SIAM J. COMPUT. © 2006 Society for Industrial and Applied Mathematics
Vol. 35, No. 3, pp. 627–646

SUBLINEAR GEOMETRIC ALGORITHMS*

BERNARD CHAZELLE†, DING LIU†, AND AVNER MAGEN‡

Abstract. We initiate an investigation of sublinear algorithms for geometric problems in two and three dimensions. We give optimal algorithms for intersection detection of convex polygons and polyhedra, point location in two-dimensional triangulations and Voronoi diagrams, and ray shooting in convex polyhedra, all of which run in expected time $O(\sqrt{n})$, where n is the size of the input. We also provide sublinear solutions for the approximate evaluation of the volume of a convex polytope and the length of the shortest path between two points on the boundary.

Key words. sublinear algorithms, approximate shortest paths, polyhedral intersection

AMS subject classifications. 68Q25, 68W05, 68W20, 68W40

DOI. 10.1137/S009753970444572X

1. Introduction.

As an outgrowth of the recent work on property testing, the study of sublinear algorithms has emerged as a field unto itself, and great strides have been made in the context of graph and combinatorial problems [30]. Large geometric datasets often call for algorithms that examine only a small fraction of the input, but it is fair to say that sublinear computational geometry is still largely uncharted territory. If preprocessing is allowed, then, of course, this is an entirely different story [3, 23]. For example, checking whether a point lies in a convex 3-polyhedron can be done in logarithmic time with linear preprocessing. However, little of this technology is of any use with massive datasets, since examining the whole input—let alone preprocessing it—is out of the question. Sublinear algorithms have been given for dynamic problems [17] or in situations where a full multidimensional data structure is available [10]. There has also been work on geometric property testing, both in an approximate [11, 12, 18] and exact [24] setting.

In this paper, sublinearity is understood differently. The input is taken to be in any standard representation with no extra assumptions. For example, a planar subdivision or a polyhedron is given in classical edge-based fashion (e.g., doubly connected edge list (DCEL), winged-edge), with *no extra preprocessing.* This implies that we can pick an edge at random in constant time, but we cannot sample randomly among the neighbors of a given vertex in constant time. Our motivation is twofold: (i) we seek the minimal set of computational assumptions under which sublinearity is achievable; (ii) the assumptions should be realistic and nonrestrictive. Note, for example, that sublinear separation algorithms for convex objects are known [6, 15], but all of them require preprocessing, so they fall outside our model. Under these conditions one might ask whether there exist any interesting "offline" problems that can be solved in sublinear time. The answer is yes. Note that randomization is a necessity be-

*Received by the editors August 16, 2004; accepted for publication (in revised form) June 20, 2005; published electronically January 6, 2006. A preliminary version of this paper appeared in *Proceedings of the* 35th *Annual ACM Symposium on Theory of Computing,* San Diego, 2003, pp. 531–540. This work was supported in part by NSF grants CCR-998817 and ARO grant DAAH04-96-1-0181 and by the NEC Research Institute.

http://www.siam.org/journals/sicomp/35-3/44572.html

†Department of Computer Science, Princeton Univeristy, Princeton, NJ, 08544 (chazelle@cs.princeton.edu, dingliu@cs.princeton.edu).

‡Department of Computer Science, University of Toronto, Toronto, ON, M5S3G4 Canada (avner@cs.toronto.edu).

cause, in a deterministic setting, most problems in computational geometry require looking at the entire input. There has been some (but little) previous work on sublinear geometric algorithms as we define them, specifically point location in two- and three-dimensional Delaunay triangulations of sets of random points [14, 27]. As far as we know, however, these are the only works that fall inside our model. Here is a summary of our results. In all cases, n denotes the input size, and all polyhedra are understood to be in $\mathbf{R}^3$:

- an optimal $O(\sqrt{n}\,)$ time algorithm for checking whether two convex polyhedra intersect, reporting an intersection point if they do and a separating plane if they do not;
- optimal $O(\sqrt{n}\,)$ time algorithms for point location in planar convex subdivisions with $O(1)$ maximum face size and two-dimensional Voronoi diagrams, finding the nearest neighbor on a convex polyhedron, and ray shooting-type problems in convex polyhedra.

In contrast with property testing, it is important to note that our algorithms never err. All the algorithms are of Las Vegas type, and randomization affects the running time but not the correctness of the output.[1] Devroye, Mücke, and Zhu [14] showed that a simple technique for point location in two-dimensional Delaunay triangulations, namely random sampling then walking from the nearest sample to the query, has expected running time (roughly) $O(n^{1/3})$ for n *random* input points and a random query. This does not contradict the optimality of our $O(n^{1/2})$ bound because the points must be chosen randomly in [14].

We also consider optimization problems for which approximate solutions are sought. We give

- an $O(\varepsilon^{-1}\sqrt{n}\,)$ time algorithm for approximating the volume of a convex polytope with arbitrary relative error $\varepsilon > 0$;
- an $O(\varepsilon^{-5/4}\sqrt{n}\,) + f(\varepsilon^{-5/4})$ time algorithm for approximating the length of the shortest path between two points on the boundary of a convex polyhedron with arbitrary relative error $\varepsilon > 0$. Here, $f(n)$ denotes the complexity of the *exact* version of problem. This implies that the complexity of our algorithm is $O(\sqrt{n}\,)$ for any fixed $\varepsilon > 0$.

The shortest path problem for polyhedral surfaces has been extensively studied, drawing its motivation from applications in route planning, injection molding, and computer assisted surgery [1, 21, 26]. In the convex case (the one at hand), an $O(n^3 \log n)$ algorithm was given by Sharir and Schorr [32], later improved by Mitchell, Mount, and Papadimitriou [25] to $O(n^2 \log n)$ and by Chen and Han [7] to $O(n^2)$; therefore, it is known that $f(n) = O(n^2)$. More recently, Kapoor [22] has announced a proof that $f(n) = O(n \log^2 n)$, which would make our algorithm run in time $O(\varepsilon^{-5/4}\sqrt{n}\,)$. This improves on Agarwal et al.'s algorithm [2], which runs in $O(n \log \varepsilon^{-1} + \varepsilon^{-3})$ time for any $\varepsilon > 0$.

Our method makes progress on an important geometric problem of independent interest.

- Given a convex polytope P of n vertices, how many vertices must an enclosing polytope Q have if it is to approximate any (large enough) shortest path on ∂P with relative error at most ε? We reduce to $O(\varepsilon^{-5/4})$ the best previous bound of $O(\varepsilon^{-3/2})$, due to Agarwal et al. [2].

[1]Throughout this paper, unless specified otherwise, the running times are understood in the expected sense.

A Flavor of the Techniques. As a warmup exercise, consider the classical *successor searching* problem: Given a sorted (doubly linked) list of n keys and a number x, find the smallest key $y \geq x$ (the *successor* of x) in the list or report that none exists. It is well known that this smallest key can be found in $O(\sqrt{n})$ expected time [19]. For this, we choose $\sqrt{n}$ list elements at random and find the predecessor and successor of x among those. (Perhaps only one exists.) This provides an entry point into the list, from which a naive search takes us to the successor. To make random sampling possible, we may assume that the list elements are stored in consecutive locations (say, in a table). However—and this is the key point—no assumption is made on the ordering of the elements in the table. (Otherwise we could do a binary search.)

LEMMA 1.1. *Successor searching can be done in $O(\sqrt{n}\,)$ expected time per query, which is optimal.*

Proof. For $i \geq 1$, let Q_i be the set of all elements that are at distance at most i away from the answer on the list (in either direction). Let $P_{>i}$ be the probability of not hitting Q_i after $\sqrt{n}$ random choices of the list elements. The expected distance of the answer to its nearest neighbor in the random sample is $\sum_{i\geq 1} i(P_{>i-1} - P_{>i}) = \sum_{i\geq 0} P_{>i}$. This sum is upper bounded by $\sqrt{n}\sum_{c\geq 0} P_{>c\sqrt{n}} \leq \sqrt{n}\sum_{c\geq 0}(1 - c/\sqrt{n})^{\sqrt{n}} = \sqrt{n}\sum_{c\geq 0} 2^{-\Omega(c)} = O(\sqrt{n})$. This immediately implies that the expected time of the algorithm is $O(\sqrt{n}\,)$.

For the lower bound, we use Yao's minimax principle [33]. We fix a distribution on the input, and we lower-bound the expected complexity of any deterministic algorithm. We then have the same lower bound for randomized algorithms. The input is a linked list containing the numbers 1 through n in sorted order. In our model, the list is represented by a table $T[1 \cdots n]$, with the ith element in the list stored in location $\sigma(i)$ of the table; hence, $T[\sigma(i)] = i$. The input distribution is formed by choosing the permutation σ uniformly from the symmetric group on n elements. In other words, all permutations are equally likely. The query is set to be n. In other words, the problem is to locate the last element in the list. A deterministic algorithm can be modeled as a sequence of steps of the following form: (A) pick a location $T[k]$ already visited and look up the next (or previous) item, i.e., $T[\sigma(i \pm 1)]$, where $k = \sigma(i)$; (B) compute a new index k and look up $T[k]$. Each step may involve the consideration of every piece of information gathered so far. In particular, in a B-step we may not consult either one of the adjacent items in the list before computing k (unless, of course, these items were visited earlier). In this way, $\sigma^{-1}(k)$ of a B-step is equally likely to lie anywhere in the portion of the list still unvisited. For this reason, after a A-steps and b B-steps, there is a probability at least $\left(1 - \frac{\sqrt{n}+a+b}{n}\right)^b$ that none of the last $\sqrt{n}$ elements in the list has been visited in a B-step. Right after the last B-step, either the total number of A- and B-steps exceeds $\sqrt{n}$ or, with constant nonzero probability, at least $\sqrt{n}$ A-steps (some of which may have already been taken) are required to reach the last element in the list. This immediately implies that the expected time of any deterministic algorithm is $\Omega(\sqrt{n}\,)$. $\square$

We can generalize these ideas to polygon intersection. Given two convex polygons P and Q, with n vertices each, determine whether they intersect or not and, if they do, report one point in the intersection. We assume that P and Q are given by their doubly linked lists of vertices (or edges) such that each vertex points to its predecessor and successor in clockwise order. As in successor searching, we assume that the two lists are stored in two tables to allow random sampling.

