On Approximate Range Counting and Depth*

Peyman Afshani and Timothy M. Chan
School of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{ pafshani, tmchan }@uwaterloo.ca

December 16, 2007

Abstract

We improve the previous results by Aronov and Har-Peled (SODA'05) and Kaplan and Sharir (SODA'06) and present a randomized data structure of $O(n)$ expected size which can answer 3D *approximate* halfspace range counting queries in $O(\log \frac{n}{k})$ expected time, where k is the actual value of the count. This is the first optimal method for the problem in the standard decision tree model; moreover, unlike previous methods, the new method is Las Vegas instead of Monte Carlo. In addition, we describe new results for several related problems, including approximate Tukey depth queries in 3D, approximate regression depth queries in 2D, and approximate linear programming with violations in low dimensions.

1 Introduction

Halfspace range counting arguably ranks as one of the all-time classic problems in computational geometry: how fast can one count the number of points in a query halfspace q if one is allowed to preprocess the point set in a data structure? The existing space/query trade-offs for this problem are not very satisfactory, even in low dimensions. In 3D (the main case of interest in this paper), for data structures with linear space, one is forced to be content with query time near $O(n^{2/3})$ [21] or, in terms of the actual count k, near $O(k^{2/3})$ [9]. On the other hand, logarithmic query time is attained only by data structures with near-cubic space. This is in contrast with the related problem of halfspace range *reporting* (finding all points inside q), which admits efficient algorithms in low dimension: in 2D, a data structure of Chazelle et al. [11] uses linear space and has $O(\log n + k)$ query time; and in 3D, a data structure of Chan [8] (see also Ramos [25]) uses $O(n \log\log n)$ space and has $O(\log n + k)$ query time. (See [1] for further background.)

Since it is generally believed that one cannot beat the query time of $\Omega(n^{2/3})$ with linear space for counting in 3D, researchers have turned to the *approximate* version of the problem. Here, for any fixed constant ε, if the actual count is k, any answer between $(1-\varepsilon)k$ and $(1+\varepsilon)k$ is considered correct. This 3D approximate halfspace range counting problem is the main subject of the paper.

Note that approximation is with respect to the count and not with respect to proximity—the latter option was, for example, investigated by Arya and Mount [3], where ranges are treated as "fuzzy" objects, and points too close to the boundary can either be counted or ignored. These two forms of approximation are vastly different in terms of the techniques involved. Our interest in approximating counts are motivated by applications in computational statistics (see below).

Variants of approximate halfspace range counting were actually considered early on, for example, in a 1986 paper by Edelsbrunner and Welzl [15], who studied the 2D problem with *additive* instead of relative

*A preliminary version of this paper appeared in *Proc. 23rd Sympos. Comput. Geom.*, pages 337–343, 2007. Work of the second author was supported by NSERC.

error (and called the problem "halfplanar range estimation"). If an additive error of εn is tolerable, then the problem can be solved with constant space and query time in any fixed dimension by simply working with a sample (so-called *ε-approximation*) of constant size [17, 23]. In particular, when the count k is close to n, we can get low relative error easily. So, the main challenge is in getting low relative error when k is small. In particular, for $k = 0$, we do not tolerate any error; thus, the problem should be at least as hard as range *emptiness* (deciding whether h contains any point). The existence of efficient halfspace range emptiness data structures in 2D and 3D (with $O(n)$ space and $O(\log n)$ query time [14, 24]) suggests that efficient approximate halfspace range counting structures might be possible in the same dimensions.

Indeed, that is the case, as was shown in two recent SODA papers. (Both papers are of particular relevance to us here, as some of the techniques used are related to ours.) The first, by Aronov and Har-Peled [2], described a black-box reduction from approximate halfspace range counting to range emptiness, with polylogarithmic increase in space and query time. In 3D, their resulting data structure needs $O(n \log n)$ space and $O(\log^2 n \log\log n)$ query time and gives a correct approximate answer with high probability, but is Monte Carlo in the sense that the query algorithm does not know if the returned answer is a correct approximation or not. The second paper, by Kaplan and Sharir [18], improved the query time to $O(\log^2 n)$ with the same $O(n \log n)$ space bound, by using a different strategy that combined an approximation technique of Cohen [13] with a new combinatorial lemma about overlaying lower envelopes over all prefixes of a randomly permuted sequence of planes. This query algorithm is also Monte Carlo. Subsequently, in an updated version of the first paper [2], Aronov and Har-Peled showed that the same improved query time of $O(\log^2 n)$ can be obtained directly by their original method, making the overlay lemma unnecessary. Both Aronov and Har-Peled's and Kaplan and Sharir's approaches are suboptimal by a logarithmic factor in space as well as time, and the question of obtaining an $O(n)$-size structure with $O(\log n)$ query time in 3D is left open—this situation is somewhat unsettling, considering the fundamental nature and simplicity of the problem, and the desirability of linear space and logarithmic time in practice.[1]

Our main result. We resolve the remaining open problem in this paper, by presenting a data structure using $O(n)$ expected space which can answer approximate halfspace range counting queries in 3D in $O(\log n)$ expected time or, better still, in $O(\log \frac{n}{k})$ expected time. This is optimal in terms of n and k, because a matching $\Omega(\log \frac{n}{k})$ lower bound in the algebraic decision tree model is not hard to show, even in 2D.[2]

The expected preprocessing time for our data structure is $O(n \log n)$, which is also optimal in terms of n. Moreover, unlike the previous algorithms, our query algorithm is *Las Vegas*, meaning that it always produces a correct approximate answer. Our result is obtained by bringing in *shallow cuttings* (see Section 2.2) and, at the same time, applying Kaplan and Sharir's overlay technique in a new way (see Section 2.3), thus reclaiming the usefulness of the overlay lemma. Along the way, we realize that beating $O(\log^2 n)$ query time is actually quite easy—in fact, one idea that can lead to a slightly suboptimal $O(\log n \log\log n)$ bound (see Section 2.1) is already contained in Aronov and Har-Peled's paper [2].

Other related results. We can apply some of our ideas to obtain new results on approximating the depth of a query point with respect to two definitions of depth that have been popularly studied at the intersection of robust statistics and computational geometry. The *Tukey depth* (also called halfspace depth) of a query point q with respect to a point set P is defined as the minimum number of points of P in any halfspace that contains q. The *regression depth* of a query line q with respect to a 2D point set P is defined as the minimum number of points intersected by q in any continuous motion which turns q into a vertical line [26]. Previous work in computational geometry has mostly focused on estimating the maximum depth rather than building data structures to estimate the depth of a query point/line. Points of maximum Tukey depth and lines of maximum regression depth in 2D can be computed in optimal $O(n \log n)$ expected time [10, 19], but it appears difficult to find data structures with nontrivial worst-case performance that can compute the

[1] After this work was submitted, we learn of a third paper by Har-Peled and Sharir [**?**], which contains among other results an improvement of the query time to $O(\log n \log\log n)$ but with a larger space bound of $O(n \log^{O(1)} n)$ for the 3D problem; as we will see, our result is better. A version of their paper, with Aronov, also has results in higher fixed dimensions.

