On Approximating the Depth and Related Problems[*]

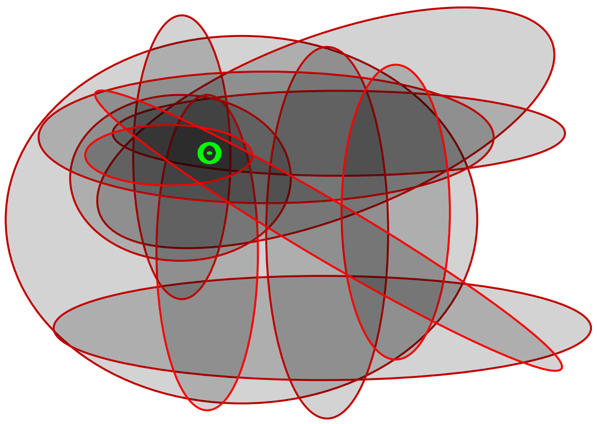
Boris Aronov[†]   Sariel Har-Peled[‡]

July 25, 2008[§]

Abstract

We study the question of finding a deepest point in an arrangement of regions, and provide a fast algorithm for this problem using random sampling, showing it sufficient to solve this problem when the deepest point is shallow. This implies, among other results, a fast algorithm for solving linear programming with violations approximately. We also use this technique to approximate the disk covering the largest number of red points, while avoiding all the blue points, given two such sets in the plane. Using similar techniques imply that approximate range counting queries have roughly the same time and space complexity as emptiness range queries.

1 Introduction



In this paper, we study the problem of computing efficiently the deepest point in a collection of regions. Here, the *depth* of the point is the number of regions that contain it. This is a natural problem that arises in optimization, where a constraint induces a region in space where it holds, and we would like to find the point that satisfies all the constraints. If no such point exists, we would like to find a point that satisfies the maximum number of constraints. As a concrete example, a linear program with n inequalities over d variables induces a set of n halfspaces in $\mathbb{R}^d$. The given linear program is feasible (i.e., it has a feasible solution) if there is a point of depth n in this arrangement of n halfspaces. Furthermore, if the given linear program is not feasible, then a point with maximum depth in the arrangement is a solution violating the least number of constraints.

[*]The latest version of this paper is available from http://sarielhp.org/p/04/depth. A preliminary version of this paper appeared as [AH05].

[†]Department of Computer and Information Science, Polytechnic University, Brooklyn, NY 11201, USA; http://cis.poly.edu/~aronov. Work on this paper was supported in part by NSF ITR Grant CCR-00-81964, by a grant from the U.S.-Israel Binational Science Foundation, and by NSA MSP Grant H98230-06-1-0016. Part of the work was carried out while B.A. visited UIUC in 2004.

[‡]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@uiuc.edu; http://sarielhp.org/. Work on this paper was partially supported by an NSF CAREER award CCR-0132901.

[§]ReLaTeXed on March 7, 2017.

In this paper, we provide several reductions for the problem of finding the deepest point, and show how to solve some depth problems using these techniques.

Computing a point of maximum depth via depth thresholding In Section 3, we present a general reduction that allows one to quickly compute an approximately deepest point in an arrangement of objects given a slower exact procedure for computing the depth exactly. More precisely, suppose we are given a *depth thresholding procedure* that can find, given an integer k, a deepest point for a set of n regions in $T_{\text{DT}}(n,k)$ time, provided the depth does not exceed k; if it does, the procedure outputs "depth $> k$". The resulting approximation algorithm, given a set $\mathcal{S}$ of n regions, can find a point of depth at least $(1-\varepsilon)k_{\text{opt}}$, where k_{opt} is the maximum number of regions covering any point. The running time of this new algorithm is $O(T_{\text{DT}}(n,\varepsilon^{-2}\log n)+n)$, assuming $T_{\text{DT}}(n,k)=\Omega(n)$, for any k. This reduction is applicable to any set of "well-behaved" regions in constant dimension.

Unless explicitly stated otherwise, all bounds on running time and space hold with high probability (i.e., at least $1-1/n^{O(1)}$) and also in expectation. The results returned by these algorithms are correct with high probability.

Next, in Section 4, we present several applications of this reduction. In Section 4.1, we show that one can solve linear programming with violations approximately in near-linear time. Explicitly, consider a linear program L with n constraints in $\mathbb{R}^d$ and a linear objective function f. Suppose k_{opt} is the minimum number of constraints that have to be violated to make L feasible and let v be the optimal solution in this case; namely, v is the point minimizing $f(v)$ with exactly k_{opt} violated constraints. Then, one can find a point u that violates at most $(1+\varepsilon)k_{\text{opt}}$ constraints of L and such that $f(u)\le f(v)$. The running time of the new algorithm is $O(n(\varepsilon^{-2}\log n)^{d+1})$. This compares favorably with the previous exact algorithm of Matoušek [Mat95] which requires $O(nk^{d+1})$ running time. (In two and three dimensions faster exact algorithms exist [Cha05]. See Section 4.1.) To appreciate this result, consider the case where $k=\sqrt{n}$. A natural approach to approximate linear programming with violations is to compute a δ-approximation [VC71] to the set of constraints in L, and apply the exact algorithm on this sample. However, in this case $\delta=\varepsilon/\sqrt{n}$, and the required random sample would include (almost) all the constraints of L, thus achieving no speedup.

An application: Finding a disc covering red points, but no blue points Consider two finite sets of points in the plane, *red* and *blue*. As another application of the aforementioned technique, we investigate the problem of finding a disk containing the largest number of red points, while avoiding all the blue points. This is a natural problem related to learning and clustering [DHS01]. For example, one might try to learn a concept believed to be a disk, from examples, where the red points represent positive examples and the blue ones represent the negative ones. A possible criterion to optimize is to say that the best concept (i.e., disk) is the one that classifies the blue points correctly (i.e., avoids them), while minimizing the error on the red points (i.e., the disk covers as many red points as possible).

The corresponding problem, in high dimensions, of computing the best such ball seems to be computationally hard, since computing the separating hyperplane minimizing the number of outliers is NP-Hard [AK95]. In fact, approximating it within a factor of $2^{\log^{0.5-\varepsilon} n}$, for any fixed $\varepsilon>0$, is "almost" NP-Hard, see [ABSS97] for details. In constant dimension d, this ball can be computed by brute-force enumeration of all possibilities, in $n^{O(d)}$ time. Motivated by this unattractive option, we develop approximation algorithms for this problem, where we look for the ball containing (approximately) the largest number of red points, while avoiding all the blue points. More specifically, we observe that every red point p induces a feasible region which is the locus of the centers of the maximal balls containing p and not containing any blue points in their interior. Thus, the problem reduces to that of finding a point in the plane (or in space) having the largest number of such regions covering it; namely, a *deepest* point in an arrangement of such regions.

In Section 4.2, we show that computing the deepest point in an arrangement of disks in the plane is 3SUM-hard [GO95] and present a $(1-\varepsilon)$-approximation algorithm with $O(n\varepsilon^{-2}\log n)$ expected running time. In Section 4.3, we investigate the geometry of the problem of, given a set of blue and a set of red points, computing the disk containing the largest number of red points while avoiding all blue points. In particular, we show that this is equivalent to, given a set of red planes and a set of blue planes in three dimensions, computing the point having the largest number of red planes below it, while having all the blue planes above it. The points lying below all blue planes and above a specific red plane form the *feasible region* of that plane. Projecting the feasible regions to the plane, we obtain a set of pseudodisks which can be manipulated efficiently using an implicit representation. Plugging this into the algorithm for computing the depth of an arrangement of pseudodisks results in an approximation algorithm, with $O(n\varepsilon^{-2}\log^2 n)$ expected running time, which returns a disk covering $(1-\varepsilon)k_{\mathrm{opt}}$ points, where k_{opt} is the number of red points covered by the optimal disk. We also study this problem in higher dimensions.

Approximate counting. Given a set of points P in $\mathbb{R}^d$ and a class of ranges, the problem of *range searching* is preprocessing P so that, given a range τ, one can quickly answer (i) *emptiness queries*: test whether $\tau \cap P = \emptyset$, (ii) *counting queries*: report the size of $\tau \cap P$, or (iii) *reporting queries*: report the elements of $\tau \cap P$. This problem and its variants have numerous applications and have received substantial attention. See [AE98] for a survey of the subject.

Interestingly, there is a complexity gap between emptiness queries and counting queries. In the (somewhat reasonable) computation model assumed by Brönnimann *et al.* [BCP93], halfspace counting queries require $\Omega^*(n^{1-2/(d+1)})$ time per query, if only linear space is allowed.[1] Matoušek [Mat93] showed how to answer such queries in $O(n^{1-1/d})$ time. On the other hand, halfspace emptiness queries can be answered in logarithmic time in two and three dimensions, and in $O^*(n^{1-1/\lfloor d/2\rfloor})$ time in higher dimensions [Mat92], using near-linear space; this is slightly faster than the aforementioned lower bound for the counting problem. Especially interesting is the situation in dimensions two and three, where the gap is between logarithmic query time for emptiness queries and polynomial query time for counting queries.

In Section 5, we show that this gap disappears if one is willing to get an approximate answer to the counting query. In particular, given a prespecified ε, $0 < \varepsilon < 1/2$, we show how to reduce a counting query to a sequence of emptiness queries of length polynomial in $\log n$ and ε^{-1}. For any range τ, this data structure reports a number μ_τ, such that $(1-\varepsilon)\,|\tau \cap P| \le \mu_\tau \le |\tau \cap P|$. Thus, approximate counting and emptiness range searching are computationally nearly equivalent. Since this circumvents the aforementioned gap, we believe it to be of independent interest.

We contrast our results with the work of Arya and Mount [AM00] that considers a different notion of approximation for the range searching problem. They approximate the range (using a distance which depends on the diameter of the query shape) and return the *exact* count inside this *approximate* shape. In our results, on the other hand, the shape is fixed but the count returned is approximated. In particular, our results apply to halfspace range searching, while Arya and Mount methods cannot be applied here, as a halfspace has infinite diameter.

Our results are based on careful application of random sampling, using several multi-resolution samples. By performing the computation at the right resolution, we are able to achieve fast running

[1]We use $f(n) = O^*(g(n))$ to express $f(n) = O(g(n)\log^c n)$ and $f(n) = \Omega^*(g(n))$ to mean $f(n) = \Omega(g(n)\log^{-c} n)$, for some constant $c > 0$.

time. This technique is by now standard in the area of randomized algorithms. Recent results that use similar techniques include [IM98, Ind00, HI00]; this list is by no means exhaustive. In the context of halfspace range searching, Chan [Cha00] used multi-resolution random samples to roughly estimate the depth of the query, and speed up halfspace range *reporting* queries, in the expected sense. Finally, Agarwal *et al.* [AHR+02] studied a similar problem (in somewhat easier settings) and also used random sampling to get a fast approximation to the point of maximum depth.