Choose a random sample of r vertices from each polygon, and let $R_p \subseteq P$ and $R_q \subseteq Q$ denote the two corresponding convex hulls. By two-dimensional linear pro-

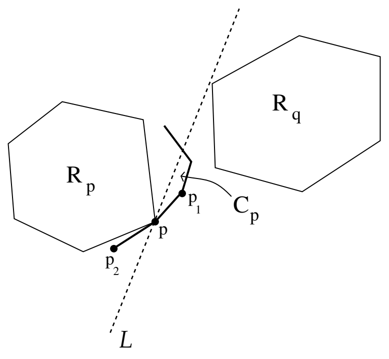

FIG. 1. *Intersecting two convex polygons.*

gramming, we can test R_p and R_q for intersection without computing them explicitly. This can be done probabilistically (or even deterministically) in linear time. There are many ways of doing that (see [5] for references). It is easy to modify the algorithm (of, say, [31]) so that in $O(r)$ time it reports a point in the intersection of R_p and R_q if there is one (in which case we are done) and a bitangent separating line $\mathcal{L}$ otherwise (Figure 1). Let p be the vertex of R_p in $\mathcal{L}$, and let p_1, p_2 be its two adjacent vertices in P. We define a polygon C_p as follows. If neither p_1 nor p_2 is on the R_q side of $\mathcal{L}$, then C_p is the empty polygon. Otherwise, by convexity exactly one of them is (say, p_1). We walk along the boundary of P starting at p_1, away from p, until we cross $\mathcal{L}$ again. This portion of the boundary, clipped by the line $\mathcal{L}$, forms the convex polygon C_p. A similar construction for Q leads to C_q.

It is immediate that $P \cap Q \neq \emptyset$ if and only if P intersects C_q or Q intersects C_p. We check the first condition and, if it fails, check the second one. We restrict our explanation to the case of $P \cap C_q$. First, we check whether R_p and C_q intersect, again using a linear time algorithm for a linear program (LP), and return with an intersection point if they do. Otherwise, we find a line $\mathcal{L}'$ that separates R_p and C_q and, using the same procedure as described above, we compute the part of P, denoted C'_p, on the C_q side of $\mathcal{L}'$. Finally, we test C'_p and C_q for intersection in time linear on their sizes, using an LP or any other straightforward linear-time algorithm for intersection detection of convex polygons. Correctness is immediate. The running time is $O(r + |C_p| + |C'_p| + |C_q| + |C'_q|)$. We can prove that $\mathbf{E}\,|C_p| = O(n/r)$. (The three-dimensional case discussed below will subsume this result, so there is no need for a proof now.) Similarly, $\mathbf{E}\,|C'_p| = \mathbf{E}\,|C_q| = \mathbf{E}\,|C'_q| = O(n/r)$. The overall complexity of the algorithm is $O(r + n/r)$, and choosing $r = \lfloor \sqrt{n} \rfloor$ gives the desired bound of $O(\sqrt{n})$.

To show optimality, consider the following distributions on pairs of polygons. One polygon is fixed, convex, and nondegenerate with one vertex in the origin and all other vertices below the x-axis. The other polygon (also convex and nondegenerate) has $n-1$ vertices above the x-axis, and one vertex, p, in the origin or in $(0, \delta)$, where δ is a positive number small enough so that p is the lowest vertex of the polygon. Moreover, the edges of this polygon are randomly ordered in the edge table. Clearly, these two polygons intersect if and only if p is in the origin. Since nothing in the structure of the input except the geometry of p reveals whether it is indeed the origin, any algorithm that detects intersection must have access to p. Now recall that the only operations allowed are the random sampling of edges and edge-traversing via links, which means that, as in Lemma 1.1, an expected time of $\Omega(\sqrt{n})$ is needed to access p. Optimality of subsequent results follows these lines very closely and shall not be proved again. We have the following.

THEOREM 1.2. *To check whether two convex n-gons intersect can be done in $O(\sqrt{n})$ time, which is optimal.*

To put Theorem 1.2 in perspective, recall that the intersection of two convex polygons can be determined in logarithmic time if the vertices are stored in an array in cyclic order [6]. The key point of our result is that, in fact, a linked list is sufficient for sublinearity. Similarly, if polyhedra are preprocessed à la Dobkin and Kirkpatrick, then fast intersection detection is possible [15]. What we show below is that sublinearity is achievable even with no preprocessing at all. Again, we use a two-stage process: In the first stage we break up the problem into r subproblems of size roughly n/r and then identify which ones actually need to be solved; in the second stage we solve these subproblems in standard (i.e., nonsublinear) fashion. Their number is constant, and hence the square root complexity. What prevents us from solving these subproblems recursively is the model's restriction to *global* random sampling. In other words, one can sample efficiently for the main problem but *not* for the subproblems.

2. Convex polyhedral intersections. Given two n-vertex convex polyhedra P and Q in $\mathbf{R}^3$, the problem is to determine whether or not they intersect: If they do, then we should report a point in the intersection; otherwise, we should report a plane that separates them. We assume that a convex polyhedron is given in any classical edge-based fashion (e.g., DCEL, winged-edge) but with no extra preprocessing. The main structure is a table of edges that allows us to pick an edge at random in constant time. There are also two tables for vertices and faces. Moreover, these tables are interconnected via pointers to make various local operations possible. For example, each edge points to its two vertices and two adjacent faces. It also points to its predecessor and successor edges in its two adjacent faces. Such a structure is a standard representation for convex polyhedra in computational geometry. It allows us to traverse a portion of a convex polyhedron in a local fashion and in time linear in the number of edges visited.

Choose a random sample of $r = \lfloor \sqrt{n} \rfloor$ edges from P and Q, and let R_p and R_q denote the convex hulls of these random edges in P and Q, respectively. We do not compute R_p and R_q explicitly but merely use their vertices to get an LP as described in the last section for the case of polygons. We use this LP to detect the intersection of R_p and R_q in $O(r)$ time by invoking a linear-time algorithm for low-dimensional linear programming. We stop with a point of intersection if there is one. Otherwise, we find a separating plane $\mathcal{L}$ that is tangent to both R_p and R_q. It is important to choose the plane $\mathcal{L}$ in a canonical fashion. To do that, we set up the LP so as to maximize, say, the coefficient α in the equation[2] $\alpha x + \beta y + \gamma z = 1$ of $\mathcal{L}$.

Next, choose a plane π normal to $\mathcal{L}$ and consider projecting P and Q onto it. (Of course, we do not actually do it.) Let p be a vertex of R_p in $\mathcal{L}$ (there could be two of them, but not more, if we assume general position between P and Q), and let p^* be its projection onto π. We also project the neighbors of p in P onto π and get $p_1^*, p_2^*, \ldots, p_k^*$. In other words, they are the set of vertices adjacent to p^* in the projection of P onto π. We test to see if any of them is on the R_q side of $\mathcal{L}$ and identify one such point, p_1, if the answer is yes (more on that below). If none of them

[2]With perturbation techniques, we can always assume general position, and hence avoid having a solution passing through the origin. We will also assume that the relative position of P and Q is general.

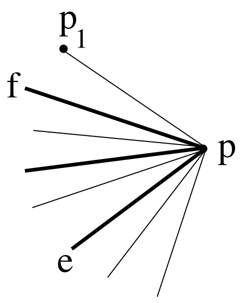

FIG. 2. *The edges of P incident to p; the thick lines form the random sample.*

is on the R_q side, then we define C_p to be the empty polyhedron. This is because in this case, P is completely on the other side of $\mathcal{L}$. Otherwise, we construct the portion of P, denoted C_p, that lies on the R_q side of $\mathcal{L}$. Note that C_p is a convex polytope, not just the boundary of P cut off by $\mathcal{L}$. We compute C_p by using a standard flooding mechanism. Beginning at p_1, we perform a depth-first search through the facial structure of P, restricted to the relevant side of $\mathcal{L}$. Because C_p is convex, the edges form a single connected component, so we never need to leave C_p. This allows us to build the entire facial representation of C_p in time proportional to its number of edges. From then on, the algorithm has the same structure as its polygonal counterpart; i.e., we compute C_p, C'_p, C_q, C'_q and perform the same sequence of tests.

