On Geometric Range Searching and Its Variants

Pingan Cheng

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

On Geometric Range Searching and Its Variants

A Dissertation
Presented to the Faculty of Natural Sciences
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
Pingan Cheng
July 31, 2023

Abstract

Given a set P of points in $\mathbb{R}^d$, how to preprocess P into a structure so that for any query range γ, we can find $P \cap \gamma$ efficiently? This problem, known as range searching, is one of the core problems in computational geometry and has countless variants. In this dissertation, we explore both the classic and modern variants of this problem.

First, we consider "semialgebraic range searching", arguably the most general version of range searching, where each query range is described by a constant number of polynomial inequalities. Almost all natural range queries, e.g., orthogonal box queries, simplex queries, polytope queries, ball queries, and so on, can be formulated as special cases of semialgebraic range queries. The study of this problem dates back to the 1980s, but efficient data structures were known only very recently using new powerful tools introduced by the "polynomial revolution" in the last decade. In this dissertation, we complement these upper bound results by presenting the first (and almost matching) lower bound to the problem.

Then, we turn to intersection searching, a generalization of range searching where the input (resp. query) objects are t (resp. $(d-t)$) dimensional for $0 \le t \le d$. Note that range searching is the special case where $t = 0$. Similar to semialgebraic range searching, new upper bounds were discovered recently using polynomial techniques, but all these upper bounds only partially improve the old bounds and seem to be very difficult to generalize. We explain this phenomenon and reveal the limitation of polynomial techniques by presenting lower bounds for these problems.

The next topic we explore is range summary queries, a variant that appeared in recent years motivated by the fact that the data size in real-world applications has become so big that simply reporting or counting the input intersecting a query is no longer efficient or useful. Instead, it is more desirable to provide a data summary for the output $P \cap \gamma$. Two of the most useful summaries are ε-heavy hitter summaries, i.e., elements with frequency at least $\varepsilon|P \cap \gamma|$, and ε-quantile summaries, i.e., a sequence of $\varepsilon^{-1}+1$ elements representing $i\varepsilon^{-1}$ quantiles in $P \cap \gamma$. We study how to generate these summaries efficiently for halfspace and dominance range queries.

Finally, we speed up multiple point location queries on (unrelated) axis-aligned planar subdivisions by generalizing a data structure technique known as "fractional cascading". Point location can be viewed as (a variant of) the dual of range searching, where we are given a planar subdivision (an embedding of a planar graph with straight line segments) as the input and we want to locate a query point in the subdivision. We provide upper and lower bounds for a variety of settings of the problem.

Resumé

Givet et sæt P af punkter i $\mathbb{R}^d$, hvordan forbehandler man P til en struktur så man effektivt kan finde $P \cap \gamma$ for hvert forespørgselsområde γ? Dette problem, kendt som områdesøgning, er et af kerneproblemerne inden for beregnlighedsgeometri og har utallige varianter. I denne afhandling undersøger vi både de klassiske og moderne varianter af denne problemstilling.

Vi først overvejer “semialgebraisk områdesøgning”, som formodes at være den mest generelle version af områdesøgning, hvor hvert forespørgselsområde er beskrevet af et konstant antal polynomielle uligheder. Næsten alle naturlige områdeforespørgsler, som f.eks. ortogonale boksforespørgsler, simpleksforespørgsler, polytopforespørgsler, kugleforespørgsler og så videre, kan formuleres som specielle tilfælde af semialgebraiske områdeforespørgsler. Undersøgelsen af dette problem går helt tilbage til 1980’erne men effektive datastrukturer var først kendt for ganske nylig ved hjælp af nye kraftfulde værktøjer introduceret af den "polynomiellerevolution" i det sidste årti. I denne afhandling supplerer vi disse øvre grænse resultater ved at præsentere den første (og næsten matchende) nedre grænse for problemet.

Derefter vender vi os mod krydssøgning, en generalisering af områdesøgning hvor input (hhv. forespørgsel) objekter er t (hhv. $(d-t)$) dimensionelle for $0 \leq t \leq d$. Bemærk, at områdesøgning er det specialtilfælde, hvor $t = 0$. Lignende med semialgebraisk områdesøgning, blev nye øvre grænser opdaget for nylig ved hjælp af polynomielle teknikker, men alle disse øvre grænser forbedrer kun delvist de gamle grænser og ser ud til at være meget svære at generalisere. Vi forklarer dette fænomen og afslører begrænsningen af polynomielle teknikker ved at præsentere nedre grænser for disse problemer.

Det næste emne, vi undersøger, er områderesuméforespørgsler, en variant, der dukker op i de senere år motiveret af det faktum, at datastørrelsen i applikationer fra den virkelige verden er blevet så stor at blot at rapportere eller tælle input, der krydser en forespørgsel, ikke længere er effektivt eller nyttigt. I stedet ønsker man at give et dataresumé for output $P \cap \gamma$. To af de mest nyttige resuméer er ε-heavy hitter-resuméer, dvs. elementer med en frekvens på mindst $\varepsilon|P \cap \gamma|$, og ε-kvantilresuméer, dvs. en sekvens af $\varepsilon^{-1} + 1$ elementer, der repræsenterer $i\varepsilon^{-1}$ kvantiteter i $P \cap \gamma$. Vi studerer, hvordan man genererer disse resuméer effektivt for halfspace og dominansområde forespørgsler .

Til sidst fremskynder vi flere punktlokaliseringsforespørgsler på (ikke relateret) aksejusterede plane underinddelinger ved at generalisere en datastrukturteknik kendt som “fractional cascading”. Punktlokalisering kan ses som (en variant af) den dobbelte områdesøgning, hvor vi får en plan underinddeling (en indlejring af en plan graf med lige linjesegmenter) som input og vi ønsker at lokalisere et forespørgselspunkt i underafdelingen. Vi giver øvre og nedre grænser for en række forskellige tilfælde af problemet.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my PhD advisor Peyman Afshani for his persistent encouragement and trust throughout the last four years, without which I could never finish this marvelous journey myself. I want to thank Peyman for introducing me to the world of computational geometry, teaching me how to be “more brave” as a researcher, and always being there to offer me invaluable advice. I would also like to thank Peyman for the extremely helpful comments and feedback during the preparation of this dissertation. I will always cherish the days we worked together. Thank you, Peyman! I also wish to thank Pankaj K. Agarwal and Esther Ezra for agreeing to be on my PhD committee.

I also thank everyone else I met in the algorithm and complexity group during my PhD studies: Lars Arge, Amik Raj Behera, Gerth Stølting Brodal, Jakob Burkhardt, Ioannis Caragiannis, Asger Drewsen, Ora Nova Fandina, Casper Benjamin Freksen, Karl Albert Friedrich Fehrs, Allan Grønlund, Zeynep Gündogar, Kristoffer Arnsfelt Hansen, Sebastian Homrighausen, Kasper Høgh, Mikael Møller Høgsgaard, Zhile Jiang, Lior Kamma, Kasper Green Larsen, Alexander Mathiasen, Konstantinos Mampentzidis, Manaswi Paraashar, Rasmus Killmann Brogaard Petersen, Nidhi Rathi, Mathias Rav, Martin Ritzert, Aniket Basu Roy, Casper Moldrup Rysgaard, Chris Schwiegelshohn, Jens Kristian Refsgaard Schou, Sudarshan Shyam, Srikanth Srinivasan, Svend Christian Svendsen, Rolf Svenning, Mads Bech Toftrup, as well as our wonderful group coordinators Maja Malmdorf Andersen, Helena Luna Bach, Signe Jensen, Dorthe Haagen Nielsen, Julie Tølbøl Rasmussen. It is great working with you.

I would like to extend my sincere thanks to Marc van Kreveld and Frank Staals for hosting me during my visit to Utrecht University. I appreciate the warm welcome and many great discussion sessions. I also want to thank my wonderful officemates Erwin Glazenburg, Sarita de Berg, and Thijs van der Horst during my visit. I am also grateful to my hosts, Timothy M. Chan and Sariel Har-Peled, in UIUC, and the PhD students in the theory group for their hospitality during my visit. Specifically, I would like to thank my coauthors, Qizheng He and David Zheng, for showing me around and the fun time we had working on many interesting problems.

Special thanks go to the Vejle squad members Amik Raj Behera and Nidhi Rathi. I am so happy to have Amik and Nidhi with me on this journey, and I will always remember the midnight table tennis sessions, the movies we watched, and the sunkissed days. I also want to say a special thank you to my previous officemate Rasmus Killmann Brogaard Petersen for introducing me to Danish culture and many

interesting discussions we had for both research and life.

Finally, I want to thank Wenyue Ma, Dalena Nguyen, Irfansha Shaik, and Rachel Zhang for being very nice friends, and I enjoy the time I spent with you in Aarhus. Especially, I would like to thank Dalena for helping me proofread the Danish version of the abstract.

Pingan Cheng,
Aarhus, July 31, 2023.

Contents

Part I

Overview

Chapter 1

Introduction

One of the central themes in computer science is how to store data in computers in an organized and structural way so that we can retrieve desired information from it efficiently. For example, suppose we are given the geographical data of a region, how should we store this information in a computer system so that for any query coordinate, we can list all the restaurants within three kilometers efficiently? Or how to design a database for a library to support queries like counting the number of computer science books that have been borrowed more than 10 times in the last six months? These problems, known as *range searching*, have the same underlying pattern: the input can be modeled as a set of points in $\mathbb{R}^d$, and the queries can be formulated as geometric shapes from a certain family, e.g., disks and rectangles in the two examples, respectively. In the first example, we want to list, or *report*, all the points intersecting the query range, while in the second example, we want to *count* the number of such points. These two problems are called *range reporting* and *range counting* respectively, and they are the two most common types of range searching.

Besides range reporting and range counting, there are also many other types of range searching problems, e.g., *range emptiness*, where we want to determine if a query range contains an input, *range max/min*, where each input is associated with a weight and we want to find the maximum/minimum weight contained in a query range, and so on. Given that there are so many problem instances of range searching, it would be convenient to abstract the common features of these problems in a uniform model. To arrive at this model, we need the help of semigroups. We briefly recap the definition of semigroups, but a complete introduction to this and other algebraic structures is beyond the scope of this dissertation. See, e.g., [Lan93] for an introduction.

Recall that a semigroup $(\mathscr{G}, \circ)$ is a collection $\mathscr{G}$ of elements (which we call weights) in which an associative binary operator $\circ$ is defined. Note that, however, we do not require the existence of an identity element or inverses in a semigroup. Let $\mathscr{P}$ be a set of input points in $\mathbb{R}^d$. We can model different range searching problems with different semigroups. For example, for range reporting, we choose $(\mathscr{G}, \circ) = (2^{\mathscr{P}}, \cup)$, i.e., the semigroup consists of the power set of $\mathscr{P}$, and the semigroup operator is the set union operator. To model the problem, we assign semigroup weight $w(p) = \{p\}$, i.e., a singleton, to each point $p \in \mathscr{P}$, and to answer a query γ, we take the total weight of all points in γ, i.e., $\sum_{p\in\gamma\cap\mathscr{P}} w(p) = \cup_{p\in\gamma\cap\mathscr{P}}\{p\}$. Similarly, to model range counting, we use semigroup $(\mathscr{G},\circ) = (\mathbb{N},+)$, and assign weight 1 to each point $p \in \mathscr{P}$.

This leads to the following formal definition of (general) range searching.

Definition 1.0.1 (Γ-range Searching [Mat94, Aga16])**.** *In a Γ-range Searching problem, we are given a set $\mathscr{P}$ of points in $\mathbb{R}^d$ where each point $p \in \mathscr{P}$ is assigned a weight $w(p)$ from a semigroup $(\mathscr{G},\circ)$, and the goal is to preprocess $\mathscr{P}$ into a data structure such that for any query $\gamma \in \Gamma$, the total weight of points in $\gamma \cap \mathscr{P}$, i.e., $\sum_{p\in\gamma\cap\mathscr{P}} w(p)$, can be computed efficiently.*

In this dissertation, range searching will be the central topic of discussion. It is an important subfield of computational geometry, the study of computational aspects of geometric problems. We focus on the lower and upper bounds of data structures that solve range searching problems. Given a data structure built for an input of size n, we use $S(n)$ and $Q(n)$ to denote its space consumption and query time. For range reporting data structures, however, we denote its query cost by $Q(n) + f(k)$, where the first part $Q(n)$ captures the cost of finding the output and the second part $f(k) \geq k$ models the cost of actually producing the output of size k. Note that reporting queries are usually easier compared to counting queries since the total query time can sometimes be charged to the cost of listing the output. This observation, made by Chazelle [Cha86], suggests designing different data structures depending on different output sizes, and this idea has been used to simplify the solutions to many problems.

The classical model of computation used in computational geometry is the *real*RAM model. In this model, we have registers and memory cells that are able to store real numbers of infinite precision. Each basic arithmetic and relational operation between reals and each memory access takes one unit amount of time; however, we disallow conversion from reals to integers. In light of the definition of range searching with semigroups, each memory cell can store a value from the semigroup, and we only allow the semigroup operation among values stored. (The operation takes again one unit amount of time.) Note that this assumption is unrealistic for reporting problems. Recall that the underlying semigroup is the power set of the input set, which means we can store any subset of the input points using one memory cell.

There are also some computational models used in specific scenarios. One of the models we will use extensively is the pointer machine model, which provides good modeling of pointer-based data structures. Pointer machines have a long history and many variants, and we will discuss this model in much more detail in Chapter 2, but informally speaking, it can be viewed as the *real*RAM model without random access, and the only way to access a memory cell is by a series of pointer navigations. In this dissertation, we will use a variant of pointer machines by Chazelle [Cha90a], which has been used to prove lower bounds for many range reporting problems, e.g., range reporting with axis-parallel boxes [Cha90a] or with simplices [CR96]. The semigroup arithmetic model is another commonly used computational model which is more powerful than the two models mentioned before. This model is often used to study the general range searching problem where the values are from any "faithful" underlying semigroup. In this model, the data structure is modeled as a collection of partial sums, and the cost to answer a query is modeled as the number of partial sums needed to answer the query under the semigroup operation. Note that since inverses are not guaranteed in semigroups, the partial sums used cannot be generated from values not belonging to the query. There are also lower bounds proved in this model, e.g., range searching with orthogonal boxes [Cha90b, Afs19] or polytopes [Cha89].

Now we give an overview of the main problems in range searching.

1.1 Orthogonal Range Searching

The query type studied earliest is probably axis-aligned boxes. An axis-aligned box or an orthogonal range in $\mathbb{R}^d$ is defined to be the Cartesian product of d intervals, each belonging to one of the d dimensions. On the one hand, this type of query has a strong connection to database systems as a database query can easily be expressed as a Cartesian product of multiple interval queries along individual dimensions. On the other hand, as we will see later, axis-aligned boxes have many nice properties, and therefore are easier to handle than other types of more general queries.

The classic way of solving orthogonal range searching is by dimensionality reduction. The idea is simple. We first build a balanced binary search tree for the coordinates in the first dimension. Then by searching a query using its first coordinate, we can determine at most $2\log n$ (canonical) subsets of points whose first coordinates are within the range. Then we solve the problem for each subset recursively in a space whose dimension is one smaller. This leads to the data structure of range trees, first described by Bently [Ben79], of size $O(n\log^{d-1} n)$ and query time $O(\log^d n)$. It is possible to slightly improve this bound for the reporting [Wil85, Cha86, AAL10, AAL12] and the counting [Cha88, CW16] variants. There are also lower bounds [Cha90a, Cha90b, Afs19] in the pointer machine and the arithmetic semigroup model respectively.

Most of the techniques used to study orthogonal range searching cannot be generalized to queries like simplices as they rely on the orthogonality of the query boxes. But nevertheless, there are some exceptions. The first important idea is the construction of "multilevel data structures". In orthogonal range searching, we build a search tree structure for each level, and this increases the space and query time by an $O(\log n)$ factor at each level. For more general queries, this is a powerful tool to reduce queries to their simplest forms, e.g., a simplicial query can be reduced to the intersection of three halfspaces. Another important concept is the lower bound techniques. The lower bound frameworks in the pointer machine [Cha90a] and the arithmetic [Cha90b] model Bernard Chazelle developed are quite general and actually work for all types of range searching problems.

1.2 Simplex and Semialgebraic Range Searching

Simplex Range Searching

Simplex range searching is probably one of the most well-studied problems in range searching. Compared to orthogonal boxes, simplices have more of a geometric favor, and indeed geometry plays a major role in the solution of simplex range searching.

In the later 1980s and early 1990s, this line of research was extensively investigated. This interest of researchers in simplex range searching is well-justified. As its name suggests, simplices are "simple" geometric objects which can be formed by simply intersecting $d+1$ halfspaces in $\mathbb{R}^d$. But behind the mask, they are powerful as well. In many applications of computational geometry and computer graphics, we approximate geometric shapes by polytopes. Since we can easily decompose polytopes into simplices, an efficient solution for simplex range searching will also imply an efficient solution for polytope queries.

Despite the innocent look, the problem of simplex range searching turns out to be rather hard. It took about a decade of intensive research by many great minds in this field to develop the tools, the cutting theorem [Cha93] and the partition theorem [Mat92a, Mat93], to solve this problem. In 1993, the first (almost) optimal data structure for this problem was finally discovered by Matoušek by combining these two tools, and he concluded a query time-space tradeoff of $S(n)Q(n)^d = \tilde{O}(n^d)$ [1] [Mat93] for the problem, which matches the earlier lower bound (up to a polylogarithmic factor) discovered by Chazelle [Cha89] in the semigroup arithmetic model. If we require the space usage to be linear, the query time is $O(n^{1-1/d})$. For higher dimensions, this essentially means that we can only do better than the trivial solution of scanning all the points marginally by a tiny $n^{1/d}$ factor. If we want queries to be answered in polylogarithmic time, then the space usage has to be rather large, namely $O(n^d)$. In light of the lower bound given by Chazelle [Cha89], there is not much room for improvement, even in the powerful semigroup arithmetic mode. This is in sharp contrast with orthogonal range searching we saw in the last section, where in near-linear space, the best data structure can answer orthogonal range searching queries in polylogarithmic time by a textbook solution of range trees [dBCvKO08].

This phenomenon is even observed in halfspace range searching, the special case of simplex range searching. A general lower bound by Arya et al. [AMX12] showed that for some semigroups, $S(n)Q(n)^d = \tilde{\Omega}(n^d)$ holds even if the queries are halfspaces. However, for the reporting variant, it is possible to break this barrier based on the "shallow" versions of the partition and the cutting theorem [Mat92b]. For example, linear-sized halfspace range reporting structures with $O(\log n + k)$ query time exist in $\mathbb{R}^2$ and $\mathbb{R}^3$ [AC09]. In general, halfspace range reporting can be solved in time $O(n^{1-1/\lfloor d/2 \rfloor})$ with linear space in $\mathbb{R}^d$ [Cha12].

However, such improvements for simplex range reporting is impossible. The most recent lower bound prior to our work by Afshani [Afs13] showed a query time lower bound of $\Omega(n^{1-1/d}/2^{\sqrt{\log n}} + k)$ for linear space simplex range reporting data

[1]In this dissertation, we use notations $\tilde{O}(\cdot), \tilde{\Theta}(\cdot), \tilde{\Omega}(\cdot)$ to hide $\log^{O(1)} n$ factors.

structures. Observe, however, that there is a quite big and annoying $2^{\sqrt{\log n}}$ factor gap between the lower and upper bound. As the first result of this dissertation, we will eliminate this gap in Chapter 2.

Semialgebraic Range Searching

As we have seen in the previous section, simplex range searching was well-understood about 30 years ago. However, the generalization of this problem, known as semialgebraic range searching, remained mysterious until very recently.

To understand this generalization, let us view simplices from a more algebraic perspective. In the language of algebra, simplices are defined by the intersection of halfspaces (subsets of $\mathbb{R}^d$) defined by $d+1$ linear inequalities. The expressive power of linear inequalities is somewhat restrictive. For example, given a disk shape described by a degree-2 polynomial inequality, there is no way to decompose it into finitely many simplices. Motivated by this limitation, we study a more general type of query known as semialgebraic sets (of constant descriptive complexity), and this gives rise to semialgebraic range searching. By a semialgebraic set (of constant descriptive complexity), we mean the intersection/union/complement of the subsets of $\mathbb{R}^d$ defined by s d-variate degree-Δ polynomial inequalities for constants s, d, Δ. A complete introduction to semialgebraic sets, polynomials, and other algebraic geometry concepts are beyond the scope of this dissertation. We refer the readers to textbooks of algebraic geometry, e.g., [CLO15] for more details.

At first glance, it might seem that semialgebraic range searching should be a much more difficult problem than its simplicial counterpart. But very surprisingly, if we restrict the space usage to be close to linear, we can solve semialgebraic range searching in roughly the same $\tilde{O}(n^{1-1/d})$ time. This was first proved for lower dimensions of $d \leq 4$ by Agarwal and Matoušek [AM94] using the tools developed for simplex range searching like the partition and cutting theorems in the early 1990s. However, to fully address the problem, we need a much more powerful generalization of these tools for semialgebraic sets. This job was undertaken by Guth and Katz [GK15] as a part of the polynomial revolution starting around 2010. Then Agarwal, Matoušek, and Sharir [AMS13] finally proved the general result for all dimensions. This surprising result spurred researchers to work on the other end of the spectrum: if we want $\log^{O(1)} n$ query time, what is the smallest possible space usage? Is $O(n^d)$ sufficient like simplex range searching? It will be very surprising if it is the case as it means that simplex and semialgebraic range searching are computationally equivalent, even though the latter problem looks much more intimidating. This is also the open problem asked explicitly at the end of their seminal paper [AMS13]. But given the result for the small-space version, this is also not inconceivable.

The second result of this dissertation answers this question. In Chapter 2, we show that semialgebraic range searching acts completely differently in the fast-query regime compared to the small-space regime. The space usage of fast-query data structures for semialgebraic range searching has to be $\Omega(n^{\beta-o(1)})$, where β is the number of parameters needed to determine the query polynomial inequality. In general, for a

d-variate degree-Δ polynomial, β can be as large as $\binom{\Delta+d}{d} - 1$. In comparison, current best fast-query data structures for simplex range searching only use $O(n^d)$ space.

1.3 Beyond Classic Range Searching

Besides these classic range searching problems, there are many variants of range searching, generalizing the problem from different perspectives. In this section, we introduce some interesting variants that we will focus on in this dissertation.

Intersection Searching

A direct generalization of range searching is intersection searching. In this problem, we allow input objects (resp. query objects) to have higher (resp. lower) dimensions than points (resp. full-dimensional objects), e.g., the input and queries can be a set of line segments, triangles, simplices, and so on. To be more specific, we now want to preprocess a set of t-dimensional objects so that given a $(d-t)$-dimensional query object, we can find the input objects intersecting the query object efficiently for $0 \le t \le d$. For example, given a collection of triangles in $\mathbb{R}^3$, we want to preprocess the triangles such that for any query ray, we can output the first triangle intersecting the query ray efficiently. This problem, known as ray shooting, is a very important problem in both computational geometry and computer graphics with many applications in the real world like video games and the movie industry.

For a long time, the standard approach to solve the problem has been to reduce it to semialgebraic range searching. However, this requires us to transform the objects to a higher dimensional parametric space, and thus the solution does not seem to be optimal. For example, to describe a line in 3D, a parametric representation (a_1z+b_2, a_2z+b_2, z) requires four parameters a_1, a_2, b_1, b_2 and a free variable z. This reduces the problem to a semialgebraic range searching problem in 4D, and by the known result [AM94], we get an upper bound of $S(n)Q(n)^4 = O(n^{4+\varepsilon})$[2]. A major improvement to this solution was found very recently by Ezra and Sharir [ES22b] who showed a better solution of $O(n^{3/2+\varepsilon})$ space and $O(n^{1/2+\varepsilon})$ query time using polynomial techniques. Since this solution lies on the tradeoff curve of $S(n)Q(n)^3 = O(n^{3+\varepsilon})$, an interesting open problem here is if we can actually show this tradeoff for the problem for all possible combinations of space and time. It is conceivable as the problem lies in $\mathbb{R}^3$ instead of $\mathbb{R}^4$. However, as we will see in Chapter 3, our third result of this dissertation suggests that such a generalization is not possible.

Summary Queries

A more recent generalization of range searching is range summary queries. As the input size grows in many real-world scenarios in the modern big data era, many drawbacks of the classic range searching solutions have gradually emerged. For one

[2]In this dissertation, ε is an arbitrarily small positive constant unless stated explicitly otherwise.

thing, although a range reporting structure finds all the data for a query of interest, answering such queries can be time-consuming when the output size is gigantic, and it is often unnecessary to output all the data in the query in the first place. For the other, range counting, median, mean, mode, and other query types for simple statistics contain too little information to be useful for data analysis. To this end, it is desirable to consider more useful "data summaries" of the output.

One useful summary is heavy hitters, a generalization of modes. Given a set S of elements, each associated with a color from a color set C, an ε-heavy hitter summary for $0 < \varepsilon < 1$ is a subset of colors $C' \subset C$ where the frequency of each color $c \in C'$ is at least $\varepsilon|C|$. We will be interested in the query version of ε-heavy hitters, namely, given a parameter ε, how to preprocess a set of colored points in $\mathbb{R}^d$ into a structure such that we can output an ε-heavy hitter summary of $P \cap \gamma$ for a query range γ efficiently?

Another summary type that has been studied quite frequently is quantiles, a generalization of medians. Given a set S of elements, each associated with a weight from $\mathbb{R}$, an ε-quantile summary for $0 < \varepsilon < 1$ is a sequence $s_0, \cdots, s_{1/\varepsilon}$ of $1/\varepsilon + 1$ elements from S where the rank of s_i is $i\varepsilon|S|$ in the sequence of S sorted by their weights. Similar to before, we study the query version of ε-quantiles, namely, given a parameter ε, how to preprocess a set of weighted points in $\mathbb{R}^d$ into a structure such that we can output an ε-quantile summary of $P \cap \gamma$ for a query range γ efficiently?

In general, it is computationally inefficient to compute ε-heavy hitter and ε-quantile summaries exactly; so instead the approximate versions are considered in the literature. In Chapter 4, as the fourth result in this dissertation, we will see efficient data structures supporting approximate range summary queries.

Iterative Search

Before we describe what iterative search is, it is helpful to look at range searching from a "dual" perspective. In the original problem, the input consists of 0-dimensional points and the queries are d-dimensional ranges. By switching the roles of the input and the query (a collection of d-dimensional ranges as the input and points as queries), we get the dual range stabbing problem.

Now we start the description of iterative search with a motivating example. Given t lists, each consisting of n numbers, we want to preprocess the numbers such that for any query number q, we can find the successors of q in all of these t lists. Note that this is range stabbing in 1D, as each list can be viewed as a collection of intervals in $\mathbb{R}$. A trivial solution to this problem takes $O(tn)$ space and $O(t \log n)$ query time by building the optimal successor searching structure, e.g., a balanced binary search tree, for each list and then query iteratively.

In 1986, Chazelle and Guibas [CG86a] invented a technique known as fractional cascading that can solve iterative search much more efficiently. The statement of their main theorem is in fact very general: if given an undirected graph where each vertex has a constant degree and is associated with a list of numbers, suppose the total number of numbers is n, then we can build a structure of $O(n)$ space such that for any query subgraph π and any query number q, we can find all the $|\pi|$ successors in time $O(\log n + |\pi|)$, turning the multiplicative dependence in the path length in the simple solution described in the previous section to an additive one. Note that this essentially means that after the initial $O(\log n)$ investment, we can find all the remaining successors in $O(1)$ time, and this is clearly optimal. This technique is very general and has many applications in computational geometry [CG86b].

However, generalizing fractional cascading to higher dimensions, i.e., each vertex is associated with a collection of higher dimensional geometric objects, seems to be difficult, and it has been shown to be impossible in 2D in the general setting [CL04]. Interestingly, as we will see in Chapter 5, as the last result described in this dissertation, we show that it is possible to circumvent the lower bound for the important special case of axis-parallel planar subdivisions and significantly outperform the trivial solution.

1.4 Roadmap of the Dissertation

This dissertation consists of two parts.

The first five chapters belong to the first part. These chapters provide an overview of the problems studied with an introduction to the background, a summary of the main results, and highlights of the ideas and techniques used. In Chapter 2, we focus on simplex and semialgebraic range searching, the two most classic range searching problems. We review the standard upper and lower bound tools and present new results for the two problems. In Chapter 3, we switch to the first generalization of range searching, intersection searching. We review the reduction between intersection searching and semialgebraic range searching, and give a timely lower bound which complements the current upper bound development of this problem. In Chapter 4, we consider a generalization arising recently in this big data era, range summary queries. We review the common summaries and recent attempts to answer range summary queries. We conclude this chapter with (near) optimal solutions to two very popular summary types. Finally in Chapter 5, we consider iterative search in which we need to solve multiple dual problems of range searching iteratively. We review the classic result in 1D, justify the hardness of generalizing it to higher dimensions in the general setting, and show a series of new results for the important special case of 2D orthogonal subdivisions where improvements are possible.

The second part consists of the corresponding six publications during my PhD studies. The contents in these chapters are identical to the papers.

1. 2D Generalization of Fractional Cascading on Axis-aligned Planar Subdivisions, in *FOCS* 2020 [AC20].

2. Lower Bounds for Semialgebraic Range Seaching and Related Problems, in *SoCG* 2021 (*JACM* version [AC23b]).

3. On Semialgebraic Range Reporting, in *SoCG* 2022 [AC22].

4. An Optimal Lower Bound for Simplex Range Reporting, in *SOSA* 2023 [AC23c].

5. Lower Bounds for Intersection Searching among Flat Objects, in *SoCG* 2023 [AC23a].
6. On Range Summary Queries, in *ICALP* 2023 [ACBRW23].

Chapter 2

Simplex and Semialgebraic Range Seaching

In this chapter, we focus on simplex and semialgebraic range searching. We start our journey with an exposition of the classic tools used to solve the two problems. We first describe the partition and cutting theorems and their polynomial versions as the upper bound techniques and then explore the lower bound techniques tailored for range reporting problems.

2.1 Upper Bound Techniques

In the literature, simplex and semialgebraic range searching have been investigated from two opposite directions, one aiming at solving the problem with near linear space, and the other with polylogarithmic query time. These two directions give rise to so-called "*small-space*" data structures based on the partition theorem and "*fast-query*" data structures based on the cutting theorem. A space-time tradeoff can be obtained by interpolating the two solutions. We start with "small-space" data structures.

Small-space Data Structures and the Partition Theorem

Early Developments Small-space data structures for both simplex and semialgebraic range searching are both based on the concept of partition trees (built from the partition theorem). The partitioning technique used for semialgebraic queries can be viewed as a generalization of that for simplex range searching.

The concept of partition trees dates back to Willard's early attempt [Wil82] to solve simplex range searching in 1982. The underlying idea in [Wil82] is simple and elegant. By the famous ham-sandwich theorem, see, e.g., [Mat99] for an introduction, given a set $\mathscr{P}$ of n points in $\mathbb{R}^2$, we can partition the plane into four disjoint regions using two lines l_1, l_2 such that each region contains $n/4$ points. The key observation is that for any query halfspace γ, one of the four regions is either fully contained in or fully outside of γ, and so we only need to solve three of the four subproblems. See Figure 2.1 for an example.

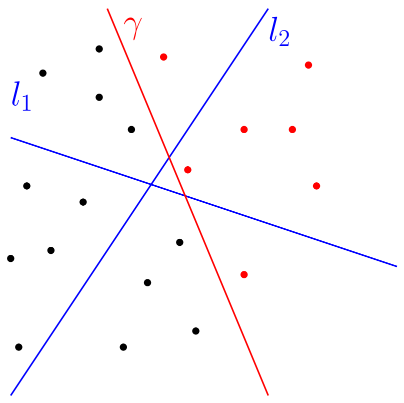


Figure 2.1: A Partition Based on the Ham-sandwich Theorem: line l_1 partitions the point sets into two subsets with equal sizes, and by the ham-sandwich theorem, we can find a line l_2 that partitions each of the two subsets further into two equally-sized subsets, which creates four equally-sized subsets in total. Line γ intersects three of these four subsets.

Thus, if the query is only a halfplane, this suggests that we build a tree structure where each node implicitly represents a subset of $\mathbb{R}^2$. For example, the root is the entire plane $\mathbb{R}^2$, and it has four children, each representing one of the four subregions formed by the two partitioning lines. The root node itself stores (the coordinates of) the two partitioning lines as well as the total semigroup weight of the points in the region it represents. This structure is built recursively until the number of points in a region corresponding to a node drops below some constant. Such trees are known as *partition trees*.

To answer a halfplane query using this structure, at each node, we only recurse to three out of the four children which are determined by the partitioning lines stored in the node. We therefore obtain the following recurrence relation for the query time

$$Q(n) = 3Q\left(\frac{n}{4}\right) + O(1),$$

which solves to $Q(n) = O(n^{\log_4 3})$. On the other hand, the space usage satisfies

$$S(n) = 4S\left(\frac{n}{4}\right) + O(1),$$

which solves to $S(n) = O(n)$. (We assume that it takes constant space to store the weight of each region.) Note that a simplex in $\mathbb{R}^2$ is the intersection of three halfplanes, so we can use this tree structure to answer simplex queries: at each node, we recurse to all the subregions intersected by the boundary of any of the halfplanes. Since the total number of nodes visited will be at most three times of those visited by each individual halfplane, the total query time is asymptotically the same.

This idea, although it does not lead to the optimal solution, introduces a very important strategy: partition the point set into subsets of roughly equal sizes so that any query range has a small crossing number, i.e., the number of subsets intersecting the boundary of the range is small, and then recursively use this strategy to build a *partition tree* to support simplicial queries.

The Classical Partition Theorem for Simplex Queries Following the work of Willard [Wil82], after a series of work attempting to improve the crossing number or generalize the partition scheme to higher dimensions [Yao83, EW86, HW87, YDEP89, CW89, Wel92, CSW92], a partition theorem with the optimal crossing number was finally discovered by Matoušek [Mat92a, Mat93] in the early 1990s.

Theorem 2.1.1 ([Mat92a, Mat93]). *Given a set $\mathscr{P}$ of n points in $\mathbb{R}^d$ for $d \geq 2$. For any integral parameter s with $2 \leq s < n$, there is a set Π of simplicices such that $s \leq |\Delta \cap \mathscr{P}| \leq 2s$ for any $\Delta \in \Pi$ and any hyperplane intersects $O(r^{1-1/d})$ simplicies, where $r = n/s$.*

To get an intuition of Theorem 2.1.1, let us consider the following example. In this example, we study an input set of n points placed uniformly at random inside a unit cube in $\mathbb{R}^d$.

Example 2.1.1. *Consider a d-dimensional hypercube $\mathscr{U} = \prod_{i=1}^{d}[0,1]$. We divide $\mathscr{U}$ into a uniform grid $\mathscr{G}$ of size $r^{1/d} \times \cdots \times r^{1/d}$. Now we sample n points in $\mathscr{U}$ uniformly at random, then in expectation, each cell of $\mathscr{G}$ has n/r points. Furthermore, any hyperplane γ intersects $O(r^{1-1/d})$ cells in $\mathscr{G}$. See Figure 2.2 for an example.*

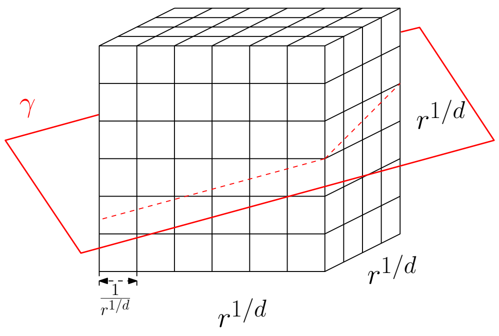


Figure 2.2: Partition Theorem for a Uniform Random Point Set.

Applying Theorem 2.1.1, Matoušek showed that we can build a more efficient partition tree for simplex range searching which achieves $O(n)$ space and $O(n^{1-1/d})$ query time. This has been simplified and improved (in terms of the preprocessing time) by Chan recently [Cha12]. To sum up, using linear space, simplex range searching can be solved in time $O(n^{1-1/d})$.

Polynomial Partitioning for Semialgebraic Queries The generalization of Theorem 2.1.1 was first explored by Guth and Katz [GK15] during the polynomial revolution beginning in around 2010. This technique is so powerful that it has been successfully applied to address many long-standing discrete geometry problems, e.g., the distinct distance problem in the paper where they introduced this technique [GK15]. However, Guth and Katz did not attempt to bound the crossing number of this partition strategy. Later, Agarwal, Matoušek, and Sharir [AMS13] further developed this technique to (almost) fully address semialgebraic range searching. The new partition theorem is listed below, where the crossing number of the partition is due to [AMS13].

Theorem 2.1.2 ([GK15] and [AMS13]). *For any $r > 1$ and a set $\mathscr{P}$ of n points in $\mathbb{R}^d$, there is a polynomial P of degree $O(r^{1/d})$ such that $Z(P)$ partitions $\mathbb{R}^d$ into r open cells such that every cell contains at most n/r points of $\mathscr{P}$. This polynomial P is called an r-partitioning polynomial. The zero set of any constant degree polynomial intersects at most $O(r^{1-1/d})$ cells in the partition.*

To better understand this technique, we give a proof of Theorem 2.1.2 in $\mathbb{R}^2$. This proof is adapted from [KMS12]. We use the polynomial ham-sandwich theorem as a black box.

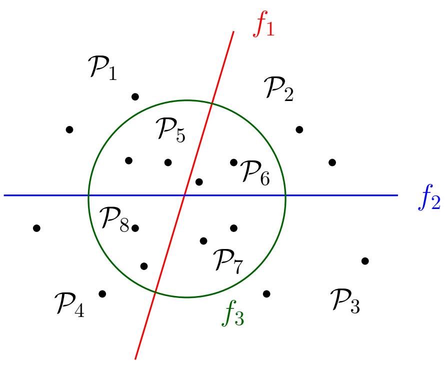
Theorem 2.1.3 (Polynomial Hams-sandwich Theorem). *Let $\mathscr{P}_1, \cdots, \mathscr{P}_r$ be a collection of r sets of points in $\mathbb{R}^2$, and let Δ be a positive integer with $\binom{\Delta+2}{2} - 1 \geq r$. There is a bivariate polynomial f of degree at most Δ that bisects $\mathscr{P}_1, \cdots, \mathscr{P}_r$ simultaneously.*

Proof of Theorem 2.1.2 in $\mathbb{R}^2$. We iteratively apply Theorem 2.1.3 to partition point sets. In the first step, we bisect $\mathscr{P}$ into two sets $\mathscr{P}_1, \mathscr{P}_2$ using a degree 1 polynomial f_1 by Theorem 2.1.3. After i iterations, we have a collection of 2^i sets, and to bisect them into 2^{i+1} sets, we need a polynomial f_i of degree no more than $\sqrt{2^{i+1}}$ by Theorem 2.1.3. We stop until we obtain a collection of r sets. It is clear that $P = \prod_{i=1}^{\log r} f_i$ is an r-partitioning polynomial and its degree is

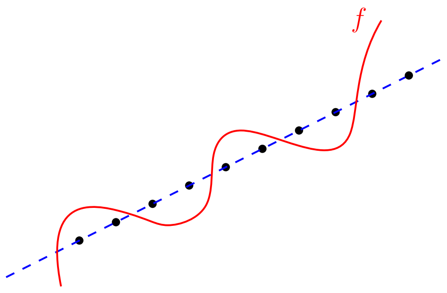
$$\sum_{i=1}^{\log r} \deg(f_i) = \sum_{i=1}^{\log r} \sqrt{2^{i+1}} = O(\sqrt{r}).$$

The crossing number part of the theorem follows from Bézout theorem. See Figure 2.3a for an example. □

An efficient algorithm for computing the partitioning polynomial in Theorem 2.1.2 was given by [AMS13]. They also addressed the problem of degeneracy, i.e., many points lie on the zero set of the partitioning polynomial. Note that this problem cannot be easily eliminated. For example, a set of points can all lie on the zero set of a line and to partition this point set into r sets, a polynomial of degree roughly r instead of $O(r^{1/d})$ is required by the fundamental theorem of algebra. See Figure 2.3b for an example. This problem is addressed by Agarwal, Matoušek, and Sharir [AMS13] by projecting these points to a lower dimensional space and recursively constructing partitioning polynomials in the new space. A simplified and improved approach was



(a) Partitioning a Point Set by Polynomials



(b) Degeneracy

Figure 2.3: Polynomial Partitioning.

later proposed by Matoušek and Patáková [MP15]. To sum up, they showed that with close to linear space, semialgebraic range searching can be solved in time $\tilde{O}(n^{1-1/d})$.

Fast-query Data Structures and the Cutting Theorem

The Dual Transformation The fast-query version of the problem can be viewed as the dual of the small-space variant and indeed the problem is solved in the dual space.

In $\mathbb{R}^2$, the point-line duality establishes a transformation between a point and a hyperplane $(a_1, \cdots, a_d) \leftrightarrow x_d + a_d = \sum_{i=1}^{d-1} a_i x_i$. This transform is incidence-preserving and order-preserving, i.e., given a point p and a hyperplane H in the primal space and their duals $\bar{p}$ and $\bar{H}$ in the dual space

- $p \in H$ if and only if $\bar{H} \in \bar{p}$;
- p is above H if and only if $\bar{H}$ is above $\bar{p}$.

See Figure 2.4 for an example in $\mathbb{R}^2$.

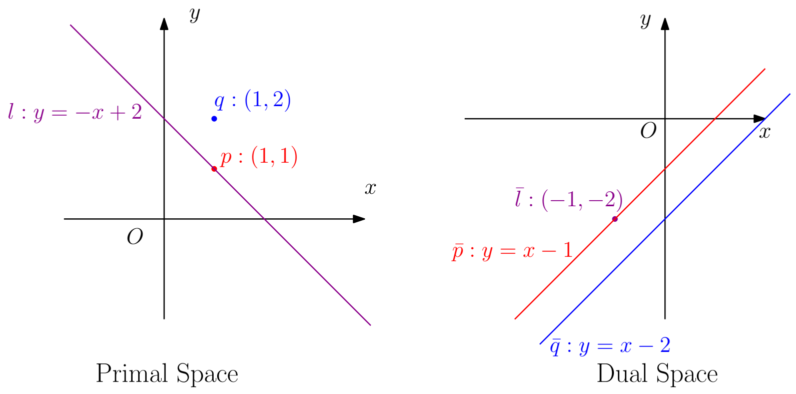


Figure 2.4: Point-line Duality.

The Classical Cutting Theorem for Simplex Range Searching As in the small-space case, we first focus on halfspace range searching. By applying duality, the problem can be formulated as follows in the dual space: given a set of n hyperplanes in $\mathbb{R}^d$, preprocess the hyperplanes into a structure such that for any query point q, the hyperplanes below q can be found efficiently. Intuitively, the arrangement formed by n hyperplanes has complexity $O(n^d)$. To answer a query, it suffices to locate which cell in the arrangement the query point is in. Thus the problem essentially reduces to a point location problem. How to efficiently answer point location queries among the arrangement of hyperplanes leads to the development of another important tool: $(1/r)$-*cuttings*.