[2] Proof: Consider $\frac{n}{2k}$ points in convex position, each duplicated $2k$ times. Deciding whether a halfplane has approximately less than k points is the same as answering emptiness queries in an input of size $\frac{n}{2k}$. (It is possible to modify the construction for nondegenerate input.)

exact depth of an arbitrary query point/line. The maximum depth in either definition is $\Theta(n)$ for all point sets, so an approximate maximum depth can be found easily by again working with a sample of constant size. The challenge is in approximating the depth of a query point/line, with low relative error, when the point/line is at small depth.

We show how one can approximate the regression depth of any query line in 2D with an $O(n)$-space[3] data structure in $O(\log n)$ time (see Section 3.1). We show how one can approximate the Tukey depth of a query point in 3D with an $O(n)$-space data structure in $O(\log n \log\log n)$ time; in 2D the query time can be improved to $O(\log n)$ (see Section 3.2).

Another related problem of fundamental importance is *linear programming (LP) with violations*. In one version, we are given n halfspaces and we want to find the smallest number k of halfspaces to delete so that the remaining halfspaces have a nonempty intersection. (Equivalently, we want the minimum depth, in another sense of the word, in an arrangement of halfspaces.) This problem was studied by Matoušek [22] and Chan [7]. In 2D and 3D, the current best running time, by Chan, is $O(n \log k + k^2 \log n)$ and $O(n \log k + k^{11/4} n^{1/4} \log^{O(1)} n)$ respectively; in higher dimensions d, the time bound is slightly less than $O(nk^{d+1})$. Aronov and Har-Peled [2] showed that a $(1+\varepsilon)$-factor approximation of the minimum k can be found by significantly faster Monte Carlo algorithms with $O(n \log\log n)$ running time in 2D and $O(n \log^{d+1} n)$ running time in any constant dimension d. We observe that with appropriate data structures, their method actually runs in $O(n \log\log n)$ time for any fixed dimension and, in fact, $O(n)$ time if $k \gg \log n \log\log n$ (see Section 3.3). We also show how to obtain an efficient $O(n \log n)$ *Las Vegas* algorithm in 3D. Similar ideas also lead to an $O(n \log n)$ Las Vegas algorithm for approximating the maximum depth in an arrangement of disks in 2D (for this problem, Aronov and Har-Peled gave an $O(n \log n)$ Monte Carlo algorithm).

Preliminaries. For the sake of ease of description we introduce the following phrases. Throughout this paper, we say that an event X happens *with high probability (w.h.p.)* if $\Pr[X] \geq 1 - 1/n^{c_0}$ for a fixed large constant c_0. We say a number x *is ε-approximately* y iff $y(1-\varepsilon) \leq x \leq y(1-\varepsilon)^{-1}$; similarly, x *is ε-approximately less (resp. greater) than* y iff $x \leq y(1-\varepsilon)^{-1}$ (resp. $x \geq y(1-\varepsilon)$). For the sake of simplicity we will not work out the precise dependences of the time/space bounds on ε, since we have not tried to optimize such dependences; we use the O_ε notation to hide factors in terms of ε, which are in all cases $1/\varepsilon^{O(1)}$ or smaller.

2 Approximate Halfspace Range Counting Queries in 3D

In this section, we describe several methods for approximate halfspace range counting in 3D, culminating in the final $O(n)$-space, $O(\log \frac{n}{k})$-time result.

2.1 Method 1: Reduce to small counts by sampling

The first method is based on a simple random sampling idea, but already beats the previous results. This method will not be used in the final solution (so this subsection may be skipped if the reader cares only about halfspace range counting), but the idea will be useful later in solving other related problems.

The idea is based on an easy application of the Chernoff bound:

Observation 2.1. *Let c_ε be a sufficiently large constant depending on ε and k be a fixed parameter such that $k \geq e_\varepsilon \log n$. Let P be a set of n elements, and let R be a random sample where each element of P is included with probability $p = \frac{c_\varepsilon \log n}{k} < 1$. Given any subset $S \subseteq P$ of size k^*, we have $\left| k^* - \frac{1}{p}|S \cap R| \right| \leq \varepsilon \max\{k, k^*\}$ w.h.p.*

Proof. Let $X_i = 1$ if the i-th element of S is in R, and 0 otherwise; these are independent, identically distributed, random variables. Let $X = |S \cap R| = \sum_{i=1}^{k^*} X_i$. According to the Chernoff inequality,

$$\Pr[|X - E(X)| \geq b\sigma(X)] \;\leq\; 2e^{-b^2/4},$$

[3]This improves over the $O(n \log\log n)$ space bound claimed in the preliminary version of this paper.

where $E(X) = pk^*$ and $\sigma(X) = \sqrt{(p-p^2)k^*}$. Substituting $b = \varepsilon\sqrt{p\max\{k,k^*\}}$, we get

$$\Pr[||S \cap R| - pk^*| \geq \varepsilon p \max\{k,k^*\}] \leq 2e^{-\varepsilon^2 pk/4} \leq n^{-c_0}$$

if c_ε is a sufficiently large. $\square$

The above observation suggests a simple but general reduction of approximate range counting to a special case of range counting in which the answer is small (bounded by $O(\log n)$).

Theorem 2.2. *Given any point set of size n and a "threshold" parameter ℓ, suppose there is a data structure that can decide whether the number of points in a query range is at most ℓ, and if so, return this number. Let $P_{\mathrm{small}}(n)$, $S_{\mathrm{small}}(n)$, and $Q_{\mathrm{small}}(n,\ell)$ be the (expected) preprocessing time, space, and query time of this data structure (assuming these functions are well-behaved).*

Then with $O_\varepsilon(P_{\mathrm{small}}(n))$ preprocessing time, $O_\varepsilon(S_{\mathrm{small}}(n))$ space and $O_\varepsilon(Q_{\mathrm{small}}(n,\log n)\log\log n)$ expected query time one can build a data structure that can approximate the number of points in a query range.

The algorithm is correct w.h.p. for any fixed query range.

Proof. For each i, we take a random sample R_i of the given point set P with sampling probability $p_i = \frac{c_\varepsilon \log n}{k_i}$ in which $k_i = \lfloor (1+\varepsilon)^i \rfloor$ for $i = \lceil \log_{1+\varepsilon}(c_\varepsilon \log n)\rceil, \ldots, \lceil \log_{1+\varepsilon} n\rceil$. We then build a small-count data structure for every sample R_i, with threshold $\ell = c_\varepsilon \log n$. Since the expected size of R_i geometrically decreases as i increases, the total preprocessing time and space is asymptotically unchanged.