2 Preliminaries

Given a set of objects $\mathcal{S}$ in $\mathbb{R}^d$ and a point $q \in \mathbb{R}^d$, let the *depth of q in $\mathcal{S}$*, $\mathrm{depth}(q, \mathcal{S})$, be the number of objects of $\mathcal{S}$ containing q. The *depth of $\mathcal{S}$* is defined as $\max_q \mathrm{depth}(q, \mathcal{S})$, with q ranging over all of $\mathbb{R}^d$. Finally, let $\mathrm{core}(\mathcal{S})$ denote the locus of points realizing $\mathrm{depth}(\mathcal{S})$.

In the remainder of the paper, we assume that the sets $\mathcal{S}$ are well-behaved. In particular, we require that the complexity of the arrangement formed by $\mathcal{S}$ be bounded by $|\mathcal{S}|^{O(d)}$. The *arrangement* formed by $\mathcal{S}$ is the decomposition of the plane (or, more generally $\mathbb{R}^d$) induced by a collection of such shapes [SA95]. The *combinatorial complexity* of an arrangement is the total number of edges, faces, and vertices in the arrangement. A *k-level* in an arrangement of curves is the closure of the set of points on the curves that have exactly k curves below them. For a set of regions R, *the union of R* is the set of points in the plane covered by at least one region of R. For a union of regions, its *complexity* is the number of edges and vertices of the arrangement lying on the boundary of the union (i.e., this is the descriptive complexity of the union). Analogous definitions apply in higher dimensions.

In the following, we need the Chernoff inequality, see [MR95].

Theorem 2.1 (Chernoff inequality.) *Let $X_1, \dots X_n$ be n independent Bernoulli trials, where*

$$\mathbf{Pr}[X_i = 1] = p_i, \quad \mathbf{Pr}[X_i = 0] = 1 - p_i, \quad Y = \sum_i X_i, \quad \text{and} \quad \mu = \mathbf{E}[Y]\,.$$

Then, for any $\delta > 0$,

$$\mathbf{Pr}[Y > (1+\delta)\mu] < \begin{cases} \exp(-\mu\delta^2/4) & \delta \le 2e - 1 \text{ (case (a))}, \\ 2^{-\mu(1+\delta)} & \delta \ge 2e - 1 \text{ (case (b))}, \end{cases} \tag{2.1}$$

and

$$\mathbf{Pr}[Y < (1-\delta)\mu] \le \exp\bigl(-\mu\delta^2/2\bigr)\,. \tag{2.2}$$

3 From Exact Depth to Fast Approximate Depth

Given a set $\mathcal{S}$ of n objects and $\varepsilon > 0$, we consider the following two problems:

Problem 1: Computing $\mathrm{depth}(\mathcal{S})$ exactly, perhaps also producing a *witness point*, i.e., a point in $\mathrm{core}(\mathcal{S})$.

Problem 2: Estimating $\delta = \mathrm{depth}(\mathcal{S})$, i.e., producing an integer k with the $(1-\varepsilon)\delta \le k \le \delta$ together with a witness point of that depth.

We will also need, as a subroutine, a procedure DEPTHTRESHOLD$(\mathcal{S}', k)$ with the following behavior: Given a collection $\mathcal{S}' \subset \mathcal{S}$ of n' objects and an integer $k > 0$, determine depth$(\mathcal{S}')$ exactly (and produce a witness point of this depth), *if* it does not exceed k, otherwise just return "depth$(\mathcal{S}') > k$". We refer to this as *depth thresholding*. Let $T_{\text{DT}}(n', k)$ be its running time.

Solving problem (1) seems to require computing the whole arrangement $\mathcal{A}(\mathcal{S})$ (at least in general settings). Therefore a running time better than $O(n^d)$ for this problem seems unlikely. As such, we will concentrate in this section on the approximation version of this problem, namely, problem (2). In particular, in this section, we show how to solve problem (2) by performing a logarithmic number of depth thresholding calls.

3.1 An approximate decision procedure

The intuitive idea behind approximating $\delta := \text{depth}(\mathcal{S})$ for a collection $\mathcal{S}$ of n objects is to perform a binary search on the value of the depth. However, we cannot afford to use an exact decision procedure (i.e., a test comparing δ to a given number k) in each round, as all known methods of computing the depth exactly essentially compute the full arrangement $\mathcal{A} = \mathcal{A}(\mathcal{S})$. (Actually, using the depth thresholding idea, one can test whether $\delta > k$ for any given k, without necessarily computing the whole arrangement, but the cost $T_{\text{DT}}(n, k)$ of doing this usually grows with k, so it is too expensive to perform the test for large values of k. This is precisely the case where the randomized procedure given below is more efficient.) The idea is to first perform a "rough" search for the right depth (up to a constant factor). Once we are in the right range, one can find the approximate depth by applying the depth thresholding procedure to a single random sample, using the relation between the depth of $\mathcal{S}$ and that of the sample.

To this end, we develop an approximate decision procedure, which, given a value k, returns "$k < \delta$" with high probability if k is significantly smaller than δ and "$k > \delta$" with high probability if k is significantly larger than δ. We start by proving the following technical lemma.

Lemma 3.1 *Let $\mathcal{S}$ be a set of n objects in $\mathbb{R}^d$, ε, $0 < \varepsilon < 1/2$, be fixed, $k > 0$ be an integer, and $R \subseteq \mathcal{S}$ be a random subset formed by picking each object with probability*

$$\psi = \psi(\varepsilon, k) := \min\left(c_1 \frac{\log n}{k\varepsilon^2}\ ,\ 1\right),$$

independently, where c_1 is a sufficiently large constant. Then, for a point $p \in \mathbb{R}^d$, we have, with high probability, that

(i) If $\text{depth}(p, R) \geq \alpha_+(\varepsilon, k) := 2k\psi$, *then with high probability,* $\text{depth}(p, \mathcal{S}) \geq (3/2)k$.

(ii) If $\text{depth}(p, R) \leq \alpha_-(\varepsilon, k) := (1-\varepsilon)k\psi$, *then with high probability* $\text{depth}(p, \mathcal{S}) \leq k$.

(iii) If $\text{depth}(p, R) \geq \alpha_-(\varepsilon, k)$ *then, with high probability,*

$$(1-\varepsilon)\,\text{depth}(p, \mathcal{S}) \leq \frac{\text{depth}(p, R)}{\psi} \leq (1+\varepsilon)\,\text{depth}(p, \mathcal{S});$$

namely, the depth of a "deep" point in $\mathcal{A}(R)$ is a good estimate of its depth in $\mathcal{A}(\mathcal{S})$.

Proof: If $\psi = 1$, there is nothing to prove, so we assume $\psi < 1$ in what follows.

(i) Consider a point $p \in \mathcal{A}(\mathcal{S})$ such that $r := \mathrm{depth}(p, \mathcal{S}) < (3/2)k$. We have that $\mu := \mathbf{E}[\mathrm{depth}(p, R)] = r\psi$ and $r(1 + \varepsilon/4) < 2k$. Therefore,

$$\nu := \mathbf{Pr}[\mathrm{depth}(p, R) \geq 2k\psi] = \mathbf{Pr}\left[\mathrm{depth}(p, R) \geq \frac{2k}{r}\mu\right]$$
$$= \mathbf{Pr}\left[\mathrm{depth}(p, R) \geq \left(1 + \left(\frac{2k}{r} - 1\right)\right)\mu\right].$$

If $\frac{2k}{r} - 1 \leq 2e - 1$, i.e., $k/e \leq r$, then, by the Chernoff inequality Eq. (2.1) (a), we have

$$\nu \leq \exp\left(-\mu\left(\frac{2k}{r} - 1\right)^2 / 4\right) = \exp\left(-r\psi\left(\frac{2k}{r} - 1\right)^2 / 4\right) \leq \exp\left(-r\left(c_1\frac{\log n}{k\varepsilon^2}\right)\frac{\varepsilon^2}{36}\right)$$
$$\leq \exp\left(-\frac{k}{e}\left(c_1\frac{\log n}{k\varepsilon^2}\right)\frac{\varepsilon^2}{36}\right) = \frac{1}{n^{\Omega(1)}}.$$

If $r \geq k/e$ then, by the Chernoff inequality (Eq. (2.1) (b)), we have

$$\nu \leq 2^{-2k\mu/r} = 2^{-2k\psi} \leq \frac{1}{n^{\Omega(1)}}.$$

(ii) Let p be a point of $\mathbb{R}^d$, such that $r = \mathrm{depth}(p, \mathcal{S}) \geq k$. We have that $\mu = \mathbf{E}[\mathrm{depth}(p, R)] = r\psi$. Arguing as above, we have

$$\mathbf{Pr}[\mathrm{depth}(p, R) < (1 - \varepsilon)k\psi] \leq \mathbf{Pr}[\mathrm{depth}(p, R) < (1 - \varepsilon)r\psi] \leq \exp\left(-\frac{\varepsilon^2}{2}r\psi\right) \leq \frac{1}{n^{\Omega(1)}}.$$

(iii) Arguing as in case (i), we have that if $\mathrm{depth}(p, R) \geq \alpha_-(\varepsilon, k)$ then $\mathrm{depth}(p, \mathcal{S}) \geq k/2$, and this holds with high probability. As such, we have that $\mu = \mathbf{E}[\mathrm{depth}(p, R)] \geq k\psi/2$. This implies, using the Chernoff inequality (Theorem 2.1), that

$$\mathbf{Pr}[\mathrm{depth}(p, R) > (1 + \varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2}{4}\mu\right) \leq \exp\left(-\frac{\varepsilon^2}{8}k\psi\right) \leq \frac{1}{n^{\Omega(1)}}.$$

Similarly,

$$\mathbf{Pr}[\mathrm{depth}(p, R) < (1 - \varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2}{2}\mu\right) \leq \frac{1}{n^{\Omega(1)}},$$

implying the claim. ■

Corollary 3.2 *Let $\mathcal{S}$ be a set of n objects, with $\delta = \mathrm{depth}(\mathcal{S})$. Let ε, $0 < \varepsilon < 1/2$, be fixed, $k \geq \delta/4$ be an integer, and $R \subseteq \mathcal{S}$ be a random subset formed by picking each object with probability*

$$\psi = \psi(\varepsilon, k) := \min\left(c_1\frac{\log n}{k\varepsilon^2}\ ,\ 1\right),$$

independently, where c_1 is a sufficiently large constant. Then, we have:

(i) If $\mathrm{depth}(R) \geq \alpha_+(\varepsilon, k) := 2k\psi$, then with high probability, $\delta \geq (3/2)k$.