Definition 2.1.1 ($(1/r)$-cuttings)**.** *Let H be a collection of n hyperplanes in $\mathbb{R}^d$ and r be a parameter with $0 < r < n$. A $(1/r)$-cutting for H is a collection of (possibly unbounded) disjoint simplices $\mathscr{C}$ that together cover $\mathbb{R}^d$ and the interior of any simplex in $\mathscr{C}$ intersects at most n/r hyperplanes in H. The collection of hyperplanes intersected by each simplex is called the conflict list of the simplex.*

The development of cuttings dates back to the late 1980s when Klarkson [Cla87] and Haussler and Welz [HW87] applied random sampling to computational geometry. After a series of works in the computational geometry community [Aga90, Aga91, Mat90, Mat91, Mat95, CF90], the optimal cutting and its construction was given by Chazelle [Cha93].

Theorem 2.1.4 (Chazelle [Cha93])**.** *Given a set H of n hyperplanes in $\mathbb{R}^d$, a $(1/r)$-cutting of size $O(r^d)$ exists. The cutting as well as the conflict list for each cell in the cutting can be constructed in time $O(nr^{d-1})$.*

To get an intuition of Theorem 2.1.4, we consider the average case. Consider a set H of n hyperplanes and we know that there are $\Theta(n^d)$ d-wise intersections among hyperplanes in H. For a cutting of size $\Theta(r^d)$, on average, each cell of the cutting should contain $\Theta((n/r)^d)$ intersections, which means the number of hyperplanes intersecting each cell should be $\Theta(n/r)$.

Using Theorem 2.1.4, we can build a *cutting tree* where each node represents a subset of $\mathbb{R}^d$. Let r be a constant. We store a $(1/r)$-cutting for each node as well as the total weights above the region the node represents. Each node has $O(r^d)$ children, each representing a cell in the cutting. The root node represents the entire $\mathbb{R}^d$ and we recursively build the tree until the number of hyperplanes intersecting the region represented by a node drops below some constant. The query time of this structure is clearly $O(\log n)$. The space of this structure satisfies

$$S(n) = cr^d S\left(\frac{n}{r}\right) + cr^d,$$

for some constant c, which solves to $O(n^{d+\varepsilon})$.

Extending the result from halfspace range searching to simplex range searching is less obvious in the fast-query setting compared to the small-space setting. This extension relies on the concept of multilevel data structures. Very recently, Chan and Zheng [CZ23] showed that it is possible to improve the space usage to $O(n^d)$.

Generalizing the Cutting Lemma to Semialgebraic Sets The fast-query version of semialgebraic range searching is again solved by dualization. Here the dual space is the parametric space of the queries. Consider a query $P(x_1, \cdots, x_d, a_1, \cdots, a_t) \leq 0$ specified by t independent parameters $a_1, \cdots, a_t$, we can turn the query into a point in $(a_1, \cdots, a_t) \in \mathbb{R}^t$ and, alternatively, each input point $(x_1, \cdots, x_d)$ into a polynomial inequality $P(x_1, \cdots, x_d, a_1, \cdots, a_t) \leq 0$ specified by d parameters $x_1, \cdots, x_d$.

In the dual parametric space, semialgebraic range searching is transformed to point location among the arrangement formed by semialgebraic sets. Similar to fast-query data structures for simplex range searching, the solution to this problem is based on the (generalized version of) the Cutting Theorem 2.1.4. In 2015, Guth [Gut15] generalized Theorem 2.1.2 and introduced the concept of generalized partitioning polynomials. This can be viewed as a generalization of the Cutting Theorem 2.1.4.

Theorem 2.1.5 ([Gut15]). *Given a set $\mathscr{S}$ of n k-dimensional semialgebraic sets in $\mathbb{R}^d$, for any positive integer r, there is a polynomial of degree at most r such that each connected component of $\mathbb{R}^d \setminus Z(P)$ intersects $O(n/r^{d-k})$ sets of $\mathscr{S}$.*

Note that by setting $k = d - 1$ and considering hyperplanes, Theorem 2.1.5 yields the bound of Theorem 2.1.4. An efficient algorithm for computing generalized partitioning polynomials for a collection of semialgebraic sets was given by Agarwal et al. recently [AAEZ21]. Essentially, this gives an $O(n^{t+\varepsilon})$ space, polylogarithmic query time data structure for semialgebraic range searching.

We mention in passing that an earlier approach to solving semialgebraic range searching is *linearization*, introduced by Yao and Yao [YY85]. Simply put, this approach reduces a semialgebraic range searching problem in $\mathbb{R}^d$ to a simplex range searching problem in $\mathbb{R}^L$, where $L > d$ is called the linearization dimension. For example, by mapping input points (x, y) to $(x, y, x^2 + y^2)$ and each query polynomial inequality $(x - a)^2 + (y - b)^2 - r^2 \leq 0$ to a linear inequality $z - 2ax - 2by + a^2 + b^2 - r^2 \leq 0$, we reduce 2D circular range searching to 3D halfspace range searching. This gives us a fast-query data structure of $\tilde{O}(n^L)$ space that solves semialgebraic range searching with linearization dimension L. In general, L can be bigger than t, the number of actual parameters needed to define a polynomial inequality, so the approach based on general polynomial partitioning [AAEZ21] usually yields better bounds.

Space-time Tradeoff

It is also possible to combine the small-space and the fast-query solutions to get a space-time tradeoff straightforwardly: build the partition tree for some level and then switch to the fast query structure. This gives us a tradeoff of $S(n)Q(n)^{(t-1)d/(d-1)} = O(n^{t+\varepsilon})$ for semialgebraic range searching where each polynomial inequality is defined by at most d variables and t parameters. For simplex range searching in $\mathbb{R}^d$, the tradeoff simplifies to $S(n)Q(n)^d = \tilde{O}(n^d)$.

2.2 Lower Bound Techniques

In this dissertation, we focus on range reporting lower bounds in the pointer machine model. We first introduce the model, and then present a lower bound framework in this model tailored for range reporting problems.

The Pointer Machine Model

Pointer-based computational models emerged very early in the literature. The first such model was probably the *Kolmogorov-Uspenskii machine* [KU58] in 1958. Later Knuth introduced a similar model known as the *linking automaton* [Knu97], and Schönhage [Sch79] independently devised the *storage modification machine*, which he showed is capable of simulating Turing machines with multidimensional tapes in real-time. In 1979, Tarjan [Tar79] augmented Knuth's *linking automaton* and defined the *pointer machine* model on which he showed lower bounds for maintaining disjoint sets. The main restrictions placed on the pointer machine model are that memory access can only be done by pointer navigations and no pointer computation is possible. However, the data types stored and arithmetic operations allowed depend on the problems to study. In this sense, the *pointer machine* model is more of a class of computational models. For example, Chazelle [Cha88] defined the *semi-arithmetic pointer machine* and the *arithmetic pointer machine* to study multidimensional searching.

The pointer machine model we will use in this dissertation is an augmented version by Chazelle [Cha90a]. In this version, the memory is modeled by a directed graph $G = (V, E)$. Each node $v \in V$ stores (a reference to) one of the input elements as well as two pointers to other nodes in G. Among the nodes in V, there is a special root node r. During the execution of an algorithm, an explored subgraph S is maintained. Initially, $S = \{r\}$, i.e., only the root is included. Then at each step, the algorithm can choose to add one vertex $v \in G \setminus S$ to S as long as there is a node $u \in S$ pointing to v. The algorithm terminates when every output element appears in S at least once. This model is granted unlimited computational power and full knowledge of G, which means that the algorithm can always choose the optimal strategy to explore the smallest subgraph S containing the output elements. Note that for proving lower bounds, we can strengthen the power of a model as we want as long as we can still prove meaningful lower bounds. However, the restriction that new nodes can only be accessed by pointer navigation still applies. Essentially, in such a variant, the space usage of a data structure will be the size of G, and the query time of any algorithm will be the size of the smallest subgraph S needed to explore to answer the query.

A Lower Bound Framework for Geometric Range Reporting

We now introduce a lower bound framework in the pointer machine model for geometric range reporting problems formulated by Chazelle [Cha90a] and later Chazelle and Rosenberg [CR96]. The intuition behind the framework is as follows. On the one hand, if we want to answer queries efficiently, the points corresponding to queries need to be stored close to each other in the directed graph G. This is because the output is generated by pointer navigations and the size of the subgraph explored is the query time, and so we want the explored subgraph to be as small as possible (but at least contains the output). On the other hand, suppose there exists a collection of queries such that their outputs share very few common points, then this means the subgraphs explored to answer these queries should also share few nodes in common. As a result, we need to store the outputs to queries relatively separately. (We can overlap some queries to save a bit of space but asymptotically the space usage is the same since there are few nodes shared by queries.) So if the output size of each query is big and we have many such queries, the space usage will be large.

We now present the framework formally.

Theorem 2.2.1 (Chazelle [Cha90a], Chazelle and Rosenberg [CR96])**.** *For a range reporting problem, assume that we can find a collection of m sets $\mathscr{S}_1, \cdots, \mathscr{S}_m$ of points, each being the output of some query of the range reporting problem, such that the following two conditions are satisfied:*

1. *The size of each point set is at least $Q(n)$, i.e., $\forall i \in 1, 2, \cdots, m, |\mathscr{S}_i| \geq Q(n)$.*
2. *The intersection of any $\alpha \geq 2$ distinct point sets is upper bounded by β, i.e., $\forall i_1, \cdots, i_\alpha \subseteq \{1, 2, \cdots, m\}, |\mathscr{S}_{i_1} \cap \cdots \mathscr{S}_{i_\alpha}| < \beta$ for two parameters α, β .*

Then there is data structure lower bound of $S(n) = \Omega\left(\frac{mQ(n)}{\alpha 2^{O(\beta)}}\right)$ for the problem.

It is possible to directly apply this framework to show lower bounds. We will later describe a direct application of this framework to prove a tight lower bound for linear-sized simplex range reporting data structures. But in most cases, it is difficult to give a construction of point sets satisfying the two conditions in the framework. Instead, this is typically done by a probabilistic argument as follows. Let $\mathbb{R}^d$ be the space where the range reporting problem is defined. We fix some region in $\mathbb{R}^d$, usually a unit hypercube, but any region with bounded volume suffices. Then we create m geometric *query ranges* instead of *m point sets* such that

- Each range intersects the region with large volume.
- The intersection of each α ranges intersect with very small volume.

The exact values of the two volumes depend on the application, but they are chosen such that if we sample n points uniformly at random in the region, then with positive probability, each region contains at least $Q(n)$ points and the intersection of every α ranges contains less than β points. We mention that as a consequence, by applying this probabilistic argument, we will inevitably lose some factors in the lower bound. We capture this in a more streamlined version of the framework.

Theorem 2.2.2. *For a range reporting problem, assume that we can find m ranges $\mathscr{R}_1, \cdots, \mathscr{R}_m$ and a unit cube $U \subset \mathbb{R}^d$ such that*

1. *The intersection volume of each range and U is at least $Q(n)/n$.*

2. *The intersection volume of any two distinct ranges is $O(1/f(n))$ for some function $f(n) \geq n2^{\sqrt{\log n}}$.*

Then the range reporting problem has a data structure space lower bound of $S(n) = \overset{o}{\Omega}(mQ(n))$[1].

We remark that the $2^{\sqrt{\log n}} = n^{o(1)}$ factor is used in the probabilistic argument to show that a worst-case construction exists with positive probability. This is also the reason why we lose an $n^{o(1)}$ in our lower bound.

Lower Bounds for Simplex and Semialgebraic Range Reporting

Prior to our results which we will introduce later in this dissertation, there are two important lower bounds for simplex range reporting, one by Chazelle and Rosenberg [CR96] and the other by Afshani [Afs13]. For any data structure of query time $Q(n) + O(k)$, the former showed a lower bound of $S(n)Q(n)^d = \Omega(n^{d-\varepsilon})$ while the latter improved it to $S(n)Q(n)^d = \Omega(n/2^{O(\sqrt{\log Q(n)})})$. Since the current upper bound is $S(n)Q(n)^d = \tilde{O}(n^d)$, there is a gap between the upper and lower bounds. Especially, when the size of the structure is linear, the lower bound is a super polylogarithmic $2^{\sqrt{\log n}}$ factor away from the upper bound.

2.3 An Overview of Our Results

We present our new lower bounds for simplex and semialgebraic range reporting in the pointer machine model.

Lower Bounds for Simplex Range Reporting

For simplex range reporting, we show the first tight lower bound for linear-sized simplex range reporting structures, improving the previous result of Afshani [Afs13] by a $2^{\sqrt{\log n}}$ factor, and it matches the upper bound by Matoušek [Mat93] and Chan [Cha12]. This result was published in SOSA 2023 [AC23c].

Theorem 2.3.1 ([AC23c]). *Any linear-space pointer machine data structure that solves simplex range reporting must use query time $\Omega(n^{1-1/d} + k)$, where k is the output size.*

More generally, we prove the following lower bound for the tradeoff between space and query time.

[1]In this dissertation, we use $\overset{o}{O}(\cdot), \overset{o}{\Theta}(\cdot), \overset{o}{\Omega}(\cdot)$ notations to hide $n^{o(1)}$ factors.

Theorem 2.3.2 ([AC23c]). *Any pointer machine data structure that solves simplex range reporting with space $S(n)$ must use query time $\Omega((n^2/S(n))^{(d-1)/d}+k)$, where k is the output size.*

So for any simplex range reporting data structures that use almost linear space, i.e., $S(n)=O(n\log^{O(1)}n)$, we improve the previous best lower bound by Afshani [Afs13].

Lower Bounds for Semialgebraic Range Reporting

For the more general semialgebraic range reporting problems, we start with the special case when the query ranges are polynomial slabs in 2D. A polynomial slab is defined to be the region between curves $y=P(x)$ and $y=P(x)+w$ for a univariate polynomial P of degree Δ and some parameter w. More formally speaking,

Definition 2.3.1. *A 2D polynomial slab, denoted by (P,w), is the region between the graphs of $y=P(x)$ and $y=P(x)+w$ for some univariate polynomial $P(x)$ and some parameter w, i.e., $(P(x),w)=\{(x,y)\in\mathbb{R}^2 : P(x)\le y\le P(x)+w\}$.*

We first describe the following lower bound result for this special case. This result was published in SoCG 2021 [AC23b].

Theorem 2.3.3 ([AC23b]). *The space usage $S(n)$ and query time $Q(n)+O(k)$ of any data structure that is able to solve 2D polynomial slab range reporting defined by polynomials of form $y=P(x)$ where P is a univariate polynomial of degree Δ must satisfy $S(n)=\Omega\left(\frac{n^{\Delta+1-o(1)}}{Q(n)^{(\Delta+3)\Delta/2}}\right)$.*

Note that by linearization, we can solve the problem with $\tilde{O}(n^{\Delta+1})$ space and polylogarithmic query time. So our lower bound is tight up to an $n^{o(1)}$ factor. The significance of Theorem 2.3.3 is that it shows semialgebraic range reporting is inherently more difficult than simplex range reporting in the fast-query case. This is in strong contrast with the small-space case where the two problems admit solutions of approximately the same query complexity.

However, there are several weaknesses of Theorem 2.3.3. Taking bivariate polynomials of degree Δ as an example, the space upper bound for fast-query data structures is $\tilde{O}(n^{\binom{\Delta+2}{2}-1})$, since the number of parameters needed to specify a general bivariate polynomial can be as large as $\binom{\Delta+2}{2}-1$. But the exponent in Theorem 2.3.3 is only $\Delta+1$. Furthermore, Theorem 2.3.3 is restricted to bivariate polynomials and multivariate polynomials are not considered.

To overcome this weakness, we generalize polynomial slabs as follows. Let P be a d-variate polynomial. We define a (generalized) polynomial slab defined by P to be $\{X\in\mathbb{R}^d : 0\le P(X)\le w\}$. As our second main result for semialgebraic range reporting lower bounds, we show the following. This result was published in SoCG 2022 [AC22].

Theorem 2.3.4 ([AC22])**.** *The space usage $S(n)$ and query time $Q(n)+O(k)$ of any data structure that is able to solve polynomial slab range reporting defined by d-variate degree-Δ polynomials must satisfy $S(n)=\Omega\left(\frac{n^{\beta-o(1)}}{Q(n)^{\Theta(\beta)}}\right)$, where $\beta=\binom{d+\Delta}{d}-1$ is the maximum number of coefficients needed to define a query polynomial.*

Note that our lower bound almost matches the current upper bound by linearization or the result by Agarwal et al. [AAEZ21].

2.4 Highlights of Ideas and Techniques

The main tool we use to prove the aforementioned lower bounds is the lower bound framework described in Theorem 2.2.1. We first give a short and simple proof of Theorem 2.3.2 in 2D to demonstrate the idea.

Lower Bound for Simplex Range Reporting

Assume that $Q(n)$, $\frac{n}{Q(n)}$, and $\frac{n}{2Q(n)^2}$ are integers for simplicity. Let $\mathscr{G}$ be an integer grid of size $\frac{n}{Q(n)}\times Q(n)$. To use Theorem 2.2.1, we need to show the existence of a collection of point sets satisfying Condition 1 and 2. We put $\mathscr{G}$ in a Cartesian system and generate these point sets by intersecting $\mathscr{G}$ with several hyperplanes, i.e., lines in $\mathbb{R}^2$. Observe that hyperplanes are degenerated simplices, so any lower bound we prove for hyperplanes also holds for simplices. Specifically, we consider lines of form

$$y=ax+b,$$

where $a=1,2,\cdots,\frac{n}{2Q(n)^2}$ and $b=1,2,\cdots,\frac{n}{2Q(n)}$. See Figure 2.5 for an example.

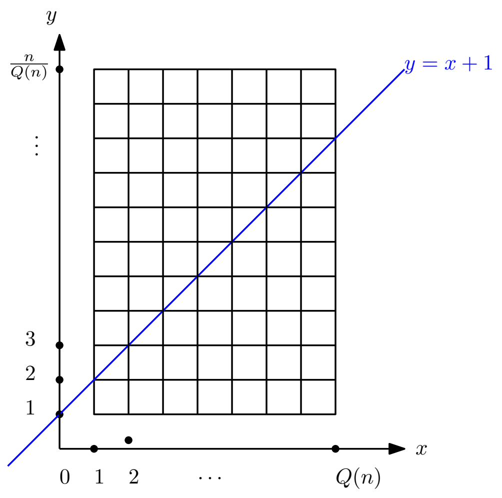


Figure 2.5: A Tight Simplex Range Reporting Lower Bound Construction in 2D

We make two observations.

1. Any line intersects exactly $Q(n)$ points.
2. Let A, B be two set of points intersected by two distinct lines, then $|A \cap B| \leq 1$.

The first observation follows from the fact that we are considering an integer grid and all the coefficients of the lines are integers as well; furthermore, the maximum y value of the intersection of any line and $\mathscr{G}$ is at most $n/Q(n)$. The second observation is a simple fact that distinct lines intersect at most at one point. The total number of lines we generated is $n^2/(4Q(n)^3)$, which is also the number of point sets we generated. By applying Theorem 2.2.1, we obtain a lower bound of

$$S(n) = \Omega\left(\frac{n^2}{Q(n)^3} \cdot Q(n)\right) \implies S(n) = \Omega\left(\frac{n^2}{Q(n)^2}\right),$$

which is a tight lower bound for simplex range reporting in $\mathbb{R}^2$.

This simple proof contains many important ideas that can be generalized to prove a lower bound in higher dimensions. First, by considering hyperplanes as degenerated simplices, we can generate point sets using an integer grid. Or even more generally, this boils down to a point-hyperplane incidence problem in incidence geometry. Second, we need to bound the points in the intersection of hyperplanes. This is trivial in $\mathbb{R}^2$, but it requires more work in higher dimensions since in a higher-dimensional space, the intersection could be a lower-dimensional hyperplane which could contain many points. Addressing this problem is one of the main challenges in this work. The high-level idea is to consider more hyperplanes and prove an upper bound of the number of hyperplanes needed so that their intersection is a single point. We omit the technical details but refer the readers to Chapter 6 for more details.

Lower Bounds for Semialgebraic Range Reporting

For the purpose of demonstration, we will show a slightly weaker version of Theorem 2.3.3. Our aim is to show the following lower bound.

Theorem 2.4.1. *The space usage $S(n)$ and query time $Q(n) + O(k)$ of any data structure that is able to solve 2D polynomial slab range reporting defined by polynomials of form $y = P(x)$ where P is a univeraite polynomial of degree Δ must satisfy $S(n) = \Omega\left(\frac{n^{\Delta+1-o(1)}}{Q(n)^{(\phi+1)(\Delta+1)-1}}\right)$ where $\phi = \binom{\Delta+1}{2}$.*

We point out that for technical reasons, it is difficult to directly give a construction (of point sets) for Theorem 2.2.1. What we will do instead is to use an encapsulation of Theorem 2.2.1, i.e., Theorem 2.2.2, to show a lower bound. The main advantage of Theorem 2.2.2 is that we now only need to construct geometric ranges with certain guarantees of their volumes which are easier to analyze.

The first step of our construction is to generate polynomials of form

$$P(x) = \sum_{i=0}^{\Delta} a_i x^i,$$

where $a_i = \tau, 2\tau, 3\tau, \cdots, \frac{1}{2(\Delta+1)}$ for $\tau = \frac{Q(n)^{\phi+1}2^{\phi\sqrt{\log n}}}{n}$ and $\phi = \binom{\Delta+1}{2}$. Based on these polynomials, we construct polynomial slabs $(P(x), w)$ and set $w = Q(n)/n$. In this construction, the total number of polynomial slabs we constructed is

$$\Theta\left(\left(\frac{1/(2(\Delta+1))}{\tau}\right)^{\Delta+1}\right) = \overset{o}{\Theta}\left(\frac{n^{\Delta+1}}{Q(n)^{(\phi+1)(\Delta+1)}}\right).$$

To use Theorem 2.2.2, we need to show that Condition 1 and 2 are satisfied. For Condition 1, first observe that for any polynomial P in our construction and any $x \in [0,1]$, we have

$$0 \le P(x) = \sum_{i=0}^{\Delta} a_i x^i \le \frac{1}{2}.$$

This means every polynomial slab is fully contained in $\mathscr{U}$ in x-interval $[0,1]$ with volume

$$\int_0^1 w\mathrm{d}x = w = \frac{Q(n)}{n},$$

where the last equality follows from $w = \frac{Q(n)}{n}$. This is exactly what we need for Condition 1.

For Condition 2, first observe that two slabs $(P_1(x), w)$ and $(P_2(x), w)$ intersect iff $|P_1(x) - P_2(x)| \le w$. We use the following Lemma (See Chapter 8 for a proof).

Lemma 2.4.1. *Let $P_1(x) = \sum_{i=0}^{\Delta} a_i x^i$ and $P_2(x) = \sum_{i=0}^{\Delta} b_i x^i$ and there is an i such that $|a_i - b_i| \ge \tau$. If $|P_1(x) - P_2(x)| \le w$ for all $x \in I$, then $|I| = O((w/\tau)^{1/\phi})$ where $\phi = \binom{\Delta+1}{2}$.*

Observe that in our construction, for any two polynomials $P_1(x), P_2(x)$, there must be an i such that the coefficients of x^i of them differ by at least τ. By Lemma 2.4.1, the corresponding slabs $(P_1(x), w)$ and $(P_2(x), w)$ must intersection with an x-interval of length $O((w/\tau)^{1/\phi}) = O(1/(Q(n)2^{\sqrt{\log n}}))$. Then their intersection has volume

$$\int_I w\mathrm{d}x = O\left(w\frac{1}{Q(n)2^{\sqrt{\log n}}}\right) = O\left(\frac{1}{n2^{\sqrt{\log n}}}\right).$$

This satisfies Condition 2. Then by Theorem 2.2.2, we obtain a space lower bound of

$$S(n) = \overset{o}{\Omega}(mQ(n)) = \overset{o}{\Omega}\left(\frac{n^{\Delta+1}}{Q(n)^{(\phi+1)(\Delta+1)-1}}\right).$$

Note that when $Q(n) = \log^{O(1)} n$, this already gives an almost tight space lower bound of $n^{\Delta+1-o(1)}$. We remark that the exponent in the denominator can be improved to $(\Delta+3)\Delta/2$ by a more refined analysis (See Chapter 7) and it has been shown to be tight for a uniform random input set (See Chapter 8). We can generalize the lower bound to polynomials with d variates of degree Δ with $\binom{\Delta+d}{d} - 1$ coefficients, which is the maximum possible number of coefficients. This almost matches the current upper bound by linearization or the result by Agarwal et al. [AAEZ21].

However, this generalization is nontrivial, and there are many issues we need to address. First of all, the key intuition behind our construction is that if two polynomials have one sufficiently different coefficient, then the two polynomials cannot stay close for too long (Lemma 2.4.1), and thus the corresponding polynomial slabs have a small intersection area. One technical point is that polynomials of form $y - P(x)$ are irreducible and this is a necessary condition for the construction to work. To see this, two reducible polynomials can share common factors even if their coefficients are arbitrarily different. This is undesirable since sharing common factors means their zero sets overlap, and thus if we generate polynomial slabs, the intersection will be infinitely large. Even if a general bivariate polynomial is irreducible, its zero set can have a complicated structure, and it is not straightforward to define a slab around it.

To overcome these issues, we restrict our attention to polynomials of form

$$P(X) = X_1 - X_2^\Delta + \sum_{\mathbf{i}} A_i X^i,$$

where indeterminates $X = (X_1, \cdots, X_d)$ and coefficients $A_i = (a_1, \cdots, a_d) = o(1)$ and degree $\mathbf{i} = (i_1, \cdots, i_d)$ with $0 \le \sum_{j=1}^{d} i_j \le \Delta$. The nice property of polynomials of this form is that they are very close to $P(X) = X_1 - X_2^\Delta$ since all other coefficients $A_i = o(1)$, which makes the analysis much simpler. However, we will still be able to generate $\binom{\Delta+2}{2} - 1$ different polynomials, which is sufficient for us to get a lower bound matching the known upper bound.

We next describe the high-level ideas behind the lower bound proof for general bivariate polynomials of form

$$P(X_1, X_2) = X_1 - X_2^\Delta + \sum_{i_1, i_2} A_{i_1 i_2} X^{i_1 i_2},$$

where $A_{i_1, i_2} = o(1)$. The key ingredient is a generalization of Lemma 2.4.1. The intuition behind Lemma 2.4.2 is as follows. Given two bivariate polynomials P_1, P_2 of this form, for any value of $X_2 \in [1,2]$, we define the X_1-distance between the zero sets $Z(P_1)$ and $Z(P_2)$, denoted by $\pi(Z(P_1), Z(P_2), X_2)$. This is well-defined as these polynomials behave similarly to $X_1 - X_2^\Delta$ in $[1,2]$. Then we generalize Lemma 2.4.1 by bounding the interval length of X_2 in which $\pi(Z(P_1), Z(P_2), X_2) \le O(w)$ as a function that depends on the gap between the coefficients of P_1, P_2.

Lemma 2.4.2. *Let $P_1(X_1, X_2) = X_1 - X_2^\Delta + \sum_{i_1, i_2} A_{i_1 i_2} X^{i_1 i_2}$ and $P_2(X_1, X_2) = X_1 - X_2^\Delta + \sum_{i_1, i_2} B_{i_1 i_2} X^{i_1 i_2}$ where $A_{i_1 i_2}, B_{i_1 i_2} = o(1)$ and $0 \le i_1 + i_2 \le \Delta$ and there is a pair (i_1, i_2) such that $|A_{i_1 i_2} - B_{i_1 i_2}| \ge \tau$. If $I = \{X_2 \in [1,2]| : \pi(Z(P_1), Z(P_2), X_2)| \le O(w)\}$, then $|I| = O((w/\tau)^{1/\mathbf{B}})$ where $\mathbf{B} = \binom{\Delta+2}{2} - 1$ and function $\pi(Z(P_1), Z(P_2), v)$ denotes the distance between the zero sets of P_1, P_2 at $X_2 = v$ along the X_1 axis.*

Observe that Lemma 2.4.2 resembles Lemma 2.4.1 in several ways. However, we remark that the proof of this generalized lemma relies heavily on the specific form of P_1, P_2 and is not generally true for arbitrary polynomials.

If we construct polynomials with gap τ in their coefficients in a similar manner as we did previously in proving Theorem 2.4.1, using this lemma, we will be able to bound the intersection area of each pair of polynomial slabs. This will give us the following lower bound result for 2D semialgebraic range searching.

Theorem 2.4.2. *The space usage $S(n)$ and query time $Q(n)+O(k)$ of any data structure that is able to solve bivariate degree-Δ polynomial range reporting satisfies $S(n)=\Omega\left(\frac{n^{\mathbf{B}}}{Q(n)^{\Theta(1)}}\right)$, where $\mathbf{B}=\binom{\Delta+2}{2}-1$ and the hidden constants depend on Δ.*

Now to generalize this lower bound to higher dimensions, we use a "slicing" idea. Let us consider trivariate polynomials for an example. Suppose we have two trivariate polynomials

$$P(X_1,X_2,X_3)=X_1-X_2^{\Delta}+\sum_{i_1,i_2,i_3} A_{i_1i_2i_3}X_1^{i_1}X_2^{i_2}X_3^{i_3},$$

and

$$Q(X_1,X_2,X_3)=X_1-X_2^{\Delta}+\sum_{i_1,i_2,i_3} B_{i_1i_2i_3}X_1^{i_1}X_2^{i_2}X_3^{i_3},$$

where $A_{i_1i_2i_3}=B_{i_1i_2i_3}=o(1)$. The idea is to treat the last indeterminant X_3 as a part of the coefficients, i.e., we can rewrite the polynomials as

$$P(X_1,X_2,X_3)=X_1-X_2^{\Delta}+\sum_{i_1,i_2} f_{i_1i_2}(X_3)X_1^{i_1}X_2^{i_2},$$

and

$$Q(X_1,X_2,X_3)=X_1-X_2^{\Delta}+\sum_{i_1,i_2} g_{i_1i_2}(X_3)X_1^{i_1}X_2^{i_2},$$

where

$$f_{i_1i_2}(X_3)=\sum_{i_3=0}^{\Delta-i_1-i_2} A_{i_1i_2i_3}X_3^{i_3}$$

and

$$g_{i_1i_2}(X_3)=\sum_{i_3=0}^{\Delta-i_1-i_2} B_{i_1i_2i_3}X_3^{i_3}.$$

To see why this helps us, note that to use the results from bivariate polynomials, we only need to make sure that there is a pair of coefficients with gap τ, i.e.,

$$\exists i_1,i_2 : |f_{i_1i_2}(X_3)-g_{i_1i_2}(X_3)|\geq\tau.$$

Alternatively, we want to bound the length of the interval in X_3 such that

$$\forall i_1,i_2 : |f_{i_1i_2}(X_3)-g_{i_1i_2}(X_3)|<\tau.$$

Observe that $f_{i_1i_2}(X_3)$ and $g_{i_1i_2}(X_3)$ are both univariate polynomials, and thus we can bound the interval length of X_3 in which this happens using Lemma 2.4.1. We make sure that the gap between $A_{i_1i_2i_3}$ and $B_{i_1i_2i_3}$ is big enough such that this "bad" interval length for X_3 is negligible. Thus, we can create polynomial slabs for trivariate polynomials respecting this new gap. The rest will be carried out by the base bivariate polynomials. A careful reader may have noticed that as long as we have a solution for d-variate polynomials, we can slice over the last dimension of a $(d+1)$-variate polynomials, and thus get a lower bound for $(d+1)$-variate polynomials.

We end this chapter with a remark that although this approach shows an almost tight lower bound of $\overset{o}{\Omega}(n^{\binom{\Delta+d}{d}-1})$ for polynomial slab reporting, it is an interesting open problem to show such lower bounds for polynomials of more general forms, i.e., polynomials not restricted to $X_1 - X_2^\Delta + \sum_{i_1,i_2} A_{i_1 i_2} X^{i_1 i_2}$ for $A_{i_1 i_2} = o(1)$. The full details for semialgebraic lower bounds are presented in Chapter 7 and Chapter 8.

Chapter 3

Intersection Searching

In the problem of intersection searching, we are given a collection of geometric objects of dimension t for $0 \le t \le d$ in $\mathbb{R}^d$ as the input, and we want to preprocess the input into a data structure so that given any query geometric object of dimension $d-t$, we can find the input objects intersecting the query efficiently. This is an important problem in computational geometry, and it has found applications in many other related fields. For example, when the input consists of a collection of triangles in $\mathbb{R}^3$ and queries are rays, we have ray shooting among triangles in 3D, one of the central problems in computer graphics. When both the inputs and queries are polygons, it models collision detection, an important problem in computer graphics and robotics.

From a theoretical perspective, the problem can be viewed as a generalization of the classic range searching problem we encountered in Chapter 2, where the inputs are points (thus zero-dimensional objects) and the queries are geometric ranges of dimension d. In intersection searching, the dimensions of the input and query objects are more flexible.

The classic way of solving intersection searching is by a reduction to semialgebraic range searching. There are many ways to do the reduction based on the specific problems we have on hand. We give a relatively simple reduction, but as we will see, this reduction will eventually help us get a lower bound.

3.1 A Reduction to Semialgebraic Range Searching

The high-level idea of this approach is quite simple. We first parametrize the input objects and then express the query object as a semialgebraic set in the parametric space to get a semialgebraic range searching problem. Next, we apply the known semialgebraic range searching results [AM94, AMS13, AAEZ21] to solve the problem. In particular, the classic result for ray shooting among triangles in 3D of $S(n)Q(n)^4 = O(n^{4+\varepsilon})$ was obtained by this approach [AM94]. Intuitively speaking, the exponent four in the tradeoff is a result of the fact that we need four parameters to define a line in 3D.

We demonstrate this approach with an example of line-hyperslab intersection reporting in 3D. A hyperslab is obtained by shifting a hyperplane in 3D by a fixed distance along some direction. We first parameterize the objects.

Representing Lines and Hyperslabs

We will consider lines and hyperslabs that are non-degenerate. Specifically, no query line or any line defining hyperslabs is parallel to any of the axes.

First, recall that a line in 3D can be parametrized by four parameters. For example, one way to represent a line is to parametrize the points on it by $(\tau, a_{1,1}\tau + a_{1,2}, a_{2,1}\tau + a_{2,2})$, where τ is a free variable and $a_{i,j}$'s are parameters. This is equivalent to a matrix-vector multiplication of form

$$\begin{bmatrix} 1 & 0 \\ a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \tau \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.1}$$

Note that any non-degenerated line can be represented this way. For technical reasons, we need to introduce an extra parameter $a_{0,1}$. So a line is represented as

$$\begin{bmatrix} a_{0,1} & 0 \\ a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \tau \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

The key point here is that even though we have five parameters, there are only four of them that are independent.

To represent a hyperslab, we consider a line as the intersection of two hyperplanes in 3D and then translate the line to get a hyperslab. This can be formulated as a linear system

$$\begin{bmatrix} 1 & b_{1,1} & b_{1,2} \\ 1 & b_{2,1} & b_{2,2} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ -1+w \end{bmatrix},$$

where $b_{i,j}$ are the parameters of the two hyperplanes defining the line and $w \in [0, w_0]$ is a parameter that describes the translation for some positive value w_0.

A Reduction from Line-Hyperslab Intersection Reporting to Semialgebraic Range Reporting

In this section, we show a reduction from line-hyperslab intersection reporting to semialgebraic range reporting. First, observe that if a line intersects a hyperslab, and they have the aforementioned representations, then the free variable τ defining the line must satisfy

$$\begin{bmatrix} 1 & b_{1,1} & b_{1,2} \\ 1 & b_{2,1} & b_{2,2} \end{bmatrix} \cdot \begin{bmatrix} a_{0,1} & 0 \\ a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \cdot \begin{bmatrix} \tau \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1+w \end{bmatrix},$$

for some $w \in [0, w_0]$. Multiplying the two matrices, we get

$$\begin{bmatrix} a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{1,i} & \sum_{i=1}^{2} a_{i,2} b_{1,i} \\ a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{2,i} & \sum_{i=1}^{2} a_{i,2} b_{2,i} \end{bmatrix} \cdot \begin{bmatrix} \tau \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 + w \end{bmatrix}.$$

Let us call this linear system $A\tau = s$. We assume $\det(A) \neq 0$, which is the case when the line and the hyperslab "properly" intersect, i.e., the line is not contained in the slab, then by Cramer's rule, we get

$$\frac{\begin{vmatrix} a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{1,i} & 0 \\ a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{2,i} & -1 + w \end{vmatrix}}{\begin{vmatrix} a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{1,i} & \sum_{i=1}^{2} a_{i,2} b_{1,i} \\ a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{2,i} & \sum_{i=1}^{2} a_{i,2} b_{2,i} \end{vmatrix}} = 1.$$

By linearity of determinants, we can rearrange this equation to get

$$\begin{vmatrix} a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{1,i} & \sum_{i=1}^{2} a_{i,2} b_{1,i} \\ a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{2,i} & \sum_{i=1}^{2} a_{i,2} b_{2,i} + 1 - w \end{vmatrix} = 0.$$

For brevity of description, let $\mathbf{a} = (a_{0,1}, a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2})$ and $\mathbf{b} = (b_{1,1}, b_{1,2}, b_{2,1}, b_{2,2})$, and by expanding the determinant, we get

$$P(\mathbf{a}, \mathbf{b}) + f(\mathbf{a}, \mathbf{b}, w) = 0,$$

where

$$P(\mathbf{a}, \mathbf{b}) = (a_{1,1}a_{2,2} - a_{2,1}a_{1,2})(b_{1,1}b_{2,2} - b_{1,2}b_{2,1}) + \sum_{i=1}^{2} (a_{i,1} - a_{0,1}a_{i,2}) b_{1,i} + \sum_{i=1}^{2} a_{0,1} a_{i,2} b_{2,i} + a_{0,1}$$

and

$$f(\mathbf{a}, \mathbf{b}, w) = -(a_{0,1} + \sum_{i=1}^{2} a_{i,1} b_{1,i}) w.$$

If we view $\mathbf{a}$ as (fixed) coefficients, then $P(\mathbf{a}, \mathbf{b})$ is a multilinear polynomial in $\mathbf{b}$. For a fixed vector $\mathbf{a}$, we define polynomial slab $(P(\mathbf{a}, \mathbf{b}), f(w_0)) = \{\mathbf{b} \in \mathbb{R}^4 : 0 \leq P(\mathbf{a}, \mathbf{b}) \leq -f(\mathbf{a}, \mathbf{b}, w_0)\}$. For brevity, we use $f(w_0)$ to denote $f(\mathbf{a}, \mathbf{b}, w_0)$ when the context is clear.

We make the following observations.

Observation 3.1.1. *A line parameterized by* $\mathbf{a}$ *intersects a hyperslab parameterized by* $\mathbf{b}$ *of width* w_0 *if and only if the point* $\mathbf{b}$ *is contained in the polynomial slab* $(P(\mathbf{a}, \mathbf{b}), f(w_0))$*, for* $P(\mathbf{a}, \mathbf{b}), f(w_0)$ *defined as above.*

Note that it is an if and only if statement, meaning, we not only get a reduction from line-hyperslab intersection reporting to semialgebraic range reporting, but we also get a reduction from (a special type of) semialgebraic range reporting to line-hyperslab intersection reporting. This latter point is important for us to show a lower bound as we will explain in more detail later. But for now, given that we have a reduction from a line-hyperslab intersection reporting problem in 3D to a semialgebraic range reporting problem in 4D, we obtain the following upper bound by applying the known semialgebraic range searching results [AMS13, MP15, AAEZ21].

Theorem 3.1.1. *Line-hyperslab intersection reporting in 3D can be solved with a space time tradeoff of $S(n)Q(n)^4 = O(n^{4+\varepsilon})$.*

3.2 Intersection Searching by Polynomial Partitioning

The ray shooting reporting result based on a reduction to semialgebraic range reporting has remained unchallenged for more than two decades until very recently. Ezra and Sharir [ES22b] observed that using the general polynomial partitioning technique developed by [Gut15, AAEZ21], more specifically, polynomial cutting in Theorem 2.1.5 we described in Chapter 2, it is possible to construct a cutting for (the edge of) triangles directly which enables us to solve the problem using divide-and-conquer in 3D without lifting the problem to 4D. They showed that by choosing parameters carefully, it is possible to achieve an $S(n) = O(n^{3/2+\varepsilon})$ space and $Q(n) = O(n^{1/2+\varepsilon})$ query time data structure for ray-shooting among triangles in 3D.

The importance of this result is that it is the first work that significantly improves the classic result of $S(n)Q(n)^4 = O(n^{4+\varepsilon})$. In fact, their result lies on the space-time tradeoff curve of $S(n)Q(n)^3 = O(n^{3+\varepsilon})$. This is rather surprising as lines and triangles are essentially four-dimensional objects. However, it seems quite hard to apply their approach to get comparable improvements for other combinations of space and query time. (For their divide-and-conquer to work, they need to pick specific parameters to get correct bounds for subproblems so that the recurrence relation yields the desired bound.) By interpolating this new result and the old classic result, we get a new (and rather unnatural) space-time tradeoff of

$$\max\{S(n)Q(n)^2 = O(n^{5/2+\varepsilon}), S(n)Q(n)^5 = O(n^{4+\varepsilon})\}.$$

This is a rather strange-looking bound and does not seem like the correct answer. Ezra and Sharir [ES22b] also asked at the end of their paper if it is possible to obtain a smooth tradeoff curve of $S(n)Q(n)^3 = O(n^{3+\varepsilon})$. Unfortunately, this was unknown prior to our work. But on the positive side, this reveals the possibility of improving known results for many other intersection searching problems. For example, intersection searching for (semialgebraic) arcs and plates in 3D [AAE+22], line segments and tetrahedrons in 4D [ES22a], and triangles and triangles in 4D [ES22a].

3.3 An Overview of Our Results

Before describing our results, let us first formally define the problem we study. We follow the convention that a t-flat is an affine subspace of dimension t. We consider the following t-flat intersection reporting problem.

Definition 3.3.1. *In the t-flat intersection reporting problem, we are given a collection of $(d-t)$-hyperslabs as the input, which are in essence shapes formed by translating a $(d-t-1)$ dimensional flat along some direction, and we are to preprocess these hyperslabs into a data structure such that given any query t-flat, we can find the $(d-t)$-flats that intersect the query efficiently.*

We show two results, one for 1-flat intersection reporting, or line-hyperslab intersection reporting in $\mathbb{R}^d$, and the other for 2-flat intersection reporting in $\mathbb{R}^4$, or triangle-triangle intersection reporting in $\mathbb{R}^4$. These results were published in SoCG 2023 [AC23a].

For line-hyperslab intersection reporting, we show the following lower bound.

