On Geometric Range Searching, Approximate Counting and Depth Problems

by

Peyman Afshani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2008

Abstract

In this thesis we deal with problems connected to range searching, which is one of the central areas of computational geometry. The dominant problems in this area are halfspace range searching, simplex range searching and orthogonal range searching and research into these problems has spanned decades.

For many range searching problems, the best possible data structures cannot offer fast (i.e., polylogarithmic) query times if we limit ourselves to near linear storage. Even worse, it is conjectured (and proved in some cases) that only very small improvements to these might be possible. This inefficiency has encouraged many researchers to seek alternatives through approximations. In this thesis we continue this line of research and focus on relative approximation of range counting problems.

One important problem where it is possible to achieve significant speedup through approximation is halfspace range counting in 3D. Here we continue the previous research done and obtain the first optimal data structure for approximate halfspace range counting in 3D. Our data structure has the slight advantage of being Las Vegas (the result is always correct) in contrast to the previous methods that were Monte Carlo (the correctness holds with high probability).

Another series of problems where approximation can provide us with substantial speedup comes from robust statistics. We recognize three problems here: approximate Tukey depth, regression depth and simplicial depth queries. In 2D, we obtain an optimal data structure capable of approximating the regression depth of a query hyperplane. We also offer a linear space data structure which can answer approximate Tukey depth queries efficiently in 3D. These data structures are obtained by applying our ideas for the approximate halfspace counting problem. Approximating the simplicial depth turns out to be much more difficult, however.

Computing the simplicial depth of a given point is more computationally challenging than most other definitions of data depth. In 2D we obtain the first data structure which uses near linear space and can answer approximate simplicial depth queries in polylogarithmic time. As applications of this result, we provide two non-trivial methods to approximate the simplicial depth of a given point in higher dimension. Along the way, we establish a tight combinatorial relationship between the Tukey depth of any given point and its simplicial depth.

Another problem investigated in this thesis is the dominance reporting problem, an important special case of orthogonal range reporting. In three dimensions, we solve this problem in the pointer machine model and the external memory model by offering the first optimal data structures in these models of computation. Also, in the RAM model and for points from an integer grid we reduce the space complexity of the fastest known data structure to optimal. Using known techniques in the literature, we can use our results to obtain solutions for the orthogonal range searching problem as well. The query complexity offered by our orthogonal range reporting data structures match the most efficient query complexities known in the literature but our space bounds are lower than the previous methods in the external memory model and RAM model where the input is a subset of an integer grid. The results also yield improved orthogonal range searching in higher dimensions (which shows the significance of the dominance reporting problem).

Intersection searching is a generalization of range searching where we deal with more complicated geometric objects instead of points. We investigate the rectilinear disjoint polygon counting problem which is a specialized intersection counting problem. We provide a linear-size data structure capable of counting the number of disjoint rectilinear polygons intersecting any rectilinear polygon of constant size. The query time (as well as some other properties of our data structure) resembles the classical simplex range searching data structures.

Acknowledgements

During my work on this thesis, I had the distict honor of being supervised by Timothy M. Chan whose insight and incredible knowledge of the field came to my rescue on many occasions. I am also in debt to his dedication and guidance; he always found time to discuss new problems or to review my drafts, even those he did not appear as a coauthor; without his intellectual input, this thesis would have been an impossibility.

I would also like to thank my friends Reza Dorrigiv, Hamid Zarrabi-Zadeh, Arash Farzan and specially, Mahya Ghandehari and Hamed Hatami who always shared their insights with me.

Finally, I would like to thank my thesis committee members Alex Lopez-Ortiz, Anna Lubiw, Chaitanya Swamy and Sariel Har-Peled for their many constructive comments.

Dedication

To my parents, Akram and Mohammadtaghi, for setting an example to revere,
to my sister, Shilan, whom I love dearly, and
finally, to Laura who made this easier with love and patience.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Range Searching

Range searching problems are among the most natural data structure problems in computational geometry. For instance, for a set of points scattered in the plane or 3D space, we can ask "report the points that lie inside a circle" or "count the points in a triangle" (Figure 1.1). In statistics, graphics, geographic information systems, databases or other fields such queries arise and require efficient solutions.

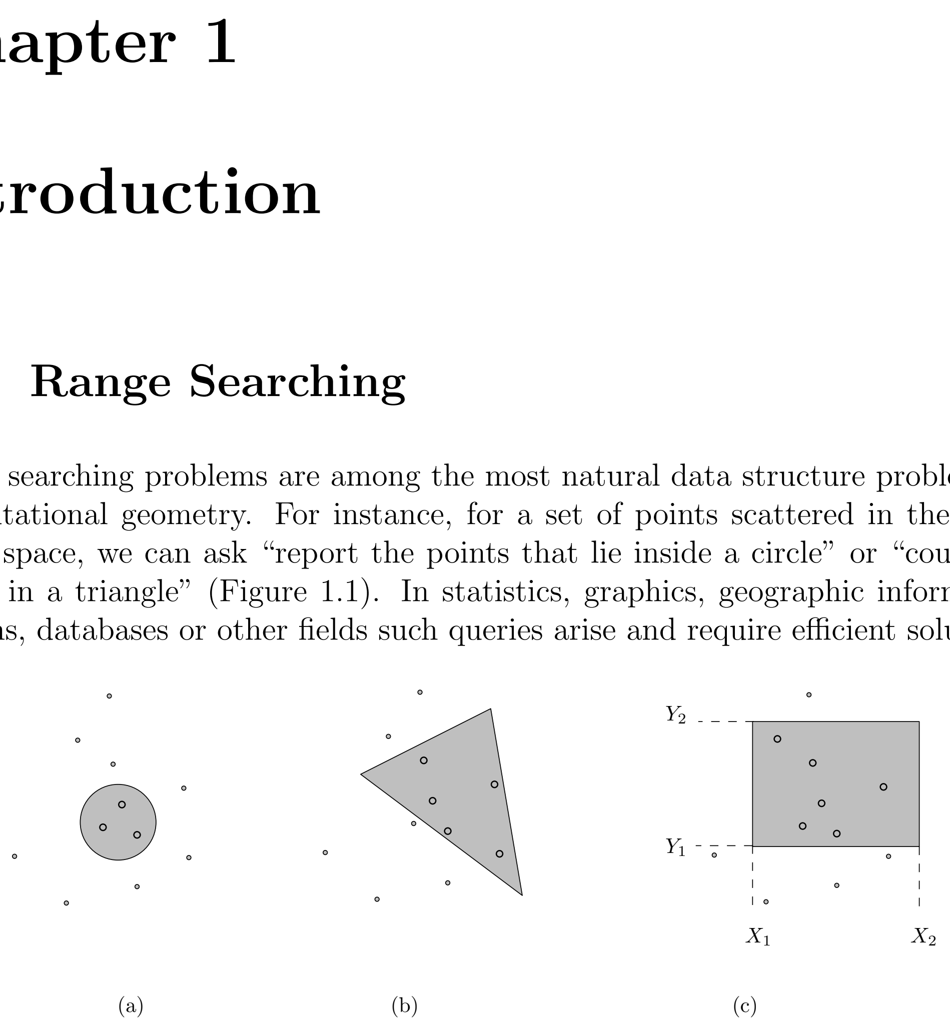


Figure 1.1: Some examples for range searching queries.

Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^d$. In a *range searching* problem we want to preprocess $\mathcal{P}$, such that for a query object q the elements of $\mathcal{P}$ contained in q can be found efficiently. Often, q is restricted to a specific class of geometric objects. The important cases of geometric range searching include the following:

- Simplex range searching: In this problem the query object is a simplex, a convex hull formed by $d + 1$ points. Since any polyhedral object can be decomposed into simplices, the simplex case is one of the most fundamental cases of range searching.

- Halfspace range searching: In this problem the query object is a halfspace. From the algorithmic perspective, a halfspace can be modelled as a simplex with one vertex placed sufficiently far away and thus this is a special case of simplex range searching. However, this special case is important as many problems such as spherical range searching can be reduced to this problem (the spherical range searching in $\mathbb{R}^d$ reduces to halfspace range searching in $\mathbb{R}^{d+1}$ via an elementary geometric transformation).

- Orthogonal range searching: Here, the query object is an orthogonal box. Many problems outside computational geometry can be reduced to this case. For instance, in the context of databases, for a given set of employees, the x and y-coordinate can represent the salary and the age of each employee respectively. Thus, a query for the rectangle $[X_1\ Y_1] \times [X_2\ Y_2]$ translates to "Give me the name of all the employees aged between Y_1 and Y_2 who earn between X_1 and X_2." (Figure 1.1(c)).

- Dominance searching: Here, the query range is determined by one point A and contains all the points that have all their coordinates smaller than A (Figure 1.2). Sometimes a solution for the general case of orthogonal searching can be obtained by using this special case as a subroutine.

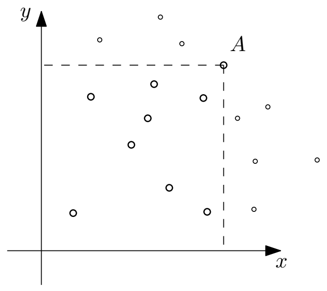


Figure 1.2: An example of a 2D dominance query determined by a point A.

Furthermore, there are many variants of each range searching problem.

- Reporting: In a range reporting problem (such as simplex range reporting) the goal is to output all the points contained in the query range. Usually, the query time of range reporting algorithm is in form of $O(f(n) + k)$ where k is the size of the output. In this thesis, we may omit k if there is no fear of ambiguity.

- Counting: In a range counting problem (such as simplex range counting) the goal is to simply count the number of points in the query range.

- Emptiness: This is a decision problem where we determine whether the query range contains at least one point or not. This is a special case of both counting and reporting variants.

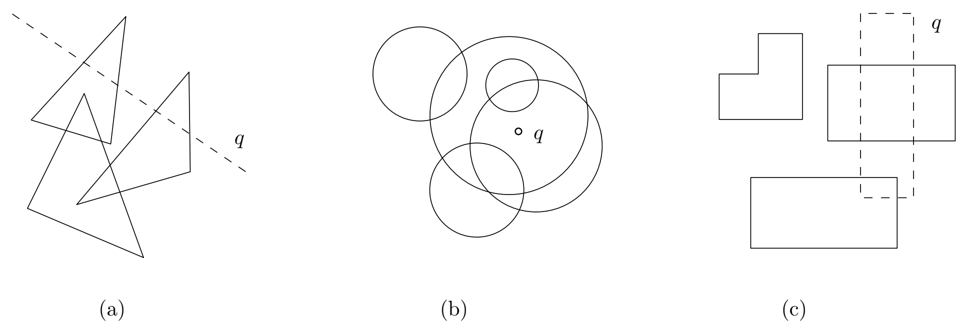


Figure 1.3: Some examples for intersection searching queries. A query q is marked with dashed lines when possible.

Intersection searching is a generalization of range reporting where the input $\mathcal{P}$ is a set of n geometric objects and given a query object q the goal is to search for the elements of $\mathcal{P}$ that intersect q. This is another fundamental geometric problem with applications in areas such as computer graphics and other fields. The following list contains some of important cases of intersection searching.

- Segment intersection searching: In this problem, the queries are line segments and the input can be any set of geometric objects. Input objects may include segments, circles, boxes or polygons (Figure 1.3(a)).
- Point intersection searching: In this problem, the queries are single points where we want to search for all the input objects containing the query point (Figure 1.3(b)). This is in some sense the dual of range searching.
- Rectilinear polygon searching: In a rectilinear polygon all edges are either vertical or horizontal. Many problems in VLSI design are related to rectilinear polygons which makes this an important special case. Here, the input and query objects are rectilinear polygons (Figure 1.3(c)).

In this context, *intersection counting* and *intersecting reporting* can be defined similarly. Usually, for a range searching (or intersection searching) data structure the important measures of efficiency are preprocessing time, space and query time.

For further background, the reader is referred to the surveys by Agarwal and Erickson [5, 10].

1.1.1 Basic notation

Throughout this thesis ε, always refers to an arbitrary positive constant, the O_ε notation hides constant factors that depend on ε and the $\tilde{O}$ notation hides polylogarithmic factors. Often, we assume that the input is in general position, which intuitively means the combinatorial structure of the input (usually a set of points) cannot be altered by any sufficiently small perturbation. For instance, in case of a planar point set this implies no three of them are on a straight line and no four of them on a circle. In many cases, there are standard techniques which remove this extra assumption [60]. For two values a and b, if $a \geq (1+\varepsilon)b$ then we say a is ε-approximately greater than b, if $a \leq (1-\varepsilon)b$ then we say a is ε-approximately less than b and if $(1-\varepsilon)b \leq a \leq (1+\varepsilon)b$ then we say a is ε-approximately equal to b.

The model of computation. Throughout this thesis, the default model of computation is the standard RAM models used in computational geometry. This RAM model is equipped with both integer and real registers. The integer registers contain $\log n$ bits and are capable of performing integer arithmetic in constant time. The real registers can contain any real number and are able to do real arithmetic in constant time; however, the conversion from reals to integers is not allowed.

In the next subsections we introduce various tools such as levels, cutting lemmas and partition theorems which play an important role in many algorithms.

1.1.2 Levels

The concept of levels is often encountered when studying problems that deal with arrangements of lines and hyperplanes. Some of the most well-known open problems of discrete geometry are about this concept and its generalization.

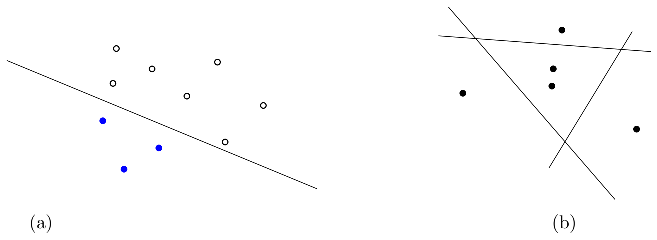


Figure 1.4: (a) A 3-set has been marked in bold. (b) Five points with only three 1-sets.

This concept was first explored in around 1970 by Lovász [88] and Erdős et al. [65]. They defined a *k-set* for a set $\mathcal{P}$ of n points in the plane as a subset of $\mathcal{P}$ of size k separated from $\mathcal{P}$ via a line ℓ (see Figure 1.4 for an example). Since then, bounding the maximum number of such k-sets as function of n and k is known as the *k-set problem*. The point-line duality [60] is an elementary geometric transformation that conserves the spatial relationship between lines and points. We represent the dual of a geometric object q (or a set $\mathcal{S}$ of geometric objects) with $\overline{q}$ (or $\overline{\mathcal{S}}$). By this duality, a point p below a line ℓ is mapped to a line $\overline{p}$ which passes below the point $\overline{\ell}$. Thus, a subset of points of $\mathcal{P}$ below a line ℓ corresponds to the subset of lines of $\overline{\mathcal{P}}$ passing below the point $\overline{\ell}$. Given an arrangement of lines $\mathcal{A}$, define the *level* of a point p as the number of lines passing directly below p (Figure 1.5(a)) and the *k-level* of $\mathcal{A}$ as the closure of the set of all the points of $\mathcal{A}$ with level equal to k (Figure 1.5(b)). The size of the k-level is the number of the vertices of $\mathcal{A}$ contained in it. In dual space, the k-set problem asymptotically translates to bounding the size of the k-level of $\mathcal{A}$. This is known as the *k-level* problem.

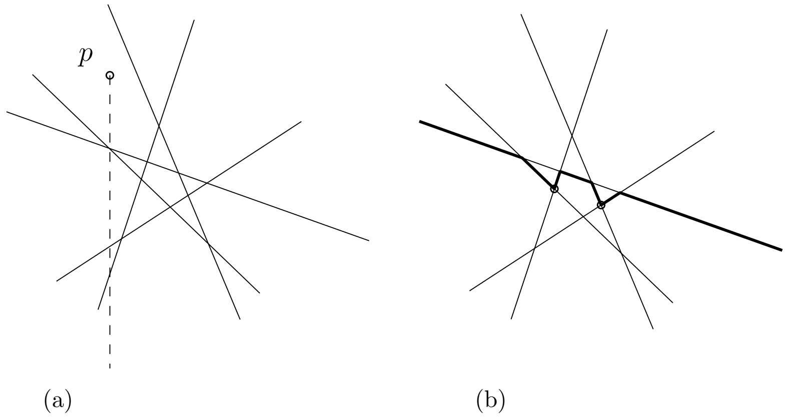


Figure 1.5: (a) A point p with level 4. (b) 2-level is drawn in bold. The marked points have level 1 but they are included in the 2-level since they lie in the closure of the points with level 2.

The k-set problem has a fascinating story. The original papers by Lovász [88] and Erdős et al. [65] established an $O(n\sqrt{k})$ bound. Erdős conjectured that the true bound lies in $o(n^{1+\varepsilon})$. The same asymptotic upper bound has been derived by different researchers (see [8, 40]). The first improvement, by Pach et al. [104], arrived after about two decades and only reduced the bound by a $\log^* k$ factor, using a complicated proof. The story of the upper bound ends with a 1998 paper by Dey [62] who obtained a bound of $O(nk^{1/3})$ using clever techniques. On the other hand, the current best lower bound is $n2^{\Omega(\sqrt{\log n})}$ [117].

The research into this problem has led to many interesting generalizations. Many proof techniques can be applied to the k-levels of *pseudo-lines* or *pseudo-segments* as well. An arrangement of x-monotone curves is called an arrangement of pseudo-lines if every two curves intersect at most once. An arrangement of curve-segments satisfying the same condition is called an arrangement of pseudo-segments. More generally, a set of x-monotone curves where each pair intersects at most s times gives rise to an arrangement of *s-intersecting curves* (the term *pseudo-parabolas* is used for $s = 2$). Many papers deal with the above generalization of k-level and related problems [7, 8, 11, 12, 39, 40, 43, 92, 107, 115, 116]. Similar questions in 3 or higher dimensions have received some attention [42, 82, 112].

Apart from the historical importance of this problem, levels frequently appear in many algorithmic problems, although sometimes *$(\leq k)$-levels* are more relevant in algorithmic applications. In an arrangement $\mathcal{A}$ of n lines in $\mathbb{R}^2$, the $(\leq k)$-level refers to the closure of the set of all the points with level at most k (see Figure 1.6). Similar to the k-set problem, here the objective is to bound the number of vertices of $\mathcal{A}$ contained in the $(\leq k)$-level of the arrangement. It is not difficult to come up with examples where every k-level of $\mathcal{A}$ has size $\Theta(n)$. Thus, $\Omega(nk)$ is an obvious lower bound on the worst case size of the $(\leq k)$-level and a matching $O(nk)$ upper bound can be proved. Here, we present a short proof which is based on the powerful technique of Clarkson and Shor [56]. To use this technique we need surprisingly few ingredients, only a non-trivial upper bound on the 0-level of the arrangement. In 2D the worst case complexity of the 0-level is $O(n)$.

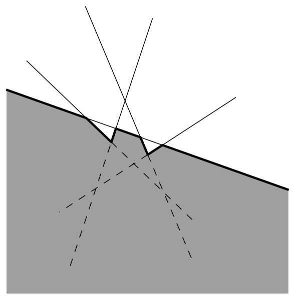


Figure 1.6: The grey area represents the (≤ 2)-level of the arrangement.

Lemma 1.1.1. *The complexity of the $(\leq k)$-level of an arrangement $\mathcal{A}$ formed by a set $\mathcal{P}$ of n lines in the plane is $O(nk)$.*

Proof. Let $\mathcal{S}$ be a random p-sample of $\mathcal{P}$ (i.e., a subset of $\mathcal{P}$ where each element is chosen independently with probability p). Let L_0 be the 0-level (also known as the lower envelope) of the arrangement formed by $\mathcal{S}$. We fix $p := k^{-1}$. We have

$$E(|L_0|) = O\left(np\right) = O\left(\frac{n}{k}\right). \tag{1.1}$$

Let $\mathcal{M}$ be the set of all the vertices contained in the $(\leq k)$-level of $\mathcal{A}$. We compute the probability of a vertex $v \in \mathcal{M}$ appearing on the lower envelope of $\mathcal{S}$. Let ℓ and ℓ' be the two lines incident to v and $\ell_1, \ldots, \ell_t$ be the lines which pass below v. Since we have assumed that v lies inside the $(\leq k)$-level of $\mathcal{A}$, we have $t \leq k$. The probability of v appearing on L_0 is equal to the probability of ℓ and ℓ' being chosen in $\mathcal{S}$ times the probability that none of the lines $\ell_1, \ldots, \ell_t$ are chosen in $\mathcal{S}$. Since $t \leq k$ we have

$$\Pr[\, v \in L_0 \,] \geq p^2 \left(1-p\right)^k = \Theta(p^2 e^{-kp}) = \Theta(k^{-2}).$$

Thus, in expectation we have $E(|L_0|) = \Omega(|\mathcal{M}|k^{-2})$. Combining with (1.1) we have

$$|\mathcal{M}|k^{-2} = O\left(\frac{n}{k}\right)$$

or in other words, $|\mathcal{M}| = O(nk)$. $\square$

We will use this technique in many places throughout this thesis. In 3D, the worst case complexity of the 0-level is $O(n)$ and the same technique yields the following bound.

Lemma 1.1.2. *The complexity of the* $(\leq k)$*-level of an arrangement of* n *hyperplanes in* $\mathbb{R}^3$ *is* $O(nk^2)$.

1.1.3 Cuttings and partition trees

Divide and conquer is one of the successful ideas in design of the algorithms. Applying this idea in geometric settings is one of the main motivations to study cuttings. Let $\mathcal{H}$ be a set of n hyperplanes in $\mathbb{R}^d$. A $1/r$*-cutting* for $\mathcal{H}$ is a set of disjoint simplices $\mathcal{C}$ which cover $\mathbb{R}^d$ with each simplex $s \in \mathcal{C}$ intersecting at most n/r hyperplanes of $\mathcal{H}$. For a simplex $s \in \mathcal{C}$, we call the subset of $\mathcal{H}$ intersecting s the *conflict list* of s. The size of the cutting is the number of simplices in $\mathcal{C}$.

The main cutting theorem is the following.

Theorem 1.1.3. *[47, 48] For every parameter* $0 < r < n$*, a* $1/r$*-cutting of size* $O(r^d)$ *for a set* $\mathcal{H}$ *of* n *hyperplanes in* $\mathbb{R}^d$ *always exists.*

The bound $O(r^d)$ on the size of the cutting is tight: n hyperplanes form $\Theta(n^d)$ vertices and a simplex intersecting m hyperplanes can contain at most $O(m^d)$ vertices of the arrangement. So each simplex in a $1/r$-cutting contains $O\left((n/r)^d\right)$ vertices and thus, there must be $\Omega\left(n^d/(n/r)^d\right) = \Omega(r^d)$ simplices in the cutting.

An application. This theorem is invaluable when designing divide and conquer algorithms. For instance, consider the halfspace range counting problem. In dual space, this problem translates to finding the number of hyperplanes from an input set $\mathcal{H}$ of n hyperplanes which pass below a query point q. Using Theorem 1.1.3, build a $1/r$-cutting $\mathcal{C}$ for $\mathcal{H}$. For every simplex $\Delta \in \mathcal{C}$, store then number of hyperplanes which pass below Δ then recurse on the set of hyperplanes which cross Δ. To answer the query q, in $O(r^d)$ time we find the simplex Δ which contains q. With a recursive call on Δ, we find the number of hyperplanes that cross Δ and pass below q. To this, we add the number of hyperplanes which pass below Δ. Let $S(n)$ and $Q(n)$ be the space complexity and the query time of the algorithm respectively. We have $S(n) = O(r^d)S(n/r) + O(r^d)$ and $Q(n) = Q(n/r) + O(r^d)$, which solve to $S(n) = O(n^{d+\varepsilon})$ and $Q(n) = O(\log n)$ if we set r to be a sufficiently large constant.

Let $\mathcal{P}$ be a set containing n points. A *simplicial partition* Π for $\mathcal{P}$ is a partition of $\mathcal{P}$ into r subsets $\mathcal{P}_1, \dots, \mathcal{P}_r$ of roughly the same size together with a list of simplices $\Delta_1, \dots, \Delta_r$ such that $\mathcal{P}_i$ lies inside Δ_i. The *crossing number* of any hyperplane h in this simplicial partition is defined as the number of simplices crossed by h. The

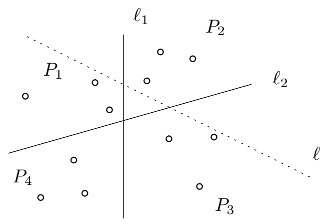


Figure 1.7: Lines ℓ_1 and ℓ_2 partition the plane into four regions each containing $n/4$ points. Any other line ℓ can cross at most three of these regions.

maximum value of the crossing number over all hyperplanes h is called the crossing number of Π. A simple construction of a simplicial partition in 2D is shown below:

Erect a vertical line ℓ_1 that partitions $\mathcal{P}$ into two subsets of size $n/2$. Using a well-known theorem [87] (ham-sandwich cut), we simultaneously partition the left and right subsets with another line ℓ_2 into subsets of size $n/4$ each (see Figure 1.7). With this construction, that any line can cross at most three of the four regions. Thus, this is a simplicial partition of size 4 with crossing number 3.

An application. Consider the halfspace range counting problem. Partition $\mathcal{P}$ into four sets $\mathcal{P}_1, \ldots, \mathcal{P}_4$ with crossing number 3. For each $\mathcal{P}_i$ repeat the same process and partition it further into four additional sets with crossing number 3 and continue this operation until the sets are of constant size are reached. The resulting structure can be represented with a tree of degree four which is called a *partition tree*. At each node we store the number of points contained in the set represented by that node. Clearly, the total storage cost is linear. For a given query halfspace h, we start from the top level of the data structure. The three subsets which cross the boundary of h can be found in constant time and by recursively calling the query on each of the three sets, we can find the number of points contained in h among them. Dealing with the fourth set is easy: either it is completely outside h, in which case we ignore it, or it is completely contained in h, in which case we add the number of points contained in it (which is precomputed) to the final answer. The query time $Q(n)$ can be bounded by the recursion $Q(n) \leq 3Q(n/4) + O(1)$, which solves to $Q(n) = O(n^{\log_4 3}) = O(n^{0.793})$. Thus, 2D halfspace range counting can be solved with linear space and $O(n^{0.793})$ query time.

Matoušek was the first to construct an optimal simplicial partition of $\mathcal{P}$ into r subsets for any value of r.

Theorem 1.1.4. *[96] For every parameter $0 < r < n$ and a point set $\mathcal{P}$ of size n, a simplicial partition Π of size r with crossing number $O(r^{1-1/d})$ exists.*

An application. Start with a simplicial partition for a parameter r and build a partition tree of degree r. Consider the halfspace range counting problem. Using this partition tree, we can obtain a linear-size data structure where the query time $Q(n)$ is determined by the recursion

$$Q(n) \leq O(r) + O(r^{1-1/d})Q(n/r).$$

Setting $r = \Theta(n^{\varepsilon})$ this recursion solves to

$$Q(n) = \tilde{O}(n^{1-1/d}) \text{ [96]}.$$

The query time can be further reduced to $O(n^{1-1/d})$ using the more complicated techniques of Matoušek [93].

Multilevel partition trees. In certain applications, we can apply the partition tree in the multilevel fashion which increases the domain of problems that can be attacked using this idea. Consider a partition tree $\mathcal{T}$ and call the subset of points represented by a node (internal or leaf) of $\mathcal{T}$ a *canonical set*. A recursion similar to the one seen above proves that if we set $r = n^{\varepsilon}$ then the set of points contained in any halfspace h is the union of $\tilde{O}(n^{1-1/d})$ canonical sets. To obtain a multilevel partition tree, we can store another partition tree (usually on a different point set associated with the canonical set) for every canonical set. We can continue this nesting operation a constant number of times and obtain a partition tree with a constant number of levels.