The question is now, How do we find p_1 (if it exists)? To simplify the analysis, once we have p, we resample by picking r edges in P at random; let E be the subset of those incident to p. To find p_1, we project on π all of the edges of E. If there exists an edge of E that is on the R_q side of $\mathcal{L}$, then we identify its endpoint as p_1. Otherwise, all the edges of E lie on one side of $\mathcal{L}$. We then identify the two extreme ones (e and f in Figure 2); being extreme means that all the other projected edges of E lie in the wedge between e and f in π. Assume that e and f are well defined and distinct. Consider the cyclic list V of edges of P incident to p. The edges of E break up V into blocks of consecutive edges. It is not hard to prove that pp_1 lies in a block starting or ending with e or f if such a p_1 (as defined above) exists. So, we examine each of these relevant blocks (at most four) exhaustively. If e and f are not both distinct and well defined, we may simply search for p_1 by checking every edge of P incident to p.

THEOREM 2.1. *Two convex n-vertex polyhedra in $\mathbf{R}^3$ can be tested for intersection in $O(\sqrt{n}\,)$ time; this is optimal.*

Proof. Optimality was already discussed in the polygonal case, and correctness follows from elementary convex geometry, so we limit our discussion to the complexity of the algorithm. Because of the resampling, the expected sizes of the blocks next to e and f (or, alternatively, the expected size of the neighborhood of p if the blocks are not distinct) are $O(n/r)$, so the running time is $O(r + n/r + \mathbf{E}\,(|C_p| + |C_q|))$, where $|C_p|$ (resp., $|C_q|$) denotes the number of edges of C_p (resp., C_q). We may exclude the other two terms $|C'_p|$ and $|C'_q|$, since our upper bound on $\mathbf{E}\,(|C_p| + |C_q|)$ will apply to them as well. Here is how to bound $\mathbf{E}\,(|C_p| + |C_q|)$ by $O(n/r)$.

We modify the sampling distribution a little. Then we argue that reverting back to the original setting does not change the asymptotic value of the upper bound. The modification is twofold: (i) we view $P \cup Q$ as a multiset M where each vertex appears as many times as its number of incident edges; (ii) R_p and R_q are formed by picking each point of M independently with probability r/n and assigning points of P to R_p and points of Q to R_q. With respect to the modified distribution, $|C_p|+|C_q|$ is proportional to the number of constraints in M that violate the LP $\mathcal{P}(R_p, R_q)$ used to define $\mathcal{L}$ (with each point of R_p and R_q defining a linear constraint). For technical reasons, we need to perturb M slightly to make it in general position. Specifically, we move each point of M away from its corresponding vertex in P or Q by an infinitesimal random amount, along the corresponding edge of P or Q. After this random perturbation, the size of the violation set of $\mathcal{P}(R_p, R_q)$ can only increase. To see this, note that constraints in M that violate $\mathcal{P}(R_p, R_q)$ before the perturbation continue to do so because the perturbation is infinitesimal; and constraints in M that lie on $\mathcal{L}$ before the perturbation may violate the LP after the perturbation.

To bound the expected size of the violation set, we apply a result proved by Gärtner and Welzl [19] (and also by Clarkson [9]). Following the notation of the "Sampling Lemma" in [19], we let the ground set S be the (perturbed) set M. After sampling a set of points R from M randomly, we set up an LP $\alpha x + \beta y + \gamma z = 1$ that separates R_p and R_q. Note that in this LP, α, β, and γ are variables and (x, y, z) is a point in R. This LP is set up so as to maximize the variable α, whose optimum value is a function of R. We set the function ϕ in [19] to be α. Under this setting, the extreme elements of R_p and R_q are their vertices on the optimum separating plane (as implied by the above LP). Since M is in general position, any R_p and R_q have three extreme elements. The Sampling Lemma in [19] then implies that the expected size of the violation set is at most $3(n-r)/(r+1) = O(n/r)$.

Let $\mathcal{D}$ be the original distribution (the one used by the actual algorithm) with r replaced by $13r$. Of course, this scaling has no asymptotic effect on the upper bound for $\mathbf{E}\left(|C_p| + |C_q|\right)$. We define an intermediate distribution $\mathcal{D}_1$ by going through each edge (u, v) of $P \cup Q$ twice, selecting it with probability r/n, and then throwing into the sample both u and v, provided that the edge (u, v) has not yet been selected. (Note that this implies that u and v are kept out with probability $(1 - r/n)^2$.) There are at most $6n$ edges in P and Q, so the probability that a sample from $\mathcal{D}_1$ is of size less than $13r$ is overwhelmingly high. Since all equal-size subsets of edges are equally likely to be chosen, $\mathbf{E}_{\mathcal{D}}\left(|C_p|+|C_q|\right)$ is nonincreasing with the sample size, and so $\mathbf{E}_{\mathcal{D}}\left(|C_p|+|C_q|\right) = O(\mathbf{E}_{\mathcal{D}_1}\left(|C_p|+|C_q|\right))$. Let $\mathcal{D}_2$ denote the modified distribution used in the calculations. Observe that $\mathcal{D}_2$ is derived from $\mathcal{D}_1$ by picking only u if (u, v) is chosen the first time it is considered for selection and then only v if it is picked the second time around. By monotonicity, we then have $\mathbf{E}_{\mathcal{D}_1}\left(|C_p| + |C_q|\right) = O(\mathbf{E}_{\mathcal{D}_2}\left(|C_p| + |C_q|\right))$. This proves that the $O(n/r)$ bound holds in the original distribution used by the algorithm.

Recall that the running time is $O(r + n/r + \mathbf{E}\left(|C_p| + |C_q|\right))$, which is $O(r + n/r)$ by the above analysis. For $r = \lfloor \sqrt{n} \rfloor$ it is $O(\sqrt{n})$. $\square$

When the two convex polyhedra intersect, the algorithm reports a point in the intersection. On the other hand, when they are disjoint, we can report a plane that separates them. Here is a brief description on how to do that. Note that we cannot simply return a separating plane for C_p and C'_q (or C'_p and C_q) because it is not necessarily separating for P and Q. Instead, we resort to geometric duality to compute the desired plane in expected $O(\sqrt{n})$ time. In a standard geometric duality transform, a vertex in the primal space is mapped to a plane in the dual space and vice versa. Moreover, the upper (resp., lower) hull of a convex polyhedron is transformed to a lower (resp., upper) envelope [13]. When P and Q are disjoint, at least one of the following must be true: (1) there exists a plane above the upper hull of P and below the lower hull of Q; (2) there exists a plane below the lower hull of P and above the upper hull of Q. Since they are symmetric, it suffices to consider the first one. By duality, such a plane dualizes to a point in the common intersection of an upper and a lower envelope, which is itself a convex polyhedron. Although this polyhedron is not available explicitly, we have access to its geometric features (vertices, edges, etc.) in constant time via the corresponding features in the primal space. Hence we can apply the above algorithm to find, in $O(\sqrt{n})$ time, an intersection point which is the dual of a separating plane for P and Q.

It is important to note that an alternative to the above approach can be found in a general scheme to solve the constant-dimensional LP in linear time due to Clarkson [9]. Clarkson suggested a randomized algorithm that finds a set of constraints of expected size $O(\sqrt{n})$ (or, in general, $O(d\sqrt{n})$, where d is the dimension) that contains a "basis," that is, a minimal set of constraints that determines the problem. Our approach is somewhat similar to that schema. Of course, there are details to be filled as to how exactly this set may be computed in time $O(\sqrt{n})$. (Clarkson's algorithm as it is would be a linear time algorithm.) In particular, an $O(\sqrt{n})$-time method of finding the violating subpolyhedron (like the one we proposed) must still be used in order to implement the alternative approach of Clarkson efficiently enough.

3. Ray shooting applications. Given a convex polyhedron P with n vertices and a directed line ℓ in $\mathbf{R}^3$, the ray shooting problem asks for the point on (the boundary of) P hit by ℓ if it exists. We apply essentially the same techniques as in convex polyhedral intersection to ray shooting and solve it in expected $O(\sqrt{n})$ time. Choose a random sample of $\lfloor \sqrt{n} \rfloor$ edges from P, and let R_p denote the convex hull of these edges. We first use an LP to detect intersection of R_p and ℓ in time $O(\sqrt{n})$. There are two cases. If R_p and ℓ do not intersect, we get a plane $\mathcal{L}$ that separates them and passes through a vertex q of R_p. Starting from q we construct the intersection C_p of P with the halfspace bounded by $\mathcal{L}$ that contains ℓ. We already explained how to do that in the previous section. Finally, we solve ray shooting for C_p and ℓ. Now suppose that R_p and ℓ intersect. We first find the point p on R_p hit by ℓ in time $O(\sqrt{n})$. We cannot afford to compute an explicit representation of R_p in time $\Omega(\sqrt{n}\log n)$. To find p we again use an LP. We can assume that ℓ is the positive x-axis by rotating the coordinate system. Of course, we do not rotate the whole polytope P. Instead, we maintain such a rotation transform implicitly. In other words, whenever we need a geometric feature (vertex, edge, etc.) of P after the rotation, we compute it from its corresponding feature on the original input in constant time. Finding p is equivalent to finding a plane $\mathcal{L}$ such that (1) all vertices of R_p are on one side of $\mathcal{L}$ (the side that contains $(+\infty, 0, 0)$); (2) the intersection point of $\mathcal{L}$ with the x-axis has its x-coordinate as large as possible. In fact, p is that intersection point. It is straightforward to formulate this problem as a three-dimensional LP and solve it in time $O(\sqrt{n})$. In particular, to ensure (2) above we minimize the coefficient α in the equation $\alpha x + \beta y + \gamma z = 1$ for $\mathcal{L}$. Once we have $\mathcal{L}$ and p, we construct C_p as before and solve the problem for C_p and ℓ. Essentially the same analysis as the proof of Theorem 2.1 shows that the expected size of C_p is $O(\sqrt{n})$. We thus have the following.