Theorem 3.3.1 ([AC23a])**.** *Any pointer machine data structure that solves line-hyperslab intersection reporting in $\mathbb{R}^d$ in time $Q(n)+O(k)$, where n is the input size and k is the output size, must use space $\overset{o}{\Omega}(n^{2(d-1)}/Q(n)^{\Theta(d^2)})$.*

Note that hyperslabs are degenerated triangles, so in 3D our lower bound for line-hyperslab intersection reporting applies to ray-triangle intersection reporting.

For triangle-triangle intersection reporting in 4D, we get:

Theorem 3.3.2 ([AC23a])**.** *Any pointer machine data structure that solves line-hyperslab intersection reporting in $\mathbb{R}^4$ in time $Q(n)+O(k)$, where n is the input size and k is the output size, must use space $\overset{o}{\Omega}(n^6/Q(n)^{125})$.*

The exponents in $Q(n)$ can potentially be improved, but most importantly, these new results answered the question raised by Ezra and Sharir about the possibility of better "fast-query" data structures.

Indeed, when the query time is $n^{o(1)}+O(k)$, our lower bounds almost match the current best upper bounds.

Theorem 3.3.3 ([AC23a])**.** *Any pointer machine data structure that solves line-hyperslab intersection reporting in $\mathbb{R}^d$ in time $n^{o(1)}+O(k)$, where n is the input size and k is the output size, must use space $\overset{o}{\Omega}(n^{2(d-1)})$.*

Theorem 3.3.4 ([AC23a])**.** *Any pointer machine data structure that solves triangle-triangle intersection reporting in $\mathbb{R}^4$ in time $n^{o(1)}+O(k)$, where n is the input size and k is the output size, must use space $\overset{o}{\Omega}(n^6)$.*

This shows improvement comparable to that given by Ezra and Sharir [ES22a] is not possible if we want to solve flat-object intersection reporting problems with $n^{o(1)}$ query time.

3.4 Highlights of Ideas and Techniques

In this section, we sketch the idea for showing a lower bound for line-hyperslab intersection reporting in 3D. This contains most of the important ideas we need to show a general lower bound for line-hyperslab intersection reporting in higher dimensions or triangle-triangle intersection reporting.

The Overall Idea

The overall idea of the lower bound is by a reduction from a special type of semialgebraic range reporting to line-hyperslab intersection reporting. Then we prove a lower bound for the special type of semialgebraic range reporting problem similar to what we did in Chapter 2. We have already seen the reduction in Observation 3.1.1, so the focal point in this section will be showing a lower bound for the semialgebraic range reporting problem. In our case, it is a (special) polynomial slab reporting problem.

Lower Bound for a Special Polynomial Slab Reporting Problem

The high-level idea used here is similar to the polynomial slab reporting lower bound proof presented in Chapter 2. For the sake of readability, we recall the approach. We will use the lower bound framework in Theorem 2.2.2. To this end, we need to construct a set of polynomial slabs in a cube $\mathscr{C}$ in 3D satisfying the two conditions in Theorem 2.2.2.

Like what we did in Chapter 2, we construct polynomial slabs by first generating a set of polynomials (of certain forms) and then generate polynomial slabs using them. Since we view $\mathbf{a}$ as coefficients and each $P(\mathbf{a},\mathbf{b})$ is treated as a polynomial in $\mathbf{b}$, we will use $P(\mathbf{b})$ to denote $P(\mathbf{a},\mathbf{b})$ for brevity. The key point of the analysis is to bound the intersection volume of two polynomial slabs, which boils down to analyzing how long the zero sets of the two corresponding polynomials can be close to each other. This is similar to Lemma 2.4.2, but we cannot directly apply Lemma 2.4.2 as the polynomials in this problem are of form

$$P(\mathbf{b}) = (a_{1,1}a_{2,2}-a_{2,1}a_{1,2})(b_{1,1}b_{2,2}-b_{1,2}b_{2,1})+\sum_{i=1}^{2}(a_{i,1}-a_{0,1}a_{i,2})b_{1,i}+\sum_{i=1}^{2}a_{0,1}a_{i,2}b_{2,i}+a_{0,1},$$

which is different from the form of polynomials considered in Chapter 2

$$Q(X) = X_1 - X_2^{\Delta} + \sum_i A_i X^i,$$

for $A_i = o(1)$ in two major ways.

First, the form of $P(\mathbf{b})$ seems to be quite messy to analyze. Recall that in Chapter 2, one reason why getting a lower bound was even possible is that $Q(X)$ is close to the irreducible polynomial $X_1 - X_2^{\Delta}$ (since A_i's are $o(1)$). However, we crucially need this for the analysis. Roughly speaking, since X_1 is very close to X_2^{Δ}, it follows that any monomial $X_1^i X_2^j$ is close to $X_2^{i\Delta+j}$, which in turn allows us to apply multivariate polynomial interpolation. It is clear that this technique cannot be possibly generalized to fit our current setting.

For the other, recall that the key ingredient we need for the construction to work is that there must be one term of two polynomials we generate whose corresponding coefficients are sufficiently different. Since all the coefficients A_i's in $Q(X)$ are independent, it is easy to generate polynomials that are sufficiently different: we just pick coefficients that are sufficiently different. However, coefficients in $P(\mathbf{b})$ are dependent and thus it is not straightforward how we can guarantee big gaps between coefficients.

To address the first issue, we prove a general lemma that works for polynomials of form $P(X_1,X_2) = X_1G(X_2) - F(X_2)$ for univariate polynomials G and F with some restrictions on their coefficients. Note that this is much more general than polynomials of form $Q(X_1,X_2) = X_1 - X_2^{\Delta} + \sum_{i_1,i_2} A_{i_1i_2}X_1^{i_1}X_2^{i_2}$ with some restrictions on the coefficients $A_{i_1i_2}$ we studied in Chapter 2. To prove a lemma similar to Lemma 2.4.2 in Chapter 2 that describes the relationship between the gaps between coefficients and the interval length in which the zeros sets of two polynomials of this form are close to each other, we need much more general tools from algebraic geometry, e.g., the resultant and the Sylvester matrix of two polynomials, and the proof is rather technical. We refer the readers to Section 9.3 in Chapter 9 for details. In our example, by setting $a_{1,1}a_{2,2} - a_{21}a_{1,2}$ to be 1 and rearranging the terms in $P(\mathbf{b})$, we get

$$\begin{aligned}P(\mathbf{b}) &= b_{2,2}b_{1,1} + a_{01}a_{2,2}b_{2,2} + (a_{1,1} - a_{0,1}a_{1,2})b_{1,1} + g(b_{1,2}, b_{2,1}),\\ &= b_{2,2}(b_{1,1} + a_{0,1}a_{2,2}) + (a_{1,1} - a_{0,1}a_{1,2})b_{1,1} + g(b_{1,2}, b_{2,1}),\end{aligned}$$

where $g(b_{1,2}, b_{2,1}) = -b_{1,2}b_{2,1} + (a_{2,1} - a_{0,1}a_{2,2})b_{1,2} + a_{0,1}a_{1,2}b_{2,1} + a_{0,1}$. We remark that we can do this without losing the number of independent parameters because we have added an extra parameter in the representation of the objects in advance. Note that this is a special case of $P(X_1,X_2) = X_1G(X_2) - F(X_2)$ by vieweing $b_{2,2}$ as X_1 and $b_{1,1}$ as X_2. We have the following lemma for this special case.

Lemma 3.4.1. *Let $P(x,y) = xy + c_{11}x + c_{12}y + c_{13}$ and $Q(x,y) = xy + c_{21}x + c_{22}y + c_{23}$. Let $I = \{y \in \mathbb{R} | \exists x : (x,y) \in (P(x,y),w) \cap (Q(x,y),w)\}$. If $|c_{1i} - c_{1j}| \geq \tau$, then $|I| = O((w/\tau)^{1/3})$.*

The second problem of dependent coefficients is associated with the forms of the coefficients in $P(\mathbf{b})$. The coefficients are now polynomials in $\mathbf{a}$, and thus we need to make sure that these coefficients are sufficiently different such that we can apply Lemma 3.4.1. The key observation is that Lemma 3.4.1 only requires one of the coefficients to have a big gap. To see why this helps, let us look at one example. Let

$$P_1(\mathbf{b}) = b_{2,2}b_{1,1} + a_{0,1}a_{2,2}b_{2,2} + (a_{1,1} - a_{0,1}a_{2,2})b_{1,1} + a_{0,1}$$

and

$$P_2(\mathbf{b}) = b_{2,2}b_{1,1} + a'_{0,1}a'_{2,2}b_{2,2} + (a'_{1,1} - a'_{0,1}a'_{2,2})b_{1,1} + a'_{0,1},$$

and we assume all $a_{i,j}, a'_{i,j}$ are big enough constants. We claim that if we can set the gap between every $a_{i,j}$ and $a'_{i,j}$ to be either 0 or at least τ, the difference of all coefficients of $P_1(\mathbf{b})$ and $P_2(\mathbf{b})$ will then be 0 or $\Theta(\tau)$. Why is this the case? By Lemma 3.4.1, if $|a_{0,1} > a'_{0,1}| \geq \tau$ then we are done. Otherwise $a_{0,1} = a'_{0,1}$, but if $|a_{2,2} - a'_{2,2}| \geq \tau$ then we are still done since $|a_{0,1}a_{2,2} - a_{0,1}a'_{2,2}| = a_{0,1}|a_{2,2} - a'_{2,2}| \geq \tau$. Similarly it holds for the coefficients of $b_{1,1}$. Of course the real forms of $P(\mathbf{b})$ are more complicated, and we need the "slicing" idea we described in Chapter 2 before we reach this base bivariate case. But the high-level idea remains the same.

Lower Bounds for More General Flat Objects

We can generalize the lower bound to more general flat objects.

It is relatively straightforward to generalize the parametric representation in Eq. (3.1) of a line to any t-flat. We represent any t-flat in $\mathbb{R}^d$ that is not parallel to any of the axes by

$$\begin{bmatrix} a_{0,1} & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ a_{1,1} & a_{1,2} & \cdots & a_{1,t} & a_{1,t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{d-t,1} & a_{d-t,2} & \cdots & a_{d-1,t} & a_{d-t,t+1} \end{bmatrix} \cdot \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_t \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix},$$

where $\tau_1, \cdots, \tau_t$ are free variables, and $a_{i,j}$'s are parameters. Note that similar to the line case, we introduced one more parameters although there are only $(d-t)(t+1)$ independent parameters.

Similarly, we can formulate $(d-t)$-hyperslabs using $(d-t)(t+1)$ parameters

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & b_{1,1} & b_{1,2} & \cdots & b_{1,d-t} \\ 0 & 1 & \cdots & 0 & b_{2,1} & b_{2,2} & \cdots & b_{2,d-t} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{t,1} & b_{t,2} & \cdots & b_{t,d-t} \\ 0 & 0 & \cdots & 0 & b_{t+1,1} & b_{t+1,2} & \cdots & b_{t+1,d-t} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{d-1} \\ x_d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1+w \end{bmatrix},$$

where $b_{i,j}$'s are parameters to specify a $(d-t-1)$-dimensional flat and $w \in [0, w_0]$.

Using the same idea of the determinants, we can establish a reduction from a special type of polynomial slab reporting to t-hyperslab intersection reporting. The actual lower bound proof for the polynomial slab reporting problem is again based on the base bivariate case and the application of the "slicing" idea.

We refer the readers to Chapter 9 for more details.

Chapter 4

Range Summary Queries

One of the main challenges that data analysis and database fields face in the modern "big data" era is the increasing amount of data. Many unprecedented applications like deep neural network based computer vision [RDGF15], natural language processing [VSP$^+$17, DCLT19], and the recent large language models [FC20, Ope23] are made possible because of the availability of gigantic data, and they have dramatically changed our life in the last decade. In this development, the traditional ways of data processing have become increasingly inefficient or inappropriate. For example, in the computational geometry context, consider the two classic variants of range searching: range reporting and counting. On the one hand, range reporting queries provide full (and in some sense unnecessarily complete) information of the interested data, and as a result, the response time to a query may be very long; on the other hand, a range counting query returns only one number, which might be output efficiently, but it contains too little information to be useful. There are some early attempts aiming to address this problem by introducing more expressive statistics, e.g., range median and range mode [BKMT05, BGJrS11, JrL11], but in many scenarios, we still need a better characterization of the output data compared to these simple statistics. This leads to the concept of data summaries.

4.1 Data Summaries

The concept of data summaries is closely associated with the steaming algorithm community where the computational model has a limited amount of memory while the data is given as a data stream of a much larger size. The target is to generate some useful information of the data in this restrictive setting. An introduction to this field is out of the scope of this book, we refer interested readers to a classic introductory book [Mut05]. In the remaining part of this section, we will mainly focus on introducing a couple of summary types that are relevant to our results.

Heavy Hitter Summaries

One of the summary types that has been studied extensively is heavy hitter summaries, a generalization of modes. Informally speaking, a heavy hitter of a data set is an element whose frequency exceeds a certain threshold, and a heavy hitter summary is a set containing all the heavy hitters. It is defined formally as follows.

Definition 4.1.1 (ϕ-heavy hitter summaries)**.** *Given a multiset S and a parameter ϕ with $0 < \phi < 1$, an element $x \in S$ is called a ϕ-heavy hitter of S iff its frequency is at least $\phi|S|$, and a ϕ-heavy hitter summary is the set of all ϕ-heavy hitters of S.*

Computing exact ϕ-heavy hitters could be challenging, and since we have restricted memory size anyway, it is more natural to consider approximate heavy hitters.

Definition 4.1.2 (ε-approximate ϕ-heavy hitter summaries)**.** *Given a multiset S of elements and parameters ε, ϕ with $0 < \varepsilon \leq \phi < 1$, an ε-approximate ϕ-heavy hitter summary contains all elements with frequency larger than $(\phi + \varepsilon)|S|$ and no elements with frequency smaller than $(\phi - \varepsilon)|S|$. The elements with frequency between $(\phi - \varepsilon)|S|$ and $(\phi + \varepsilon)|S|$ may or may not be included in the summary.*

It is also common to study the variant where S is a set instead of a multiset and each element of S is associated with a color from a color set C, and we want to output heavy hitters for colors whose frequency (the number of elements bearing this color) is above some threshold.

Quantile Summaries

Another classic summary type is quantile summaries. This is a very useful tool to understand the distribution of the data set, and it is a generalization of medians. The definition of quantile summaries is rather straightforward.

Definition 4.1.3 (ϕ-quantile summaries)**.** *Given a set S of elements and a parameter ϕ with $0 < \phi < 1$, where each element $s \in S$ is associated with a weight from $\mathbb{R}$, an element $s \in S$ is called a ϕ-quantile if its weight $w(s)$ has rank $\phi|S|$ among the weights in S. A ϕ-quantile summary is a sequence of $1/\phi + 1$ elements $(x_0, x_1, \cdots, x_{1/\phi})$ where the rank of x_i is $i\phi|S|$.*

Similar to heavy hitter summaries, we usually study the approximate version of quantile summaries in the limited memory scenario.

Definition 4.1.4 (ε-approximate ϕ-quantile summaries)**.** *Given a set S of elements and parameters ε, ϕ with $0 < \varepsilon \leq \phi < 1$, where each element $s \in S$ is associated with a weight from $\mathbb{R}$, an ε-approximate ϕ-quantile summary is a sequence of $1/\phi + 1$ elements $(x_0, x_1, \cdots, x_{1/\phi})$ where the rank of x_i is $(i\phi \pm \varepsilon)|S|$.*

4.2 Range Summary Queries

The typical problem studied in the streaming community is how to construct summaries efficiently in the streaming model. We will however be interested in the query version of it: given a set of points as the input, we are interested in constructing summaries for each output of some query range in an online fashion. In the remaining of this chapter, we will consider the restricted case when $\phi = \varepsilon$. This gives us a bound for the output size in terms of the error parameter ε, i.e., the output size of an ε-approximate heavy hitter summary is $O(1/\varepsilon)$, and that for an ε-approximate quantile summary is $\Theta(1/\varepsilon)$. We define approximate range summary queries as follows.

Definition 4.2.1 (Approximate Range Summary Queries)**.** *In the problem of approximate range summary queries, we are given a set P of points in $\mathbb{R}^d$ each associated with a weight from a set $\mathscr{S}$ (depending on the summary type) and a parameter ε with $0 < \varepsilon < 1$, and we want to preprocess P into a data structure so that for any query range γ, we are able to generate an ε-approximate ε-summary for $P \cap \gamma$ efficiently.*

For example, when γ is chosen from the family of halfspaces and $\mathscr{S}$ is a set of colors, we can define halfspace approximate heavy hitter queries; or when γ is the family of dominance ranges and $\mathscr{S}$ is $\mathbb{R}$, we can define dominance approximate quantile queries.

Prior to our work, approximate range summary queries was studied systematically only in 1D. Yi, Wang, and Wei [YWW14] showed that it is possible to build a linear-sized data structure to support approximate heavy hitter and approximate quantile summary queries in time $O(\log n + \frac{1}{\varepsilon})$, which is essentially optimal. Their result is based on an approach they described as “exponentially decomposable summaries”. The key observation of their approach is as follows. Given a set of points in 1D, we build a balanced binary search tree on top of them. We store each point in the corresponding leaf of the tree, and for each internal node, we construct a ε_i-approximate summary for the set of points which are the descendants of the node. Intuitively speaking, since the number of points of a node in level i is double that of a node in level $i+1$, we can set ε_{i+1} to be a constant factor bigger than ε_i while preserving the error. Given a query range, we need to merge the summary of $O(\log n)$ nodes, and it can be shown that merging summaries generated from error parameters forming a geometric series can be done efficiently. However, it seems hard to generalize this idea to higher dimensions. A similar idea that works for higher dimensions is “mergeable summaries” [ACH+13], but unfortunately it cannot give the optimal query time.

Mergeable Summary Based Solutions

We first introduce the concept of “mergeable summaries” used in [ACH+13]. Let D_1, D_2 be two data sets (multisets) and ε be the error parameter. Let $S(D_1, \varepsilon), S(D_2, \varepsilon)$ be two summaries generated for D_1, D_2 using parameter ε respectively. We say that summary S is mergeable if there is an algorithm that takes $S(D_1, \varepsilon)$ and $S(D_2, \varepsilon)$ as the input and generates an output $S(D_1 \uplus D_2, \varepsilon)$ for $D_1 \uplus D_2$ where $\uplus$ denotes the multiset union operator. Note that since we require the error to be preserved, we may need extra information to be able to merge two summaries, so the size of the summary might need to be $\Omega(1/\varepsilon)$ for the merging algorithm to work.

Agarwal et al. [ACH$^+$13] studied the mergeability of data summaries systematically. For ε-approximate heavy hitter summaries, they showed that we can merge two summaries of size $O(1/\varepsilon)$ to get a new summary of the same size while preserving the error. Combining with the standard tree-based data structures for range searching problems, this gives, e.g., $O(n)$ space and $O(n^{1-1/d}/\varepsilon)$ data structure for halfspace approximate heavy hitter queries. On the other hand, for quantile summaries, mergeability requires the size of the summary to be $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. Again, combining the known results for range searching results, we can get range summary data structures for approximate quantile summary queries.

An alternative approach to solve approximate range summary queries is through a technique called "independent range sampling".

Independent Range Sampling Based Solutions

In the problem of independent range sampling, we are given a set P of points in $\mathbb{R}^d$, and we want to preprocess the points into a data structure such that given a query range γ and a value t, it returns t independent samples from $P \cap \gamma$. The important feature here is "inter-query independence", meaning if we issue the same query twice, the two samples we get are independent of each other. This is a very nice feature in terms of data analysis. This line of research was initiated by Hu et al. [HQT14] and followed up by Afshani and Wei [AW17] and further by Afshani and Phillips [AP19]. For a survey of this field, we refer the readers to [Tao22].

The following standard tool from the sampling theory connects independent range sampling and approximate summary queries.

Theorem 4.2.1 (ε-Approximation)**.** *Let (P,Γ) be a finite set system. For any parameter ε with $0 < \varepsilon < 1$, a subset $A \subset P$ is called an ε-approximation for (P,Γ) if*

$$\forall \gamma \in \Gamma, \left| \frac{|\gamma \cap A|}{|A|} - \frac{|\gamma \cap P|}{|P|} \right| \le \varepsilon.$$

For example, when P is a point set and Γ the collection of all possible subsets of P created by halfspaces in $\mathbb{R}^d$, we obtain a finite set system (P,Γ). This is usually known as a geometric set system, and an important observation is that different set systems created by different ranges have different complexities. For instance, given the same ground set P, simplices will be able to create more subsets than halfspaces. This is captured by the notion of VC-dimension [VC15], which plays an important role in the size of the ε-approximation.

To see why Theorem 4.2.1 is useful for range summary queries, consider, e.g., halfspace approximate heavy hitter summary queries. We first build the independent sampling data structure for the point set P, and to answer a query γ, we sample $\Theta(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ elements from $P \cap \gamma$ for a parameter $0 < \delta < 1$. It is known that by sampling $\Theta(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ elements from P for a set system (P,Γ) with bounded VC-dimension, we can get an ε-approximation for (P,Γ) with probability at least $1-\delta$ [VC15]. After obtaining an ε-approximation, note that Theorem 4.2.1 guarantees that the frequency of an element in the ε-approximation differs by at most $\varepsilon|P\cap\gamma|$, which is exactly what we need. Thus we can directly operate over the ε-approximation without blowing up the error too much. However, although this approach is simple, it requires us to sample $\Omega(\varepsilon^{-2})$ points, and thus the query time is $\Omega(\varepsilon^{-2})$; furthermore, it is Monte Carlo.

4.3 Halfspace and Dominance Range Searching

The main query types we will be considering are halfspace and dominance ranges. A halfspace range is defined by a linear inequality while a dominance range is defined by a point $r=(r_1,\cdots,r_d)$ in $\mathbb{R}^d$ and it contains all points $q=(q_1,\cdots,q_d)$ such that $q_i\le r_i$ for all $i=1,\cdots,d$.

In Chapter 2, we have seen how we can solve simplex range searching in time $O(n^{1-1/d})$ query time and $O(n)$ space. This bound holds for reporting queries as well. However, for halfspace range reporting queries, it is possible to answer queries much faster with linear space.

In a seminal paper [Mat92a], Matoušek showed that halfspace range reporting can be solved with a space-time tradeoff of $S(n)Q(n)^{\lfloor d/2\rfloor}=\tilde{O}(n^{\lfloor d/2\rfloor})$ by applying "shallow" versions of Partition Theorem 2.1.1 and Cutting Theorem 2.1.4. The exponent $\lfloor d/2\rfloor$ has the following intuitive explanation. We mention without proving that if we can solve halfspace emptiness, then we can essentially solve halfspace range reporting by an approach called parametric searching [Meg83] with only a polylogarithmic extra factor. Given a query halfspace, to determine if it contains a query point, it suffices to check if a point on the convex hull of the point set is contained in the halfspace. It is well known that the convex hull of a set of points in $\mathbb{R}^d$ has complexity $O(n^{\lfloor d/2\rfloor})$. For more details on the complexity of the convex hull size, we refer the readers to the classic discrete geometry book by Matoušek [Mat02].

More directly, Matoušek [Mat92a] developed the tools of shallow cuttings to solve halfspace range reporting. It is a variant of shallow cutting (see Theorem 2.1.4), and it has been widely used in halfspace and dominance range searching problems. We state the version in 3D, but it is possible to generalize the result to higher dimensions [Mat92a, Cha12]. Given a collection of hyperplanes in 3D, unlike the normal cuttings which generate simplices covering the entire space in 3D, k-shallow cuttings cover $(\le k)$-levels, i.e., the set of points with at most k hyperplanes passing below it. The shallow cutting theorem provides the guarantees of how many simplices are needed and the number of hyperplanes crossing each simplex as in the normal cuttings. More formally, the shallow cutting theorem is stated as follows.

Theorem 4.3.1 (3D Halfspace Shallow Cuttings [Mat92a, Cha12])**.** *Given a collection S of n hyperplanes in $\mathbb{R}^3$, a k-shallow cutting $\mathscr{C}$ for S is a collection of vertical prisms that cover the $(\le k)$-level of S and the number of hyperplanes intersecting* *each prism is upper bounded by $O(k)$. The number of such prisms, i.e., the size of $\mathscr{C}$, is upper bounded by $O(n/k)$.*

For an illustration in 2D, see Figure 4.1. The blue polygonal lines indicate the upper boundary of (≤ 2)-levels of the line arrangement. The shallow cutting contains three simplices (unbounded from below). Note that the simplices can intersect points of higher levels, but the key point is that all (≤ 2)-levels are covered by them.

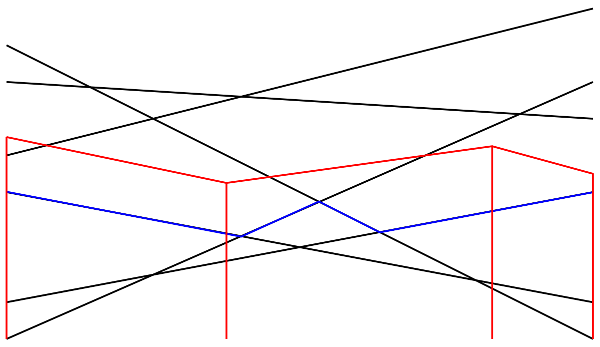


Figure 4.1: A 2D 3-Shallow Cutting

The shallow cutting theorem is usually used in a hierarchical fashion. Given a collection H of n hyperplanes in 3D, we build a hierarchy of 2^i-shallow cuttings for $i = 1, 2, \cdots, \log n$. Then given a query point, we can locate the first cell in this cutting hierarchy above it efficiently.

Lemma 4.3.1 (Afshani and Chan [AC07])**.** *Given a hierarchy of 2^i-shallow cuttings for $i = 1, 2, \cdots, \log n$ built for an arrangement of n planes in 3D, and for any query point q, we can find the first shallow cutting level k_i above q as well as the prism in that level containing q in time $O(\log n)$.*

Note that it is relatively easy to find the level and the prism in time $O(\log n \log\log n)$ by a binary search over levels and point locations on 2D planar subdivisions [dBCvKO08]. A finer probabilistic analysis gets rid of the $O(\log\log n)$ factor [AC07] . As an application of Lemma 4.3.1, we can solve approximate 3D halfspace range counting with a constant approximation factor in $O(n)$ space and $O(\log n)$ time [AC07].

Similar linear space and logarithmic query time results are also possible for dominance range reporting and approximate dominance range counting [AHZ10]. This is not surprising as dominance ranges can be viewed as a special case of halfspace ranges by an observation made by Chan, Larsen, and Pătraşcu [CLP11].

4.4 An Overview of Our Results

We study halfspace and dominance approximate heavy hitter summary (AHHS) and approximate quantile summary (AQS) queries in 2D and 3D. The model of computation we use is the *real*RAM model equipped with integer registers of size $w = \log n$. The following results were published in ICALP 2023 [ACBRW23].

We show the following results of AHHS queries.

Theorem 4.4.1 ([ACBRW23]). *3D halfspace AHHS queries can be answered with a data structure of space $O(n \log_w \frac{1}{\varepsilon})$ and query time $O(\log n + \frac{1}{\varepsilon})$.*

Note that the query time is worst-case optimal since the size of an ε-approximate heavy hitter summary is $O(1/\varepsilon)$. When the error parameter $\varepsilon = \Omega(1/\log^{O(1)} n)$, the solution is optimal. We can further reduce the space usage to linear for dominance ranges for any $\varepsilon > 0$.

Theorem 4.4.2 ([ACBRW23]). *3D dominance AHHS queries can be answered with a data structure of space $O(n)$ and query time $O(\log n + \frac{1}{\varepsilon})$.*

For AQS queries, we show the following.

Theorem 4.4.3 ([ACBRW23]). *3D halfspace AQS queries can be answered with a data structure of space $O(n \log^2 \frac{1}{\varepsilon} \log_w \frac{1}{\varepsilon})$ and query time $O(\log n + \frac{1}{\varepsilon} \log^2 \frac{1}{\varepsilon})$.*

Note that we need to pay some extra polylogarithmic factors in $\frac{1}{\varepsilon}$ in both space and query time compared to AHHS queries. We will see in the next section why this is the case. But interestingly, this is avoidable for dominance queries.

Theorem 4.4.4 ([ACBRW23]). *3D dominance AQS search can be answered with a data structure of space $O(n)$ and query time $O(\log n + \frac{1}{\varepsilon})$.*

A comparison between our new and known results is given in Table 4.1. The table is borrowed from [ACBRW23].

4.5 Highlights of Main Ideas and Techniques

We first present the main ideas behind our halfspace AHHS query structures.

AHHS Queries

We show a slightly worse $O(n \log \frac{1}{\varepsilon})$ space, $O(\log n + \frac{1}{\varepsilon})$ query time structure for halfspace AHHS queries. We briefly mention dominance AHHS queries at the end of this section. The main idea is to combine ε-approximation with shallow cuttings.

We transform the problem to the dual space using duality introduced in Chapter 2 and then construct hierarchical $(2^i/\varepsilon)$-shallow cuttings for $i = 0, 1, \cdots, \log n$. Note that by the guarantee given by Theorem 4.3.1, if there are k hyperplanes passing below a query point, then the prism we identify intersects $O(k)$ hyperplanes. For each prism, we build a data structure for the hyperplanes intersecting it to facilitate query answering. This is now a simpler problem since the total error allowed is εk while the total number of hyperplanes we need to consider is only $O(k)$. In other words, now the

Table 4.1: Our main results compared with Mergeability-based [ACH$^+$13] (denoted by M) and Independent Range Sampling (IRS)-based [AW17] (denoted by I) solution. The IRS-based solutions are randomized with success probability $1-\delta$ for a parameter $0<\delta<1$. F is the number of colors of the input. $w=\Theta(\log n)$ is the word size of the machine. † indicates optimal solutions.

Summary Query Types	**Space**	**Query Time**	**Remark**
3D AHHS Halfspace	$O(n)$ $O(n)$ $O(n\log_w \frac{1}{\varepsilon})$	$O(\log n+\frac{1}{\varepsilon}n^{2/3})$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon})$	M [ACH$^+$13] I [AW17] **New**
3D AHHS Dominance	$O(n)$ $O(n)$ $O(n)$	$O(\log n+\frac{1}{\varepsilon}\log^3 n)$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon})$	M [ACH$^+$13] I [AW17] **New†**
3D AQS Halfspace	$O(n)$ $O(n)$ $O(n\log^2\frac{1}{\varepsilon}\log_w\frac{1}{\varepsilon})$	$O(\log n+\frac{1}{\varepsilon}n^{2/3}\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})$	M [ACH$^+$13] I [AW17] **New**
3D AQS Dominance	$O(n)$ $O(n)$ $O(n)$	$O(\log n+\frac{1}{\varepsilon}\log^3 n\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon})$	M [ACH$^+$13] I [AW17] **New†**

error can be viewed as over the entire hyperplane set instead of the output hyperplanes (when we restrict ourselves to one prism). Indeed, we can show the following theorem for this problem.

Theorem 4.5.1 (Theorem 10.3.5, Chapter 10)**.** *Given a collection of H hyperplanes in $\mathbb{R}^3$ each associated with a color and a parameter ε, we can build a data structure of size $O(\min(|H|,\varepsilon^{-\frac{2d}{d+1}}))$ with query time $O(\varepsilon^{-1})$ such that given any halfspace query γ, we can output the colors in the query halfspace with frequency at least $\varepsilon|H|$.*

The value $O(\varepsilon^{-\frac{2d}{d+1}})$ is due to the fact that the ε-approximation of halfspace ranges is $O(\varepsilon^{-\frac{2d}{d+1}})$ [MV17]. With this theorem, at shallow cutting level i, we build an $O(\min(2^i/\varepsilon,\varepsilon^{-\frac{2d}{d+1}}))$ space data structure for the hyperplanes intersecting each prism and there are $n\varepsilon/2^i$ prisms in total. Note that when $0\le i\le\log\varepsilon^{\frac{1-d}{d+1}}$, we need to spend $O(n)$ space per level. But for higher levels, we spend $O(n)$ space in total. To sum up, we have a data structure of $O(n\log\frac{1}{\varepsilon})$ space that solves halfspace AHHS queries in time $O(\log n+\frac{1}{\varepsilon})$.

To obtain Theorem 4.4.1, we need more ideas. First, we can actually improve Theorem 4.5.1 by a better way of counting frequency colors using integer registers in the computational model we use. Second, we can use bit-packing to achieve a better data structure that counts exactly the frequencies of colors in a query halfspace range (a.k.a., type-2 range counting). They together allow us to compress every w levels for level i with $0 \leq i \leq \log \varepsilon^{\frac{1-d}{d+1}}$. Thus we can improve the $\log \frac{1}{\varepsilon}$ factor to $\log_w \frac{1}{\varepsilon}$. We remark that it is also possible to enhance this data structure so that it produces the approximate frequencies of the approximate heavy hitters it generates.

For dominance ranges, we apply roughly the same idea. The main reason why we can get a better bound is that the ε-approximation size for dominance ranges is much smaller ($\tilde{O}(1/\varepsilon)$ to be specific), and then we can further compress the layers (in fact to constant layers).

AQS Queries

We again start with halfspace AQS queries.

For halfspace AQS queries, again, we first apply the dual transformation and then hierarchical shallow cuttings. Now for each prism, we collect all the planes intersecting it and classify them into $t = \Theta(1/\varepsilon)$ lists $L_1, L_2, \cdots, L_t$, each of size at most $\varepsilon k/2$, based on the increasing order of their weights. Let q be some query point. Now consider the prism we locate for q. Suppose we can count the number of planes c_i passing below q efficiently among all lists L_i for $i = 1, 2, \cdots, t$, then we are done. This is because we can just walk through lists $L_1, \cdots, L_t$ and maintain a running counter c to record the number of hyperplanes below q we have encountered so far. Whenever the counter reaches $j\varepsilon k$ for $j = 0, 1, \cdots, 1/\varepsilon$ after adding c_i to c, we report an arbitrary hyperplane in L_i passing below q. This is okay because there are at most $\varepsilon k/2$ hyperplanes in each list, and so the error is at most $\varepsilon k/2$.

However, it is very inefficient to count each c_i exactly as exact range counting queries are expensive. Since we are computing approximate quantiles, the natural idea is to use approximate counting. However, we need to make sure that the error does not blow up since by adding c_i to c the error accumulates. We thus need to make the error of approximate counting smaller. Also, note that we need to approximate c_i for each of the $\Theta(1/\varepsilon)$ lists, and thus approximating c_i's separately will be inefficient. We would need to compute the approximate counts in a parallel fashion. It turns out that we can apply AHHS to address both problems. We now elaborate on the details.

To compute counts in parallel, we assign a unique color to each list and color the planes in the list with the color. Then we collect the planes and build a halfspace AHHS search structure with the approximate parameter set to be $\varepsilon/4$. By doing this, we can approximate each c_i with error $\varepsilon k/4$. This might look useless since the total error could still exceed the budget after adding four lists. Our next idea is to build a tree structure on top of the lists. The original t lists will be the leaves, and each internal node will contain the union of hyperplanes in the leaves of its subtree. We assign unique colors to the nodes and color the planes accordingly. Then we build our halfspace AHHS structure for all the planes in the tree. The total number of planes is $O(k \log t)$, and to approximate c_i, we only need to use $O(\log t)$ approximate counts. This means we can set the error parameter to be $\varepsilon/(\tau \log^2 t)$ for some big enough constant τ. This is the main reason why we have to pay an extra $\log^2 \frac{1}{\varepsilon}$ factor in the space and query time in Theorem 4.4.3.

The solution for dominance ranges is a bit different. The main reason is that it is possible to count c_i exactly by exploiting the fact that dominance ranges are orthogonal to the axes, and therefore we can use bit-tricks and do divide-and-conquer using the recursive grid idea [ABR00]. Thus we can get rid of the extra $\log^2 \frac{1}{\varepsilon}$ factor introduced by approximate counting in halfspace AQS queries.

We refer the readers to Chapter 10 for details.

Chapter 5

2D Generalization of Fractional Cascading on Axis-aligned Planar Subdivisions

Consider the dual version of range searching: we want to preprocess a collection of geometric ranges into a data structure such that we can identify the ranges stabbed by a query point efficiently. One way to solve the problem is to identify the cell which contains the query point in the arrangement formed by the ranges. This problem, known as point location, is one of the earliest problems studied in computational geometry. See, e.g., [dBCvKO08] for an introduction. In this chapter, we will explore "iterative search", which can be viewed as multiple point locations with the same query point but unrelated arrangements.

We start this chapter with an introduction to the classic (1D) fractional cascading approach which solves iterative search optimally. Then we consider iterative search in higher dimensions and describe the difficulty of generalizing fractional cascading to higher dimensions. Finally, we present our new 2D fractional cascading results and explain the key ideas and intuitions.

5.1 (1D) Fractional Cascading

The technique of fractional cascading was invented by Chazelle and Guibas [CG86a, CG86b] to solve the following "iterative search" problem.

Definition 5.1.1 (Iterative Search)**.** *Let $G = (V,E)$ be a connected undirected graph where the degree of each vertex $v \in V$ is bounded by some fixed constant, and each vertex is associated with a list of elements chosen from a totally ordered universe $\mathcal{U}$. We call G a catalog graph. Let n be the total number of elements attached.*

In the iterative search problem, we are to preprocess G as well as the associated lists into a data structure such that given any query (q,π) where $q \in \mathcal{U}$ and $\pi \subset G$

is a connected subgraph, we can find the successors of q in the lists attached to the vertices in π efficiently.

A very natural solution of $O(n)$ space and $O(|\pi|\log n)$ query time to the problem is to build the optimal successor search structures for all the lists separately, and then query each list in π one by one. However, the result of fractional cascading [CG86a, CG86b] tells us that with $O(n)$ space, surprisingly, it is possible to improve the query time to $O(\log n + |\pi|)$ in any pointer-based computational models.

Theorem 5.1.1 (Fractional Cascading [CG86a, CG86b]). *Iterative search can be solved in $O(n)$ space and $O(\log n + |\pi|)$ time.*

To illustrate the main idea of fractional cascading, let us consider a simpler problem where G is a path and $\pi \subset G$ is a subpath and each list has an equal size. Let $t = |G|$ be the total number of vertices in G and $L_1, \cdots, L_t$ be the lists attached to the vertices $v_1, \cdots, v_t$. Now we augment each list L_i to L'_i as follows. First we keep $L'_t = L_t$ and proceed backwards from L_t to L_1. We take every second element of L'_{i+1} to form a sample S_{i+1}. L'_i is then the union of L_i and S_{i+1}. To answer queries efficiently, for each element in L'_i, we maintain a successor pointer s which points to its successor in L_i (pointing to *null* if no successor exists) as well as a forward pointer f which points to its successor in S_{i+1}. Furthermore, for each element in S_{i+1}, we maintain a back pointer b which points to the corresponding element in L'_{i+1}. We build an optimal successor search data structure for each L'_i. See Figure 5.1 for an example.

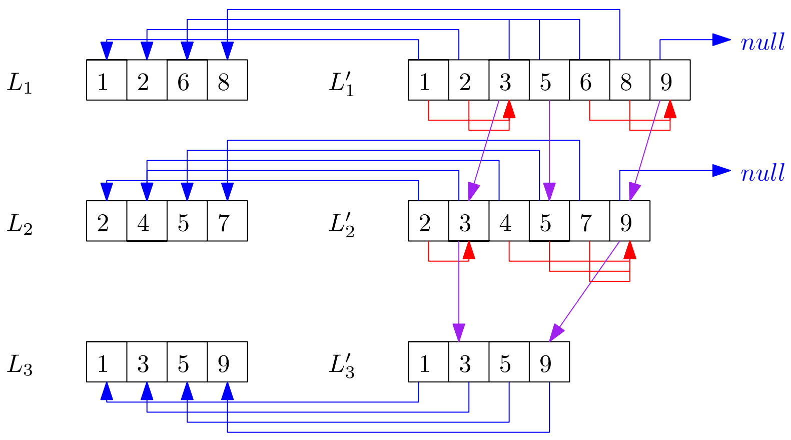


Figure 5.1: An example of fractional cascading. Blue, red, purple arrows are s, f, b pointers.

Let $\pi = v_l, v_{l+1}, \cdots, v_r$. To answer a query (q, π), we first locate the successor of q in L'_l. We use the successor pointer to find the successor in L_l and then use the forward pointer to find the successor of q in S_{l+1}. Using the back pointer, we proceed to L'_{l+1}. Note that the successor of q in L_{l+1} could be between two elements in L'_{l+1} and so we need to check one more element, but that takes constant time. We then proceed similarly using the successor, forward, and back pointers for all the lists.

The performance of this structure is relatively easy to analyze. First, observe that the size of each augmented list is $L'_i = \sum_{j=0}^{t-i} \frac{n}{t2^j} = O(\frac{n}{t})$. So the total size of the structure we built is still $O(n)$. After the initial $O(\log n)$ investment of querying the successor search structure, the subsequent successors can be found in $O(1)$ time by navigating the three pointers. Thus the query time is $O(\log n + |\pi|)$.

Since iterative search is a common problem in computational geometry, this technique has many applications. Chazelle and Guibas themselves showed the versatility of this technique in a companion paper [CG86b]. In Chapter 1, we introduced a two-level range tree structure for orthogonal range searching. Answering a query in this data structure reduces to answering $O(\log n)$ one-dimensional range searching queries attached to a tree structure, which is iterative search. There are also many follow-up works. This technique was also simplified using randomization by Sen [Sen95]. Mehlhorn and Näher [MN90] explored dynamic fractional cascading, and Afshani [Afs21] recently presented corresponding lower bounds.

5.2 Iterative Search in 2D and Early Attempts

Now we define the notion of iterative search in higher dimensions. In essence, iterative search is a series of point locations.

Successor Search and Point Location

One way to view successor search from a geometric perspective is the following. Given a collection of n points in the real line, the points partition the line into $n+1$ disjoint intervals. Finding the successor of a query point is equivalent to finding the interval to which the query point belongs.

A very natural generalization of this geometric perspective for successor search in 2D gives point location: we are given a planar subdivision, i.e., a planar embedding of a planar graph where all the edges are straight line segments, and for each query point, we want to find the cell containing the query point. See Figure 5.2 for an example.

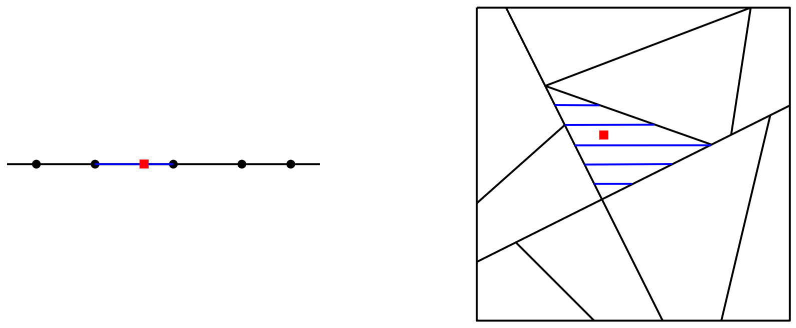


Figure 5.2: Left: Successor search for a query (red square) finds an interval (blue segment). Right: Point location for a query (red square) finds a cell (blue shaded cell).

