question. Thus, there are only $O(n^3)$ such intervals. So the problem is reduced to this: given m intervals on a line, where $m = O(n^3)$, build a data structure of size $O(m)$, such that, given a query x, all k intervals that contain x can be reported in $O(k + \log m)$ time. The *window list* of [8] does just that.

References

[1] Agarwal, P.K., Erickson, J. *Geometric range searching and its relatives*, in Advances in Discrete and Computational Geometry, eds. Chazelle, B., Goodman, J.E., Pollack, R., *Contemporary Mathematics* 223, Amer. Math. Soc., 1999, pp. 1–56.

[2] Alon, N., Spencer, J.H. *The Probabilistic Method*, 2nd ed., Wiley-Interscience, 2000.

[3] Arge, L., Vengroff, D.E., Vitter, J.S. *External-memory algorithms for processing line segments in geographic information systems*, Proc. 3rd Annual Euro. Symp. Algo. (1995), 295–310.

[4] Atallah, M.J., Cole, R., Goodrich, M.T. *Cascading divide-and-conquer: A technique for designing parallel algorithms*, SIAM J. Comput. 18 (1989), 499–532.

[5] de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O. *Computational Geometry*, Springer-Verlag, 1997.

[6] Bertino, E., Catania, B., Shidlovsky, B. *Towards optimal indexing for segment databases*, Technical report, University of Milano, Italy, 1998.

[7] Buddhikot, M.M., Suri, S., Waldvogel, M. *Fast layer-4 packet classification using space decomposition techniques*, Proc. Protocols for High Speed Networks, Salem, MA, 1999.

[8] Chazelle, B. *Filtering search: A new approach to query-answering*, SIAM J. Comput. 15 (1986), 703–724.

[9] Chazelle, B. *Lower bounds on the complexity of polytope range searching*, J. Amer. Math. Soc. 2 (1989), 637–666.

[10] Chazelle, B. *Lower bounds for orthogonal range searching: I. The reporting case*, J. ACM 37 (1990), 200–212.

[11] Chazelle, B. *An optimal algorithm for intersecting three-dimensional convex polyhedra*, SIAM J. Comput. 21 (1992), 671–696.

[12] Chazelle, B. *The Discrepancy Method: Randomness and Complexity*, Cambridge University Press, 2000.

[13] Chazelle, B., Edelsbrunner, H., Grigni, M., Guibas, L.J., Hershberger, J. Sharir, M., Snoeyink, J. *Ray shooting in polygons using geodesic triangulations*, Algorithmica 12 (1994), 54–68.

[14] Chazelle, B., Guibas, L.J. *Fractional cascading: I. A data structuring technique, II. Applications*, Algorithmica 1 (1986), 133–191.

[15] Chazelle, B., Rosenberg, B. *Simplex range reporting on a pointer machine*, Comput. Geom.: Theory and Appl. 5 (1996), 237–247.

[16] Dehne, F., Ferreira, A., Rau-Chaplin, A. *Parallel fractional cascading on hypercube multiprocessors*, Comput. Geom.: Theory and Appl. 2 (1992), 141–167.

[17] Dobkin, D.P., Kirkpatrick, D.G. *A linear algorithm for determining the separation of convex polyhedra*, J. Algorithms 6 (1985), 381–392.

[18] Goodrich, M.T. *Efficient parallel techniques for computational geometry*, Ph.D. Thesis, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, 1987.

[19] Hershberger, J., Suri, S. *A pedestrian approach to ray shooting: Shoot a ray, take a walk*, J. Algorithms 18 (1995), 403–431.

[20] Kirkpatrick, D.G. *Optimal search in planar subdivisions*, SIAM J. Comput. 12 (1983), 28–35.

[21] Lakshman, T.V., Stiliadis, D. *High-speed policy-based packet forwarding using efficient multi-dimensional range matching*, Proc. ACM SIGCOMM (1998), 191–202.

[22] Matoušek, J. *Geometric range searching*, ACM Comput. Surv. 26 (1994), 421–461.

[23] Mehlhorn, K. *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*, Springer-Verlag, Berlin, 1984.

[24] Mehlhorn, K., Näher, S. *Dynamic fractional cascading*, Algorithmica 5 (1990), 215–241.

[25] Mehlhorn, K., Näher, S., Alt, H. *A lower bound on the complexity of the Union-Split-Find problem*, SIAM J. Comput. 17 (1988), 1093–1102.

[26] Morimoto, Y., Fukuda, T., Morishita, S., Tokuyama, T. *Implementation and evaluation of decision trees with range and region splitting*, Constraint 2 (1997), 163–189.

[27] Sen, S. *Fractional cascading revisited*, J. Algorithms 19 (1995), 161–172.

[28] Tamassia, R., Vitter, J.S. *Optimal cooperative search in fractional cascaded data structures*, Algorithmica, 15 (1996).

[29] Tarjan, R.E. *A class of algorithms which require nonlinear time to maintain disjoint sets*, J. Comput. System Sci. 18 (1979), 110–127.