THEOREM 3.1. *Given a convex polyhedron with n vertices and a directed line, we can compute their intersection explicitly in optimal $O(\sqrt{n})$ time.*

This sublinear time algorithm for ray shooting towards a convex polyhedron gives us useful ammunition for all sorts of location problems.

Given the Delaunay triangulation $\mathcal{T}$ of a set S of n points in the plane and a query point q, consider the problem of locating q, i.e., retrieving the triangle of $\mathcal{T}$ that contains it. The Delaunay triangulation can be given in any classical edge-based data structure (e.g., DCEL), as long as it supports $O(1)$ time access to a triangle from a neighboring triangle. We use the close relationship between Delaunay triangulations and convex hulls given by the mapping $h : (x, y) \mapsto (x, y, x^2 + y^2)$. As is well known, the Delaunay triangulation of S is facially isomorphic to the lower hull of $h(S)$ (i.e., the part of the convex hull that sees $z = -\infty$). In this way, point location in $\mathcal{T}$ is equivalent to ray shooting towards the convex hull, where the ray originates from the query point q and shoots in the positive z-direction. Obviously, any facial feature of the convex hull can be retrieved in constant time from its corresponding feature in the Delaunay triangulation. (The one exception is the set of faces outside the lower hull: we can simplify matters by adding a dummy vertex to the hull at $z = \infty$.)

The same argument can be used for point location in Voronoi diagrams. Recall that each point (p_x, p_y) is now lifted to the plane $Z = 2p_xX + 2p_yY - (p_x^2 + p_y^2)$, which is tangent to the paraboloid $Z = X^2 + Y^2$. The Voronoi diagram of S is isomorphic to the lower envelope of the arrangement formed by the n tangent planes. Note that any vertex (resp., edge) of the envelope can be derived in constant time from the three (resp., two) faces incident to the corresponding vertex (resp., edge).

THEOREM 3.2. *Point location in the Delaunay triangulation or Voronoi diagram of n points in the plane can be done in optimal $O(\sqrt{n}\,)$ time.*

Observe that algorithms for computing a Delaunay triangulation or a Voronoi diagram often supply an efficient point location data structure as a by-product, and thus sublinear time point location in Delaunay triangulations or Voronoi diagrams may be of lesser interest. However, our algorithm is still useful when the triangulation/diagram is huge and we cannot afford to store it together with the point location structure. Our algorithm for point location in Delaunay triangulations also has its limitations: It works only because of the known correspondence between a Delaunay triangulation and a special convex polyhedron. It cannot perform point location in arbitrary planar triangulations. In the next section, we use a different method to achieve sublinear time point location in arbitrary triangulations or convex subdivisions with $O(1)$ maximum face size.

We consider the following problem, which will arise in our subsequent discussion of volume approximation and shortest path algorithms. Given a convex polyhedron P with n vertices and a point q, let $n_P(q)$ denote the (unique) point of P that is closest to q. Of course, we can assume that q does not lie inside P, which we can test by using the previous algorithm. To compute $n_P(q)$ we extract a sample polyhedron R_p of size $\sqrt{n}$ (as we did before) and find $n_{R_p}(q)$. Since we just have a collection of vertices of R_p instead of its full facial representation, it is not obvious how to find $n_{R_p}(q)$ in time $O(\sqrt{n})$. For this purpose, we express this problem as an *LP-type* problem and solve it using the method in [5] (see Chapter 8). A reformulation of the problem would be to seek the plane $\mathcal{L}$ that separates q from the vertices of R_p and maximizes the distance from q to it. To apply the method in [5], we view each vertex of R_p as a constraint. We also check that all the assumptions (i.e., monotonicity, locality, violation test, and range space oracle) needed to solve this problem efficiently hold. See [5] for details. Thus we get $\mathcal{L}$ in time $O(\sqrt{n})$: it is tangent to R_p at $n_{R_p}(q)$ and normal to the segment $qn_{R_p}(q)$. Next, we compute the intersection C_p of P with the halfspace bounded by $\mathcal{L}$ that contains q. Again, a similar analysis shows that the expected size of C_p is $O(\sqrt{n})$. Obviously, $n_P(q) = n_{C_p}(q)$, so we can finish the work by exhaustive search in C_p.

Theorem 3.3. *Given a convex polyhedron P with n vertices and a point q, the nearest neighbor of q in P can be found in $O(\sqrt{n}\,)$ time.*

We can compute a related function by similar means. Given a directed line ℓ, consider an orthogonal system of coordinates with ℓ as one of its axes (in the positive direction), and define $\xi_P(\ell)$ to be any point of P with maximum ℓ-coordinate. If we choose a point q at infinity on ℓ, then $\xi_P(\ell)$ can be chosen as $n_P(q)$, and so we can apply Theorem 3.3.

Another function we can compute in this fashion maps a plane $\mathcal{L}$ and a direction ℓ in $\mathcal{L}$ to the furthest point of P in $\mathcal{L}$ along ℓ: in other words, $\xi_P(\mathcal{L},\ell) = \xi_{\mathcal{P}\cap\mathcal{L}}(\ell)$. Again, the nonobvious part is computing $\xi_{R_p}(\mathcal{L},\ell)$ in time $O(\sqrt{n})$ for a sample polytope R_p. As in the case of ray shooting, we can assume without loss of generality that $\mathcal{L}$ is the xy-plane and ℓ is the positive x-direction. Finding $\xi_{R_p}(\mathcal{L},\ell)$ is the same as finding a plane $\mathcal{L}'$ such that (1) all vertices of R_p are on one side of $\mathcal{L}'$ (the side that contains $(-\infty,0,0)$); (2) $\mathcal{L}'$ is parallel to the y-axis; (3) the intersection point of $\mathcal{L}'$ with the x-axis has its x-coordinate as small as possible. We solve this problem in time $O(\sqrt{n})$ by formulating it as a three-dimensional LP. Other parts of the algorithm (e.g., constructing C_p) and its analysis are similar to other problems discussed in this section. We summarize our results.

Theorem 3.4. *Given a convex polyhedron P with n vertices, a directed line ℓ, and a plane π, the points $\xi_P(\ell)$ and $\xi_P(\pi,\ell)$ can be found in $O(\sqrt{n}\,)$ time.*

4. Point location in convex subdivisions.

Given a convex planar subdivision $\mathcal{S}$ with n edges and a query point q, the point location problem asks for the face of $\mathcal{S}$ that contains q. In the previous section, we provided an $O(\sqrt{n})$-time point location algorithm where $\mathcal{S}$ is a Delaunay triangulation or a Voronoi diagram. Devroye, Mücke, and Zhu [14] also showed that a simple "walk-through" technique locates a query point in the Delaunay triangulation of n *random* points in the plane in expected (roughly) $O(n^{1/3})$ time. Here we show that a slight variation of the walk-through technique actually locates a query point in *any* planar triangulation (not necessarily Delaunay or formed by random points) in expected $O(\sqrt{n})$ time, which is optimal. Our algorithm generalizes to planar subdivisions with $O(1)$ maximum face size. We also give a simple argument showing an $\Omega(n)$ lower bound for point location in subdivisions with large faces.

Theorem 4.1. *Point location in an n-edge convex planar subdivision with $O(1)$ maximum face size can be done in optimal $O(\sqrt{n}\,)$ time.*

Proof. First, we consider the case of a triangulation. For an edge e in the triangulation and a query point q, we use q_e to denote the nearest neighbor of q on e. It is natural to define the Euclidean distance between q and e as $|qq_e|$. We start by sampling $\sqrt{n}$ edges of the triangulation at random. Let e be the edge in the random sample that has the smallest Euclidean distance to q. We walk from q_e toward[3] q by traversing all triangles crossed by qq_e one by one. Given any edge-based representation of the triangulation (such as DCEL), it takes constant time to traverse from one triangle to the next. We stop at the triangle that contains q and output it as the answer.

The running time (besides the sampling stage) is proportional to the number of triangles crossed by qq_e. Thus it suffices to show that the expected number of

[3]It is important to walk towards q from its nearest neighbor on the nearest edge. In contrast, previous algorithms [27] either walk from an endpoint (or the midpoint) of the nearest edge or sample by vertices and walk from the nearest sample vertex. These algorithms do not have sublinear expected running time for arbitrary triangulations.

triangulation edges crossed by qq_e is $O(\sqrt{n})$. For this, we rank each edge according to its Euclidean distance to q. Since the rank of every edge crossed by qq_e is smaller than that of e, the number of edges crossed by qq_e is at most the rank of e. The claimed time bound then follows from the fact that the smallest rank of $\sqrt{n}$ random edges has expectation $O(\sqrt{n})$. It is straightforward to generalize this algorithm to planar subdivisions with $O(1)$ maximum face size. $\square$

What if the subdivision has large faces: Is sublinear time point location still possible? The answer is no.