An application. Let $\mathcal{A}$ be a set of n line segments in $\mathbb{R}^d$. Consider the problem of reporting the subset of $\mathcal{A}$ crossed by a query hyperplane h. For every line segment denote one endpoint as the left endpoint and the other as the right endpoint. First, build a partition tree on the left endpoints. This single partition tree allows us to represent the set of points below q as the union of $\tilde{O}(n^{1-1/d})$ canonical subsets. Consider one such canonical subset $\mathcal{P}_i$ and an arbitrary point $p \in \mathcal{P}_i$. We know p lies below h. Observe that the segment s whose left endpoint is p intersects h if and only if the right endpoint p' of s lies above h. Thus, we can solve this problem by storing a secondary partition tree on the set of right endpoints corresponding to $\mathcal{P}_i$. The segments with their right endpoints below q can be reported in a similar fashion. The final result is a near-linear-size data structure which can answer segment intersection queries in $\tilde{O}(n^{1-1/d})$ time.

1.1.4 Shallow cutting and partition theorems

For some applications, better results can be obtained by alternate "shallow" versions of the cutting and partition theorem as noticed by Matoušek [97].

In the shallow version of the cutting theorem we are not interested in covering the whole space. For two parameter k and r, a *k-shallow $1/r$-cutting* for a set $\mathcal{H}$ of hyperplanes is a set of disjoint simplices $\mathcal{C}$ which cover the $(\leq k)$-level of $\mathcal{H}$

with each simplex $s \in \mathcal{C}$ intersecting at most n/r hyperplanes of $\mathcal{H}$. Clearly, any ordinary cutting is also a shallow cutting but a better bound is possible for shallow cuttings.

Lemma 1.1.5. *[97] For a set $\mathcal{H}$ of n hyperplanes in $\mathbb{R}^d$ and parameters $r, k < n$ a k-shallow $1/r$-cutting of size $O(r^d(k/n)^{\lceil d/2 \rceil})$ always exists.*

Usually, the most useful case of the above shallow cutting theorem is when $r = n/k$ where the bound becomes $O((n/k)^{\lfloor d/2 \rfloor})$. In 3D, we use the following modification of this lemma.

Lemma 1.1.6. *For any set of n planes in $\mathbb{R}^3$ and a parameter k, there exists a k-shallow $O(k/n)$-cutting of size $O(n/k)$ that covers the $(\leq k)$-level. The cells in the cutting are all vertical prisms unbounded from below (simplices with one vertex at $(0, 0, -\infty)$).*

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 for constant $\varepsilon > 0$; we can also construct the conflict lists of all cells in the same time.

The first part follows from Lemma 1.1.5. The construction time for the cuttings follows from an algorithm by Ramos [109]. That vertical prisms suffice was observed by Chan and the construction of vertical prisms involves computing the convex hull of vertices of original shallow cutting [36]. With $O(n \log n)$ expected preprocessing we build a halfspace range reporting data structure (see Section 1.2.2). The computation of the conflict lists can be carried out in additional $O(n(\log n + k)/k)$ expected time for each k: issue $O(n/k)$ halfspace range reporting queries (one for each vertex of the prisms), each requiring $O(\log n + k)$ time. Summing $O(n(\log n + k)/k)$ over all k still gives $O_\varepsilon(n \log n)$.

Consider one level of the cutting. We project the prisms on the plane, forming a planar configuration composed of $O(n/k)$ triangles (i.e., a triangulation). For each triangle, we store the equation of the plane corresponding to its corresponding vertical prism. For any point $q \in \mathbb{R}^3$, if we can find the triangle which contains the projection of q, then we can easy check whether q lies inside the corresponding vertical prism. In a 2D triangulation composed of m triangles, finding the triangle which contains a given point is known as *the planar point location problem* [63, 108] and can be solved using $O(m)$ space and $O(\log m)$ query time. Thus, in $O(\log(n/k))$ time, we can test whether q lies below a level of the cutting and if so, also find the vertical prism that contains q.

For a given point set $\mathcal{P}$, a halfspace is said to be *k-shallow* if it contains at most k points of $\mathcal{P}$. The following is the shallow version of the partition theorem.

Theorem 1.1.7. *[97] Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^d$ and k and r be parameters satisfying $k < n/r$. There exists a simplicial partition of size $O(r)$ such that the crossing number of any k-shallow halfspace is $O(r^{1-1/\lfloor d/2 \rfloor} + \log r)$.*

Remarks. The bound on the crossing number offered by the above theorem is tight for $d > 3$. Whether or not it can improved for $d \leq 3$ has been an open question going back to Matoušek [97] who wrote: "For $d = 2, 3$, it would be interesting to see to what extent the crossing number can be improved. It is not clear whether a crossing number bounded by a constant can be attained (we conjecture that it cannot)." In the appendix, we confirm this conjecture for $d = 3$, by exhibiting an example where the crossing number is $\Omega(\log r / \log\log r)$, proving that the upper bound is almost tight.

1.2 Classical Range Searching Results

In this section we review some known results in range searching. Many of the results discussed in this section use the ideas described in the previous section and thus they can be regarded as further applications of these tools.

1.2.1 Simplex range searching

The general simplex range searching problem is difficult in the sense that the best algorithms either consume too much space or have high query costs. The best simplex range searching algorithm using linear space and can answer queries in $O(n^{1-1/d})$ time [93]. This query time is conjectured to be optimal. As we have seen in the previous section, the partition theorem is a central part of this algorithm (although Matoušek used a heavily modified partition tree with additional properties).

Chazelle, Sharir and Welzl [54] were the first to propose trade-offs between space and the query complexity of the simplex range searching algorithms. They described a data structure which uses m units of space and can answer simplex range searching queries in $O(n^{1+\varepsilon}/m^{1/d})$ time. Such a trade-off can be obtained by building a partition tree in which we stop when the size of each node gets close to m, then implementing an algorithm with logarithmic query time and high space complexity on the resulting subsets. Matoušek [93] improved the query time to $O(n/m^{1/d} \log^{d+1}(m/n))$ using $O(m)$ units of space.

The algorithms obtained by Matoušek are considered to be close to optimal as there are lower bounds in the so-called semigroup model (see [68, 125] for more details) which almost match the complexity of his data structures. Essentially, it has been demonstrated that if the data structure uses m units of space then the query time is bounded below by $\Omega(n/\sqrt{m})$ in 2D and $\Omega(n/(m^{1/d} \log n))$ for dimensions three and higher [46]. It is interesting to note that these lower bounds do not work for halfspaces but they still work for strips (the region surrounded by two parallel hyperplanes).

1.2.2 Halfspace range searching

Since halfspace range searching is a special case of simplex range searching, the previous upper bounds are still valid for this problem. For instance, for the halfspace range counting problem the best bound still comes from simplex range searching: linear space and $O(n^{1-1/d})$ query time. We continue this section with a discussion of halfspace range reporting.

In the plane, Chazelle, Guibas and Lee [51] presented an algorithm with $O(n)$ space and $O(\log n + k)$ query time. For 3D, the first result with query time of $O(\log n + k)$ came from Chazelle and Preparata [53]. The algorithm as described by them has the space complexity of $O(n \log^8 n (\log \log n)^4)$ but plugging in the bound $O(nk^2)$ on the complexity of the $(\leq k)$-level obtained by random sampling (see Lemma 1.1.1) improves this to $O(n \log^2 n \log \log n)$. A result with optimal query time and $O(n \log n)$ space was provided by Aggarwal et al. [15] but the preprocessing time was not optimal. Much later, Chan [35] presented an algorithm with $O(n \log n)$ space and $O(n \log n)$ expected preprocessing and $O(\log n + k)$ expected query time. The space bound was later reduced to $O(n \log \log n)$ by Ramos [109] and the query time was made worst-case (the journal version of Chan's original paper also contains an improved algorithm with $O(n \log \log n)$ space [37]). For halfspace range reporting in higher dimensions the best query time is due to Clarkson and Shor [56]. Their algorithm can answer queries in $O(\log n + k)$ time using $O(n^{[d/2]+\varepsilon})$ space. Matoušek also covered the other extreme by an algorithm with $O(n \log \log n)$ space and $O(n^{1-1/[d/2]} \log^{O(1)} n + k)$ query time.

In this case, the halfspace emptiness problem is also hard. Matoušek offered an improvement in polylogarithmic factors for the query time: a data structure with $O(n^{1-1/[d/2]} 2^{O(\log^* n)})$ query time and linear space. There are various lower bounds for the halfspace emptiness queries and related problems [32, 67, 66] which work in a restricted "partition graph" model. Assuming s and t are the storage and the query time of the algorithm, these lower bounds prove $st^d = \Omega((n/\log n)^{d-(d-1)/(d+1)})$.

1.2.3 Orthogonal range searching

Orthogonal range searching has been studied extensively together with its many variants and special cases. We first consider the 2D case where the queries are orthogonal boxes.

One special case is when one coordinate of the queries is always fixed at infinity; in this case the query is a rectangle with one side at infinity, and thus is called a 3-sided query. A 2-sided query is one where one x-coordinate and one y-coordinate are fixed at infinity. We do not consider a query with both infinite x-coordinates or y-coordinates a 2-sided query since it reduces to a one-dimensional problem.

This problem can be generalized to higher dimensions. In $\mathbb{R}^d$ a query for orthogonal range reporting is a box $[X_1\ Y_1] \times [X_2\ Y_2] \times \dots [X_d\ Y_d]$ and a k-sided query can be defined similarly. A d-sided query in $\mathbb{R}^d$ leads to the idea of dominance: we say a point A *dominates* a point B if and only if all the coordinates of A are greater than those of B. A d-sided query translates into a dominance searching problem.

To review the possible solutions for orthogonal range searching, we start with the one-dimensional case. Here, the queries are intervals and we are asked to return (or count) the points contained within a query interval. In a pointer machine, this problem can be solved optimally by a binary search. We can extend this solution to higher dimensions by paying an extra $\log n$ factor in preprocessing, space and query complexities with each added dimension [30] (using range trees). This results in a data structure with $O(n \log^{d-1} n)$ space and $O(\log^d n + k)$ query time (or $O(\log^d n)$ for counting) in $\mathbb{R}^d$. It is possible to improve the query time of the algorithm for $d = 2$ to $O(\log n + k)$ [50, 90] thus improving the query time for all dimensions greater than one to $O(\log^{d-1} n + k)$. Chazelle provides another improvement by using an α-ary tree for a suitable choice of α [44]. Using his idea, it is possible to reduce the space complexity of the 2D case further to $O(n \log n / \log\log n)$ while still maintaining the same query time. This also generalizes to higher dimensions. In 2D, McCreight solves the 3-sided queries optimally using linear space and $O(\log n + k)$ query time [101].

A popular idea in orthogonal range searching is reduction to the rank space. We can briefly describe this idea as follows. Assume $x_1, \ldots, x_n$ are the x-coordinates of the input point set sorted in increasing order. We can change x_i to i without distorting the order of the points. Repeating this process on all the coordinates ensures that the input is a subset of the grid $\{1, \ldots, n\}^d$. If the smallest x-coordinate of the query is x_q, then we can replace x_q with index i such that $x_{i-1} < x_q \le x_i$. Repeating this for all the query coordinates maps the query coordinates to the new grid coordinates; however, we need to solve the following subproblem called the *predecessor search*: given a set of integers $\mathcal{S}$, preprocess them such that for a given query number x_q one can find the largest $x \in \mathcal{S}$ with $x \le x_q$.

The reduction to rank space maps the input coordinates to integers which makes it possible to store the coordinates of the points in the integer registers, making various bit manipulation techniques possible. This allows us improve the space complexity of various orthogonal range reporting algorithms in this model. In 2D, Chazelle [45] obtains a data structure with $O(n \log^\varepsilon n)$ space and $O(\log n + k)$ query time. For 3D, Alstrup et al. [18] offer a data structure with $O(n \log^{1+\varepsilon} n)$ space and $O(\log n + k)$ query time. As before, we pay a $\log n$ factor in space and query time to extend these results to one dimension higher; however, Alstrup et al. [18] provide an alternative: we can choose to pay a $\log^{1+\varepsilon} n$ factor in space while suffering only a $\log n / \log\log n$ factor increase in the query time.

In the word RAM model we assume the input is a subset of a $\{1, \ldots, U\}^d$ integer grid, with the word size being at least $\log U$ bits. Within this framework, we can solve the predecessor search problem in $O(\log\log U)$ time and with linear space [124] (see also [69, 105, 106]). This results in a 2D orthogonal range reporting data structure that uses $O(n \log^\varepsilon n)$ space and can answer queries in $O(\log\log U + k)$ time [18]. In 3D, a data structure of Nekrich [102] uses $O(n \log^4 n)$ space and can answer queries in $O(\log\log U + (\log\log n)^2 + k)$ time. As before, his method generalizes to higher dimensions resulting in an $O(n\log^{d+1+\varepsilon} n)$ space data structure with query time of $O\left(\log^{d-3} n/(\log\log n)^{d-5} + k\right)$ for $d \geq 4$ [102] on the standard RAM.

In the external memory model, the data is organized in blocks of size B and the query complexity is measured by the number of blocks accessed during the query process. We only survey results with near linear space and polylogarithmic query complexity in this model. Arge et al. [20] describe a linear-size data structure that solves the 2D 3-sided queries with $O(\log_B n + k/B)$ query I/Os. Combining it with the filtering technique of Chazelle [44] they obtain a data structure for the 2D orthogonal range reporting problem with $O(\log_B n + k/B)$ query I/Os which uses $O((n/B)\log(n/B)/\log\log_B n)$ space; this is optimal [20]. For 3D orthogonal range reporting, Vengroff and Vitter present a data structure using $O((n/B)\log^4 n/\log\log_B n))$ space and $O(\log_B n + k/B)$ query I/Os [121].

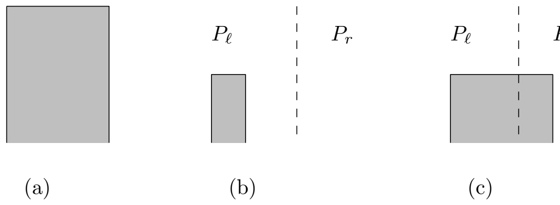


Figure 1.8: (a) A 3-sided query. (b) The query can be answered by recursing on P_ℓ. (c) The query can be answered by two 2-sided queries on P_ℓ and P_r.

As we have explained, dominance and k-sided queries are important special cases of orthogonal range reporting. One reason is that it is possible to reduce $(k+1)$-sided queries to k-sided queries with a small overhead [102, 121, 113]. We can explain this basic idea in 2D, although it is more frequently used in 3D:

Assume we have a data structure capable of answering 2-sided queries in $Q(n)$ time which consumes $S(n)$ units of space. We use this data structure to answer 3-sided queries as follows. Without loss of generality, assume the 3-sided queries that we are interested in are those with a top, left and right side (Figure 1.8(a)). Sort the input points according to their x-coordinates and let x_m be the median x-coordinate. Partition the point set into two sets P_ℓ and P_r with P_ℓ containing the $n/2$ points with smallest x-coordinates. On both sets P_ℓ and P_r, implement a data structure capable of answering any 2-sided query then recurse on P_ℓ and P_r. This increases the space complexity to $O(S(n)\log n)$. Now consider a 3-sided query. We have two cases: either the query does not cross the vertical line through x_m, or it does (Figure 1.8(b,c)). In the former case, we can safely recurse on one of the sets P_ℓ or P_r, since the query rectangle does not contain any point from the other set. In the latter case, the query decomposes into two 2-sided queries, which can be answered immediately using the data structures on P_ℓ and P_r. This naive implementation results in the query time of $O(\log n + Q(n))$. However, we can improve this by noticing that the extra $\log n$ factor comes from finding the proper vertical line which decomposes the 3-sided query into two 2-sided queries. It is possible to reduce this step into another predecessor search subproblem and lower the query time for 3-sided queries to $O(\log\log U + Q(n))$. This type of reduction motivates us to consider dominance reporting problem in greater detail, since it is the simplest among k-sided queries.

In 2D, 2-sided and 3-sided queries can be solved optimally. In 3D and in the pointer machine model, Makris and Tsakalidis [91] achieve the query time of $O(\log n \log\log n + k)$ with linear space, improving an old result from 1987 [49]. Also, in the same paper they achieve the query time of $O((\log\log U)^2 \log\log\log U + k \log\log U)$ in the word RAM model and with linear space. Further results in the RAM model include an algorithm with linear space and query time of $O(\log n / \log\log n + k)$ [78] assuming integer inputs. However, these are not the fastest data structures, since if we allow $O(n \log n)$ space we can achieve the query time of $O((\log\log n)^2 + \log\log U + k)$ [102]. In the external memory model, there are fewer results and it is believed that solving orthogonal range reporting problems is more difficult than in the main memory model [113]. Currently, the best algorithm uses $O(n \log_B n)$ space and can answer queries with optimal $O(\log_B n + k/B)$ I/Os [121, 122].

1.3 Robust Statistics

Many problems in robust statistics are related to geometric range searching, specifically, range counting. As our techniques will have implications in this area, we will introduce basic concepts in computational statistics.

Statistical data points are usually accompanied by errors, exceptions and anomalies since virtually all the practical ways of gathering data are imprecise. For instance, consider a series of measurements of a certain phenomenon. It is quite improbable that we get the exact same values when repeating the said measurement. A fundamental question is how to use the resulting sequence of numbers to get a single good estimate. While we can assume the errors are from a normal distribution with unknown mean and variance, this may not be true in practice. In this section, we review ideas on how to obtain an estimate, given an input of data points, without making any assumptions on the nature of the error.

Comparing mean and median. Assume we are given a set $\mathcal{P}$ of n real values and we are asked to find the "center" or "representative" of this point set.

The trivial strategy of averaging the numbers can be problematic in some cases. For instance, if $n-1$ points of $\mathcal{P}$ are between 0 and 1 and just one point is above n, then the averaging strategy will return an answer greater than 1 whereas a "typical" point of $\mathcal{P}$ lies between 0 and 1 (remember we have not assumed anything about the nature of the error). In practice, sometimes such erroneous data points are removed by researchers themselves.

Instead consider the median of the given numbers. In our example the median is between 0 and 1 and it is clear that a few very imprecise points cannot alter the median by much. In other words, the median is automatically robust against outliers. The following definition provides one theoretical way to measure this tolerance to outliers.

Breakdown value. Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^d$. Assume we have access to a function F which given $\mathcal{P}$ outputs a single point as the estimate (intuitively, the single point which is closest to the "true" value and minimizes the error). The robustness of function F is measured by its *breakdown value* which is defined as the fraction of the points which need to be moved until this estimate can be moved to infinity.

For example, the median function has breakdown value of 1/2; if one moves less than half the points to infinity, then the median would still be confined in an interval formed by the remaining points. On the other hand, the mean function has the breakdown of value $1/n$.

While the main emphasis in robust statistics is to find or compute good estimators, many definitions allow us to assign an attribute known as *depth* to every point in the space, with a point of maximum depth chosen as the estimator. In the remainder of this section we review some of the most commonly used definitions of data depth.

1.3.1 Tukey depth

The *Tukey depth* of a point q is defined as the minimum number of points in any halfspace that contains q. This is also known as the *halfspace depth*. If q is outside the convex hull of $\mathcal{P}$, then there exists a hyperplane which separates q from $\mathcal{P}$ and thus the Tukey depth of q is 0. On the other hand, Helly's theorem implies the existence of a point with Tukey depth at least $n/(d+1)$: Consider halfspaces that contain more than $n - (n/(d+1))$ points of $\mathcal{P}$. Since there are less than $n/(d+1)$ points outside every such halfspace, any $d+1$ of them contain an element of $\mathcal{P}$ and thus have a common point. By Helly's theorem, all of them have a common point. That common point has Tukey depth at least $n/(d+1)$. It is not difficult to see that the breakdown value of Tukey depth is $1/(d+1)$.

In 1D, the Tukey depth of a point is easily determined by its rank. Thus, depth of a point and the point of maximum depth (which is the median) can be computed in linear time. In 2D, the situation is different and Tukey depth of a given point can be computed in $O(n \log n)$ time, which is optimal [16]. This computation is not too difficult: Let q be the given point. Sort the points radially around q as in Figure 1.9. Since only the location of the points on the unit circle around q is important, the problem reduces to a one-dimensional problem: given n points in an interval of size 2π, find an interval of size π containing maximum number of points.

This can be solved in $O(n)$ time given the sorted order of points [16]. In the same paper, Aloupis et al. [16] prove the matching lower bound of $\Omega(n \log n)$ by reducing it to the set equality problem (whether two given sets $\mathcal{A}$ and $\mathcal{B}$ are identical).

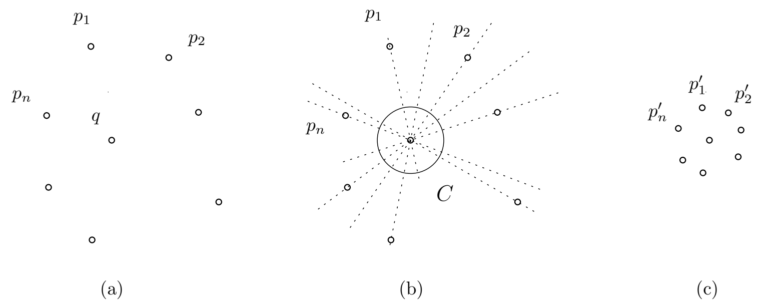


Figure 1.9: Sorting points radially around q. Only the final location on the circle C is relevant to the Tukey depth of q.

Computing a point of maximum Tukey depth is not as easy. In 1991, Matoušek [95] proposed an $O(n \log^5 n)$ time algorithm to find the Tukey median in 2D. The running time was later improved to $O(n \log^4 n)$ by Langerman and Steiger [86] and was finally made optimal by Chan [41] who provided an $O(n \log n)$ expected time algorithm. In higher dimensions the situation is worse. Chan's algorithm is able to find a Tukey median in $\mathbb{R}^d, d > 2$, in $O(n^{d-1})$ expected time which is believed to be optimal.

A related concept here is a *centerpoint* which is a point with halfspace depth at least $n/(d+1)$. As already explained, any point set has at least one centerpoint. Obviously, algorithms computing a Tukey median also compute a centerpoint, but in 2D one can find a centerpoint in linear time [77].

1.3.2 Simplicial depth

The simplicial depth [111] of a point $q \in \mathbb{R}^d$ with respect to a point set $\mathcal{P} \subset \mathbb{R}^d$ is defined as the number of simplices formed by $d+1$ points of $\mathcal{P}$ which contain q. This is another important definition of data depth. Unfortunately, this notion of data depth appears to be more computationally challenging than other definitions of depth, like the Tukey depth. For instance, in 2D the best algorithm computing a simplicial median (a point with the maximum simplicial depth) runs in $O(n^4)$ time [17]. In higher dimensions the situation is worse and the only strategy seems to be the brute-force one.

Interestingly, computing the simplicial depth of a single point in 2D is considered even before its formal definition [83], because it translates to counting the number of triangles containing the given point which is of independent interest. At least three papers describe algorithms to compute the simplicial depth of a given point in 2D in $O(n \log n)$ time [70, 83, 111], which is optimal [16]. As with Tukey depth, sorting points radially around the given point reduces the problem to a one-dimensional problem.

The situation is far more complicated in 3D. Finding the simplicial depth of a given point can be done in $O(n^2)$ time in 3D and the solution came after a failed attempt (see [70] and the correction of its flaw [55]).

For dimensions four and higher, there seems to be no significant improvements over the trivial $O(n^{d+1})$ brute-force solution.

1.3.3 The $L1$ median

The $L1$ *median* of a point set $\mathcal{P}$ is defined as the point q which minimizes the sum of distances to points of $\mathcal{P}$. This appears in an old problem proposed by Fermat to Torricelli in which he asked for the $L1$ median of three points. Because of this, finding the $L1$ median for larger point sets is sometimes known as the generalized Fermat-Torricelli problem (or Fermat-Weber problem). Unfortunately, going beyond three points has proved to be much more difficult. It is believed that the exact computation of $L1$ median is not possible but there are various numerical methods designed to approximate the median. The $L1$ median has the breakdown value of 1/2, which is the maximum possible.

1.3.4 Regression and hyperplane depth

Rousseeuw and Hubert [110] introduced the *regression depth* to address the problem of finding a "representative" hyperplane for a set of data points. The regression depth of a hyperplane h with respect to a point set $\mathcal{P}$ is defined as the minimum number of points of $\mathcal{P}$ intersected during any continuous motion which turns h into a vertical hyperplane (a hyperplane parallel to the d-th coordinate vector). In dual space, this corresponds to *hyperplane depth* of a point q that is defined with respect to a set of hyperplanes as the minimum number of hyperplanes intersected by a ray originating from q.

The existence of a hyperplane of regression depth $n/(d+1)$ for a point set of size n (which was a conjecture of Rousseeuw and Hubert [110]) was proved by Amenta et al. [19].

In 2D van Kreveld et al. [120] presented an algorithm that finds a point of maximum depth in $O(n \log^2 n)$ time. This has been improved to $O(n \log n)$ by Langerman and Steiger [86, 85].

1.3.5 Other definitions of depth

The *Oja depth* is similar to the simplicial depth: in Oja depth of a point q, we sum the volumes of all the simplices created by d points of the original point set and q.

The point with the minimum Oja depth is called the *Oja median.* The *Tverberg depth* is also another depth measure similar to simplicial depth: the Tverberg depth of a point q is the maximum number of vertex disjoint subsets from the point set $\mathcal{P}$ which contain q inside their convex hulls [19] (see [13] for more details and references).

For any point set $\mathcal{P}$, there exists a point with Tverberg depth of $n/(d+1)$ [118]. In 2D the Tverberg depth is equal to the Tukey depth when the Tukey depth is less than $n/3$. This relation is shown to be false in 3D [27]. We are not aware of other nontrivial results relating these two depth concepts.

1.4 Our Results and Organization of the Thesis

The high query time of some of the best range searching data structures is a motivation to consider approximations as one possible way to lower the query time. In Chapter 2 we review possible approaches to approximation, together with some background material.

We consider relative approximation of halfspace range counting queries in Chapter 3. We continue the previous attempts and present the first data structure that uses linear space and can answer 3D approximate halfspace range counting queries in $O(\log n)$ expected time. Unlike the previous algorithms where the correctness was guaranteed with high probability, our results are always correct. Our techniques imply new results for problems related to statistical depth. In 2D, we show the regression depth of a query line can be approximated in $O(\log n)$ time using a linear-size data structure. Similar ideas can be used for the Tukey depth problem, yielding an optimal 2D algorithm ($O(n)$ space, $O(\log n)$ query time) and a near optimal algorithm in 3D (linear space, $O(\log n \log\log n)$ query time). The results of this chapter have appeared in a SoCG paper [4].

In Chapter 4 we focus on the approximate simplicial depth problem. Unfortunately, the techniques from the previous chapter do not apply to this problem. In 2D, we solve this problem using a data structure of size $\tilde{O}(n)$ capable of answering approximate simplicial depth queries in $\tilde{O}(1)$ time (as a reminder, the $\tilde{O}(\cdot)$ notation hides the polylogarithmic factors). This planar algorithm has at least two applications in higher dimensions. One is an $\tilde{O}(n^{d-2})$ time algorithm for approximating the simplicial depth of a given point and another is a data structure of size $\tilde{O}(n^2)$ capable of answering 3D approximate simplicial depth queries in $\tilde{O}(n^{2/3})$ time. Other results in this chapter include a tight combinatorial relationship between the simplicial depth of a point and its Tukey depth.