(ii) If $\mathrm{depth}(R) \leq \alpha_-(\varepsilon, k) := (1 - \varepsilon)k\psi$, then with high probability $\delta \leq k$.

(iii) For all $p \in \mathbb{R}^d$, such that $\mathrm{depth}(p, R) \geq \alpha_-(\varepsilon, k)$ we have, with high probability, that

$$(1 - \varepsilon)\,\mathrm{depth}(p, \mathcal{S}) \leq \frac{\mathrm{depth}(p, R)}{\psi} \leq (1 + \varepsilon)\,\mathrm{depth}(p, \mathcal{S}).$$

Proof: Follows immediately from Lemma 3.1, since the complexity of $\mathcal{A}(\mathcal{S})$ is polynomial. Indeed, place a point inside each face of $\mathcal{A}(\mathcal{S})$, and apply the lemma to all these points. ■

3.1.1 If we know where to look, we can find it

Assume that we are given an estimate y of $\delta := \mathrm{depth}(\mathcal{S})$, with $y \leq \delta \leq 8y$. Then we can compute approximately the depth of $\mathcal{S}$. Indeed, compute a random sample R of $\mathcal{S}$ as specified by Lemma 3.1, for $k = y/2$. Next, perform depth thresholding on R with threshold $M := 16y\psi = O(\varepsilon^{-2}\log n)$, where $\psi = \psi(\varepsilon/4, k)$. It is easy to verify that, with high probability (by the Chernoff inequality), the depth of R is bounded from above by M (as the expected maximum depth is $M/2$). Furthermore, $\delta > k$, and by Corollary 3.2 (ii), we have $\mathrm{depth}(R) > \alpha_-(\varepsilon/4, k)$ (again, with high probability). Let v be the witness point returned by DepthTreshold(R, M) with the maximum depth in R. The depth of v in $\mathcal{A}(R)$ exceeds $\alpha_-(\varepsilon/4, k)$, and by Corollary 3.2 (iii), $\mathrm{depth}(v, R)/\psi$ is a $(1 \pm \varepsilon/4)$-approximation to $\mathrm{depth}(v, \mathcal{S})$. In particular,

$$\mathrm{depth}(v, R) \leq (1 + \varepsilon/4)\psi\,\mathrm{depth}(v, \mathcal{S}).$$

Now, consider a point $q \in \mathrm{core}(\mathcal{S})$, that is with $\mathrm{depth}(q, \mathcal{S}) = \mathrm{depth}(\mathcal{S})$. As before, with high probability, $\mathrm{depth}(q, R) \geq \alpha_-(\varepsilon/4, k)$, and we have

$$(1 - \varepsilon/4)\psi\,\mathrm{depth}(\mathcal{S}) \leq \mathrm{depth}(q, R) \leq \mathrm{depth}(v, R) \leq (1 + \varepsilon/4)\psi\,\mathrm{depth}(v, \mathcal{S}).$$

Namely, $\mathrm{depth}(v, \mathcal{S}) \geq (1 - \varepsilon)\,\mathrm{depth}(\mathcal{S})$ and

$$(1 - \varepsilon/4)\,\mathrm{depth}(\mathcal{S}) \leq \frac{\mathrm{depth}(v, R)}{\psi} \leq (1 + \varepsilon/4)\,\mathrm{depth}(\mathcal{S}).$$

Computing v takes $O(n + T_{\mathrm{DT}}(2n\psi(\varepsilon/4, k), O(\varepsilon^{-2}\log n)))$ time and succeeds with high probability, where $2n\psi(\varepsilon/4, k)$ is an upper bound on the size of R that holds with high probability. Let FindDeepPoint$(\mathcal{S}, y)$ denote this algorithm.

3.1.2 Testing the water

It remains to find a good estimate of the maximum depth of $\mathcal{S}$. Assume that we have a guess k, such that $\delta \leq 4k$. We construct procedure DepthTest that outputs either "guess too big" or a range $[y, 4y]$ containing δ with high probability.

The procedure DepthTest works by computing the sample R, according to Corollary 3.2, and applying the depth thresholding to R; formally, we execute DepthTreshold$(R, \alpha_+(\varepsilon, k))$. Consider the random variable $X = \mathrm{depth}(R)$.

If $X \geq \alpha_+(\varepsilon, k)$ (i.e., DepthTreshold returned "depth exceeds threshold") then it follows that $\delta \geq (3/2)k$ by Corollary 3.2 (i). In addition, since we assumed that $\delta \leq 4k$, we have that $(3/2)k \leq \delta \leq 4k$, and DepthTest returns $[k, 4k]$ as the range containing δ.

If $X \leq \alpha_-(\varepsilon, k)$ then by Corollary 3.2(ii), with high probability, $\delta \leq k$. Namely, our guess k of the depth is too large, and DepthTest outputs "guess too high".

If $\alpha_-(\varepsilon, k) < X < \alpha_+(\varepsilon, k)$ then, by Lemma 3.1 (iii), with high probability, we have that $(1 - \varepsilon)k \leq \delta \leq 2k(1 + \varepsilon)$, so DepthTest returns $[k/2, 4k]$ as the range containing δ.

The running time of DepthTest is dominated by the running time of DepthTreshold.

3.1.3 The algorithm

The procedure DepthTest can tell us if the guess k for the maximum depth is in the right range, or alternatively that the guess is too large. We use it to perform an exponential decreasing search for the right range, as follows.

Theorem 3.3 *Let $\mathcal{S}$ be a set of n objects in $\mathbb{R}^d$ of depth δ. Suppose there exists an algorithm implementing depth thresholding for depth at most k in a set of m objects in time $T_{\mathrm{DT}}(m,k)$. Then, given a prespecified parameter ε, $0 < \varepsilon < 1/2$, one can compute a point of depth at least $(1-\varepsilon)\delta$ in $O(n + T_{\mathrm{DT}}(r', \delta') \log n)$ expected time, where $\delta' = \min(c_2 \varepsilon^{-2} \log n, n)$ and $r' = \min(n, \delta' n/\delta)$. In fact, the running time is $O(n + T_{\mathrm{DT}}(r', \delta'))$ if $T_{\mathrm{DT}}(r', k)$ is $\Omega(r')$, for any k. The output is correct with high probability.*

Proof: Let $k_i = n/2^i$, for $i = 1, \ldots, \lceil \lg n \rceil$. In the ith iteration, we estimate whether $\delta \leq k_i$, using DepthTest. This requires $O(n + T_{\mathrm{DT}}(r_i, \delta_i))$ time, where δ_i and r_i is the depth and the size of the ith random sample, respectively. If k_i is too big according to DepthTest then we continue to the next iteration.

Otherwise, we know that $k_i \leq \delta \leq 4k_i$. We now use FindDeepPoint$(\mathcal{S}, k_i)$ and this returns the required point and its depth, see Section 3.1.1. Note, that as far a running time, this invocation of FindDeepPoint takes (asymptotically) the same time as the last call to DepthTest (as both procedures essentially "just" call DepthTreshold).

Since at the last call to the FindDeepPoint we have $k_i \leq \delta \leq 4k_i$, it follows that the expected depth of the random sample used is $O(\varepsilon^{-2} \log n)$. Note, that the sizes of the samples used DepthTest form an increasing geometric sequence. Thus, the last iteration uses the largest sample. Furthermore, the expected depth of the sample increases with each sample. Therefore, using Chernoff inequality again (we omit the straightforward, but tedious details), we deduce that, with high probability, $\delta_i = O(\varepsilon^{-2} \log n)$, for $i \geq 1$.

Since we use $O(\log n)$ calls to DepthTest, the claim follows. As for the improved running time, observe that in those $O(\log n)$ calls, as mentioned above, we use samples of geometrically increasing sizes. Thus the overall running time is dominated by the cost of FindDeepPoint, which results in a call to the depth thresholding algorithm. With high probability, the sample size in the last iteration is $O(c_2 (n/\delta) \varepsilon^{-2} \log n)$ as claimed.

To achieve further speedup, observe that we can compute all the random samples in advance. Let $R_1 := \mathcal{S}$, and let R_i be the sample computed from R_{i-1}, by picking each element with probability $1/2$ (this is known as *gradation*). Clearly, this takes $O(n)$ time, with high probability, and we can use the appropriate random sample when needed. Thus, the overall time to compute the random samples is linear. (Note, that although the samples are no longer independent, the analysis still works since we have used the union bound to bound the probability of failure.) ■

In the above algorithm the call to FindDeepPoint can be carried out using the results of the last DepthTest call issued by the algorithm. Therefore, the procedure FindDeepPoint can be merged with DepthTest to create a simpler algorithm. We believe, however, that the current presentation is more intuitive (and we hope that the reader will agree).

3.2 Finding a point of minimum depth

For a set of objects $\mathcal{S}$ in $\mathbb{R}^d$, let $\mathrm{depth}_{\min}(\mathcal{S})$ denote the depth of a point in $\mathbb{R}^d$ covered by fewest objects in $\mathcal{S}$. An algorithm computes a *minimum depth thresholding* for a set X of objects, if it returns a point of minimum depth at most k in $\mathcal{A}(X)$, in time $T_{\mathrm{minDT}}(|X|, k)$, or alternatively reports that all points in $\mathcal{A}(X)$ are of depth larger than k.

It is easy to verify that above discussion can be adopted to this "complementary" setting.

Theorem 3.4 *Let $\mathcal{S}$ be a set of n objects in $\mathbb{R}^d$ of minimum depth $\delta = \mathrm{depth}_{\min}(\mathcal{S})$. Suppose there exists an algorithm implementing depth thresholding for depth at most k in a set of m objects in time*

$T_{\mathrm{minDT}}(m,k)$. Then, given a prespecified parameter ε, $0 < \varepsilon < 1/2$, one can compute a point of depth at most $(1+\varepsilon)\delta$ in $O(n + T_{\mathrm{minDT}}(r', \delta')\log n)$ expected time, where $\delta' = \min(c_2\varepsilon^{-2}\log n, n)$ and $r' = \min(n, \delta' n/\delta)$. In fact, the running time is $O(n + T_{\mathrm{minDT}}(r', \delta'))$ if $T_{\mathrm{minDT}}(r', k)$ is $\Omega(r')$, for any k. The output is correct with high probability.