To approximate the (unknown) number k^* of points inside a query halfspace h, we do an approximate binary search. By Observation 2.1, for a given k_i, we can determine whether k^* is $O(\varepsilon)$-approximately less than k_i or $O(\varepsilon)$-approximately greater than k_i, by testing whether $\frac{1}{p_i}|h \cap R_i| \leq k_i$ or not, i.e., whether $|h \cap R_i| \leq c_\varepsilon \log n$. This can be done by using the small-count data structure for R_i. After $O(\log\log_{1+\varepsilon} n)$ iterations of binary search on the k_i's, we arrive at a value that $O(\varepsilon)$-approximates k^* w.h.p. (We can readjust ε by a constant factor if necessary.) The query time is $O(Q_{\mathrm{small}}(n, c_\varepsilon \log n)\log\log_{1+\varepsilon} n)$.

One special case remains: what if k^* is less than all the k_i's, i.e., $k^* \leq c_\varepsilon \log n$? In this case, we can directly answer the query using a small-count data structure for P in $Q_{\mathrm{small}}(n, c_\varepsilon \log n)$ time. $\square$

Chan's 3D halfspace reporting data structure [8] achieves an expected query time of $Q_{\mathrm{small}}(n,\ell) = O(\log n + \ell)$ with expected preprocessing time $P_{\mathrm{small}}(n) = O(n \log n)$. The original version of the data structure requires $O(n \log n)$ space, which was improved to $O(n \log\log n)$ [8, 25]. However, in the threshold version where one common value of ℓ is used throughout, the data structure can be made to use $S_{\mathrm{small}}(n) = O(n)$ space (since one layer of shallow cuttings is sufficient, as one can see from [8] and Ramos' paper [25]). We thus obtain a data structure for 3D approximate halfspace range counting with $O_\varepsilon(n \log n)$ expected preprocessing time, $O_\varepsilon(n)$ expected space, and $O_\varepsilon(Q_{\mathrm{small}}(n,\log n)\log\log n) = O_\varepsilon(\log n \log\log n)$ expected query time. This is almost optimal, though Monte Carlo.

Note that the algorithm is correct for all queries w.h.p. (after readjusting c_0), since the number of combinatorially "different" halfspaces is polynomial.

Remarks. Ironically, essentially the same Chernoff-bound observation was used before in Aronov and Har-Peled's paper [2], not for halfspace range counting but for offline problems like linear programming with violations. One reason that Aronov and Har-Peled obtained a worse result for range counting is perhaps their insistence on reducing approximate counting to *emptiness*, although to be fair, their main focus was not in optimizing logarithmic factors.

2.2 Method 2: Approximate levels by shallow cuttings

To get an optimal query time, we change tactics entirely. Ultimately we will need a nontrivial combination of several ideas, but to start we introduce one idea that, in itself, leads to another suboptimal method with the same time bound as Method 1. The main advantage of this new method is that it is Las Vegas.

We switch to dual space, where we want to preprocess a set H of n planes in $\mathbb{R}^3$ in a data structure that can answer the following queries: given any query point q, approximate the number k^* of planes below q. This value is also called the *level* of q. Call the locus of all points of level at most k the $(\leq k)$-*level*. Call the boundary of this locus the k-*level*.

Our idea is to use approximate levels rather than exact levels. In 3D, the best upper bound on the complexity of the exact k-level currently is $O(nk^{3/2})$ [27]. However, the total complexity of the k'-level for all $k' = 0, \ldots, k$ is $O(nk^2)$ [12], which means that the average complexity of a k'-level with $(1-\varepsilon)k \leq k' \leq (1+\varepsilon)k$ is $O_\varepsilon(nk)$. This is still too large for our purposes. We show that a form of approximate k-level exists with complexity $O_\varepsilon(\frac{n}{k})$ only, which surprisingly is sublinear for nonconstant k.

The main tool is Matoušek's shallow cutting lemma, one version of which is stated below. An ε-*cutting* of a set H of n planes is a collection of nonoverlapping cells (tetrahedra) such that each cell intersects at most εn planes of H. The conflict list of a cell refers to the list of planes intersecting the cell. We say that the cutting *covers* a region if the union of the simplices contains the region.

Lemma 2.3. *For any set of n planes in $\mathbb{R}^3$ and a parameter k, there exists an $O(\frac{k}{n})$-cutting of size $O(\frac{n}{k})$ that covers the $(\leq k)$-level. The cells in the cutting are all vertical prisms unbounded from below.*

Furthermore, we can construct these cuttings for all k of the form $\lfloor (1+\varepsilon)^i \rfloor$ simultaneously in $O_\varepsilon(n \log n)$ expected time; we can also construct the conflict lists of all cells in the same time.

Proof. The first part is due to Matoušek [20]. The construction time for the cuttings follows from an algorithm by Ramos [25]. That vertical prisms suffice was observed by Chan [7, Lemma 3.1]. The conversion to vertical prisms, as well as the computation of the conflict lists, can be carried out in additional $O(\frac{n}{k}(\log n + k))$ expected time for each k, after $O(n \log n)$ expected preprocessing time (essentially, the computation involves $O(\frac{n}{k})$ halfspace range reporting queries, each requiring $O(\log n + k)$ time [8]). Summing $O(\frac{n}{k}(\log n + k))$ over all k still gives $O_\varepsilon(n \log n)$. □

We formally define an ε-*approximate* $(\leq k)$-*level* to be a region that contains the $(\leq (1-\varepsilon)k)$-level and is contained in the $(\leq (1+\varepsilon)k)$-level. (For our purposes, we do not require the region to be a terrain.) Using the shallow cutting lemma, we get:

Lemma 2.4. *For any set of n planes in $\mathbb{R}^3$ and a parameter k, there exists an $O(\varepsilon)$-approximate $(\leq k)$-level of size $O_\varepsilon(\frac{n}{k})$.*

Furthermore, we can construct such approximate levels for all k of the form $\lfloor (1+\varepsilon)^i \rfloor$ simultaneously in $O_\varepsilon(n \log n)$ expected time; in the same time, we can also build a linear-size data structure that can decide whether a query point lies inside such an approximate $(\leq k)$-level in $O(\log \frac{n}{k})$ time.

Proof. Construct an $O(\frac{k}{n})$-cutting of size $O(\frac{n}{k})$, covering the $(\leq k)$-level by Lemma 2.3. For each vertical prism Δ, construct an ε-cutting (covering Δ) of the $O(k)$ planes in the conflict list. As is well known, for a constant ε, there exists an ε-cutting (covering $\mathbb{R}^3$) of constant size [14, 23] (more precisely, of size $O(\varepsilon^{-3})$) and the cutting can be constructed in time linear in the number of planes, i.e., $O(k)$ time. (Constant-size cuttings can in fact be constructed by "elementary" methods, such as prune-and-search.) Overall all $O(\frac{n}{k})$ prisms for a given k, this process produces $O_\varepsilon(\frac{n}{k})$ subcells and takes $O_\varepsilon(n)$ time.