THEOREM 4.2. *There exists an n-edge planar subdivision such that any randomized algorithm for point location in this subdivision has expected running time $\Omega(n)$.*

Proof. Consider the following problem first: We are given a doubly linked list of numbers. We know exactly one of them is nonzero and want to find out that special number. We can use an argument similar to the proof of Lemma 1.1 to show that any randomized algorithm has to spend $\Omega(n)$ expected time on this problem. Returning to point location, consider a rectangle with corners $(-1, 0)$, $(-1, n + 1)$, $(1, 0)$, and $(1, n+1)$. By breaking its two vertical sides into $n+1$ unit length segments, we get a face with $2n + 4$ edges. Finally, we pick an integer i from 1 to n and add a horizontal edge from $(-1, i)$ to $(1, i)$. This gives us a two-face subdivision. Given the query point $(0, (n + 1)/2)$, a deterministic algorithm must find the horizontal edge in the middle to locate the query correctly, and the only way to do that is through a visit to one of its four adjacent edges. This is similar in spirit to the list-checking problem considered above. In other words, in both problems we try to find one of $O(1)$ special elements in a list[4] of size $\Theta(n)$. We thus get the same lower bound of $\Omega(n)$. $\square$

5. Volume approximation. We seek to approximate the volume of a convex polytope P. We proceed in two stages. First, we compute a large enough enclosed ellipsoid, which we use to rescale P affinely. This is intended to make P *round* enough so that good Hausdorff distance approximation yields good volume approximation. Second, we use a standard construction of Dudley [16] to find, via the methods of the previous section, an enclosing polytope of $O(1/\varepsilon)$ vertices whose boundary is at Hausdorff distance at most ε from P.

STAGE 1. We begin by computing, in $O(\sqrt{n}\,)$ time, a polytope $P' \subseteq P$, such that $\texttt{vol}\,(P') \geq c_0\,\texttt{vol}\,(P)$ for some constant $c_0 > 0$. Compute the six points $\xi_P(\ell)$ for $\ell = \pm x,\ \pm y,\ \pm z$. These points come in pairs, so let w_1, w_2 be the pair forming the largest distance. Given a point w on the line $\mathcal{L}$ passing through w_1 and w_2, let P_w denote the intersection of P with the plane through w that is orthogonal to $\mathcal{L}$. Let w_0 be the midpoint of w_1w_2 (Figure 3). We first show that if S is a set of points in P_{w_0} such that

$$\texttt{area}\,(\texttt{conv}\,(S)) \geq c_1\,\texttt{area}\,(P_{w_0}) \tag{1}$$

for some constant $c_1 > 0$, then $\texttt{vol}\,(\texttt{conv}\,(S \cup \{w_1, w_2\})) \geq c_2\texttt{vol}\,(P)$ for some other constant $c_2 > 0$. Therefore, we can take $P' = \texttt{conv}\,(S \cup \{w_1, w_2\})$ to achieve our goal. Indeed, assume we have such a set S. As a straightforward consequence of Pythagorean theorem, we find that $\texttt{diam}\,(P) \leq \sqrt{3}\,d(w_1, w_2)$; therefore, the orthogonal projection of P on $\mathcal{L}$ is a segment $v_1v_2 \supseteq w_1w_2$ of length at most $\sqrt{3}\,d(w_1, w_2)$. This implies that, for any w in $\mathcal{L}$, $\texttt{area}\,(P_w) \leq 12\,\texttt{area}\,(P_{w_0})$. To see why, observe that if, say, $w \in v_1w_0$, then, by convexity, P_w is enclosed in the cone with apex w_2 and base

[4]In the point location problem there are two lists of vertical edges instead of one. This requires only a small modification of the argument.

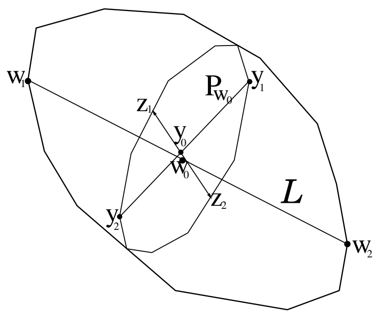

FIG. 3. *Approximating P from within.*

P_{w_0}. Therefore, P_w lies in a copy of P_{w_0} scaled by at most $d(w, w_2)/d(w_0, w_2) \leq 2\sqrt{3}$, which proves our claimed upper bound on $\texttt{area}\,(P_w)$. Of course, the same argument can be repeated if $w \in w_0v_2$. Since $\texttt{vol}\,(P) = \int_{v_1}^{v_2} \texttt{area}\,(P_w)\, dw$, we can conclude that the four quantities

$$\texttt{vol}\,(P), \qquad \texttt{vol}\,(\texttt{conv}\,(P_{w_0} \cup \{v_1, v_2\})),$$

$$\texttt{vol}\,(\texttt{conv}\,(P_{w_0} \cup \{w_1, w_2\})), \qquad \texttt{vol}\,(\texttt{conv}\,(S \cup \{w_1, w_2\}))$$

are all equal up to within constant factors.

We now show how to find a set S satisfying (1). We essentially repeat in two dimensions what we did so far in three dimensions. Specifically, we take a, b to be two mutually orthogonal vectors both normal to $\mathcal{L}$, and let π be the plane spanned by a and b. We compute the four points (two pairs) $\xi_P(\pi, \ell)$ for $\ell = a, -a, b, -b$. Let y_1, y_2 be the more distant pair (analogous to w_1, w_2 before). Let y_0 be the midpoint of y_1, y_2, and let segment ℓ_{y_0} be the intersection of P with the line in π orthogonal to y_1y_2. We can find the two endpoints z_1, z_2 of ℓ_{y_0} using ray shooting. Using almost the same argument as the one showing that $\texttt{conv}\,(P_{w_0} \cup \{w_1, w_2\})$ has a volume proportional to $\texttt{vol}\,(P)$, we get that the quadrilateral with vertex set $S = \{y_1, y_2, z_1, z_2\}$ has an area proportional to $\texttt{area}\,(P_{w_0})$, and thus satisfies (1). We comment that a similar approach to the one we described above was used by Barequet and Har-Peled [4]. The difference is that they approximate the volume of a convex polytope from outside by a bounding box, whereas we approximate it from within.

Let $\mathcal{E}$ be the largest ellipsoid enclosed in $P' = \texttt{conv}\,(\{y_1, y_2, z_1, z_2, w_1, w_2\})$, also known as the Löwner–John ellipsoid. It is computable in constant time within any fixed relative error by solving a constant-size quadratic program [20]. As is well known, its volume is at least $(1/\text{dim})^2$ times that of the enclosing polytope; therefore,

$$\texttt{vol}\,(\mathcal{E}) \geq \frac{1}{9}\,\texttt{vol}\,(P') \geq c \cdot \texttt{vol}\,(P)$$

for some constant $c > 0$. Make the center of the ellipsoid the origin of the system of coordinates and use the ellipsoid's positive semidefinite matrix to rescale P. To do that, we consider the linear transformation that takes the ellipsoid into a ball of the same volume. Specifically, if $x^T A^T Ax \leq 1$ is the equation of the ellipsoid, then we consider the transformation $T = A/(\det A)$. The polytope TP has the same volume as P, but it is *round*; namely, it contains a ball B of volume $\Omega(\texttt{vol}\,(TP))$. Thus, we might as well assume that P has this property to begin with. Note that P is also enclosed in a concentric ball B' that differs from B by only a constant-factor scaling. (If not, then TP would contain a point p so far away from B that the convex hull of p and B, although contained in P, would have volume much larger than $\texttt{vol}\,(B)$, and hence $\texttt{vol}\,(P)$, which would give a contradiction.) Finally, by rescaling we can also assume that P is enclosed in the unit ball and its volume is bounded below by a positive constant. By Theorems 3.1 and 3.4, all of the work in Stage 1 can be done in $O(\sqrt{n}\,)$ time.

Stage 2. We implement Dudley's construction [16] of a convex polytope Q such that (i) $Q \supseteq P$; (ii) $Q \subset P_\varepsilon$, where P_ε is the Minkowski sum of P with a ball of radius ε; (iii) Q has $O(1/\varepsilon)$ vertices. Dudley's result was used constructively in [2]. The difference here is that our implementation is sublinear. We compute an $\sqrt{\varepsilon}$-net on the unit sphere,[5] and project this net down to ∂P, using the nearest neighbor function n_P as a projection map. Finally, we form Q as the intersection of the $O(1/\varepsilon)$ halfspaces bounded by the appropriate tangent planes passing through the vertices of the projected net. With suitable use of the nearest neighbor algorithm of Theorem 3.3, we can implement the entire construction in time $O(\varepsilon^{-1}\sqrt{n}\,)$ for the projection construction (since the facial representations of P and TP are the same, the algorithm can use TP as though it had its full facial representation at its disposal) and $O(\varepsilon^{-1}\log\varepsilon^{-1})$ for intersecting the halfspaces needed to form Q. Since we can obviously assume that Q does not have more vertices than P, there is no need for ε to be smaller than, say, $1/n^2$. This implies that the entire construction time is dominated by $O(\varepsilon^{-1}\sqrt{n}\,)$. (In fact, we can further reduce this running time to roughly $O(\varepsilon^{-1/2}\sqrt{n}\,)$ by exploiting the fact that the $O(\varepsilon^{-1}\,)$ nearest neighbor queries can be answered in a more efficient batch mode. Similarly, we can also get a slight improvement on the running time in Theorem 6.2.)