We define the complexity of a planar subdivision to be the sum of the numbers of vertices, edges, and faces in it. Now we can define the 2D iterative search problem.

Definition 5.2.1 (2D Iterative Search)**.** *Let $G = (V,E)$ be a connected undirected graph where the degree of each vertex $v \in V$ is bounded by some fixed constant, and each vertex is associated with a collection of planar subdivisions. We call G a catalog graph. Let n be the total complexity of these planar subdivisions.*

In the 2D iterative search problem, we are asked to preprocess G as well as the associated planar subdivisions into a data structure so that given any query (q,π), where $q \in \mathcal{U}$ and $\pi \subset G$ is a connected subgraph, we can find the cells in subdivisions in π that contain q efficiently.

A Lower Bound for General 2D Fractional Cascading

Similar to 1D successor search, any improvement to 2D point location immediately leads to improvements to many other problems, e.g., 3D nearest neighbor searching and 2D ray shooting.

However, Chazelle and Liu [CL04] showed that 2D fractional cascading is impossible in general. Actually, the 2D proof we have seen for Theorem 2.3.2 in Chapter 2 already gave a lower bound for fractional cascading. Recall that in our construction, we showed a lower bound for the following problem: given a set of n points in a 2D grid of size $Q(n) \times n/Q(n)$, any data structure that can report all the grid points intersecting a query line in query time $Q(n)$ must use space $\Omega(n^2/Q(n)^2)$. We show that this problem reduces to 2D fractional cascading. First, observe that the duals to all the points sharing the same x-coordinate in the grid are parallel lines. Thus if we dualize each column of the grid, we get a collection of parallel lines. Adding a bounding box to each collection of parallel lines, we get a collection of planar subdivisions, and the total complexity is n. On the other hand, the dual of a query line is a point. Note that finding all the points intersecting a query line in the primal problem is equivalent to finding all the cells in the dual subdivisions containing a query dual point. The latter problem is exactly 2D fractional cascading. This establishes a reduction, and thus the lower bound for 2D simplex range reporting applies to 2D fractional cascading. See Figure 5.3 for an example.

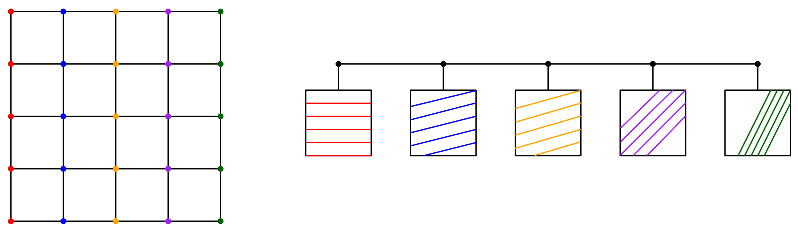


Figure 5.3: Reducing point-hyperplane reporting to 2D fractional cascading

A similar construction was given in [CL04]. This gives the following Theorem.

Theorem 5.2.1 (Chazelle and Liu [CL04]). *Any data structure that solves 2D iterative search in time $\log^{O(1)} n + O(|\pi|)$ where n is the input size and $|\pi|$ is the length of the query subgraph, must use space $\tilde{\Omega}(n^2)$.*

Essentially, this means fractional cascading is a one-dimensional technique and dashes the hope of generalizing it to higher dimensions. Also, note that the hard instance construction is not pathological: it is merely a collection of parallel strips attached to a path.

Better Upper Bounds for Special Cases

Although by Theorem 5.2.1, it is generally impossible to solve 2D iterative search significantly better than the trivial solution, in some interesting special cases, it is possible to circumvent this bottleneck.

One of the interesting examples is when the subdivision is axis-aligned, meaning all the edges are parallel to the x or y-axis. When studying orthogonal range reporting, Afshani et al. [AAL12] observed that by applying the known results for rectangle stabbing (defined later), iterative point locations among axis-aligned subdivisions following any path in a tree of height $O(\log n)$ can be answered in time $O(\log^{3/2} n)$ and space $O(n)$. This is an $O(\sqrt{\log n})$ improvement to the simple solution of $O(\log^2 n)$.

Another interesting case is the arrangement of lines. Since overlaying subdivisions formed by lines will not increase the complexity asymptotically, Chan and Zheng [CZ22] recently observed that under certain conditions it is possible to construct a randomized data structure that matches the 1D fractional cascading bound.

5.3 An Overview of Our Results

We start a systematic study of 2D fractional cascading in axis-aligned planar subdivisions. In this variant, each node of the graph is associated with an axis-aligned planar subdivision. We study the cases when the underlying catalog graph G is a path, a (degree-bounded) tree, or a (degree-bounded) graph. And the query subgraph π can be a subpath, subtree, or a subgraph of G accordingly. We present both upper and lower bounds for each of the cases. Our results can be summarized in the following table from [AC20]. These results were published in FOCS 2020 [AC20].

In the table, $\alpha(n)$ and $\alpha_v(n)$ are defined as follows. We first define $\alpha_2(n) = \log n$, and then we define $\alpha_i(n) = \alpha^*_{i-1}(n)$, the number of times we need to apply the $\alpha_{i-1}(\cdot)$ function to n until we reach a fixed constant. $\alpha(n)$ corresponds to the value of i such that $\alpha_i(n)$ is at most a fixed constant.

5.4 Highlights of Ideas and Techniques

Upper Bounds

We achieve the upper bound by combining several known tools in an innovative way.

Table 5.1: Our Results

Graph	**Query**	**Space**	**Query Time**	**Tight?**
Tree	Path	$O(n\alpha_c(n))$	$O(\min\{\lvert\pi\rvert\sqrt{\log n}, c\sqrt{\lvert\pi\rvert}\log n\} + \log n + \lvert\pi\rvert)$	Up to $\alpha_c(n)$ factor
Tree	Path	$O(n)$	$O(\min\{\lvert\pi\rvert\sqrt{\log n}, \alpha(n)\sqrt{\lvert\pi\rvert}\log n\} + \log n + \lvert\pi\rvert)$	Up to $\alpha(n)$ factor
Tree	Subtree	$O(n)$	$O(\log n + \lvert\pi\rvert\sqrt{\log n})$	yes
Graph	Path or Subgraph	$O(n)$	$O(\log n + \lvert\pi\rvert\sqrt{\log n})$	yes

The first tool we will use is a rectangle stabbing data structure by Afshani et al. [AAL12]. Rectangle stabbing is the natural dual problem of orthogonal range searching. In this problem, we are given a set $\mathscr{R}$ of axis-aligned rectangles in $\mathbb{R}^d$, and the goal is to preprocess $\mathscr{R}$ into a data structure so that for any query point q, we can report the rectangles intersecting q efficiently.

Theorem 5.4.1 (Afshani et al. [AAL12])**.** *There is a data structure of space $O(n\rho\log^{d-2}n)$ and query time $O(\log n(\log n/\log\rho)^{d-2}+k)$ for any parameter $\rho \geq 2$ that solves rectangle stabbing reporting in $\mathbb{R}^d$.*

We now present an optimal solution for 2D iterative search when G is a path and π is a subpath of G using Theorem 5.4.1.

An Optimal Subpath Query Data Structure for Catalog Paths

To apply rectangle stabbing structures, we first need to ensure each cell in the subdivisions is a rectangle. This can be achieved by applying trapezoidal decomposition (See, e.g., [dBCvKO08] for details). Simply put, this is done by shooting two vertical rays from each vertex, one upwards and one downwards, until they hit the edges. By the standard argument, this will only increase the complexity of the subdivision by a constant factor. Second, we divide G into $\lceil |G|/\log n\rceil$ subpaths, each of length at most $\log n$, and for each subpath, we collect all the rectangles in the subdivisions of all vertices in it and build a rectangle stabbing structure on top of it using Theorem 5.4.1. Note that when $d = 2$, we have a rectangle stabbing reporting structure of size $O(n)$ and query time $O(\log n + k)$. So the data structure we just built has total size $O(n)$. To answer a query (q,π), we query $\lceil|\pi|/\log n\rceil$ rectangle stabbing structures, each taking $O(\log n)$ time. So the query time is $O(\log n + |\pi|)$, which is clearly optimal.

This essentially gives the following result when the underlying graph is a path.

Theorem 5.4.2 ([AC20])**.** *Given a path where each vertex is associated with an axis-aligned planar subdivision of total complexity n, then we can build a data structure of size $O(n)$ so that for any query (q,π), where q is a query point and π is a query subpath, we can report all cells containing q along π in time $O(\log n + |\pi|)$.*

More General Catalog Graphs

This result we just obtained can be used as a subroutine to accelerate more general problems. For example, to answer path queries on catalog trees, we can decompose the tree into $O(\log n)$ paths using heavy path decomposition [ST83]. Then for each path, we build the data structure in Theorem 5.4.2. This data structure is of size linear and will give the optimal query time for query paths of length at least $\log^2 n$.

Now we handle the case when the length of a query path is smaller than $\log^2 n$. A relatively simple case is when the length of the query path is smaller than $\log n$. To answer such queries efficiently, we build local data structures for each path in the tree of length at most $\log r$ for a parameter $r > 0$ to fix later. Assume the degree of each vertex in the tree is at most three, then each vertex in the tree is contained in at most

$$\sum_{j=0}^{\log r}\sum_{i=0}^{j} 3^i \cdot 3^{j-i} = \Theta(r^{\log 3}\log r)$$

paths. If we build the data structure in Theorem 5.4.2 for each path, we will have a data structure of size $O(nr^{\log 3}\log r)$, which answers path queries in time $O(|\pi|\log_r n)$. To get linear space, r has to be a constant, but this gives us a rather bad $O(|\pi|\log n)$ query time. To reduce the query time, we need one extra idea.

The main observation here is that we do not need to store all the rectangles in the path query data structure. Suppose given a subdivision of complexity n_i, we can find a more coarse subdivision for it such that it contains $O(n_i/r^2)$ cells (of rectangles) and each cell contains $O(r^2)$ cells of the original subdivision, and then we can just store the cells of this coarse subdivision in the path query data structure. This will reduce the space to $O(nr^{\log 3}\log r/r^2) = O(n)$, and the cost is that we need to do one more point location among the coarse cell to find the original cell. But since each coarse cell contains at most r^2 original cells, this takes only $O(\log r)$ time. We set $r = 2^{\sqrt{\log n}}$, this means the total query time will be

$$O\left(\frac{|\pi|}{\log r}\cdot(\log n + \log r\cdot\log r)\right) = O\left(|\pi|\sqrt{\log n}\right).$$

Now the only remaining question is how to guarantee we can generate such a coarse subdivision. Fortunately, the result of "intersection-sensitive cutting" guarantees the existence of such coarse subdivisions.

Theorem 5.4.3 ([dBS95]). *Given a set H of n line segments in the plane with A intersections, we can construct a collection of $O(r + Ar^2/n^2)$ rectangles for $1 \le r \le n$ such that each rectangle intersects $O(n/r)$ line segments. The cutting and the conflict lists can be found in time $O(n\log r + Ar/n)$ by a randomized algorithm.*

We remark that this is the generalization of Theorem 2.1.4 we described in Chapter 2. Also note that here in our application $A = n$, and so the number of rectangles we get is $O(r)$, and each rectangle has complexity $O(n/r)$.

This gives us the following result.

Theorem 5.4.4 ([AC20]). *Given a tree where each vertex is associated with an axis-aligned planar subdivision of total complexity n, then we can build a data structure of size $O(n)$ so that for any query (q,π), where q is a query point and π is a query path, we can report all cells containing q along π in time $O(\log n + |\pi|\sqrt{\log n})$.*

This strategy is quite general, and it can be applied (with small changes) to obtain better path or subgraph queries for catalog graphs. The main difference is that the degree of each vertex in a general graph we consider is no longer three, but since it is still a constant, it is still fine. We omit the details here and refer the readers to Chapter 11 for details. We remark that this bound is tight when the length of the query path is at most $\log n$ as we will show later. Next, we consider the case when the length the query paths is between $\log n$ and $\log^2 n$.

Handling Query Paths of Length between $\log n$ and $\log^2 n$

We borrow the observation made by Afshani et al. [AAL12]. For simplicity, we assume the catalog tree is a perfect binary tree of height between $\log n$ and $\log^2 n$. We demonstrate the main idea by the following concrete example.

In Figure 5.4, we have a perfect binary tree (the left-hand side) of height three. We assign z-intervals to nodes of this tree in different levels as follows. The root node has z-interval $[0,1]$. The left child of the root node gets assigned $[0,1/2]$, and the right child $[1/2,1]$. We repeat this pattern for nodes in the third layer. Then we lift the subdivisions attached to the nodes to 3D by assigning the corresponding z-intervals. This gives the collection of rectangles on the right-hand side.

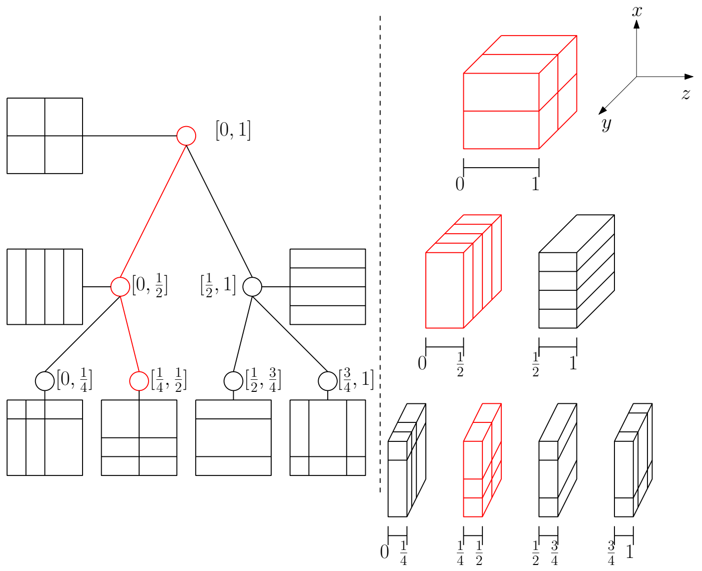


Figure 5.4: Equivalence between 2D iterative search and rectangle stabbing.

The key observation here is that given a path query (q,π), we can turn q into a 3D point by attaching an appropriate z-coordinate. For example, in the path query in Figure 5.4, we attach z-coordinate $(1/4+1/2)/2=3/8$ for the query path (in red color). And only the z-intervals of the red 3D rectangles can possibly contain the query point, which correspond to the subdivisions in the query path π.

In general, if we focus on root-to-leaf path queries. Let $l_1,\cdots,l_N$ be the leaves of a perfect binary tree ordered from left to right. For leaf i, we assign z-interval $[(i-1)/N, i/N]$ to all the rectangles in the subdivision attached to it. For an internal node, its z-interval is the union of the z-intervals of the leaves in its subtree. Then if we collect all these 3D rectangles and build a 3D rectangle stabbing data structure, for a root-to-leaf path query (q,π), it suffices to attach the z-coordinate of rectangles in the leaf node in π to q and query the 3D rectangle stabbing structure. Similar to what we did before, we create a coarse subdivision for each subdivision first. For each subdivision of size n_i, we create a coarse subdivision of size n_i/r such that each cell in it contains r original cells. Using the data structure in Theorem 5.4.1, we get a data structure of space $O(\frac{n}{r}\rho\log n)$ and query time $O(\log^2 n/\log\rho+|\pi|)$. By setting $\rho=\frac{r}{\log n/r}$ and $r=2^{\log n/\sqrt{|\pi|}}$, we get linear space and $O(|\sqrt{\pi}|\log n)$ query time.

Note that here we are only considering root-to-leaf path queries. In general, we would like to be able to handle query paths of arbitrary lengths. For simplicity, we only consider paths starting from the root for now. First observe that if we know that the query path is of length between h_1 and h_2 for two parameter $0<h_1<h_2$, we can build a data structure of space $O(n\log\frac{h_2}{h_1})$ and query time $O(\sqrt{|\pi|}\log n)$. This is achieved by first building a root-to-leaf data structure covering paths of length h_2. This data structure is useful for paths of length between $h_2/2$ and h_2. Then we build a root-to-leaf data structure covering paths of length $h_2/2$. This data structure is useful for paths of length between $h_2/4$ and $h_2/2$. We repeat this process $O(\log\frac{h_2}{h_1})$ times so that we can answer any query along path π of length $h_1<|\pi|<h_2$ efficiently in time $O(\sqrt{|\pi|}\log n)$. The space usage is simply bounded by $O(n\log\frac{h_2}{h_1})$. If we set $h_1=\log n$ and $h_2=\log^2 n$ then we get a space bound of $O(n\log\log n)$.

To reduce the space usage, observe that we can afford to build another layer of more coarse subdivisions over the coarse subdivision for paths of length between h_1 and $\log^2 n/\log^2\log n$ without exploding the space. Again we build an intersection-sensitive cutting of $n_i/\log n$ cells for each subdivision of complexity n_i, the space usage reduces to

$$S(n)=O\left(\frac{n}{\log n}\cdot\log\log n+n\right)=O(n),$$

and we need to do one more level of point location, which results in the (asymptotically) same query time

$$Q(n)=O\left(\sqrt{|\pi|}\log n\right)+O(|\pi|\log\log n)=O\left(\sqrt{|\pi|}\log n\right),$$

where the last equality follows from $|\pi|\le\frac{\log^2 n}{\log^2\log n}$.

For the remaining paths of length between $\log^2 n/\log^2\log n$ and h_2, using the data structure in the last paragraph results in a space usage of $O(n\log\log\log n)$ while the query time remains the same. Note that this is a better data structure for paths of length between h_1 and h_2. We then bootstrap and apply this idea multiple times ($\Theta(\log^* n)$ times to be exact) to further reduce the space. However, as a side-effect, we need to pay an extra $\Theta(\log^* n)$ factor. We remark that it is possible to bootstrap in a more efficient way which reduces the extra iterated logarithm factor to an inverse Ackermann factor. Also, by a finer analysis and some extra ideas, this structure also works for paths that do not start from the root. We refer the readers to Chapter 11 for further details. We conclude with the main upper bound theorem we obtained.

Theorem 5.4.5 ([AC20])**.** *Given a tree of height between* $\log n$ *and* $\log^2 n$ *where each vertex is associated with an axis-aligned planar subdivision of total complexity n, then we can build a data structure of size* $O(n\alpha_c(n))$ *for any constant c and the c-th function of the inverse Ackermann hierarchy* $\alpha_c(n)$ *so that for any query* (q,π)*, where q is a query point and* π *is a query path of length between* $\log n$ *and* $\log^2 n$*, we can report all cells containing q along* π *in time* $O(\log n + \sqrt{|\pi|}\log n)$*. Alternatively, we can build a data structure of linear space and query time* $O(\alpha(n)\sqrt{|\pi|}\log n)$ *where* $\alpha(n)$ *is the inverse Ackermann function.*

Lower Bounds

The high-level idea behind these lower bounds is the following. We first reduce a special rectangle stabbing problem to the 2D iterative search problem to which we want to show a lower bound. Then we show a lower bound for the rectangle stabbing problem using a lower bound framework by Afshani [Afs13]. By the reduction, this gives us a lower bound for the specific 2D iterative search problem. We first describe the lower bound framework.

A Lower Bound Framework for Geometric Range Stabbing

Similar to the lower bound framework for range reporting, this lower bound framework works in the pointer machine model of computation we described in Section 2. The difference is that here the lower bound is for the range stabbing problem where the input is a collection of geometric ranges and the query is a point. This is a natural dual problem of the range reporting problem.

Theorem 5.4.6 (Afshani [Afs13])**.** *Let* $S(n)$ *be the space usage and* $Q(n)+\gamma k$ *be the query time of any data structure that solves a geometric range stabbing reporting problem where n is the input size, k is the output size, and* $\gamma > 0$*. Let U be a unit cube. Suppose there exists a collection of n such geometric ranges* $\mathcal{S}_1,\cdots,\mathcal{S}_n$*, such that the following two conditions are satisfied:*

1. *Each point* $q \in U$ *is contained in exactly t ranges and* $\gamma t \geq Q(n)$*, for some parameter* $t > 0$ *.*
2. *The intersection volume of any two distinct ranges is upper bounded by v, for a parameter* $v > 0$ *.*

Then $S(n) = \Omega\left(\frac{t}{v2^{O(\gamma)}}\right)$.

Main Ideas to Lower Bounds

We sketch the lower bound proof for path queries of a catalog tree when the length of the query path is at most $\frac{\log n}{2}$, which contains most of the ideas we use for other cases.

As mentioned before, we need to reduce a (special) rectangle stabbing problem to the 2D iterative search problem we are interested in. This reduction is the reverse of the one we have seen when showing the upper bound of roo-to-leaf path queries on a perfect binary tree. This reduction is demonstrated in Figure 5.5.

On the left-hand side of Figure 5.5, we have three collections of rectangles from top to bottom. The first collection consists of rectangles with z-interval $[0,1]$. The second collection consists of two types of rectangles where the first type consists of rectangles with z-interval $[0,\frac{1}{2}]$ and the second type $[\frac{1}{2},1]$. Similarly, the third collection consists of four types of rectangles of different z-intervals.

We project these rectangles to the xy-plane and then attach these projections to a binary tree (the right-hand side of Figure 5.5). The key observation here is that any query point with its z-coordinate ranging between 0 and 1 will intersect exactly one rectangle in each collection, and these rectangles form a path in the right-side tree. This is equivalent to an iterative point location root-to-leaf path query in the binary tree. This establishes a reduction from a (special) rectangle stabbing problem to a 2D iterative search problem with axis-aligned subdivisions.

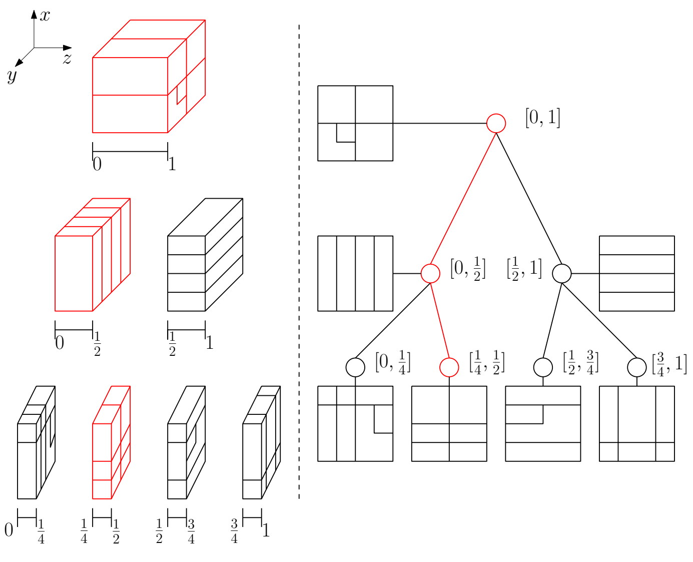


Figure 5.5: Reducing a rectangle stabbing problem to a 2D iterative search problem.

In general, a rectangle stabbing problem instance consisting of h collections of rectangles where collection i contains 2^{i-1} types of rectangles whose z-intervals spanning disjoint z-intervals in $[0,1]$ as described in the example enables us to establish a reduction to 2D iterative search along a catalog binary tree of height h, and the subdivisions are the projections of the rectangles in the xy-plane.

The next step is to show a lower bound for this special rectangle stabbing problem. The high-level idea is to generate rectangles of h different shapes. The side lengths of these shapes along the z-axis are decreasing in a geometric series $1, 1/2, 1/2^2, \cdots, 1/2^{h-1}$. We also ensure that the volume of each rectangle is $V = h/n$. We tile a unit cube using each shape. Note that we need $1/V$ copies of each shape to tile a unit cube. In total, we get a collection of $h \cdot 1/V = n$ rectangles. This forms a rectangle stabbing instance that reduces to iterative 2D point locations. Note that each point in the unit cube is contained in exactly h rectangles which is the output size. Suppose $\gamma h \geq Q(n)$ for some parameter γ, then the first condition in Theorem 5.4.6 is satisfied.

To satisfy the second condition in Theorem 5.4.6, we generate rectangles of shape

$$\frac{1}{2^{rj}} \times 2^{rj} \cdot 2^{ir+j} \cdot V \times \frac{1}{2^{ir+j}},$$

where $i = 0, 1, \cdots, \frac{h}{r} - 1$ and $j = 0, 1, \cdots, r-1$. Note that this satisfies our requirement for the side-lengths and each shape has volume V. With this construction, we can show that there exist two rectangles whose intersection is upper bounded by $v = \frac{V}{2^r}$ by case analysis. Since $h \leq \frac{\log n}{2}$, by setting $r = \frac{\sqrt{\log n}}{4}$, we make sure that our construction is valid, i.e., the side-length of any rectangle is no more than 1.

Thus by the lower bound framework in Theorem 5.4.6, we get

$$S(n) = \Omega\left(\frac{h}{v 2^{O(\gamma)}}\right) = \Omega\left(\frac{h 2^r}{V 2^{O(\gamma)}}\right) = \Omega\left(\frac{n 2^{\sqrt{\log n}/4}}{2^{O(\gamma)}}\right),$$

where the second equality follows from $v = \frac{V}{2^r}$, and the last equality follows from $V = h/n$ and $r = \frac{\sqrt{\log n}}{4}$. This means that there is a $\gamma = \Omega(\sqrt{\log n})$, such that $S(n) = \Omega(n)$. Alternatively, for any data structure with space usage $S(n) = O(n)$, the first condition of Theorem 5.4.6 must not hold, meaning, we must have $Q(n) > \gamma h \implies Q(n) = \Omega(h\sqrt{\log n})$. Thus for a root-to-leaf query with $|\pi| = h$, we have a query time lower bound of $\Omega(|\pi|\sqrt{\log n})$ for any linear space data structure.

This gives us the following theorem.

Theorem 5.4.7 ([AC20]). *Any linear space data structure that can solve 2D iterative search where the underlying graph is a degree-bounded tree associated with orthogonal planar subdivisions of complexity n and each query subgraph is a path π of length at most $\frac{\log n}{2}$ must have query time $\Omega(|\pi|\sqrt{\log n})$.*

We mark that the requirement of $h \leq \frac{\log n}{2}$ is important for the construction to work. We need a different lower bound construction when the length of the query paths is between $\log n$ and $\log^2 n$. We refer the readers to Chapter 11 for details.

Part II

Publications

Chapter 6

An Optimal Lower Bound for Simplex Range Reporting

Abstract

We give a simplified and improved lower bound for the simplex range reporting problem. We show that given a set P of n points in $\mathbb{R}^d$, any data structure that uses $S(n)$ space to answer such queries must have $Q(n) = \Omega((n^2/S(n))^{(d-1)/d} + k)$ query time, where k is the output size. For near-linear space data structures, i.e., $S(n) = O(n \log^{O(1)} n)$, this improves the previous lower bounds by Chazelle and Rosenberg [CR96] and Afshani [Afs13] but perhaps more importantly, it is the first ever tight lower bound for any variant of simplex range searching for $d \geq 3$ dimensions.

We obtain our lower bound by making a simple connection to well-studied problems in incident geometry which allows us to use known constructions in the area. We observe that a small modification of a simple already existing construction can lead to our lower bound. We believe that our proof is accessible to a much wider audience, at least compared to the previous intricate probabilistic proofs based on measure arguments by Chazelle and Rosenberg [CR96] and Afshani [Afs13].

The lack of tight or almost-tight (up to polylogarithmic factor) lower bounds for near-linear space data structures is a major bottleneck in making progress on problems such as proving lower bounds for multilevel data structures. It is our hope that this new line of attack based on incidence geometry can lead to further progress in this area.

6.1 Introduction

In the problem of simplex range reporting, we are given a set P of n points in $\mathbb{R}^d$ as input and we want to preprocess P into a structure such that given any query simplex γ, we can report $P \cap \gamma$ efficiently. It is known that given $O(n)$ space, the problem can be solved using $Q(n) = O(n^{1-1/d} + k)$ query time where k is the output size, i.e., $|P \cap \gamma|$ [Cha12]. However, current best lower bounds only match this upper bound in the plane [CR96, Afs13] and the best known lower bound is off by a factor of $2^{O(\sqrt{\log n})}$ in higher dimensions [Afs13]. Closing this gap has been a long-standing open problem for this fundamental problem in computational geometry.

In this paper, we prove a tight query time lower bound for simplex range reporting in the pointer machine model in the case when the space usage is linear. Our proof dramatically simplifies the previously known (suboptimal) proofs in [CR96] and [Afs13]. We obtain the result by observing a connection to incidence geometry which allows us to use simple deterministic "grid-based" constructions and avoid the intricate probabilistic construction and measure analysis used in the previous proofs [Afs13, CR96].

Related Work.

Simplex range reporting is a classical and fundamental problem in computational geometry and can be viewed as the most general case of range searching as far as linear constraints are concerned. Indeed, by using multilevel data structures [Aga17] and polyhedron triangulation, any range intersection reporting problem with constant complexity linear inputs and queries reduces to simplex range reporting. In general, there are many flavors of the problem. Here, we focus on the reporting variant where given a query simplex, the goal is to output the list of points inside the query, a.k.a. "simplex range reporting". However, counting variants are also well-studied where the points have weights from a semi-group and given the query, the goal is to output the sum of the weights of the points inside the query, a.k.a. "simplex range searching".

We now quickly review the history of the problem. All the upcoming results apply to both variants of the problem. When discussing a data structure, we use $S(n)$ to refer to the space complexity and $Q(n)$ to refer to the query time (ignoring the time required to produce the output). Thus, with our notation, a data structure for simplex range reporting uses $S(n)$ space and it has the query time of $Q(n)+O(k)$.

The first nontrivial result for the problem dates back to the early 1980s [Wil82]. After many early attempts [EW86, Yao83, YDEP89, Avi84, Col85, YY85, HW87, Wel88, CW89], significant progress was made after the discovery of fundamental tools such as the partition theorem [Mat91, Mat93, Cha12] and cutting lemma [Mat91, Cha93, dBS95]. The first near-optimal solution of $O(n^{1+\varepsilon})$ space and $O(n^{1-1/d+\varepsilon}+k)$ query time[1] was found by Chazelle, Sharir, and Welzl [CSW92] and it was simplified and slightly improved by Matoušek [Mat93]. Finally, in 2012, Chan [Cha12] removed the ε factors in the space and query time [CSW92].

It is clear from the above bounds that simplex range searching is a difficult problem since using linear space, we can only improve the trivial query bound by an $n^{1/d}$ factor. In 1989, Chazelle formally proved the difficulty of the problem by showing a query time lower bound of $Q(n)=\Omega(n^{1-1/d}/\log n)$ for the general simplex range searching problem given linear space in the semigroup arithmetic model [Cha89]. Unlike the upper bounds, this lower bound does not apply to the simplex range reporting problem. Seven years later, Chazelle and Rosenberg [CR96] overcame this issue, and they showed that if the query time is $O(n^{\delta}+k)$, then the data structure must use $\Omega(n^{d-d\delta-\varepsilon})$ space, where k is the output size. Note that the conjectured space-

[1]In this paper, ε is an arbitrarily small positive constant.

time trade-off for this problem is $S(n) = O((n/Q(n))^d)$ and thus this lower bound is a factor n^ε factor away from this bound. It was observed by Afshani [Afs13] that another lower bound of Chazelle and Liu [CL04] for the two-dimensional fractional cascading problem in fact achieves the aforementioned conjecture space-time trade-off for simplex range reporting in the plane ($d = 2$). However, for $d \geq 3$, the only improvement is a lower bound by Afshani [Afs13] who showed a tighter query time lower bound of $\Omega(n^{1-1/d}/2^{O(\sqrt{\log n})})$ [Afs13] which narrows the gap from a polynomial (n^ε) factor to a sub-polynomial ($2^{O(\sqrt{\log n})}$) one. Completely eliminating this gap seems like a challenging problem since the techniques used by the previous lower bounds inherently tie to a long-standing open problem known as the Heilbronn's triangle problem [Rot76].

The lack of tight lower bounds for the simplex range reporting problem is also a bottleneck in trying to obtain lower bounds for some more complicated problems, for instance, for multilevel data structures (i.e., data structures that involve multiple levels of simplex range searching data structures).

Our Contribution.

We simplify and improve the lower bound for simplex range reporting by Chazelle and Rosenberg [CR96] and Afshani [Afs13]. Specifically, we show a lower bound of $Q(n) = \Omega((n^2/S(n))^{(d-1)/d} + k)$ for the problem. When $S(n) = O(n)$, we get a clean lower bound of $Q(n) = \Omega(n^{(d-1)/d} + k)$, which is the **first tight lower bound** for simplex range reporting for $d \geq 3$. By a known technique [Afs13], our result also improves the lower bound for halfspace range reporting in 9 and higher dimensions. Along the way, we made the observation that the point-hyerplane incidence problem is highly related to proving lower bounds for simplex range reporting.

6.2 Preliminaries of the Pointer Machine Lower Bound Framework

We will prove the lower bound for simplex range reporting in (an augmented version of) the pointer machine model. In this model, the data structure is modeled as a directed graph M. In each cell of M, we store an element of the input set $\mathscr{S}$ as well as two pointers to other cells. To find the answer to a query q, i.e., a subset $\mathscr{S}_q \subset \mathscr{S}$, the algorithm starts at a special "root" cell and explores a connected subgraph such that all elements in $\mathscr{S}_q$ can be found in some cell in the subgraph. During the process, we only charge for pointer navigations. Let M_q be the smallest connected subgraph in which every element of $\mathscr{S}_q$ is stored at least once. Clearly, $|M|$ is a lower bound for the space usage and $|M_q|$ is a lower bound for the query time. Note that this grants the algorithm unlimited computational power as well as full information about the structure of M.

We use the following pointer machine lower bound framework tailored for geometric range reporting problems by Chazlle [Cha90a] and Chazelle and Rosenberg [CR96].

Theorem 6.2.1 (Chazlle [Cha90a] and Chazelle and Rosenberg [CR96])**.** *Suppose there is a data structure of space $S(n)$ which can answer range reporting queries in time $Q(n)+O(k)$ where n and k are the input and output sizes respectively. Assume we can show the existence of a set $\mathscr{S}$ of n points such that there exist m subsets $q_1, q_2, \cdots, q_m \subset \mathscr{S}$, where $q_i, i = 1,2,\cdots,m$, is the output of some query and they satisfy the following two conditions: (i) for all $i = 1,2,\cdots,m$, $|q_i| \geq Q(n)$; and (ii) the size of the intersection of every $\beta \geq 2$ distinct subsets $q_{i_1}, q_{i_2}, \cdots, q_{i_\beta}$ is upper bounded by some value α, i.e., $|q_{i_1} \cap q_{i_2} \cap \cdots \cap q_{i_\beta}| \leq \alpha$. Then $S(n) = \Omega(\frac{\sum_{i=1}^m |q_i|}{\beta 2^{O(\alpha)}}) = \Omega(\frac{mQ(n)}{\beta 2^{O(\alpha)}})$.*

6.3 A Lower Bound for Simplex Range Reporting

Simplex Range Reporting Lower Bounds Through the Incidence Geometry Lens.

Now we proceed to prove the lower bound. Our first observation is that to get a lower bound for simplex range reporting, we only need to study a specific incidence geometry problem. This is due to the fact that hyperplanes are degenerated simplicies, and so to show a lower bound for simplex range reporting using Theorem 6.2.1, it suffices to give a point-hyperplane configuration satisfying the two conditions in Theorem 6.2.1. Stated in the language of incidence geometry, the first condition requires each hyperplane to be incident to enough (at least $Q(n)$) points. The second condition requires us to bound the size of $K_{\alpha,\beta}$ in the incidence graph. To put it more formally, Theorem 6.2.1 implies the following lemma:

Lemma 6.3.1. *If there exist a set P of $n > 0$ points and a set H of $m > 0$ hyperplanes each incident to at least $t \geq Q(n)$ points (called t-rich hyperplanes) in $\mathbb{R}^d$ with no complete bipartite subgraph $K_{\alpha,\beta}$ in the incidence graph $P \times H$, then the simplex range reporting problem has a lower bound of $S(n) = \Omega(\frac{mt}{\beta 2^{O(\alpha)}}) = \Omega(\frac{mQ(n)}{\beta 2^{O(\alpha)}})$.*

It turns out that the relationship between the number of point-hyperplane incidences and $K_{\alpha,\beta}$ is a well-studied problem in the incidence geometry community [BK03, AS07, She16, BCV19]. However, this is not directly relevant to us as we require each hyperplane to be "rich". The closest result of the problem we can find is the very recent work by Patáková and Sharir [PS22]. They showed the existence of n points and $\Theta(n^d/t^{d+1})$ t-rich hyperplanes with no $K_{\alpha,\beta}$ in the incidence graph for $\beta = 2$ and $\alpha = O(t^{(d-2)/(d-1)})$. They also showed a matching lower bound for the size of α given $\beta = 2$.

Unfortunately, their result does not give us a useful lower bound.[2] The main reason for this is that the lower bound in Lemma 6.3.1 has a $2^{O(\alpha)}$ factor in the

[2]In fact, by plugging the parameters in [PS22] in Lemma 6.3.1, we can only get a lower bound of $Q(n) = \Omega((\log \frac{n^d}{S(n)})^{\frac{d-1}{d-2}})$.

denominator and so to show a nontrivial lower bound, α has to be sub-logarithmic. In our proof, we will still use the construction in [PS22], but we prove an upper bound for β by fixing $\alpha = 2$. Note that this is the opposite to the case considered in [PS22].

A Simple Point-Hyperplane Incidence Geometry Lemma.

Here, we prove the following lemma:

Lemma 6.3.2. *There exists a configuration of n points and $m = \Theta(n^d/t^{d+1})$ t-rich hyperplanes with no $K_{2,\beta}$ in the incidence graph where $\beta = \Theta(n^{d-2}/t^{d(d-2)/(d-1)})$ for any positive integer $t \le cn^{1-1/d}$ for some small enough positive constant c.*

We consider the same construction in [PS22]. For the completeness and readability, we present the construction and reprove some basic facts we will use. W.l.o.g., we assume that $t^{1/(d-1)}$ and n/t are integers; otherwise we can increase t and decrease n slightly to ensure the assumption. (It can be easily shown that t, n will remain asymptotically the same after the process.) Let G be an integer grid in $\mathbb{R}^d$ of size $t^{1/(d-1)} \times t^{1/(d-1)} \times \cdots \times t^{1/(d-1)} \times n/t$. Clearly, G has n grid points. We construct hyperplanes of form

$$X_d = b + \sum_{i=1}^{d-1} a_i X_i,$$

where $a_i \in \{1, \cdots, A\}$ and $b \in \{1, \cdots, B\}$ for $A = \lfloor \frac{n}{dt^{d/(d-1)}} \rfloor$ and $B = \lfloor \frac{n}{dt} \rfloor$. Since $t \le cn^{1-1/d}$ for a small enough positive constant c, $A, B \ge 1$ and so our construction is valid. We create all the possible distinct hyperplanes by picking a_i's and b as above. Let $\mathscr{H}$ be the set of all the hyperplanes we generated this way. As we have A choices for each a_i and B choices for b, the total number of hyperplanes we generated is $m = |\mathscr{H}| = A^{d-1}B = \Theta(n^d/t^{d+1})$.

Now consider a hyperplane $h_j \in \mathscr{H}$ and its intersection with G. Observe that all the coefficients of h_j are positive integers. This means that plugging in an integer value x_i for X_i for $i = 1, \cdots, d-1$ will yield the integer value $x_d = b + \sum_{i=1}^{d-1} a_i x_i$ thus a point $(x_1, \cdots, x_d)$ with integer coordinates that lies on h_j. The value x_d is maximized when b is set to B and all a_i's are set to A. Furthermore, the largest value of the first $d-1$ dimensions of G is $t^{1/(d-1)}$. Since

$$B + (d-1)At^{1/(d-1)} \le n/t,$$

each hyperplane in $\mathscr{H}$ intersects exactly $(t^{1/(d-1)})^{d-1} = t$ grid points.

Finally, we bound β given $\alpha = 2$. We use the following simple lemma. This is the only new property we show in this construction and it has a very simple proof.

Lemma 6.3.3. *Any subset $\mathscr{H}' \subset \mathscr{H}$ of size $|\mathscr{H}'| \ge A^{d-2} + 1$ contains at most one point in common.*

Proof. We do proof by contradiction. Assume hyperplanes in $\mathscr{H}'$ have two distinct points g_1 and g_2 in common, then there must be at least one coordinate on which they differ. Note that the d-th coordinate cannot be the only difference between g_1 and g_2 because hyperplanes in $\mathscr{H}$ are not parallel to the d-th axis. W.l.o.g., we can assume that g_1 and g_2 differ in their $(d-1)$-th coordinate. By the pigeonhole principle, there will be two hyperplanes $h_1, h_2 \in \mathscr{H}'$ that have identical first $d-2$ coefficients. Assume h_1 is defined by coefficients $a_1, \cdots, a_{d-2}, a_{1,d-1}, b_1$ and h_2 is defined by coefficients $a_1, \cdots, a_{d-2}, a_{2,d-1}, b_2$. We can view h_1 and h_2 as linear functions, f_1 and f_2, from $\mathbb{R}^{d-1}$ to $\mathbb{R}$. Let $X^{(d-1)} = (X_1, \cdots, X_{d-1})$. We thus write

$$f_1(X^{(d-1)}) = b_1 + a_{1,d-1}X_{d-1} + \sum_{i=1}^{d-2} a_i X_i$$

and

$$f_2(X^{(d-1)}) = b_2 + a_{2,d-1}X_{d-1} + \sum_{i=1}^{d-2} a_i X_i.$$

Consider the function

$$D(X^{(d-1)}) = f_1(X^{(d-1)}) - f_2(X^{(d-1)}) = b_1 - b_2 + (a_{1,d-1} - a_{2,d-1})X_{d-1}.$$

Let g_1' and g_2' be the projection of g_1 and g_2 onto the first $d-1$ dimensions. Since h_1 and h_2 pass through points g_1 and g_2 we have

$$D(g_1') = D(g_2') = 0.$$

However, the function $D(\cdot)$ is essentially a univariate linear function (i.e., a line in the coordinate system defined by the $(d-1)$-th and d-th axes). Furthermore, since g_1 and g_2 have distinct $(d-1)$-th coordinates, it follows that this function is zero on two distinct points. This implies that the function $D(\cdot)$ must be identical to the zero function which implies $h_1 = h_2$, a contradiction. Thus, the lemma follows. $\square$

According to Lemma 6.3.3, there is no $K_{2,\beta}$ in the incidence graph of our construction for

$$\beta = A^{d-2} + 1 = \Theta(n^{d-2}/t^{d(d-2)/(d-1)}).$$

This completes the proof of Lemma 6.3.2.

Combining Them Together.