For each subcell $\delta \subset \Delta$ of the ε-cutting, we compute the level ℓ_δ of some arbitrary point in δ, and if $\ell_\delta \leq k$, we include δ in the approximate $(\leq k)$-level. This construction satisfies the desired property, because each subcell δ intersects $O(\varepsilon k)$ planes and so the levels of any two points in the subcell differ by at most $O(\varepsilon k)$. We can compute ℓ_δ by performing a halfspace range reporting query back in primal space in $O(\log n + k)$ expected time each [8], so the total time for this step is $O(\frac{n}{k}(\log n + k))$, which sums to $O_\varepsilon(n \log n)$.

For a given query point q, we can find the vertical prism Δ in the $O(\frac{k}{n})$-cutting containing q in $O(\log \frac{n}{k})$ time by planar point location [14, 24] on the xy-projection of the prisms. Afterwards, we can find the subcell $\delta \subset \Delta$ containing q in constant time and see if δ was included in the approximate $(\leq k)$-level. □

The above lemma immediately suggests a data structure for our problem:

Theorem 2.5. *With $O_\varepsilon(n \log n)$ expected preprocessing time one can build a data structure of size $O_\varepsilon(n)$ which can answer approximate 3D halfspace range counting queries in $O_\varepsilon(\log n \log\log n)$ worst-case time. The query algorithm is always correct.*

Proof. Let $k_i = \lfloor (1+\varepsilon)^i \rfloor$ for $i = 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil$, and construct an approximate ($\leq k_i$)-level for each i. The expected preprocessing time is $O_\varepsilon(n \log n)$. The total space is given by a geometric series $O(\sum_i \frac{n}{k_i}) = O_\varepsilon(n)$.

To approximate the number k^* of points below the query point q, we do an approximate binary search. For a given k_i, we can determine whether k^* is $O(\varepsilon)$-approximately less than k_i or $O(\varepsilon)$-approximately greater than k_i, by testing whether q lies inside the approximate ($\leq k_i$)-level or not, in $O(\log \frac{n}{k_i})$ time. After $O(\log\log_{1+\varepsilon} n)$ iterations of binary search on the k_i's, we arrive at a value that is $O(\varepsilon)$-approximately k^*. The query time is $O(\log n \log\log_{1+\varepsilon} n)$. □

Remark. The above data structure uses a hierarchy of shallow cuttings and is similar in spirit to Chan's data structure for halfspace range reporting [8], which uses a hierarchy of lower envelopes of random samples. Lower envelopes of samples share some similar characteristics as shallow cuttings and are more practical for implementation but, without additional ideas, do not seem to yield Las Vegas results as good as the above data structure. In the next method, though, we will employ lower envelopes of random subsets, but in conjunction with our shallow-cutting-based method.

2.3 Method 3: Combine with randomized incremental construction

The last ingredient we need is a technique by Kaplan and Sharir:

Lemma 2.6. *Let $h_1, \ldots, h_n$ be a random permutation of n given planes in $\mathbb{R}^3$. With $O(n \log n)$ expected preprocessing time one can build a data structure of expected size $O(n \log n)$ so that given a query point q, one can find the smallest index j such that h_j lies below q in $O(\log n)$ expected time. In fact, the expected query bound can be reduced to $O(\log j)$.*

Proof. The first part is due to Kaplan and Sharir [18], and is derived from their combinatorial lemma stating that the overlay of all the lower envelopes encountered during a randomized incremental construction has expected complexity $O(n \log n)$.

For the improvement to $O(\log j)$, let $j_i = 2^{2^i}$ for $i = 1, 2, \ldots, \lceil \log\log n \rceil$ and build the above data structure for the prefix $h_1, \ldots, h_{j_i}$, which is itself a random permutation, for each i. The expected preprocessing time and space remain asymptotically unchanged. To answer a query, we query the prefix $h_1, \ldots, h_{j_i}$ for $i = 1$, $i = 2$, and so on, until an answer is found. The total expected query time is $O(\sum_{j_{i-1} \leq j} \log j_i) = O(\log j)$. □

The usefulness of the above lemma is explained by the following observation:

Observation 2.7. *Let $h_1, \ldots, h_n$ be a random permutation of a set H. Given any subset $S \subseteq H$ of size k^*, let j be the smallest index with $h_j \in S$. Let $k = \frac{n}{j}$. Then the probability that $k^* < k/b$ or $k^* > bk$ is $O(1/b)$.*

Proof. The event $k^* < k/b$ implies that $j < \frac{n}{bk^*}$ and so at least one of $h_1, \ldots, h_{n/(bk^*)}$ is in S; this happens with probability at most $\frac{n}{bk^*} \cdot \frac{k^*}{n} = 1/b$. On the other hand, $k^* > bk$ implies that $j > bn/k^*$ and so $h_1, \ldots, h_{bn/k^*}$ are all not in S; this happens with probability at most $(1 - k^*/n)^{bn/k^*} = 1/e^{\Omega(b)}$. □

We now present our final method. The key idea is to use Kaplan and Sharir's lemma to obtain an initial estimate $k = \frac{n}{j}$ which approximates the unknown count k^* well with good probability ("well" and "good" in the sense of the above observation). With the availability of this initial estimate, we can speed up Method 2: namely, we can replace the approximate binary search (which is the cause of the extra $\log\log n$ factor) with a simple linear search. In the analysis, we bound the overall expected query time by a geometric series.

Theorem 2.8. *With $O_\varepsilon(n \log n)$ expected preprocessing time, one can build a data structure of expected size $O_\varepsilon(n)$ which can answer approximate 3D halfspace range counting queries in $O_\varepsilon(\log \frac{n}{k^*})$ expected time for any fixed query halfspace. Here k^* is the actual value of the count and the query algorithm is always correct.*

Proof. Our data structure consists of the data structure from Method 2 (Theorem 2.5), augmented with the data structure in Lemma 2.6 applied to a prefix $h_1, \ldots, h_{n/\log n}$ of a random permutation of size $n/\log n$. Since the number of planes is $O(n/\log n)$, the expected space is $O(n)$.

To approximate the number k^* of planes below a query point q, we first compute the smallest index j such that h_j lies below q in $O(\log j)$ time. Let $k = \frac{n}{j}$ and suppose $k_s \le k < k_{s+1}$. We apply a linear search starting at k_s. Recall that we can determine whether k^* is $O(\varepsilon)$-approximately less than $k_{s\pm i}$ or $O(\varepsilon)$-approximately greater than $k_{s\pm i}$, in $O(\log \frac{n}{k_{s\pm i}})$ time by querying an approximate level. If k^* is $O(\varepsilon)$-approximately less than k_s, we repeatedly $O(\varepsilon)$-approximately compare k^* with $k_{s-1}, k_{s-2}, \ldots$; otherwise, we repeatedly $O(\varepsilon)$-approximately compare k^* with $k_{s+1}, k_{s+2}, \ldots$ With $O(i)$ iterations of the search, we eventually arrive at a value $k_{s\pm i}$ that is $O(\varepsilon)$-approximately k^*.