In fact, if we are given a target depth k, there is no need to perform the binary search for the right depth, and we obtain the following:

Theorem 3.5 *Let $\mathcal{S}$ be a set of n objects in $\mathbb{R}^d$. Let $f(\cdot)$ be a function defined over $\mathbb{R}^d$. Let `Alg` be an algorithm that, given $X \subseteq \mathcal{S}$, and a parameter t, can report the point $x \in \mathbb{R}^d$, such that $f(x)$ is the minimum among all points contained in t or fewer regions of X (if no such point exists `Alg` outputs "infeasible"). Suppose that the running time of `Alg` is $T_{\mathrm{minDT}}(|X|, t)$. Then, given prespecified parameters $\varepsilon > 0$ and k, one can compute a point $u \in \mathbb{R}^d$ of depth at most $(1+\varepsilon)k$ in $\mathcal{S}$, in $O(n + T_{\mathrm{minDT}}(r', \delta'))$ expected time, where $\delta' = \min(c_2\varepsilon^{-2}\log n, n)$ and $r' = \min(n, \delta' n/k)$. Furthermore, $f(u) \le f(v_k)$, where v_k is a point minimizing f among all points in $\mathbb{R}^d$ of depth k or less in $\mathcal{A}(\mathcal{S})$s. The out is correct with high probability.*

4 Applications

4.1 Linear programming with violations

Given a linear program L with n constraints in $\mathbb{R}^d$, one can determine whether there exists a point that violates at most k constraints of L in $O(nk^{d+1})$ time [Mat95]. This problem can be solved in $O((n + k^2)\log n)$ time in two dimensions, and in $O(n + k^{11/4}n^{1/4})$ time in three dimensions [Cha05]. Using these algorithms to implement depth thresholding in Theorem 3.4 and Theorem 3.5 results in the following:

Theorem 4.1 *Let L be a linear program with n constrains in $\mathbb{R}^d$, and let f be the objective function to be minimized. Let k_{opt} be the minimum number of constraints that must be violated to make L feasible, and let v be the point minimizing $f(v)$ with k_{opt} constraints violated. Then one can output a point $u \in \mathbb{R}^d$ such that u violates at most $(1+\varepsilon)k_{\mathrm{opt}}$ constraints of L, and $f(u) \le f(v)$. The results returned are correct with high probability. The expected running time (which also holds with high probability) of this algorithm is*

- $O\Big(n\log(\varepsilon^{-1}\log n) + (\varepsilon^{-1}\log n)^{O(1)}\Big)$, *for $d = 2, 3$, and*
- $O\Big(n(\varepsilon^{-2}\log n)^{d+1}\Big)$, *for $d > 3$.*

Remark 4.2 For the algorithm of Theorem 4.1, if k is specified in advance, one can find a point u violating at most $(1+\varepsilon)k$ constraints of L, with $f(u) \le f(v_k)$, where v_k is the optimal solution violating at most k constraints. If L is not feasible when violating only k constraints, the algorithm may output "infeasible".

Note, that this observation implies a number of new results, see the introduction of Chan [Cha05] for a list of problems that can be solved approximately using our techniques. For example, in $\mathbb{R}^d$, given n points, an integer k, $0 < k < n$, and $\varepsilon > 0$, one can find a spherical shell containing all but $(1+\varepsilon)k$ points, which has a volume smaller than the minimum-volume spherical shell containing all but k points. The resulting running time is $O(n(\varepsilon^{-1}\log n)^{O(d)})$.

Remark 4.3 Observe, that if k_{opt} is sufficiently large in Theorem 4.1, then the running time is in fact linear, as the samples used by the algorithm are sufficiently small. In particular, for $k_{\rm opt} = \Omega\Big((\varepsilon^{-2}\log n)^{d+2}\Big)$, the running time of the algorithm of Theorem 4.1 is $O(n)$.

Remark 4.4 (Handling Outliers.) Using Theorem 3.5 and plugging it into the techniques of Har-Peled and Wang [HW04], one can solve shape fitting problems with outliers in near-linear time. For example, given a set P of n points in the plane and parameters $k, \varepsilon, \delta > 0$, one can compute an annulus A, in expected $O(n + (\varepsilon^{-1}\delta^{-1}\log n)^{O(1)})$ time. With high probability, the annulus A contains all but at most $(1+\varepsilon)k$ points of P and its width is at most $(1+\delta)w_{\rm opt}(P,k)$, where $w_{\rm opt}(P,k)$ is the minimum width of an annulus containing all but k points of P. Note that the approximation here is both in terms of the width of the annulus and the number of outliers. This result was known before only for the case when the number of outliers was large (i.e., close to n). Our methods can be applied to all the problems studied by Har-Peled and Wang [HW04]. To prevent this paper from deteriorating into a shopping list, we omit any further details.

4.2 Approximating the deepest point in a pseudodisk arrangement

Lemma 4.5 *Determining a point of maximum depth in an arrangement of n disks is* 3SUM*-hard.*

Proof: It is 3SUM-hard to decide, given a set of lines in the plane with integer coefficients, whether any three of the lines have a point in common [GO95]. Consider such a set L of n lines. We assume they are given by equations of the form $ax + by = c$, with $a, b, c \in \mathbb{Z}$.

One can compute in $O(n \log n)$ time an axis-parallel square Q containing all vertices of the arrangement $\mathcal{A}(L)$. Indeed, let M be the coefficient with the largest absolute value appearing in the n given lines. For two lines $ax + by = c$ and $a'x + b'y = c'$, the coordinates of their intersection point is, by Cramer's rule, a ratio of two 2×2 determinants with entries from among these six numbers. So the x coordinate of such an intersection point is in $[-2M^2, 2M^2]$. Arguing similarly for y, we conclude that the square $Q = [-2M^2, 2M^2]^2$ contains all the intersection points of the lines (and it can be computed in $O(n)$ time). Alternatively, one can compute the leftmost, rightmost, topmost and bottommost vertices in the arrangement of the lines L in $O(n \log n)$ time [Mat91].

Furthermore, one can compute a lower bound Δ on the distance between a vertex of $\mathcal{A}(L)$ and a line of L not passing through it. Indeed, consider a line ℓ: $ax + by = c$, where $|a|, |b|, |c| \le M$, and a point $p = (\alpha, \beta)$ which is formed by the intersection of two given lines. The distance of p from ℓ is

$$x = \frac{a\alpha + b\beta - c}{\sqrt{a^2 + b^2}}.$$

Now, $a\alpha + b\beta - c$ is a rational number with dominator smaller than $4M^4$, since the denominator of α and β are smaller than $2M^2$ by the above argument, and a, b, c are integers. Thus, if the numerator of x is non-zero, its absolute value is at least $1/(4M^4)$. Therefore, if x is not zero then

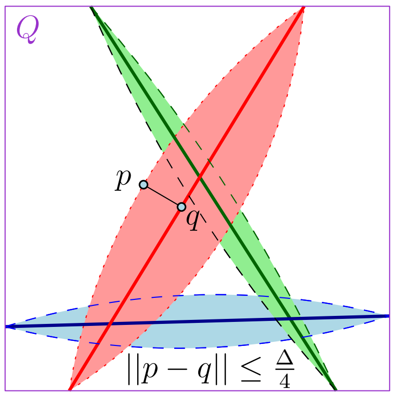


Figure 4.1

$$|x| = \frac{|a\alpha + b\beta - c|}{\sqrt{a^2 + b^2}} \ge \frac{1/4M^4}{\sqrt{2M^2}} > \frac{1}{8M^5} =: \Delta.$$

Next, we replace each line $\ell \in L$ by two sufficiently large disks D_ℓ, D'_ℓ of equal radii, such that $\partial D_\ell \cap \partial D'_\ell = \ell \cap \partial Q$ (so that $D_\ell \cap D'_\ell$ is symmetric with respect to ℓ) and $D_\ell \cap D'_\ell$ lies in the strip of halfwidth $\Delta/4$ centered at ℓ. Namely, we replace the segment $\ell \cap Q$ by a lens formed by the intersection of two "large" disks. The union of the two disks that form the lens contains the square Q. By construction, three original lines intersect if and only if the three corresponding lenses share a point. See Figure 4.1.

Let $\mathcal{D}$ be the resulting set of $2n$ disks. Note that no point outside Q is covered by more than n of the disks. Moreover, there is a point in Q contained in at least $n+3$ disks of $\mathcal{D}$ if and only if some three lines of L share a point. The reduction takes linear time. ■

Having argued that computing the depth exactly appears difficult, we turn to approximating it. Consider a family $\mathcal{S}$ of n *x-monotone pseudodisks*, i.e., a collection of regions, each bounded by a connected closed curve of constant algebraic complexity, such that any vertical line intersects a region in a connected interval, if at all, and such that the boundaries of any two regions cross at most twice. In the following, we assume that we can perform the geometric primitives on pseudodisks in constant time per operation. We need the following facts:

(i) The complexity of the union $U := \bigcup \mathcal{S}$ of n x-monotone pseudodisks is $O(n)$ [KLPS86].

(ii) Using a randomized incremental construction, one can compute U in $O(n \log n)$ expected time [MMP$^+$94]. Furthermore, in the same amount of time one can preprocess U for $O(\log n)$-time point-location queries.

(iii) For a set $\mathcal{S}$ of n pseudodisks, the total complexity of faces of depth at most k in their arrangement $\mathcal{A} = \mathcal{A}(\mathcal{S})$ is $O(nk)$ [CS89, Sha03, Sha91].

(iv) Using a randomized incremental construction, one can compute all faces at depth at most k in $\mathcal{A}$ in time $O(nk + n \log n)$ [BY98]. (In fact, if we are interested in "depth thresholding" rather than construction of a subset of $\mathcal{A}$, this can be done with a standard plane sweep in $O(nk \log n)$ deterministic time [dBvKOS00].)

Theorem 4.6 *Given a set $\mathcal{S}$ of n pseudodisks, one can compute a point q of depth at least $(1 - \varepsilon)\,\mathrm{depth}(\mathcal{S})$, in $O(n\varepsilon^{-2} \log n)$ expected time. The output is correct with high probability.*

Proof: Given any set X of m pseudodisks, we can compute the deepest point in $\mathcal{A}(X)$, in $T(m, \delta) = O(m\delta + m \log m)$ time, using the algorithm of fact (iv) above, where $\delta := \mathrm{depth}(X)$. The theorem now follows from Theorem 3.3. ■

4.3 Finding a Ball with Maximum Number of Red Points

We consider the following two problems.