Now we are ready to show a lower bound for simplex range reporting. Suppose $\lceil Q(n) \rceil < cn^{1-1/d}$, where c is the constant in Lemma 6.3.2, then we can set $t = \lceil Q(n) \rceil$ and Lemma 6.3.2 applies. By Lemma 6.3.1, we obtain a lower bound of

$$S(n) = \Omega\left(\frac{\Theta\left(\frac{n^d}{Q(n)^{d+1}}\right) \cdot Q(n)}{\Theta\left(\frac{n^{d-2}}{Q(n)^{d(d-2)/(d-1)}}\right) \cdot 2^{O(2)}}\right) = \Omega\left(\frac{n^2}{Q(n)^{\frac{d}{d-1}}}\right) \implies Q(n) = \Omega\left(\left(\frac{n^2}{S(n)}\right)^{\frac{d-1}{d}}\right).$$

On the other hand, if $\lceil Q(n) \rceil \geq cn^{1-1/d}$, then there is nothing to prove since this is already a lower bound. To sum up, we have proved the following theorem:

Theorem 6.3.1. *The simplex range reporting problem has a lower bound of* $Q(n) = \Omega((n^2/S(n))^{(d-1)/d})$.

6.4 Open Problems

There are three main open problems. The first and the major open problem is to show a tight lower bound for super-linear space data structures for simplex range reporting. Our current construction is only optimal when the space usage is restricted to linear. Although it is one of the most important cases for the problem, it would be desirable to obtain a tight space-time tradeoff. The main challenge here is to generate more t-rich hyperplanes without increasing β too much while restricting α to be small, say a constant.

Second, it is open if we can achieve tight lower bounds for other models of computation. For example, can we get a tight query time lower bound for the general simplex range searching problem in the semigroup arithmetic model given linear space? In this model, it is also possible to formulate a lower bound framework based on the point-hyperplane incidence property. But in this case, we need to bound α such that its value decreases proportional to β. See [Cha90a, Cha00] for the classical lower bound framework in this model. Unfortunately, our construction does not have this property.

Finally, it is interesting to see if such improvement can be made in related problems like multilevel data structures as well as the dual stabbing problems.

Chapter 7

Lower Bounds for Semialgebraic Range Searching and Stabbing Problems

Abstract

In the semialgebraic range searching problem, we are given a set of n points in $\mathbb{R}^d$ and we want to preprocess the points such that for any query range belonging to a family of constant complexity semialgebraic sets (Tarski cells), all the points intersecting the range can be reported or counted efficiently. When the ranges are composed of simplices, the problem is well-understood: it can be solved using $S(n)$ space and with $Q(n)$ query time with $S(n)Q(n)^d = \tilde{O}(n^d)$ where the $\tilde{O}(\cdot)$ notation hides polylogarithmic factors and this trade-off is tight (up to $n^{o(1)}$ factors). In particular, there exist "low space" structures that use $O(n)$ space with $O(n^{1-1/d})$ query time [Mat93, Cha12] and "fast query" structures that use $O(n^d)$ space with $O(\log n)$ query time [CZ23]. However, for general semialgebraic ranges, only "low space" solutions are known, but the best solutions [AMS13] match the same trade-off curve as simplex queries, with $O(n)$ space and $\tilde{O}(n^{1-1/d})$ query time. It has been conjectured that the same could be done for the "fast query" case but this open problem has stayed unresolved.

Here, we disprove this conjecture. We give the first nontrivial lower bounds for semialgebraic range searching and other related problems. More precisely, we show that any data structure for reporting the points between two concentric circles, a problem that we call 2D annulus reporting, with $Q(n)$ query time must use $S(n) = \overset{o}{\Omega}(n^3/Q(n)^5)$ space where the $\overset{o}{\Omega}(\cdot)$ notation hides $n^{o(1)}$ factors, meaning, for $Q(n) = \log^{O(1)} n$, $\overset{o}{\Omega}(n^3)$ space must be used. In addition, we study the problem of reporting the subset of input points in a polynomial slab defined by $\{(x,y) \in \mathbb{R}^2 : P(x) \le y \le P(x) + w\}$, where $P(x) = \sum_{i=0}^{\Delta} a_i x^i$ is a univariate polynomial of degree Δ and $a_0, \cdots, a_\Delta, w$ are given at the query time, a problem that we call polynomial slab reporting. For this, we show a space lower bound of $\overset{o}{\Omega}(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2})$, which implies that for $Q(n) = \log^{O(1)} n$, we must use

$\overset{o}{\Omega}(n^{\Delta+1})$ space. We also consider the dual semialgebraic stabbing problems of semialgebraic range searching, and present lower bounds for them. In particular, we show that in linear space, any data structure that solves 2D annulus stabbing problems must use $\Omega(n^{2/3})$ query time. Note that this almost matches the upper bound obtained by lifting 2D annuli to 3D. Like semialgebraic range searching, we also present lower bounds for general polynomial slab stabbing problems. Again, our lower bounds are almost tight for linear size data structures in this case.

7.1 Introduction

We address one of the biggest open problems of the recent years in the range searching area. Our main results are lower bounds in the pointer machine model of computation that essentially show that the so-called "fast query" version of the semialgebraic range reporting problem is "impervious" to the algebraic techniques. Our main result reveals that to obtain polylogarithmic query time, the data structure requires $\overset{o}{\Omega}(n^{\Delta+1})$ space[1], where the constant depends on Δ, n is the input size, and $\Delta+1$ is the number of parameters of each "polynomial inequality" (these will be defined more clearly later). Thus, we refute a relatively popular recent conjecture that data structures with $\overset{o}{O}(n^d)$ space and polylogarithmic query time could exist, where d is the dimension of the input points. Surprisingly, the proofs behind these lower bounds are simple, and these lower bounds could have been discovered years ago as the tools we use already existed decades ago.

Range searching is a broad area of research in which we are given a set P of n points in $\mathbb{R}^d$ and the goal is to preprocess P such that given a query range $\mathscr{R}$, we can count or report the subset of P that lies in $\mathscr{R}$. Often $\mathscr{R}$ is restricted to a fixed family of ranges, e.g., in simplex range counting problem, $\mathscr{R}$ is a simplex in $\mathbb{R}^d$ and the goal is to report $|P \cap \mathscr{R}|$, or in halfspace range reporting problem, $\mathscr{R}$ is a halfspace and the goal is to report $P \cap \mathscr{R}$. Range searching problems have been studied extensively and they have numerous variants. For an overview of this topic, we refer the readers to an excellent survey by Agarwal [GOT18].

Another highly related problem which can be viewed as the "dual" of this problem is range stabbing: we are given a set R of ranges as the input and the goal is to preprocess R such that given a query point p, we can count or report the ranges of R containing p efficiently. Here, we focus on the reporting version of range stabbing problems.

Range Searching: A Very Brief Survey

Simplex Range Searching

Simplices is one of the most fundamental families of queries. In fact, if the query is decomposable (such as range counting or range reporting queries), then simplices

[1] $\overset{o}{\Omega}(\cdot)$, $\overset{o}{O}(\cdot)$, $\overset{o}{\Theta}(\cdot)$ notations hide $n^{o(1)}$ factors and $\tilde{\Omega}(\cdot)$, $\tilde{O}(\cdot)$, $\tilde{\Theta}(\cdot)$ notations hide $\log^{O(1)} n$ factors.

can be used as "building blocks" to answer more complicated queries: for a query $\mathscr{R}$ which is a polyhedral region of $O(1)$ complexity, we can decompose it into $O(1)$ disjoint simplices (with a constant that depends on d) and thus answering $\mathscr{R}$ can be reduced to answering $O(1)$ simplicial queries.

Simplicial queries were hotly investigated in 1980s and this led to development of two important tools in computational geometry: cuttings and the partition theorem and both of them have found applications in areas not related to range searching.

Cuttings and Fast Data Structures "Fast query" data structures can answer simplex range counting or reporting queries in polylogarithmic query time but by using $O(n^d)$ space and they can be built using cuttings. In a nut-shell, given a set H of n hyperplanes in $\mathbb{R}^d$, a $\frac{1}{r}$-cutting is a decomposition of $\mathbb{R}^d$ into $O(r^d)$ simplices such that each simplex is intersected by $O(n/r)$ hyperplanes of H. These were developed by some of the pioneers in the range searching area, such as Clarkson [Cla87], Haussler and Welzl [HW87], Chazelle and Friedman [CF90], Matoušek [Mat91], finally culminating in a result of Chazelle [Cha18] who optimized various aspects of cuttings. Using cuttings, one can answer simplex range searching queries with $O(n^d)$ space and $O((\log n)^{d+1})$ query time [Mat93]. Recently, the query time has been improved to $O(\log n)$ by Chan and Zheng [CZ23]. An interested reader can refer to a survey on cuttings by Chazelle [Cha18].

The Partition Theorem and Space-efficient Data Structures At the opposite end of the spectrum, simplex range searching queries can be answered using linear space but with a higher query time of $O(n^{1-1/d})$, using partition trees and the related techniques. This branch of techniques has a very interesting history. In 1982, Willard [Wil82] cleverly used the ham sandwich theorem to obtain a linear-sized data structure with a query time of $O(n^\gamma)$ for some positive constant $\gamma < 1$ for simplicial queries in 2D. After a number of attempts that either improved the exponent or generalized the technique to higher dimensions, Welzl [Wel88] in 1988 provided the first optimal exponent for the partition trees, then Chazelle *et al.* [CW89] provided the first near-linear size data structure with a query time of roughly $O(n^{1-1/d})$. Finally, a data structure with $O(n)$ space and $O(n^{1-1/d})$ query time was given by Matoušek [Mat93]. This was also simplified recently by Chan [Cha12].

Space/Query Time Trade-off It is possible to combine fast query data structures and linear-sized data structures to solve simplex queries with $S(n)$ space and $Q(n)$ query time such that $S(n)Q(n)^d = \tilde{O}(n^d)$. This trade-off between space and query time is optimal, at least in the pointer machine model and in the semigroup model [Afs13, AC23c, CR96, Cha89] (up to $n^{o(1)}$ or $\log^{O(1)} n$ factors depending on the model of computation).

Multi-level Structures, Stabbing and Other Related Queries By using multi-level data structures, one can solve more complicated problems where both the input and the query shapes can be simplicial objects of constant complexity. The best multi-level data structures use one extra $\log n$ factor in space and query time per level [Cha12] and there exist lower bounds that show space/query time trade-off should blow up by at least $\log n$ factor per level [AD18]. This means that problems such as simplex stabbing (where the input is a set of simplices and we want to output the simplices containing a given query point) or simplex-simplex containment problem (where the input is a set of simplices, and we want to output simplices fully contained in a query simplex) all have the same trade-off curve of $S(n)Q(n)^d = \tilde{O}(n^d)$ between space $S(n)$ and query time $Q(n)$.

Thus, one can see that the simplex range searching as well as its generalization to problems where both the input and the query ranges are "flat" objects is very well understood. However, there are many natural query ranges that cannot be represented using simplices, e.g., when query ranges are spheres in $\mathbb{R}^d$. This takes us to semialgebraic range searching.

Semialgebraic Range Searching

A semialgebraic set is defined as a subset of $\mathbb{R}^d$ that can be described as the union or intersection of $O(1)$ ranges, where each range is defined by a d-variate polynomial inequality of degree at most Δ, defined by at most $B = \binom{d+\Delta}{d}$ values given at the query time; we call B the *parametric dimension*. For instance, with $B = 3$, $\Delta = 2$, and given three values a, b and c at the query time, a circular query can be represented as $\left\{(X,Y) \in \mathbb{R}^2 | (X-a)^2 + (Y-b)^2 \le c^2\right\}$. In semialgebraic range searching, the queries are semialgebraic sets.

Before the recent "polynomial method revolution", the tools available to deal with semialgebraic range searching were limited, at least compared to simplex queries. One way to deal with semialgebraic range searching is through linearization [YY85]. This idea maps the input points to $\mathbb{R}^L$, for some potentially large parameter L, such that each polynomial inequality can be represented as a halfspace. Consequently, semialgebraic range searching can be solved with the space/query time trade-off of $S(n)Q(n)^L = \tilde{O}(n^L)$. The exponent of $Q(n)$ in the trade-off can be improved (increased) a bit by exploiting that in $\mathbb{R}^L$, the input set actually lies in a d-dimensional surface [AM94].

In 2009, Zeev Dvir [Dvi09] proved the discrete Kakeya problem with a very elegant and simple proof, using a polynomial method. Within a few years, this led to a revolution in discrete and computational geometry, one that was ushered in by Katz and Guth's almost tight bound on Erdős distinct distances problem [GK15]. For a while, the polynomial method did not have much algorithmic consequences but this changed with the work of Agarwal, Matoušek, and Sharir [AMS13] where they showed that at least as long as linear-space data structures are considered, semialgebraic range queries can essentially be solved within the same time as simplex queries (ignoring some lower order terms). Later developments (and simplifications) of their approach by Matoušek and Patáková [MP15] lead to the currently best results: a data structure with linear size and with a query time of $\tilde{O}(n^{1-1/d})$.

Fast Queries for Semialgebraic Range Searching: an Open Problem Nonetheless, despite the breakthrough results brought on by the algebraic techniques, the fast query case still remained unsolved, even in the plane: e.g., the best known data structures for answering circular queries with polylogarithmic query time still use $\tilde{O}(n^3)$ space, by using linearization to $\mathbb{R}^3$. The fast query case of semialgebraic range searching has been explicitly mentioned as a major open problem in multiple recent publications[2]. In light of the breakthrough result of Agarwal *et al.* [AMS13], it is quite reasonable to conjecture that semialgebraic range searching should have the same trade-off curve of $S(n)Q(n)^d = \tilde{O}(n^d)$.

Nonetheless, the algebraic techniques have failed to make sufficient advances to settle this open problem. The best known "fast query" result by Agarwal *et al.* [AAEZ21] uses $O(n^{B+\varepsilon})$ space. In general, B can be much larger than d and thus it leaves a big gap between the currently best upper bound and the conjectured one. Given that it took a revolution caused by the polynomial method to advance our knowledge of the "low space" case of semialgebraic range searching, it is not too outrageous to imagine that perhaps equally revolutionary techniques are needed to settle the "fast query" case of semialgebraic range searching.

Semialgebraic Range Stabbing

Another important problem is semialgebraic stabbing, where the input is a set of n semialgebraic sets, i.e., "ranges", and queries are points. The goal is to output the input ranges that contain a query point. Here, "fast query" data structures are possible, for example by observing that an arrangement of n disks in the plane has $O(n^2)$ complexity, counting or reporting the disks stabbed by a query point can be done with $O(n^2)$ space and $O(\log n)$ query time (ignoring the output size) using slab and persistence[3] [dBCvKO08]. However, this simple idea does not work for general semialgebraic sets in higher dimensions. The main bottleneck before the recent "polynomial method revolution" is that for dimensions higher than 4 no technique was able to decompose the arrangement formed by a collection of algebraic surfaces into an arrangement with the property that each cell is a constant-complexity semialgebraic set and the complexity of the decomposed arrangement is close to that of the original arrangement. This was also one of the roadblocks for the "fast query" version of semialgebraic range searching. Recently, Guth [Gut15] showed the existence of polynomials whose connected components give such decompositions for arbitrary dimensions. An efficient algorithm for computing such partitioning polynomials was given very recently by Agarwal *et al.* [AAEZ21], and as one of the results, they

[2]To quote Agarwal *et al.* [AMS13],"[a] very interesting and challenging problem is, in our opinion, the fast-query case of range searching with constant-complexity semialgebraic sets, where the goal is to answer a query in $O(\log n)$ time using roughly $O(n^d)$ space." The same conjecture is repeated in a different survey [Aga17] and it is also emphasized that the question is even open for disks in the plane, "... whether a disk range-counting query in $\mathbb{R}^2$ be answered in $O(\log n)$ time using $O(n^2)$ space?".

[3]The arrangement of n circles has complexity $O(n^2)$. Disk stabbing queries are reduced to point location queries, which can be answered by building vertical slabs and persistent structures along the x-axis.

showed it is possible to build "fast query" data structures for semialgebraic stabbing problems using $O(n^{d+\varepsilon})$ space for n semialgebraic sets in $\mathbb{R}^d$. However, comparing to range searching problems, in this stabbing scenario, it seems difficult to make advancements in the "low space" side of things; e.g., for the planar disk stabbing problem, the only known data structure with $O(n)$ space is one that uses linearization to 3D that results in $\tilde{O}(n^{2/3})$ query time.

Our Results

Our main results are lower bounds in the pointer machine model of computation for four central problems defined below. In the *2D polynomial slab reporting* problem, given a set $\mathscr{P}$ of n points in $\mathbb{R}^2$, the task is to preprocess $\mathscr{P}$ such that given a query 2D polynomial slab $\mathscr{R}$, the points contained in the polynomial slab, i.e., $\mathscr{R} \cap \mathscr{P}$, can be reported efficiently. Informally, a 2D polynomial slab is the set of points (x,y) such that $P(x) \leq y \leq P(x) + w$, for some univariate polynomial $P(x)$ of degree Δ and value w given at the query time. In the *2D polynomial slab stabbing* problem, the input is a set of n polynomial slabs and the query is a point q and the goal is to report all the slabs that contain q. Similarly, in the *2D annulus reporting* problem, the input is a set P of n points in $\mathbb{R}^2$ and the query is an "annulus", the region between two concentric circles. Finally, in *2D annulus stabbing* problem, the input is a set of n annuli, the query is a point q and the goal is to report all the annuli that contain q.

For polynomial slab queries, we show that if a data structure answers queries in $Q(n) + O(k)$ time, where k is the output size, using $S(n)$ space, then $S(n) = \overset{o}{\Omega}(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2})$; the hidden constants depend on Δ. So for "fast queries", i.e., $Q(n) = \tilde{O}(1)$, $\overset{o}{\Omega}(n^{\Delta+1})$ space must be used. This is *almost tight* as the exponent matches the upper bounds obtained by linearization as well as the recent upper bound of Agarwal *et al.* [AAEZ21]! Also, we prove that any structure that answers polynomial slab stabbing queries in $Q(n) + O(k)$ time must use $\Omega(n^{1+2/(\Delta+1)}/Q(n)^{2/\Delta})$ space. In the "low space" setting, when $S(n) = O(n)$, this gives $Q(n) = \Omega(n^{1-1/(\Delta+1)})$. This is once again *almost tight*, as it almost matches the upper bounds obtained by linearization for when $S(n) = \tilde{O}(n)$.

For the annulus reporting problem, first note that annuli are geometric objects of a different type comparing to polynomial slabs in the sense that the degrees of x and y can both be more than 1, and so the intersection of two annuli can be more complicated than that of two polynomial slabs. As a result, the techniques we used for polynomial slabs does not apply. However, by studying the intersection of annuli of certain special properties, we manage to get the same $S(n) = \overset{o}{\Omega}(n^3/Q(n)^5)$ lower bound as polynomial slab reporting when $\Delta = 2$. For the annulus stabbing problem, we show $S(n) = \Omega(n^{3/2}/Q(n)^{3/4})$, e.g., in "low space" setting when $S(n) = O(n)$, we must have $Q(n) = \Omega(n^{2/3})$; compare this with simplex stabbing queries that can be solved with $O(n)$ space and $\tilde{O}(\sqrt{n})$ query time. As before, this is almost tight, as it almost matches the upper bounds obtained by linearization to 3D for when $S(n) = \tilde{O}(n)$.

Somewhat disappointedly, no revolutionary new technique is required to obtain these results. We use novel ideas in the construction of “hard input instances” but otherwise we use the two widely used pointer machine lower bound frameworks by Chazelle [Cha90a], Chazelle and Rosenberg [CR96], and Afshani [Afs13]. Our results are summarized in Table 1.

Table 7.1: Our Results, $*$ indicates this paper. In the table, $\overset{o}{\Omega}(\cdot)$ and $\overset{o}{O}(\cdot)$ notations hide $n^{o(1)}$ factors, and $\tilde{O}(\cdot)$ notation hides $\log^{O(1)} n$ factors.

Problem	**Lower Bound**	**Upper Bound**
2D Polynomial Slab Reporting	$S(n) = \overset{o}{\Omega}\left(\frac{n^{\Delta+1}}{Q(n)^{(\Delta+3)\Delta/2}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^{\Delta+1}}{Q(n)^{2\Delta}}\right)$ [AM94, AMS13, Mat93, AAEZ21]
When $Q(n) = \overset{o}{O}(1)$	$S(n) = \overset{o}{\Omega}\left(n^{\Delta+1}\right)^*$	$S(n) = \overset{o}{O}\left(n^{\Delta+1}\right)$ [AM94, AMS13, Mat93, AAEZ21]
2D Annulus Reporting	$S(n) = \overset{o}{\Omega}\left(\frac{n^3}{Q(n)^5}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^3}{Q(n)^4}\right)$ [AM94, AMS13, Mat93, AAEZ21]
When $Q(n) = \overset{o}{O}(1)$	$S(n) = \overset{o}{\Omega}\left(n^3\right)^*$	$S(n) = \overset{o}{O}\left(n^3\right)$ [AM94, AMS13, Mat93, AAEZ21]
2D Polynomial Slab Stabbing	$S(n) = \Omega\left(\frac{n^{1+2/(\Delta+1)}}{Q(n)^{2/\Delta}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^2}{Q(n)^{(\Delta+1)/\Delta}}\right)$ [AAEZ21]
When $S(n) = \overset{o}{O}(n)$	$Q(n) = \overset{o}{\Omega}\left(n^{1-1/(\Delta+1)}\right)^*$	$Q(n) = \overset{o}{O}\left(n^{1-1/(\Delta+1)}\right)$ [AAEZ21]
2D Annulus Stabbing	$S(n) = \Omega\left(\frac{n^{3/2}}{Q(n)^{3/4}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^2}{Q(n)^{3/2}}\right)$ [AAEZ21]
When $S(n) = \overset{o}{O}(n)$	$Q(n) = \overset{o}{\Omega}\left(n^{2/3}\right)^*$	$Q(n) = \overset{o}{O}\left(n^{2/3}\right)$ [AAEZ21]

7.2 Preliminaries

We first review the related geometric reporting data structure lower bound frameworks. The model of computation we consider is the pointer machine model. There are many variants of the pointer machine model, but generally speaking in this model the memory is represented by a directed graph M where each vertex represents a memory cell. A memory cell can either store an input element or an integer as well as a constant number of pointers to other cells. A memory cell can only be reached via pointer navigations from other memory cells (starting from a fixed “root” cell) and thus random accesses are disallowed in this model. Different variants of the model differ on what type of operations they allow the machine to perform as well as other aspects of the computation. Note that all the upper bounds in Table 1 can be implemented in a pointer machine model.

The variant we use is proposed by Chazelle [Cha90a] and it is defined as follows. Let $\mathscr{S}$ be the input set of elements. Assume a query q requires a subset $\mathscr{S}_q \subset \mathscr{S}$ to be output. Given the query q, the number of pointer navigations to answer the query forms a lower bound for the query time and thus we only charge the pointer navigations to the query time. Let M_q be the smallest connected subgraph such that every element of $\mathscr{S}_q$ is stored in at least one element of M_q. Clearly, $|M|$ is a lower bound for space and $|M_q|$ is a lower bound for query time. Note that this (implicitly) grants the algorithm unlimited computational power as well as full information about the structure of M (since we do not charge those to the query time). Also note that we can assume that each node has two pointers to other nodes as it will increase the query time by only a constant factor.

In our variant, there are two main lower bound frameworks, one for range reporting [Cha90a, CR96], and the other for its dual, range stabbing [Afs13]. We describe them in detail here.

A Lower Bound Framework for Range Reporting Problems

The following result by Chazelle [Cha90a] and later Chazelle and Rosenberg [CR96] provides a general lower bound framework for range reporting problems. In the problem, we are given a set $\mathscr{S}$ of n points in $\mathbb{R}^d$ and the queries are from a set $\mathscr{R}$ of ranges. The task is to build a data structure such that given any query range $\mathscr{R} \in \mathscr{R}$, we can report the points intersecting the range, i.e., $\mathscr{R} \cap \mathscr{S}$, efficiently.

Theorem 7.2.1 (Chazelle [Cha90a] and Chazelle and Rosenberg [CR96]). *Suppose there is a data structure for range reporting problems that uses at most $S(n)$ space and can answer any query in $Q(n)+O(k)$ time where n is the input size and k is the output size. Assume we can show that there exists an input set $\mathscr{S}$ of n points satisfying the following: There exist m subsets $q_1, q_2, \cdots, q_m \subset \mathscr{S}$, where $q_i, i = 1, \cdots, m$, is the output of some query and they satisfy the following two conditions: (i) for all $i = 1, \cdots, m$, $|q_i| \geq Q(n)$; and (ii) the size of the intersection of every $\alpha \geq 2$ distinct subsets $q_{i_1}, q_{i_2}, \cdots, q_{i_\alpha}$ is bounded by some value c, i.e., $|q_{i_1} \cap q_{i_2} \cap \cdots \cap q_{i_\alpha}| \leq c$. Then $S(n) = \Omega(\frac{\sum_{i=1}^m |q_i|}{\alpha 2^{O(c)}}) = \Omega(\frac{mQ(n)}{\alpha 2^{O(c)}})$.*

To use this framework, we need to exploit the property of the considered problem and come up with a construction that satisfies the two conditions above. Often, the construction is randomized and thus one challenge is to satisfy condition (ii) in the worst-case. This can be done by showing that the probability that (ii) is violated is very small and then using a union bound to prove that with positive probability the construction satisfies (ii) in the worst-case.

A Lower Bound Framework for Range Stabbing Problems

Range stabbing problems can be viewed as the dual of range reporting problems. In this problem, we are given a set $\mathscr{R}$ of n ranges, and the queries are from a set $\mathscr{Q}$ of n points. The task is to build a data structure such that given any query point $q \in \mathscr{Q}$, we can report the ranges "stabbed" by this query point, i.e., $\{\mathscr{R} \in \mathscr{R} : \mathscr{R} \cap q \neq \emptyset\}$, efficiently. A recent framework by Afshani [Afs13] provides a simple way to get the lower bound of such problems.

Theorem 7.2.2 (Afshani [Afs13]). *Suppose there is a data structure for range stabbing problems that uses at most $S(n)$ space and can answer any query in $Q(n)+O(k)$ time where n is the input size and k is the output size. Assume we can show that there exists an input set $R \subset \mathscr{R}$ of n ranges that satisfy the following: (i) every query point* *of the unit square U is contained in at least $t \geq Q(n)$ ranges; and (ii) the area of the intersection of every $\alpha < t$ ranges is at most v. Then $S(n) = \Omega(\frac{t}{v2^{O(\alpha)}}) = \Omega(\frac{Q(n)}{v2^{O(\alpha)}})$.*

This is very similar to framework of Theorem 7.2.1 but often it requires no derandomization.

Derandomization Lemmas

As mentioned before, one challenge of using Chazelle's framework is to show the existence of a hard instance in the worst case while the construction is randomized. To do that, we present two derandomization lemmas.

Lemma 7.2.1. *Let $\mathscr{P}$ be a set of n points chosen uniformly at random in a square S of side length n in $\mathbb{R}^2$. Let $\mathscr{R}$ be a set of ranges in S and $2 \leq t \leq n$ be a positive integer such that (i) the intersection area of any t ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots \mathscr{R}_t \in \mathscr{R}$ is bounded by $O\left(n/2^{\sqrt{\log n}}\right)$; (ii) the total number of t-wise intersections is bounded by $O\left(n^{2m}\right)$ for $m \geq 1$. Then with probability $> \frac{1}{2}$, for all distinct ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots, \mathscr{R}_t \in \mathscr{R}$, $|\mathscr{R}_1 \cap \mathscr{R}_2 \cap \cdots \mathscr{R}_t \cap \mathscr{P}| < 3m\sqrt{\log n}$ for a sufficiently large n.*

Proof. Consider any intersection region $\rho \in S$ of t ranges with area A. Let X be an indicator random variable with

$$X_i = \begin{cases} 1, \text{the } i\text{-th point is inside } \rho, \\ 0, \text{otherwise.} \end{cases}$$

Let $X = \sum_{i=1}^{n} X_i$. Clearly, $\mathbb{E}[X] = \frac{A}{n}$. By Chernoff's bound,

$$\Pr\left[X \geq (1+\delta)\frac{A}{n}\right] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\frac{A}{n}},$$

for any $\delta > 0$. Let $\tau = (1+\delta)\frac{A}{n}$, then

$$\Pr[X \geq \tau] < \frac{e^{\delta\frac{A}{n}}}{(1+\delta)^{\tau}} < \frac{e^{\tau}}{(1+\delta)^{\tau}} = \left(\frac{eA}{n\tau}\right)^{\tau}.$$

Now we pick $\tau = 3m\sqrt{\log n}$, since $A \leq cn/2^{\sqrt{\log n}}$ for some constant c, we have

$$\Pr\left[X \geq 3m\sqrt{\log n}\right] < \left(\frac{ce}{2^{\sqrt{\log n}}3m\sqrt{\log n}}\right)^{3m\sqrt{\log n}} < \frac{(ce)^{3m\sqrt{\log n}}}{n^{3m}}.$$

Since the total number of intersections is bounded by $O(n^{2m})$, the number of cells in the arrangement is also bounded by $O(n^{2m})$ and thus by the union bound, for sufficiently large n, with probability $> \frac{1}{2}$, the number of points in every intersection region is less than $3m\sqrt{\log n}$. □

Lemma 7.2.2. *Let $\mathscr{P}$ be a set of n points chosen uniformly at random in a square S of side length n in $\mathbb{R}^2$. Let $\mathscr{R}$ be a set of ranges in S such that (i) the intersection area of any range $\mathscr{R} \in \mathscr{R}$ and S is at least cnt for some constant $c \geq 4m$ and a parameter $t \geq \log n$, where $m \geq 2$; (ii) the total number of ranges is bounded by $O\left(n^{m+1}\right)$. Then with probability $> \frac{1}{2}$, for every range $\mathscr{R} \in \mathscr{R}$, $|\mathscr{R} \cap \mathscr{P}| \geq t$ for sufficiently large n.*

Proof. The proof of this lemma is similar to the one for Lemma 7.2.1. We pick n points in S uniformly at random. Let X_{ij} be the indicator random variable with

$$X_{ij} = \begin{cases} 1, \text{point } i \text{ is in range } j, \\ 0, \text{otherwise.} \end{cases}$$

We know that the area of each range is at least cnt. Then the expected number of points in each range is ct. Consider an arbitrary range, let $X_j = \sum_{i=1}^{n} X_{ij}$, then by Chernoff's bound

$$\Pr\left[X_j < \left(1 - \frac{c-1}{c}\right) ct\right] < e^{-\frac{\left(\frac{c-1}{c}\right)^2 ct}{2}}$$
$$\implies \Pr[X_j < t] < e^{-\frac{(c-1)^2 t}{2c}} < \frac{1}{n^{\frac{(c-1)^2}{2c}}} \leq \frac{1}{n^{2m-1+1/(8m)}}.$$

The second last inequality follows from $t \geq \log n$ and the last inequality follows from $c \geq 4m$ and $m \geq 2$. Since the total number of ranges is bounded by $O(n^{m+1})$, by a standard union bound argument, the lemma holds. □

7.3 2D Polynomial Slab Reporting and Stabbing

We first consider the case when query ranges are 2D polynomial slabs. The formal definition of 2D polynomial slabs is as follows.

Definition 7.3.1. *Let $P(x) = \sum_{i=0}^{\Delta} a_i x^i$, where $a_\Delta \neq 0$, be a degree Δ univariate polynomial. A 2D polynomial slab is a pair $(P(x), w)$, where $P(x)$ is called the base polynomial and $w > 0$ the width of the polynomial slab. The polynomial slab is then defined as $\mathbb{R}^2 : P(x) \leq y \leq P(x) + w\}$.*

2D Polynomial Slab Reporting

We consider the 2D polynomial slab reporting problem in this section, where the input is a set $\mathscr{P}$ of n points in $\mathbb{R}^2$, and the query is a polynomial slab. This is an instance of semialgebraic range searching where we have two polynomial inequalities which only differ by their constant terms where each inequality has degree Δ and it is defined by $\Delta + 1$ parameters given at the query time (thus, $B = \Delta + 1$). Note that $\Delta + 1$ is also the dimension of linearization for this problem, meaning, the 2D polynomial slab reporting problem can be lifted to the simplex range reporting problem in $\mathbb{R}^{\Delta+1}$. Our main result shows that for fast queries (i.e., when the query time is polylogarithmic), this is tight, by showing an $\overset{o}{\Omega}(n^{\Delta+1})$ space lower bound, in the pointer machine model of computation.

Before we present the lower bound, we first introduce a simple property of polynomials, which we will use to upper bound the intersection area of polynomials. Given a univariate polynomial $P(x)$, the following simple lemma establishes the relationship between the leading coefficient and the maximum range within which its value is bounded.

Lemma 7.3.1. *Let $P(x) = \sum_{i=0}^{\Delta} a_i x^i$ be a degree Δ univariate polynomial where $\Delta > 0$ and $|a_\Delta| \geq d$ for some positive d. Let w be any positive value and x_l be a parameter. If $|P(x)| \leq w$ for all $x \in [x_l, x_l + t]$, then $t \leq (\Delta+1)^3 \left(\frac{w}{d}\right)^{\frac{1}{\Delta}}$.*

Proof. First note that w.l.o.g., we can assume $x_l = 0$, because otherwise we can consider a new polynomial $P'(x) = P(x + x_l)$. Since $P'(x)$ is still a degree Δ univariate polynomial with $|a_\Delta| \geq d$, and for all $x \in [0,t]$, $P'(x) = P(x+x_l)$, to bound t, we only need to consider $P'(x)$ on interval $[0,t]$.

Assume for the sake of contradiction that $t > (\Delta+1)^3 \left(\frac{w}{d}\right)^{\frac{1}{\Delta}}$. We show that this will lead to $|a_\Delta| < d$.

We pick $\Delta + 1$ different points (x_i, y_i), where $x_i \in [0,t]$ and $y_i = P(x_i)$, on the polynomial. Then $P(x)$ can be expressed as

$$P(x) = \sum_{i=0}^{\Delta} y_i \prod_{j=0, j\neq i}^{\Delta} \frac{x - x_j}{x_i - x_j}.$$

The coefficient of the degree Δ term is therefore

$$a_\Delta = \sum_{i=0}^{\Delta} y_i \prod_{j=0, j\neq i}^{\Delta} \frac{1}{x_i - x_j}.$$

We pick $x_i = \frac{t}{i+1}$ for $i = 0, 1, \cdots, \Delta$ and we therefore obtain

$$a_\Delta = \sum_{i=0}^{\Delta} y_i \prod_{j=0, j\neq i}^{\Delta} \frac{1}{\frac{t}{i+1} - \frac{t}{j+1}}.$$

We now upper bound $\left|\frac{1}{\frac{t}{i+1} - \frac{t}{j+1}}\right|$. We assume $i < j$, the case for $i > j$ is symmetric. When $i < j$,

$$\left|\frac{1}{\frac{t}{i+1} - \frac{t}{j+1}}\right| = \frac{1}{\frac{t}{i+1} - \frac{t}{j+1}} = \frac{(i+1)(j+1)}{t(j-i)} < \frac{(\Delta+1)^2}{t},$$

where the last inequality follows from $i, j = 0, 1, \cdots, \Delta$ and $j - i \geq 1$. Also by assumption, $|y_i| \leq w$, we therefore have

$$|a_\Delta| < \frac{w(\Delta+1)(\Delta+1)^{2\Delta}}{t^\Delta} < d,$$

where the last inequality follows from $t > (\Delta+1)^3 \left(\frac{w}{d}\right)^{\frac{1}{\Delta}}$. However, in $P(x)$, $|a_\Delta| \geq d$, a contradiction. Therefore, $t \leq (\Delta+1)^3 \left(\frac{w}{d}\right)^{\frac{1}{\Delta}}$. □

With Lemma 7.3.1 at hand, we now show a lower bound for polynomial slab reporting.

Theorem 7.3.1. *Let $\mathscr{P}$ be a set of n points in $\mathbb{R}^2$. Let $\mathscr{R}$ be the set of all 2D polynomial slabs $\{(P(x),w) : \deg(P) = \Delta \geq 2, w > 0\}$. Then any data structure for $\mathscr{P}$ that solves polynomial slab reporting for queries from $\mathscr{R}$ with query time $Q(n)+O(k)$, where k is the output size, uses $S(n) = \overset{o}{\Omega}\left(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2}\right)$ space.*

Proof. We use Chazelle's framework to prove this theorem. To this end, we will need to show the existence of a hard input instance. We do this as follows. In a square S, we construct a set of special polynomial slabs with the following properties: (i) The intersection area of any two slabs is small; and (ii) The area of each slab inside S is relatively large. Intuitively and consequently, if we sample n points uniformly at random in S, in expectation, few points will be in the intersection of two slabs, and many points will be in each slab. Intuitively, this satisfies the two conditions of Theorem 7.2.1. By picking parameters carefully and a derandomization process, we get our theorem. Next, we describe the details.

Consider a square $S = [0,n] \times [0,n]$. Let d_i for $i = 1,2,\cdots,\Delta$ and w be some positive parameters to be specified later. We generate a set of $\Theta\left(\frac{n^\Delta}{2^\Delta \prod_{i=1}^{\Delta} d_i} \cdot \frac{n}{w}\right)$ polynomial slabs $(P(x),w)$ with

$$P(x) = \left(\sum_{i=1}^{\Delta} \frac{j_i d_i x^i}{n^i}\right) + kw$$

where $j_i = \lfloor \frac{n}{2d_i} \rfloor, \lfloor \frac{n}{2d_i} \rfloor + 1, \cdots, \lfloor \frac{n}{d_i} \rfloor$ for $1 \leq i \leq \Delta$ and $k = \lfloor \frac{n}{4w} \rfloor, \lfloor \frac{n}{4w} \rfloor + 1, \cdots, \lfloor \frac{n}{2w} \rfloor$. Note that we normalize the coefficients such that for any polynomial slab in range $x \in [0,n]$, a quarter of the x-interval length (or equivalently the area) of this slab is contained in S if $w < n/6$. See Figure 7.1 for an example. To show this, it is sufficient to show that every polynomial is inside S, for every $x \in [0,n/4]$. As all the coefficients of the polynomials are nonnegative, it is sufficient to upper bound $P(n/4)$, among all the polynomials $P(x)$ that we have generated. Similarly, this maximum is attained when all the coefficients are set to their maximum value, i.e., when $j_i = \lfloor n/d_i \rfloor$ and $k = \lfloor n/(2w) \rfloor$, whose value is upper bounded by polynomial $P_u(x) = \left(\sum_{i=1}^{\Delta} x^i/n^{i-1}\right) + \frac{n}{2}$. Now it easily follows that $P_u(n/4) < 5n/6$.

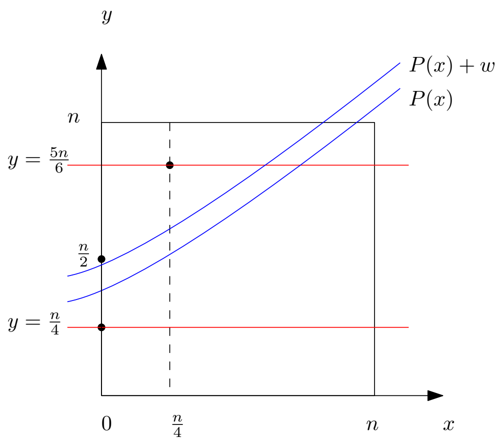


Figure 7.1: Intersection of a Polynomial Slab in Our Construction and S

Then, the claim follows from the following simple observation.

Observation 7.3.1. *The area of a polynomial slab $(P(x),w)$ for when $a \le x \le b$ is $(b-a)w$.*

Proof. The claimed area is $(\int_a^b (P(x)+w)dx) - (\int_a^b P(x)dx) = \int_a^b w dx = (b-a)w$. □

Next, we bound the area of the intersection of two polynomial slabs. Consider two distinct slabs $\mathscr{R}_p = (P(x),w)$ and $\mathscr{R}_q = (Q(x),w)$. Observe that by our construction, if $P(x)$ and $Q(x)$ only differ in their constant terms, their intersection area is 0. So we only consider the case that there exists some $0 < i \le \Delta$, such that the coefficients for x^i are different in $P(x)$ and $Q(x)$. As each slab is created using two polynomials of degree Δ, $\mathscr{R}_q \cap \mathscr{R}_p$ can have at most $O(\Delta)$ connected regions. Consider one connected region $\mathfrak{R}$ and let the interval $\eta = [x_1,x_2] \subset [0,n]$, be the projection of $\mathfrak{R}$ onto the X-axis. Define the polynomial $R(x) = P(x) - Q(x)$ and observe that we must have $|R(x)| \le w$ for all $x \in [x_1,x_2]$. We now consider the coefficient of the highest degree term of $R(x)$. Let $j_i d_i/n^i$ (resp. $j_i' d_i/n^i$) be the coefficient of the degree i term in $P(x)$ (resp. $Q(x)$). Clearly, if $j_i = j_i'$, then the coefficient of x^i in $R(x)$ will be zero. Thus, to find the highest degree term in $R(x)$, we need to consider the largest index i such that $j_i \ne j_i'$; in this case, $R(x)$ will have degree i and coefficient of x_i will have absolute value $\left|(j_i - j_i')d_i/n^i\right| \ge d_i/n^i$. By Lemma 7.3.1, $x_2 - x_1 \le O(\Delta^3)\left(\frac{wn^i}{d_i}\right)^{1/i}$. Next, by Observation 7.3.1, the area of the intersection of $\mathscr{R}_q$ and $\mathscr{R}_p$ is $O(\Delta^3)nw\left(\frac{w}{d_i}\right)^{1/i}$.

We pick $d_i = c\Delta^{3i}w^{i+1}2^{i\sqrt{\log n}}$ and $w = 16\Delta Q(n)$, for a large enough constant c. Then, the intersection area of any two polynomial slabs is bounded by $n/2^{\sqrt{\log n}}$. Since in total we have generated $O(n^{\Delta+1})$ slabs, the total number of pairwise intersections they can form is bounded by $O(n^{2(\Delta+1)})$. By Lemma 7.2.1, with probability $> \frac{1}{2}$, the number of points of $\mathscr{P}$ in any intersection of two polynomial slabs is at most $3(\Delta+1)\sqrt{\log n}$. Also, as we have shown that the intersection area of every slab with

S is at least $nw/4 = 4\Delta nQ(n)$, by Lemma 7.2.2, with probability more than $\frac{1}{2}$, each polynomial slab has at least $Q(n)$ points of $\mathscr{P}$[4].

It thus follows that with positive probability, both conditions of Theorem 7.2.1 are satisfied, and consequently, we obtain the lower bound of

$$S(n) = \Omega\left(\frac{Q(n) \cdot \frac{n^{\Delta}}{2^{\Delta}\prod_{i=1}^{\Delta} d_i} \cdot \frac{n}{w}}{2^{3(\Delta+1)\sqrt{\log n}}}\right) = \overset{o}{\Omega}\left(\frac{n^{\Delta+1}}{Q(n)^{(\Delta+3)\Delta/2}}\right),$$

where the last equality follows from $d_i = c\Delta^{3i}w^{i+1}2^{i\sqrt{\log n}}$, $w = 16\Delta Q(n)$, and

$$\prod_{i=1}^{\Delta} d_i = c\Delta^{3(1+\Delta)\Delta/2}w^{(2+\Delta+1)\Delta/2}2^{\sqrt{\log n}(1+\Delta)\Delta/2} = Q(n)^{(\Delta+3)\Delta/2}n^{o(1)}. \qquad \square$$

So for the "fast query" case data structure, by picking $Q(n) = \log^{O(1)} n$, we obtain a space lower bound of $S(n) = \overset{o}{\Omega}(n^{\Delta+1})$.