The probability that k^* is $O(\varepsilon)$-approximately $k_{s\pm i} \approx (1+\varepsilon)^{\pm i} k$ is at most $O(1/(1+\varepsilon)^i)$ by Observation 2.7. Thus, the total expected query time is upper-bounded by

$$\sum_{i=1}^{\infty} \frac{1}{(1+\varepsilon)^i} \cdot O\left(i \log \frac{n(1+\varepsilon)^i}{k^*}\right) \;=\; O_\varepsilon\left(\log \frac{n}{k^*}\right).$$

One special case remains: what if j exceeds $n/\log n$, i.e., $k < \log n$? Here, we cannot compute the actual value of j or k. Instead we start the search with $k_s \le \log n < k_{s+1}$. If k^* is $O(\varepsilon)$-approximately less than $k_s \le \log n$, we can directly answer the query using a small-count data structure in $Q_{\text{small}}(n, \log n) = O(\log n) = O(\log \frac{n}{k^*})$ time. Otherwise, the probability that k^* is $O(\varepsilon)$-approximately $k_{s+i} \approx (1+\varepsilon)^i \log n$ is even smaller by Observation 2.7 since $k < \log n$, and the same query bound holds. □

3 Related Problems

We now apply similar techniques to solve other related problems.

3.1 Approximate regression depth queries in 2D

The problem of computing the regression depth of a query line in 2D reduces to the following in dual space: Given a set H of n lines in $\mathbb{R}^2$, preprocess them in a data structure so that given any query point q, we can find the minimum number k^* of lines intersected by a ray over all rays originating from q. Following [4], call k^* the *undirected depth* of q. Call the locus of all points of undirected depth at most k the *$(\le k)$-envelope*. Call the boundary of this locus the *k-envelope*. In measuring the complexity of a k-envelope or a chain, we will include all vertices in the arrangement that lie on the chain, including those making angles of π (in other words, every line contributes as many vertices as its number of intersections with the boundary).

We can adapt Method 2 to solve the problem. We first need an analog of Lemma 2.4. Define an *ε-approximate $(\le k)$-envelope* to be a region that contains the $(\le (1-\varepsilon)k)$-envelope and is contained in the $(\le (1+\varepsilon)k)$-envelope. In the preliminary version of this paper, we prove the existence of an approximate $(\le k)$-envelope of size $O(\min\{n, \frac{n}{k}\log n\})$. Here, we present a different proof of a slightly improved bound of $O(\frac{n}{k}\alpha(k))$, where $\alpha(\cdot)$ is the inverse Ackermann function.

Lemma 3.1. *For any set H of n lines in $\mathbb{R}^2$, the total complexity of the k'-envelope over all $k' = 0, \ldots, k$ is $O(nk)$.*

Proof. First observe that in 2D, the 0-envelope consists of all unbounded cells in the arrangement and has linear complexity by the zone theorem [14], since the unbounded cells intersect the line $x = -M$ or $x = M$ for a sufficiently large M.

We apply Clarkson and Shor's technique [12], by picking a random sample in which each line is chosen with probability $\frac{1}{k}$. The lemma follows from the fact that a vertex v of the arrangement which lies in the $(\le k)$-envelope of H has $\Omega(\frac{1}{k^2})$ chance of surviving in the sample and being a 0-envelope vertex. □

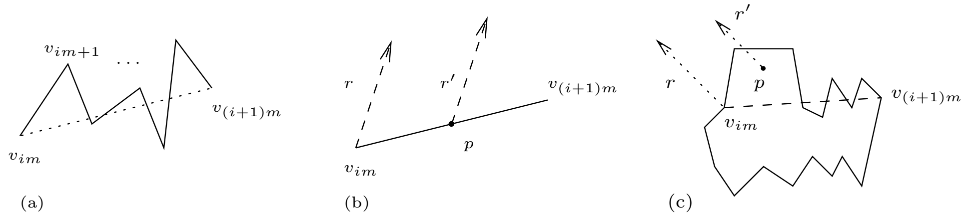

Figure 1: (a) An edge of a simplified polygon (in dotted line). (b) Proof of Lemma 3.2(i). (c) Proof of Lemma 3.2(ii).

In 2D, one can obtain an approximate level of size $O(\frac{n}{k})$ by taking an exact level of size $O(n)$ and applying a "simplification" process (e.g., as in [15]). We show that approximate envelopes can be constructed in a similar fashion. The modification is not trivial, as k-envelopes have a more complicated geometry than k-levels. In particular, the k-envelope may consist of multiple polygons; all points inside these polygons have undirected depth at least k and all points outside have undirected depth at most k.

Given a polygon $A = \langle v_0, v_1, \ldots, v_t \rangle$, we define its *m-simplification* as the polygon $A' = \langle v_0, v_m, v_{2m}, \ldots, v_{m\lfloor \frac{t}{m} \rfloor}, v_t \rangle$ (see Figure 1(a)). Note that even if A is a simple polygon, the simplified polygon A' may self-intersect. We say that a point p is *outside* a nonsimple polygon C if p is in the outer connected component of $\mathbb{R}^2 - C$ (i.e., the component containing infinity); otherwise, p is *inside* C. Similarly, p is outside a collection C of polygons (possibly with holes) if p is in the outer component of $\mathbb{R}^2 - \bigcup_{A \in C} A$. The following lemma encompasses all the properties of simplified polygons that we need:

Lemma 3.2. *Let C be the collection of polygons defining the k-envelope and C' be the collection of the m-simplifications of these polygons. Then* (i) *all points inside C' have undirected depth at least $k-m$;* (ii) *all points outside C' have undirected depth at most $k+m$; and* (iii) *C' has at most $O(|C'|m)$ crossings.*

Proof. (i) Consider any ray r' from an arbitrary point inside C' which intersects C', say, at a point p on the segment $v_{im}v_{(i+1)m}$. Draw a ray r parallel to r' from v_{im} (Figure 1(b)). Then r hits at least k lines since v_{im} has undirected depth k. Any line which crosses only one of these two rays intersects the segment $v_{im}v_{(i+1)m}$ and thus creates a vertex on the chain $\langle v_{im}, v_{im+1}, \ldots, v_{(i+1)m} \rangle$, but there are only m such vertices. So, r' hits at least $k - m$ lines.

(ii) Let p be outside C'. Then p must be outside the simplified polygon A' for some $A = \langle v_0, v_1, \ldots, v_t \rangle$ in C. If p is outside A, then p has undirected depth at most k. We may assume that p is inside A.