The 3D dominance reporting problem is studied in Chapter 5. We present the first optimal data structures in the pointer machine model ($O(n)$ space, $O(\log n)$ query time) and the external memory model ($O(n)$ space, $O(\log_B n)$ query I/Os) and lower the space complexity of the fastest data structure by a $\log n$ factor in the word RAM model (our result has $O(n)$ space and $O(\log\log U + (\log\log n)^2 + k)$

query time). These can be used to obtain new data structures for orthogonal range searching: one with $O(n \log^3 n)$ space and $O(\log\log U + (\log\log n)^2 + k)$ query time in the word RAM model and another with $O(n \log^3 n)$ space and $O(\log_B n)$ query I/Os in the external memory model. As before, these 3D orthogonal results can be extended to higher dimensions. The results contained in this chapter will appear in an ESA 2008 paper [1].

In Chapter 6 we study the following problem in intersection counting: given a set of disjoint rectilinear polygons, preprocess them such that the number of polygons intersecting a query rectilinear polygon of constant size can be found efficiently. We solve this problem using linear space and $\tilde{O}(n^{6/7})$ query time. This is one of the few intersection counting data structures where the input objects have non-constant size while the size of the query objects can be any constant value. A preliminary version of the results of this chapter has appeared in an ESA 2006 paper [3].

Chapter 2

Approximate Counting: The Basics

As we surveyed previously, many range searching problems such as simplex and halfspace range searching are hard in the sense that no single algorithm achieves both near linear storage complexity and low query time; many researchers point to lower bounds in some models of computation as evidence that the existing data structures for halfspace and simplex range searching cannot be improved significantly. This justifies the attempts made to obtain approximate versions of range searching problems. This approximation can be achieved through the following methods:

1. through approximating the ranges,
2. through ε-nets and ε-approximations, and
3. through relative ε-approximation.

In this thesis, we will only consider the third form of approximation but we will examine the first two categories in the following sections.

2.1 Approximating the Ranges

One possible approach to approximation is by approximating the ranges. This was first done by Arya and Mount [26].

Let $\mathcal{P}$ be the input point set and let r be a bounded query range of diameter ω and $\varepsilon > 0$ be an arbitrary constant. We are interested in the list of points contained in r; however, we allow the data structure to either report or discard any point within $\varepsilon\omega$ distance from the boundary of r. Intuitively, the range r is regarded as an object with fuzzy border.

One drawback of this approach is that it becomes meaningless for thin or unbounded objects: for unbounded objects diameter is undefined and for thin objects (those with width to diameter ratio of less than ε) the empty set is a trivial correct answer. In particular, this definition does not make sense for halfspace range searching.

Nonetheless, for bounded objects, this particular definition makes polylogarithmic query times possible using only linear space. Arya and Mount present a data structure with $O(n)$ space and $O(\log n + (\frac{1}{\varepsilon})^d)$ query time for general queries and $O(\log n + (\frac{1}{\varepsilon})^{d-1})$ for convex queries.

This model has continued to receive considerable attention. For the case of spherical range searching, Arya et al. [25] provide a space/query trade-off: for any parameter γ, they design a data structure with $O(n\gamma^d \log(\frac{1}{\varepsilon}))$ space and $O((\gamma + \frac{1}{(\varepsilon\gamma)^d}) \log n)$ query time. When the dimension d is not considered a constant, we encounter the curse of dimensionality in which the query time is no longer polynomial. Chazelle et al. [52] have tackled this issue and obtained a data structure which can answer approximate halfspace range searching queries with $\tilde{O}(d/\varepsilon^2)$ query time using $dn^{O(\varepsilon^{-2})}$ space. Of course to define an approximate version of the halfspace range searching, they used a slightly different model: they assumed the diameter of the input point set is at most one and for a given query halfspace h the points within the distance ε from the boundary may or may not be reported.

2.2 ε-nets and ε-approximations

The concepts of ε-nets and ε-approximations have a rich history. From combinatorial perspective, these two concepts are defined for a finite set system $(\mathcal{S}, \mathcal{X})$ in which $\mathcal{X}$ is a ground set and $\mathcal{S} \subset 2^{\mathcal{X}}$. Throughout this section, we assume $|\mathcal{X}| = n$. An ε-net for this set system is a set $\mathcal{N} \subset \mathcal{X}$ with the property that for any $\mathcal{A} \in \mathcal{S}$ with $|\mathcal{A}| \geq \varepsilon n$ we have $\mathcal{A} \cap \mathcal{N} \neq \emptyset$. In other words, $\mathcal{N}$ must intersect all the "large" sets of the set system. This condition is obviously satisfied if we take $\mathcal{N} = \mathcal{X}$ and thus the goal is to minimize the size of the ε-net.

One possible way to construct an ε-net is via random sampling. Assume we include every element of $\mathcal{X}$ in $\mathcal{N}$ with probability p. The probability that no element of a set $\mathcal{A} \in \mathcal{S}$ gets sampled is

$$\Pr[\, \mathcal{X} \cap \mathcal{A} = \emptyset \,] = (1-p)^{|\mathcal{A}|} = \Theta(e^{-|\mathcal{A}|p}).$$

Thus, assuming $|\mathcal{A}| \geq \varepsilon n$ we can set $p = C \log |\mathcal{S}| / \varepsilon n$ and by choosing C to be sufficiently large we can ensure that

$$\Pr[\, \mathcal{X} \cap \mathcal{A} = \emptyset \,] < 1/3|\mathcal{S}|.$$

This means with probability at least $2/3$ the set $\mathcal{X}$ fails all the events $\mathcal{X} \cap \mathcal{A} = \emptyset$ and thus is an ε-net. On the other hand, with probability at least $2/3$ the set

$\mathcal{X}$ will contain $O(pn)$ elements, proving the existence of an ε-net of size $O(pn) = O(\log |\mathcal{S}|/\varepsilon)$. This construction can be made deterministic via derandomization [79, 89].

The above bound on the size of the ε-net can be made independent of $|\mathcal{S}|$ for a very important case of set systems; this brings us to the concept known as the *VC-dimension*. Let $(\mathcal{S}, \mathcal{X})$ be a set system. For a subset $\mathcal{A} \subset \mathcal{X}$, the induced set system $\mathcal{S}_{|\mathcal{A}}$ is defined as $\{S \cap \mathcal{A} | S \in \mathcal{S}\}$; furthermore, we say $\mathcal{A}$ is shattered if the size of the induced set system is $2^{|\mathcal{A}|}$ (the maximum possible). The VC-dimension of the set system $(\mathcal{S}, \mathcal{X})$ is defined as the maximum size of a shattered set.

Theorem 2.2.1. *[94] If the set system $(\mathcal{S}, \mathcal{X})$ has VC-dimension d, then the size of any induced set system $\mathcal{S}_{|\mathcal{A}}$ is $O(m^d)$ where $m = |\mathcal{A}|$. In fact, $|\mathcal{S}_{|\mathcal{A}}| \le \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{d}$ and this bound is tight in the worst case.*

In particular, if we set $\mathcal{A} = \mathcal{X}$ we get $\mathcal{S}_{|\mathcal{X}} = \mathcal{S}$ thus, $|\mathcal{S}| = O(n^d)$. For more details on VC-dimension, the reader is referred to various books and surveys on this topic [94, 99].

This concept has a wide range of applications, for example in machine learning. We only briefly review the ideas that are relevant to computational geometry and specifically to geometric counting.

To see the usefulness of VC-dimension in computational geometry, consider the 2D circle range searching problem with a set $\mathcal{P}$ of n input points and circles as queries. We formulate this problem using the set system $(\mathcal{D}, \mathcal{P})$ in which $\mathcal{D}$ contains all the subsets of $\mathcal{P}$ which can be contained in a circle. By examining small cases, it is easy to see that no point set of size four can be shattered. Thus, this set system has VC-dimension at most three. Usually the set systems obtained from geometric problems in constant dimensions tend to have constant VC-dimension as well.

For the case of set systems with low VC-dimension, the bound on the size of an ε-net can be improved.

Theorem 2.2.2. *[75] There exists an ε-net of size $O(d(1/\varepsilon)\ln(1/\varepsilon))$ for a set system $(\mathcal{S}, \mathcal{X})$ of VC-dimension d. Furthermore, this bound is asymptotically tight.*

Note that the above bound is independent of n and is constant for constant ε.

An ε-net does not provide a true estimate of size; this task is done with the related concept of *ε-approximation*. For a set system $(\mathcal{S}, \mathcal{X})$, an ε-approximation is a set $\mathcal{A} \subset \mathcal{X}$ such that for every $S \in \mathcal{S}$ we have $||\mathcal{A} \cap S|/|\mathcal{A}| - |S|/n| \le \varepsilon$. Notice that this condition implies the difference between $n|\mathcal{A} \cap S|/|\mathcal{A}|$ and $|S|$ is at most εn; in other words, computing $|\mathcal{A} \cap S|$ allows us to approximate $|S|$ with an additive error of εn. Note that an ε-approximation is also an ε-net. The following result yields ε-approximations of constant size for constant ε.

Theorem 2.2.3. *[98, 100] For a set system $(\mathcal{S}, \mathcal{X})$ of VC-dimension d, an ε-approximation of size $O\left((1/\varepsilon)^{2-2/(d+1)}(\log(1/\varepsilon))^{2-1/(d+1)}\right)$ exists.*

A slightly worse bound can obtained by random sampling.

Theorem 2.2.4. *[75] Let $(\mathcal{S}, \mathcal{X})$ be a set system of VC-dimension d. A set $\mathcal{A}$ built by choosing every element of $\mathcal{X}$ with probability $cd\varepsilon^{-2}\log(d\varepsilon^{-1})/n$ is an ε-approximation with constant probability for a sufficiently large constant c.*

Since many counting problems in computational geometry deal with set systems of constant VC-dimension, the above theorem provides a very efficient and simple method to obtain an approximation with additive error of εn.

2.3 Relative Approximations

Approximating ranges does not provide satisfactory solutions for problems such as halfspace range counting and the approximation provided by ε-approximations is too imprecise. Another alternative is to use relative approximations: if the final "count" is k, then any answer between $(1-\varepsilon)k$ and $(1+\varepsilon)k$ is a valid ε-relative approximation given a constant $\varepsilon > 0$. Note that when the value of the count is zero, an ε-relative approximation must return the exact value of the count. Thus, an obvious requirement for a problem to admit an efficient relative approximation algorithm is that there is an efficient algorithm for the emptiness problem. This requirement immediately rules out problems such as simplex range counting and the halfspace range counting in dimensions 4 and above (for both problems it is conjectured that the emptiness problem is almost as hard as the counting variant).

A problem that greatly benefits from relative approximation is halfspace range counting in 3D. To answer halfspace emptiness queries, we can simply compute and store the convex hull of the point set $\mathcal{P}$. A given query halfspace h is non-empty if and only if it intersects the convex hull of $\mathcal{P}$. The convex hull has linear size in 2D and 3D so the emptiness queries can be answered in $O(\log n)$ query time using linear space [60].

Another type of problem that benefits from relative approximation come from robust statistics. Many notations of depth in robust statistics can be regarded as results of counting procedures; for instance, the Tukey depth of a point q can be regarded as counting the minimum number of points in a halfspace which includes q, and the simplicial depth of q can be regarded as counting the number of simplices which contain q. For these definitions of depth, the corresponding emptiness problem can be solved efficiently as well.

Sometimes it is possible to reduce a relative approximate counting problem to an emptiness problem (with some overhead). In the remainder of this chapter, we will discuss these reductions after a brief review of the required probabilistic tools.

2.3.1 Probabilistic tools

We begin this section with a discussion of a well known theorem on the distribution of a sum of independent random variables known as *Chernoff inequality*:

Chernoff Inequality. *Let $X_1, \dots X_n$ be n independent random variables with $|X_i| \leq 1$. For the random variable $X = \sum_{i=1}^{n} X_i$ with variance σ^2 we have*

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

In most applications of the above inequality, all X_i's are identical 0-1 random variables with $\Pr[\,X_i = 1\,] = p$ for a fixed parameter p. The following alternate inequality will be useful.

Lemma 2.3.1. *Let X_i, $1 \leq i \leq n$ be n identically distributed independent 0-1 random variables with $\Pr[\,X_i = 1\,] = p \leq 1/2$ and $\Pr[\,X_i = 0\,] = 1 - p$. For the random variable $X = \sum_{i=1}^{n} X_i$ we have*

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

Proof. We have

$$\sigma^2(X) = \sum_{i=1}^{n} \sigma^2(X_i) = \sum_{i=1}^{n} (p - p^2) = n(p - p^2) \;\geq\; np/2$$

and

$$E(X) = pn.$$

Plugging in these values in Chernoff inequality with $\gamma = 2\alpha\sigma$ yields

$$\begin{aligned} &\Pr[\,|X - E(X)| \geq 2\alpha\sigma^2\,] \leq 2e^{-\alpha^2\sigma^2} \\ \Rightarrow\quad &\Pr[\,|X - E(X)| \geq \alpha np\,] \leq 2e^{-\alpha^2 np/2} \\ \Rightarrow\quad &\Pr[\,|X - E(X)| \geq \alpha E(X)\,] \leq 2e^{-\alpha^2 E(X)/2}. \end{aligned}$$

□

Lemma 2.3.2. *Let c_ε be a sufficiently large constant depending on ε and k be a fixed parameter such that $k \geq c_\varepsilon \log n$. Let $\mathcal{P}$ be a set of n elements, and let $\mathcal{R}$ be a random sample where each element of $\mathcal{P}$ is included with probability $p = (c_\varepsilon \log n)/k < 1$. Given any subset $\mathcal{S} \subseteq \mathcal{P}$ of size k^*, we have $|k^* - |\mathcal{S} \cap \mathcal{R}|/p| \;\leq\; \varepsilon \max\{k, k^*\}$ with high probability.*

Proof. Let $X_i = 1$ if the i-th element of $\mathcal{S}$ is in $\mathcal{R}$, and 0 otherwise; these are independent, identically distributed, random variables. Let $X = |\mathcal{S} \cap \mathcal{R}| = \sum_{i=1}^{k^*} X_i$. According to Lemma 2.3.1,

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

where $E(X) = pk^*$. Let $K := \max\{k, k^*\}$. Substituting $\alpha = \varepsilon pK/E(X)$, we get

$$\Pr[\,||\mathcal{S} \cap \mathcal{R}| - pk^*| \geq \varepsilon pK\,] \;\leq\; 2e^{-\varepsilon^2 p^2 K^2/(2E(X))} \;\leq\; 2e^{-\varepsilon^2 pK^2/(2k^*)} \;\leq\; n^{-c_0}$$

if c_ε is a sufficiently large. □

2.3.2 Reduction to emptiness

Aronov and Har-Peled [21] have observed that an efficient solution for the emptiness problem can be used to build an efficient solution for the relative approximation with small overhead in query and space complexity, provided the problem fits a certain model.

Theorem 2.3.3. *[21, 80] Given a set $\mathcal{P}$ of n geometric objects, suppose a data structure can answer emptiness queries in $Q(n)$ time using $S(n)$ space and $P(n)$ preprocessing time.*

Then in $O_\varepsilon(P(n)\log^2 n)$ time, one can build another data structure that uses $O_\varepsilon(S(n)\log n)$ space and can provide an ε-relative approximation for the number of elements of $\mathcal{P}$ intersected by a query range in $O_\varepsilon(Q(n)\log n \log\log n)$ time.

The correctness is guaranteed with high probability for any fixed query range.

We provide a quick sketch of the idea behind the proof. For a sufficiently large constant C, build $m := C\log n$ random samples $\mathcal{R}_i \subset \mathcal{P}$ in which each element of $\mathcal{P}$ is chosen with probability $p = k^{-1}/2$ in each $\mathcal{R}_i$ (sometimes we will use the notation p-random sample to describe $\mathcal{R}_i$). Implement the emptiness data structure on all point sets $\mathcal{R}_i$. Let q be the query range. Let k' be the number of elements of $\mathcal{P}$ intersected by q. We claim by issuing one emptiness query for each $\mathcal{R}_i$ we can detect whether k' is ε-approximately greater, less than or equal to k. For simplicity, we only consider the case when k' is ε-approximately less than k. It is easily checked the probability that q intersects no element of $\mathcal{R}_i$ is

$$p_{\text{empty}} := (1 - \frac{1}{2k})^{k'} \geq (1 - \frac{1}{2k})^{k} \geq \frac{1}{2}.$$

This event can be detected using the emptiness data structure but the probabilistic guarantee is too weak. Let X_i be the 0-1 random variable corresponding to the result of the emptiness query on $\mathcal{R}_i$ where $X_i = 1$ corresponds to q intersecting no element of $\mathcal{R}_i$. According to what we just said, we have $\Pr[\,X_i = 1\,] = p_{\text{empty}} \geq 1/2$. Let $X = \sum_{i=1}^{m} X_i$ and apply the Chernoff inequality in Lemma 2.3.1. This implies

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

By definition $E(X) = p_{\text{empty}}m$ so if we set C large enough, with high probability we are guaranteed to witness roughly $p_{\text{empty}}m$ emptiness events.

The other cases are similar: If k' is ε-approximately larger than k then we will witness roughly $(1-p_{\text{empty}})m$ non-emptiness events. This enables us to do a binary search among the possible values of k but since there are only $O_\varepsilon(\log n)$ of such values (i.e., only for $k = (1+\varepsilon)^i$, $0 \leq i = O_\varepsilon(\log n)$), we only need $O_\varepsilon(\log\log n)$ binary search steps. This results in the query time of $O_\varepsilon(Q(n)\log n\log\log n)$.

This is a Monte Carlo algorithm, in other words, there is a small chance that the result of the query is incorrect. This reduction has a wide range of applications.

For instance, combined with the solution for the 3D halfspace range emptiness queries with $S(n) = O(n)$ and $Q(n) = O(\log n)$, it provides a data structure using $O_\varepsilon(n \log n)$ space with query time of $O_\varepsilon(\log^2 n \log\log n)$ for the approximate halfspace range counting in 3D. Similar ideas can be used to obtain algorithms for approximate Tukey and regression depth queries. In the next section we will review another general reduction with even better results.

2.3.3 Reduction to small counts

In this section we describe how to approximate a counting problem using a "small count" data structure instead of an emptiness algorithm.

Theorem 2.3.4. *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 $P_{\text{small}}(n)$, $S_{\text{small}}(n)$, and $Q_{\text{small}}(n, \ell)$ be the (expected) preprocessing time, space, and query time of this data structure (assuming these functions are well-behaved).*

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

The algorithm is correct with high probability for any fixed query range.

Proof. For each i, we take a random sample $\mathcal{R}_i$ of $\mathcal{P}$ with sampling probability $p_i = (c_\varepsilon \log n)/k_i$ 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 $\mathcal{R}_i$, with threshold $\ell = c_\varepsilon \log n$. Since the expected size of $\mathcal{R}_i$ 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 Lemma 2.3.2, 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 $|h \cap \mathcal{R}_i|/p_i \le k_i$ or not, i.e., whether $|h \cap \mathcal{R}_i| \le c_\varepsilon \log n$. This can be done by using the small-count data structure for $\mathcal{R}_i$. 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^* with high probability. (We can readjust ε by a constant factor if necessary.) The query time is $O(Q_{\text{small}}(n, c_\varepsilon \log n) \log\log_{1+\varepsilon} n)$.

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

Compared to the reduction offered by Aronov and Har-Peled in Theorem 2.3.3, the "small count" data structure used in our reduction is more difficult to construct than the emptiness data structure; however, our approach introduces no extra factors in space and preprocessing time and only a small factor in the query time.

We can build an efficient small count data structure for the halfspace range counting problem in 3D, using an idea which will reappear later in this thesis. In the dual problem, we must build a small count data structure for the problem of finding the number of hyperplanes that pass below a query point. Let $\mathcal{H}$ be the set of n hyperplanes and q be the query point. Build an ℓ-shallow $O(\ell/n)$-cutting for $\mathcal{H}$ using Lemma 1.1.6. For every prism D in the cutting we store the list of hyperplanes that cross D. There are $O(n/\ell)$ prisms and each is crossed by $O(\ell)$ hyperplanes and thus the total storage cost is linear. If q is contained in a prism D, then the value of the count can be found in $O(\ell)$ time by examining all the hyperplanes crossing D. If this is not the case, then the value of the count is greater than ℓ. This results in a linear-size thresholding algorithm with query time of $Q_{\text{small}}(n, \ell) = O(\log n + \ell)$. 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 $O_\varepsilon(Q_{\text{small}}(n, \log n) \log\log n) = O_\varepsilon(\log n \log\log n)$ expected query time. This is almost optimal, though Monte Carlo.

Note that the algorithm is correct for all queries with high probability (after readjusting c_0), since the number of combinatorially "different" halfspaces is polynomial. We will give still more efficient 3D solutions to approximate halfspace range counting in the next chapter.

Chapter 3

Approximate Halfspace Range Counting

As we discussed, the existing space/query trade-offs for halfspace range counting are not very satisfactory, even in low dimensions. We remind the reader that in 3D (which is the main case of interest in this chapter) for data structures with linear space, it is generally believed that one cannot do better than query time of $O(n^{2/3})$ or, in terms of the actual count k, $O(k^{2/3})$ [38].

Previously, we have seen two general reductions which enable us to attack this problem and obtain some efficient results. In this chapter, we describe several new methods for approximate halfspace range counting in 3D, with still better space and query time.

Variants of approximate halfspace range counting were actually considered early on, for example, in a 1986 paper by Edelsbrunner and Welzl [123], who studied the 2D problem with *additive* instead of relative 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 ε-approximations of constant size by using Theorem 2.2.3 or Theorem 2.2.4. 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, but the case $k = 0$ can be easily solved, hinting that the approximate halfspace range counting problem can also be solved efficiently.

Indeed, that is the case, as was shown in two recent papers. We have already mentioned Aronov and Har-Peled's [21] work, which proposed 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

	Space	Query time	Notes
Aronov & Har-Peled [21]	$O(n \log n)$	$O(\log^2 n \log\log n)$	MC
Aronov & Har-Peled [22]	$O(n \log n)$	$O(\log^2 n)$	MC
Kaplan & Sharir [80]	$O(n \log n)$	$O(\log^2 n)$	MC
Har-Peled & Sharir [73]	$O(n \log^{O(1)} n)$	$O(\log n \log\log n)$	MC
This thesis	$O(n)$	$O(\log n)$	LV

Table 3.1: Results on approximate halfspace range counting. MC and LV stand for Monte Carlo and Las Vegas respectively.

$O(n \log n)$ space and $O(\log^2 n \log\log n)$ query time but is Monte Carlo. The second paper, by Kaplan and Sharir [80], 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 [57] 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 [22], 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. In a third paper, Har-Peled and Sharir [73] among other results discuss 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. A summary of these results is shown in Table 3.

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 was 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.In this chapter, we answer this open question, by giving a new solution with $O(n)$ space and $O(\log(n/k))$ query time in 3D.

3.1 An Optimal Solution

3.1.1 Approximate levels by shallow cuttings

Our optimal solution will involve 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 offered by the reduction in Theorem 2.3.4. The main advantage of this new method is that it is Las Vegas.

We work in dual space, where we want to preprocess a set $\mathcal{H}$ 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, or in other words, the level of q.

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 $O(nk^{3/2})$ [112]. However, we know the total complexity of the k'-level for all $k' = 0, \dots, k$ is $O(nk^2)$ (Theorem 1.1.2) which means that the average complexity of a k'-level with $(1 - \varepsilon)k \le k' \le (1+\varepsilon)k$ is $O_\varepsilon(nk)$. This is still too large for our purposes. We show that a form of approximate k-level exists with complexity $O_\varepsilon(n/k)$ only, which surprisingly is sublinear for non-constant k. The main tool is the modification of Matoušek's shallow cutting lemma presented in Lemma 1.1.6.

We formally define an *ε-approximate $(\le k)$-level* to be a collection of simplices that contains the $(\le (1-\varepsilon)k)$-level and is contained in the $(\le (1+\varepsilon)k)$-level. The size of an ε-approximate $(\le k)$-level is the number of simplices in the collection. Using the shallow cutting lemma, we get:

Lemma 3.1.1. *For any set of n planes in $\mathbb{R}^3$ and a parameter k, there exists an $O(\varepsilon)$-approximate $(\le k)$-level of size $O_\varepsilon(n/k)$.*

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 $(\le k)$-level in $O(\log(n/k))$ time.

Proof. Construct a k-shallow $O(k/n)$-cutting of size $O(n/k)$, covering the $(\le k)$-level by Lemma 1.1.6. For each vertical prism Δ in the cutting, consider the $O(k)$ planes that cross Δ (i.e., the conflict list). We use the cutting theorem (Theorem 1.1.3) to build an ε-cutting of constant size (more precisely, of size $O(\varepsilon^{-3})$) for the conflict list of Δ. This can be done in time linear in the number of planes, i.e., $O(k)$ time and ensures that each subcell $\delta \subset \Delta$ intersects $O(\varepsilon k)$ planes. Summing over all $O(n/k)$ prisms for a given k, this process produces $O_\varepsilon(n/k)$ 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 $\ell_\delta \le k$, we include δ in the approximate $(\le k)$-level and discard the remaining subcells. This construction satisfies the desired property, because each remaining subcell δ intersects $O(\varepsilon k)$ planes and so the levels of any two points in δ differ by at most $O(\varepsilon k)$. But δ contains a point in $(\le k)$-level, so, all the points of δ have level at most $k(1 + O(\varepsilon))$. Computing ℓ_δ corresponds to computing the number of hyperplanes below an arbitrary point in δ. This is equivalent to a halfspace range counting query in primal space which can be solved using a halfspace range reporting algorithm in $O(\log n + k)$ time. So, the total time for this step is $O(n(\log n + k)/k)$, which sums to $O_\varepsilon(n \log n)$.

For a given query point q, we can find the vertical prism Δ in the k-shallow $O(k/n)$-cutting containing q in $O(\log(n/k))$ time by planar point location. Afterwards, we can find the subcell $\delta \subset \Delta$ containing q in constant time and see if δ was included in the approximate $(\le k)$-level. □

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

Theorem 3.1.2. *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 $O(\sum_i n/k_i) = O_\varepsilon(n)$.

To approximate the number k^* of planes 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. This can be done in $O(\log(n/k_i))$ 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 [37], 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.

3.1.2 Combine with randomized incremental construction

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

Lemma 3.1.3. *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 [80], 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 3.1.4. *Let $h_1, \ldots, h_n$ be a random permutation of a set $\mathcal{H}$. Given any subset $\mathcal{S} \subseteq \mathcal{H}$ of size k^*, let j be the smallest index with $h_j \in \mathcal{S}$. Let $k = n/j$. Then the probability that $k^* < k/b$ or $k^* > bk$ is $O(1/b)$.*

Proof. The event $k^* < k/b$ implies that $j < n/bk^*$ and so at least one of the planes $h_1, \ldots, h_{n/(bk^*)}$ is in $\mathcal{S}$; this happens with probability at most $n/bk^* \cdot k^*/n = 1/b$. On the other hand, $k^* > bk$ implies that $j > bn/k^*$ and so $h_1, \ldots, h_{bn/k^*}$ are all not in $\mathcal{S}$; 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 $k = n/j$ 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 the method used in Section 3.1.1: 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 3.1.5. *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 $O_\varepsilon(\log(n/k^*))$ 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 Theorem 3.1.2, augmented with the data structure in Lemma 3.1.3 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 $k = n/j$ and suppose $k_s \leq k < k_{s+1}$. 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 $O(\log(n/k_{s\pm i}))$ 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 $k_{s\pm i}$ is at most $O(1/(1+\varepsilon)^i)$ by Observation 3.1.4. 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 there is no index j such that h_j lies below q (i.e., $k < \log n$)? Here, we start the search with $k_s \leq \log n < k_{s+1}$. If k^* is

$O(\varepsilon)$-approximately less than $k_s \leq \log n$, we can directly answer the query using a small-count data structure in $Q_{\text{small}}(n, \log n) = O(\log n) = O(\log(n/k^*))$ time. Otherwise, the probability that k^* is $O(\varepsilon)$-approximately $k_{s+i} \approx (1+\varepsilon)^i \log n$ is even smaller by Observation 3.1.4 since $k < \log n$, and the same query bound holds. □

3.2 Approximate Regression Depth Queries in 2D

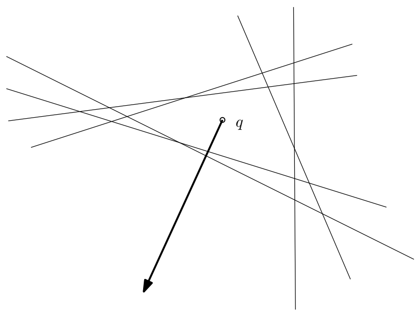


Figure 3.1: A point q with undirected depth two. Any ray through q intersects at least two lines.