2D Polynomial Slab Stabbing

By small modifications, our construction can also be applied to obtain a lower bound for (the reporting version of) polynomial slab stabbing problems using Theorem 7.2.2.

One modification is that we need to generate the slabs in such a way that they cover the entire square S. The framework provided through Theorem 7.2.2 is more stream-lined and derandomization is not needed and we can directly apply the "volume upper bound" obtained through Lemma 7.3.1. There is also no $n^{o(1)}$ factor loss (our lower bound actually uses $\Omega(\cdot)$ notation). The major change is that we need to use different parameters since we need to create n polynomial slabs, as now they are the input.

Theorem 7.3.2. *Give a set $\mathscr{R}$ of n 2D polynomial slabs $\{(P(x), w) : deg(P) = \Delta \geq 2, w > 0\}$, any data structure for $\mathscr{R}$ solving the 2D polynomial slab stabbing problem with query time $Q(n) + O(k)$ uses $S(n) = \Omega\left(\frac{n^{1+2/(\Delta+1)}}{Q(n)^{2/\Delta}}\right)$ space, where k is the output size.*

Proof. We use Afshani's lower bound framework as described in Theorem 7.2.2. First we generate n polynomial slabs in a unit square $S = [0,1] \times [0,1]$ as follows. Consider $\Theta\left(\frac{1}{2^{\Delta}\prod_{i=1}^{\Delta} d_i} \cdot \frac{\Delta}{w}\right)$ polynomial slabs with their base polynomials being:

$$P(x) = \sum_{i=1}^{\Delta} j_i d_i x^i + kw,$$

where $j_i = \lfloor \frac{1}{2d_i} \rfloor, \lfloor \frac{1}{2d_i} \rfloor + 1, \cdots, \lfloor \frac{1}{d_i} \rfloor$ for $1 \leq i \leq \Delta$ and $k = -\lceil \frac{\Delta}{w} \rceil, -\lceil \frac{\Delta}{w} \rceil + 1, \cdots, \lceil \frac{\Delta}{w} \rceil$ for parameters $d_i > 0$, $i = 1, 2, \cdots, \Delta$ and w to be specified later. Note that by our construction, any point in the unit square is covered by $t = \Theta(1/(2^{\Delta}\prod_{i=1}^{\Delta} d_i))$ polynomial

[4]Since we work in the pointer machine, $Q(n) \geq \log n$, and thus Lemma 7.2.2 applies.

slabs. To see this, consider each polynomial of form

$$P'(x) = \sum_{i=1}^{\Delta} j_i d_i x^i,$$

where $j_i = \lfloor \frac{1}{2d_i} \rfloor, \lfloor \frac{1}{2d_i} \rceil + 1, \cdots, \lfloor \frac{1}{d_i} \rfloor$ for $1 \le i \le \Delta$. By shifting $P'(x)$ vertically with distance kw for $k = -\lceil \frac{\Delta}{w} \rceil, -\lceil \frac{\Delta}{w} \rceil + 1, \cdots, \lceil \frac{\Delta}{w} \rceil$, we generate a series of adjacent 2D polynomial slabs. For $x \in [0,1]$, $0 \le P'(x) \le \Delta$ for each $P'(x)$. The maximum value is achieved by picking $j_i = \lfloor 1/d_i \rfloor$ for each $i = 1, 2, \cdots, \Delta$. Observe that S is completely contained between polynomials $y = P'(x) - \lceil \frac{\Delta}{w} \rceil w$ and $y = P'(x) + \lceil \frac{\Delta}{w} \rceil w$. As a result, for each $P'(x)$ (i.e., each choice of indices j_i's), the square S is covered exactly once. This implies S is covered $t = \prod_{i=1}^{\Delta} \left(\lfloor \frac{1}{d_i} \rfloor - \lfloor \frac{1}{2d_i} \rfloor + 1 \right) = \Theta(1/(2^{\Delta} \prod_{i=1}^{\Delta} d_i))$ times, which is the number of choices we have for the indices $j_1, \cdots, j_{\Delta}$. Therefore, each point in S is covered by t polynomial slabs.

We set $d_i = c_1 \left(\frac{1}{Q(n)^{2/(\Delta(\Delta+1))} w^{2/(\Delta+1)}} \right)^i w$ for some sufficiently small constant c_1 such that S is covered by $t \ge Q(n)$ polynomial slabs. We then set $w = c_2 \frac{Q(n)}{n}$ for some suitable constant c_2 according to c_1 such that the total number of polynomial slabs we have generated is exactly n.

By Lemma 7.3.1 and a similar argument as in the proof of Theorem 7.3.1, the intersection area of any two polynomial slabs in our construction is bounded by

$$v \le O(\Delta) w (\Delta+1)^3 \left(\frac{w}{d_i} \right)^{\frac{1}{i}} = f(\Delta) \frac{Q(n)^{1+\frac{2}{\Delta}}}{n^{1+\frac{2}{\Delta+1}}},$$

where $f(\Delta) = O(1)$ is some value depending on Δ only and $i = 1, 2, \cdots, \Delta$ is the largest degree where the two polynomials differ in their coefficients. Since $t \ge Q(n)$, then according to Theorem 7.2.2,

$$S(n) = \Omega\left(\frac{t}{v}\right) = \Omega\left(\frac{n^{1+\frac{2}{\Delta+1}}}{Q(n)^{\frac{2}{\Delta}}} \right). \qquad \square$$

So for any data structure that solves the 2D polynomial slab stabbing problem using $S(n) = O(n)$ space, Theorem 7.3.2 implies that its query time must be $Q(n) = \Omega(n^{1-1/(\Delta+1)})$.

7.4 2D Annulus Reporting and Stabbing

2D Annulus Reporting

In this subsection, we show that any data structure that solves 2D annulus reporting with $\log^{O(1)} n$ query time must use $\overset{o}{\Omega}(n^3)$ space. Recall that an annulus is the region between two concentric circles and the *width* of the annulus is the difference between the radii of the two circles. In general, we show that if the query time is $Q(n) + O(k)$, then the data structure must use $\overset{o}{\Omega}(n^3/Q(n)^5)$ space. We will still use Chazelle's framework.

We first present a technical geometric lemma which upper bounds the intersection area of two 2D annuli. We will later use this lemma to show that with probability more than $1/2$, a random point set satisfies the first condition of Theorem 7.2.1.

Lemma 7.4.1. *Consider two annuli of width w with inner radii of r_1, r_2, where $w < r_1 \le r_2$ and $r_1, r_2 = \Theta(n)$ and $w = o(n)$ for a parameter $n > 0$. Let d be the distance between the centers of the two annuli. When $w \le d < r_2$, the intersection area of the two annuli is bounded by $O\left(wn\sqrt{\frac{w^2}{(g+w)d}}\right)$, where $g = \max\{r_1 - r_2 + d, 0\}$.*

The proof sketch. For the complete proof see Appendix 7.A. When $w \le d \le r_2 - r_1 + 2w$, the intersection region is contained in two triangle-like regions. We only bound the triangle-like region $\tilde{\triangle} PQR$ in the upper half annuli as shown in Figure 7.2. We can show that its area is asymptotically upper bounded by the product of its base length $|QR| = O(w)$ and its height h. We bound h by observing that $\frac{hd}{2}$ is the area of triangle $\triangle PO_1O_2$ but we can also obtain its area of using Heron's formula, given its three side lengths. This gives $h = O(n\sqrt{w/d})$. Since in this case $g \le 2w$, the intersection area is upper bounded by $O\left(wn\sqrt{\frac{w^2}{(g+w)d}}\right)$ as claimed.

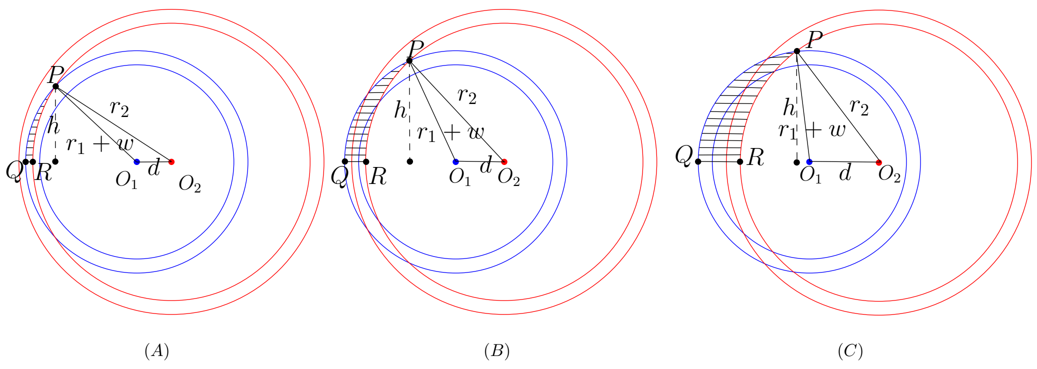
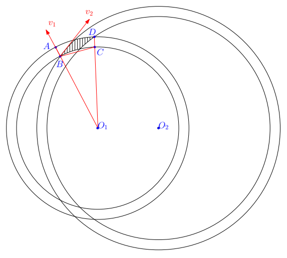
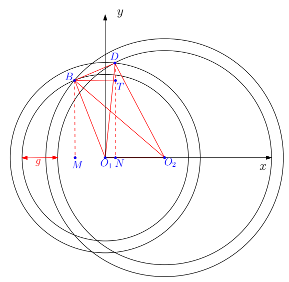


Figure 7.2: Intersections When d is Small

When $r_2 - r_1 + 2w \le d \le r_2$, the intersection region consists of two quadrilateral-like regions. See Figure 7.3a for an example. Again we only consider the quadrilateral-like region $\tilde{\square}$ (the shaded region) in the upper half of the annuli, which is contained in a partial annulus, $\tilde{\mathscr{R}}_{ABCD}$.

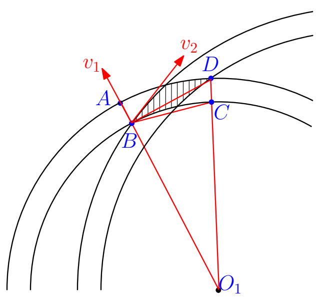


(a) Cover an Intersection by A Partial Annulus

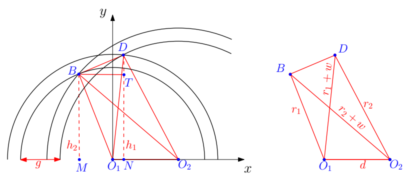


(b) Bound the Length of $|BD|$

Figure 7.3: Cover a Quadrilateral-like Region by a Partial Annulus



(a) Zoomed in Details of Figure 7.3a



(b) Zoomed in Details of Figure 7.3b

Figure 7.4: Zoomed in Details of Figure 7.3

We show the area of $\tilde{\mathscr{R}}_{ABCD}$ is asymptotically bounded by $|BC| \cdot w$, where $|BC|$ is the distance between the two endpoints of the inner arc. We upper bound $|BC|$ by $|BD|$. We use the algebraic representation of the two annuli, to bound the length of the projection of BD on the X-axis by $\Theta\left(\frac{wn}{d}\right)$; See Figure 7.3b. We use Heron's formula to bound the length of the projection of BD on the Y-axis by $O\left(n\sqrt{w^2/dg}\right)$. The sum of the length of the two projections yields the claimed bound. $\square$

Remark 7.4.1. *It is possible to give a closed form solution to the intersection area. However, the closed form contains a difference between two square roots whose sizes are very close to each other and it is hard to approximate this value.*

We use Chazelle's framework to obtain a lower bound for 2D annulus reporting. Let S_1 and S_2 be two squares of side length n that are placed $10n$ distance apart and S_2 is directly to the left of S_1. We generate the annuli as follows. We divide S_1 into a $\frac{n}{T} \times \frac{n}{T}$ grid where each cell is a square of side length $T < n$. For each grid point, we construct a series of circles as follows. Let O be a grid point. The first circle generated for O must pass through a corner of S_2 and two horizontal sides of S_2, as shown in Figure 7.5. Then we create a series of circles centered at O by increasing the radius by increments of w, for some fixed $w < T$ and w, T are monotonically increasing in n

and $wT = o(n)$, as long as it does not intersect the left side of S_2. Every consecutive two circles define an annulus centered at O. We repeat this for every grid point in S_1 and this makes up our set of queries. The input points are placed uniformly randomly inside S_2.

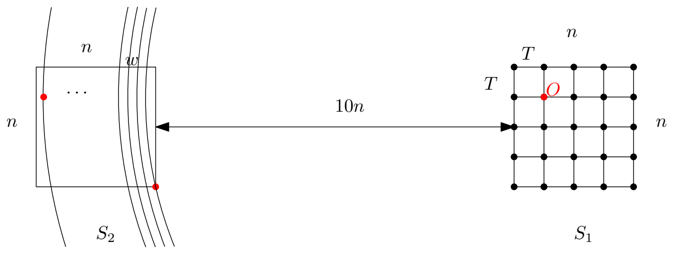


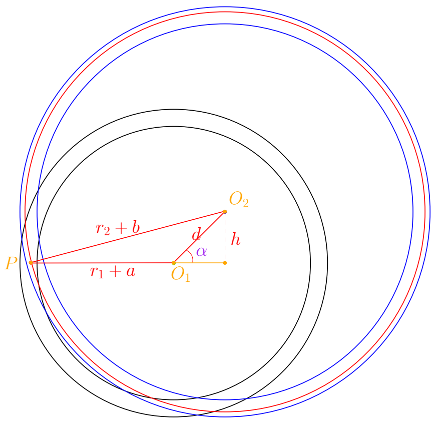
Figure 7.5: Generate a Family of Annuli at Point O

We now show that for the annuli we constructed, the intersection of ℓ annuli is not too large, for some ℓ we specify later. More precisely we prove the following.

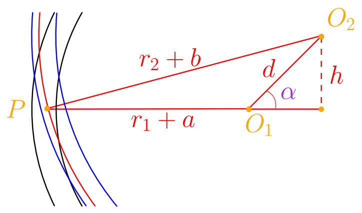
Lemma 7.4.2. *In our construction, there exists a large enough constant c such that in any subset of $\ell = cw^2/\sqrt{T}$ annuli, we can find two annuli such that their intersection has area $O\left(nw\sqrt{\frac{1}{T}}\right)$.*

The proof sketch. For the complete proof see Appendix 7.B. Let $\mathscr{S}$ be a set of $\ell = cw^2/\sqrt{T}$ annuli. Suppose for the sake of contradiction that we cannot find two annuli in $\mathscr{S}$ whose intersection area is $O\left(nw\sqrt{\frac{1}{T}}\right)$, i.e., any two annuli inter with area $\omega\left(nw\sqrt{\frac{1}{T}}\right)$. Consider any two annuli from $\mathscr{S}$. Let the inner circle radii of the two annuli be r_1, r_2 s.t. $r_1 \leq r_2$ and their widths be w. By our construction, $w < T$, $r_1, r_2 = \Theta(n)$. Since w, T are monotonically increasing in n and $wT = o(n)$, for any positive constant c', when n is big enough, we have $w < c'wT < r_2$. Then by Lemma 7.4.1, for any constant c', the intersection area of any two annuli in our construction with distance $c'wT$ between their centers is $O\left(nw\sqrt{\frac{1}{T}}\right)$ for large enough n. Therefore, the maximum distance between any two annuli in $\mathscr{S}$ must be $o(wT)$.

Let P be a point in the intersection of annuli in $\mathscr{S}$. Consider an arbitrary annulus $\mathscr{R}_1 \in \mathscr{S}$ centered at O_1 and another annulus $\mathscr{R}_2 \in \mathscr{S}$ centered at O_2 for some $O_2 \notin PO_1$. For $\mathscr{R}_1, \mathscr{R}_2$ to contain P, we must have $|PO_1| = r_1 + a, |PO_2| = r_2 + b$ for $0 \leq a, b \leq w$. See Figure 7.6 for an example. Also $|O_1O_2| = d$, by exploiting the shape of $\triangle PO_1O_2$ and applying Lemma 7.4.1, we can compute an upper bound for the distance between O_2 and PO_1, namely, $h = d \sin \alpha = o(w\sqrt{T})$, where α is the angle between O_1O_2 and PO_1. This implies that $\mathscr{S}$ must fit in a rectangle of size $o(wT) \times o(w\sqrt{T})$. Since the grid cell size is $T \times T$, only $o(w^2/\sqrt{T})$ annuli are contained in such a rectangle, a contradiction. $\square$



(a) Overview



(b) Zoomed in Details of Figure 7.6a

Figure 7.6: Intersection of Two Annuli

We are now ready to plug in some parameters in our construction. We set $T = w^2 2^{2\sqrt{\log n}}$. First, we claim that from each grid point O, we can draw $\Theta(n/w)$ circles; Let C_1, C_2, C_3, and C_4 be the corners of S_2 sorted increasingly according to their distance to O. As S_1 and S_2 are placed $10n$ distance apart, an elementary geometric calculation reveals that C_1 and C_2 are vertices of the right edge of S_2, meaning, the smallest circle that we draw from O passes through C_2 and we keep drawing circles, by incrementing their radii by w until we are about to draw a circle that is about to contain C_3. We can see that $|OC_3| - |OC_2| = \Theta(n)$ and thus we draw $\Theta(n/w)$ circles from O. As we have $\Theta((n/T)^2)$ grid cells, it thus follows that we have $\Theta(n^3/(T^2 w))$ annuli in our construction.

Also by our construction, the area of each annulus within S_2 is $\Theta(wn)$. To see this, let P be an arbitrary point in S_1, let A, B be the intersections of some circle centered at P as in Figure 7.7.

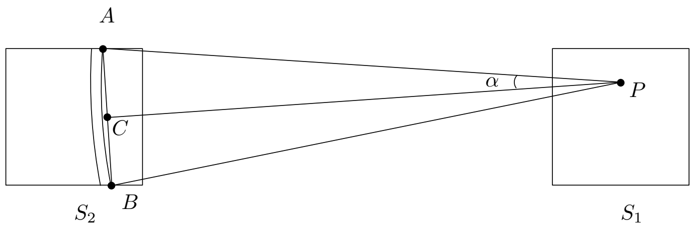


Figure 7.7: The Angle of an Annulus

We connect AB and let C be the center of AB. Let $\alpha = \angle APC$. In the triangle $\triangle ABP$, all the sides are within constant factors of each other and thus $\alpha = \Theta(1)$ and so the area of the annulus inside S_2 is at least a constant fraction of the area of the entire annulus.

Suppose we have a data structure that answers 2D annulus reporting queries in

$Q(n)+O(k)$ time. We set $w = c''Q(n)$ for a large enough constant c'' such that the area of each annulus within S_2 is at least $\Theta(wn) > 8nQ(n)$. Since we have shown that we have $\Theta(n^3/(T^2w)) = O(n^3)$ annuli in our construction, by Lemma 7.2.2, if we sample n points uniformly at random in S_2, then with probability more than $1/2$, each annulus contains at least $Q(n)$ points.

Also by our construction, the total number of intersections of l annuli is upper bounded by that of two annuli, which is bounded by $O(n^6)$ and by our choice of T, $O\left(nw\sqrt{\frac{1}{T}}\right) = O\left(\frac{n}{2^{\sqrt{\log n}}}\right)$. Then by Lemma 7.2.1 and Lemma 7.4.2, with probability $> \frac{1}{2}$, a point set of size n picked uniformly at random in S_2 satisfies that the number of points in any of the intersection of $cw^2/\sqrt{T}$ annuli is no more than $9\sqrt{\log n}$.

Now by union bound, there exist $\Theta\left(\frac{n^3}{wT^2}\right)$ point sets such that each set is the output of some 2D annulus query and each set contains at least $Q(n)$ points. Furthermore, the intersection of any $cw^2/\sqrt{T}$ sets is bounded by $9\sqrt{\log n}$. Then by Theorem 7.2.1, we obtain a lower bound of

$$S(n) = \Omega\left(\frac{Q(n)n^3\sqrt{T}}{wT^2w^2 2^{O(\sqrt{\log n})}}\right) = \overset{o}{\Omega}\left(\frac{n^3}{Q(n)^5}\right).$$

This proves the following theorem about 2D annulus reporting.

Theorem 7.4.1. *Any data structure that solves 2D annulus reporting on point set of size n with query time $Q(n)+O(k)$, where k is the output size, must use $\overset{o}{\Omega}\left(n^3/Q(n)^5\right)$ space.*

So for any data structure that solves 2D annulus reporting in time $Q(n) = \log^{O(1)} n$, Theorem 7.4.1 implies that $\overset{o}{\Omega}\left(n^3\right)$ space must be used.

2D Annulus Stabbing

Modifications similar to those done in Subsection 7.3 can be used to obtain the following lower bound.

Theorem 7.4.2. *Any data structure that solves the 2D annulus stabbing problem with query time $Q(n)+O(k)$, where k is the output size, must use $S(n) = \Omega(n^{3/2}/Q(n)^{3/4})$ space.*

Proof. We use Afshani's lower bound framework as in Theorem 7.2.2. We construct annuli similar to the way as we did in the proof of Theorem 7.4.1, but with some differences. Let S_1 be a unit square, we decompose S_1 into a grid where each grid cell is a square of size $T \times T$, for some parameter T to be determined later. Let S_2 be another unit square to the left of S_1 with distance 10. For each grid point in S_1, we generate a family of circles centered at the point where the first circle is the first one tangent to the right side of S_2. See Figure 7.8 for an example. Let r_0 be the radius of this circle. We generate other circles by increasing the radius by w each time. The last

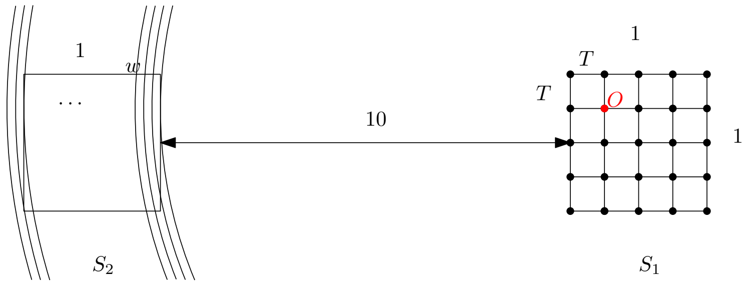


Figure 7.8: Generate a Family of Annuli at Point O

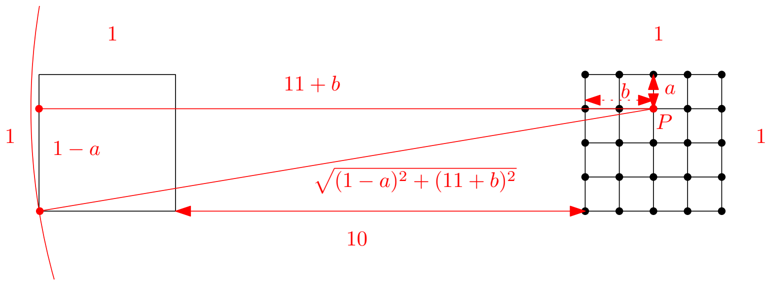


Figure 7.9: The Last Valid Circle

circle is the one with radius at least $r_0 + (\sqrt{122} - 9)$. The choice of constant $\sqrt{122} - 9$ is due to the following. Consider a point P in the upper half of S_1, let a (resp. b) be the distance from P to the upper (resp. left) side of S_1. The case for the lower half of S_1 is symmetric. Consider the last circle intersecting S_2 only at the corners. See Figure 7.9 for an example. The difference of radius lengths between the last circle and the first circle is

$$f = \sqrt{(1-a)^2 + (11+b)^2} - 10 - b.$$

Simple analysis shows that $f \leq \sqrt{122} - 10$. This shows as long as the last circle has radius at least $\sqrt{122} - 9 + r_0$, it is completely outside of S_2. Like we did previously, we consider the region between two consecutive annuli to be an annulus. Note that by our construction, for any point in S_2, it is contained in one of the annuli we generated for a given grid point. The total number of annuli we have generated is therefore

$$\left(\frac{1}{T} + 1\right)^2 \cdot \frac{\sqrt{122} - 9}{w}.$$

We set $T = 1/(2\sqrt{Q(n)} - 1)$ and $w = 4(\sqrt{122} - 9)Q(n)/n$. Then the total number of annuli we have generated is n. Furthermore, each query point is contained in $t = (1/T + 1)^2 \geq Q(n)$ annuli.

By setting $w \leq T$, which implies $Q(n) = O(n^{2/3})$, we can use Lemma 7.4.1 to upper bound the intersection area of two annuli in our construction by $v = O(w\sqrt{\frac{w}{T}}) =$

$O(Q(n)^{7/4}/n^{3/2})$. Then by Theorem 7.2.2,

$$S(n) = \Omega\left(\frac{t}{v}\right) = \Omega\left(\frac{n^{3/2}}{Q(n)^{3/4}}\right). \qquad \square$$

So for any data structure that solves the 2D annulus stabbing problem using $O(n)$ space, Theorem 7.4.2 implies that its query time must be $Q(n) = \Omega(n^{2/3})$.

7.5 Conclusion and Open Problems

We investigated lower bounds for range searching with polynomial slabs and annuli in $\mathbb{R}^2$. We showed space-time tradeoff bounds of $S(n) = \overset{o}{\Omega}(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2})$ and $S(n) = \overset{o}{\Omega}(n^3/Q(n)^5)$ for them respectively. Both of these bounds are almost tight in the "fast query" case, i.e., when $Q(n) = \log^{O(1)} n$ (up to a $n^{o(1)}$ factor). This refutes the conjecture of the existence of a data structure that can solve semialgebraic range searching in $\mathbb{R}^d$ using $\overset{o}{O}(n^d)$ space and $\log^{O(1)} n$ query time. We also studied the "dual" polynomial slab stabbing and annulus stabbing problems. For these two problems, we obtained lower bounds $S(n) = \Omega(n^{1+2/(\Delta+1)}/Q(n)^{2/\Delta})$ and $S(n) = \Omega(n^{3/2}/Q(n)^{3/4})$ respectively. These bounds are tight when $S(n) = O(n)$. Our work, however, brings out some very interesting open problems.

To get the lower bounds for the polynomial slabs, we only considered univariate polynomials of degree Δ. In this setting, the number of coefficients is at most $\Delta + 1$, and we have also assumed they are all independent. It would be interesting to see if similar lower bounds can be obtained under more general settings. In particular, as the maximum number of coefficients of a bivariate polynomial of degree Δ is $\binom{\Delta+2}{2}$, it would interesting to see if a $\overset{o}{\Omega}(n^{\binom{\Delta+2}{2}-1})$ space lower bound can be obtained for the "fast query" case.

It would also be interesting to consider space-time trade-offs. For instance, by combining the known "fast query" and "low space" solutions for 2D annulus reporting, one can obtain data structures with trade-off curve $S(n) = \tilde{O}(n^3/Q(n)^4)$, however, our lower bound is $S(n) = \overset{o}{\Omega}(n^3/Q(n)^5)$ and it is not clear which of these bounds is closer to the optimal bound. For the annulus searching problem in $\mathbb{R}^2$, in our lower bound proof, we considered a random input point set, since in most cases a random point set is the hardest input instance and our analysis seems to be tight. Therefore, we conjecture that our lower bound could be tight, at least when $Q(n)$ is small enough. We believe that it should be possible to obtain the trade-off curve of $S(n) = \tilde{O}(n^3/Q(n)^5)$ when the input points are uniformly random in the unit square and $Q(n)$ is not too big.

Finally, another interesting direction is to study the lower bound for the counting variant of semialgebraic range searching.

Acknowledgement: The authors would like to thank Esther Ezra for sparking the initial ideas behind the proof.

Appendices

7.A Proof of Lemma 7.4.1

Lemma 7.4.1. *Consider two annuli of width w with inner radii of r_1, r_2, where $w < r_1 \leq r_2$ and $r_1, r_2 = \Theta(n)$ and $w = o(n)$ for a parameter $n > 0$. Let d be the distance between the centers of the two annuli. When $w \leq d < r_2$, the intersection area of the two annuli is bounded by $O\left(wn\sqrt{\frac{w^2}{(g+w)d}}\right)$, where $g = \max\{r_1 - r_2 + d, 0\}$.*

Proof. We first prove the lemma for when $w \leq d \leq r_2 - r_1 + 2w$, observe that $g \leq 2w$ and the intersections are all bounded by two triangle-like regions. We consider the triangle-like region $\tilde{\triangle}PQR$ in the upper half of the annuli as shown in Figure 7.A.1. The area of the other triangle-like region can be bounded symmetrically. When $0 \leq r_2 - r_1 < 2w$, we can trivially bound the area by $O(nw) = O(nw\frac{w}{d}) = O(nw\sqrt{w^2/(g+w)d})$. So we only focus on the case $r_2 - r_1 \geq 2w$. Note that for two circles to intersect, we must have $d \geq r_2 - r_1 - w$.

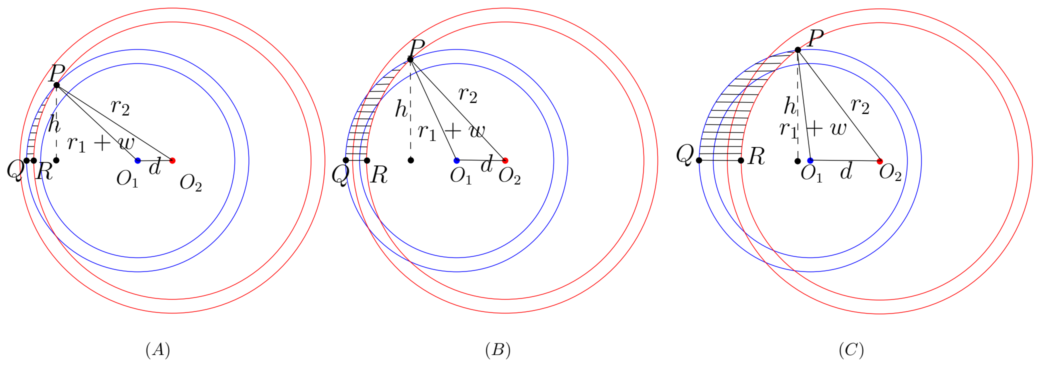


Figure 7.A.1: Intersections When d is Small

To upper bound its area, we first compute its "height" h. Consider $\triangle PO_1O_2$, by Heron's Formula, its area is

$$\begin{aligned}
A_{\triangle PO_1O_2} &= \sqrt{\frac{r_1+r_2+w+d}{2}\cdot\frac{r_2+d-r_1-w}{2}\cdot\frac{r_1+d+w-r_2}{2}\cdot\frac{r_1+r_2+w-d}{2}} \\
&= O(\sqrt{r_2\cdot d\cdot w\cdot r_1}) \\
&= O\left(n\sqrt{dw}\right),
\end{aligned}$$

where the second equality follows from

$$\begin{aligned}
d \le r_2-r_1+2w &\implies r_1+r_2+w+d \le 2r_2+3w \le 5r_2, \\
r_2-r_1-w \le d &\implies r_2+d-r_1-w \le 2d, \\
d \le r_2-r_1+2w &\implies r_1+d-r_2+w \le 3w, \\
r_2-r_1-w \le d &\implies r_1+r_2+w-d \le 2r_1+2w \le 4r_1.
\end{aligned}$$

Then we bound h by

$$h = \frac{2A_{\triangle PO_1O_2}}{d} = O\left(n\sqrt{\frac{w}{d}}\right).$$

To bound the area of $\tilde{\triangle}PQR$, we move the outer circle of the annulus centered at O_1 along O_1O_2 such that it passes R as in Figure 7.A.2. Let S be a point in this new circle such that $PS \parallel QR$. The area of region $PSRQ$ formed by the outer circle of the annulus centered at O_1 and the new circle, as well as PS and QR, i.e., the shaded region in Figure 7.A.2, is $O(wh)$ by a simple integral argument. Since $r_2 \ge r_1 + 2w$ and $w = o(r_1)$, the x-coordinate of P is smaller than that of O_1 and the curvature of the inner circle of the annulus centered at O_2 is no larger than that of the outer circle of O_1, hence $\tilde{\triangle}PQR$ is contained in this region. So $A_{\tilde{\triangle}PQR} = O(wh) = O(nw\sqrt{w/d}) = O(nw\sqrt{w^2/(g+w)d})$.

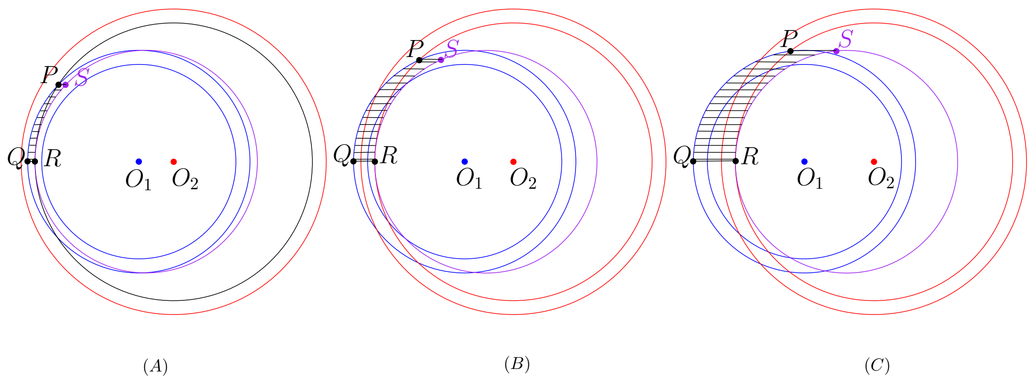


Figure 7.A.2: Bound the Intersection Area for Small d

In the case where $r_2 - r_1 + 2w < d < r_2$, the intersections consist of two quadrilateral-like regions, one at the upper half and the other at the lower half. In this case, $g > 2w$.

We show how to bound the area of the quadrilateral-like region at the upper half, the other one can be bounded symmetrically.

First note that each quadrilateral-like region $\tilde{\Box}$ (the shaded region) is contained in a partial annulus as in Figure 7.A.3. We generate this partial annulus by shooting a ray l_l from the center O_1 such that $\tilde{\Box}$ is completely to the right of this ray, to create the left boundary AB, where A, B are the intersection of the ray and the outer and inner circles of the annulus $\mathscr{R}_1$ centered at O_1 respectively. And similarly, shoot another ray l_r to create the right boundary CD. Note that $\tilde{\Box}$ can only intersect the left ray at point B and the right ray at point D. To see this, consider the line l_t tangent to circle $(O_2, r_2 + w)$ at point B. Let $B = (x, y)$, it is easy to compute that the slope of l_t is $\frac{d-x}{y}$. Consider a vector $\vec{v}_1 = (x, y)$ in the direction of ray O_1B and vector $\vec{v}_2 = (\frac{y}{d-x}, 1)$ of l_t pointing upwards. If v_1 and v_2 form a right turn, $\tilde{\Box}$ is completely to the right of O_1B, i.e., l_l intersects $\tilde{\Box}$ at B. The cross product of $\vec{v}_1$ and $\vec{v}_2$ is easily computed to be $(x, y) \times (\frac{y}{d-x}, 1) = \frac{dx-(x^2+y^2)}{d-x}$. Since B is the intersection of circle (O_1, r_1) and $(O_2, r_2 + w)$, it must be a solution to the system

$$\begin{cases} x^2 + y^2 = r_1^2 \\ (x-d)^2 + y^2 = (r_2 + w)^2 \end{cases},$$

which implies that

$$2dx = d^2 + r_1^2 - (r_2 + w)^2,$$

and so

$$d - x = d - \frac{d^2 + r_1^2 - (r_2 + w)^2}{2d} = \frac{d^2 + (r_2 + w)^2 - r_1^2}{2d} > 0,$$

since $r_2 \geq r_1$ and $d > r_2 - r_1 + 2w > 0$. Thus

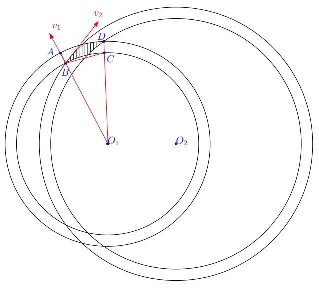
$$\frac{dx - (x^2 + y^2)}{d - x} = \frac{d^2 + r_1^2 - (r_2 + w)^2 - 2r_1^2}{2(d-x)} = \frac{d^2 - (r_1^2 + (r_2 + w)^2)}{2(d-x)} < \frac{r_2^2 - (r_1^2 + (r_2 + w)^2)}{2(d-x)} < 0,$$

where the second last inequality follows from $d < r_2$, and the last inequality follows from $d - x > 0$. So $\vec{v}_1$ and $\vec{v}_2$ form a right turn during this process. Thus l_l intersects $\tilde{\Box}$ at point B. Similarly, we can show that l_r intersects $\tilde{\Box}$ at point D. Now we bound the area of the partial annulus.

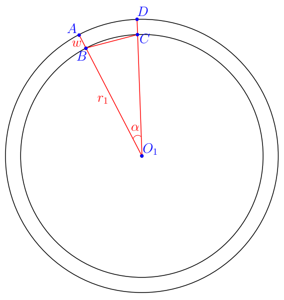
First, we observe that the area of the partial annulus is bounded by the product of the width of the annulus, w, and the length BC, within some constant factor.

To see this, consider the possible partial annulus generated by any two annuli, see Figure 7.A.3b for an example. One important observation is that for any intersection, the angle α of the partial annulus containing it is no more than π. The area of the partial annulus is clearly $\alpha(r_1 w + w^2/2) = \Theta(r_1 w \alpha)$. Note that for $0 \leq \alpha \leq \pi$, let $\beta = \alpha/2$, it is a simple fact that

$$\frac{1}{2}\beta \leq \sin\beta \leq \beta$$

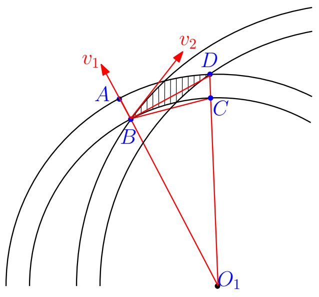


(a) Cover an Intersection by A Partial Annulus

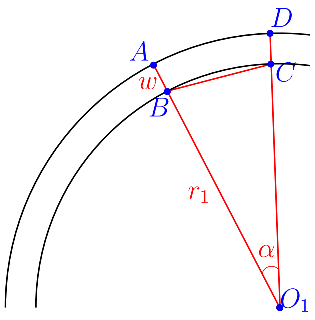


(b) A Partial Annulus Example

Figure 7.A.3: Cover a Quadrilateral-like Region by a Partial Annulus



(a) Zoomed Details of Figure 7.A.3a



(b) Zoomed in Details of Figure 7.A.3b

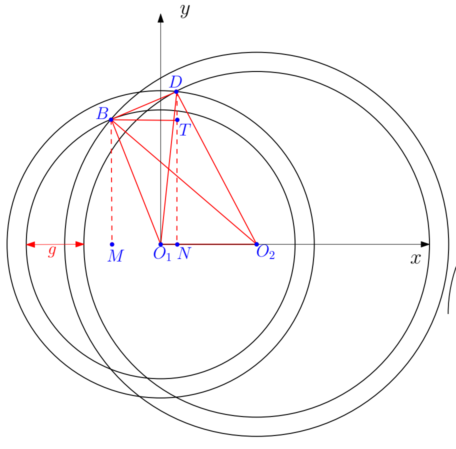
Figure 7.A.4: Zoomed in Details of Figure 7.A.3

for $0 \le \beta \le \pi/2$. So for β in this range, $\sin\beta = \Theta(\beta)$. Thus we can compute $|BC| = 2r_1 \sin\beta = \Theta(r_1\alpha)$, which implies that indeed $w \cdot |BC|$ gives us the area of the partial annulus, within some constant factor.

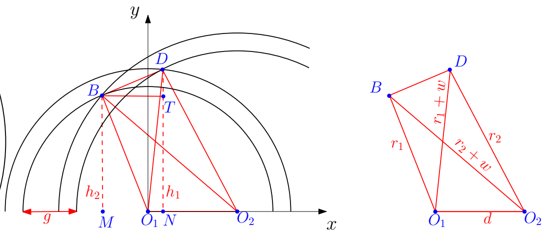
Thus in Figure 7.A.3a, the area of the partial annulus is $w \cdot |BC|$. Since by triangle inequality $|BC| \le |BD| + |CD| = |BD| + w$, we bound $|BD|$. By Lemma 7.A.1 we show below, we know $|BD| = O\left(n\sqrt{\frac{w^2}{dg}}\right)$. Therefore, the area of intersection is bounded by $O\left(wn\sqrt{\frac{w^2}{(g+w)d}} + w^2\right) = O\left(wn\sqrt{\frac{w^2}{(g+w)d}}\right)$ as claimed since $d, g = O(n)$. $\square$

Lemma 7.A.1. *In $\mathbb{R}^2$, given two annuli centered at O_1, O_2 of width w with inner radius lengths being r_1, r_2 respectively, where $w < r_1 \le r_2$ and $r_1, r_2 = \Theta(n)$ for a parameter $n > 0$. Let d be the distance between two centers of the annuli satisfying $r_2 - r_1 + 2w < d < r_2$ and let $g = r_1 + d - r_2 > 2w$ be the distance between two inner circles of two annuli along the direction of O_1O_2. Consider the quadrilateral-like region $\tilde{\square}$ formed by four arcs in Lemma 7.4.1, we have $|BD| = O\left(n\sqrt{\frac{w^2}{gd}}\right)$*

Proof. W.l.o.g., we assume that O_1 is the origin of the Cartesian coordinate system and O_2 is on the x-axis as shown in Figure 7.A.5.



(a) Overview



(b) Zoomed in Details of Figure 7.A.5a

Figure 7.A.5: Bound the Length of $|BD|$

Given any point P in the plane, we use x_P (resp. y_P) to denote its x (resp. y) coordinate. We first compute $|BT| = x_{BD} = x_D - x_B$, where x_{BD} is the length of BD along the x-axis. Since $B = (x_B, y_B)$ is a intersection of the circle centered at O_1 with radius r_1 and the one centered at O_2 with radius $r_2 + w$, B is a solution to the following system of equations

$$\begin{cases} x^2 + y^2 = r_1^2 \\ (x-d)^2 + y^2 = (r_2+w)^2. \end{cases}$$

So x_B satisfies

$$2dx - d^2 = r_1^2 - (r_2+w)^2,$$

which implies

$$x_B = \frac{r_1^2 - (r_2+w)^2 + d^2}{2d}. \tag{7.1}$$

Similarly, we obtain

$$x_D = \frac{(r_1+w)^2 - r_2^2 + d^2}{2d}. \tag{7.2}$$

So by (7.1) and (7.2), we obtain

$$|BT| = x_{BD} = x_D - x_B = \frac{w(r_1+r_2+w)}{d} = \Theta\left(\frac{wn}{d}\right). \tag{7.3}$$

Now we compute $|DT| = y_{BD} = h_1 - h_2$. Let μ_1 be the area of $\triangle DO_1O_2$ and μ_2 be the area of $\triangle BO_1O_2$.