Since p is outside A', it is possible to connect p to infinity through a curve c not crossing A'. Since p is inside A, the number of intersections between c and A must be odd. This implies that for some v_{im}, the number of intersections between c and the chain $A_i = \langle v_{im}, v_{im+1}, \cdots, v_{(i+1)m} \rangle$ is odd.

Observe that p is inside the shape formed by adding the segment $v_{im}v_{(i+1)m}$ of A' to the chain A_i, since the number of intersections of c with this boundary is odd. Thus, we have a situation similar to Figure 1(c). Consider a ray r from v_{im} which crosses k lines, and draw a ray r' parallel to r from p. Then any line which crosses r' but not r must also cross the chain A_i, but there can be at most m such lines. So, p has undirected depth at most $k + m$.

(iii) Suppose two segments $v_{im}v_{(i+1)m}$ and $w_{jm}w_{(j+1)m}$ of C' cross. Since the original k-envelope C does not have crossings, one of two possibilities must hold: the chain $A_i = \langle v_{im}, v_{im+1}, \ldots, v_{(i+1)m} \rangle$ intersects the segment $w_{jm}w_{(j+1)m}$, in which case we give $w_{jm}w_{(j+1)m}$ one charge; or the chain $B_j = \langle w_{jm}, w_{jm+1}, \ldots, w_{(j+1)m} \rangle$ intersects the segment $v_{im}v_{(i+1)m}$, in which case we give $v_{im}v_{(i+1)m}$ one charge.

Any line which intersects the segment $v_{im}v_{(i+1)m}$ must create a vertex on the chain A_i, but there are only m such vertices. Thus, each of the $|C'|$ segments receives $O(m)$ charges. $\square$

With one additional combinatorial fact stated below, we can derive our lemma on approximate envelopes.

Lemma 3.3. *Given an arrangement of N line segments in $\mathbb{R}^2$ with X intersections, the outer face has complexity $O(N\alpha(\lceil X/N \rceil))$.*

Proof. By a standard result [16], the outer face has complexity $O(N\alpha(N))$. To obtain an X-sensitive bound, we use known results on *intersection-sensitive cuttings* (e.g., [**?**]): there exists a partition of $\mathbb{R}^2$ into $O(r + X(r/N)^2)$ triangles, such that each triangle intersects $O(N/r)$ segments. For each triangle Δ, let S_Δ be the segments clipped to Δ. Since each vertex in Δ of the outer face of the overall arrangement must be on the outer face of S_Δ, the complexity of the overall outer face is at most $O\left(\sum_\Delta |S_\Delta|\alpha(|S_\Delta|)\right) = O((r + X(r/N)^2) \cdot (N/r)\alpha(N/r))$. Setting $r = \min\{N^2/X, N\}$ yields the $O(N\alpha(\lceil X/N \rceil))$ bound. ☐

Lemma 3.4. *For any set of n lines in $\mathbb{R}^2$ and a parameter k, there exists an $O(\varepsilon)$-approximate $(\leq k)$-envelope of size $O_\varepsilon(\frac{n}{k}\alpha(k))$.*

Proof. According to Lemma 3.1, the average complexity of a k'-envelope for a random k' between $(1-\varepsilon)k$ and $(1+\varepsilon)k$ is $O_\varepsilon(n)$. Let C be such a k'-envelope. We return the outer face of the m-simplifications of the polygons in C, with the parameter $m = \varepsilon k$. By Lemma 3.2(i,ii), the resulting polygon is an $O(\varepsilon)$-approximate $(\leq k)$-envelope.

Note that any polygon in C with complexity less than m can be simplified to the empty polygon and be discarded. The total number of vertices in the simplified polygons is $N = O_\varepsilon(\frac{n}{k})$ and the number of crossings is $X = O_\varepsilon(Nk)$ by Lemma 3.2(iii). By Lemma 3.3, the complexity of the outer face is $O_\varepsilon(\frac{n}{k}\alpha(k))$. ☐

We can now use the above lemma to prove the following theorem.

Theorem 3.5. *One can build a data structure that uses $O_\varepsilon(n)$ space and can answer approximate regression depth queries in 2D in $O_\varepsilon(\log n)$ worst-case time.*

Proof. Let $k_i = \lfloor (1+\varepsilon)^i \rfloor$ for $i = 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil$, and construct an $(\varepsilon/3)$-approximate $(\leq k_i)$-envelope E_i for each i by Lemma 3.4. The total size of the E_i's is $O_\varepsilon(\sum_i \frac{n}{k_i}\alpha(k_i)) = O_\varepsilon(n)$. Finally, we store all these approximate envelopes in a point location data structure. Note that there are no intersections between the boundaries of the E_i's.

To approximate k^* for a query point q, we return the smallest k_i such that q lies inside E_i. This can be done in $O_\varepsilon(\log n)$ time by a single planar point location query on the combined subdivision formed by the boundaries of the E_i's. ☐

3.2 Approximate Tukey depth queries in 3D

The problem of computing the Tukey depth of a query point in 3D reduces to the following in dual space: Given a set H of n planes in $\mathbb{R}^3$, preprocess them in a data structure so that given any query plane q, we can find the smallest value k^* such that q intersects the $(\leq k^*)$-level in the arrangement of H. (Technically, we need to compute also the smallest value k^{**} such that q intersects the $(\geq n - k^{**})$-level, and return the smaller of the two values, but computing k^{**} is similar.)

We can adapt either Method 1 or Method 2 to solve this problem. The latter, in particular, leads to a Las Vegas result:

Theorem 3.6. *One can preprocess a 3D point set of size n in $O_\varepsilon(n \log n)$ expected time into a data structure of $O_\varepsilon(n)$ size such that the Tukey depth of any query point q can be approximated in $O_\varepsilon(\log n \log\log n)$ worst-case time. The query algorithm is always correct.*

Proof. Let $k_i = \lfloor (1+\varepsilon)^i \rfloor$ for $i = 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil$, and construct an approximate $(\leq k_i)$-level L_i for each i by Lemma 2.4, as in the proof of Theorem 2.5. For each i, we compute the upper hull U_i of the $O(\frac{n}{k_i})$ vertices of L_i. This takes time $O(\sum_i \frac{n}{k_i} \log \frac{n}{k_i}) = O_\varepsilon(n \log n)$.

To approximate k^* for a query plane q, we do an approximate binary search. For a given k_i, we can determine whether k^* is $O(\varepsilon)$-approximately greater than k_i or $O(\varepsilon)$-approximately less than k_i, by testing whether q lies strictly above the upper hull U_i or not, in $O(\log n)$ time (back in primal space, this corresponds testing whether a point lies below a lower envelope of planes, which reduces to planar point location). After $O(\log\log_{1+\varepsilon} n)$ iterations of binary search, we arrive at a value that is $O(\varepsilon)$-approximately k^*. The query time is $O(\log n \log\log_{1+\varepsilon} n)$. ☐

Remarks. For Tukey depth queries in 2D, we can improve the query time to $O_\varepsilon(\log n)$ by the same idea as in the proof of Theorem 3.5 (replacing binary search with a single planar point location query).