The problem of computing the regression depth of a query line in 2D reduces to the following in dual space: Given a set $\mathcal{H}$ 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 [31], call k^* the *undirected depth* of q (Figure 3.1). Call the locus of all points of undirected depth at most k the ($\leq k$)*-envelope*. Call the boundary of this locus the k*-envelope* [120]. In measuring the complexity of a k-envelope or a polygonal chain, we will include all vertices in the arrangement that lie on the chain, even 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 the method used in Section 3.1.1 to solve the problem. We first need an analog of Lemma 3.1.1. Define an ε*-approximate* ($\leq k$)*-envelope* to be a region that contains the ($\leq (1-\varepsilon)k$)-envelope and is contained in the ($\leq (1+\varepsilon)k$)-envelope. We prove the existence of an ε-approximate ($\leq k$)-envelope of size $O(n\alpha(k)/k)$, where $\alpha(\cdot)$ is the inverse Ackermann function.

Lemma 3.2.1. *For any set $\mathcal{H}$ 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 [63], since the unbounded cells intersect the line $x = -M$ or $x = M$ for a sufficiently large M.

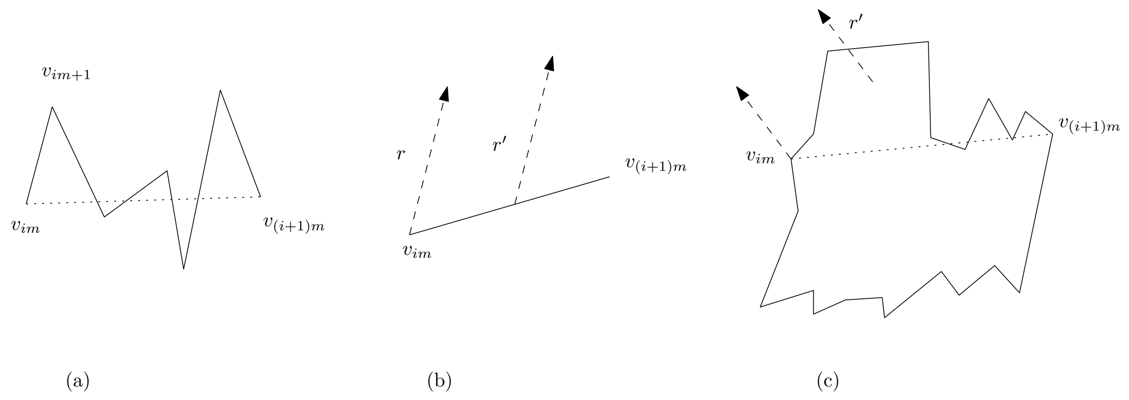


Figure 3.2: (a) An edge of a simplified polygon (in dotted line). (b) Proof of Lemma 3.2.2(i). (c) Proof of Lemma 3.2.2(ii).

We apply Clarkson and Shor's technique [56], similar to one used to prove Lemma 1.1.1, by picking a random sample in which each line is chosen with probability $1/k$. The lemma follows from the fact that a vertex v of the arrangement which lies in the $(\leq k)$-envelope of $\mathcal{H}$ has $\Omega(1/k^2)$ chance of surviving in the sample and being a 0-envelope vertex. □

In 2D, one can obtain an approximate level of size $O(n/k)$ by taking an exact level of size $O(n)$ and applying a "simplification" process (e.g., as in [123]). 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, we do not know if they are always connected and thus may consist of multiple polygons [120]; 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 $A' = \langle v_0, v_m, v_{2m}, \ldots, v_{m\lfloor t/m \rfloor}, v_t \rangle$ (see Figure 3.2(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 non-simple 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 3.2.2. *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 3.2(b)). Then r hits at least k lines since v_{im}

has undirected depth k. Any line of $\mathcal{H}$ 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 $\langle v_{im}, v_{im+1}, \ldots, v_{(i+1)m}\rangle$ (as a reminder, we allow vertices to create angles of π on the boundary), but there are only m such vertices. So, r' hits at least $k - m$ lines.

(ii) Let p be outside C'. If p is outside C, then it has undirected depth at most k. Thus, we may assume p is inside a polygon $A \in C$ for some $A = \langle v_0, v_1, \ldots, v_t\rangle$ but outside the simplified polygon 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 3.2(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 of $\mathcal{H}$ 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. Consider two chains $A_i = \langle v_{im}, v_{im+1}, \ldots, v_{(i+1)m}\rangle$ and $B_j = \langle w_{jm}, w_{jm+1}, \ldots, w_{(j+1)m}\rangle$. Since the original k-envelope C does not have crossings, one of two possibilities must hold: A_i intersects the segment $w_{jm}w_{(j+1)m}$, in which case we give $w_{jm}w_{(j+1)m}$ one charge; or B_j 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.

□

Sadly, the above simplification procedure does not give us an ε-approximate ($\leq k$)-envelope of size $O_\varepsilon(n/k)$ since the segments obtained by the simplification procedure can self-intersect. Fortunately, it is possible to control the number of self-intersections using the combinatorial fact stated below.

Lemma 3.2.3. *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 [74], 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., [59]): 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 3.2.4. *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 $O_\varepsilon(n\alpha(k)/k)$.*

Proof. According to Lemma 3.2.1, 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 3.2.2(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 $N = O_\varepsilon(n/k)$ and the number of crossings is $X = O_\varepsilon(Nk)$ by Lemma 3.2.2(iii). By Lemma 3.2.3, the complexity of the outer face is $O_\varepsilon(n\alpha(k)/k)$.
□

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

Theorem 3.2.5. *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 build an $(\varepsilon/3)$-approximate $(\leq k_i)$-envelope E_i for each i by Lemma 3.2.4. The total size of the E_i's is $O_\varepsilon(\sum_i n/k_i\alpha(k_i)) = O_\varepsilon(n)$. 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. □

3.3 Approximate Tukey Depth Queries

Consider a point set $\mathcal{P}$ of size n and let $\mathcal{A}$ be the arrangement formed by the set $\overline{\mathcal{P}}$ of hyperplanes dual to $\mathcal{P}$. A query point q has Tukey depth at least m if and only if every halfspace through q contains at least m points of the point set $\mathcal{P}$. In dual space this means there are at least m hyperplanes passing above and below every point of the hyperplane $\overline{q}$; in other words, the hyperplane $\overline{q}$ lies between the m-level and $(n-m)$-level of $\mathcal{A}$. Thus, the problem of computing the Tukey depth of a query point reduces to the following in dual space: Given a set $\mathcal{H}$ of n hyperplanes in $\mathbb{R}^d$, preprocess them in a data structure so that given any query hyperplane q, we can find the smallest value k^* such that q intersects the $(\leq k^*)$-level in the arrangement of H.

First, we investigate this problem in 3D. We can either use the general reduction to emptiness obtained by Aronov and Har-Peled (Theorem 2.3.3) or the reduction to small counts offered by us (Theorem 2.3.4) or the methods used in Section 3.1.1 to attack this problem. The last option has the advantage of giving us a Las Vegas algorithm (instead of a Monte Carlo one) with the same performance as one obtained using Theorem 2.3.4.

Theorem 3.3.1. *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 $i = 1, \dots, \lceil \log_{1+\varepsilon} n \rceil$, and construct an approximate $(\leq k_i)$-level L_i for each i by Lemma 3.1.1, as in the proof of Theorem 3.1.2. For each i, we compute the upper hull U_i of the $O(n/k_i)$ vertices of L_i. This takes time $O(\sum_i n/k_i \log n/k_i) = O_\varepsilon(n \log n)$.

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 $O(\log n \log \log_{1+\varepsilon} n)$. $\square$

The problem in 2D can be solved in an identical way, although there is no need to perform $O_\varepsilon(\log\log n)$ steps of binary search. In the plane, we can simply overlay all the upper hulls U_i and replace the approximate binary search steps with a single point location query.

Theorem 3.3.2. *One can preprocess a planar 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 $O_\varepsilon(\log n)$ worst-case time. The query algorithm is always correct.*

Chapter 4

Approximate Simplicial Depth

Unlike the Tukey and regression depth measures, approximating the simplicial depth of a given query point appears to be more difficult. To begin with, this problem does not fit in the framework of Theorems 2.3.3 and 2.3.4. While the ε-approximation techniques are applicable to this problem, they do not yield any relative approximations. We do not know of any previous results on approximate simplicial depth queries. We can only mention the result of Eppstein et al. [28], which provides an efficient way to approximate the depth of a query point in 2D in a restricted streaming model but only with an additive error of εn^3.

Technical difficulties. As we have repeatedly seen in the previous chapters, one standard technique to approximate various geometric measures is the use of uniform random samples combined with Chernoff type inequalities; however, these existing techniques seem insufficient to solve our problem. One particular troubling situation is depicted in Figure 4.1. In this figure no high probability bound can be achieved for the simplicial depth of q in a uniform random sample despite the fact that q is far from being an outlier (it has Tukey depth of $\Theta(n^{1/3})$).

We overcome such difficulties by using a diverse set of tools and techniques such as martingales and Azuma's inequality, which were seldom used in the previous papers in the area.

For a set $\mathcal{S}$ and a point q, we denote the simplicial depth and the Tukey depth of q in $\mathcal{S}$ with $\sigma_{\mathcal{S}}(q)$ and $\tau_{\mathcal{S}}(q)$ respectively. We might omit the subscript $\mathcal{S}$ if there is no fear ambiguity. For a point $p \in \mathcal{S}$, we use the notation $\omega_{q,\mathcal{S}}(p)$ (called the *weight* of p) to denote the number of simplices formed by points of $\mathcal{S}$ that contain q with p as one of the vertices. Clearly we have, $\sum_{p\in\mathcal{S}} \omega_{q,\mathcal{S}}(p) = (d+1)\sigma_{\mathcal{S}}(q)$. For simplicity, sometimes we will only use $\omega(p)$ if q and $\mathcal{S}$ can be deduced from the context. We may use the notation $f \ll g$ and $f \gg g$ to denote $f = O(g)$ and $f = \Omega(g)$ respectively.

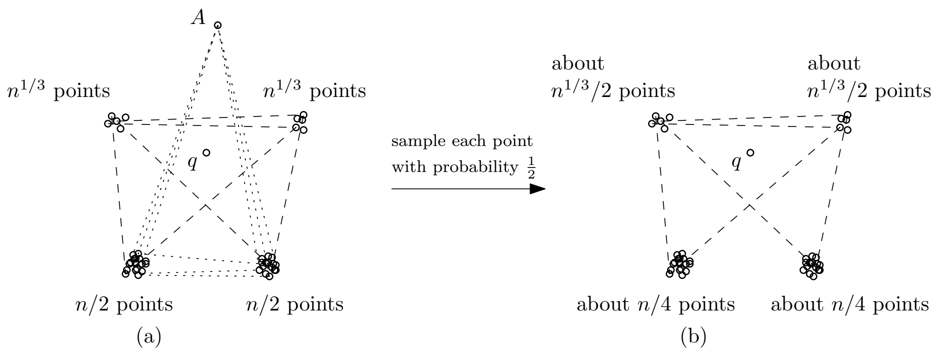


Figure 4.1: (a) A point with simplicial depth $\Theta(n^2)$ in a point set of $\Theta(n)$ points. (b) The random sample misses A and the $n^2/4$ triangles that share A as a vertex; the simplicial depth of q is now $O(n^{5/3})$.

4.1 Bounding the Simplicial Depth with Tukey Depth

We first examine the relationship between the simplicial depth of a point and its Tukey depth. We start with the following easy lemma.

Lemma 4.1.1. *Let $x \in \mathbb{R}^d$ be a point which is contained in a simplex S. For every point $p \in \mathbb{R}^d$, there exists a unique vertex v of S such that the simplex $S \cup \{p\} \setminus \{v\}$ contains x.*

Proof. Consider the ray r from x in the direction $x - p$. Since x is inside S, the ray intersects a unique boundary face of S. The vertex v opposite to this face is the unique vertex claimed in the lemma. □

The following bound relates the two notions of depth.

Lemma 4.1.2. *For any point set $\mathcal{P} \subset \mathbb{R}^d$ and any $q \in \mathbb{R}^d$ we have $\sigma_{\mathcal{P}}(q) = \Omega(|\mathcal{P}|\tau^d_{\mathcal{P}}(q))$ and $\sigma_{\mathcal{P}}(q) = O(|\mathcal{P}|^d \tau_{\mathcal{P}}(q))$.*

Proof. If $\tau_{\mathcal{P}}(q) = 0$ then q is outside the convex hull of $\mathcal{P}$ and there is nothing left to prove. So assume $\tau_{\mathcal{P}}(q) > 0$. By Carathéodory's theorem [33, 34, 71] there exists a simplex S_1 formed by $d+1$ points of $\mathcal{P}$ which contains q. Removing S_1 from $\mathcal{P}$ reduces the Tukey depth of q by at most $d+1$ (in fact at most d, but this can only improve the constant factors in the lemma). By repeating this operation we can find m disjoint subsets $\mathcal{S}_1, \ldots, \mathcal{S}_m \subset \mathcal{P}$ of size $d+1$, $m \geq \tau_{\mathcal{P}}(q)/d+1$, such that the simplex formed by each $\mathcal{S}_i$ contains q.

Let $\mathcal{A} = \bigcup_i^m \mathcal{S}_i = \Theta(m)$. Bárány [29] proves that $\sigma_{\mathcal{A}}(q) = \Omega(m^{d+1})$. Denote by $\mathcal{A}_\Delta$ the set of all the simplices which contain q with vertices from $\mathcal{A}$. We can assume $|\mathcal{A}| \leq |\mathcal{P}|/2$, otherwise $\mathcal{A}_\Delta$ already contains $\Omega(|\mathcal{P}|^{d+1})$ simplices and there is nothing to prove. By Lemma 4.1.1, for every $p \in \mathcal{P} \setminus \mathcal{A}$, we can produce $\Omega(|\mathcal{A}_\Delta|/|\mathcal{A}|) = \Omega(m^d) = \Omega(\tau^d_{\mathcal{P}}(q))$ different simplices with p as a vertex. For two points p and p' in $\mathcal{P} \setminus \mathcal{A}$ the corresponding simplices are distinct. In total they produce $\Omega(|\mathcal{P}|\tau^d_{\mathcal{P}}(q))$ different simplices.

To see the upper bound, consider a halfspace h which passes through q and contains $\tau_{\mathcal{P}}(q)$ points. Every simplex containing q must have at least one point from this halfspace, and the maximum number of such simplices is $|\mathcal{P}|^d\tau_{\mathcal{P}}(q)$. $\square$

Remarks. The bounds mentioned in the above lemma are tight. To see the tightness of the upper bound, consider a simplex S and a point q inside it. Replace one vertex of the simplex with a cluster of m points placed closely to each other, and replace all the remaining vertices with clusters of n points with $m \leq n$. The resulting point set $\mathcal{P}_1$ contains $\Theta(n)$ points with $\sigma_{\mathcal{P}_1}(q) = mn^d$ and $\tau_{\mathcal{P}_1}(q) = m$. The tightness of the lower bound is realized by a very similar construction but using clusters of size m at every vertex except one, and using a cluster of size n at the remaining vertex. The resulting point set $\mathcal{P}_2$ contains $\Theta(n)$ points with $\sigma_{\mathcal{P}_2}(q) = m^d n$ and $\tau_{\mathcal{P}_2}(q) = m$.

4.1.1 Properties of random samples

Most papers that deal with approximations use Chernoff-type inequalities to obtain their desired high probability bounds. The dependence among random variables corresponding to the simplices prevents us from using any inequality that deals with independent random variables. Thus, we need to turn to more complicated concentration bounds, such as Azuma's inequality which deals with martingales: a martingale is a series of random variables $X_1, \ldots, X_n$ such that $E(X_{i+1}|X_1, \ldots, X_i) = E(X_i)$.

Azuma's Inequality. *Suppose $\{X_k\}_{k=0}^n$ is a martingale with the property that $|X_{i+1} - X_i| \leq c_i$. Then*

$$\Pr[\,|X_n - X_0| \geq t\,] \leq e^{-\frac{t^2}{2\sum_{k=1}^n c_k^2}}.$$

The following lemma captures some of the useful properties of the simplicial depth of a given point in a uniform random sample.

Lemma 4.1.3. *Let $\mathcal{P} \subset \mathbb{R}^d$ and $q \in \mathbb{R}^d$ be a point set of size n and an arbitrary point respectively. Define $M := \max_{p\in\mathcal{P}} \omega_{q,\mathcal{P}}(p)$ and let t be an arbitrary parameter. For any α-sample $\mathcal{S}$ from $\mathcal{P}$, $1/2 \leq \alpha \leq 1$, we have $\Pr[\,|E(\sigma_{\mathcal{S}}(q)) - \sigma_{\mathcal{S}}(q)| \geq \gamma(E(\sigma_{\mathcal{S}}(q)) + rM)\,] \leq e^{-\Omega(\gamma^2 r)}$.*

Proof. We build a martingale in the usual way of revealing presence or absence of points of $\mathcal{S}$ in an arbitrary order. Let $p_1, \ldots, p_n$ be the points of $\mathcal{P}$. Let X_i be the random variable corresponding to the expected number of simplices containing q where the expectation is taken over the points $p_{i+1}, \ldots, p_n$ (i.e., X_i is a function of our random choices regarding $p_1, \ldots, p_i$). According to this definition, X_n is equal to $\sigma_{\mathcal{S}}(q)$ (when all the values of random variables have been revealed) while X_0 is equal to $E(\sigma_{\mathcal{P}}(q))$. Since every simplex survives with probability α^{d+1} in the random sample $\mathcal{S}$, we have $X_0 = E(\sigma_{\mathcal{P}}(q)) = \alpha^{d+1}\sigma_{\mathcal{P}}(q)$.

It is not difficult to see that the sequence of random variables $X_0, \ldots, X_n$ has the martingale property. By our assumptions, each point can participate in at most M simplices, and so revealing the value of each point brings about $O(M)$ changes to the expected value of X. Define $c_i := |X_{i+1} - X_i|$. We have

$$\begin{aligned}
X_n &= \sigma_{\mathcal{S}}(q)\\
X_0 &= E(\sigma_{\mathcal{S}}(q)) = \alpha^{d+1}\sigma_{\mathcal{P}}(q)\\
c_i &= |X_{i+1} - X_i| \ll \omega_q(p_{i+1}) \leq M\\
\sum_{i=1}^{n} \omega_q(p_i) &= (d+1)\sigma_{\mathcal{P}}(q) \ll X_0\\
\sum_{i=1}^{n} c_i &\ll X_0.
\end{aligned}$$

Let r be an arbitrary parameter. According to Azuma's inequality, we have

$$\Pr[\,|X_n - X_0| \geq \gamma(X_0 + rM)\,] \leq e^{\frac{-\gamma^2(X_0+rM)^2}{\sum c_i^2}} \leq e^{\frac{-\gamma^2(X_0+rM)^2}{M\sum c_i}} \leq e^{-\Omega(\gamma^2 r)}.$$

□

Since $\alpha = \Theta(1)$, we can set $r = \Omega(\gamma^{-2}\log n)$ and obtain the following corollary.

Corollary 4.1.4. *Let $\mathcal{S}$ be a α-random sample of $\mathcal{P}$, $1/2 \leq \alpha \leq 1$. The simplicial depth of q in $\mathcal{P}$ can approximated by $\sigma_{\mathcal{S}}(q)/\alpha^{d+1}$. The maximum additive error with high probability is $O(\gamma\sigma_{\mathcal{P}}(q) + M\gamma^{-1}\log n)$ where $M := \max_{p\in\mathcal{P}}\omega_q(p)$.*

Remarks. There are many articles that deal with concentration of functions of independent random variables and the results obtained in many of them can possibly be applied to our problem as well. For instance, we can formulate our problem easily in Van Vu's polynomial model [84] although the final bound will be slightly worse than what we have obtained above. We can also use Talagrand's inequality [114] and obtain an equivalent result to that of Lemma 4.1.3. These imply the result of Lemma 4.1.3 might not be optimal in terms of dependence on γ although the situation depicted in Figure 4.1 implies we cannot hope to get a better bound than $e^{-\Theta(\gamma r)}$.

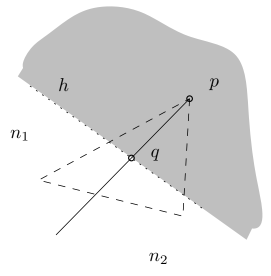


Figure 4.2: There are at most $x = C'\tau_{\mathcal{P}}(q)$ points above h. The number of dashed triangles is at least $(n_1 - x)(n_2 - x)$ and at most n_1n_2; these two values are very close if x is small in comparison to n_1 and n_2.

4.2 Approximating the simplicial depth in 2D

In this section we consider the 2D case of the problem, which can be stated as a triangle counting problem: given a set of n points $\mathcal{P}$ in the plane, store them in a data structure capable of approximating the number of triangles formed by points of $\mathcal{P}$ which contain a query point q.

The observation that different points of the input can contribute different values to the simplicial depth is a major obstacle that needs to be overcome. We can describe our general strategy to deal with this difficulty using the following toy problem.

Problem 4.2.1. *Assume n closed boxes $B_1, \ldots, B_n$ with box B_i containing b_i objects are given such that $b_1 \geq b_2 \geq \cdots \geq b_n$. Find an ε-approximation of the sum $s := \sum_{i=1}^{n} b_i$ by opening a small number of boxes.*

We propose the following strategy to solve the problem. Let k be a small (polylogarithmic) parameter. Compute $s_1 = \sum_{i=1}^{k} b_i$ by opening k boxes. For the remaining boxes, discard each box with probability 1/2 and then recursively compute the approximate number of objects contained in the remaining boxes. Let s_2 be this value. Return $s_1 + 2s_2$. It is easy to see that the algorithm works in the expected sense since the expected value of s_2 is $(s - s_1)/2$. It is not difficult so show that by setting k to be a polylogarithmic factor the final result is correct with high probability. It is also easy to see that the algorithm opens $O(k \log n)$ boxes.

We follow a similar strategy for our problem. The planar geometry of the problem provides us with the following simple observation which plays a crucial role in our algorithm.

Observation 4.2.1. *Let $\mathcal{P} \subset \mathbb{R}^2$ be a set of n points, $q \in \mathbb{R}^2$ an arbitrary point and h a halfplane containing x points with q on its boundary. Consider the line $\overline{pq}$ for a point $p \in h$ and assume it partitions $\mathcal{P}$ into two sets of sizes n_1 and n_2. If $n_1, n_2 \geq C_\varepsilon x$ then n_1n_2 is a relative ε-approximation of $\omega_q(p)$ in which C_ε is a constant depending on ε.*

Furthermore, with $O_\varepsilon(n)$ space it is possible to obtain a 2ε-approximation of $\omega_q(p)$ in $O_\varepsilon(\log n)$ time.

Proof. It can be argued from Figure 4.2 that there are at least $(n_1 - x)(n_2 - x)$ triangles containing q that have p as a vertex. On the other hand, if a triangle with p as one vertex contains q, then it must have one vertex from either side of line $\overline{pq}$. Thus, the maximum number of such triangles is $n_1 n_2$ which means

$$(n_1 - x)(n_2 - x) \leq \omega_{q,\mathcal{P}}(p) \leq n_1 n_2.$$

The assumption $n_1, n_2 \geq C_\varepsilon x$ implies $xn_1, xn_2 \leq C_\varepsilon^{-1} n_1 n_2$ which combined with the above inequality yields

$$n_1 n_2 - 2C_\varepsilon^{-1} n_1 n_2 \leq (n_1 - x)(n_2 - x) \leq \omega_{q,\mathcal{P}}(p) \leq n_1 n_2.$$

Thus, $n_1 n_2$ is a relative ε-approximation of $\omega_{q,\mathcal{S}}(p)$ if C_ε is chosen to be sufficiently large. Using the approximate halfspace range counting data structure developed in the previous section, it is possible to obtain a 2ε-approximations of the term $n_1 n_2$. □

For a point $p \in \mathcal{P}$, we call the minimum number of points of $\mathcal{P}$ at either side of line $\overline{pq}$ its *dissection value with respect to q and $\mathcal{P}$* (or dissection value for short). The above observation intuitively implies that under the right conditions, approximating the weight of a point (i.e., $\omega_{q,\mathcal{P}}(p)$) is equivalent to approximating its dissection value. It also means (again under the right conditions) that a larger dissection value implies a larger weight. We now return to solve the main problem of this chapter.

Consider a series of random samples $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_r$ in which $\mathcal{S}_0 = \mathcal{P}$ and $\mathcal{S}_{i+1}$ is a 1/2-random sample from $\mathcal{S}_i$. For technical reasons, we solve a slightly more general problem in which an additional set of points $\mathcal{D}$ (called the discard set) is given and we are asked to approximate the simplicial depth of q in $\mathcal{S}_i \backslash \mathcal{D}$. Initially, $\mathcal{D}$ is set to $\emptyset$. Throughout the algorithm we maintain the invariant that $\mathcal{D} \leq 2m$ for a sufficiently large parameter m with high probability; if it fails, we can switch to a brute-force algorithm and the expected cost of this will still be $O(1)$. Also, if $\tau_{\mathcal{S}_i \backslash \mathcal{D}}(q) = \Omega(|\mathcal{S}_i|)$ then the simplicial depth can be approximated using ε-approximations (since an additive error is acceptable). By our invariant $|\mathcal{D}|$ is small, thus we can assume

$$\tau_{\mathcal{S}_i \backslash \mathcal{D}}(q) = O(|\mathcal{S}_i|) \text{ and } \tau_{\mathcal{S}_i}(q) = O(|\mathcal{S}_i|). \tag{4.1}$$

This can be tested efficiently using our data structure for approximate planar Tukey depth queries implemented on $\mathcal{S}_i$ since $\mathcal{D}$ is assumed to have few points by the invariant. The main steps of our algorithm, excluding the failure of the invariant and the above easy case, are presented in Algorithm 1. Below, C and C_ε are constants which must be chosen sufficiently large. **ApproxRangeCount** is an approximate halfspace range counting data structure and **ApproxRangeCount**($\mathcal{S}$,

ℓ, "Above/Below", ε) finds a relative ε-approximation of the number of points of $\mathcal{S}$ lying above or below a given line ℓ.

The procedure **small** (which shall be discussed later) kicks in when the Tukey depth of the query point is small ($\tau_{\mathcal{S}_i}(q) \leq C_\varepsilon m$) and returns a relative approximation of the simplicial depth. For the rest of this analysis we assume

$$\tau_{\mathcal{S}_i}(q) \geq C_\varepsilon m. \tag{4.2}$$

By choosing C_ε large enough, this implies $\tau_{\mathcal{S}_i}(q)$ is ε-approximately equal to $\tau_{\mathcal{S}_i \setminus \mathcal{D}}(q)$ and $\omega_{q,\mathcal{S}_i}(p)$ is ε-approximately equal to $\omega_{q,\mathcal{S}_i \setminus \mathcal{D}}(p)$ for all points p with dissection value greater than $C_\varepsilon \tau_{S_i}(p)$. We call the value of $\omega_{q,\mathcal{S}_i}(p)$ the *weight* of p and use *approximate weight* to refer to its ε-approximation.