Problem 4.7 Given two sets of red and blue points in $\mathbb{R}^d$, denoted by R and B, respectively, with a total of n points, find a ball that contains the maximum number of red points, while not containing any blue point in its interior.

Problem 4.8 Given two sets of red and blue hyperplanes in $\mathbb{R}^d$, denoted by R and B, respectively, of total size n, find a point lying below (or on) all blue hyperplanes and below (or on) the maximum number of red hyperplanes.

The second problem appears more abstract and less natural. However, it is more general, as the former reduces to the latter, albeit in one higher dimension. Indeed, if one applies the standard "lifting transformation" [Ede87] which maps balls in $\mathbb{R}^d$ to hyperplanes in $\mathbb{R}^{d+1}$ and points in $\mathbb{R}^d$ to points on the standard paraboloid in $\mathbb{R}^{d+1}$ in such a manner that a point lies within a ball if and only if the corresponding lifted point lies below the corresponding hyperplane, an instance of Problem 4.7 is transformed into that of Problem 4.8. More precisely, we lifted the two points sets by one dimension, such that now we look for a hyperplane lying below all the blue points, and that has as many red points below it as possible. Now, using point/hyperplane duality that preserves above/below relationship between points and hyperplanes, we get Problem 4.8.

We now explain how to solve Problem 4.8 efficiently for $d = 3$ (which corresponds to solving Problem 4.7 for disks in the plane), both exactly and approximately, and later discuss higher-dimensional extensions.

Theorem 4.9 *Given sets R and B of red and blue planes in $\mathbb{R}^3$, respectively, with a total of n planes, one can compute a point that lies below (or on) all blue planes and above (or on) the maximum number k_{opt} of red planes in time $O(n^2 \log n)$.*

Given in addition a parameter $\varepsilon > 0$, one can compute a point with the property that it lies below (or on) all blue planes and above (or on) $(1-\varepsilon)k_{\text{opt}}$ red planes, in expected time $O(n\varepsilon^{-2}\log^2 n)$.

Proof: We reduce the problem to that of depth in pseudodisk arrangement. Let $\mathcal{B}$ be the (convex) set of points lying below (or on) all of the blue planes. Without loss of generality, we can restrict our attention to points on $\partial\mathcal{B}$. A point $p \in \partial\mathcal{B}$ lies on or above a plane $\pi \in R$ if and only if its projection p' to the xy-plane lies in the projection of $\pi \cap \mathcal{B}$, which is a convex set in the plane; we denote it by C_π.

We claim that $\{C_\pi \mid \pi \in R\}$ is a family of (convex, and therefore x-monotone) pseudodisks. Indeed, the intersection points of ∂C_π and $\partial C_{\pi'}$ are the projections of the points in $\pi \cap \pi' \cap \partial\mathcal{B}$, which is the intersection of a convex surface with a straight line.

At this point, we observe that $\mathcal{B}$ can be computed in time $O(n \log n)$ time (for example, by computing the convex hull of the dual points, see [Ede87]) and preprocessed into the Dobkin-Kirkpatrick hierarchy [DK90], so that C_π can be manipulated implicitly. More specifically, it is easily checked that the following operations can be performed on the resulting pseudodisks in logarithmic time: Computation of boundary intersection points of two pseudodisks, computation of x-extreme points of a pseudodisk, computation of points of intersection of a pseudodisk with a y-vertical line. Armed with these operations, we can compute the entire arrangement and thus its depth in time $O(n^2 \log n)$ as in fact (ii) above. Similarly, Theorem 3.3 together with the implicit representation yield the desired approximation algorithm. ∎

We now turn to the higher-dimensional version of the problem. Recall that we aim to compute a point above a maximum number of red hyperplanes, while lying below all blue hyperplanes. Checking if such a point of depth k exists can be performed as follows: Compute the $(\leq k)$-levels of the arrangement $\mathcal{A}(R \cup B)$ [AES99, Mul91]. Traverse the 1-skeleton of the resulting structure, clipping away the portions of those levels that lie above the lower envelope of B and record the highest-depth unclipped vertex. This takes time proportional to the time spent on computing the $(\leq k)$-levels of $\mathcal{A}(R \cup B)$, which is $T(n,k) = O\big(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil}\big)$ [AES99, Mul91]. Using this observation to implement depth thresholding in the algorithm of Theorem 4.6, we obtain the following:

Theorem 4.10 *Given sets R and B of red and blue hyperplanes in $\mathbb{R}^d$, for $d \geq 4$, $|R| + |B| = n$, and a parameter $\varepsilon > 0$, one can compute a point that lies below (or on) all blue hyperplanes and above (or on) at least $(1-\varepsilon)k_{\text{opt}}$ hyperplanes, where k_{opt} is the maximum number achievable. The running time of the algorithm is $O(n^{\lfloor d/2 \rfloor}(\varepsilon^{-2}\log n)^{\lceil d/2 \rceil})$ with high probability.*

Corollary 4.11 *Given sets R and B of red and blue points in $\mathbb{R}^d$, for $d \geq 3$, $|R| + |B| = n$, and a parameter $\varepsilon > 0$, one can compute a ball $\mathcal{B}$ that contains $(1-\varepsilon)k_{\mathrm{opt}}$ red points, while avoiding all blue points, where k_{opt} is the maximum number of red points that can be covered by such a ball. The running time of the algorithm is $O(n^{\lceil d/2 \rceil}(\varepsilon^{-2}\log n)^{\lceil (d+1)/2 \rceil})$.*

Can the algorithm running time in Theorem 4.10 be further improved? We leave this as an open problem for further research.

5 From Emptiness to Approximate Range Counting

Assume that there exists an *emptiness testing* data structure that can be constructed in $T(n)$ time for a set $\mathcal{S}$ of n objects such that, given a query range τ, we can check in $Q(n)$ time whether τ intersects any of the objects in $\mathcal{S}$. Let $S(n)$ be the space required to store this data structure. In this section, we show how to build a data structure that quickly returns an approximate number of objects in $\mathcal{S}$ intersecting τ using emptiness testing as a subroutine.

In particular, let μ_τ denote the number of objects of $\mathcal{S}$ intersected by τ. Below we use $\varepsilon > 0$ to denote the required approximation quality; namely, we would like the data structure to output a number α_τ such that $(1-\varepsilon)\mu_\tau \leq \alpha_\tau \leq \mu_\tau$.

5.1 The decision procedure

Given parameters $z \in [1, n]$ and ε, with $1/2 > \varepsilon > 0$, we construct a data structure, such that, for any δ, with $1/2 > \delta \geq \varepsilon$, and a query range τ, we can decide, with high probability, whether $\mu_\tau < z$ or $\mu_\tau \geq z$. The twist that makes it efficiently solvable is that the data structure is allowed to make a mistake if $\mu_\tau \in [(1-\delta)z, (1+\delta)z]$.

The data structure. Let $\mathcal{R}_1, \ldots, \mathcal{R}_M$ be M independent random samples of $\mathcal{S}$, formed by picking every element with probability $1/z$, where $M := N(\varepsilon)$,

$$N(\varepsilon) := \left\lceil c_3 \varepsilon^{-2} \log n \right\rceil,$$

and c_3 is a sufficiently large absolute constant. Build M separate emptiness-query data structures $D_1, \ldots, D_M$, for the sets $\mathcal{R}_1, \ldots, \mathcal{R}_M$, respectively, and put $\mathcal{D} = \mathcal{D}(z, \varepsilon) := \{D_1, \ldots, D_M\}$.

Answering a query. Consider a query range τ and approximation quality δ, $1/2 > \delta \geq \varepsilon$, and let $X_i = 1$ if τ intersects any of the objects of R_i and $X_i = 0$ otherwise, for $i = 1, \ldots, N = N(\delta)$; note that we use N rather than all M of the samples R_i in answering the query. The value of X_i can be determined using a single emptiness query in D_i. Compute $Y_\tau := \sum_{1 \leq i \leq N} X_i$.

For a range σ of depth k, the probability that σ does not avoid all the elements of $\mathcal{R}_i$ is

$$\rho(k) := 1 - \left(1 - \frac{1}{z}\right)^k.$$

If a range σ has depth z, $\mathbf{E}[Y_\sigma] = N\rho(z) =: \Delta$. Our data structure returns "depth$(\tau, \mathcal{S}) < z$" if $Y_\tau < \Delta$, and "depth$(\tau, \mathcal{S}) \geq z$" otherwise.

Correctness. In the following, we show that with high probability the data structure indeed returns the correct answer if the depth of the query range is outside the "uncertainty" range $[(1-\delta)z, (1+\delta)z]$. For simplicity of exposition, we assume in the following that $z \geq 10$ (the case $z < 10$ follows by similar arguments). Consider a range τ of depth at most $(1-\delta)z$. The data structure returns wrong answer if $Y_\tau > \Delta$. We will show that the probability of this event is polynomially small. The other case, where τ has depth at least $(1+\delta)z$ but $Y_\tau < \Delta$ is handled in a similar fashion.

Lemma 5.1 *The probability* $\mathbf{Pr}[Y_\tau > \Delta \mid \text{depth}(\tau, \mathcal{S}) \leq (1-\delta)z]$ *does not exceed* n^{-c_6}*, where* $c_6 = c_6(c_3) > 0$ *depends only on* c_3 *and can be made arbitrarily large by a choice of a sufficiently large* $c_3 > 0$.

The proof of this lemma follows from the Chernoff inequality. It is tedious and not very insightful; thus it is delegated to Appendix A. This implies the following lemma.

Lemma 5.2 *Given a set* $\mathcal{S}$ *of* n *objects, a parameter* $0 < \varepsilon < 1/2$*, and* $z \in [0, n]$*, one can construct a data structure* $\mathcal{D}(z)$ *which, given a range* τ *and a parameter* $1/2 > \delta \geq \varepsilon$*, returns either* LOW *or* HIGH*. If it returns* LOW*, then* $\mu_\tau \leq (1+\delta)z$*, and if it returns* HIGH *then* $\mu_\tau \geq (1-\delta)z$*. The data structure might return either answer if* $\mu_\tau \in [(1-\delta)z, (1+\delta)z]$.