By Heron's formula,

$$\begin{cases} \mu_1 = \sqrt{\frac{r_1+w+r_2+d}{2} \cdot \frac{r_2+d-r_1-w}{2} \cdot \frac{r_1+w+d-r_2}{2} \cdot \frac{r_1+w+r_2-d}{2}} \\ \mu_2 = \sqrt{\frac{r_1+r_2+w+d}{2} \cdot \frac{r_2+w+d-r_1}{2} \cdot \frac{r_1+d-r_2-w}{2} \cdot \frac{r_1+r_2+w-d}{2}}. \end{cases}$$

So

$$\frac{\mu_1}{\mu_2} = \sqrt{\frac{(r_2+d-r_1-w)(r_1+w+d-r_2)}{(r_2+w+d-r_1)(r_1+d-r_2-w)}}.$$

We define two variables $X = g-w$ and $Y = 2d-g-w$ to simplify notations. Recall that $g = r_1+d-r_2 > 2w$ and $d-g = r_2-r_1 \geq 0$ and so

$$\frac{\mu_1}{\mu_2} = \sqrt{\frac{(2d-g-w)(g+w)}{(2d-g+w)(g-w)}} = \sqrt{\frac{(X+2w)Y}{X(Y+2w)}} = \sqrt{1+\frac{2w(Y-X)}{XY+2wX}} = \sqrt{1+\frac{2w(Y-X)}{X(Y+2w)}}$$
$$\leq \sqrt{1+\frac{2w}{X}} = \sqrt{1+O\left(\frac{w}{g}\right)},$$

where the last inequality and equality follow from $w > 0$, $X = g-w = \Theta(g) > 0$, and $Y - X = 2d-g-w-(g-w) = 2(d-g) \geq 0$. Note that

$$\begin{cases} \mu_1 = \frac{h_1 d}{2} \\ \mu_2 = \frac{h_2 d}{2} \end{cases}.$$

Since $2w < g$, we therefore have

$$\frac{h_1}{h_2} = \frac{\mu_1}{\mu_2} = \sqrt{1+O\left(\frac{w}{g}\right)} = 1+O\left(\frac{w}{g}\right) \implies \frac{h_2}{h_1} = 1-O\left(\frac{w}{g}\right).$$

By applying Heron's formula for $\triangle DO_1O_2$, we obtain

$$h_1 = O\left(n\sqrt{\frac{w}{d}}\right) = O\left(n\sqrt{\frac{g}{d}}\right).$$

So

$$|DT| = y_{BD} = h_1 - h_2 = h_1\left(1-\frac{h_2}{h_1}\right) = h_1 \cdot O\left(\frac{w}{g}\right) = O\left(n\sqrt{\frac{w^2}{dg}}\right). \tag{7.4}$$

Therefore, by Equation 7.3 and Equation 7.4, and $g = r_1+d-r_2 \leq d$, we have

$$|BD| \leq |BT| + |DT| = O\left(n\sqrt{\frac{w^2}{dg}}\right). \qquad \square$$

7.B Proof of Lemma 7.4.2

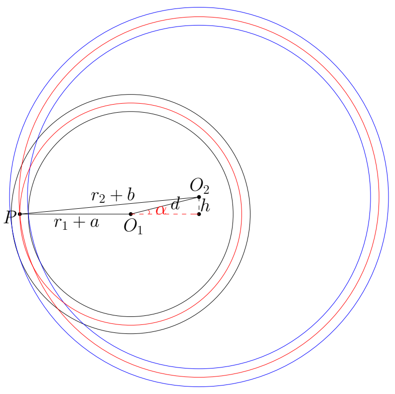
Lemma 7.4.2. *In our construction, there exists a large enough constant c such that in any subset of $\ell = cw^2/\sqrt{T}$ annuli, we can find two annuli such that their intersection has area $O\left(nw\sqrt{\frac{1}{T}}\right)$.*

Proof. We consider for the sake of contradiction that in our construction for every set $\mathscr{S}$ of $l = cw^2/\sqrt{T}$ annuli for any positive constant c, we cannot find two annuli in $\mathscr{S}$ such that their intersection area is $O(nw\sqrt{1/T})$.

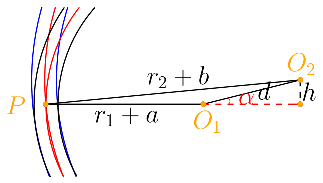
First observe that by our construction, any two annuli from the same family have zero intersection area. Also the distance between the centers of any two annuli is less than the radius of any annulus. Furthermore, the width of an annulus is always less than the distance between the centers of any two annuli. So by Lemma 7.4.1, the intersection area of any two annuli in our construction with center distance $\Omega(wT)$ is $O(nw\sqrt{1/T})$. So for our assumption to hold, the distance between the centers of any two annuli in $\mathscr{S}$ is $o(wT)$.

Let P be a point in the intersection of $\mathscr{S}$, then P is contained in every annulus in $\mathscr{S}$. Now consider an arbitrary annulus $\mathscr{R}_1 \in \mathscr{S}$ centered at O_1 and another annulus $\mathscr{R}_2 \in \mathscr{S}$ centered at O_2 for some O_2 not in line PO_1. Connect PO_1 and PO_2, for $\mathscr{R}_1, \mathscr{R}_2$ to contain P, we must have $|PO_1| = r_1 + a$ and $|PO_2| = r_2 + b$ for $0 \le a, b \le w$ as shown in Figure 7.B.1.

We first consider the case when the distance between the centers of $\mathscr{R}_1$ and $\mathscr{R}_2$ is no more than $r_2 - r_1 + 2w$. In this case, their intersection area is upper bounded by $O(nw\sqrt{w/d})$ according to Lemma 7.4.1.



(a) Overview



(b) Zoomed in Details of Figure 7.B.1a

Figure 7.B.1: Intersection of Two Annuli When Distance Between Centers is Small

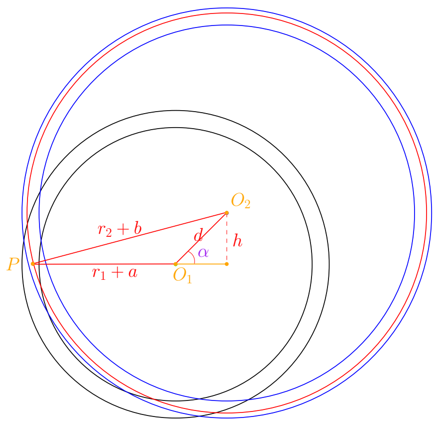
By Heron's formula,

$$\begin{aligned} Area_{\triangle PO_1O_2} &= \sqrt{\frac{r_1+a+r_2+b+d}{2} \cdot \frac{r_2-r_1+b-a+d}{2} \cdot \frac{r_1+a-r_2-b+d}{2} \cdot \frac{r_1+a+r_2+b-d}{2}} \\ &= O(r_1\sqrt{dw}) \\ &= \Theta(r_1 h), \end{aligned}$$

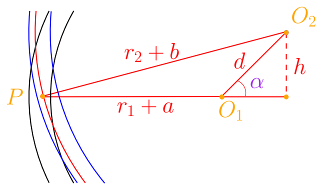
where the second equality follows from $a, b \le w \le d$ and $r_2 - r_1 - w \le d \le r_2 - r_1 + 2w$, and h is the distance between O_2 and line PO_1. This implies $h = O(\sqrt{dw})$. Since $d = o(wT)$, We obtain that

$$h = o(w\sqrt{T}). \tag{7.5}$$

Now we consider the case when the distance between the centers of $\mathscr{R}_1$ and $\mathscr{R}_2$ is more than $r_2 - r_1 + 2w$. See Figure 7.B.2 for an example.



(a) Overview



(b) Zoomed in Details of Figure 7.B.2

Figure 7.B.2: Intersection of Two Annuli When Distance Between Centers is Large

Let $g = r_1 + d - r_2$ be the distance between the inner circle of $\mathscr{R}_1$ and the inner circle of $\mathscr{R}_2$. Let α be the angle between O_1O_2 abd PO_1. Note that $\alpha > 0$ since O_2 is not in PO_1. W.l.o.g., we assume $0 < \alpha \leq \pi/2$. The situations for other values of α are symmetric. Then

$$\cos(\pi - \alpha) = \frac{(r_1 + a)^2 + d^2 - (r_2 + b)^2}{2d(r_1 + a)} \tag{7.6}$$

$$\implies -\cos\alpha = \frac{(r_1 + a)^2 + d^2 - (r_2 + b)^2}{2d(r_1 + a)} \tag{7.7}$$

$$\implies 1 - \cos\alpha = \frac{(r_1 + a + d)^2 - (r_2 + b)^2}{2d(r_1 + a)} \tag{7.8}$$

$$\implies 1 - \cos\alpha = \frac{(r_1 + a + d - r_2 - b)(r_1 + r_2 + a + b + d)}{2d(r_1 + a)} \tag{7.9}$$

$$\implies 1 - \cos\alpha = \frac{(g + a - b)(r_1 + r_2 + a + b + d)}{2d(r_1 + a)} \tag{7.10}$$

$$\implies g = \Theta(d(1 - \cos\alpha)), \tag{7.11}$$

where the second last implication follows from $g = r_1 + d - r_2$ and the last implication follows from the fact that in our construction $a, b \leq w$, $2w \leq g = r_1 + d - r_2$, $r_2 - r_1 + 2w \leq d \leq r_1 + r_2$, and $r_1 + r_2 = \Theta(n)$.

So according to Lemma 7.4.1 the intersection area of $\mathscr{R}_1, \mathscr{R}_2$ is upper bounded by

$$A = O\left(wn\sqrt{\frac{w^2}{dg}}\right).$$

Let $A = \omega\left(nw\sqrt{\frac{1}{T}}\right)$, by equation (7.11), we get

$$d^2(1-\cos\alpha) = o(w^2T), \tag{7.12}$$

Since $d = \frac{h}{\sin\alpha} = \frac{h}{\sqrt{1-\cos^2\alpha}}$, and $0 < \alpha \le \pi/2$, we plug d in inequality (7.12) and obtain

$$\frac{h^2}{1-\cos^2\alpha}(1-\cos\alpha) = \frac{h^2}{1+\cos\alpha} = o(w^2T).$$

This implies

$$h = o\left(\sqrt{w^2T(1+\cos\alpha)}\right) = o\left(w\sqrt{T}\right). \tag{7.13}$$

So in order to have no two annuli intersecting with area $O(nw\sqrt{1/T})$, the distance between the centers of annuli in $\mathscr{S}$ and PO_1 must be $o(w\sqrt{T})$. We have already shown that the distance between any two centers is $o(wT)$. Together they imply that the centers of $\mathscr{S}$ fit in a rectangle of shape $o(wT) \times o(w\sqrt{T})$. However, in our construction, each grid cell is of size $T \times T$, this implies we only have $o(w^2/\sqrt{T})$ centers in the rectangle. But we should have $cw^2/\sqrt{T}$ centers, a contradiction. $\square$

Chapter 8

On Semialgebraic Range Reporting

Abstract

Semialgebraic range searching, arguably the most general version of range searching, is a fundamental problem in computational geometry. In the problem, we are to preprocess a set of points in $\mathbb{R}^D$ such that the subset of points inside a semialgebraic region described by a constant number of polynomial inequalities of degree Δ can be found efficiently.

Relatively recently, several major advances were made on this problem. Using algebraic techniques, "near-linear space" data structures [AMS13, MP15] with almost optimal query time of $Q(n) = O(n^{1-1/D+o(1)})$ were obtained. For "fast query" data structures (i.e., when $Q(n) = n^{o(1)}$), it was conjectured that a similar improvement is possible, i.e., it is possible to achieve space $S(n) = O(n^{D+o(1)})$. The conjecture was refuted very recently by Afshani and Cheng [AC23b]. In the plane, i.e., $D = 2$, they proved that $S(n) = \Omega(n^{\Delta+1-o(1)}/Q(n)^{(\Delta+3)\Delta/2})$ which shows $\Omega(n^{\Delta+1-o(1)})$ space is needed for $Q(n) = n^{o(1)}$. While this refutes the conjecture, it still leaves a number of unresolved issues: the lower bound only works in 2D and for fast queries, and neither the exponent of n nor $Q(n)$ seems to be tight even for $D = 2$, as the best known upper bounds have $S(n) = O(n^{\mathbf{m}+o(1)}/Q(n)^{(\mathbf{m}-1)D/(D-1)})$ where $\mathbf{m} = \binom{D+\Delta}{D} - 1 = \Omega(\Delta^D)$ is the maximum number of parameters to define a monic degree-Δ D-variate polynomial, for any constant dimension D and degree Δ.

In this paper, we resolve two of the issues: we prove a lower bound in D-dimensions, for constant D, and show that when the query time is $n^{o(1)} + O(k)$, the space usage is $\Omega(n^{\mathbf{m}-o(1)})$, which almost matches the $\tilde{O}(n^{\mathbf{m}})$ upper bound and essentially closes the problem for the fast-query case, as far as the exponent of n is considered in the pointer machine model. When considering the exponent of $Q(n)$, we show that the analysis in [AC23b] is tight for $D = 2$, by presenting matching upper bounds for uniform random point sets. This shows either the existing upper bounds can be improved or to obtain better lower bounds a new fundamentally different input set needs to be constructed.

8.1 Introduction

In the classical semialgebraic range searching problem, we are to preprocess a set of n points in $\mathbb{R}^D$ such that the subset of points inside a semialgebraic region, described by a constant number of polynomial inequalities of degree Δ can be found efficiently. Recently, two major advances were made on this problem. First, in 2019, Agarwal et al. [AAEZ21] showed for polylogarithmic query time, it is possible to build a data structure of size $\tilde{O}(n^\beta)$ space[1], where β is the number of parameters needed to specify a query polynomial. For example, for $D = 2$, a query polynomial is in the form of $\sum_{i+j\le\Delta} a_{ij}x^i y^j \le 0$ where a_{ij}'s are specified at the query time, and when $\Delta = 4$, β can be as large as 14 (technically, there are 15 coefficients but one coefficient can always be normalized to be 1). In this case, a major conjecture was that if this space bound could be improved to $\tilde{O}(n^D)$ (e.g., for $\Delta = 4$, from $\tilde{O}(n^{14})$ to $\tilde{O}(n^2)$). Very recently, Afshani and Cheng [AC23b] refuted this conjecture by showing an $\overset{o}{\Omega}(n^{\Delta+1})$ lower bound. However, there are two major limitations of their lower bound. First, their lower bound only works in $\mathbb{R}^2$, while the upper bound in [AAEZ21] holds for all dimensions. Second, their lower bound only works for queries of form $y - \sum_{i=0}^{\Delta} x^i \le 0$ and thus their lower bound does not give a satisfactory answer to the problem in the general case. For example, for $D = 2, \Delta = 4$, they show a $\overset{o}{\Omega}(n^5)$ lower bound whereas the current best upper bound is $\tilde{O}(n^{14})$. In general, their space lower bound is at most $\overset{o}{\Omega}(n^{\Delta+1})$ while the upper bound of [AAEZ21] can be $\tilde{O}(n^{\Theta(\Delta^2)})$, which leaves an unsolved wide gap, even for $D = 2$. Another problem brought by [AAEZ21] is the space-time tradeoff. When restricted to queries of the form $y - \sum_{i=0}^{\Delta} x^i \le 0$, the current upper bound tradeoff is $S(n) = \tilde{O}(n^{\Delta+1}/Q(n)^{2\Delta})$ [MP15, AAEZ21] while the lower bound in [AC23b] is $S(n) = \overset{o}{\Omega}(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2})$. Even for $\Delta = 2$, we observe a discrepancy between an $S(n) = \tilde{O}(n^3/Q(n)^4)$ upper and an $S(n) = \overset{o}{\Omega}(n^3/Q(n)^5)$ lower bound.

Here, we make progress in both lower and upper bound directions. We give a general lower bound in D dimensions that is tight for all possible values of β. Our lower bound attains the maximum possible β value $\mathbf{m}_{D,\Delta} = \binom{D+\Delta}{D} - 1$, e.g., $\overset{o}{\Omega}(n^{14})$ for $D = 2, \Delta = 4$. Thus, our lower bounds almost completely settle the general case of the problem for the fast-query case, as far as the exponent of n is concerned. This improvement is quite non-trivial and requires significant new insights that are not avaiable in [AC23b]. For the upper bound, we present a matching space-time tradeoff for the two problems studied in [AC23b] for uniform random point sets. This shows their lower bound analysis is tight. Since for most range searching problems, a uniform random input instance is the hardest one, our results show that current upper bound based on the classical method might not be optimal. We develop a set of new ideas for our results which we believe are important for further investigation of this problem.

[1] $\tilde{\Omega}(\cdot), \tilde{O}(\cdot), \tilde{\Theta}(\cdot)$ notations hide $\log^{o(1)} n$ factors; $\overset{o}{\Omega}(\cdot), \overset{o}{O}(\cdot), \overset{o}{\Theta}(\cdot)$ notations hide $n^{o(1)}$ factors.

Background

In range searching, the input is a set of points in $\mathbb{R}^D$ for a fixed constant D. The goal is to build a structure such that for a query range, we can report or find the points in the range efficiently. This is a fundamental problem in computational geometry with many practical uses in e.g., databases and GIS systems. For more information, see surveys by Agarwal [GOT18] or Matoušek [Mat94]. We focus on a fundamental case of the problem where the ranges are semialgebraic sets of constant complexity which are defined by intersection/union/complementation of $O(1)$ polynomial inequalities of constant degree at most Δ in $\mathbb{R}^D$.

The study of this problem dates back to at least 35 years ago [YY85]. A linear space and $O(n^{1-1/D+o(1)})$ query time structure is given by Agarwal, Matoušek, and Sharir [AMS13], due to the recent "polynomial method" breakthrough [GK15]. However, it is not entirely clear what happens to the "fast-query" case: if we insist on polylogarithmic query time, what is the smallest possible space usage? Early on, some believed that the number of parameters plays an important role and thus $\tilde{O}(n^{\beta})$ space could be a reasonable conjecture [Mat94], but such a data structure was not found until 2019 [AAEZ21]. However, after the "polynomial method" revolution, and specifically after the breakthrough result of Agarwal, Matoušek and Sharir [AMS13], it could also be reasonably conjectured that $\tilde{O}(n^D)$ could also be the right bound. However, this was refuted recently by Afshani and Cheng [AC23b] who showed that in 2D, and for polynomials for the form $y - \sum_{i=0}^{\Delta} x^i \le 0$, there exists an $\overset{o}{\Omega}(n^{\Delta+1})$ space lower bound for data structures with query time $\overset{o}{O}(1)$. However, this lower bound does not go far enough, even in 2D, where a semialgebraic range can be specified by bivariate monic polynomial inequalities[2] of form $\sum_{i,j:i+j\le\Delta} a_{ij}x^i y^j \le 0$ with $a_{0\Delta} = -1$. In this case, β can be as large as $\mathbf{m}_{2,\Delta} = \binom{\Delta+2}{2} - 1 = \Theta(\Delta^2)$, and much larger than $\Delta + 1$ even for moderate Δ (e.g., for $\Delta = 4$, "5" versus "14", for $\Delta = 5$, "6" versus "20" and so on). Another main weakness is that their lower bound is only in 2D, but the upper bound [AAEZ21] works in arbitrary dimensions.

The correct upper bound tradeoff seems to be even more mysterious. Typically, the tradeoff is obtained by combining the linear space and the polylogarithmic query time solutions. For simplex range searching (i.e., when $\Delta = 1$), the tradeoff is $S(n) = \tilde{O}(n^D/Q(n)^D)$ [Mat93], which is a natural looking bound and it is also known to be optimal. The tradeoff bound becomes very mysterious for semialgebraic range searching. For example, for $D = 2$ and when restricted to queries of the form $y - \sum_{i=0}^{\Delta} x^i \le 0$, combining the existing solutions yields the bound $S(n) = \tilde{O}(n^{\Delta+1}/Q(n)^{2\Delta})$ whereas the known lower bound [AC23b] is $S(n) = \overset{o}{\Omega}(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2})$. One possible reason for this gap is that the lower bound construction is based on a uniform random point set, while in practice, the input can be pathological. But in general the uniform random point set assumption is not too restrictive for range searching problems. Almost all known lower bounds rely on this assumption: e.g., half-space

[2]We define that a D-variate polynomial $P(X_1, X_2, \cdots, X_D)$ is monic if the coefficient of X_2^{Δ} is -1.

range searching [BCP93, AMM06, AMX12], orthogonal range searching [Cha90a, Cha90b, Afs19], simplex range searching [Cha89, CR96, Afs13].

Our Results

Our results consist of two parts. First, we study a problem that we call "the general polynomial slab range reporting". Formally, let $P(X)$ be a monic D-variate polynomial of degree at most Δ, a general polynomial slab is defined to be the region between $P(X) = 0$ and $P(X) = w$ for some parameter w specified at the query time. Unlike [AC23b], our construction can reach the maximum possible parameter number $\mathbf{m}_{D,\Delta}$. For simplicity, we use $\mathbf{m}$ instead of $\mathbf{m}_{D,\Delta}$ when the context is clear. We give a space-time tradeoff lower bound of $S(n) = \overset{o}{\Omega}(n^{\mathbf{m}}/Q(n)^{\Theta((\Delta^2+D\Delta)\mathbf{m})})$, which is (almost) tight when $Q(n) = n^{o(1)}$. We also generalize the lower bound for the annuli reporting problem studied in [AC23b] to higher dimensions. For this problem, we show a space lower bound of $S(n) = \overset{o}{\Omega}(n^{D+1}/Q(n)^{2D})$, which is again (almost) tight in the fast-query time case.

For the second part, we present data structures that match the lower bounds studied in the work by Afshani and Cheng [AC23b]. We show that their lower bounds for 2D polynomial slabs and 2D annuli are tight for uniform random point sets. Our bound shows that current tradeoff given by the classical method of combining extreme solutions [MP15, AAEZ21] might not be tight. We shred some lights on the upper bound tradeoff and develop some ideas which could be used to tackle the problem. Our results are summarized in Table 1.

Table 1: Our Results (marked by *). Our upper bounds are for uniform random point sets.

Query Types	**Lower Bound**	**Upper Bound**
General Polynomial Slabs $\left(\mathbf{m} = \mathbf{m}_{D,\Delta} = \binom{D+\Delta}{D} - 1\right)$	$S(n) = \overset{o}{\Omega}\left(\frac{n^{\mathbf{m}}}{Q(n)^{\Theta(\mathbf{m})}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^{\mathbf{m}}}{Q(n)^{\Theta(\mathbf{m})}}\right)$ [MP15, AAEZ21]
When $Q(n) = \overset{o}{O}(1)$	$S(n) = \overset{o}{\Omega}(n^{\mathbf{m}})^*$	$S(n) = \tilde{O}(n^{\mathbf{m}})$ [MP15, AAEZ21]
2D Semialgebraic Sets $\left(\mathbf{m} = \mathbf{m}_{2,\Delta} = \binom{2+\Delta}{2} - 1\right)$	$S(n) = \overset{o}{\Omega}\left(\frac{n^{\mathbf{m}}}{Q(n)^{\mathbf{m}+\mathbf{m}^2(\mathbf{m}-1)-1}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^{\mathbf{m}}}{Q(n)^{2\mathbf{m}-2}}\right)$ [MP15, AAEZ21] $S(n) = \tilde{O}\left(\frac{n^{\mathbf{m}}}{Q(n)^{3\mathbf{m}-4}}\right)^*$
2D Polynomial Slabs	$S(n) = \overset{o}{\Omega}\left(\frac{n^{\Delta+1}}{Q(n)^{(\Delta+3)\Delta/2}}\right)$ [AC23b]	$S(n) = \tilde{O}\left(\frac{n^{\Delta+1}}{Q(n)^{2\Delta}}\right)$ [MP15, AAEZ21] $S(n) = \tilde{O}\left(\frac{n^{\Delta+1}}{Q(n)^{(\Delta+3)\Delta/2}}\right)^*$
2D Annuli	$S(n) = \overset{o}{\Omega}\left(\frac{n^3}{Q(n)^5}\right)$ [AC23b]	$S(n) = \tilde{O}\left(\frac{n^3}{Q(n)^4}\right)$ [MP15, AAEZ21] $S(n) = \tilde{O}\left(\frac{n^3}{Q(n)^5}\right)^*$

Technical Contributions

Compared to the previous lower bound in [AC23b], we need to wrestle with many complications that stem from the algebraic geometry nature of the problem. In Section 8.3, we cover them in greater detail, but briefly speaking, the technical heart of the results in [AC23b] is that "two univariate polynomials $P_1(x)$ and $P_2(x)$ that have sufficiently different leading coefficients, cannot pass close to each other for too long. However, this claim is not true for even bivariate polynomials, since $P_1(x,y)$ and $P_2(x,y)$ could have infinitely many roots in common and thus we can have $P_1(x,y) - P_2(x,y) = 0$ in an unbounded region of $\mathbb{R}^2$. Overcoming this requires significant innovations.

8.2 Preliminaries

In this section, we introduce some tools we will use in this paper. We will mainly use the lower bound tools used in [AC23b]. For more detailed introduction, we refer the readers to [AC23b].

Geometric Lower Bound Frameworks

We first present a lower bound framework in the pointer machine model of computation for range reporting problems. It is a streamlined version of the framework by Chazelle [Cha90a] and Chazelle and Rosenberg [CR96]. In essence, this is an encapsulation of the way the framework is used in [AC23b].

In a nutshell, in the pointer machine model, the memory is represented as a directed graph where each node can store one point and it has two pointers to two other nodes. Given a query, starting from a special "root" node, the algorithm explores a subgraph that contains all the input points to report. The size of the explored subgraph is the query time.

Intuitively, for range reporting, to answer a query fast, we need to store its output points close to each other. If each query range contains many points to report and two ranges share very few points, some points must be stored multiple times, thus the total space usage must be big. We present the framework, and refer the readers to Appendix 8.A for the proof.

Theorem 8.2.1. *Suppose the D-dimensional geometric range reporting problems admit an $S(n)$ space and $Q(n) + O(k)$ query time data structure, where n is the input size and k is the output size. Let $\mu^D(\cdot)$ denote the D-dimensional Lebesgue measure. (We call this D-measure for short.) Assume we can find $m = n^c$ ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots, \mathscr{R}_m$ in a D-dimensional cube $\mathscr{C}^D$ of side length $|l|$ for some constant c such that (i) $\forall i = 1, 2, \cdots, m, \mu^D(\mathscr{R}_i \cap \mathscr{C}^D) \geq 4c|l|^D Q(n)/n$; and (ii) $\mu^D(\mathscr{R}_i \cap \mathscr{R}_j) = O(|l|^D/(n2^{\sqrt{\log n}}))$ for all $i \neq j$. Then, we have $S(n) = \overset{o}{\Omega}(mQ(n))$.*

A Lemma for Polynomials

Given a univariate polynomial and some positive value w, the following lemma from [AC23b] upper bounds the length of the interval within which the absolute value of the polynomial is no more than w. We will use this lemma as a building block for some of our proofs.

Lemma 8.2.1 (Afshani and Cheng [AC23b])**.** *Given a degree-Δ univariate polynomial $P(x) = \sum_{i=0}^{\Delta} a_i x^i$ where $|a_\Delta| > 2$ and $\Delta > 0$. Let w be any positive value. If $|P(x)| \le w$ for all $x \in [x_0, x_0 + t]$ for some parameter x_0, then $t = O((w/|a_\Delta|)^{1/\Delta})$.*

Useful Properties about Matrices

In this section, we recall some useful properties about matrices. We first recall some properties of the determinant of matrices. One important property is that the determinant is mutilinear:

Lemma 8.2.2. *Let $A = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix}$ be a $n \times n$ matrix where $\mathbf{a}_i$'s are vectors in $\mathbb{R}^n$. Suppose $\mathbf{a}_j = r \cdot \mathbf{w} + \mathbf{v}$ for some $r \in \mathbb{R}$ and $\mathbf{w}, \mathbf{v} \in \mathbb{R}^n$, then the determinant of A, denoted* $\det(A)$*, is*

$$\begin{aligned} \det(A) &= \det\left(\begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_{j-1} & \mathbf{a}_j & \mathbf{a}_{j+1} & \cdots & \mathbf{a}_n \end{bmatrix}\right) \\ &= r \cdot \det\left(\begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_{j-1} & \mathbf{w} & \mathbf{a}_{j+1} & \cdots & \mathbf{a}_n \end{bmatrix}\right) \\ &\quad + \det\left(\begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_{j-1} & \mathbf{v} & \mathbf{a}_{j+1} & \cdots & \mathbf{a}_n \end{bmatrix}\right). \end{aligned}$$

One of the special types of matrices we will use is the Vandermonde matrix which is a square matrix where the terms in each row form a geometric series, i.e., $V_{ij} = x_i^{j-1}$ for all indices i and j. The determinant of such a matrix is $\det(V) = \prod_{1 \le i < j \le n} (x_j - x_i)$.

Given an n-tuple $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$ where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$, we can define a generalized Vandermonde matrix V^* defined by λ, where $V^*_{ij} = x_i^{\lambda_{n-j+1}+j-1}$. The determinant of V^* is known to be the product of the determinant of the induced Vandermonde matrix V_{V^*} with $V_{ij} = x_i^{j-1}$ and the Schur polynomial $s_\lambda(x_1, x_2, \cdots, x_n) = \sum_T x_1^{t_1} \cdots x_n^{t_n}$, where the summation is over all semistandard Young tableaux [You01] T of shape λ. The exponents $t_1, t_2, \cdots, t_n$ are all nonnegative numbers. The following lemma bounds the determinant of a generalized Vandermonde matrix.

Lemma 8.2.3. *Let V^* be a generalized Vandermonde matrix defined by $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$ where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$. If $n, \lambda_1 = \Theta(1)$, and for all i, $x_i = \Theta(1)$, then* $\det(V^*) = \Theta(\det(V_{V^*}))$*, where V_{V^*} is the induced Vandermonde matrix with $V_{ij} = x_i^{j-1}$.*

8.3 Lower Bound for Range Reporting with General Polynomial Slabs

In this section, we prove our main lower bound for general polynomial slabs. We first formally define polynomial slabs.

Definition 8.3.1. *A general polynomial slab in $\mathbb{R}^D$ is a triple (P,a,b) where $P \in \mathbb{R}[X]$ is a degree-Δ D-variate polynomial and a,b are two real numbers such that $a < b$. A general polynomial slab is defined as $\{X \in \mathbb{R}^D : a \le P(X) \le b\}$. Note that due to rescaling, we can assume that the polynomial is monic.*

Before presenting our results, we first describe the technical challenges of this problem. We explain why the construction used in [AC23b] cannot be generalized in an obvious way and give some intuition behind our lower bound construction.

Technical Challenges

Our goal is a lower bound of the form $\overset{o}{\Omega}(n^{\mathbf{m}}/Q(n)^{\Theta(\mathbf{m})})$. To illustrate the challenges, consider the case $D = 2$ and the unit square $\mathscr{U} = \mathscr{U}^2 = [0,1] \times [0,1]$. To use Theorem 9.2.1, we need to generate about $\overset{o}{\Omega}(n^{\mathbf{m}})$ polynomial slabs such that each slab should have width approximately $\Omega(Q(n)/n)$, and any two slabs should intersect with area approximately $O(1/n)$. Intuitively, this means two slabs cannot intersect over an interval of length $\Omega(1/Q(n))$.

In Lemma 8.2.1, for univariate polynomials, the observation behind their construction is that when the leading coefficients of two polynomials differ by a large number, the length of the interval in which two polynomials are close to each other is small. However, when we consider general bivariate polynomials in $\mathbb{R}^2$, this observation is no longer true. For example, consider $P_1(x,y) = (x+1)(1000x^2+y)$ and $P_2(x,y) = (x+1)(x^2+1000y)$. The leading coefficients are 1000 and 1 respectively, but since P_1, P_2 have a common factor $(x+1)$, their zero sets have a common line. Thus any slab of width $Q(n)/n$ generated for these two polynomial will have infinite intersection area, which is too large to be useful.

At first glance, it might seem that this problem can be fixed by picking the polynomials randomly, e.g., each coefficient is picked independently and uniformly from the interval $[0,1]$, as a random polynomial in two or more variables is irreducible with probability 1. Unfortunately, this does not work either but for some very nontrivial reasons. To see this, consider picking coefficients uniformly at random from range $[0,1]$ for bivariate polynomials $P(x,y) = \sum_{i+j\le\Delta} a_{ij}x^iy^j$. The probability of pick a polynomial with $0 \le a_{0j} \le \frac{1}{n}$ for all a_{0j} is $\frac{1}{n^{\Delta+1}}$. For such polynomials, $0 \le P(0,y) \le \frac{\Delta+1}{n}$ for $y \in [0,1]$. Suppose we sampled two such polynomials, then the two slabs generated using them will contain $x = 0$ for $y \in [0,1]$, meaning, the two slabs will have too large of an area ($\Omega(Q(n)/n)$) in common, so we cannot have that. Unfortunately, if we sample more than $n^{\Delta+1}$ polynomials, this will happen with probability close to one, and there seems to be no easy fix. A deeper insight into the issue is given below.

Map a polynomial $\sum_{i+j\le\Delta} a_{ij}x^iy^j$ to the point $(a_{00}, a_{01}, \cdots, a_{\Delta 0})$ in $\mathbb{R}^{\mathbf{m}}$. The above randomized construction corresponds to picking a random point from the unit cube $\mathscr{U}^{\mathbf{m}}$ in $\mathbb{R}^{\mathbf{m}}$. Now consider the subset Γ of $\mathbb{R}^{\mathbf{m}}$ that corresponds to reducible polynomials. The issue is that Γ intersects $\mathscr{U}^{\mathbf{m}}$ and thus we will sample polynomials that are close to reducible polynomials, e.g., a sampled polynomial with $a_{0j} = 0 \in [0, \frac{1}{n}]$ is close to the reducible polynomial with $a_{0j} = 0$. Pick a large enough sample and two points will lie close to the same reducible polynomial and thus they will produce a "large" overlap in the construction. Our main insight is that there exists a point $\mathbf{p}$ in $\mathscr{U}^{\mathbf{m}}$ that has a "fixed" (i.e., constant) distance to Γ; thus, we can consider a neighborhood around $\mathbf{p}$ and sample our polynomials from there. However, more technical challenges need to be overcome to even make this idea work but it turns out, we can simply pick our polynomials from a grid constructed in the small enough neighborhood of some such point $\mathbf{p}$ in $\mathbb{R}^{\mathbf{m}}$.

A Geometric Lemma

In this section, we show a geometric lemma which we will use to establish our lower bound. In a nutshell, given two monic D-variate polynomials P_1, P_2 and a point $p = (p_2, p_3, \cdots, p_D) \in \mathbb{R}^{D-1}$ in the $(D-1)$-dimensional subspace perpendicular to the X_1-axis, we define the distance between $Z(P_1)$[3] and $Z(P_2)$ along the X_1-axis at point p to be $|a-b|$, where $(a, p_2, \cdots, p_D) \in Z(P_1)$ and $(b, p_2, \cdots, p_D) \in Z(P_2)$. In general, this distance is not well-defined as there could be multiple a and b's satisfying the definition. But we can show that for a specific set of polynomials, a, b can be made unique and thus the distance is well-defined. For P_1, P_2 with "sufficiently different" coefficients, we present a lemma which upper bounds the $(D-1)$-measure of the set of points p at which the distance between $Z(P_1)$ and $Z(P_2)$ is "small". Intuitively, this can be viewed as a generalization of Lemma 8.2.1. We first prove the lemma in 2D for bivariate polynomials, and then extend the result to higher dimensions.

We begin by defining notations we will use during the proof.

Definition 8.3.2. *Let $I^D \subseteq \{(i_1, i_2, \cdots, i_D) \in \mathbb{N}^D\}$[4], $D \geq 1$, be a set of D-tuples where each tuple consists of nonnegative integers. We call I^D an index set (of dimension D). Let $X^D = (X_1, X_2, \cdots, X_D)$ be a D-tuple of indeterminates. When the context is clear, we use X for simplicity. Given an index set I^D, we define*

$$P(X) = \sum_{i \in I^D} A_i X^i,$$

where $A_i \in \mathbb{R}$ is the coefficient of X^i and $X^i = X_1^{i_1} X_2^{i_2} \cdots X_D^{i_D}$, to be a D-variate polynomial. For any $i \in I^D$, we define $\sigma(i) = \sum_{j=1}^{D} i_j$. Let Δ be the maximum $\sigma(i)$ with $A_i \neq 0$, and we say P is a degree-Δ polynomial. Given a D-tuple T, we use $T_{:j}$ to denote a j-tuple by taking only the first j components of T. Also, we use notation T_j to specify the j-th component of T. Conversely, given a $(D-1)$-tuple t and a value v, we define $t \oplus v$ to be the D-tuple formed by appending v to the end of t.

We will consider polynomials of form

$$P(X) = X_1 - X_2^{\Delta} + \sum_{i \in I^D} A_i X^i,$$

[3] $Z(P)$ denotes the zero set of polynomial P.

[4] In this paper, $\mathbb{N} = \{0, 1, 2, \cdots\}$.

where $0 \le A_i = O(\varepsilon) = o(1)$ for all $\sigma(i) \le \Delta$ except that $A_i = 0$ for $i = (0, \Delta, 0, \cdots, 0)$. Intuitively, these are monic polynomials packed closely in the neighborhood of $P(X) = X_1 - X_2^D$. For simplicity, we call them "packed" polynomials. We will prove a property for packed polynomials that are "sufficiently distant". More precisely,

Definition 8.3.3. *Given two distinct packed degree-Δ D-variate polynomials P_1, P_2, we say P_1, P_2 are "distant" if each coefficient of $P_1 - P_2$ has absolute value at least $\xi_D = \delta\tau^{\mathbf{B}}(\eta\tau)^{(D-2)\Delta} > 0$ if not zero for parameters $\delta, \eta, \tau > 0$ and $\eta\tau = O((1/\varepsilon)^{1/\mathbf{B}})$, where $\mathbf{B} = \binom{\mathbf{b}}{2}$ and $\mathbf{b} = \mathbf{m}_{2,\Delta}$ is the maximum number of coefficients needed to define a monic degree-Δ bivariate polynomial.*

We will use the following simple geometric observation. See Appendix 8.B for the proof.

Observation 8.3.1. *Let P be a packed D-variate polynomial and $a = (a_1, a_2, \cdots, a_D) \in Z(P)$. If $a_i \in [1,2]$ for all $i = 2, 3, \cdots, D$, then there exists a unique a_1 such that $0 < a_1 = O(1)$.*

With this observation, we can define the distance between the zero sets of two polynomials along the X_1-axis at a point in $[1,2]^{D-1}$ of the subspace perpendicular to the X_1 axis.

Definition 8.3.4. *Given two packed polynomials P_1, P_2 and a point $p = (p_2, p_3, \cdots, p_D) \in [1,2]^{D-1}$, we define the distance between $Z(P_1)$ and $Z(P_2)$ at p, denoted $\pi(Z(P_1), Z(P_2), p)$, to be $|a - b|$ s.t. $a, b > 0$, and $(a, p_2, p_3, \cdots, P_D) \in Z(P_1)$ and $(b, p_2, p_3, \cdots, P_D) \in Z(P_2)$.*

Now we show a generalization of Lemma 8.2.1 to distant bivariate polynomials in 2D.

Lemma 8.3.1. *Let P_1, P_2 be two distinct distant bivariate polynomials. Let $I = \{y : \pi(Z(P_1), Z(P_2), y) = O(w) \wedge y \in [1,2]\}$, where $w = \delta/\eta^{\mathbf{B}} = o(1)$. Then $|l| = O(\frac{1}{\eta\tau})$.*

Proof. We prove it by contradiction. The idea is that if the claim does not hold, then we can "tweak" the coefficients of P_2 by a small amount such that the tweaked polynomial and P_1 have $\mathbf{b}$ common roots. Next, we show this implies that the tweaked polynomial is equivalent to P_1. Finally we reach a contradiction by noting that by assumption at least one of the coefficients of P_1 and P_2 is not close. Let $P_1(x,y) = x - y^\Delta + \sum_{i=0}^{\Delta}\sum_{j=0}^{\Delta-i} a_{ij}x^iy^j$ and $P_2(x,y) = x - y^\Delta + \sum_{i=0}^{\Delta}\sum_{j=0}^{\Delta-i} b_{ij}x^iy^j$ where by definition all a_{ij}'s and b_{ij}'s are $O(\varepsilon)$. Suppose for the sake of contradiction that $|l| = \omega(\frac{1}{\eta\tau})$. We pick $\mathbf{b}$ values $y_1, y_2, \cdots, y_{\mathbf{b}}$ in I s.t. $|y_i - y_j| \ge |l|/\mathbf{b}$ for all $i \ne j$. Let $x_1, x_2, \cdots, x_{\mathbf{b}}$ be the corresponding values s.t. $(x_k, y_k) \in Z(P_1)$ in the first quadrant, i.e., $P_1(x_k, y_k) = 0$ for $k = 1, 2, \cdots, \mathbf{b}$. Note that

$$P_1(x_k, y_k) = 0 \equiv x_k - y_k^\Delta + \sum_{i=0}^{\Delta}\sum_{j=0}^{\Delta-i} a_{ij}x_k^iy_k^j = 0 \implies x_k = y_k^\Delta - O(\varepsilon),$$

since $a_{ij} = O(\varepsilon)$ and $x_k, y_k = O(1)$ by Observation 8.3.1. Since $\pi(Z(P_1), Z(P_2), y_k) = O(w)$ for all $y_k \in I$, let $(x_k + \Delta x_k, y_k)$ be the points on $Z(P_2)$, we have $P_2(x_k + \Delta x_k, y_k) = P_2(x_k, y_k) + \Theta(\Delta x_k) = 0$. Since $|\Delta x_k| = O(w)$, $P_2(x_k, y_k) = \mathbf{g}_k$ for some $|\mathbf{g}_k| = O(w)$. We would like to show that we can "tweak" every coefficient b_{ij} of $P_2(x,y)$ by some value $\mathbf{d}_{ij}$, to turn P_2 into a polynomial Q s.t. $Q(x_k, y_k) = 0, \forall k = 1, 2, \cdots, \mathbf{b}$. If so, for every pair (x_k, y_k),

$$\begin{aligned} Q(x_k, y_k) &= x_k - y_k^\Delta + \sum_{i=0}^{\Delta} \sum_{j=0}^{\Delta - i} (b_{ij} + \mathbf{d}_{ij}) x_k^i y_k^j \\ &= P_2(x_k, y_k) + \sum_{i=0}^{\Delta} \sum_{j=0}^{\Delta - i} \mathbf{d}_{ij} x_k^i y_k^j \\ &= \mathbf{g}_k + \sum_{i=0}^{\Delta} \sum_{j=0}^{\Delta - i} \mathbf{d}_{ij} (y_k^\Delta - O(\varepsilon))^i y_k^j \\ &= \mathbf{g}_k + \sum_{i=0}^{\Delta} \sum_{j=0}^{\Delta - i} \mathbf{d}_{ij} (y_k^{i\Delta} - O(\varepsilon)) y_k^j, \end{aligned}$$

where the last equality follows from $\varepsilon = o(1)$ and $1 \le y_k \le 2$. So to find $\mathbf{d}_{ij}$'s and to be able to tweak $P_2(x,y)$, we need to solve the following linear system

$$\begin{bmatrix} 1 & y_1 & y_1^2 & \cdots & y_1^{\Delta-1} & y_1^\Delta - O(\varepsilon) & \cdots & y_1^{\Delta^2} - O(\varepsilon) \\ 1 & y_2 & y_2^2 & \cdots & y_2^{\Delta-1} & y_2^\Delta - O(\varepsilon) & \cdots & y_2^{\Delta^2} - O(\varepsilon) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & y_{\mathbf{b}} & y_{\mathbf{b}}^2 & \cdots & y_{\mathbf{b}}^{\Delta-1} & y_{\mathbf{b}}^\Delta - O(\varepsilon) & \cdots & y_{\mathbf{b}}^{\Delta^2} - O(\varepsilon) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{d}_{00} \\ \mathbf{d}_{01} \\ \vdots \\ \mathbf{d}_{\Delta 0} \end{bmatrix} = \begin{bmatrix} -\mathbf{g}_1 \\ -\mathbf{g}_2 \\ \vdots \\ -\mathbf{g}_{\mathbf{b}} \end{bmatrix},$$

where the exponents of y_k are generated by $i\Delta + j$ for $i, j \in \{0, 1, 2, \cdots, \Delta\}$, $j \neq \Delta$, and $i + j \le \Delta$. Let us call the above linear system $A \cdot \mathbf{d} = \mathbf{g}$.