The procedure **dissectionReport** returns a halfplane h containing $O(\tau_{\mathcal{S}_i}(q))$ points with q on its boundary and a set $\mathcal{A} \subset \mathcal{S}_i \cap h$ of size at least $3m$. Let x be the number of points in h. Intuitively, if h contains more than $3m$ points with dissection value greater than $C_\varepsilon x$, then **dissectionReport** must report $3m$ points with "approximately" the maximum dissection value. Otherwise, it reports all the points with dissection value greater than $C_\varepsilon x$ and some other arbitrary points. To be precise, for every $p \in \mathcal{A}$ either the dissection value of p is ε-approximately less than $C_\varepsilon x$ or the dissection value of p is ε-approximately greater than the ($3m$)-th largest dissection value in h. The latter also implies the weight of p is approximately greater than the ($3m$)-th largest weight in h. Finding $\mathcal{A}$ is not easy at all and we shall discuss the details at the end of this section.

Algorithm 1 ApproxSimplicialDepth($\mathcal{S}_i$, q, $\mathcal{D}$)

```
 1: $m \leftarrow C\varepsilon^{-3} \log^3 |\mathcal{P}|$
 2: if $\tau_{\mathcal{S}_i}(q) \leq C_\varepsilon m$ then
 3:     call small($\mathcal{S}_i$, $q$)
 4: else
 5:     $(h, \mathcal{A}) \leftarrow$ dissectionReport($\mathcal{S}_i$, $q$)
 6:     $\mathcal{T} \leftarrow$ sort($\mathcal{A} \setminus \mathcal{D}$)
 7:     for $j = 1$ ; $j \leq m$ do
 8:         $n_1 \leftarrow$ ApproxRangeCount($\mathcal{S}_i$, $\overline{a_j q}$, "Above", $\varepsilon$)
 9:         $n_2 \leftarrow$ ApproxRangeCount($\mathcal{S}_i$, $\overline{a_j q}$, "Below", $\varepsilon$)
10:         if $n_1$ or $n_2 < C_\varepsilon \tau_{\mathcal{S}_i}(q)$ then
11:             Break the for loop.
12:         end if
13:         $\hat{s} \leftarrow \hat{s} + n_1 n_2$
14:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{a_j\}$
15:     end for
16:     $\mathcal{D} \leftarrow \mathcal{D} \cap \mathcal{S}_{i+1}$
17:     Return $\hat{s}$ + $8 \cdot$ ApproxSimplicialDepth($\mathcal{S}_{i+1}$, $q$, $\mathcal{D}$)
18: end if
```

For each $\mathcal{S}_i$ we implement data structures capable of answering approximate halfspace range counting queries and approximate Tukey depth queries (see Chapter 3). Using techniques discussed in the previous chapter, lines (2), (8) and (9) can be done in $\tilde{O}(1)$ time. At line (6), the procedure **sort**$(\mathcal{A} \setminus \mathcal{D})$ sorts the points of $\mathcal{A} \setminus \mathcal{D}$ decreasingly according to their approximate weights. At this point, all but one of the requirements of Observation 4.2.1 are satisfied; line (10) checks for this last requirement. Line (13) uses Observation 4.2.1 to 2ε-approximate $\omega(a_i)$. Thus, $\hat{s}$ is 2ε-approximately equal to the sum of weights of the points processed in the for loop. Finally, a recursive call to $\mathcal{S}_{i+1}$ computes the simplicial depth in a smaller input excluding the points processed in the for loop. So, assuming that line (5) can be completed in $\tilde{O}(1)$ time, it is clear that the total query time is $\tilde{O}(1)$ and it remains to prove the correctness of the algorithm.

Let t be the value of the variable j at the end of the for loop and $\mathcal{T}$ be the set $\{a_1, \dots, a_t\}$. $\hat{s}$ is a relative 2ε-approximation of $s := \sum_{j=1}^{t} \omega_{q,\mathcal{S}}(a_j)$ and thus an 3ε-approximation of $\sum_{j=1}^{t} \omega_{q,\mathcal{S}\setminus\mathcal{D}}(a_j)$; however, s counts the number of triangles containing q with at least one vertex from $\mathcal{T}$ with the triangles having two points of $\mathcal{T}$ being double counted. The assumption 4.1 implies during every execution of line (13) we have $n_1 + n_2 \geq |\mathcal{S}_i|/2$ (since n_1 and n_2 are approximations of number of points on either side the corresponding line). Thus, $n_1 n_2 \geq |\mathcal{S}_i| C_\varepsilon \tau_{\mathcal{S}_i}(q)/4$ and $s \geq t|\mathcal{S}_i| C_\varepsilon \tau_{\mathcal{S}_i}(q)/4$. The number of triangles having two points from $\mathcal{T}$ is at most $|\mathcal{T}|^2 |\mathcal{S}_i| = t^2 |\mathcal{S}_i|$. By assumption 4.2, this is negligible in comparison to $\hat{s}$ since

$$t^2|\mathcal{S}_i| \leq t|\mathcal{S}_i| m \leq t|\mathcal{S}_i| \tau_{\mathcal{S}_i}(q) C_\varepsilon^{-1} \leq s C_\varepsilon^{-2}.$$

Thus,

$$\sigma_{\mathcal{S}_i\setminus\mathcal{D}}(q) = \hat{s} + s_{err} + \sigma_{\mathcal{S}_i\setminus(\mathcal{D}\cup\mathcal{T})}(q) \quad \text{where} \quad |s_{err}| \leq 4\varepsilon s. \tag{4.3}$$

We know for every $p \in \mathcal{A}$, either the dissection value of p is ε-approximately less than $C_\varepsilon x$ or the dissection value of p is ε-approximately greater than the $(3m)$-th largest dissection value in h. Thus, for any point $p \in h \setminus (\mathcal{D} \cup \mathcal{T})$, either $\omega(p) := \omega_{q,\mathcal{S}_i\setminus\mathcal{D}} = O(\hat{s}/m)$ or the dissection value of p is at most $2C_\varepsilon^{-1}\tau_{\mathcal{S}_i}(q)$ (otherwise, it would have been included in $\mathcal{T}$). This implies

$$M := \max_{p \in h\setminus(\mathcal{D}\cup\mathcal{T})} \omega(p) \ll \max\{C_\varepsilon^{-1}\tau_{\mathcal{S}_i}(q)|\mathcal{S}_i|, \frac{\hat{s}}{m}\} \ll C_\varepsilon^{-1}\tau_{\mathcal{S}_i}(q)|\mathcal{S}_i| + \frac{\hat{s}}{m}. \tag{4.4}$$

Using Corollary 4.1.4 with parameter $\gamma = \varepsilon/\log n$ and substituting M from (4.4), we get

$$\begin{aligned}
\sigma_{\mathcal{S}_i\setminus(\mathcal{D}\cup\mathcal{T})}(q) &= 8\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q) + f_{err} \\
|f_{err}| &\ll \gamma\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q) + M\gamma^{-1}\log n \\
&\ll \gamma\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q) + \gamma^{-1}C_\varepsilon^{-1}\tau_{\mathcal{S}_i}(q)|\mathcal{S}_i|\log n + \gamma^{-1}\frac{\hat{s}}{m}\log n.
\end{aligned}$$

Since $m = \Theta(\gamma^{-2}\varepsilon^{-1}\log n)$, we have $\gamma^{-1}\frac{\hat{s}}{m}\log n \ll \varepsilon\gamma\hat{s}$. Furthermore, according to Lemma 4.1.2

$$\sigma_{\mathcal{S}_i\setminus(\mathcal{D}\cup\mathcal{T})}(q) \gg (|\mathcal{S}_i| - (|\mathcal{D}| + |\mathcal{T}|))\tau^2(q) \gg |\mathcal{S}_i|\tau(q)^2 \gg \\ |\mathcal{S}_i|\tau(q)m \gg \gamma^{-2}\varepsilon^{-1}\tau_{\mathcal{S}_i}|\mathcal{S}_i|(q)\log n.$$

In other words,

$$\gamma^{-1}\varepsilon^{-1}\tau_{\mathcal{S}_i}(q)|\mathcal{S}_i|\log n \ll \gamma\sigma_{\mathcal{S}_i\setminus(\mathcal{D}\cup\mathcal{T})}(q)$$

Combining these inequalities we get

$$\begin{aligned} \sigma_{\mathcal{S}_i\setminus\mathcal{D}}(q) &= \hat{s} + 8\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q) + s_{err} + f_{err} \\ |s_{err}| &\leq 4\varepsilon s \\ |f_{err}| &\ll \gamma\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q) + \gamma\sigma_{\mathcal{S}_i\setminus(\mathcal{D}\cup\mathcal{T})}(q) + \varepsilon\gamma\hat{s}. \end{aligned}$$

The value of $\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q)$ can be approximated recursively by updating $\mathcal{D}' = (\mathcal{D}\cup\mathcal{T})\cap\mathcal{S}_{i+1}$ and calling the query q with the set $\mathcal{D} = \mathcal{D}'$ on $\mathcal{S}_{i+1}$. Assume the recursive call at line (17) returns a value $\hat{\sigma}$ which is ε'-approximately equal to $\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q)$. Since $8\hat{\sigma}$ is approximately equal to $8\sigma_{\mathcal{S}_{i+1}\setminus(\mathcal{D}\cup\mathcal{T})}(q)$, the above inequalities imply that the value $\hat{s} + 8\hat{\sigma}$ computed at line (17) is δ-approximately equal to $\sigma_{\mathcal{S}_i\setminus\mathcal{D}}(q)$ where $\delta = \max\{4\varepsilon, \varepsilon' + O(\gamma)\}$; however, since there are only $\log n$ steps of recursion and $\gamma = \varepsilon/\log n$, the final result is a relative $O(\varepsilon)$-approximation of $\sigma_{\mathcal{S}_i}(q)$.

4.2.1 Filling the gaps

Now, we return to fill the two missing parts of the algorithm.

small($\mathcal{S}_i$, q). As in the main case, we first find a halfplane h containing $O(\tau_{\mathcal{S}_i}(q))$ points then use Observation 4.2.1 to approximate $\omega(p)$ for every point $p \in h$ with the dissection value of $\Omega(\varepsilon^{-1}\tau_{\mathcal{S}_i}(q))$. Summing up these values amounts to the approximate contribution of these points to the simplicial depth and as before the amount of double counting is negligible. Remove these points from h and let $a_1, \dots, a_k$ be the list of remaining points sorted radially around q. Since one side of every line $\overline{a_i q}$ contains $O(\varepsilon^{-1}\tau_{\mathcal{S}_i}(q))$ points, we can issue exact halfspace range counting queries in $O(\varepsilon^{-1}\tau_{\mathcal{S}_i}(q))$ time and compute the exact number of points contained in the wedges formed by the lines $\overline{a_i q}$ (see Figure 4.3); the simplicial depth of q can be exactly computed using this information.

dissectionReport($\mathcal{S}_i$, q). First we show how to find the halfplane h. We remind the reader of the subproblem which must be solved.

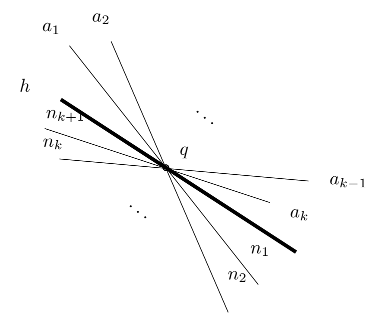


Figure 4.3: n_i denotes the number of points of $\mathcal{S}_i$ contained in each wedge.

Problem 4.2.2. *Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^2$, and $\varepsilon, m > 0$ be fixed parameters. Build a data structure of size $\tilde{O}(n)$ such that given a query point q, in $\tilde{O}(1)$ time one can find a halfspace h and a subset $\mathcal{A} \subset \mathcal{P} \cap h$. Furthermore, if $\tau_{\mathcal{P}}(q) \geq Cm$ for a sufficiently large constant C, then $\mathcal{A}$ must have the following properties: (i) h must have q on its boundary and contain $\Theta(\tau_{\mathcal{P}}(q))$ points, (ii) $|\mathcal{A}| \geq \min\{m, |\mathcal{H}|\}$, (iii) for every $p \in \mathcal{A}$ either the dissection value of p is $O(\tau_{\mathcal{S}_i}(q))$ or $\omega(p)$ is approximately greater than the $(3m)$-th largest weight in h.*

The requirement that $\mathcal{A}$ must be a subset of $h \cap \mathcal{P}$ is quite annoying since h depends on the query point and thus we do not have access to $h \cap \mathcal{P}$ during the preprocessing phase. First we try to remove this requirement. This is done by employing the following simple lemma which is Matoušek's shallow cutting in dual space.

Lemma 4.2.2. *Let $\mathcal{P}$ be a point set of n points in $\mathbb{R}^2$. For every k we can build a set $\mathcal{C}_k$ of $O(n/k)$ halfplanes with each halfplane containing $O(k)$ points of $\mathcal{P}$, such that for every point $q \in \mathbb{R}^2$ with $\tau_{\mathcal{P}}(q) \leq k$, there exists a halfplane $h \in \mathcal{C}$ which contains q. Furthermore, h can be found in $O(\log n)$ time.*

Proof. In the dual space $\mathcal{P}$ corresponds to a set $\overline{\mathcal{P}}$ of n lines and q corresponds to a line $\overline{q}$ which intersects either the k-level or the $(n-k)$-level of the arrangement $\mathcal{A}$ formed by $\overline{\mathcal{P}}$. Consider the first case since the similar case can be handled similarly. By Matoušek's shallow cutting lemma we can build a set $\mathcal{C}$ of $O(n/k)$ triangles which cover the k-level of $\mathcal{A}$. Since $\overline{q}$ intersects the k-level of $\mathcal{A}$, there exists a vertex v of $\mathcal{C}$ which lies directly above $\overline{q}$. In the primal space, v corresponds to a halfplane which contains q and $O(k)$ additional points of $\mathcal{P}$. Thus, the set of halfplanes claimed in the lemma can be taken as the set of halfplanes dual to the vertices of the triangles obtained by the shallow cutting lemma. □

For every $\mathcal{S}$, we build the above set of halfplanes for $k = 2^i, i = 0, \ldots, \lfloor \log \mathcal{S}_i \rfloor$ and obtain a set $\mathcal{C}$ of $O(n)$ halfplanes containing a total of $O(n \log n)$ points with the property that for any given point q we can find a halfplane $h_i \in \mathcal{C}$ which contains q and $O(\tau_{\mathcal{S}_i}(q))$ additional points. The halfplane h outputted by the subroutine **dissectionReport** is a halfplane through q and parallel to h_i.

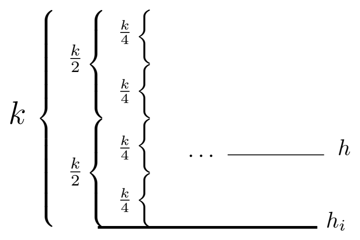


Figure 4.4: The points above h can be expressed as the union of $O(\log k)$ sets represented by brackets; the number before each bracket denotes the number of points contained in the corresponding set.

For a halfplane h_i containing k points, it is easily seen that we can build $O(k)$ sets $\mathcal{H}_i^{(1)}, \dots, \mathcal{H}_i^{(t_i)}$ containing $O(k \log k)$ points in total such that the points contained in any halfplane h parallel to h_i can be expressed as the union of the $O(\log k)$ such sets (see Figure 4.4). Thus, by causing only a polylogarithmic factor increase in the space and query time of the algorithm, we can assume $\mathcal{A}$ is required to be subset of a fixed set $\mathcal{H}$. This reduces our problem to the following.

Problem 4.2.3. *Let $\mathcal{P}, \mathcal{H}$ be sets containing n points in $\mathbb{R}^2$ and $\varepsilon, m > 0$ be fixed parameters. Build a data structure of size $\tilde{O}(n)$ such that given a query point q in $\tilde{O}(1)$ time one can find a halfspace h and a subset $\mathcal{A} \subset \mathcal{H}$. (i) h must have q on its boundary and contain $\Theta(\tau_{\mathcal{P}}(q))$ points, (ii) $|\mathcal{A}| \geq \min\{3m, |\mathcal{H}|\}$, (iii) for every $p \in \mathcal{A}$ either the dissection value of p is $O(\tau_{\mathcal{S}_i}(q))$ or $\omega(p)$ is approximately greater than the $(3m)$-th largest weight in h.*

We also notice that it suffices to solve the "decision" version of the above problem in which we are given a fixed parameter M and we are asked to report $3m$ points with dissection value approximately greater than M. It is clear by running $\log n$ instances of the this decision version for $M = n/2^i, i = 0, \dots, \lfloor \log n \rfloor$ we can compute the set $\mathcal{A}$.

To solve this decision version, we switch to the dual space. A point $p \in \mathcal{H}$ has dissection value at least M with respect to q and $\mathcal{P}$ if and only if the dual of the line $\ell := \overline{pq}$ (which is a point represented by $\overline{\ell}$) neither is contained in the $(\leq M)$-level nor in the $(\geq |\mathcal{P}| - M)$-level of $\overline{\mathcal{P}}$. An observation of Chan [36] implies that there exists a convex polygonal chain P_L of size $O(|\mathcal{P}|/M)$ which contains the M-level of $\overline{\mathcal{P}}$ and is contained in an $O(M)$-level of $\overline{\mathcal{P}}$. Consider a similar polygonal chain P_U for the $(\geq |\mathcal{P}| - M)$-level of $\overline{\mathcal{P}}$. Thus, if $\overline{\ell}$ is inside either P_L or P_U, then the dissection value of p is approximately less than M, otherwise the dissection value of p is approximately greater than M. Also, notice that $\overline{\ell}$ is the intersection point of the lines $\overline{q}$ and $\overline{p} \in \overline{\mathcal{H}}$. In other words, our problem reduces to reporting up to $3m$ lines from a given set of lines $\overline{\mathcal{H}}$ which intersect a query line $\overline{q}$ outside two fixed convex polygons P_L and P_U.

To solve this subproblem, we compute the intersection points of the query line $\overline{q}$ with the convex polygons P_U, P_L and the lines $x = \infty$ and $x = -\infty$ in $O(\log n)$ time

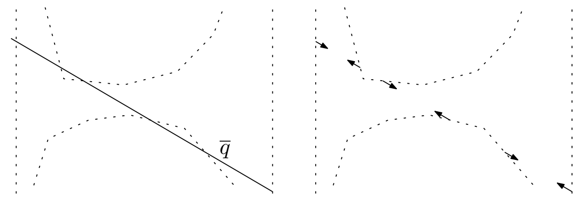


Figure 4.5: Decomposing the query line $\overline{q}$ into a constant number of rays.

and then decompose this line into a constant number of rays as shown in Figure 4.5. This further reduces our problem to a ray shooting problem (finding the first object hit by a ray) in an arrangement of lines. Unfortunately, we cannot use any of the standard ray shooting results since we are aiming for a polylogarithmic query time with near linear space which is impossible with the current ray shooting algorithms. Nonetheless, we can exploit a key property of our problem which is the observation that all the query rays are originating from a fixed convex object. For simplicity, we only describe how to solve the ray shooting queries for rays originating from the boundary of P_L.

The convexity of P_L implies its zone (the set of vertices of the arrangement adjacent to the cells crossed by P_L) has near linear complexity [24] and can be constructed in $\tilde{O}(|\mathcal{H}|)$ time [58]. Constructing the zone of P_L enables us to find the first line intersected by a ray originating from the boundary of P_L.

By taking a random k^{-1}-sample $\mathcal{H}_k$ of $\mathcal{H}$ we can answer a wider range of queries. In fact, we can use the techniques of Chan [37] and report the first $3m$ lines intersected by a query ray in $O(\log|\mathcal{H}| + 3m)$ time using $\tilde{O}(|\mathcal{H}|)$ space by building a series of random samples.

Finally, using all the above data structures we obtain the following theorem.

Theorem 4.2.3. *It is possible to preprocess a point set $\mathcal{P} \subset \mathbb{R}^2$ of n points in $\tilde{O}(n)$ expected time using $\tilde{O}(n)$ space such that given a query point q a relative ε-approximation for the simplicial depth of q can be found in $\tilde{O}(1)$ expected time for any arbitrary fixed constant $\varepsilon > 0$. The final result is correct with high probability.*

4.3 Approximate Simplicial Depth in Higher Dimensions

In this section we consider the problem of approximating the simplicial depth of a given point q in a set $\mathcal{P}$ of n points in $\mathbb{R}^d$. We will use the following observation made by Gil et al. [70].

Observation 4.3.1. *Let q be a point inside a simplex $a_1 \dots a_{d+1}$ and let a'_i be a point on the ray $\overrightarrow{qa_i}$. The simplex defined by $a_1 \dots a_{i-1}a'_ia_{i+1} \dots a_{d+1}$ contains q.*

The above lemma essentially means we can move every point of the input on rays originating from the point q without changing its simplicial depth.

Theorem 4.3.2. *Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^d$, $d \geq 3$, and p be an arbitrary point. A relative ε-approximation of the simplicial depth of p can be obtained in $\tilde{O}(n^{d-2})$ expected time with high probability.*

Proof. First we find a halfspace h through p containing a set $\mathcal{A} \subset \mathcal{P}$ of $O(\tau(p))$ points. This can be done in $\tilde{O}(n^{\lfloor n/2 \rfloor})$ time using various methods for instance, by using the general reduction used by Aronov and Har-Peled (Theorem 2.3.3). Ignoring polylogarithmic factors, this is essentially the time needed to decide whether a point is outside the convex hull $\mathcal{P}$ and if so, to find the halfspace separating it from $\mathcal{P}$.

Let ℓ be the boundary hyperplane of h and without loss of generality assume $\mathcal{A}$ lies above ℓ. Consider a hyperplane ℓ' parallel to ℓ with the remaining set of points $\mathcal{B} := \mathcal{P} \setminus \mathcal{A}$ sandwiched between ℓ and ℓ'. By using central projection from p, map a point $b \in \mathcal{B}$ to a point b' on the hyperplane ℓ' (Figure 4.6(a)). Let $\mathcal{B}'$ be the set of the projected points. According to Observation 4.3.1, replacing $\mathcal{B}$ with $\mathcal{B}'$ does not change the simplicial depth and thus in the remainder of the proof we will focus on the set $\mathcal{P}' := \mathcal{A} \cup \mathcal{B}'$.

Any simplex containing p must have at least one point from $\mathcal{A}$. Thus, we can express the simplicial depth of p as the sum of the number of simplices containing exactly one point from $\mathcal{A}$ (denoted by $\sigma^{(1)}(p)$) and the number of simplices containing two or more points from $\mathcal{A}$ (denoted by $\sigma^{(2+)}(p)$).

For a point $a \in \mathcal{A}$ denote the intersection of the ray $\overrightarrow{ap}$ with ℓ' by a'. The number of simplices containing p with a as their only vertex from $\mathcal{A}$ is exactly equal to the number of $d-1$ dimensional simplices formed by the points of $\mathcal{B}'$ which contain the point a' which is a $(d-1)$-dimensional problem. Using Theorem 4.2.3 as our base case, we can build a $\tilde{O}(n^{d-2})$ time algorithm capable of finding a relative ε-approximation of $\sigma^{(1)}(p)$.

For the remaining term of $\sigma^{(2+)}(p)$ notice that the total number of simplices with at least two vertices from $\mathcal{A}$ is $O(\tau^2(p)n^{d-1})$. By Lemma 2.3.2, to obtain a relative approximation we can pick $\tilde{O}\left(\tau^2(p)n^{d-1}/\sigma^{(2+)}(p)\right)$ random simplices. We observe that since $\sigma^{(2+)}(p)+\sigma^{(1)} = \sigma(p) = \Omega(\tau^d(p)n)$, only $\tilde{O}\left(\tau^2(p)n^{d-1}/\tau^d(p)n\right) = \tilde{O}(n^{d-2})$ random samples is sufficient to give an additive error of $O(\varepsilon\sigma(p))$. Combining these two cases will result in an $\tilde{O}(n^{d-2})$ time algorithm. $\square$

4.3.1 Approximate simplicial depth in 3D

In this section we describe a slight modification of the algorithm of Theorem 4.3.2 which improves its query time at the expense of raising its space complexity. This can be seen as an application of our techniques in the previous sections since we introduce no new ideas and simply use a different combination of the previous techniques.

Unfortunately, since we are aiming for an $o(n)$ query time, we cannot use Observation 4.3.1 and in fact this can be seen as the biggest challenge in getting an $o(n)$ query time in 3D. But since we are aiming for an $\tilde{O}(n^{2/3})$ query time, we have access to a variety of simplicial range searching data structures including the following lemma which can be obtained using standard techniques in simplicial range searching such as multi-level partition trees.

Lemma 4.3.3. *Given n simplices in 3D, one can store them in a data structure of size $\tilde{O}(n)$ using $O(n^{1+\varepsilon})$ preprocessing time such that the number of simplices containing a query point q can be counted in $O(n^{2/3})$ time.*

The following lemma extends Lemma 4.2.2 to 3D and can be proved in an identical way.

Lemma 4.3.4. *Let $\mathcal{P}$ be a point set of n points in $\mathbb{R}^3$. For every k, we can build a set $\mathcal{C}_k$ of $O(n/k)$ halfspaces with each halfspace containing $O(k)$ points of $\mathcal{P}$ such that for every point $q \in \mathbb{R}^3$ with $\tau_{\mathcal{P}}(q) \leq k$ there exists a halfspace $h \in \mathcal{C}$ which contains q. Furthermore, h can be found in $O(\log n)$ time.*

Theorem 4.3.5. *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^3$, with $\tilde{O}(n^2)$ preprocessing time one can build a data structure of size $\tilde{O}(n^2)$ such that the approximate simplicial depth of any query point q can be computed in $\tilde{O}(n^{2/3})$ time. The correctness of the final result holds with high probability.*

Proof. Corollary 4.1.4 implies that we can reduce the problem to a random 1/2-sample of $\mathcal{P}$ provided $M = \tilde{O}(\sigma_{\mathcal{P}}(q))$, where $M := \max p \in \mathcal{P}\omega(p)$; however, according to Lemma 4.1.2, $\sigma_{\mathcal{P}}(q) = \Omega(n\tau^3_{\mathcal{P}}(q))$ implying $M = \tilde{O}(\sigma_{\mathcal{P}}(q))$ if $\tau_P(q) \geq n^{2/3}\log^c n$. Thus, we can essentially reduce the problem to the case when $\tau_{\mathcal{P}}(q) = \tilde{O}(n^{2/3})$.

Similar to the 2D case, using Lemma 4.3.4 we build a fixed set $\mathcal{C}$ of halfspaces such that for every query point q we can find a halfspace $h \in \mathcal{C}$ containing q and $O(\tau_{\mathcal{P}}(q))$ points of $\mathcal{P}$. Next, we follow the proof steps and terminology of Theorem 4.3.2 and show it is possible to approximate $\sigma^{(1)}(q)$ in $\tilde{O}(n^{2/3})$ time using $\tilde{O}(n^2)$ space.

Unfortunately, we cannot use Observation 4.3.1 as the required central projection depends on the query point; however, we are able to apply a modification of the observation and use a central projection from every point of $p \in h$ rather than q. Since q is contained in h, a simplex S with p as one of its vertices contains q if and only the corresponding simplex with the projected points contains q (see Figure 4.6(b)). We implement the 2D data structure on the projected points for every $p \in h$. This would consume $\tilde{O}(n^2)$ space but will enable us to find a relative ε-approximation of $\sigma^{(1)}(q)$ in $\tilde{O}(n^{2/3})$ time by issuing one 2D ε-approximate simplicial depth query for every point of h.