The data structure $\mathcal{D}$ *consists of* $M = O(\varepsilon^{-2} \log n)$ *emptiness data structures. The space and time needed are* $O(S(2n/z)\varepsilon^{-2} \log n)$ *and* $O(Q(2n/z)\delta^{-2} \log n)$*, where* $S(m)$ *and* $Q(m)$ *are the space needed for a single emptiness data structure storing* m *objects and the time needed for a single query in such a structure, respectively. All bounds hold with high probability.*

Proof: The lemma follows immediately from the above discussion. The only missing part is observing that, by the Chernoff inequality, $|\mathcal{R}_i| \leq 2n/z$, with high probability. ■

The path to answering approximate counting query is now clear. We use Lemma 5.2 to perform binary search, repeatedly narrowing the range containing the answer. We stop when the size of the range is within our error tolerances. At the start of the process, this range is large, so we use large values of δ. As the range narrows δ is reduced. (Recall that δ determines the query cost in Lemma 5.2 by controlling the number of samples whose range emptiness data structures are consulted to answer the query.)

One difficulty is that Lemma 5.2 works only for $\delta < 1/2$. We next strengthen it considerably for much larger values of δ. Intuitively, then number of "experiments" (i.e., emptiness queries) we need to perform is only $\Theta((\log n)/\log \delta)$ to answer a LOW/HIGH query and have a high probability guarantee.

Lemma 5.3 *Given the data structure of Lemma 5.2,* z *a value to compare to, and* $\delta > c_5$ *one can decide for a query range* τ *if* $\mu_\tau < z/(1+\delta)$ *or* $\mu_\tau \geq z(1+\delta)$*. Here* c_5 *is a sufficiently large constant. The data structure is allowed to return any answer if* $\mu_\tau \in [z/(1+\delta), (1+\delta)z]$ *This requires* $N = \lceil c_6 (\log n)/\ln \delta \rceil$ *emptiness queries, and the answer returned is correct with high probability, where* c_6 *is an appropriate absolute constant.*

The proof of this lemma is a tedious but careful application of Chernoff inequality, deferred to Appendix B.

Remark 5.4 Note, that there is a gap between the ranges of δ to which Lemma 5.2 and Lemma 5.3 are applicable, namely when δ is between $1/2$ and c_5. If we need to apply either lemma in such a situation, we use Lemma 5.2 with $\delta = 1/4$ a constant number of times.

5.2 The data structure

Lemma 5.2 and Lemma 5.3 provide us with a tool for performing a "binary" search for the count value μ_τ of a range τ. For small values of i, we just build a separate data structure $\mathcal{D}_i = \mathcal{D}(v_i, \varepsilon_i)$ of Lemma 5.3 for depth values $v_i = i/2$, for $i = 1, \ldots, U = O(\varepsilon^{-1})$. For depth i, we use accuracy

$$\varepsilon_i = 1/8i \tag{5.1}$$

(i.e., this is the value of ε when using Lemma 5.2). Using these data structures, we can decide whether the query range count is at least U, or smaller than U. If it is smaller than U, then we can perform a binary search to find its exact value. The result is correct with high probability.

Next, consider the values $v_j = (U/4)(1+\varepsilon/16)^j$, for $j = U+1, \ldots, W$, where $W := c \log_{1+\varepsilon/16} n = O(\varepsilon^{-1} \log n)$, for an appropriate choice of an absolute constant $c > 0$, so that $v_W = n$. We build a data structure $\mathcal{D}_j = \mathcal{D}(v_j, \varepsilon/16)$ for each $z = v_j$, using Lemma 5.2.

In the end of this process, we have built $\mathcal{D}_1, \ldots, \mathcal{D}_W$.

Answering a query. Given a range query τ, each data structure in our list returns LOW or HIGH. Moreover, with high probability, if we were to query all the data structures, we would get a sequence of HIGHs, followed by a sequence of LOWs. It is easy to verify that the value associated with the last data structure returning HIGH (rounded to the nearest integer) yields the required approximation. We can use binary search on $\mathcal{D}_1, \ldots, \mathcal{D}_W$ to locate this changeover value using a total of $O(\log W) = O(\log(\varepsilon^{-1} \log n))$ queries in the structures of $\mathcal{D}_1, \ldots, \mathcal{D}_W$. Namely, the overall query time is $O(Q(n)\varepsilon^{-2}(\log n)\log(\varepsilon^{-1}\log n))$.

Answering queries more efficiently. To speed up the query, we organize the search into $O(\log(n/\varepsilon))$ rounds. Treat $\mathcal{D}_1, \ldots, \mathcal{D}_W$ as a linked list L_M, where $M = \lceil \lg W \rceil = O(\log(\varepsilon^{-1} \log n))$. Next, we build a data structure (which is somewhat similar to a skip-list), where L_{i-1} is formed from L_i by picking every other element of L_i, such that the base list L_1 has, say, 4 to 8 elements.

The search now continues in a top-down fashion starting from the list L_1. At the ith stage, we maintain pointers to four consecutive data structures in L_i, such that the left two return HIGH and the right two return LOW. Thus the exact answer μ_τ must lie between the depth associated with the first and the fourth of these data structures in L_i. The corresponding portion of L_{i+1} delimited by these two data structures in L_i is a sublist of (at most) seven data structures in L_{i+1}. We now query the at most three new data structures in the list (i.e., the ones we have not queried on previous rounds) using Lemma 5.2 and Lemma 5.3 to determine the sublist of four consecutive data structures whose range contain the query depth. The key observation, leading to a more efficient query time, is that we answer the queries, on the ith level, using an error parameter δ_i which is as large as possible, such that the error intervals of all these data structures are disjoint.

We continue in this process until we reach the bottom level. Clearly, we have the required approximation and we return the value associated with, say, the leftmost data structure in this sublist as the approximation.

We can set $\delta_1 := n^{1/4}$ and $(1+\delta_i/c)^2 = 1 + \delta_{i-1}/c$, where $\delta_{i-1} > 1$ and c is some constant. Therefore, $\delta_i = O(\sqrt{\delta_{i-1}})$, as long as, say, $\delta_i > c^2$. To bound the number of emptiness queries used in this range, where the error is large, we use Lemma 5.3, implying that the number of emptiness queries used is

$$\sum_{i,\delta_i > c^2} O\left(\frac{\log n}{\log \delta_i}\right) = \sum_{i=O(1)}^{\lg\lg n} O\left(\frac{\log n}{\log\left(2^{2^{\lg\lg n - i}}\right)}\right) = \sum_{i=O(1)}^{\lg\lg n} O\left(\frac{\log n}{2^{\lg\lg n - i}}\right) = O(\log n).$$

Next, we bound the number of emptiness queries used when $\delta < 1/2$. In the jth level of this more refined search, we use $\delta'_j = 1/2^j$. Thus, the number of emptiness queries is

$$\sum_{j=2}^{\lg(1/\varepsilon)} O\left(\frac{\log n}{\delta'^2_j}\right) = \sum_{j=2}^{\lg(1/\varepsilon)} O\bigl(2^{2j}\log n\bigr) = O(\varepsilon^{-2}\log n),$$

by Lemma 5.2. The number of queries with δ between $1/2$ and c_5 is constant and therefore does not affect our asymptotic analysis, see Remark 5.4.

Lemma 5.5 *Given a set $\mathcal{S}$ of n objects and a data structure that answers emptiness queries in $Q(n)$ time and can be built in $T(n)$ time, one can construct in $O(T(n)\varepsilon^{-3}\log^2 n)$ time a data structure that, given a range τ, outputs a number α_τ, with $(1-\varepsilon)\mu_\tau \le \alpha_\tau \le \mu_\tau$, where μ_τ is the number of objects in $\mathcal{S}$ intersecting τ. The query time is*

$$O\bigl(\varepsilon^{-2}Q(n)\log n\bigr)\,.$$

The output of the query is correct with high probability for all queries and the running time bounds hold with high probability.

(In the above lemma, we assumed the number of queries is polynomial.)

Space and time requirements Assume that the emptiness data structure uses $S(n)$ space to store n elements, and it can be constructed in $T(n)$ preprocessing time.

We need a technical definition: If a data structure D can be constructed for n elements using $S(n)$ space in $T(n)$ time, such that $S(n/i) = O(S(n)/i^\lambda)$ and $T(n/i) = O(T(n)/i^\lambda)$, for all n and $1 \le i \le n$, for a constant $\lambda \ge 1$, then the data structure has *degree* λ. (The O notation might hide constants that depend on λ.)

The space requirements to implement the data structure of Lemma 5.5 for the low range $[1, O(1/\varepsilon)]$ can be reduced by using precision $\delta := \varepsilon_i = 1/8i$ (see Eq. (5.1)) when considering numbers in the interval $I_i = [i/4, (i+1)/4]$ which contains only a single integer. Therefore, the data structure returns the exact answer in this case. The space needed is

$$\begin{aligned} S_1 &= \sum_{i=1}^{O(1/\varepsilon)} O\bigl(i^2 S(n/8i)\log n\bigr) = \sum_{i=1}^{O(1/\varepsilon)} O\Bigl(i^2\frac{S(n)}{i^\lambda}\log n\Bigr) = O\Bigl(S(n)\log n\sum_{i=1}^{O(1/\varepsilon)}\frac{1}{i^{\lambda-2}}\Bigr)\\ &= O(H_{\lambda-2}(1/\varepsilon)S(n)\log n), \end{aligned}$$

where $H_d(k) = \sum_{i=1}^k 1/i^d$. For the intervals $I_{U+1},\ldots,W$, the space is at most proportional to

$$\begin{aligned} \sum_{i=1}^{W-U}\frac{S(n/(\varepsilon^{-1}(1+\varepsilon/16)^i))}{\varepsilon^2}\log n &\le \sum_{i=1}^{W-U}\frac{(\varepsilon/(1+\varepsilon/16)^i)^\lambda}{\varepsilon^2}O(S(n)\log n)\\ &\le \sum_{i=1}^{W-U}\frac{O(S(n)\log n)}{\varepsilon^{2-\lambda}(1+\varepsilon/16)^{i\lambda}}\\ &\le \frac{O(S(n)\log n)}{\varepsilon^{2-\lambda}}\sum_{i=1}^{W-U}\frac{1}{(1+\varepsilon/16)^{i\lambda}}\\ &\le \frac{O(S(n)\log n)}{\varepsilon^{2-\lambda}}\sum_{i=1}^{\infty}\frac{1}{(1+\varepsilon)^i} = \frac{O(S(n)\log n)}{\varepsilon^{3-\lambda}}. \end{aligned}$$

Thus, the overall space required is $O((\varepsilon^{\lambda-3} + H_{\lambda-2}(1/\varepsilon))S(n)\log n)$. A similar bound holds on the preprocessing time.

The result Putting everything together, we have the following result.