3.3 Approximate LP with violations

Lastly, we address the problem of LP with violations: given a set H of n halfspaces in $\mathbb{R}^d$, find a point that violates (i.e., lies outside) the smallest number k_{opt} of halfspaces.

We can apply Method 1 to get the following reduction, which was originally obtained by Aronov and Har-Peled [2]. We include the proof to prepare for a later result:

Theorem 3.7. *Suppose there is an algorithm for LP with violations running in $T_{\text{small}}(n, k_{\text{opt}})$ time (assuming that the function T_{small} is well-behaved). Then we can solve the problem approximately in $O_\varepsilon(T_{\text{small}}(\frac{n}{k_{\text{opt}}}\log n, \log n))$ time. This algorithm is correct w.h.p.*

Proof. We first solve the approximate decision problem: given a threshold k, determine whether k_{opt} is ε-approximately less than k or ε-approximately greater than k.

The solution is simple: take a random sample R with sampling probability $p = \frac{c_\varepsilon \log n}{k}$ and solve the problem for R with threshold $pk = c_\varepsilon \log n$ in $O(T_{\text{small}}(\frac{c_\varepsilon \log n}{k}, c_\varepsilon \log n))$ time. By Observation 2.1, if a fixed point violates ε-approximately less (resp. greater) than pk halfspaces in R, then it violates $O(\varepsilon)$-approximately less (resp. greater) than k halfspaces in H w.h.p. Thus, the answer is correct w.h.p., since the number of combinatorially "different" points is polynomial.

To approximate k_{opt}, we just run the above approximate decision algorithm for $k = n, (1-\varepsilon)n, (1-\varepsilon)^2 n, \ldots$, until k is $O(\varepsilon)$-approximately k_{opt}. The total running time forms a geometric series and increases only by a constant factor. ☐

In d dimensions, Matoušek [22] gave an algorithm for LP with violations with running time $O(nk_{\text{opt}}^{d+1})$. For small $k_{\text{opt}} < n^\alpha$ for a constant $\alpha > 0$ depending on d, Chan [5, 6] showed how to improve the running time to $T_{\text{small}}(n, k_{\text{opt}}) = O(n \log k_{\text{opt}})$ by using data structures for linear programming queries; in other words, $T_{\text{small}}(n, k_{\text{opt}}) = O(n \log k_{\text{opt}} + k_{\text{opt}}^{O(1)})$ is a valid upper bound for all k_{opt}. The above theorem then implies an algorithm with running time $O_\varepsilon(T_{\text{small}}(\frac{n}{k_{\text{opt}}}\log n, \log n)) = O_\varepsilon(\frac{n}{k_{\text{opt}}}\log n \log k_{\text{opt}} + \log^{O(1)} n)$ for all k_{opt}. For $k_{\text{opt}} = \omega(\log n \log\log n)$, this is sublinear.

For $k_{\text{opt}} = O(\log n \log\log n)$, we can switch back to the $O(n \log k_{\text{opt}})$ algorithm. This implies an $O(n \log\log n)$ bound for all k_{opt} in any constant dimension.

The above algorithm is quintessentially Monte Carlo. We show how to obtain a Las Vegas algorithm in 3D (and thus in 2D as well), by using shallow cuttings as in Method 2:

Theorem 3.8. *We can solve the problem of approximate LP with violations in 3D in $O_\varepsilon(n \log n)$ expected time. This algorithm is always correct.*

Proof. We follow the algorithm from Theorem 3.7. The difference is that in solving the approximate decision problem for a given k, we try to certify that the chosen sample R, with sampling probability $p = \frac{c_\varepsilon \log n}{k}$, is indeed "good".

To do so, let H^- (resp. H^+) be the bounding planes of the lower (resp. upper) halfspaces. Let $R^- = R \cap H^-$ (resp. $R^+ = R \cap H^+$). We will certify that for each $k' = 0, \ldots, pk$, every point of level k' in R^- has level about $\frac{k'}{p}$ in H^- with additive error at most $O(\varepsilon k)$. With a similar procedure for H^+, we will then be able to guarantee the correctness of the decision algorithm. By Observation 2.1, certification succeeds w.h.p., since the number of combinatorially different points is polynomial; if certification fails at any moment, we can afford to switch to a brute-force algorithm.

To perform the certification, we follow the approximate level construction from the proof of Lemma 2.4. Consider each vertical prism Δ of the $O(\frac{k}{n})$-cutting covering the $(\leq k)$-level of H^-. Recall that Δ is further divided into a constant number of subcells δ via an ε-cutting.

Let R_Δ^- be the planes in R^- intersecting Δ. We construct the arrangement of R_Δ^-. For each feature f in this arrangement of level k' with $k' \leq pk$, we examine every subcell $\delta \subset \Delta$ intersecting f and confirm that ℓ_δ is about $\frac{k'}{p}$ with additive error $O(\varepsilon k)$. Since the levels of any two points in a subcell differ by at most $O(\varepsilon k)$, this property is sufficient for the certification. Since there are only a constant number of subcells in Δ, the cost per Δ is $O(|R_\Delta^-|^3)$, which has expected value $O(p^3k^3) = O_\varepsilon(\log^3 n)$, as Δ intersects $O(k)$ planes of H^-.

In fact, we can reduce some of the logarithmic factors by observing that it suffices to consider only k' of the form $\lfloor (1+\varepsilon)^i \rfloor$. Instead of examining all features f of the whole arrangement of R_Δ^-, we can construct approximate $(\leq k')$-levels of R_Δ^- over all such k', of total size $O(|R_\Delta^-|)$, and examine only features of the boundaries of these approximate levels. The cost per Δ is now $O_\varepsilon(|R_\Delta^-| \log |R_\Delta^-|)$, which has expected value $O_\varepsilon(\log n \log\log n)$.

The total extra cost for the certification, excluding $O(n \log n)$ preprocessing, is thus $O_\varepsilon(\frac{n}{k} \log n \log\log n)$. We can approximate k_{opt} as before. The total expected time of the algorithm becomes $O_\varepsilon(\frac{n}{k_{\text{opt}}} \log n \log k_{\text{opt}} + n\log n + \frac{n}{k_{\text{opt}}} \log n \log\log n)$. As before, for $k_{\text{opt}} = O(\log\log n)$, we can switch to the $O(n \log k_{\text{opt}})$ algorithm. □

Using a similar approach, we can also obtain an $O(n \log n)$ Las Vegas algorithm for the following problem: given n disks in 2D, find a point that has the maximum *depth*, i.e., lies inside the maximum number of disks.