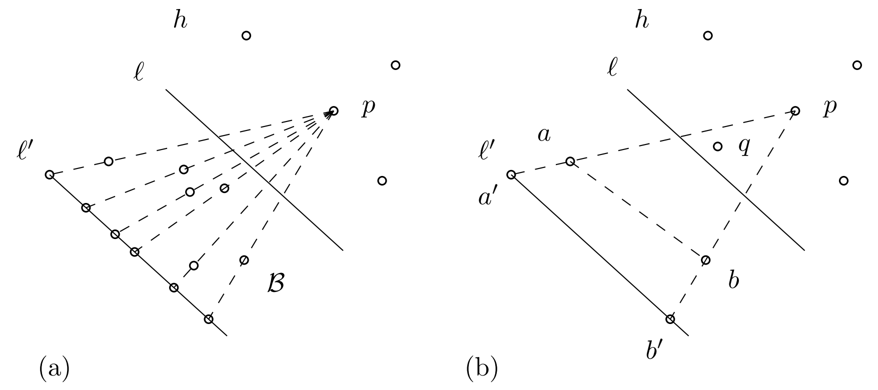


Figure 4.6: (a) Projecting the set of points outside h onto a parallel plane from point p. (b) Triangles pab and $pa'b'$ both contain q.

Finally, as with the case in Theorem 4.3.2, we need to sample $\tilde{O}(n/\tau(q))$ simplices containing two or more points from h and count the fraction of these containing q. Using Lemma 4.3.3 this phase can be done in $\tilde{O}(n^{2/3})$ time, using $\tilde{O}(n)$ space and $O(n^{1+\gamma})$ preprocessing time for an arbitrary constant $\gamma > 0$. $\square$

Chapter 5

Dominance Queries

As we have mentioned in the first chapter, dominance queries are a very important special case of orthogonal range searching queries. In this chapter, we focus our attention on dominance reporting problem in 3D and show that this problem can be attacked using the techniques that are used in the previous sections of this thesis, namely, approximate levels (Chapter 3).

5.1 Introduction

Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^d$. In the dominance reporting problem, we are given a query point $q = (q_1, \ldots, q_d)$ and we are asked to find all the points $p = (x_1, \ldots, x_d) \in \mathcal{P}$ such that $x_i < q_i$, $1 \leq i \leq d$. A comparison of our results with the previous work is shown in Table 5.1. When used as the base case for the orthogonal range reporting problem, our results also imply new improvements for this problem as well. These implications, together with the corresponding previous best results, are shown in Table 5.2.

Unlike previous methods for the dominance problem, we borrow ideas and techniques from non-orthogonal range searching, namely, halfspace range searching. We observe that both halfspace range reporting and dominance reporting in 3D can be attacked within the same framework and using the same array of techniques. This observation has apparently eluded the previous researchers. For instance, we use a version of the shallow cutting lemma (Lemma 1.1.5). Previously, in the orthogonal setting, Vengroff and Vitter [121] and Nekrich [102] have used a similar concept but under a different name ("B-approximate boundaries") and with non-optimal, complicated constructions.

One might wonder whether other techniques for halfspace range searching such as the shallow partition theorem can also be applied in this context. We show the existence of a novel partition-type theorem for the dominance reporting problem (it is not a partition per se but resembles the partition theorem in "spirit"). This theorem leads to an optimal external memory data structure. In contrast, an

Model	Space	Query complexity	Source
PM	$O(n)$	$O(\log n \log\log n + k)$	[91]
PM	$O(n \log n)$	$O(\log n + k)^*$	[49]
PM	$O(n)$	$O(\log n + k)^*$	here
RAM	$O(n)$	$O((\log\log U)^2 \log\log\log U + k \log\log U)$	[91]
RAM	$O(n)$	$O(\frac{\log n}{\log\log n} + k)$	[78]
RAM	$O(n \log n)$	$O((\log\log n)^2 + \log\log U + k)$	[102]
RAM	$O(n)$	$O((\log\log n)^2 + \log\log U + k)$	here
EM	$O(n \log n)$	$O(\log_B n + k/B)^*$	[121]
EM	$O(n)$	$O(\log_B n + k/B)^*$	here
IEM	$O(n \log n)$	$O(\log\log_B U + (\log\log n)^2 + k/B)$	[103]
IEM	$O(n \log_B n)$	$O(\log\log_B U + (\log\log n)^2 + k/B)$	here

Table 5.1: Results on 3D dominance reporting problem. Here, n, k, B are input, output and block size respectively. The results on a RAM assume the input is from a $U \times U \times U$ integer grid. PM, EM and IEM stand for pointer machine, external memory and external memory with integer inputs respectively. Optimal query complexities are marked with a star. The last two rows assume an external memory model of computation in the RAM model where the input is on a $U \times U \times U$ integer grid.

optimal external memory result for the halfspace range reporting problem is not known yet [6].

As a consequence of our results, we obtain two new orthogonal range reporting algorithms consuming $O(n \log^3 n)$ space; one with $O((\log\log n)^2 + \log\log U + k)$ query time in the RAM model and another with $O(\log_B n + k/B)$ I/Os in the external memory model. Previously, the best results consumed $O(n \log^4 n)$ space in both cases. We only use randomization in the preprocessing part and our query bounds are all worst case.

5.2 Dominance Reporting in 3D

5.2.1 Preliminaries

To remind the reader, we say a point $A \in \mathbb{R}^d$ dominates another point $B \in \mathbb{R}^d$ if and only if all the coordinates of A are greater than those of B. This is our source of inspiration for defining a special form of geometric range that we call a *downward corner*. A downward corner is uniquely determined by a point $A \in \mathbb{R}^d$ (called the *apex*) and contains all the points of $\mathbb{R}^d$ which are dominated by A. Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^d$. To make the notation consistent, we reserve the symbol $\mathtt{r}$ for downward corners and with an abuse of notation, sometimes we use $\mathtt{r}$ to refer both to the geometric object and the subset of $\mathcal{P}$ inside the geometric

Model	Dimension	Space	Query complexity	Source
RAM	$d = 3$	$O(n \log^4 n)$	$O((\log\log n)^2 + \log\log U + k)$	[102]
RAM	$d = 3$	$O(n \log^3 n)$	$O((\log\log n)^2 + \log\log U + k)$	here
RAM	$d > 3$	$O(n \log^{d+1+\varepsilon} n)$	$O(\log^{d-3} n/(\log\log n)^{d-5} + k)$	[102]
RAM	$d > 3$	$O(n \log^{d-2+\varepsilon} n)$	$O(\log^{d-2} n/(\log\log n)^{d-2} + k)$	[18]
RAM	$d > 3$	$O(n \log^{d+\varepsilon} n)$	$O(\log^{d-3} n/(\log\log n)^{d-5} + k)$	here
EM	$d = 3$	$O(n \log^4 n)$	$O(\log_B n + k/B)$	[121]
EM	$d = 3$	$O(n \log^3 n)$	$O(\log_B n + k/B)$	here

Table 5.2: The fastest known orthogonal range reporting algorithms for $d \geq 3$. Here, n, k, B are input, output and block size respectively. EM stand for external memory.

object. In this chapter, we define an *approximate k-level* $\mathcal{L}_k$ as a set of downward corners with the following two properties: (i) any downward corner $\mathtt{r} \in \mathcal{L}_k$ must contain at most $c_1 k$ points of $\mathcal{P}$ and (ii) any downward corner $\mathtt{r}'$ which contains at most k points of $\mathcal{P}$ must be contained in a downward corner $\mathtt{r} \in \mathcal{L}_k$. Here, c_1 can be chosen to be an arbitrary constant (by our algorithms). The size of an approximate level is the number of its downward corners. Finally, for the sake of simplicity of description, we assume the input point set and the query points are in general position; a restriction that can be overcome using standard tricks.

5.2.2 Optimal approximate levels

Consider a set $\mathcal{S}$ of n downward corners. We define the level of a point $p \in \mathbb{R}^3$ to be the number of downward corners of $\mathcal{S}$ which contain p. As with the case of halfspaces, we define the $(\leq k)$-level to be the set of all the vertices of the arrangement $\mathcal{A}$ formed by $\mathcal{S}$ with level at most k. Thus, the (≤ 0)-level of $\mathcal{A}$ contains all the vertices of the arrangement that are not inside any downward corner of $\mathcal{S}$.

One crucial requirement of any optimal result on approximate levels is a linear bound on the size of the (≤ 0)-level of $\mathcal{A}$.

Lemma 5.2.1. *For a set $\mathcal{S}$ of n downward corners the size of the (≤ 0)-level is $O(n)$.*

Proof. Sweep a plane h parallel to the xy plane from $z = +\infty$ to $z = -\infty$. We count the vertices of the (≤ 0)-level of the arrangement as they appear on this sweep plane.

The apex A of an element $\mathtt{r} \in \mathcal{S}$ will appear on h when the z-coordinate of h becomes equal to the z-coordinate of A and it disappears as soon as another point q' on h dominates A (in 2D sense). The crucial observation is that if a point disappears from h it no longer can contribute any new vertices to the (≤ 0)-level

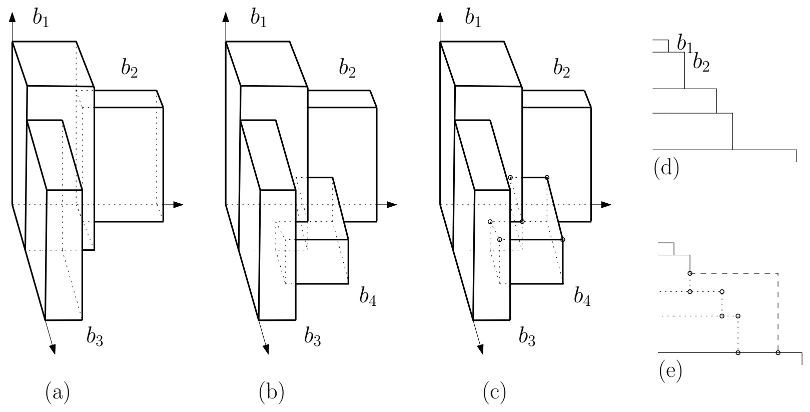


Figure 5.1: (a) Boxes b_1, b_2 and b_3 already swept (b) The sweep plane discovers b_4 and from this point b_1 can be ignored. (c) Marks denote the new vertices on the 0-level. (d,e) View on plane h

(Figure 5.1(a,b,c)). So, at any moment we have an active set of downward corners on this plane with none dominating another; these points form a chain on this plane (Figure 5.1(d)). Assume a new point c_{t+1} appears on this plane. If c_{t+1} creates j new vertices then it will have to dominate and remove $\Omega(j-4)$ active points from h (Figure 5.1(c,e)). A simple charging argument implies that number of vertices of the (≤ 0)-level is $O(n)$. □

The shallow cutting lemma of Agarwal et al. [9] operates on a general class of surfaces and thus accepts a parameter $\phi(r)$ which is the worst case size of the (≤ 0)-level of any collection of r surfaces. The above lemma implies that in our problem we have $\phi(r) = O(r)$. Combining this with the theorem of Agarwal et al. [9] we obtain the following lemma.

Lemma 5.2.2. *Given a set $\mathcal{S}$ of n downward corners in 3D and a parameter k, one can build a set $\mathcal{B}$ of $O(n/k)$ boxes which cover the ($\leq k$)-level of the arrangement formed by $\mathcal{S}$ where each box is intersected by $O(k)$ downward corners.*

Proof. With slight perturbations we can turn a downward corner into a continuous surface which fits the framework of Agarwal et al. [9] and use their shallow cutting lemma with $r = n/k$. The fact that the set $\mathcal{B}$ can be taken as a set of boxes follows from the vertical decomposition used by Agarwal et al. [9]. □

The above shallow cutting result can be used to construct approximate levels.

Lemma 5.2.3. *There exists an approximate k-levels of size $O(n/k)$, $1 \leq k \leq n$, for the dominance reporting problem.*

Proof. Let $\mathcal{P}$ be an input set of size n. For a point $p \in \mathbb{R}^3$, define an *upward corner with apex p* to be the subset of $\mathbb{R}^3$ which dominates p. Let $\mathcal{S}$ be the set of n upward corners determined by points of $\mathcal{P}$ as apexes and let $\mathcal{A}$ be the arrangement formed by $\mathcal{S}$. A point reflection with origin can transform $\mathcal{A}$ into an arrangement of n downward corners, A', and thus we can use Lemma 5.2.2 and build a collection $\mathcal{B}'$ of $O(n/k)$ boxes which cover the $(\leq k)$-level of $\mathcal{A}'$. Perform the point reflection on elements of $\mathcal{B}'$ and let $\mathcal{B}$ be the resulting set of boxes. For every box in $b \in \mathcal{B}$, place the vertex with the maximum coordinates, denoted with $m(b)$, in a set $\mathcal{C}$. We claim the set of downward corners defined by apexes in $\mathcal{C}$ is an approximate level.

Consider a downward corner $\mathtt{r}$ with apex A which contains less than k points of $\mathcal{P}$. This means that there are less than k upward corners of $\mathcal{S}$ which contain A. The reflection A' of A by the origin lies in the $(\leq k)$-level of $\mathcal{A}'$ and thus there is a box $b \subset \mathcal{B}$ which contains A. The downward corner defined by $m(b) \in \mathcal{C}$ contains $\mathtt{r}$.

On the other hand, Lemma 5.2.2 implies that every box $b' \in \mathcal{B}'$ lies in the $(\leq O(k))$-level of $\mathcal{A}'$. Thus, every vertex of $b \in \mathcal{B}$ can dominate at most $O(k)$ vertices of $\mathcal{P}$. □

5.3 Solving the Dominance Reporting Problem

To solve the dominance reporting problem in the RAM and the pointer machine models, we simply need a linear size data structure with polylogarithmic query time to combine it with our lemma on approximate levels. For instance, we can either use the data structure of Makris and Tsakalidis [91] or Edelsbrunner and Chazelle [49]. For the moment assume that we have access to a linear size data structure with $O(\log^2 n + k)$ query time.

Theorem 5.3.1. *Given a set of n points $\mathcal{P}$ in $\mathbb{R}^3$, dominance reporting queries can be answered in $O(\log n + k)$ worst case time in a pointer machine and using linear space.*

Proof. Let $\mathcal{A}$ be a data structure consuming linear space which can answer dominance reporting queries in $O(\log^2 n + k)$ time. Build an approximate $(\log^2 n)$-level $\mathcal{C}$. For every downward corner $\mathtt{r} \in \mathcal{C}$ implement the data structure $\mathcal{A}$ on the points contained in $\mathtt{r}$ and at the end build on extra copy for the whole point set $\mathcal{P}$.

If $k = \Omega(\log^2 n)$ then we can use the data structure on $\mathcal{P}$ to answer the query in $O(\log^2 n + k) = O(k)$ time. Otherwise, one downward corner $\mathtt{r}$ in $\mathcal{C}$ will contain q. As with the case of halfspaces, finding q is equivalent to a point location query in a 2D orthogonal arrangement and thus can be solved in $O(\log n)$ time. Since $\mathtt{r}$ will contain at most $\log^2 n$ points, the query can be answered in $O((\log\log^2 n)^2 + k) = O((\log\log n)^2 + k)$ time using the data structure implemented on $\mathtt{r}$. Combining all these gives us a query time of $O(\log n + k)$. □

The exact same idea can be applied in the word RAM model, by employing the point location data structure of [61] which offers the query time of $O((\log\log U)^2)$ in a $U \times U \times U$ integer grid.

Theorem 5.3.2. *Given n points in $U \times U \times U$ integer grid, dominance reporting queries can be answered in $O((\log\log U)^2 + k)$ worst case time in the RAM model using linear space.*

By reduction to the rank space as described in Section 1.2.3, we get:

Corollary 5.3.3. *For n points in $\mathbb{R}^3$, dominance reporting queries can be answered in $O((\log\log n)^2 + \log\log U + k)$ time using only linear space.*

Also, by using the standard techniques described in Section 1.2.3, orthogonal range reporting queries reduce to dominance reporting queries, resulting in the following corollary.

Corollary 5.3.4. *There exists a data structure capable of answering 3D orthogonal range reporting queries on a $U \times U \times U$ grid in $O((\log\log n)^2 + \log\log U + k)$ time, using $O(n \log^3 n)$ space.*

The above can be extended to higher dimensions.

Corollary 5.3.5. *In a RAM, Orthogonal range reporting can be solved in $\mathbb{R}^d$ using $O(n \log^{d+\varepsilon} n)$ space and with $O(\log^{d-3} n/(\log\log n)^{d-5} + k)$ query time.*

We also note that any improvements to the data structure for point location in a planar rectangular subdivision [61] will automatically improve our query times as well.

Unfortunately, we cannot do the same trick to obtain an optimal algorithm in the external memory model, since in this model, to our knowledge, there is no linear space algorithm with reasonable query time to combine with our approximate levels. Thus, to get an optimal algorithm in the external memory model, we need to develop additional tools and ideas. This is done in the next section.

5.4 The External Memory Model

We use B to denote the block size in the external memory model. Using our result on approximate levels, we can re-obtain the same $O(n \log n)$ bound on the space as Vengroff and Vitter [122]; although our data structure will be much simpler. Nonetheless, to reduce the space complexity, we need the following lemma.

Lemma 5.4.1. *Let $\mathcal{P} \subset \mathbb{R}^3$ be a set of n points such that the level of each point is at most m. We can find $t = O(n/m)$ sets, $\mathcal{V}_1, \dots, \mathcal{V}_t \subset \mathcal{P}$, $|\mathcal{V}_i| = O(m)$ such that for any downward corner $\mathtt{r}$ containing k points there exist $s = O(k/m)$ sets $\mathcal{V}_{t_1}, \dots, \mathcal{V}_{t_s}$ with $|\mathcal{P}_{\mathtt{r}} \setminus (\bigcup_{i=1}^{s} \mathcal{V}_{t_i})| = O(m)$ in which $\mathcal{P}_{\mathtt{r}} = \mathcal{P} \cap \mathtt{r}$.*

Proof. Let $\mathcal{C} = \{\mathtt{r}_1, \ldots, \mathtt{r}_t\}$ be an approximate Cm-level for a constant C to be determined later. With a slight abuse of the notation, we will use $\mathtt{r}_i$ to refer to both the downward corner $\mathtt{r}_i$ and the subset of $\mathcal{P}$ contained in $\mathtt{r}_i$. We claim $\mathtt{r}_1, \ldots, \mathtt{r}_t$ are the sets claimed in the lemma. By Lemma 5.2.3 we know $t = O(n/m)$.

Consider a downward corner $\mathtt{r}$ containing k points. According to Lemma 5.2.3, we can find an approximate m-level, $\mathcal{C}' = \{\mathtt{r}'_1, \ldots, \mathtt{r}'_t\}$, of size $t' = O(k/m)$ for the points inside $\mathtt{r}$. By definition, $\mathcal{C}'$ covers the $(\leq m)$-level of $\mathcal{P}_\mathtt{r}$ and so every point of $\mathcal{P}_\mathtt{r}$ is contained in at least one downward corner of $\mathcal{C}'$. Thus, $\mathcal{P}_\mathtt{r} = \bigcup_{i=1}^{t'} \mathtt{r}'_i$. If we could show that for every $\mathtt{r}'_i \in \mathcal{C}'$ there is another downward corner $\mathtt{r}_j \in \mathcal{C}$ which contains $\mathtt{r}'_i$, then our lemma could be easily solved. Unfortunately this is not true and in fact, $\mathtt{r}'_i$ may contain $\Omega(n)$ points of $\mathcal{P}$ (although it can only contain $O(m)$ points of $\mathcal{P}_\mathtt{r}$). Because of this we aim for a slightly weaker claim.

Let (x, y, z) be the coordinates of the apex A of $\mathtt{r}$ and (x', y', z') be the coordinates of the apex A'_i of $\mathtt{r}'_i$. By Lemma 5.2.3 we know each $\mathtt{r}'_i$ contains $O(m)$ points; assume the constant in the O notation is c. Pick $C > c$. We have four important cases:

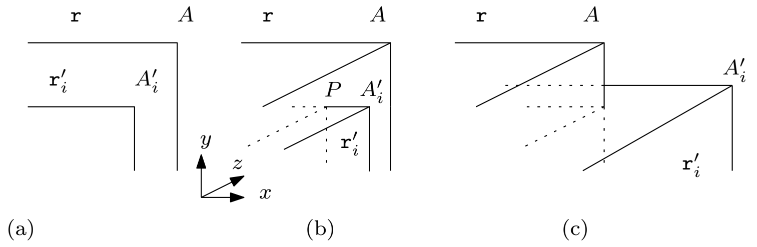


Figure 5.2: (a) $\mathtt{r}'_i$ is contained inside $\mathtt{r}$. (b) Only the x-coordinate of A'_i is greater than that of A. (c) Two coordinates of A'_i are greater than those of A.

1. A dominates A'_i (Figure 5.2(a)): In this case $\mathtt{r}'_i$ contains at most Cm points of $\mathcal{P}$ and is contained in at least one downward corner $\mathtt{r}_i \in \mathcal{C}$. Thus, in this case $\mathtt{r}'_i \subset \mathtt{r}_i$.

2. Only one coordinate of A'_i is not dominated by that of A (Figure 5.2(b)): Without loss of generality assume it is the x-coordinate (i.e., $x < x', y > y'$ and $z > z'$). In this case, the point $Q = (x, y', z')$ is contained in $\mathtt{r}$ and thus dominates at most Cm points of $\mathcal{P}$ which means Q is contained in at least one downward corner $\mathtt{r}_i \in \mathcal{C}$. Thus, in this case $\mathtt{r}'_i \cap \mathtt{r} \subset \mathtt{r}_i$.

3. Only one coordinate of A dominates that of A'_i (Figure 5.2(c)): This case can only happen for three elements of $\mathcal{C}'$, once for each coordinate; for instance, if $x' > x$ and $z' > z$, then $\mathtt{r}'_i$ contains at most Cm points with minimum y-coordinates in $\mathcal{P}_\mathtt{r}$.

4. A'_i dominates A. This case can only happen if $\mathcal{P}_\mathtt{r}$ contains less than Cm points.

For every downward corner $\mathtt{r}'_i \in \mathcal{C}'$, the first two cases provide us with another downward corner $\mathtt{r}_i \in \mathcal{C}$ such that $\mathtt{r}'_i \cap \mathtt{r} \subset \mathtt{r}_i$. The other two cases only cover $O(m)$ points. Thus, we can find at most t' downward corners $\mathtt{r}_1, \dots \mathtt{r}_{t''}, \in \mathcal{C}$ such that $|\mathcal{P}_{\mathtt{r}} \setminus (\cup_{i=1}^{t''} \mathtt{r}_{t_i})| = O(m)$ with $t'' = O(k/m)$. □

Remarks. The closest theorem in the context of the halfspace range searching to the above lemma is the shallow partition theorem by Matoušek. However, the above lemma does not partition the point set and does not cover all the points inside the downward corner $\mathtt{r}$. Also, it can be viewed as "output-sensitive" in the sense that the number of sets contained or intersected by $\mathtt{r}$ depends on the number of points, k, contained in $\mathtt{r}$. It has been observed that such dependence on k is a desirable property [23]. Thus, an interesting question is whether it is possible to obtain a similar result for halfspace range searching.

Lemma 5.4.2. *There is a data structure for a set $\mathcal{P}$ of n points in $\mathbb{R}^3$ which can answer dominance reporting queries with $O(\log_B n + k/B)$ I/Os using $O(n)$ space.*

Proof. Partition $\mathcal{P}$ into subsets $\mathcal{P}_1, \dots, \mathcal{P}_r$ in the following way: define $\mathcal{P}_1$ to be the set of points $p \in \mathcal{P}$ with level at most $B \log_B n$, remove $\mathcal{P}_1$ and repeat and continue this operation until $\mathcal{P}$ is partitioned. This construction ensures that every point $p \in \mathcal{P}_i$ has level at most $O(B \log_B n)$ in $\mathcal{P}_i$.

Assume for every $\mathcal{P}_i$ we have a data structure which uses $O(|\mathcal{P}_i|)$ space and can answer queries with $O(\log_B |\mathcal{P}_i| + k/B)$ I/Os. Given a query $\mathtt{r}$, we start from $\mathcal{P}_1$ and using the data structure implemented on $\mathcal{P}_1$ we return all the points of $\mathcal{P}_1$ inside $\mathtt{r}$ and then move on to the next set $\mathcal{P}_2$ and continue this until we reach a point set $\mathcal{P}_i$ which does not contain any point in $\mathtt{r}$; at this point we terminate the search. The crucial observation is that if $\mathtt{r}$ contains at least one point from $\mathcal{P}_{i+1}$ then it must contain at least $B \log_B n$ points from $\mathcal{P}_{i-1}$. This implies $k = \Omega(iB \log_B n)$ and thus the total query complexity will add up to $O((i+1)\log_B n + k/B) = O(\log_B n + k/B)$. In short, this means that it suffices to solve the problem for point sets $\mathcal{P}$ in which the level of every point is at most $B \log_B n$. This will be our assumption in the rest of the proof.

Let $m = B \log_B n$. Using Lemma 5.4.1, compute $t = O(n/m)$ sets, $\mathcal{V}_1, \dots, \mathcal{V}_t$, $|\mathcal{V}_i| = O(m)$ and store the points of each set $\mathcal{V}_i$ sequentially. Consider a downward corner $\mathtt{r}$ containing k points. According to Lemma 5.4.1 there are $s = O(k/m)$ sets $\mathcal{V}_{t_1}, \dots, \mathcal{V}_{t_s}$ such that $|\mathcal{P}_{\mathtt{r}} \setminus (\cup_{i=1}^{s} \mathcal{V}_{t_i})| = O(m)$. We can represent the points inside $\mathtt{r}$ using $O(k/m)$ pointers to sets $\mathcal{V}_{t_i}$ and an additional list of $O(m)$ points. This is our storage scheme for the list of points inside a downward corner $\mathtt{r}$.

To make the data structure, we build a hierarchy of approximate k_i-levels for $k_i = 2^i m, 0 \le i \le O(\log(n/B))$ and we store the list of points in every downward corner of the approximate levels using our storage scheme. Every downward corner in an approximate $2^i m$-level has $O(2^i m)$ points and thus will be stored using $O(2^i)$ pointers and a list of $O(m)$ points. Since this approximate level contains $O(n/2^i m)$

ranges, the total space consumption for this level will be $O(n/m + m)$. Summing this up over all the approximate levels and including the space needed to store the sets $\mathcal{V}_1, \ldots, \mathcal{V}_t$ yields the space complexity of

$$O\left(n + \frac{n \log(n/B)}{m} + m \log(n/B)\right).$$

A simple calculation reveals this is always $O(n)$ for all values of B.

To answer the query, we find the smallest i such that a downward corner $\mathtt{r}_j$ of an approximate $2^i m$-level contains $\mathtt{r}$. This can be done with $O(i+1)$ steps of point location, once for every approximate level up to the i-th one. This will also ensure that $\mathtt{r}_j$ contains $\Theta(m2^i) = \Theta(k)$ points. The output can be determined by a linear scan of all the points in $\mathtt{r}_j$; however, we have not stored the list of points of $\mathtt{r}_j$ directly and thus we must perform $O(k/m)$ I/Os to just access the pointers, $O(k/B)$ I/Os to access the list of points referenced by these pointers and finally an additional $O(m/B)$ I/Os to access the list of points stored at $\mathtt{r}_j$. This amounts to $O\left((i+1)\log_B n + k + m/B\right) = O(\log_B n + k/B)$ I/Os. $\square$

Combined with the standard reductions (e.g., see [102]), we can obtain the following corollary.

Corollary 5.4.3. *There is a data structure for a set of n points in $\mathbb{R}^3$ that can answer orthogonal range reporting queries using $O(n \log^3 n)$ space and $O(\log_B n + k/B)$ I/Os.*

Using an external memory point location data structure of Nekrich [103], we can also have the following result.

Corollary 5.4.4. *There is a data structure for a set of n points in $\mathbb{R}^3$ which that can answer dominance reporting queries using $O(n \log_B n)$ space and $O(\log \log_B U + (\log \log n)^2 + k/B)$ I/Os.*

The super-linear space complexity of the above corollary stems from the super-linear requirement of the point location data structure.