Theorem 5.6 *We are given a set of $\mathcal{S}$ of n objects. Assume that one can construct, in $T(n)$ time, a data structure of degree λ that answers emptiness queries in $Q(n)$ time using $S(n)$ space. Then one can construct a data structure in $O((\varepsilon^{\lambda-3}+H_{\lambda-2}(1/\varepsilon))T(n)\log n)$ time, using $O((\varepsilon^{\lambda-3}+H_{\lambda-2}(1/\varepsilon))S(n)\log n)$ space, such that, given a range τ, it outputs a number α_τ, such that $(1-\varepsilon)\mu_\tau \leq \alpha_\tau \leq \mu_\tau$, and the query time is $O(\varepsilon^{-2}Q(n)\log n)$, where $H_d(k) := \sum_{i=1}^{k} 1/i^d$ and μ_τ is the exact answer to the counting query. The result returned is correct with high probability for all queries.*

(We assumed in the above theorem that the number of queries is polynomial.)

The number of emptiness queries used by Theorem 5.6 is probably tight. Indeed, just deciding if the query depth lies in the range $[z/(1+\varepsilon), z(1+\varepsilon)]$ requires $O(\varepsilon^{-2}\log n)$ emptiness queries by the Chernoff inequality. Since the Chernoff inequality is tight it seems unlikely that the number of emptiness queries can be improved. Notice that this argument only applies to the case where emptiness queries are treated as a "black box" operations. If this assumption is removed, certain improvements are possible [AHS07].

5.3 Applications

5.3.1 Halfplane and halfspace range counting

Using the data structure of Dobkin and Kirkpatrick [DK85], one can answer emptiness halfspace range searching queries in logarithmic time, when $d = 2, 3$. In this case, we have $S(n) = O(n)$, $T(n) = O(n\log n)$, $Q(n) = O(\log n)$, and $\lambda = 1$.

Corollary 5.7 *Given a set P of n points in two (resp., three) dimensions, and a parameter $\varepsilon > 0$, one can construct in expected $O(n\varepsilon^{-2}\log^2 n)$ time a data structure of size $O(n\varepsilon^{-2}\log n)$, such that, given a halfplane (resp. halfspace) τ, it outputs a number α, such that $(1-\varepsilon)\left|\tau \cap P\right| \leq \alpha \leq \left|\tau \cap P\right|$, and the query time is $O\big(\varepsilon^{-2}\log^2 n\big)$. The result returned is correct with high probability for all queries.*

Using the standard lifting of points in $\mathbb{R}^2$ to the paraboloid in $\mathbb{R}^3$ implies a similar result for approximate range counting for disks, as a disk range query in the plane reduces to a halfspace range query in three dimensions.

Corollary 5.8 *Given a set of P of n points in two dimensions, and a parameter ε, one can construct a data structure in $O(n\varepsilon^{-2}\log^2 n)$ expected time, using $O(n\varepsilon^{-2}\log n)$ space, such that given a disk τ, it outputs a number α, such that $(1-\varepsilon)\left|\tau \cap P\right| \leq \alpha \leq \left|\tau \cap P\right|$, and the query time is $O\big(\varepsilon^{-2}\log^2 n\big)$. The result returned is correct with high probability for all possible queries.*

These algorithms are faster by a polynomial factor in n than previously known exact procedures for answering the corresponding questions.

Notice, that our techniques also apply to the higher dimensions settings in an obvious way. Since the resulting bounds are somewhat less appealing, we refrain from stating them explicitly.

5.3.2 Depth queries

By computing the union of a set of n pseudodisks in the plane, and preprocessing the union for point-location queries, one can perform "emptiness" queries in logarithmic time. This is done by computing the union [EHS04], and then performing a point-location queries in it [dBvKOS00]. (Again, we are assuming here that we can implement the geometric primitives on the pseudodisks in constant time.) The space needed is $O(n)$ and it takes $O(n \log n)$ time to construct the data structure. Thus, we get the following result.

Corollary 5.9 *A set of $\mathcal{S}$ of n pseudodisks in the plane can preprocessed in expected $O(n\varepsilon^{-2} \log^2 n)$ time, using $O(n\varepsilon^{-2} \log n)$ space, such that given a query point q, one can output a number α, such that $(1-\varepsilon)\,\mathrm{depth}(p, \mathcal{S}) \le \alpha \le \mathrm{depth}(p, \mathcal{S})$, and the query time is $O\big(\varepsilon^{-2} \log^2 n\big)$. The result returned is correct with high probability for all possible queries.*

6 Conclusions

In this paper, we have shown the connection between depth thresholding, emptiness, and approximate depth computation. Note, that Theorem 5.6 is better than the result in the conference version of this paper [AH05].

There are several natural problems for further research:

- Computing a square containing largest number of red points, while avoiding the blue points. Of course, the same question can be asked for other families of shapes, such as rectangles, ellipses, translates, or homothets of a fixed shape, etc.

- Proving a lower bound on the problem of computing the ball containing the largest number of red points while avoiding the blue points. In particular, is this problem 3SUM-hard in two dimensions?

We would also like to point out that the approximate depth computation in the plane does not require "pseudodisk-ness." Indeed, it is sufficient to have a family of objects with small complexity of the union, such as the one described by [Efr05]. An analogous statement holds in three dimensions, though there are few known classes of objects with small union complexity there for which the computation of the union can be performed efficiently.

6.1 Permutations, emptiness, and randomized incremental construction

In a followup to this work, Haim Kaplan and Micha Sharir [KS06] showed that one can use the "permutations technique" of Edith Cohen [Coh97] to approximate the depth of a query range. The idea is to randomly permute the set of objects $\mathcal{S}$ and compute the first index in the permutation whose corresponding object in $\mathcal{S}$ meets the query range. If the index is i, the estimate of the depth is n/i. Repeating this process a sufficient number of times implies, by the Chernoff inequality, that the estimate is within acceptable error range. To find the lowest index i where this happens, one can perform the query inside a history graph of a randomized incremental construction algorithm for computing the union of the objects. This yields polylogarithmic improvement over the proceedings version of this paper for some specific applications (but no improvement over the current version). See [KS06] for details.

We observe that up to polylogarithmic factors, the two techniques are equivalent. Indeed, it is sufficient to approximate the first index in the permutation that contains the query range up to a factor of $1 + \varepsilon$, which can be reduced to a binary search over emptiness data structures.

The other direction is somewhat more interesting. Our data structure has $O(\varepsilon^{-1}\log n)$ data structures at different resolutions, each comprised of $O(\varepsilon^{-2}\log n)$ emptiness data structures built over random samples. If we were to use instead of independent random subsets of the input in our data structures, a gradation (we remind the reader that a gradation is a sequence of random samples, where the ith sample is a random sample of the higher $(i-1)$th-level sample) then the sequence of queries performed along the emptiness data structures associated with this gradation is essentially equivalent to the permutation query. Thus, in some intuitive, rough, and very informal sense, the data structure of Kaplan and Sharir performs the "same" computations as our data structure. The "sameness" here should be taken with several sizeable grains of salt as everything looks similar from afar. It is interesting and surprising that these two different approaches to the problem are so similar once one inspects the underlying machinery.

Acknowledgments

The authors would like to thank Jeff Erickson and Edgar Ramos for useful discussion on the problems studied in this paper. Haim Kaplan and Micha Sharir have pointed out that the space requirement of depth queries, as described in an earlier version of this paper [AH05] can be improved. We then also improved the query time in Theorem 5.6.

The authors would also like to thank the anonymous referees for their detailed and insightful comments that lead to hopefully significant improvements in the presentation of this paper.

References

[ABSS97] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.

[AE98] 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.* AMS Press, Providence, RI, 1998.

[AES99] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29:912–953, 1999.

[AH05] B. Aronov and S. Har-Peled. On approximating the depth and related problems. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, pages 886–894, 2005.

[AHR$^+$02] P. K. Agarwal, T. Hagerup, R. Ray, M. Sharir, M. H. M. Smid, and E. Welzl. Translating a planar object to maximize point containment. In *Proc. 10th Annu. European Sympos. Algorithms*, pages 42–53, 2002.

[AHS07] B. Aronov, S. Har-Peled, and M. Sharir. On approximate halfspace range counting and relative epsilon-approximations. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 327–336, 2007.

[AK95] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theo. Comp. Sci.*, 147(1–2):181–210, 1995.

[AM00] S. Arya and D. M. Mount. Approximate range searching. *Comput. Geom. Theory Appl.*, 17:135–152, 2000.

[BCP93] H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching? *Discrete Comput. Geom.*, 10:143–155, 1993.

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998.

[Cha00] T. M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000.

[Cha05] T. M. Chan. Low-dimensional linear programming with violations. *SIAM J. Comput.*, pages 879–893, 2005.

[Coh97] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Sys. Sci.*, 55(3):441–453, 1997.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[dBvKOS00] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2nd edition, 2001.

[DK85] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6:381–392, 1985.

[DK90] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Internat. Colloq. Automata Lang. Program.*, volume 443 of *Lect. Notes in Comp. Sci.*, pages 400–413. Springer-Verlag, 1990.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Heidelberg, 1987.

[Efr05] A. Efrat. The complexity of the union of (α, β)-covered objects. *SICOMP*, 34(4):775–787, 2005.

[EHS04] E. Ezra, D. Halperin, and M. Sharir. Speeding up the incremental construction of the union of geometric objects in practice. *Comput. Geom. Theory Appl.*, 27(1):63–85, 2004.

[GO95] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

[HI00] S. Har-Peled and P. Indyk. When crossings count - approximating the minimum spanning tree. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 166–175, 2000.

[HW04] S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33(2):269–285, 2004.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 604–613, 1998.

[Ind00] P. Indyk. *High-Dimensional Computational Geometry*. PhD thesis, Stanford, 2000.

[KLPS86] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[KS06] H. Kaplan and M. Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar voronoi diagrams, and approximate range counting. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 484–493, 2006.

[Mat91] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

[Mat92] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.

[Mat93] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.

[Mat95] J. Matoušek. On geometric optimization with few violated constraints. *Discrete Comput. Geom.*, 14:365–384, 1995.

[MMP$^+$94] J. Matoušek, N. Miller, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *SIAM J. Comput.*, 23(1):154–169, 1994.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[Mul91] K. Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete Comput. Geom.*, 6:307–338, 1991.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

[Sha91] M. Sharir. On k-sets in arrangements of curves and surfaces. *Discrete Comput. Geom.*, 6:593–613, 1991.