We now show that $\texttt{vol}\,(Q) = (1+O(\varepsilon))\texttt{vol}\,(P)$. Recall that P is "sandwiched" between two concentric balls B and B' such that $\texttt{rad}\,(B') = 1$ and $\texttt{rad}\,(B) = \Omega(1)$. We may assume that B and B' are centered at the origin. Since $Q \subset P_\varepsilon$, we have $\texttt{vol}\,(Q - P) \le \texttt{vol}\,(P_\varepsilon - P) \le \texttt{area}\,(P_\varepsilon)\cdot 2\varepsilon \le \texttt{area}\,(B_{1+2\varepsilon})\cdot 2\varepsilon = O(\varepsilon)$, where $B_{1+2\varepsilon}$ is a ball centered at the origin with radius $1+2\varepsilon$. The upper bound on $\texttt{vol}\,(P_\varepsilon - P)$ is obtained by integration over thin shells of increasing area from ∂P to ∂P_ε. Since $\texttt{vol}\,(P) = \Omega(1)$, we then have $\texttt{vol}\,(Q) = (1+O(\varepsilon))\texttt{vol}\,(P)$.

Theorem 5.1. *Given any $\varepsilon > 0$, it is possible to approximate the volume of an n-vertex convex polytope with arbitrary relative error $\varepsilon > 0$ in time $O(\varepsilon^{-1}\sqrt{n}\,)$.*

6. Approximate shortest paths.

Given a convex polyhedron P with n vertices and two points s and t on its boundary ∂P, the problem is to find the shortest path between s and t outside the interior of P. It is well known that the shortest path lies on the boundary ∂P. In fact, it is easy to construct instances where any reasonable approximation of the shortest path on ∂P involves $\Omega(n)$ edges. This rules out sublinear algorithms, unless we are willing to follow paths outside of P. We show how to compute a path between s and t whose length exceeds the minimum by a factor of at most $1+\varepsilon$ for any $\varepsilon > 0$.

Our algorithm relies on a new result of independent interest. Let $d_P(s,t)$ denote the length of the shortest path between s and t in ∂P. Given a point $v \in \partial P$, let H_v be the supporting plane of P at v (or any such plane if v is a vertex), and let H_v^+ denote the halfspace bounded by H_v that contains P. Given $\varepsilon > 0$, we say that a

[5]This is a collection of $O(\varepsilon^{-1})$ points on the sphere such that any spherical cap of radius $\sqrt{\varepsilon}$ contains at least one of the points.

convex polytope Q is an *ε-wrapper* of P if (c_0 is an absolute constant discussed below)

(i) Q encloses P;

(ii) the Hausdorff distance between ∂P and ∂Q is at most $\varepsilon\,\texttt{diam}\,(P)$;

(iii) given any $s, t \in \partial P$ such that $d_P(s,t) \geq c_0\,\texttt{diam}\,(P)$, $d_{\widehat{Q}}(s,t) \leq (1+\varepsilon) d_P(s,t)$, where $\widehat{Q} = Q \cap H_s^+ \cap H_t^+$.

LEMMA 6.1. *Any convex 3-polytope has an ε-wrapper of size $O(1/\varepsilon)^{5/4}$ for any $\varepsilon > 0$.*

This result improves on the $O(1/\varepsilon)^{3/2}$ bound of Agarwal et al. [2]. The use of a wrapper is self-evident. First, we clip the polytope to ensure that $d_P(s,t) \geq c_0\,\texttt{diam}\,(P)$ (section 6.1). Next, we compute an ε-wrapper (section 6.2) and approximate the shortest path between s and t by computing the shortest path between the two points in $\partial\widehat{Q}$. This can be done in quadratic time by using an algorithm by Chen and Han [7]. The resulting path, which is of length $(1+O(\varepsilon))d_P(s,t)$, can be shortened to $(1+\varepsilon)d_P(s,t)$ by rescaling ε suitably. Note that in (iii) the condition on s and t being sufficiently far apart is essential. It is a simple exercise to show that no variant of a wrapper can accommodate all pairs (s,t) simultaneously. If $f(n)$ denotes the complexity of the *exact* version of problem, then we have the following.

THEOREM 6.2. *Given any $\varepsilon > 0$ and two points s, t on the boundary of a convex polytope P of n vertices, it is possible to find a path between s and t outside P of length at most $(1+\varepsilon)d_P(s,t)$ in time $O(\varepsilon^{-5/4}\sqrt{n}\,) + f(\varepsilon^{-5/4})$.*

We refer the reader back to the introduction for a discussion of the implication of this result in view of the state-of-the-art on the function $f(n)$.

6.1. Computing short paths.

Given two points $s, t \in \partial P$, our first task is to ensure that $d_P(s,t) \geq c_0\,\texttt{diam}\,(P)$ for some constant $c_0 > 0$. To do this, we first compute a value δ such that $\delta \leq d_P(s,t) \leq 8\delta$. We will substitute for P the intersection P' of P with a clipping box centered at s of side length 16δ. Obviously, the shortest paths between s and t relative to P and P' are identical. The only computational primitive we need is the nearest neighbor function of Theorem 3.3. Note that we need only this function relative to P (not to P'). In fact, we first use this function to compute a few sample points on ∂P (see section 6.2). We then discard sample points that are outside of the clipping box. The remaining points together with the clipping box are used to compute an ε-wrapper of P'.

To compute a constant-factor approximation for $d_P(s,t)$, we adapt an algorithm of Har-Peled [21] to our sublinear setting. All that is needed is an implementation of the following primitive: Given two rays r_1, r_2 from a fixed point $p \in P$, let H be the plane spanned by these two rays, and let C denote the two-dimensional cone in H wedged between r_1 and r_2. Given an additional query ray $r \in H$ (not necessarily emanating from p), we need to compute $\xi_{C\cap P}(H, r)$. By Theorem 3.4, this can be done in $O(\sqrt{n}\,)$ time.

6.2. The ε-wrapper construction.

Assuming without loss of generality that $\texttt{diam}\,(P) = 1$, it suffices to prove the following.

THEOREM 6.3. *Given any $\varepsilon > 0$ and a convex polytope P of n vertices with diameter 1, there exists a convex polytope Q with $O(\varepsilon^{-5/4})$ vertices such that* (i) *$Q \supseteq P$;* (ii) *the Hausdorff distance between ∂P and ∂Q is $O(\varepsilon)$; and* (iii) *given any $s, t \in \partial P$ such that $d_P(s,t) \geq c_0$ for some constant c_0, $d_{\widehat{Q}}(s,t) \leq (1+O(\varepsilon))d_P(s,t)$, where $\widehat{Q} = Q \cap H_s^+ \cap H_t^+$.*

We first show how to construct Q. Let S be a sphere of radius 2 centered at some arbitrary point in P. Draw a grid G of longitudes and latitudes on S, so that

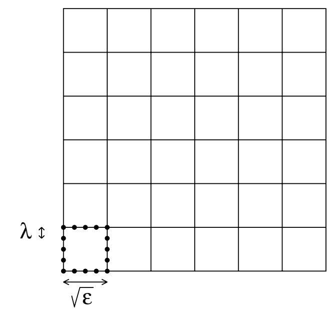

FIG. 4. *The grid G.*

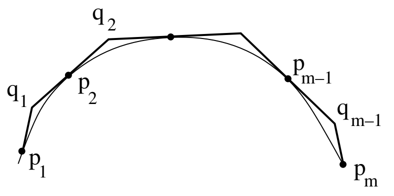

FIG. 5. *The path σ.*

each cell is of length $\sqrt{\varepsilon}$ by $\sqrt{\varepsilon}$ (with an exception made for the last latitude and longitude if $\sqrt{\varepsilon}$ does not divide π). All lengths in this discussion are Euclidean, *except* in this case where the length of a circular arc refers to its corresponding angle. We choose a parameter $\lambda = \varepsilon^{3/4}$ and subdivide each side of a cell into subarcs of length λ (Figure 4). In this way each cell has $O(\sqrt{\varepsilon}/\lambda)$ vertices, and the whole construction defines a set V of $O(1/\lambda\sqrt{\varepsilon}\,)$ vertices. For each point $v \in V$, we compute $n_P(v)$, its nearest neighbor in ∂P, and define

$$Q = \bigcap \{\, H^+_{n_P(v)} \,|\, v \in V \,\}. \tag{2}$$

It is immediate from our choice of λ that Q has $O(\varepsilon^{-5/4})$ vertices.[6] Every point of the sphere S has at least one vertex of G at distance $O(\sqrt{\varepsilon}\,)$. By a result of Dudley [16], this implies part (ii) of Theorem 6.3. Since (i) is obvious, it remains for us to prove (iii).

Borrowing terminology from Agarwal et al. [2], we say that a pair $(\sigma, \mathcal{H})$ forms a *supported path* of P if $\sigma = p_1, q_1, p_2, q_2, \ldots, q_{m-1}, p_m$ is a polygonal line disjoint from the interior of P and $\mathcal{H} = H_{p_1}, \ldots, H_{p_m}$ is a sequence of supporting planes of P, such that $q_{i-1}p_i$ and p_iq_i both lie in H_{p_i}, with $q_0 = p_1$ and $q_m = p_m$ (Figure 5). For $0 < i < m$, the *folding angle* α_i at q_i is the dihedral angle of the wedge between H_{p_i} and $H_{p_{i+1}}$ (the one that does not contain P). The folding angle of σ is defined as $\alpha(\sigma) = \sum_{0<i<m} \alpha_i$.