Chapter 6

Rectilinear Polygon Counting

In this section, we turn to a problem in the area of intersection searching: Let $\mathcal{P}$ be a set of n disjoint rectilinear polygons and let q be a query rectilinear polygon of constant size. Our goal is to count the number of polygons of set $\mathcal{P}$ which intersect q (whether containment counts as intersection can be specified in the input).

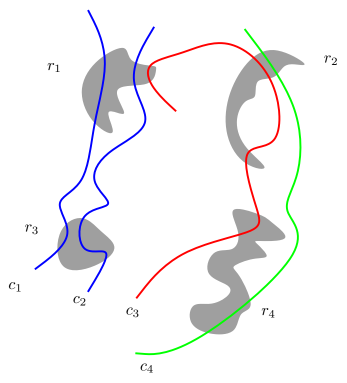
Rectilinear objects appear in many applications, specially areas such as VLSI design. In computational geometry, many researchers have studied problems related to intersection reporting and intersection counting of orthogonal objects. In this area, the simplest problem is to process an input set of n vertical line segments so that one can report line segments intersecting a horizontal query line segment. This problem was originally introduced as *the rectilinear line segment intersection problem* and was solved in $O(\log n + k)$ time using $O(n \log n)$ space [119].

The more complicated formulations of the problem allow for input segments to be marked with a color. This is known as the *colored intersection problem* in which the goal is to report or count the set of colors of the input objects intersected by the query object. The colored version of the rectilinear line segment intersection problem can be solved in $O(\log n + k)$ time using $O(n \log n)$ space [14]; the counting variant can be solved in $O(\log^2 n)$ time using $O(n \log^2 n)$ space [72]. If the input is a collection of rectilinear polygons of total complexity n, then we can color all the segments of a polygons with the same color and thus report or count the number of polygons intersecting a query segment within the same time bound.

These techniques do not require disjointness of the given rectilinear polygons; however, only the reporting variants are able to deal with queries more complicated than a line segment but in that case the query performance will depend on k which in worst case can can be $\Omega(n)$. In fact, we are not aware of any solution for the counting problem that can deal with *both* complex input (i.e., polygons) and queries that are composed of more than one line segment.

In the next section we develop a combinatorial technique that allows us attack the disjoint rectilinear polygon counting problem and finally in Section 6.2 obtain a non-trivial solution for this problem.

6.1 A Simple Combinatorial Lemma



(a)

$$\begin{aligned}&\mathcal{X} = \{r_1, r_2, r_3, r_4\}\\ &\mathcal{S} = \{\\ &\qquad \{r_1, r_2, r_4\},\\ &\qquad \{r_1, r_3\},\\ &\qquad \{r_2, r_4\}\\ &\}\end{aligned}$$

(b)

Figure 6.1: An example of four regions $r_1, \ldots, r_4$ and four curves $c_1, \ldots, c_4$ with curves c_1 and c_2 being the only two equivalent curves.

Consider a set $\mathcal{R}$ of n disjoint regions with simply connected boundaries and a set $\mathcal{C}$ of disjoint simple curves in the plane. We say two curves are *equivalent* if they intersect the exact same set of regions from $\mathcal{R}$ (see Figure 6.1(a)). Generally, there could be up to 2^n curves such that no two are equivalent (one for each subset of $\mathcal{R}$). However, we aim to show that if the curves are disjoint, then we cannot have too many such curves. In fact, there can be at most $\Theta(n^3)$ curves such that no two are equivalent.

We mention that a slightly weaker bound can be obtained by using VC-dimension techniques (Section 2.2). To apply these techniques to our problem, we first need to define an appropriate set system $(\mathcal{X}, \mathcal{B})$. The ground set $\mathcal{X}$ in our case is the set of all the regions $\mathcal{R}$. For every curve $c \in \mathcal{C}$ we include the subset of $\mathcal{R}$ crossed by c in $\mathcal{B}$. Notice that equivalent curves result in the same subset of $\mathcal{R}$ and thus size of $\mathcal{B}$ is equal to the number of non-equivalent curves (see Figure 6.1(b)). Thus, we need to show that the set system defined above has constant VC-dimension.

This is done by a K_5-avoidance argument. Let $\mathcal{A}$ be a set of five regions $r_1, \ldots, r_5$. The induced set system $S|_{\mathcal{A}}$ cannot contain all the $\binom{5}{2}$ subsets of size two of $\mathcal{A}$ (see Figure 6.2): The set of curves that cross exactly two regions can be regarded as the set of edges of a graph built on regions. This graph will be planar (since the curves are disjoint by definition) and thus cannot contain a K_5 as a subgraph meaning $\mathcal{A}$ cannot be shattered.

This implies the size of any shattered set is at most four and thus the VC-dimension of the set system is at most four. According to Theorem 2.2.1, the maximum number of non-equivalent curves is $O(n^4)$. While this looks like an interesting bound, it is not tight. In the remainder of this section we describe two improvements upon this bound. We start off with a special case of the problem.

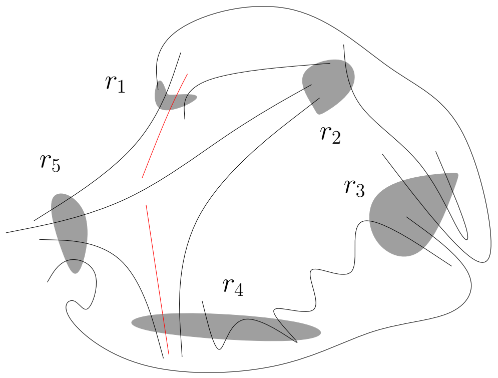


Figure 6.2: It is impossible to connect r_1 to r_4 without either crossing an existing curve or going through a third region.

Lemma 6.1.1. *Assume $\mathcal{R}$ is a set of n disjoint regions and $\mathcal{C}$ is a set of pairwise non-intersecting curves with common endpoints p and q, $p \neq q$. If no two curves are equivalent in $\mathcal{C}$, then $|\mathcal{C}| = O(n)$.*

Proof. Let $c_1, c_2, \ldots, c_m$ be the given curves, ordered clockwise around p. We define c_m to be before c_1 and c_1 to be after c_m, resulting in a cyclic order. For every i, consider two adjacent curves c_i and c_{i+1}. These two curves are not equivalent, so they do not pass through the same set of regions. This implies that there is at least one region r which intersects exactly one of them. We charge c_i to r. Obviously, we have charged $m-1$ units in total. In the cyclic ordering of the curves, consider the curves c_i and c_j such that all the curves from c_i to c_j intersect r and all the curves from c_j to c_i do not intersect r. The total charge of r is at most two since only c_{i-1} and c_j can be charged to r. This proves $|\mathcal{C}| = O(n)$. □

Lemma 6.1.2. (The Main Lemma) *Assume $\mathcal{R}$ is a set of n disjoint regions and $\mathcal{C}$ is a set of pairwise non-intersecting curves. If no two curves are equivalent in $\mathcal{C}$, then $|\mathcal{C}| = O(n^3)$.*

Proof. Consider one curve $c \in \mathcal{C}$. Begin from one endpoint of c and start erasing (or shrinking) c from that endpoint. This process can be viewed as moving the endpoint along the curve c. We continue this shrinking process until the endpoint of c lies on a boundary point p of a region r such that r and c only intersect at p. We repeat the same process for the other endpoint of c and call the resulting curve c'. In other words, the curve c' is a minimal sub-curve of c which intersects the exact same regions as c; thus this operation preserves the equivalence relation. (See Figure 6.3.)

We do this operation on all the curves in $\mathcal{C}$. Let $\mathcal{C}'$ be the set of shrunken curves. Every curve $c \in \mathcal{C}'$ has the property that it starts from (and ends at) the boundary of a region r and never passes through that region again. For two regions r_i and r_j, let $\mathcal{C}_{ij}$ be the set of curves that have one endpoint in r_i and another endpoint in

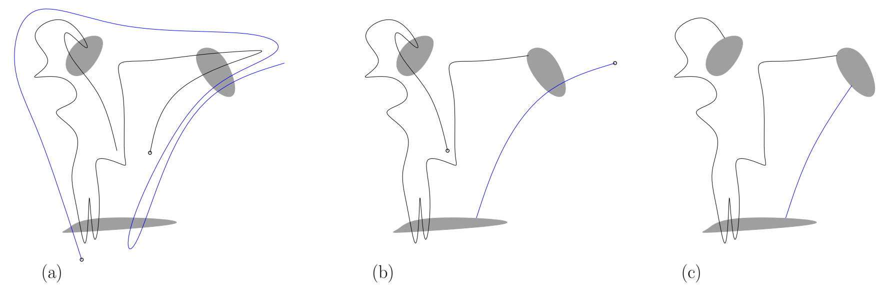


Figure 6.3: (a) Erasing from the endpoints marked with circle. (b) Erasing from the other endpoints. (c) The final curves.

r_j. If we consider only the curves in $\mathcal{C}_{ij}$, then we can contract the regions r_i and r_j to two points and apply Lemma 6.1.1. This implies that $\mathcal{C}_{ij}$ contains $O(n)$ curves. (For the special case $i = j$, we have $|\mathcal{C}_{ii}| \leq 1$, since at most one curve is entirely contained in r_i.) There are $O(n^2)$ different sets of $\mathcal{C}_{ij}$ and thus the total number of curves in $\mathcal{C}'$ and $\mathcal{C}$ is $O(n^3)$. □

The above lemma is optimal in terms of n (as we shall see later) but we can still improve upon it by introducing another parameter:

Lemma 6.1.3. *Assume $\mathcal{R}$ is a set of n disjoint regions and $\mathcal{C}$ is a set of pairwise non-intersecting curves. If no two curves are equivalent in $\mathcal{C}$, then $|\mathcal{C}| = O(nk^2)$ in which k is the maximum number of regions crossed by a curve.*

Proof. We use the random sampling idea of Clarkson and Shor demonstrated in Lemma 1.1.1.

Take a random sample $\mathcal{Q} \subseteq \mathcal{R}$ where each region is included with probability $1/k$. We define a planar (multi)graph $G_\mathcal{Q}$ where vertices are the contracted regions of $\mathcal{Q}$, as follows. Assume the curves have been shrunk as in the earlier proof. Fix two regions r_i and r_j. Let $c_1, \ldots, c_m$ be the curves in $\mathcal{C}_{ij}$ in clockwise order around r_i. For $m = 1$, if r_i and r_j are in $\mathcal{Q}$ and all of the $\leq k$ regions intersecting c_1 are not in $\mathcal{Q}$, then add c_1 to $G_\mathcal{Q}$ as an edge between r_i and r_j. Observe that the probability that c_1 is added is at least $1/k^2(1-1/k)^k = \Omega(1/k^2)$. For $m > 1$, take each consecutive pair (c_t, c_{t+1}) and let $r(c_t, c_{t+1})$ be a region intersected by c_{t+1} but not c_t, or a region intersected by c_t but not c_{t+1}. Color the pair *red* in the former case, and *blue* otherwise. Without loss of generality, assume that at least half of all pairs are red. For a red pair (c_t, c_{t+1}), if r_i, r_j, and $r(c_t, c_{t+1})$ are in $\mathcal{Q}$ and all of the $\leq k$ regions intersecting c_t are not in $\mathcal{Q}$, then add the curve c_t to the graph $G_\mathcal{Q}$ as an edge between r_i and r_j. Observe that the probability that c_t is added is at least $1/k^3(1-1/k)^k = \Omega(1/k^3)$. Thus, $E[\,G_\mathcal{Q}\,] = \Omega(|\mathcal{C}|/k^3)$.

On the other hand, $E[|V(G_\mathcal{Q})|] = O(n/k)$. By Euler's formula, every planar graph with all face lengths at least 3 (in particular, every simple planar graph) has a linear number of edges. Our graph $G_\mathcal{Q}$ is planar but not simple. However, between any 2 parallel edges, there is at least one vertex in $G_{\mathcal{Q}}$: for any two parallel edges c_t and c_u between r_i and r_j, the fact that c_t and c_u are added as edges implies these two curves do not intersect any region except r_i and r_j and furthermore, c_{t+1} intersects $r(c_t, c_{t+1})$ and c_{u+1} intersects $r(c_u, c_{u+1})$. Because of this property, we may assume that $G_{\mathcal{Q}}$ has no faces of length 2 (by adding extra edges to isolated vertices if necessary). Thus, $E[|E(G_{\mathcal{Q}})|] = O(n/k)$.

We can conclude that $|\mathcal{C}|/k^3 = O(n/k)$. $\square$

We can apply Lemmas 6.1.2 and 6.1.3 to the case of connected components formed by a set of axis-parallel rectangles, or more generally, polygons. Assume we have a set $\mathcal{P}$ of polygons. A connected component $\mathcal{C}_i$ is a set of polygons such that any two polygons $p_1, p_2 \in \mathcal{C}_i$ are connected by a chain of intersecting polygons in $\mathcal{C}_i$. Assume $\mathcal{P}$ forms m connected components. Given a set $\mathcal{R}$ of n disjoint regions, we say two connected components are equivalent if and only if the set of regions that they intersect is identical.

Corollary 6.1.4. *Assume a set of simple polygons forms a set $\mathcal{C}$ of connected components and $\mathcal{R}$ is a set of n disjoint regions. If there exists no pairwise equivalent components in $\mathcal{C}$, then $|\mathcal{C}| = O(n^3)$. Furthermore, if X is the total number of intersections of the polygons' boundaries with the regions, then $|\mathcal{C}| = O(n + n^{1/3}X^{2/3})$.*

Proof. If a component c_i completely contains a region r_j (i.e., r_j is completely inside the union of all polygons in c_i), then we delete both c_i and r_j. Since no other component can intersect r_j, this operation preserves the equivalence relation. By repeating this operation, we remove a total of $O(n)$ components and at the end no region is completely contained in a connected component. Thus, a region r_j intersects c_i if and only if r_j intersects the boundary of a polygon in c_i. So, it suffices to consider a set of line segments rather than polygons.

Pick one connected component c_i and look at the arrangement created by the line segments of c_i. Pick one cell except the outer cell of the arrangement and cut one bounding segment of this cell at some arbitrary point (as in Figure 6.4). This operation connects this cell to its neighboring cell. We repeat this for all the other remaining cells until only one cell, the outer cell, is left in the arrangement. Define the curve c_i' as the Eulerian tour of this arrangement.

The resulting curves are disjoint and they intersect the same set of regions as their corresponding connected component. Using Lemma 6.1.2 we conclude that $|\mathcal{C}| = O(n^3)$.

To get a bound sensitive to X, observe that there are at most X/k components intersecting more than k regions. Using Lemma 6.1.3 we conclude that $|\mathcal{C}| = O(nk^2 + X/k)$. We can set $k = (X/n)^{1/3}$. $\square$

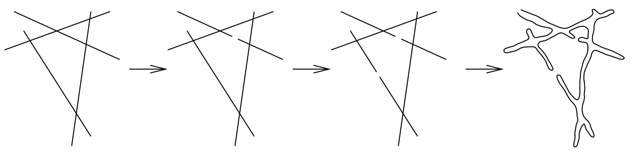


Figure 6.4: Changing a connected component into a curve.

Remark. The bounds in Lemmas 6.1.2 and 6.1.3 and Theorem 6.1.4 are all tight, as we can see from the following example of a set $\mathcal{R}$ of $\Theta(n)$ regions and a set $\mathcal{C}$ of $\Theta(nk^2)$ curves, for any given $k \leq n$: Let $\mathcal{R}$ contain the $2k$ vertical segments $\{i\} \times [0, n+1]$ for $i = -k, \ldots, -1$ and $i = 1, \ldots, k$, as well as n short vertical segments $\{0\} \times [t - \varepsilon, t + \varepsilon]$ for $t = 1, \ldots, n$ (see Figure 6.5(a)). Let $\mathcal{C}$ contain nk^2 horizontal segments $[i, j] \times \{t\}$ for all $i = -k, \ldots, -1$, $j = 1, \ldots, k$, and $t = 1, \ldots, n$. Small perturbations can ensure that the segments in $\mathcal{C}$ are disjoint. No two segments in $\mathcal{C}$ are equivalent. Furthermore, all the curves are horizontal segments. This construction immediately proves the tightness of Lemmas 6.1.2 and 6.1.3. To see the tightness of Corollary 6.1.4, notice that in this construction we have $X = \Theta(nk^3)$ and the number of segments (i.e., curves) is $\Omega(nk^2) = \Omega(n + n^{1/3}X^{2/3})$.

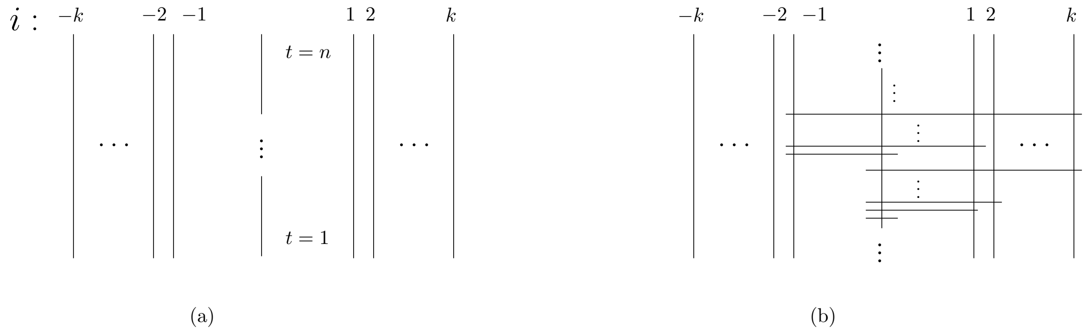


Figure 6.5: A set of $\Theta(n)$ regions and $\Theta(nk^2)$ curves.

One of the main motivations behind combinatorial lemma described in this section, was the dynamic connectivity. As we do not discuss dynamic algorithms in this thesis, we simply mention that using the above lemma, we can obtain a data structure that supports connectivity queries in a set of orthogonal rectangles in $O(1)$ time and updates in $\tilde{O}(n^{10/11}) = O(n^{0.910})$ time [3].

6.2 Intersection Counting for Disjoint Rectilinear Polygons

Let $\mathcal{S}$ be a set of rectilinear disjoint polygons containing a total of n vertices. A query q in our problem is another rectilinear polygon of constant size and the answer must be the number of polygons of $\mathcal{S}$ which intersect q(see Figure 6.6(a)).

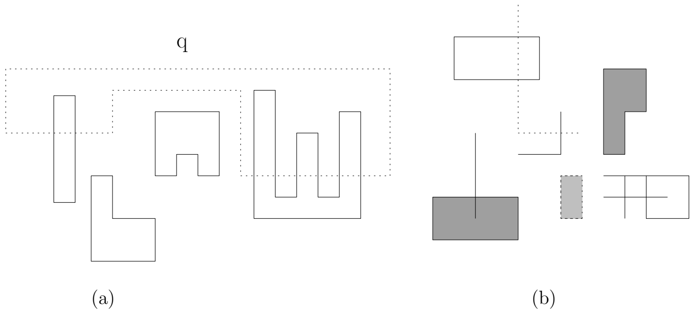


Figure 6.6: (a) An example of disjoint rectilinear polygon counting. (b) The generalization of the problem where the objects are connected components formed by a set of polygons and the query is not even assumed to be a single connected component. If a polygon is shaded then containment counts as intersection for that polygon.

We generalize this problem slightly in the following way. We remove the assumption that the polygons of $\mathcal{S}$ are disjoint but instead ask for the number of connected components intersected by q. Let $\mathcal{C}$ be the set of connected components formed by elements of $\mathcal{S}$. A connected component $c \in \mathcal{C}$ is said to intersect q if q intersects or is contained in at least one of the polygons composing c. Furthermore, we allow the query object to be any set of rectilinear objects of total constant size.

Fix a parameter t and build a $t \times t$ grid by drawing vertical (and similarly horizontal) lines at every n/t-th corner vertex of $\mathcal{S}$ so that each vertical or horizontal slab contains $O(n/t)$ vertices. We call this grid a t-grid (see Figure 6.7).

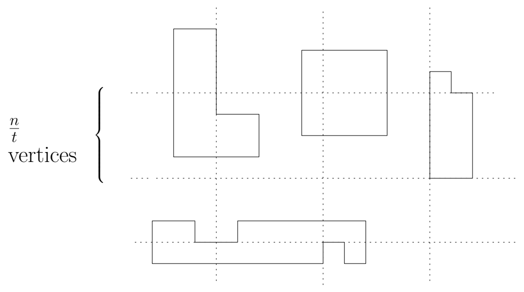


Figure 6.7: An example of a q-grid.

Let the set of regions $\mathcal{R}$ be the set of $\Theta(t^2)$ non-intersecting (vertical and horizontal) line segments which form this grid. We define the equivalence of two connected components as before and give subroutines that help in computing and maintaining the corresponding equivalence classes.

We have developed the combinatorial aspects of our results in Lemmas 6.1.2 and 6.1.3 and Corollary 6.1.4 but for these to have any meaningful algorithmic results we must be able to build and manipulate the set of equivalent classes defined by $\mathcal{R}$. This is done as follows.

Consider a connected component $c \in \mathcal{C}$. We define a canonical *representation* for the set of regions (grid segments) intersected by c, with the intention that two components are equivalent (i.e., they cross the same set of regions) if and only if their representations are identical. (A naive bit-vector representation of size $\Theta(t^2)$ would be too long for our purposes.)

First, if a polygon of c entirely contains a region r, then c is the only component of its equivalence class. In this case we store r as the representation for c. If this is not the case, then we proceed to find a representation for the set of regions intersected by b, the union of all line segments bounding the polygons of c. Consider the i-th row of the grid (a horizontal slab) and let $r_{i,0}, \cdots, r_{i,t}$ be the regions (vertical grid segments) contained in this row, ordered from left to right. We represent the regions intersected by b in this row by a list of *intervals*, $(r_{i,j_1}, r_{i,j'_1}), \cdots, (r_{i,j_k}, r_{i,j'_k})$, where $j_1 \leq j'_1 \leq \cdots \leq j_k \leq j'_k$ (for an example see Figure 6.8). Here, an interval $(r_{i,j}, r_{i,j'})$ indicates that b intersects regions $r_{i,j+1}$ to $r_{i,j'-1}$ but not the regions $r_{i,j}$ and $r_{i,j'}$. We build a similar representation for the columns of the grid (vertical slabs) and define the representation of c to be the concatenation of these lists for all the rows and columns of the grid. Note that the size of this representation is $O(\min\{|c|, t^2\})$, where $|c|$ denotes the number of rectangles in c.

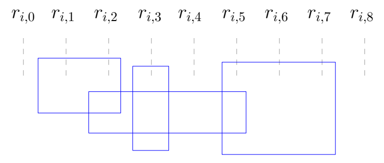


Figure 6.8: A connected component with interval representation of $(r_1, r_4), (r_4, r_8)$ for the i-th row.

Consider a set $\mathcal{S}$ of simple polygons forming a set $\mathcal{C}$ of

Lemma 6.2.1. *Given a t-grid and a set $\mathcal{S}$ of simply polygons of total complexity n we can find the connected components and the set of equivalence classes in $\tilde{O}(n)$ time.*

Proof. The connected components can be found in $O(n \log n)$ time by a sweep-line algorithm [76]. To build the classes, we need to compute the representation for each connected component c. Since elements of $\mathcal{S}$ are rectilinear polygons, we can do this by considering vertical and horizontal segments separately. Within each row, the key subproblem is to compute the union of the x-intervals of the horizontal segments contained in the row. By sorting and scanning, the union of any m given (one-dimensional) intervals can be constructed in $O(m \log m)$ time. Within each column, we have a similar subproblem. The total time to compute the representation for c is therefore $O(|c| \log |c|)$.

If we interpret the representation of the components as long strings, we can compare the representation of two components c_1 and c_2 in $O(\min\{|c_1|, |c_2|\})$ time using the lexicographical ordering of strings. Since the total size of these strings is $O(n)$, we can sort the strings lexicographically in $\tilde{O}(n)$ time, for example, by mergesort. This will put all the elements of the same class in consecutive order. Finally, a linear scan can be used to separate the components into classes. $\square$

Theorem 6.2.2. *Given a set of disjoint simple rectilinear polygons in the plane of total complexity n, we can build a data structure in $\tilde{O}(n)$ time and space, so that we can count the number of polygons intersecting a query component of constant size in $\tilde{O}(n^{6/7})$ time.*

Proof. We build a t-grid for a parameter t and compute the connected components and the corresponding set of classes according to Lemma 6.2.1 and let M be the number of resulting equivalent classes. According to Lemma 6.1.4 we have $M = O(t^6)$. For each polygon, we keep an intersection-search data structure [64]. For each equivalence class ℓ, we record the number of its components. Finally, we break down the class of components intersecting no regions into $O(t^2)$ smaller classes, one for each cell of the grid.

Let q be the query object. We call the slabs of the grid which contain a vertex of q the *special* slabs; each special slab contains $O(n/t)$ corners and thus the total number of components with a corner in these slabs is $O(n/t)$. In $O(n/t)$ time we count such components manually and mark them to prevent double counting in the next step.

Now we claim that if a connected component c_1 of a class ℓ intersects q, then any other member c_2 of ℓ will intersect q as well. We consider two cases. The first case is when an element of q entirely contains c_1 (Figure 6.9(a)). Note that if c_1 does not intersect any region then its cell must be contained by q since we have removed all the components from the special slabs of q. So we can assume c_1 intersects at least one region r_1 (Figure 6.9(b)). Since c_1 does not have any corners in the special slabs (otherwise it would be removed), q entirely contains r_1. Since c_1 and c_2 are equivalent, c_2 must also intersect r_1, and thus q. The second case is when c_1 intersects a bounding segment s of q; say s is horizontal (Figure 6.9(c)). Since c_1 does not have any corners in the special slabs, we can find a grid cell (rectangle) such that c_1 cuts through the cell vertically while s' cuts through it horizontally. Since c_1 and c_2 are equivalent and c_2 does not have any corners in the special slabs either, c_2 must cut through the same cell vertically and must intersect s, and thus q.

The above implies that if an unmarked component of a class ℓ intersects q, then any other unmarked member of ℓ intersects q as well. So, we go through each class ℓ, and if an arbitrary unmarked component in ℓ intersects q, we increase the counter by the number of unmarked components in ℓ. This takes $O(M) = \tilde{O}(t^6)$ additional time. The total query time $\tilde{O}(t^6 + n/t)$ is asymptotically minimized for $t = n^{1/7}$. $\square$

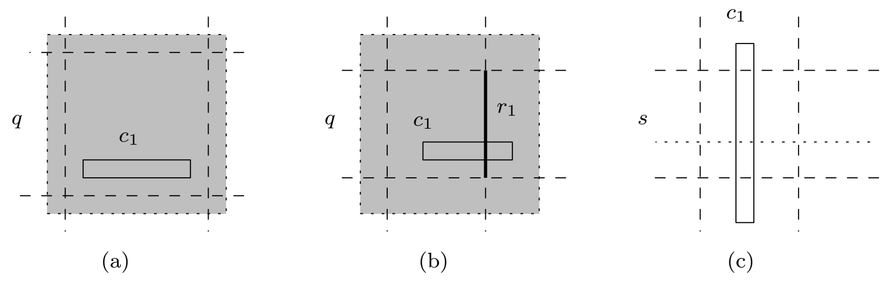


Figure 6.9: The query object (or parts of it) is drawn with dotted lines and the segments of the grid is drawn with dashed lines. c_1 represents a connected component.

Chapter 7

Conclusion

In this thesis we have studied problems in range searching, intersection counting, and statistical depth. While we have resolved some of the problems we consider, there are still questions left open.