[Sha03] M. Sharir. The Clarkson-Shor technique revisited and extended. *Comb., Prob. & Comput.*, 12(2):191–201, 2003.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

Appendix A: Proof of Lemma 5.1

Proof: The probability $\alpha := \mathbf{Pr}[Y_\tau > \Delta \mid \mathrm{depth}(\tau, \mathcal{S}) \le (1-\delta)z]$ is maximized when $\mathrm{depth}(\tau, \mathcal{S}) = (1-\delta)z$. Thus

$$\alpha \le \beta := \mathbf{Pr}[Y_\tau > \Delta \mid \mathrm{depth}(\tau, \mathcal{S}) = (1-\delta)z]\,.$$

We argue that this probability is polynomially small.

Observe that $\mathbf{Pr}[X_i = 1] = \rho\big((1-\delta)z\big)$, where $\rho(k) := 1 - (1-1/z)^k$. Therefore, we have

$$\mathbf{E}[Y_\tau] = N\rho((1-\delta)z) = N \cdot \left(1 - \left(1 - \frac{1}{z}\right)^{(1-\delta)z}\right) \ge N \cdot \left(1 - e^{-(1-\delta)}\right) \ge \frac{N}{3}, \tag{.1}$$

since $1 - 1/z \le \exp(-1/z)$. The last inequality holds since $\delta \le 1/2$. By definition, $\Delta = N\rho(z)$, therefore

$$\xi := \frac{\Delta}{\mathbf{E}[Y_\tau]} = \frac{1 - \left(1-\frac{1}{z}\right)^z}{1 - \left(1-\frac{1}{z}\right)^{(1-\delta)z}} = 1 + \frac{\left(1-\frac{1}{z}\right)^{(1-\delta)z} - \left(1-\frac{1}{z}\right)^z}{1 - \left(1-\frac{1}{z}\right)^{(1-\delta)z}}$$
$$= 1 + \left(1 - \frac{1}{z}\right)^{(1-\delta)z} \frac{1 - \left(1-\frac{1}{z}\right)^{\delta z}}{1 - \left(1-\frac{1}{z}\right)^{(1-\delta)z}}.$$

Since $(1-1/z)/e \le (1-1/z)^z \le 1/e$ [MR95], $(1-1/z)^{(1-\delta)z} \ge \left((1-1/z)/e\right)^{1-\delta} \ge (1-1/z)/e$. This implies that

$$\xi \ge 1 + \frac{1}{e}\left(1 - \frac{1}{z}\right) \frac{1 - \exp(-\delta)}{1 - \left(1 - \frac{1}{z}\right)\frac{1}{e}}$$
$$\ge 1 + \frac{1}{2} \cdot \frac{1}{e} \cdot \frac{1}{1/2} \cdot (1 - \exp(-\delta))\,,$$

since $(1-1/x)^x \le 1/e$ (for $x \ge 1$) [MR95] and $z \ge 10$. Observe that $\exp(-\delta) \le 1 - \delta + \delta^2/2$, using the Taylor expansion of $e^{-\delta}$, for $\delta > 0$. Therefore

$$\xi \ge 1 + \frac{1}{e}\left(\delta - \delta^2/2\right) \ge 1 + \frac{\delta}{6}.$$

Deploying Chernoff inequality, we have that if $\mathrm{depth}(\tau, \mathcal{S}) = (1-\delta)z$ then

$$\beta = \mathbf{Pr}[Y_\tau > \Delta] \le \mathbf{Pr}\big[Y_\tau > \xi\,\mathbf{E}[Y_\tau]\,\big] \le \mathbf{Pr}\big[Y_\tau > (1+\delta/6)\,\mathbf{E}[Y_\tau]\,\big]$$
$$\le \exp\!\big(-\mathbf{E}[Y_\tau]\,(\delta/6)^2/4\big) \le \exp\left(-\frac{N\delta^2}{3 \cdot 36 \cdot 4}\right) \le \exp\left(-\frac{\delta^2 \left\lceil c_3 \delta^{-2} \log n \right\rceil}{432}\right) \le n^{-c_3/432}$$

by Eq. (2.1) (a) and Eq. (.1). ■

Appendix B: Proof of Lemma 5.3

Proof: Recall that $Y_\tau = \sum_{i=1}^N X_i$. Again, we can assume that $\mathrm{depth}(\tau, \mathcal{S}) = z/(1+\delta)$. We want to bound the probability that this "light" range would be considered to be "heavy" (i.e., $Y_r > N\rho(z)$). We have that

$$\mathbf{E}[Y_\tau] = N\left(1 - \left(1 - \frac{1}{z}\right)^{z/(1+\delta)}\right) \le N\left(1 - \exp\left(-\frac{2z/(1+\delta)}{z}\right)\right) \le N\frac{2}{1+\delta}, \tag{.2}$$

since $1 - 1/z \geq \exp(-2/z)$ and $1 - x \leq \exp(-x)$, for $0 \leq x \leq 1/2$. By similar reasoning,

$$\mathbf{E}[Y_\tau] \geq N\left(1 - \exp\left(-\frac{z/(1+\delta)}{z}\right)\right) \geq \frac{N}{2(1+\delta)}. \tag{.3}$$

Observe that

$$\lim_{z\to\infty} \rho(z) = \lim_{n\to\infty}\left(1 - \left(1 - \frac{1}{z}\right)^z\right) = 1 - \frac{1}{e}.$$

Thus $\rho(z) > 1/2$ since z is sufficiently large (we remind the reader that we assumed that $z \geq 10$ for the sake of simplicity of exposition). Thus

$$\zeta = \mathbf{Pr}[Y_\tau > N\rho(z)] \leq \mathbf{Pr}[Y_\tau > N/2] \leq \mathbf{Pr}\left[Y_\tau > \frac{1+\delta}{4}\,\mathbf{E}[Y_\tau]\right]$$
$$\leq \mathbf{Pr}\left[Y_\tau > \left(1 + \frac{\delta}{5}\right)\mathbf{E}[Y_\tau]\right]$$

by Eq. (.2) and since δ is sufficiently large. Observe that since $\delta \geq c_5$, we have that

$$\frac{e^{\delta/5}}{(1+\delta/5)^{(1+\delta/5)/3}} \leq \frac{1}{e}.$$

Now, by the Chernoff inequality, we have

$$\zeta < \left(\frac{e^{\delta/5}}{(1+\delta/5)^{1+\delta/5}}\right)^{\mathbf{E}[Y_\tau]} \leq \left((1+\delta/5)^{-(1+\delta/5)/2}\right)^{\mathbf{E}[Y_\tau]},$$

since $\delta \geq c_5$. Observe that

$$\frac{(1+\delta/5)\,\mathbf{E}[Y_\tau]}{2} \geq \frac{(1+\delta/5)}{4(1+\delta)}N \geq \frac{1}{20}N,$$

by Eq. (.3). As $\ln(1+\delta/5) \geq (\ln\delta)/2$, we have

$$\zeta \leq (1+\delta/5)^{-N/20} \leq \exp\left(-\frac{N}{40}\ln\delta\right) < \frac{1}{n^{\Omega(1)}},$$

for c_6 sufficiently large, since $N = \lceil c_6(\log n)/\ln\delta\rceil$.

The other direction is slightly more challenging. Assume that $\mathrm{depth}(\tau, \mathcal{S}) = z(1+\delta)$, and let $Z_\tau = N - Y_\tau$. Observe that if $z/(1+\delta) \leq 1$, then any answer returned by the algorithm would be valid in this case (here we are trying to bound the case that a heavy range is reported as being "light"). Therefore, we assume that $z \geq 1+\delta$. Observe that

$$\mathbf{E}[Z_\tau] = N\left(1 - \frac{1}{z}\right)^{z(1+\delta)} \leq N\exp(-(1+\delta)).$$

(Specifically, $\mathbf{E}[Z_\tau] \leq N/10^6$ since $\delta \geq c_5$, and thus $\mathbf{E}[Y_\tau]$ is very close to N in this case.) Now, since z is large enough, we have that $\rho(z) < 1 - 1/10$ as $\rho(z) \approx 1 - 1/e$. Therefore

$$\mathbf{Pr}\Big[Y_\tau < N\rho(z)\Big] \leq \mathbf{Pr}\left[Y_\tau < \frac{9}{10}N\right] = \mathbf{Pr}\left[N - Z_\tau < \frac{9}{10}N\right] = \mathbf{Pr}\left[Z_\tau > \frac{N}{10}\right]$$

$$
\begin{aligned}
&\leq \mathbf{Pr}\left[Z_\tau > \frac{\mathbf{E}[Z_\tau]}{10\exp(-(1+\delta))}\right]\\
&= \mathbf{Pr}\Big[Z_\tau > (1+\xi)\,\mathbf{E}[Z_\tau]\Big],
\end{aligned}
$$

where $\xi := (1/10)\exp(1+\delta) - 1 > 10^8$. The variable Z_τ can be interpreted as the sum of N independent 0/1 variables (i.e., its the complement of the variables forming Y_τ). By applying the Chernoff inequality to Z_τ, we obtain

$$
\varrho := \mathbf{Pr}[Y_\tau < N\rho(z)] \leq \mathbf{Pr}\Big[Z_\tau > (1+\xi)\,\mathbf{E}[Z_\tau]\Big] \leq \left(\frac{e^\xi}{(1+\xi)^{(1+\xi)}}\right)^{\mathbf{E}[Z_\tau]}.
$$

Now, since $(1-1/z)^z \geq (1/e)(1-1/z) \geq \exp(-1-2/z) \geq \exp(-(1+2/\delta))$, since $z \geq 1+\delta \geq \delta$. As such,

$$
\begin{aligned}
\mathbf{E}[Z_\tau] = N\left(1-\frac{1}{z}\right)^{z(1+\delta)} &\geq N\exp(-(1+\delta)(1+2/\delta))\\
&\geq \frac{N}{\exp(1+\delta+2/\delta+2)} \geq \frac{N}{c_7\xi},
\end{aligned}
$$

where $c_7 = 20e^3$. Thus, as $\ln(1+\xi) = \ln(1/10) + 1 + \delta \geq \ln\delta$, we have

$$
\begin{aligned}
\varrho &\leq \left((1+\xi)^{-(1+\xi)/2}\right)^{\mathbf{E}[Z_\tau]} \leq (1+\xi)^{-(1+\xi)N/c_7\xi} \leq \exp\left(-\frac{N\ln(1+\xi)}{c_7}\right)\\
&\leq \exp\left(-\frac{N}{c_7}\ln\delta\right) < \frac{1}{n^{\Omega(1)}},
\end{aligned}
$$

since $N = \lceil c_6(\log n)/\ln\delta\rceil$ and c_6 sufficiently large. ■