LEMMA 6.4 (Agarwal et al. [2]). *Given $s, t \in \partial P$, there exists a supported path σ of P with $O(1/\varepsilon)$ edges, joining s and t, such that*

$$d_P(s,t) \le |\sigma| \le (1+\varepsilon) d_P(s,t) \qquad \textit{and} \qquad \alpha(\sigma) = O(\varepsilon^{-1/2}).$$

[6]To be precise, we actually want to compute an ε-wrapper for P' instead of P. For this, we need to remove from the point set V all vertices whose nearest neighbors in ∂P fall outside of the clipping box in section 6.1. We also need to clip Q by that box. We omit these minor details.

To help build intuition for the remainder of our discussion, it is useful to sketch the proof of the lemma. Mapping the grid G to P via the nearest neighbor function n_P creates a grid $n_P(G)$ on ∂P (with curved, possibly degenerate edges). It is convenient to think of P as a smooth manifold by infinitesimally rounding the vertices and edges. It does not much matter how we do that, as long as the end result endows each point $p \in \partial P$ with an (outward) unit normal vector η_p that is a continuous function of p. Note that in this way, for any $u \in S$, the vectors $un_P(u)$ and $\eta_{n_P(u)}$ are collinear, and the function n_P is a bijection. The fundamental property of the nearest neighbor function is that it is nonexpansive. We need only a weak version of that fact, which follows directly from Lemmas 4.3 and 4.4 in [16].

LEMMA 6.5 (Dudley [16]). *Given two points $p, q \in \partial P$, $|pq|$ and $\angle(\eta_p, \eta_q)$ are both in $O(\,|n_P^{-1}(p)n_P^{-1}(q)|\,)$.*

This implies that, for any two points $p, q \in \partial P$ in the same cell of the mapped grid $n_P(G)$, both $|pq|$ and $\angle(\eta_p, \eta_q)$ are in $O(\sqrt{\varepsilon}\,)$. We shortcut the shortest path on ∂P from s to t to form a supported path σ that passes through each cell at most once. In this manner, we identify $O(1/\varepsilon)$ points $p_1, \dots, p_m$ on ∂P, where p_i (resp., p_{i+1}) is the entry (resp., exit) point of the path through the ith cell in the sequence. The points p_i lie on the edges of $n_P(G)$. There are two exceptions, $p_1 = s$ and $p_m = t$, which might lie in the interior of the cell. Next, we connect each pair (p_i, p_{i+1}) by taking the shortest path on $H_{p_i} \cup H_{p_{i+1}}$. The path intersects $H_{p_i} \cap H_{p_{i+1}}$ at a point denoted q_i. (Note that q_i might be infinitesimally close to p_i.) This forms a supported path σ with $O(1/\varepsilon)$ vertices $s = p_1, q_1, p_2, q_2, \dots, q_{m-1}, p_m = t$. The only real difference from the proof in [2] is that we skip the final "trimming" step and keep the points p_i unchanged. We mention two useful, immediate consequences of Lemma 6.5.

- The folding angle at q_i is $O(\sqrt{\varepsilon}\,)$.
- For each $1 \le i \le m$, the point p_i belongs to ∂P and, for $i \ne 1, m$, there exists a point $w_i = n_P(v_i)$, where $v_i \in V$, such that both $|p_i w_i|$ and $\angle(\eta_{p_i}, \eta_{w_i})$ are in $O(\lambda)$.

From σ we build a curve σ' of length $(1 + O(\varepsilon))|\sigma|$ that joins s and t outside the interior of $\widehat{Q}$. The classical result below shows that the shortest path on $\partial\widehat{Q}$ from s to t cannot be longer than σ', which proves Theorem 6.3.

THEOREM 6.6 (Pogorelov [29]). *Given a convex body C, let γ be a curve joining two points $s, t \in \partial C$ outside the interior of C. Then the length of γ is at least that of the shortest path joining s and t on ∂C.*

We now explain how to construct σ'. For $0 < i < m$, let (p_i, η_{p_i}) and (q_i, η_{p_i}) be the rays emanating from p_i and q_i, respectively, in the direction normal to H_{p_i} away from P. Together with the segments $p_i q_i$ and $q_i p_{i+1}$, the four rays (p_i, η_{p_i}), (q_i, η_{p_i}), $(q_i, \eta_{p_{i+1}})$, and $(p_{i+1}, \eta_{p_{i+1}})$ define a polyhedral surface Σ_i, which consists of two unbounded rectangles, Σ_i^1 and Σ_i^3, joined together at q_i by an unbounded triangle, Σ_i^2 (Figure 6). Note that the surface is in general nonplanar, but Σ_i^2 is always normal to the line $H_{p_i} \cap H_{p_{i+1}}$. Out of Σ_i we carve a polyhedral strip S_i as follows. Fix a large enough constant $c > 0$, and let K_i denote the plane $H_{p_i} + c\lambda^2 \eta_{p_i}$. In other words, K_i is a parallel copy of H_{p_i} translated by $c\lambda^2$ away from P. As usual, the superscripted K_i^+ denotes the halfspace enclosing P. Recall that w_i is the nearest neighbor of v_i defined earlier. We need to consider

$$S_i = \Sigma_i \cap \Big\{ (K_i^+ \cap K_{i+1}^+) \cup (H_{w_i}^+ \cap H_{w_{i+1}}^+) \Big\}.$$

Again, we have two exceptions for $i = 1, m-1$, where we use $H_{p_1}^+$ instead of $H_{w_1}^+$ and

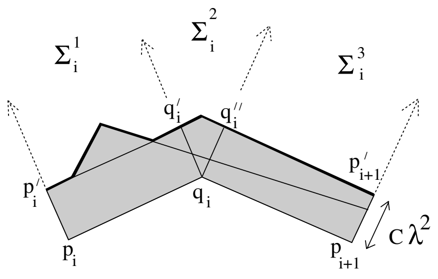

FIG. 6. *The curve σ_i'.*

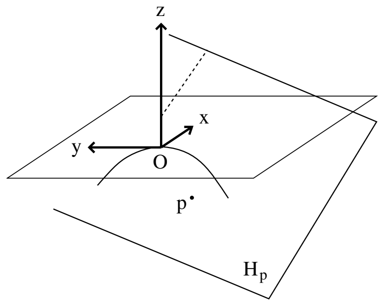

FIG. 7. *How H_p intersects the xz plane.*

$H_{p_m}^+$ instead of $H_{w_m}^+$.

Let $p_i p_i'$ be the edge of S_i incident to p_i collinear with η_{p_i}. We denote by σ_i' the portion of ∂S_i between p_i' and p_{i+1}' and define σ' as $\bigcup_{0<i<m} \sigma_i'$. To provide a connection to s and t, we also add to σ' the segments $p_1 p_1'$ and $p_m p_m'$. To show that σ' is a connected curve outside the interior of $\widehat{Q}$ of length $(1+O(\varepsilon))|\sigma|$ requires a simple technical lemma.

LEMMA 6.7 (Figure 7). *Given an orthogonal system of reference (O, xyz), assume that P is tangent to the xy-plane at O and lies below it. Given a point p on ∂P, if $|n_P^{-1}(O) n_P^{-1}(p)| < \delta$, for some small enough $\delta > 0$, then the intersection of H_p with the xz-plane has for an equation $Z = aX + b$, where $|a| = O(\delta)$ and $0 \leq b = O(\delta^2)$.*

Proof. By Lemma 6.5, the normal to H_p forms a small angle $\theta = O(\delta)$ with the z-axis, so the plane H_p, being nonparallel to the z-axis, can be expressed as $Z = aX + cY + b$. The cross product between the normal $(a, c, -1)$ and the z-axis vector is the vector $(c, -a, 0)$. By the cross product formula, its length, which is $\sqrt{a^2+c^2}$, is also equal to $\sqrt{a^2+c^2+1}\sin\theta$. It follows that $a^2+c^2 = O(a^2+c^2+1)\delta^2$; therefore,

$$a^2 + c^2 = \frac{O(\delta^2)}{1 - O(\delta^2)} = O(\delta^2), \tag{3}$$

and hence $|a| = O(\delta)$. By convexity of P, the plane H_p intersects the nonnegative part of the z-axis, and p_z, the z-coordinate of p, is nonpositive. By (3) and $|Op| = O(\delta)$, it follows that

$$0 \leq b = p_z - ap_x - cp_y \leq \sqrt{a^2+c^2}\sqrt{p_x^2+p_y^2} = O(\delta^2). \qquad \square$$

We examine each σ_i' separately, omitting the cases $i = 1, m-1$, which are trivial modifications of the general case $1 < i < m-1$. The curve σ_i' lies outside the interior of $H_{w_i}^+ \cap H_{w_{i+1}}^+$, and hence of $\widehat{Q}$. It is naturally broken up into three parts, $\sigma_i^j \subset \Sigma_i^j$ ($j = 1, 2, 3$), each one of them being a polygonal curve whose edges lie in any one of four planes: K_i, K_{i+1}, H_{w_i}, and $H_{w_{i+1}}$. Applying Lemma 6.7 with $(p_i, \overrightarrow{p_iq_i}, \overrightarrow{p_ip_i'})$ in the role of (O, x, z) and w_i in the role of p, we find that H_{w_i} intersects the segment p_ip_i' for c large enough (note that this c is the one in the definition of K_i); similarly, $H_{w_{i+1}}$ intersects $p_{i+1}p_{i+1}'$. This shows that p_i' is the intersection of the ray (p_i, η_{p_i}) with the plane K_i; therefore, p_i' is *the same point* in the definition of σ_i' and σ_{i-1}', thus proving that the curve σ' is, indeed, connected. (The danger was having p_i' defined by $H_{w_{i+1}}$.) We now bound the length of σ_i'.