Theorem 3.9. *We can compute a $(1+\varepsilon)$-factor approximation to the maximum depth in an arrangement of n disks in 2D in $O_\varepsilon(n \log n)$ expected time. This algorithm is always correct.*

Proof. First we can get a Monte-Carlo $O_\varepsilon(n \log n)$ algorithm, as described by Aronov and Har-Peled [2]: Let $T_{\text{small}}(n, k_{\text{opt}})$ denote the time required to solve the problem when the depth is k_{opt}; we can naively bound $T_{\text{small}}(n, k_{\text{opt}})$ by $O(n \log n + nk_{\text{opt}})$ by explicitly constructing the entire disk arrangement [2]. By an appropriate analog of Theorem 3.7, we can then solve the problem in time $O_\varepsilon(T_{\text{small}}(\frac{n}{k_{\text{opt}}} \log n, \log n)) = O_\varepsilon(\frac{n}{k_{\text{opt}}} \log^2 n)$, which is $O_\varepsilon(n \log n)$ for $k_{\text{opt}} = \omega(\log n)$. For $k_{\text{opt}} = O(\log n)$, we can switch to the $O(n \log n + nk_{\text{opt}})$ algorithm.

Now, to turn this into a Las Vegas algorithm, we use certification via shallow cuttings again. By applying the standard lifting map to transform the disks into planes in 3D [14], the problem is equivalent to approximating the maximum level for an arrangement of n planes over all points on the unit paraboloid. We can certify whether a sample of planes R chosen during Theorem 3.7 is "good" in exactly the same manner as in the proof of Theorem 3.8. The total expected time becomes $O_\varepsilon(\frac{n}{k_{\text{opt}}} \log^2 n + n \log n + \frac{n}{k_{\text{opt}}} \log n \log\log n)$. As before, for $k_{\text{opt}} = O(\log n)$, we can switch to the $O(n \log n + nk_{\text{opt}})$ algorithm. □

An intriguing question that remains is whether approximate LP with violations can be solved in linear time, Las Vegas or Monte Carlo. We don't know how even in 2D. For constant-factor approximations, however, we have the following result:

Theorem 3.10. *We can obtain a factor-$O(1)$ approximation for LP with violations in 2D in $O_\varepsilon(n)$ expected time. This algorithm is correct w.h.p.*

Proof. Let H^- (resp. H^+) be the bounding lines of the lower (resp. upper) halfplanes. We consider the following subproblem given an upper bound K to k_{opt}:

> (*) Find the largest $k \leq K$ such that there is a line separating L_k^- and L_k^+, where L_k^- denotes the k-level of H^- and the L_k^+ denotes the $(|H^+| - k)$-level of H^+.

We claim that this k approximates k_{opt} to within a factor of 4. Suppose there is a point q that violates k constraints. Then q is below L_k^- and above L_k^+, so no line can separate the L_k^- and L_k^+. Conversely, suppose no line separates L_k^- and L_k^+. Then the upper hull of L_k^- and the lower hull of L_k^+ must intersect, so there is a point q below an upper hull edge u^-v^- of L_k^- and above a lower hull edge u^+v^+ of L_k^+. Since u^- and v^- are on the k-level of H^-, q lies below the $(2k)$-level of H^-; similarly, q is above the $(2k)$-level of H^+. So q violates at most $4k$ constraints.

Now, to solve (*) exactly, we observe the close similarity of this subproblem with the problem of finding a point of maximum Tukey depth studied by Chan [10]. (The only difference is that there, after duality, we seek a line separating the k-level L_k^- and the $(|H^+|-k)$-level L_k^+ of a common set $H^- = H^+$.) The same "implicit linear programming" technique by Chan is applicable and reduces the problem further to this simpler problem:

(**) Given $k \le K$ and a line ℓ, decide whether ℓ separates L_k^- and L_k^+.

Specifically, if (**) can be solved in $D(n,K)$ time and $D(n,K)/n^\varepsilon$ is a monotone increasing function in n, then (*) can be solved in $O(D(n,K))$ expected time.

Solving (**) amounts to deciding whether a given line intersects a k-level, which reduces to a one-dimensional problem of selecting the K smallest and K largest elements, doable in $O(n + K \log n)$ time; e.g., see [**?**, proof of Theorem 5.2]. We replace $O(n + K \log n)$ with the upper bound $D(n,K) = O(n + K(n/K)^\varepsilon \log n)$ (so that $D(n,K)/n^\varepsilon$ is monotone increasing). We conclude that there is a Las Vegas algorithm for (*) with expected running time $O(n + K(n/K)^\varepsilon \log n)$, which is linear for $K = O(n/\log^2 n)$. For $k_{\mathrm{opt}} = \omega(n/\log^2 n)$, we can switch to the Monte Carlo sublinear algorithm noted before (with running time $O_\varepsilon(\frac{n}{k_{\mathrm{opt}}} \log n \log k_{\mathrm{opt}} + \log^{O(1)} n)$). □

Remark. The approximation factor 4 can be improved, but it is not clear how to bring the constant arbitrarily close to 1.

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999.

[2] B. Aronov and S. Har-Peled. On approximating the depth and related problems. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 886–894, 2005. Updated version at `http://valis.cs.uiuc.edu/~sariel/research/papers/04/depth/`, downloaded in November 2006.

[3] S. Arya and D. M. Mount. Approximate range searching. *Computational Geometry: Theory and Applications*, 17(3-4):135–152, 2000.

[4] M. Bern and D. Eppstein. Multivariate regression depth. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 315–321, 2000.

[5] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 284–290, 1996.

[6] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete and Computational Geometry*, 16:369–387, 1996.

[7] T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34:879–893, 2000.

[8] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\le k$)-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000.

[9] T. M. Chan. On enumerating and selecting distances. *International Journal of Computational Geometry and Applications*, 11:291–304, 2001.

[10] T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *Proceedings of the 15th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 430–436, 2004.

[11] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76–90, 1985.

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

[13] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55:441–453, 1997.

[14] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[15] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM Journal on Computing*, 15:271–284, 1986.

[16] S. Hart and M. Sharir. Nonlinearity of Daveport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6(2):151–177, 1986.

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

[18] H. Kaplan and M. Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar Voronoi diagrams, and approximate range counting. In *Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 484–493, 2006.

[19] S. Langerman and W. Steiger. The complexity of hyperplane depth in the plane. *Discrete and Computational Geometry*, 30(2):299–309, August 2003.

[20] J. Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, 2(3):169–186, 1992.

[21] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(2):157–182, 1993.

[22] J. Matoušek. On geometric optimization with few violated constraints. *Discrete and Computational Geometry*, 14:365–384, 1995.

[23] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1994.

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

[25] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, pages 390–399, 1999.

[26] P. J. Rousseeuw and M. Hubert. Regression depth. *Journal of American Statistical Association*, 94(446):388–402, 1999.

[27] M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for k-sets in three dimensions. *Discrete and Computational Geometry*, 26:195–204, 2001.