As we have discussed in Chapter 2, one general important question is how to improve the efficiency of range searching data structures. We have provided partial answers to this question using relative approximations, but there are still many ideas (such as assuming a probability distribution on the query range and obtaining a distribution sensitive range searching data structure) that can be tried.

The results in Chapter 3 leave two open questions: approximate Tukey depth and regression depth queries in 3D. For the former problem we have provided an almost optimal answer, with an extra $\log\log n$ factor, but we have not obtained any nontrivial results for the second problem. The trivial lower bound for both problems is linear space and $\Omega(\log n)$ query time.

In Chapter 4, we have described the first data structure to approximate planar simplicial depth queries in $\tilde{O}(1)$ time using $\tilde{O}(1)$ space; however, we have not optimized the logarithmic factors for our 2D approximate simplicial depth data structure. While it seems difficult to obtain an data structure with linear space and $O(\log n)$ query time, some of the logarithmic factors could perhaps be removed. Another important problem is to improve the efficiency of the simplicial depth algorithms (exact or approximate) in higher dimensions. Obtaining any nontrivial lower bound for this problem would also be an important result, as there are none in the literature at the moment.

In Chapter 5, we have shown that the dominance reporting problem reduces to point location in a rectilinear arrangement. Thus, an improved point location data structure in the word RAM model would result in an improved orthogonal range searching data structure in 3 and higher dimensions. Alternatively, we can directly turn to the orthogonal range searching data structures and try to improve the space bounds of the data structures obtained here. A very recent manuscript by Karpinski and Nekrich [81] reduces some of the $\log n$ factors from the space complexity but increases the query time slightly to $O(\log\log U + (\log\log n)^3 + k)$. Another open question is whether it is possible to prove an analogue of Lemma 5.4.1 for halfspace range reporting in 3D. If so, then halfspace range reporting can be solved optimally in various models of computation as well (In a very recent collaboration with Timothy M. Chan, we have solved the halfspace range reporting model in the standard RAM model but the solution in the external memory model is still a $\log^* n$ factor off from optimal [2]). The lower bound we present in the appendix shows that we cannot hope to obtain an analogue of Lemma 5.4.1 via a partition.

In Chapter 6, we have described a combinatorial lemma regarding a set of regions and a set of curves. Using this we have obtained a data structure for disjoint rectilinear polygon counting with $\tilde{O}(n^{6/7})$ query time. It would be interesting to see what other applications can be found for this lemma. Perhaps it is possible to lower the exponent of our query time, although a polylogarithmic query time seems out of reach. Also, since the combinatorial lemma applies to non-rectilinear polygons as well, another open question is whether a similar counting data structure can be built without the rectilinear assumption. In general, it would be worthwhile to have a more detailed study of the connection between this combinatorial lemma, intersection searching and "VC-dimension" of disjoint polygons.

Bibliography

[1] P. Afshani. On dominance reporting in 3D. In *ESA'08: Proceedings of the 16th conference on Annual European Symposium*, pages 41–51, 2008. 20

[2] P. Afshani and T. M. Chan. Optimal halfspace range reporting in three dimensions. To appear in *SODA'09: Proceedings of the 20th Annual Symposium on Discrete Algorithms.* 76

[3] P. Afshani and T. M. Chan. Dynamic connectivity for axis-parallel rectangles. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 16–27, 2006. *Algorithmica*, to appear. 20, 70

[4] P. Afshani and T. M. Chan. On approximate range counting and depth. In *SCG '07: Proceedings of the 23rd Annual Symposium on Computational Geometry*, pages 337–343. ACM Press, 2007. Invited to Discrete and Computational Geometry (SOCG special issue). 19

[5] P. K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *CRC Handbook of Discrete and Computational Geometry.* CRC Press, Inc., 2004. 3

[6] P. K. Agarwal, L. Arge, J. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient searching with linear constraints. *Journal of Computer and System Sciences*, 61(2):194–216, 2000. 56

[7] P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete and Computational Geometry*, Volume 19:315–331, 1998. 5

[8] P. K. Agarwal, B. Aronov, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. In *SCG '97: Proceedings of the 13th Annual Symposium on Computational Geometry*, pages 30–38. ACM, 1997. 5

[9] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 2000. 58

[10] 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, 1999. 3

[11] P. K. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudo-circles and their applications. *Journal of the ACM*, 51(2):139–186, 2004. 5

[12] P. K. Agarwal and M. Sharir. Pseudo-line arrangements: duality, algorithms, and applications. *SIAM Journal on Computing*, 34(3):526–552, 2005. 5

[13] P. K. Agarwal, M. Sharir, and E. Welzl. Algorithms for center and Tverberg points. In *SCG '04: Proceedings of the 12th Annual Symposium on Computational Geometry*, pages 61–67. ACM Press, 2004. 19

[14] P. K. Agarwal and M. van Kreveld. Polygon and connected component intersection searching. *Algorithmica*, 15:626–660, 1996. 65

[15] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *STOC '90: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 331–340. ACM Press, 1990. 12

[16] G. Aloupis, C. Cortés, F. Gómez, M. Soss, and G. Toussaint. Lower bounds for computing statistical depth. *Journal of Computer and System Sciences*, 40(2):223–229, 2002. 16, 17

[17] G. Aloupis, S. Langerman, M. Soss, and G. Toussaint. Algorithms for bivariate medians and a Fermat–Torricelli problem for lines. *Computational Geometry Theory and Applications*, 26(1):69–79, 2003. 17

[18] S. Alstrup, G. S. Brodal, and T. Rauhe. New data structures for orthogonal range searching. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 198. IEEE Computer Society, 2000. 13, 57

[19] N. Amenta, M. W. Bern, D. Eppstein, and S.-H. Teng. Regression depth and center points. *Discrete and Computational Geometry*, 23:305–323, 1998. 18, 19

[20] L. Arge, V. Samoladas, and J. S. Vitter. On two-dimensional indexability and optimal range search indexing. In *PODS '99: Proceedings of the 18th Symposium on Principles of Database Systems*, pages 346–357. ACM, 1999. 14

[21] B. Aronov and S. Har-Peled. On approximating the depth and related problems. In *SODA '05: Proceedings of the 16th Annual Symposium on Discrete Algorithms*, pages 886–894. Society for Industrial and Applied Mathematics, 2005. 26, 29, 30

[22] B. Aronov and S. Har-Peled. On approximating the depth and related problems, 2005. http://valis.cs.uiuc.edu/~sariel/research/papers/04/depth/. 30

[23] B. Aronov, S. Har-Peled, and M. Sharir. On approximate halfspace range counting and relative epsilon-approximations. In *SCG '07: Proceedings of the 23rd Annual Symposium on Computational Geometry*, pages 327–336. ACM, 2007. 62

[24] B. Aronov, M. Pellegrini, and M. Sharir. On the zone of a surface in a hyperplane arrangement. *Discrete and Computational Geometry*, 9(2):177–186, 1993. 50

[25] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate spherical range counting. In *SODA '05: Proceedings of the 16th Annual Symposium on Discrete Algorithms*, pages 535–544. Society for Industrial and Applied Mathematics, 2005. 22

[26] S. Arya and D. M. Mount. Approximate range searching. *Computational Geometry Theory and Applications*, 17(3-4):135–152, 2000. 21

[27] D. Avis. The m-core properly contains the m-divisible points in space. *Pattern Recognition Letters*, 14(9):703–705, 1993. 19

[28] A. Bagchi, A. Chaudhary, D. Eppstein, and M. T. Goodrich. Deterministic sampling and range counting in geometric data streams. In *SCG '04: Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 144–151. ACM Press, 2004. 39

[29] I. Bárány. A generalization of Carathédory's theorem. *Discrete Mathematics*, 40:141–152, 1982. 40

[30] J. L. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980. 13

[31] M. Bern and D. Eppstein. Multivariate regression depth. *Discrete and Computational Geometry*, 28:1–17, 2002. 34

[32] H. Brönnimann, B. Chazelle, and J. Pach. How hard is half-space range searching? *Discrete and Computational Geometry*, 10:143–155, 1993. 12

[33] C. Carathéodory. Über den variabilitätsbereich der koeffizienten von potenzreihen, die gegebene werte nicht annehmen. *Math. Ann*, 64:95–115, 1907. 40

[34] C. Carathéodory. Über den variabilitätsbereich der fourierschen konstanten für positiven harmonischen funktionen. *Rend. Circ. Mat. Palermo*, 32:193–217, 1911. 40

[35] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 586–595. IEEE Computer Society, 1998. 12

[36] T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34:879–893, 2000. 10, 49

[37] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000. 12, 32, 50

[38] T. M. Chan. On enumerating and selecting distances. *International Journal of Computational Geometry and Applications*, 11:291–304, 2001. 29

[39] T. M. Chan. On levels in arrangements of curves. *Discrete and Computational Geometry*, 29:375–393, 2003. 5

[40] T. M. Chan. On levels in arrangements of curves, II: A simple inequality and its consequences. *Discrete and Computational Geometry*, 34:11–24, 2004. 5

[41] T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *SODA '04: Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 430–436. Society for Industrial and Applied Mathematics, 2004. 17

[42] T. M. Chan. On levels in arrangements of surfaces in three dimensions. In *SODA '05: Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms*, pages 232–240. Society for Industrial and Applied Mathematics, 2005. 5

[43] T. M. Chan. On levels in arrangements of curves, III: further improvements. In *SCG'08: Proceedings of the 24th Annual Symposium on Computational Geometry*, pages 85–93, 2008. 5

[44] B. Chazelle. Filtering search: a new approach to query answering. *SIAM Journal on Computing*, 15(3):703–724, 1986. 13, 14

[45] B. Chazelle. Functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988. 13

[46] B. Chazelle. Lower bounds on the complexity of polytope range searching. *Journal of the American Mathematical Society*, 2:637–666, 1989. 11

[47] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. 7

[48] B. Chazelle. Cuttings. In *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2005. 7

[49] B. Chazelle and H. Edelsbrunner. Linear space data structures for two types of range search. *Discrete and Computational Geometry*, Volume 2(1):113,126, 1987. 15, 56, 59

[50] B. Chazelle and L. J. Guibas. Fractional cascading: A data structuring technique with geometric applications. *Algorithmica*, 1:133–162, 1986. 13

[51] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76–90, 1985. 12

[52] B. Chazelle, D. Liu, and A. Magen. Approximate range searching in higher dimension. *Computational Geometry Theory and Applications*, 39(1):24–29, 2008. 22

[53] B. Chazelle and F. P. Preparata. Halfspace range search: an algorithmic application of k-sets. *Discrete and Computational Geometry*, 1(1):83–93, 1986. 12

[54] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992. 11

[55] A. Y. Cheng and M. Ouyang. On algorithms for simplicial depth. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 53–56, 2001. 18

[56] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4:387–421, 1989. 6, 12, 35

[57] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997. 30

[58] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discrete and Computational Geometry*, 14:261–286, 1995. 50

[59] M. de Berg and O. Schwarzkopf. Cuttings and application. *International Journal of Computational Geometry and Applications*, 5:343–355, 1995. 36

[60] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997. 4, 24

[61] M. de Berg, M. van Kreveld, and J. Snoeyink. Two- and three-dimensional point location in rectangular subdivisions. *Journal of Algorithms*, 18(2):256–277, 1995. 59, 60

[62] T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete and Computational Geometry*, 19:373–382, 1998. 5

[63] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1987. 10, 34

[64] H. Edelsbrunner and H. A. Maurer. On the intersection of orthogonal objects. *Information Processing Letters*, 13:177–181, 1981. 73

[65] P. Erdős, L. Lovász, A. Simmons, and E. Straus. Dissection graphs of planar point sets. In J. N. Srivastava, editor, *A Survey of Combinatorial Theory*, pages 139–154. North-Holland, 1973. 4, 5

[66] J. Erickson. New lower bounds for halfspace emptiness. In *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 472–481. IEEE Computer Society, 1996. 12

[67] J. Erickson. Space-time tradeoffs for emptiness queries. *SIAM Journal on Computing*, 29(6):1968–1996, 2000. 12

[68] M. L. Fredman. A lower bound on the complexity of orthogonal range queries. *Journal of the ACM*, 28(4):696–705, 1981. 11

[69] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993. 13

[70] J. Gil, W. Steiger, and A. Wigderson. Geometric medians. *Discrete Mathematics*, 108(1-3):37–51, 1992. 17, 18, 50

[71] P. M. Gruber and J. M. Wills, editors. *Handbook of Convex Geometry : Two-Volume Set*. North Holland, August 1993. 40

[72] P. Gupta, R. Janardan, and M. Smid. Computational geometry: generalized intersection searching. In *Handbook of Data Structures and Applications*, chapter 64, pages 1–17. Chapman & Hall/CRC, 2005. 65

[73] S. Har-Peled and M. Sharir. Relative ϵ-approximations in geometry, 2006. http://valis.cs.uiuc.edu/~sariel/research/papers/06/relative/. 30

[74] S. Hart and M. Sharir. Nonlinearity of Daveport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6(2):151–177, 1986. 36

[75] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete and Computational Geometry*, 2:127–151, 1987. 23, 24

[76] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310–323, 1983. 72

[77] S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete and Computational Geometry*, 12:291–312, 1994. 17

[78] J. JaJa, C. W. Mortensen, and Q. Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *ISAAC 2004: International Symposium on Algorithms and Computation*, pages 558–568, 2004. 15, 56

[79] D. S. Johnson. Approximation algorithms for combinatorial problems. In *STOC '73: Proceedings of the 5th annual ACM symposium on Theory of computing*, pages 38–49. ACM, 1973. 23

[80] H. Kaplan and M. Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar Voronoi diagrams, and approximate range counting. In *SODA '06: Proceedings of the 17th Annual Symposium on Discrete Algorithms*, pages 484–493. ACM Press, 2006. 26, 30, 32

[81] M. Karpinski and Y. Nekrich. Space-efficient multi-dimensional range reporting. see http://arxiv.org/abs/0806.4361v1. 75

[82] N. Katoh and T. Tokuyama. k-levels of concave surfaces. *Discrete and Computational Geometry*, 27:567–584, 2002. 5

[83] S. Khuller and J. S. B. Mitchell. On a triangle counting problem. *Information Processing Letters*, 33(6):319–321, 1990. 17

[84] J. H. Kim and V. H. Vu. Concentration of multivariate polynomials and its applications. *Combinatorica*, 20:417–434, 2000. 42

[85] S. Langerman and W. Steiger. An optimal algorithm for hyperplane depth in the plane. In *SODA '00: Proceedings of the 11th Annual Symposium on Discrete Algorithms*, pages 54–59. Society for Industrial and Applied Mathematics, 2000. 18

[86] S. Langerman and W. L. Steiger. Optimization in arrangements. In *STACS '03: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, pages 50–61. Springer-Verlag, 2003. 17, 18

[87] C.-Y. Lo and W. L. Steiger. An optimal time algorithm for ham-sandwich cuts in the plane. In *Proceedings of the 2nd Canadian Conference on Computational Geometry*, pages 5–9, 1990. 8

[88] L. Lovász. On the number of halving lines. *Annal. Univ. Scie. Budapest. de Rolando Eötvös Nominatae, Sectio Math.*, 14:107–108, 1971. 4, 5

[89] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975. 23

[90] G. S. Lueker. A data structure for orthogonal range queries. In *FOCS '78: Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 28–34, 1978. 13

[91] C. Makris and A. Tsakalidis. Algorithms for three-dimensional dominance searching in linear space. *Inf. Process. Lett.*, 66(6):277–283, 1998. 15, 56, 59

[92] A. Marcus and G. Tardos. Intersection reverse sequences and geometric applications. *Journal of Combinatorial Theory, Series A*, 113(4):675–691, 2006. 5

[93] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(2):157–182, 1993. 9, 11

[94] J. Matoušek. Geometric set systems. *European Congress of Mathematics*, 2:1–27, 1998. 23

[95] J. Matoušek. Computing the center of a planar point set. *Discrete and Computational Geometry*, pages 221–230, 1991. 17

[96] J. Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8(3):315–334, 1992. 8, 9

[97] J. Matoušek. Reporting points in halfspaces. *Computational Geometry Theory and Applications*, 2(3):169–186, 1992. 9, 10, 11, 87

[98] J. Matoušek. Tight upper bounds for the discrepancy of halfspaces. *Discrete and Computational Geometry*, 13:593–601, 1995. 23

[99] J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, 2002. 23

[100] J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and approximations for bounded vc-dimension. *Combinatorica*, 13:455–466, 1995. 23

[101] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985. 13

[102] Y. Nekrich. A data structure for multi-dimensional range reporting. In *SCG '07: Proceedings of the 23rd Annual Symposium on Computational Geometry*, pages 344–353. ACM, 2007. 13, 14, 15, 55, 56, 57, 63

[103] Y. Nekrich. I/O-efficient point location in a set of rectangles. In *LATIN 2008: Theoretical Informatics*, pages 687–698, 2008. 56, 63

[104] J. Pach, W. Steiger, and E. Szemerédi. An upper bound on the number of planar k-sets. *Discrete and Computational Geometry*, 7(2):109–123, 1992. 5

[105] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *STOC '06: Proceedings of the thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 232–240, 2006. 13

[106] M. Pătraşcu and M. Thorup. Randomization does not help searching predecessors. In *SODA '07: Proceedings of the 18th Annual Symposium on Discrete Algorithms*, pages 555–564, 2007. 13

[107] R. Pinchasi and R. Radoicic. Topological graphs with no self-intersecting cycle of length 4. In *SCG '03: Proceedings of the 19th Annual Symposium on Computational Geometry*, pages 98–103. ACM, 2003. 5

[108] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985. 10

[109] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *SCG '99: Proceedings of the 15th Annual Symposium on Computational Geometry*, pages 390–399. ACM Press, 1999. 10, 12

[110] P. J. Rousseeuw and M. Hubert. Regression depth. *Journal of American Statistical Association*, 94(446):388–402, 1999. 18

[111] P. J. Rousseeuw and I. Ruts. Bivariate location depth. *Journal of Applied Statistics*, 45(4):516–526, 1996. 17

[112] M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for k-sets in three dimensions. *Discrete and Computational Geometry*, 26:195–204, 2001. 5, 31

[113] S. Subramanian and S. Ramaswamy. The P-range tree: a new data structure for range searching in secondary memory. In *SODA '95: Proceedings of the 6th annual ACM-SIAM symposium on Discrete algorithms*, pages 378–387, 1995. 14, 15

[114] M. Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de L'IHÉS*, 81:73–205, 1995. 42

[115] H. Tamaki and T. Tokuyama. How to cut pseudoparabolas into segments. *Discrete and Computational Geometry*, 19:265–290, 1998. 5

[116] H. Tamaki and T. Tokuyama. A characterization of planar graphs by pseudo-line arrangements. *Algorithmica*, 35:269–285, 2003. 5

[117] G. Tóth. Point sets with many k-sets. *Discrete and Computational Geometry*, 26:187–194, 2001. 5

[118] H. Tverberg. A generalization of Radon's theorem. *Journal of the London Mathematical Society*, 41:123–128, 1966. 19

[119] V. K. Vaishnavi and D. Wood. Rectilinear line segment intersection, layered segment trees, and dynamization. *Journal of Algorithms*, 3:160–176, 1982. 65

[120] M. van Kreveld, J. S. B. Mitchell, P. Rousseeuw, M. Sharir, J. Snoeyink, and B. Speckmann. Efficient algorithms for maximum regression depth. In *SCG '99: Proceedings of the 15th Annual Symposium on Computational Geometry*, pages 31–40. ACM Press, 1999. 18, 34, 35

[121] D. E. Vengroff and J. S. Vitter. Efficient 3-D range searching in external memory. In *STOC '96: Proceedings of the 28th Annual ACM Aymposium on Theory of Computing*, pages 192–201. ACM, 1996. 14, 15, 55, 56, 57

[122] J. S. Vitter. External memory algorithms and data structures: dealing with massive data. *ACM Comput. Surv.*, 33(2):209–271, 2001. 15, 60

[123] E. Welzl and H. Edelsbrunner. Constructing belts in two-dimensional arrangements with applications. *SIAM Journal on Computing*, 15:271–284, 1986. 29, 35

[124] D. E. Willard. Log-logarithmic worst-case range queries are possible in space $\theta(n)$. *Information Processing Letters*, 17:81–84, 1983. 13

[125] A. C. Yao. On the complexity of maintaining partial sums. *SIAM Journal on Computing*, 14:277–288, 1985. 11

Appendices

A Lower Bound on the 3D Shallow Partition Theorem

In his 1992 paper, Matoušek [97] asked whether the $O(\log r)$ crossing number bound in the 3D shallow partition theorem (Theorem 1.1.7) is optimal. Besides being a natural combinatorial question, lowering crossing number to $O(1)$ would have potential algorithmic applications. We show, however, that $O(1)$ is not possible. Given two parameters r and t, $t \geq r$, we describe a point set $\mathcal{P}$ of $O(tr)$ points such that for any partition of $\mathcal{P}$ into r sets of size t there exists a halfspace with crossing number $\Omega(\log r / \log\log r)$ that contains at most $2t$ points. To improve the readability, we first describe the combinatorial structure of the point set. For simplicity, we assume that t and r are powers of two.

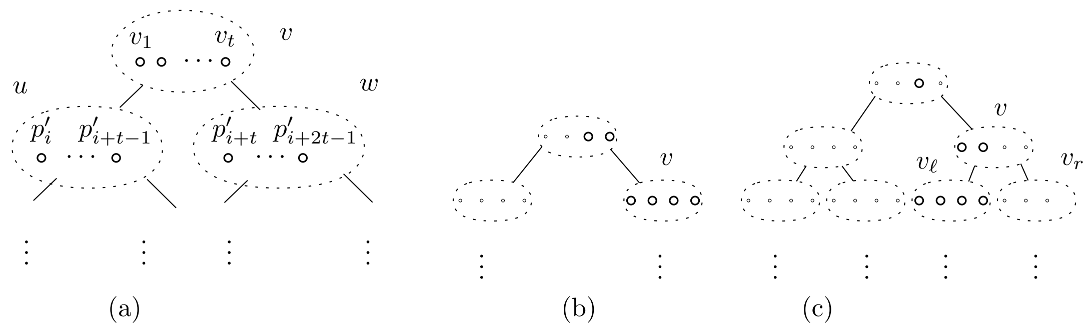


Figure 7.1: (a) The tree T. (b) The larger circles represents the elements of $h(v)$. (c) We have chosen to include the left child of v and thus the right half of every list is discarded.

Consider a full binary tree T with r leaves with an ordered list of t points placed in each node of the tree (Figure 7.1(a)); this order is important and will play a crucial role when embedding the points in 3D. The height of T is $\log r$ and in total contains $O(tr)$ points. We claim it is possible to embed these points in 3D such that a set of "basic" halfspaces with the following definition exists. For every node v, a unique basic halfspace exactly contains all the points of v together with a sublist of the points in every ancestor of v; we denote this halfspace with

$h(v)$. Let v_ℓ and v_r be the left and right child of v respectively. To obtain $h(v_\ell)$, first we discard right half of every list in $h(v)$, then include all the points in v_ℓ (Figure 7.1(b,c)). $h(v_r)$ can be formed in a similar fashion. It is easy to see that all the basic halfspaces have less than $2t$ points.

Consider a partition of this point set into subsets of size $O(t)$. Think of points in the same subset as having the same color. We want to find a basic halfspace that contains $\Omega(\log r/\log\log r)$ colors. Let m be the maximum number of colors contained in a basic halfspace and $d := c\log m$ for a sufficiently large constant c.

Consider a random walk $v_1, v_2, \ldots, v_{\log r}$ from the root to the leaves in which at each inner node the probability of descending to one of the two children is $1/2$. We use $X_i, i > d$, to denote the event that the node v_i contains at most $t/4$ points with colors that also appear in $h(v_{i-d})$. $h(v_{i-d})$ contains at most m colors, so the tree contains $O(tm)$ points that share colors with $h(v_{i-d})$. The number of descendants of v_{i-d} at level i is $2^d = 2^c m$ thus on average any of them has $O(tm)/(2^c m)$ points that share colors with $h(v_{i-d})$. This means by choosing c large enough we can ensure that $\Pr[\,X_i\,] \geq 2/3$.

Let Y_i be the event that at most half of the points of $h(v_{\log r}) \cap v_i$ have colors which also appear in $h(v_{i-d})$. For any two leaves u and w that are descendants of v_i, the subsets $h(u) \cap v_i$ and $h(w) \cap v_i$ are disjoint. Using this property, it is easy to see that $\Pr[\,Y_i \,|\, X_i\,] \geq 1/2$ and thus $\Pr[\,Y_i\,] \geq \Pr[\,Y_i \wedge X_i\,] \geq 1/3$. This implies in the sequence v_{1+id}, $0 \leq i < (\log r)/d$, on average $(\log r)/3d$ of the events Y_{1+id} happen. Each such event marks the appearance of a new color in $h(v_{\log r})$ as we scan the colors of $h(v_{\log r})$ from root to leaf. This proves the existence of a basic halfspace with $(\log r)/(3d)$ colors, so $(\log r)/(3d) \leq m$, which solves to $m = \Omega(\log r/\log\log r)$.

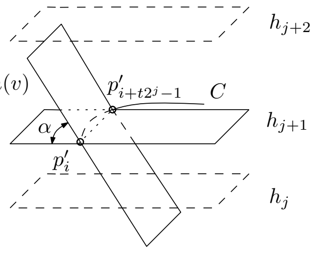


Figure 7.2: We are free to choose the angle α and the distance from the next plane h_{j+2}.

It remains to show how to embed the claimed set of points. Consider the unit circle centered on origin of the xy-plane and let C be the part which lies in the first quadrant of the plane. We embed the points such that their projection (which we denote with $\pi(\cdot)$) on the xy-plane lies on C. Also, consider $\log r$ planes $h_1, \ldots, h_{\log r}$ all parallel to the xy-plane in which h_i is determined by the equation $Z = -d_i$. We will set the value of each d_i later but for the moment assume $d_1 = 0 < d_2 < \cdots < d_{\log r}$. All the points at level i of the tree are placed on $h_{\log r - i+1}$

(the level of the root is zero). Thus, for a point at level i of the tree, knowing its the projection on C completely determines its location in space. Let $p_1, \ldots, p_{tr}$ be the sequence of points contained in the leaves, as read from the leftmost leaf to the rightmost one. Position them such that their projection on C lies in the above order when read clockwise (the amount of spacing between the points is not important and we only require it to be non-zero). We say a point q lies between p_i and p_j if $\pi(q)$ lies on the arc $\pi\widehat{(p_i)\pi(p_j})$.

Consider an inner node v at height j with points $v_1, \ldots, v_t$ (height of the leaves is defined to be zero). Assume the list of points contained in the leaves of the tree rooted by v is $p_i, \ldots, p_{i+t2^j-1}$. Position the points of v such that the first $t/2^j$ points (v_1 up to $v_{t/2^j}$) lie between p_i and p_{i+t-1}, the next $t/2^j$ between p_{i+t} and p_{i+2t-1} and continue until all the points of v are in place (as a reminded, they are being placed on h_{j+1}). Let $p'_i, \ldots, p'_{i+t2^j-1}$ be the projection of the points $p_i, \ldots, p_{i+t2^j-1}$ on h_{j+1}. The basic halfspace $h(v)$ is determined by a hyperplane which passes through p'_i and p'_{i+t2^j-1}. From Figure 7.2 it can be argued that by picking the angle α sufficiently close to $\pi/2$ we can guarantee that $h(v)$ exactly includes the points which are below h_{j+1} and between p_i and p_{i+t2^j-1}; next, by choosing h_{j+2} sufficiently above the h_{j+1} we can ensure that no point above h_{j+1} is in $h(v)$.

This proves the bound on the crossing number in the shallow partition theorem cannot be improved beyond $\Omega(\log r/\log\log r)$ for $r \leq \sqrt{n}$. We remark that the $\Omega(\log r/\log\log r)$ bound is tight for the family of point sets described here.