- By Lemma 6.7 the slopes of the edges of σ_i^1 are chosen among 0 for K_i; $O(\sqrt{\varepsilon})$ for K_{i+1}; $O(\lambda)$ for H_{w_i}; and $O(\sqrt{\varepsilon})$ for $H_{w_{i+1}}$. It follows that $|\sigma_i^1| \leq |p_iq_i|/\cos\theta$, where $\theta = O(\sqrt{\varepsilon})$; therefore, $|\sigma_i^1| = (1 + O(\varepsilon))|p_iq_i|$. The same argument shows that $|\sigma_i^3| = (1 + O(\varepsilon))|q_ip_{i+1}|$.
- Let q_i', q_i'' be the endpoints of the curve σ_i^2 (Figure 6), and let a, a', b, b' be the distances along the ray (q_i, η_{p_i}) from q_i to K_i, K_{i+1}, H_{w_i}, and $H_{w_{i+1}}$, respectively. By definition of S_i,

$$|q_iq_i'| = \max\Big\{ \min\{a, a'\}, \min\{b, b'\} \Big\}.$$

Obviously, $a = c\lambda^2$ and, by Lemma 6.7, $b = O(\lambda|p_iq_i| + \lambda^2)$. This implies that $|q_iq_i'| = O(\lambda|p_iq_i| + \lambda^2)$ and, by the same argument,

$$|q_iq_i'| + |q_iq_i''| = O(\lambda(|p_iq_i| + |q_ip_{i+1}|) + \lambda^2).$$

Within Σ_i^2, the curve σ_i^2 is a polygonal line consisting of at most a constant number of edges. It is not difficult to see that for any vertex v of σ_i^2 (including q_i' and q_i''), the angle between q_iv and edges of σ_i^2 incident to v is $\pi/2 \pm O(\sqrt{\varepsilon})$. This follows from a simple geometric observation: given any plane H whose normal makes with q_iv an angle at most α, the angle formed by q_iv and any line on H lies in the range $[\pi/2 - \alpha, \pi/2 + \alpha]$. Since any of the edges of σ_i^2 lies on one of four planes, K_i, K_{i+1}, H_{w_i}, and $H_{w_{i+1}}$, and the normal of each of them makes an angle of $O(\sqrt{\varepsilon})$ with q_iv, the claim follows. Because the folding angle of $O(\sqrt{\varepsilon})$ can be assumed to be less than, say, $\pi/2$, this implies that the curve σ_i^2 lies entirely at a distance $O(|q_iq_i'| + |q_iq_i''|)$ from q_i. It follows that $|\sigma_i^2| = O(|q_iq_i'| + |q_iq_i''|)\sqrt{\varepsilon}$.

Putting everything together we find that

$$|\sigma_i'| = (1 + O(\varepsilon) + O(\lambda\sqrt{\varepsilon}))(|p_iq_i| + |q_ip_{i+1}|) + O(\lambda^2\sqrt{\varepsilon}).$$

In view of the fact that $|p_1p_1'| = |p_mp_m'| = c\lambda^2$, summing up over all $|\sigma_i'|$'s (there are $O(1/\varepsilon)$ of them),

$$\begin{aligned} |\sigma'| &= (1 + O(\varepsilon) + O(\lambda\sqrt{\varepsilon}))|\sigma| + O(\lambda^2/\sqrt{\varepsilon}) \\ &= (1 + O(\varepsilon))|\sigma| + O(\varepsilon) \\ &= (1 + O(\varepsilon))|\sigma|, \end{aligned}$$

which completes the proof of Theorem 6.3. Note that the setting of λ is made to ensure that the additive term $O(\lambda^2/\sqrt{\varepsilon})$ is $O(\varepsilon)$.

Acknowledgments. We wish to thank Pankaj Agarwal, Funda Ergun, Sariel Har-Peled, Joe Mitchell, and Ronitt Rubinfeld for several helpful discussions. We also wish to thank the anonymous referees for their comments and suggestions.

REFERENCES

[1] P. K. Agarwal, S. Har-Peled, and M. Karia, *Computing approximate shortest paths on convex polytopes*, Algorithmica, 33 (1999), pp. 227–242.

[2] P. K. Agarwal, S. Har-Peled, M. Sharir, and K. Varadarajan, *Approximating shortest paths on a convex polytope in three dimensions*, J. ACM, 44 (1997), pp. 567–584.

[3] P. K. Agarwal and J. Erickson, *Geometric range searching and its relatives*, in Advances in Discrete and Computational Geometry, B. Chazelle, J. E. Goodman, and R. Pollack, eds., Contemp. Math. 223, AMS, Providence, RI, 1999, pp. 1–56.

[4] G. Barequet and S. Har-Peled, *Efficiently approximating the minimum-volume bounding box of a point set in three dimensions*, J. Algorithms, 38 (2001), pp. 91–109.

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

[6] B. Chazelle and D. P. Dobkin, *Intersection of convex objects in two and three dimensions*, J. ACM, 34 (1987), pp. 1–27.

[7] J. Chen and Y. Han, *Shortest paths on a polyhedron*, in Proceedings of the Sixth Annual Symposium on Computational Geometry, ACM, New York, 1990, pp. 360–369.

[8] K. L. Clarkson and P. W. Shor, *Applications of random sampling in computational geometry*, II, Discrete Comput. Geom., 4 (1989), pp. 387–421.

[9] K. L. Clarkson, *Las Vegas algorithms for linear and integer programming when the dimension is small*, J. ACM, 42 (1995), pp. 488–499.

[10] A. Czumaj, F. Ergun, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler, *Sublinear-time approximation of Euclidean minimum spanning tree*, in Proceedings of the Fourteenth Annual Symposium on Discrete Algorithms, ACM, New York, 2003, pp. 813–822.

[11] A. Czumaj and C. Sohler, *Property testing with geometric queries*, in Proceedings of the 9th Annual European Symposium on Algorithms, Springer-Verlag, Heidelberg, 2001, pp. 266–277.

[12] A. Czumaj, C. Sohler, and M. Ziegler, *Property testing in computational geometry*, in Proceedings of the 8th Annual European Symposium on Algorithms, Springer-Verlag, Heidelberg, 2000, pp. 155–166.

[13] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin, 1997.

[14] L. Devroye, E. P. Mücke, and B. Zhu, *A note on point location in Delaunay triangulations of random points*, Algorithmica, 22 (1998), pp. 477–482.

[15] D. P. Dobkin and D.G. Kirkpatrick, *Determining the separation of preprocessed polyhedra—a unified approach*, in Automata, Languages and Programming, Springer-Verlag, New York, 1990, pp. 400–413.

[16] R. M. Dudley, *Metric entropy of some classes of sets with differentiable boundaries*, J. Approx. Theory, 10 (1974), pp. 227–236.

[17] D. Eppstein, *Dynamic Euclidean minimum spanning trees and extrema of binary functions*, Discrete Comput. Geom., 13 (1995), pp. 111–122.

[18] F. Ergun, S. Kannan, Kumar, S. Ravi, R. Rubinfeld, and M. Viswanathan, *Spot-checkers*, J. Comput. System Sci., 60 (2000), pp. 717–751.

[19] B. Gärtner and E. Welzl, *A simple sampling lemma: Analysis and applications in geometric optimization*, Discrete Comput. Geom., 25 (2001), pp. 569–590.

[20] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[21] S. Har-Peled, *Approximate shortest-path and geodesic diameter on convex polytopes in three dimensions*, Discrete Comput. Geom., 21 (1999), pp. 217–231.

[22] S. Kapoor, *Efficient computation of geodesic shortest paths*, in Proceedings of the 31st Annual Symposium on Theory of Computing, ACM, New York, 1999, pp. 770–779.

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

[24] K. Mehlhorn, S. Naher, T. Schilz, S. Schirra, M. Seel, R. Seidel, and C. Uhrig, *Checking geometric programs or verification of geometric structures*, Comput. Geom., 12 (1999), pp. 85–103.

[25] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, *The discrete geodesic problem*,

SIAM J. Comput., 16 (1987), pp. 647–668.

[26] J. S. B. Mitchell, *An algorithmic approach to some problems in terrain navigation*, in Autonomous Mobile Robots: Perception, Mapping and Navigation, IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 408–427.

[27] E. P. Mücke, I. Saias, and B. Zhu, *Fast randomized point location without preprocessing in two and three-dimensional Delaunay triangulations*, Comput. Geom., 12 (1999), pp. 63–83.

[28] K. Mulmuley, *Output sensitive and dynamic constructions of higher order Voronoi diagrams and levels in arrangements*, J. Comput. System Sci., 47 (1993), pp. 437–458.

[29] A. V. Pogorelov, *Extrinsic geometry of convex surfaces*, Transl. Math. Monogr. 35, AMS, Providence, RI, 1973.

[30] D. Ron, *Property testing*, in Handbook on Randomization, Vol. II, S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 597–649.

[31] R. Seidel, *Small-dimensional linear programming and convex hulls made easy*, Discrete Comput. Geom., 6 (1991), pp. 423–434.

[32] M. Sharir and A. Schorr, *On shortest paths in polyhedral spaces*, SIAM J. Comput., 15 (1985), pp. 193–215.

[33] A. C. Yao, *Probabilistic computations: Towards a unified measure of complexity*, in Proceedings of the 18th Annual Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1977, pp. 222–227.