By Lemma 8.2.2, $\det(A) = \det(A^*) + \sum_{l=1}^{\Theta(1)} \det(A_l)$, where A^* is a generalized Vandermonde matrix defined by an $\mathbf{b}$-tuple $\lambda = (\Delta^2 - \mathbf{b}, \ldots, 0)$, and each A_l is a matrix with some columns being $O(\varepsilon)$. Since $\mathbf{b} = \binom{2+\Delta}{2} - 1$ is $\Theta(1)$, by Lemma 8.2.3, we can bound $\det(A^*)$ by $\Theta(\det(V_{A^*}))$, where V_{A^*} is the induced Vandermonde matrix. Since $\left|y_i - y_j\right| = \Omega(|I|)$ for $i \neq j$, $\det(V_{A^*}) = \prod_{1 \le i < j \le \mathbf{b}} (y_j - y_i)) = \Omega(|I|^{\mathbf{B}})$. On the other hand, for every matrix A_l, there is at least one column where the magnitude of all the entries is $O(\varepsilon)$. Since all other entries are bounded by $O(1)$, by the Leibniz formula for determinants, $|\det(A_l)| = O(\varepsilon) = O((\frac{1}{\eta\tau})^{\mathbf{B}})$. Since $|I|^{\mathbf{B}} = \omega((\frac{1}{\eta\tau})^{\mathbf{B}})$, we can bound $|\det(A)| = \Omega(|I|^{\mathbf{B}})$ and in particular $|\det(A)| \neq 0$ and thus the above system has a solution and the polynomial Q exists. Furthermore, we can compute $\mathbf{d} = A^{-1}\mathbf{g} = \frac{1}{\det(A)} C \cdot \mathbf{g}$, where C is the cofactor matrix of A. Since all entries of A are bounded by $O(1)$, then the entries of C, being cofactors of A, are also bounded by $O(1)$. Since $|\mathbf{g}_k| = O(w)$ and $|I| = \omega(\frac{1}{\eta\tau})$, for every $k = 1, 2, \cdots, \mathbf{b}$, we have $\left|\mathbf{d}_{ij}\right| = O(w/|I|^{\mathbf{B}}) = o(w(\eta\tau)^{\mathbf{B}}) = o(\delta\tau^{\mathbf{B}})$.

However, since both $Z(P_1)$ and $Z(Q)$ pass through these $\mathbf{b}$ points, both P_1 and Q should satisfy $A \cdot \mathbf{c}_1 = 0$ and $A \cdot \mathbf{c}_2 = 0$, where $\mathbf{c}_1, \mathbf{c}_2$ are their coefficient vectors respectively. But since $\det(A) \neq 0$, $\mathbf{c}_1 = \mathbf{c}_2$, meaning, $P_1 \equiv Q$. This means for every $i, j = 0, 1, \cdots, \Delta$, where $j \neq \Delta$ and $i + j \leq \Delta$, $\left|a_{ij} - b_{ij}\right| = \mathbf{d}_{ij} = o(\delta\tau^{\mathbf{B}})$. However, by assumption, if two polynomials are not equal, then there exists at least one c_{ij} such that they differ by at least $\delta\tau^{\mathbf{B}}$, a contradiction. So $|l| = O(\frac{1}{\eta\tau})$. □

We now generalize Lemma 8.3.1 to higher dimensions.

Lemma 8.3.2. *Let P_1, P_2 be two distinct distant D-variate polynomials. Let $S = \{X : \pi(Z(P_1), Z(P_2), X) = O(w) \wedge X \in [1,2]^{D-1}\}$, where $w = \delta/\eta^{\mathbf{B}} = o(1)$. Then $\mu^{D-1}(S) = O(\frac{1}{\eta\tau})$.*

Proof. We prove the lemma by induction. The base case when $D = 2$ is Lemma 8.3.1. Now suppose the lemma holds for dimension $D - 1$, we prove it for dimension D. Observe that we can rewrite a D-variate polynomial $P(X) = X_1 - X_2^{\Delta} + \sum_{i \in I^D} A_i X^i$ as $P(X) = X_1 - X_2^{\Delta} + \sum_{j \in I^D_{:D-1}} (f_j(X_D)) X^j_{:D-1}$, where $f_j(X_D) = \sum_{k=0}^{\Delta - \sigma(j)} A_{j \oplus k} X_D^k$. Consider two distinct distant D-variate polynomials $P(X) = X_1 - X_2^{\Delta} + \sum_{i \in I^D} A_i X^i$ and $Q(X) = X_1 - X_2^{\Delta} + \sum_{i \in I^D} B_i X^i$. Let f_j, g_j be the corresponding coefficients for $X^j_{:D-1}$. Note that there exists some j such that $f_j \not\equiv g_j$ because P_1, P_2 are distinct. Let $h_j(X_D) = f_j(X_D) - g_j(X_D)$ and observe that h_j is a univariate polynomial in X_D. We show that the interval length of X_D in which $\left|h_j(X_D)\right| < \xi_{D-1}$ is upper bounded by $O(\frac{1}{\eta\tau})$ for any $h_j(X_D) \not\equiv 0$. Pick any $h_j(X_D) \not\equiv 0$ and note that this means there exists at least one coefficient of $h_j(X_D)$ that is nonzero. By assumption, each coefficient of $h_j(X_D)$ has absolute value at least ξ_D if not zero. If the constant term is the only nonzero term, then the interval length of X_D in which $\left|h_j(X_D)\right| < \xi_{D-1}$ is 0, since $\left|h_j(X_D)\right| \geq \xi_D > \xi_{D-1}$ by definition. Otherwise by Lemma 8.2.1, the interval length $|r|$ for X_D in which $\left|h_j(X_D)\right| < \xi_{D-1}$ is upper bounded by

$$|r| = O\left(\left(\frac{\xi_{D-1}}{\xi_D}\right)^{1/\Delta}\right) = O\left(\left(\frac{1}{(\eta\tau)^{\Delta}}\right)^{1/\Delta}\right) = O\left(\frac{1}{\eta\tau}\right).$$

Since the total number of different j's is $\Theta(1)$, the total number of $h_j(X_D)$ is then $\Theta(1)$. So the total interval length for X_D within which there is some nonzero $h_j(X_D)$ with $\left|h_j(X_D)\right| < \xi_{D-1}$ is upper bounded by $\Theta(1) \cdot O(\frac{1}{\eta\tau}) = O(\frac{1}{\eta\tau})$. Since we are in a unit hypercube, we can simply upper bound $\mu^{D-1}(S)$ by $O(\frac{1}{\eta\tau}) \cdot \Theta(1) = O(\frac{1}{\eta\tau})$ in this case. Otherwise, $\left|h_j(X_D)\right| \geq \xi_{D-1}$ for all j, and by the inductive hypothesis, the $(D-2)$-measure of S in $[1,2]^{D-2}$ is upper bounded by $O(\frac{1}{\eta\tau})$. Integrating over all X_D, $\mu^{D-1}(S)$ is bounded by $O(\frac{1}{\eta\tau})$ in this case as well. □

Lower Bound for General Polynomial Slabs

Now we are ready to present our lower bound construction. We will use a set $\mathscr{S}$ of D-variate polynomials in $\mathbb{R}[X]$ of form:

$$P(X) = X_1 - X_2^{\Delta} + \sum_{i \in I^D} A_i X^i,$$

where X is a D-tuple of indeterminates, I^D is an index set containing all D-tuples i satisfying $\sigma(i) \le \Delta$, and each $A_i \in \{k\xi_D : k \in \{\lfloor \frac{\varepsilon}{2\xi_D} \rfloor, \lfloor \frac{\varepsilon}{2\xi_D} \rfloor + 1, \cdots, \lfloor \frac{\varepsilon}{\xi_D} \rfloor\}\}$ for some $\xi_D = \delta\tau^{\mathbf{B}}(\eta\tau)^{(D-2)\Delta}$ to be set later, except for one special coefficient: we set $A_i = 0$ for $i = (0, \Delta, 0, \cdots, 0)$. Note that every pair of the polynomials in $\mathscr{S}$ is distant. A general polynomial slab is defined to be a triple $(P, 0, w)$ where $P \in \mathscr{S}$ and w is a parameter to be set later. We need $w = o(\varepsilon)$ and $\varepsilon = o(1)$.

We consider a unit cube $\mathscr{U}^D = \prod_{i=1}^{D} [1,2] \subseteq \mathbb{R}^D$ and use Framework 9.2.1. Recall that to use Framework 9.2.1, we need to lower bound the intersection D-measure of each slab we generated and $\mathscr{U}^D$, and upper bound the intersection D-measure of two slabs.

Given a slab $(P, 0, w)$ in our construction, first note that both P and $P - w$ are packed polynomials. We define the width of $(P, 0, w)$ to be the distance between $Z(P)$ and $Z(P - w)$ along the X_1-axis. The following lemma shows that the width of each slab we generate will be $\Theta(w)$ in $\mathscr{U}^D$. See Appendix 8.C for the proof.

Lemma 8.3.3. *Let $P_1 \in \mathscr{S}$ and $P_2 = P_1 - r$ for any $0 \le r = O(w)$. Then $\pi(Z(P_1), Z(P_2), X) = \Theta(r)$ for any $X \in [1,2]^{D-1}$.*

The following simple lemma bounds the $(D-1)$-measure of the projection of the intersection of the zero set of any polynomial in our construction and $\mathscr{U}^D$ on the $(D-1)$-dimensional subspace perpendicular to X_1-axis. See Appendix 8.D for the proof.

Lemma 8.3.4. *Let $P \in \mathscr{S}$. The projection of $Z(P) \cap \mathscr{U}^D$ on the $(D-1)$-dimensional space perpendicular to the X_1-axis has $(D-1)$-measure $\Theta(1)$.*

Combining Lemma 8.3.3 and Lemma 8.3.4, we easily bound the intersection D-measure of any slab in our construction and $\mathscr{U}^D$.

Corollary 8.3.1. *Any slab in our construction intersects $\mathscr{U}^D$ with D-measure $\Theta(w)$.*

Combining Lemma 8.3.3 and Lemma 8.3.2, we easily bound the intersection D-measure of two slabs in our construction in $\mathscr{U}^D$.

Corollary 8.3.2. *Any two slabs in our construction intersect with D-measure $O(\frac{w}{\eta\tau})$ in $\mathscr{U}^D$.*

Since there are at most $\mathbf{m} = \binom{D+\Delta}{D} - 1$ parameters for a degree-Δ D-variate monic polynomial, the number of polynomial slabs we generated is then

$$\Theta\left(\left(\frac{\varepsilon}{\xi_D}\right)^{\mathbf{m}}\right) = \Theta\left(\left(\frac{n}{Q(n)^{1+2\mathbf{B}+(D-2)\Delta}2^{((D-2)\Delta+2\mathbf{B})\sqrt{\log n}}}\right)^{\mathbf{m}}\right) = O(n^{\mathbf{m}}),$$

by setting $\delta = wQ(n)^{\mathbf{B}}$, $\eta = Q(n)$, $\tau = 2^{\sqrt{\log n}}$, $\varepsilon = \frac{1}{Q(n)^{\mathbf{B}}2^{\mathbf{B}\sqrt{\log n}}}$, and $w = c_w Q(n)/n$ for a sufficiently large constant c_w. We pick c_w s.t. each slab intersects $\mathscr{U}^D$ with D-measure, by Corollary 8.3.1, $\Omega(w) \ge 4\mathbf{m}Q(n)/n$. By Corollary 8.3.2 the D-measure of the intersection of two slabs is upper bounded by $O(\frac{w}{Q(n)2^{\sqrt{\log n}}}) = O(\frac{1}{n2^{\sqrt{\log n}}})$. By Theorem 9.2.1, we get the lower bound $S(n) = \overset{o}{\Omega}\left(n^{\mathbf{m}}/Q(n)^{\mathbf{m}+2\mathbf{mB}+\mathbf{m}(D-2)\Delta-1}\right)$. Thus we get the following result.

Theorem 8.3.1. *Let $\mathscr{P}$ be a set of n points in $\mathbb{R}^D$, where $D \geq 2$ is an integer. Let $\mathscr{R}$ be the set of all D-dimensional generalized polynomial slabs $\{(P,0,w) : \deg(P) = \Delta \geq 2, w > 0\}$ where $P \in \mathbb{R}[X_1, X_2, \cdots, X_D]$ is a monic degree-Δ polynomial. Let* $\mathbf{b}$ *(resp.* $\mathbf{m}$*) be the maximum number of parameters needed to specify a moinc degree-Δ bivariate (resp. D-variate) polynomial. Then any data structure for $\mathscr{P}$ that can answer generalized polynomial slab reporting queries from $\mathscr{R}$ with query time $Q(n) + O(k)$, where k is the output size, must use $S(n) = \overset{o}{\Omega}\left(\frac{n^{\mathbf{m}}}{Q(n)^{\mathbf{m}+2\mathbf{mB}+\mathbf{m}(D-2)\Delta-1}}\right)$ space, where and* $\mathbf{B} = \binom{\mathbf{b}}{2}$.

We mention in passing that Afshani and Cheng also studied the lower bounds for 2D annuli [AC23b]. It is also possible to generalize their construction and argument to higher dimensions in a straightforward way to obtain a lower bound for sphere shells of form $\{p \in \mathbb{R}^D : r \leq \|p - O\|_2 \leq r + w\}$ defined by its center O, radius r, and width w in $\mathbb{R}^D$. Note that sphere shells cannot be modeled by polynomial slabs we considered in the proof of Theorem 8.3.1 and so the lower bound proved there does not apply to sphere shells. We simply state the result here and refer the readers to Appendix 8.E for details.

Theorem 8.3.2. *Any data structure that solves sphere shell reporting problems in D dimensions on a point set of size n with query time $Q(n) + O(k)$, where $D \geq 3$ and k is the output size, must use $S(n) = \overset{o}{\Omega}(n^{D+1}/Q(n)^{2D})$ space.*

8.4 Data Structures for Uniform Random Point Sets

In this section, we present data structures for an input point set $\mathscr{P}$ uniformly randomly distributed in a unit square $\mathscr{U} = [0,1] \times [0,1]$ for semialgebraic range reporting queries in $\mathbb{R}^2$. Our hope is that some of these ideas can be generalized to build more efficient data structures for general point sets. To this end, we show two approaches based on two different assumptions: one assumes the query curve has bounded curvature, and the other assumes bounded derivatives. We show that for any degree-Δ bivariate polynomial inequality, we can build a data structure with space-time tradeoff $S(n) = \tilde{O}(n^{\mathbf{m}}/Q(n)^{3\mathbf{m}-4})$[5], which is optimal for $\mathbf{m} = 3$ [AC23b]. When the query curve has bounded derivatives for the first Δ orders within $\mathscr{U}$, this bound sharpens to $\tilde{O}(n^{\mathbf{m}}/Q(n)^{((2\mathbf{m}-\Delta)(\Delta+1)-2)/2})$, which matches the lower bound in [AC23b] for polynomial slabs generated by inequalities of form $y - \sum_{i \leq \Delta} a_i x^i \geq 0$. Since any polynomial can be factorized into a product of $O(1)$ irreducible polynomials, and we can show that the zero set of any irreducible polynomial has bounded curvature (See

[5]In this section, we use $\mathbf{m}$ to denote the number of parameters needed to define a degree-Δ **bivariate** polynomial.

Appendix 8.F for more details), we can express the original range by a semialgebraic set consisting of $O(1)$ irreducible polynomials. We mention that both data structures can be made multilevel, then by the standard result of multilevel data structures, see e.g., [Mat93] or [Aga17], it suffices for us to focus on one irreducible polynomial inequality. So the curvature-based approach works for all semialgebraic sets. For both approaches, the main ideas are similar: we first partition $\mathscr{U}$ into a $Q(n) \times Q(n)$ grid G, and then build a set of slabs in each cell of G to cover the boundary $\partial\mathscr{R}$ of a query range $\mathscr{R}$. The boundaries of each slab consist of the zero sets of lower degree polynomials. We build a data structure to answer degree-Δ polynomial inequality queries inside each slab, then use the boundaries of slabs to express the remaining parts of $\mathscr{R}$. This lowers the degree of query polynomials, and then we can use fast-query data structures to handle the remaining parts. We assume our data structure can perform common algebraic operations in $O(1)$ time, e.g., compute roots, compute derivatives, etc.

A Curvature-based Approach

The main observation we use is that when the total absolute curvature of $\partial\mathscr{R}$ is small, the curve behaves like a line, and so we can cover it using mostly "thin" slabs, and a few "thick" slabs when the curvature is big. See Figure 1 for an example. We use the curvature as a "budget": thin slabs have few points in them so we can afford to store them in a "fast" data structure and the overhead will be small. Doing the same with the thick slabs will blow up the space too much so instead we store them in "slower" but "smaller" data structures. The crucial observation here is that for any given query, we only need to use a few "thick" slabs so the slower query time will be absorbed in the overall query time.

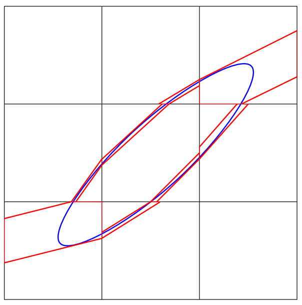


Figure 1: Cover an Ellipse with Slabs of Different Widths

The high-level idea is to build a two-level data structure. For the bottom-level, we build a multilevel simplex range reporting data structure [Mat93] with query time $\tilde{O}(1) + O(k)$ and space $S(n) = \tilde{O}(n^2)$. For the upper-level, for each cell C in G and a parameter $\alpha = 2^i/Q(n)$, for $i = 0, \cdots, \lfloor \log Q(n) \rfloor$, we generate a series of parallel disjoint slabs of width $\alpha/Q(n)$ such that they together cover C. Then we rotate these slabs by angle $\gamma = j/Q(n)$, for $j = 1, 2, \cdots, \lfloor 2\pi Q(n) \rfloor$. For each slab we generated during this process, we collect all the points in it and build a $\tilde{O}(Q(n)\alpha) + O(k)$ query time and $\tilde{O}((n/(Q(n)\alpha))^{\mathbf{m}})$ space data structure by linearization [YY85] to $\mathbb{R}^{\mathbf{m}}$ and using simplex range reporting [Mat93].

The following lemma shows we can efficiently report the points close to $\partial\mathscr{R}$ using slabs we constructed. For the proof of this lemma, we refer the readers to Appendix 8.G.

Lemma 8.4.1. *We can cut $\partial\mathscr{R}$ into a set $\mathscr{S}$ of $O(Q(n))$ sub-curves such that for each sub-curve σ, we can find a set S_σ of slabs that together cover σ. Let P_σ be the subset of the input that lies inside the query and inside the slabs, i.e., $P_\sigma = \mathscr{R} \cap \mathscr{P} \cap (\cup_{s \in S_\sigma} s)$. P_σ can be reported in time $Q(n)\tilde{O}(\kappa_\sigma + 1/Q(n)) + O(|P_\sigma|)$, where κ_σ is the total absolute curvature of σ. Furthermore, for any two distinct $\sigma_1, \sigma_2 \in \mathscr{S}$, $s_1 \cap s_2 = \emptyset$ for all $s_1 \in S_{\sigma_1}, s_2 \in S_{\sigma_2}$.*

With Lemma 8.4.1, we can now bound the total query time for points close to $\partial\mathscr{R}$ by $\sum_\sigma Q(n)\tilde{O}(\kappa_\sigma + 1/Q(n)) + O(t_\sigma) = \tilde{O}(Q(n)) + O(t_1)$, where t_1 is the output size. An important observation is that after covering $\partial\mathscr{R}$, we can express the remaining regions by the boundaries of the slabs used and G, which are linear inequalities and so we can use simplex range reporting. Lemma 8.4.2 characterizes the remaining regions. See Appendix 8.H for the proof.

Lemma 8.4.2. *There are $O(Q(n))$ remaining regions and each region can be expressed using $O(1)$ linear inequalities. These regions can be found in time $O(Q(n))$.*

With Lemma 8.4.2, the query time for the remaining regions is $\tilde{O}(Q(n)) + O(t_2)$, where t_2 is the number of points in the remaining regions. Then the total query time is easily computed to be bounded by $\tilde{O}(Q(n)) + O(k)$, where $k = t_1 + t_2$.

To bound the space usage for the top-level data structure, note that we have $Q(n)^2$ cells, for each α, we generate $\Theta(\frac{1/Q(n)}{\alpha/Q(n)}) = \Theta(1/\alpha)$ slabs for each of the $\Theta(Q(n))$ angles. Since points are distributed uniformly at random, the expected number of points in a slab of width $\alpha/Q(n)$ in a cell C is $O(n \cdot \frac{1}{Q(n)} \cdot \frac{\alpha}{Q(n)})$. So the space usage for the top-level data structure is

$$S(n) = \sum_\alpha Q(n)^2 \cdot \Theta\left(\frac{1}{\alpha}\right) \cdot \Theta(Q(n)) \cdot \tilde{O}\left(\frac{O\left(n \cdot \frac{1}{Q(n)} \cdot \frac{\alpha}{Q(n)}\right)}{Q(n)\alpha}\right)^{\mathbf{m}} = \tilde{O}\left(\frac{n^{\mathbf{m}}}{Q(n)^{3\mathbf{m}-4}}\right).$$

On the other hand, we know that the space usage for the bottom-level data structure is $\tilde{O}(n^2)$. So the total space usage is bounded by $\tilde{O}(\frac{n^{\mathbf{m}}}{Q(n)^{3\mathbf{m}-4}})$ for $\mathbf{m} \geq 3$.

We therefore obtain the following theorem.

Theorem 8.4.1. *Let $\mathscr{R}$ be the set of semialgebraic ranges formed by degree-Δ bivariate polynomials. Suppose we have a polynomial factorization black box that can factorize polynomials into the product of irreducible polynomials in time $O(1)$, then for any $\log^{O(1)} n \leq Q(n) \leq n^{\varepsilon}$ for some constant ε, and a set $\mathscr{P}$ of n points distributed uniformly randomly in $\mathscr{U} = [0,1] \times [0,1]$, we can build a data structure* *of space $\tilde{O}(n^{\mathbf{m}}/Q(n)^{3\mathbf{m}-4})$ such that for any $\mathscr{R} \in \mathscr{R}$, we can report $\mathscr{R} \cap \mathscr{P}$ in time $\tilde{O}(Q(n)) + O(k)$ in expectation, where $\mathbf{m} \geq 3$ is the number of parameters needed to define a degree-Δ bivariate polynomial and k is the output size.*

A Derivative-based Approach

If we assume that the derivative of $\partial\mathscr{R}$ is $O(1)$, the previous curvature-based approach can be easily adapted to get a derivative-based data structure. See Appendix 8.I for details. We can even do better by using slabs whose boundaries are the zero sets of higher degree polynomials instead of linear polynomials. Using Taylor's theorem, we show that we can cover the boundary of the query using "thin" slabs of lower degree polynomials, similar to the approach above. The full details are presented in Appendix 8.J.

Theorem 8.4.2. *Let $\mathscr{R}$ be the set of semialgebraic ranges formed by degree-Δ bivariate polynomials with bounded derivatives up to the Δ-th order. For any $\log^{O(1)} n \leq Q(n) \leq n^{\varepsilon}$ for some constant ε, and a set $\mathscr{P}$ of n points distributed uniformly randomly in $\mathscr{U} = [0,1] \times [0,1]$, we can build a data structure which uses space $\tilde{O}(n^{\mathbf{m}}/Q(n)^{((2\mathbf{m}-\Delta)(\Delta+1)-2)/2})$ s.t. for any $\mathscr{R} \in \mathscr{R}$, we can report $\mathscr{P} \cap \mathscr{R}$ in time $\tilde{O}(Q(n)) + O(k)$ in expectation, where $\mathbf{m}$ is the number of parameters needed to define a degree-Δ bivariate polynomial and k is the output size.*

Remark 8.4.1. *We remark that our data structure can also be adapted to support semialgebraic range searching queries in the semigroup model.*

8.5 Conclusion and Open Problems

In this paper, we essentially closed the gap between the lower and upper bounds of general semialgebraic range reporting in the fast-query case at least as far as the exponent of n is concerned. We show that for general polynomial slab queries defined by D-variate polynomials of degree at most Δ in $\mathbb{R}^D$ any data structure with query time $n^{o(1)} + O(k)$ must use at least $S(n) = \overset{o}{\Omega}(n^{\mathbf{m}})$ space, where $\mathbf{m} = \binom{D+\Delta}{D} - 1$ is the maximum possible parameters needed to define a query. This matches current upper bound (up to an $n^{o(1)}$ factor).

We also studied the space-time tradeoff and showed an upper bound that matches the lower bounds in [AC23b] for uniform random point sets.

The remaining big open problem here is proving a tight bound for the exponent of $Q(n)$ in the space-time tradeoff. There is a large gap between the exponents in our lower bound versus the general upper bound. Our results show that current upper bound might not be tight. On the other hand, our lower bound seems to be suboptimal when the query time is $n^{\Omega(1)} + O(k)$. Both problems seem quite challenging, and probably require new tools.

Appendices

8.A Proof of Theorem 9.2.1

Theorem 9.2.1. *Suppose the D-dimensional geometric range reporting problems admit an $S(n)$ space and $Q(n)+O(k)$ query time data structure, where n is the input size and k is the output size. Let $\mu^D(\cdot)$ denote the D-dimensional Lebesgue measure. (We call this D-measure for short.) Assume we can find $m = n^c$ ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots, \mathscr{R}_m$ in a D-dimensional cube $\mathscr{C}^D$ of side length $|l|$ for some constant c such that (i) $\forall i = 1,2,\cdots,m, \mu^D(\mathscr{R}_i \cap \mathscr{C}^D) \geq 4c|l|^D Q(n)/n$; and (ii) $\mu^D(\mathscr{R}_i \cap \mathscr{R}_j) = O(|l|^D/(n2^{\sqrt{\log n}}))$ for all $i \neq j$. Then, we have $S(n) = \overset{o}{\Omega}(mQ(n))$.*

First we present the original lower bound framework by Chazelle [Cha90a] and Chazelle and Rosenberg [CR96].

Theorem 8.A.1. *Suppose the D-dimensional geometric range reporting problems admit an $S(n)$ space $Q(n)+O(k)$ query time data structure, where n is the input size and k is the output size. Assume we can find m subsets $q_1, q_2, \cdots, q_m \subset \mathscr{S}$ for some input point set $\mathscr{S}$, where each $q_i, i = 1, \cdots, m$ is the output of some query and they satisfy the following two conditions: (i) for all $i = 1, \cdots, m$, $|q_i| \geq Q(n)$; and (ii) $|q_{i_1} \cap q_{i_2} \cap \cdots q_{i_\alpha}| \leq c$ for some value $c \geq 2$. Then, we have $S(n) = \Omega(\frac{\sum_{i+1}^m |q_i|}{\alpha 2^{O(c)}}) = \Omega(\frac{mQ(n)}{\alpha 2^{O(c)}})$.*

A common way to use this framework is through a "volume" argument, i.e., we generate a set of geometric ranges in a hypercube and then show that they satisfy the following two properties:

- Each range intersects the hypercube with large Lebesgue measure;
- The Lebesgue measure of the intersection of any k ranges is small.

Then if we sample n points uniformly at random in the hypercube, we obtain $\mathscr{S}$ in Theorem 8.A.1 in expectation. However, we generally want to show a lower bound for the worst case, then we need a way to derandomize to turn the result to a worst-case lower bound. We now introduce some derandomization techniques, which are direct generalizations of the 2D version of the derandomization lemmas in [AC23b]. Given a D-dimensional hypercube $\mathscr{C}^D$ of side length $|l|$ and a set of ranges. The first lemma shows that when each range intersects $\mathscr{C}^D$ with large D-dimensional Lebesgue measure (For simplicity, we will call such a measure D-measure and denoted by $\mu^D(\cdot)$.) and the number of ranges is not too big, then with high probability, each range will contain many points.

Lemma 8.A.1. *Let $\mathscr{C}^D$ be a hypercube of side length $|l|$ in $\mathbb{R}^D$. Let $\mathscr{R}$ be a set of ranges in $\mathscr{C}^D$ satisfying two following conditions: (i) the D-measure of the intersection of any range $\mathscr{R} \in \mathscr{R}$ and $\mathscr{C}^D$ is at least $c|l|^D t/n$ for some constant $c \geq 4k$ and a parameter $t \geq \log n$ for some value $k \geq 2$; (ii) the total number of ranges is bounded by $O(n^{k+1})$. Now if we sample a set $\mathscr{P}$ of n points uniformly at random in $\mathscr{C}^D$, then with probability $> 1/2$, $|\mathscr{P} \cap \mathscr{R}| \geq t$ for all $\mathscr{R} \in \mathscr{R}$.*

Proof. We pick n points in $\mathscr{C}^D$ uniformly at random. Let X_{ij} be the indicator random variable with

$$X_{ij} = \begin{cases} 1, \text{point } i \text{ is in range } j, \\ 0, \text{otherwise.} \end{cases}$$

Since $\mu^D(\mathscr{R}) \geq c|l|^D t/n$ for every $\mathscr{R} \in \mathscr{R}$, the expected number of points in each range is at least ct. Consider an arbitrary range, let $X_j = \sum_{i=1}^n X_{ij}$, then by Chernoff's bound

$$\Pr\left[X_j < \left(1 - \frac{c-1}{c}\right) ct\right] < e^{-\frac{\left(\frac{c-1}{c}\right)^2 ct}{2}}$$
$$\implies \Pr[X_j < t] < e^{-\frac{(c-1)^2 t}{2c}} < \frac{1}{n^{\frac{(c-1)^2}{2c}}} \leq \frac{1}{n^{2k-1+1/(8k)}},$$

where the second last inequality follows from $t \geq \log n$ and the last inequality follows from $c \geq 4k$. Since the total number of ranges $O(n^{k+1})$, by the union bound, for $k \geq 2$ and a sufficiently large n, with probability $> \frac{1}{2}$, $|\mathscr{P} \cap \mathscr{R}| \geq t$ for all $\mathscr{R} \in \mathscr{R}$. □

The second lemma tells a different story: when the D-measure of the intersection of any k ranges is small, and the number of intersection is not too big, then with high probability, each intersection has very few points.

Lemma 8.A.2. *Let $\mathscr{C}^D$ be a hypercube of side length $|l|$ in $\mathbb{R}^D$. Let $\mathscr{R}$ be a set of ranges in $\mathscr{C}^D$ satisfying the following two conditions: (i) the D-measure of the intersection of any $t \geq 2$ distinct ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots, \mathscr{R}_t \in \mathscr{R}$ is bounded by $O(|l|^D/(n2^{\sqrt{\log n}}))$; (ii) the total number of intersections is bounded by $O(n^{2k})$ for $k \geq 1$. Now if we sample a set $\mathscr{P}$ of n points uniformly at random in $\mathscr{C}^D$, then with probability $> 1/2$, $|\mathscr{R}_1 \cap \mathscr{R}_2 \cap \cdots \cap \mathscr{R}_t \cap \mathscr{P}| < 3k\sqrt{\log n}$ for all distinct ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots, \mathscr{R}_t \in \mathscr{R}$.*

Proof. We consider the intersection $\rho \in \mathscr{C}^D$ of any t ranges and let $A = \mu^D(\rho)$. Let X be an indicator random variable with

$$X_i = \begin{cases} 1, \text{the } i\text{-th point is inside } \rho, \\ 0, \text{otherwise.} \end{cases}$$

Let $X = \sum_{i=1}^{n} X_i$. Clearly, $\mathbb{E}[X] = \frac{An}{|l|^D}$. By Chernoff's bound,

$$\Pr\left[X \geq (1+\delta)\frac{An}{|l|^D}\right] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\frac{An}{|l|^D}},$$

for any $\delta > 0$. Let $\tau = (1+\delta)\frac{An}{|l|^D}$, then

$$\Pr[X \geq \tau] < \frac{e^{\delta\frac{An}{|l|^D}}}{(1+\delta)^{\tau}} < \frac{e^{\tau}}{(1+\delta)^{\tau}} = \left(\frac{eAn}{|l|^D\tau}\right)^{\tau}.$$

Let $\tau = 3k\sqrt{\log n}$, since $A \leq c|l|^D/(n2^{\sqrt{\log n}})$ for some constant c, we have

$$\Pr\left[X \geq 3k\sqrt{\log n}\right] < \left(\frac{ce}{2^{\sqrt{\log n}}3k\sqrt{\log n}}\right)^{3k\sqrt{\log n}} < \frac{(ce)^{3k\sqrt{\log n}}}{n^{3k}}.$$

Since the total number of intersections is bounded by $O(n^{2k})$, the number of cells in the arrangement is also bounded by $O(n^{2k})$ and thus by the union bound, for sufficiently large n, with probability $> \frac{1}{2}$, the number of points in every intersection region is less than $3k\sqrt{\log n}$. □

We now prove Theorem 9.2.1.

Proof. We sample a set $\mathscr{P}$ of n points uniformly at random in $\mathscr{C}^D$. Since each range $\mathscr{R}_i$ has $\mu^D(\mathscr{R}_i) \geq 4c|l|^D Q(n)/n$, and the number of ranges is $m = n^c$, then by Lemma 8.A.1, with probability more than 1/2, $|\mathscr{P} \cap \mathscr{R}_i| \geq Q(n)$ for all $i = 1, 2, \cdots, m$. Since the intersection of any two ranges is upper bounded by $O(|l|^D/(n2^{\sqrt{\log n}}))$ and the total number of intersections is $O(m^2) = O(n^{2c})$, then by Lemma 8.A.2, with probability more than 1/2, $\left|\mathscr{R}_i \cap \mathscr{R}_j \cap \mathscr{P}\right| = O(\sqrt{\log n})$ for distinct ranges $\mathscr{R}_i, \mathscr{R}_j$. By the union bound, there is a point set such that both conditions in Theorem 8.A.1 are satisfied, then we obtain a lower bound of

$$S(n) = \Omega\left(\frac{mQ(n)}{2 \cdot 2^{O(\sqrt{\log n})}}\right) = \overset{o}{\Omega}(mQ(n)).$$

□

8.B Proof of Observation 8.3.1

Observation 8.3.1. *Let P be a packed D-variate polynomial and $a = (a_1, a_2, \cdots, a_D) \in Z(P)$. If $a_i \in [1,2]$ for all $i = 2, 3, \cdots, D$, then there exists a unique a_1 such that $0 < a_1 = O(1)$.*

Proof. We only need to show that there exists only one solution to equation $0 = a_1 - a_2^{\Delta} + f(a_1)$ when $a_1 > 0$ and the solution has value $O(1)$, where $f(a_1)$ is a polynomial in a_1 with nonnegative coefficients. Since $1 \leq a_2 \leq 2$, it easily follows. □

8.C Proof of Lemma 8.3.3

Lemma 8.3.3. *Let $P_1 \in \mathscr{S}$ and $P_2 = P_1 - r$ for any $0 \le r = O(w)$. Then $\pi(Z(P_1), Z(P_2), X) = \Theta(r)$ for any $X \in [1,2]^{D-1}$.*

Proof. Pick any point $p = (p_1, p_2, \cdots, p_D) \in Z(P_1)$, and $p' = (p'_1, p_2, \cdots, p_D) \in Z(P_2)$ such that $p_i \in [1,2]$ for all $i = 2,3,\cdots,D$, and $p'_1 = p_1 + \gamma$. Clearly, $0 < \gamma < 1$ because $0 \le r = O(w) = o(1)$. By definition

$$P_2(p') = p_1 + \gamma + p_2^D + \left(\sum_i A_i (p_1+\gamma)^{i_1} p_2^{i_2} p_3^{i_3} \cdots p_D^{i_D}\right) - r = P_1(p) + \Theta(\gamma) - r = 0.$$

So $\gamma = \Theta(r)$, meaning, $\pi(Z(P_1), Z(P_2), p) = \Theta(r)$ for $X \in [1,2]^{D-1}$. □

8.D Proof of Lemma 8.3.4

Lemma 8.3.4. *Let $P \in \mathscr{S}$. The projection of $Z(P) \cap \mathscr{U}^D$ on the $(D-1)$-dimensional space perpendicular to the X_1-axis has $(D-1)$-measure $\Theta(1)$.*

We first bound the length of the y-interval within which a packed bivariate can intersect $\mathscr{U}^2$.

Lemma 8.D.1. *Let P be a packed bivariate polynomial. Then $\sigma = Z(P)$ is fully contained in $\mathscr{U}^2$ for some y-interval of length $\Theta(1)$.*

Proof. We show that σ is sandwiched by curves $\sigma_l : x - y^\Delta + c\varepsilon = 0$ for some sufficiently large constant c and $\sigma_r : x - y^\Delta = 0$ in $\mathscr{U}^2$. We intersect $\sigma_l, \sigma, \sigma_r$ with line $y = y_*$ for $y_* \in [1,2]$ and denote the intersections to be $(x_l, y_*), (x_m, y_*), (x_r, y_*)$ respectively. Since σ is of form $x - y^\Delta + \sum_{i=0}^{\Delta} \sum_{j=0}^{\Delta-i} c_{ij} x^i y^j = 0$, $x_m = y_*^\Delta - O(\varepsilon)$ because $0 \le c_{ij} = O(\varepsilon)$ and $0 < x = O(1)$ when $y_* \in [1,2]$ by Observation 8.3.1. So for sufficiently large c, $x_l \le x_m \le x_r$. It is elementary to compute that σ_l and σ_r intersect $x = 1$ at point $(1, \sqrt[\Delta]{1+c\varepsilon}), (1,1)$ respectively, and intersect $x = 2$ at point $(2, \sqrt[\Delta]{2+c\varepsilon}), (2, \sqrt[\Delta]{2})$ respectively in the first quadrant. So the intersection of σ with $x = 1$ (resp. $x = 2$) has y-value between 1 and $\sqrt[\Delta]{1+c\varepsilon}$ (resp. $\sqrt[\Delta]{2}$ and $\sqrt[\Delta]{2+c\varepsilon}$). So the projection of $\sigma \cap \mathscr{U}^2$ onto the y-axis has length at least $\sqrt[\Delta]{2} - \sqrt[\Delta]{1+c\varepsilon}$. Since $\varepsilon = o(1)$, the lemma holds. □

Now we prove Lemma 8.3.4.

Proof. We intersect $Z(P)$ with $X_i = a_i \in [1,2]$ for $i = 3,4,\cdots,D$. The resulting polynomial will be a packed bivariate polynomial. By Lemma 8.D.1, we know the intersection of the zero set of this bivariate polynomial and $\mathscr{U}^2$ has 1-measure $\Theta(1)$ in the X_2-axis. Integrating over all X_i for $i = 3,4,\cdots,5$, $Z(P)$ intersects $\mathscr{U}^D$ with $(D-1)$-measure $\Theta(1)$ in the subspace perpendicular to the X_1-axis. □

8.E Lower Bound for Sphere Shell Ranges

In this section, we show a lower bound for range reporting with ranges being sphere shells in higher dimensions. We first give a definition for D-dimensional sphere shells.

Definition 8.E.1. *A D-dimensional sphere shell, or a D-sphere shell for short, is a tupe (O, r, w), where $O \in \mathbb{R}^D$ is a point and $r, w \in \mathbb{R}$ are two real values. We call O the center of the sphere shell, r the radius of the sphere shell, and w the width of the sphere shell. A D-dimensional sphere shell is defined to be the set of points with distance between r and $r+w$ to point O, i.e., $\{p \in \mathbb{R}^D : r \le \|p-O\|_2 \le r+w\}$.*

In our proof, we will need the following geometric lemma to bound the intersection measure of two D-sphere shells.

Lemma 8.E.1. *Let (O_1, r_1, w) and the(O_2, r_2, w) be two D-sphere shells, where $D \ge 3$, $r_1, r_2 = \Theta(n)$ and $w = o(n)$. Let $d = \|O_1 - O_2\|_2 > w$ be the distance between points O_1, O_2. Then the D-measure of the intersection of (O_1, r_1, w) and the(O_2, r_2, w) is bounded by $O(n^{D-1}w^2/d)$.*

Proof. We bound the D-measure of the intersection I of two D-sphere shells $\mathscr{S}_1, \mathscr{S}_2$ as follows. W.l.o.g, we assume the center of $\mathscr{S}_1$ is the origin, and the center of $\mathscr{S}_2$ is $(d, 0, 0, \cdots, 0)$, i.e., the first coordinate is d and all other $D-1$ coordinates are 0's.

To bound the intersection, we consider two hyperplanes $x_1 = a_b$ and $x_1 = a_e$ such that the intersection region of $\mathscr{S}_1, \mathscr{S}_2$ is sandwiched by the two hyperplanes. Note that all points $p = (x_1, x_2, \cdots, x_D)$ in the intersection must satisfy

$$\begin{cases} r_1^2 \le \sum_{i=1}^D x_i^2 \le (r_1+w)^2 \\ r_2^2 \le (x_1-d)^2 + \sum_{i=2}^D x_i^2 \le (r_2+w)^2 \end{cases}.$$

This implies

$$\begin{cases} r_1^2 - x_1^2 \le \sum_{i=2}^D x_i^2 \le (r_1+w)^2 - x_1^2 \\ r_2^2 - (x_1-d)^2 \le \sum_{i=2}^D x_i^2 \le (r_2+w)^2 - (x_1-d)^2 \end{cases}.$$

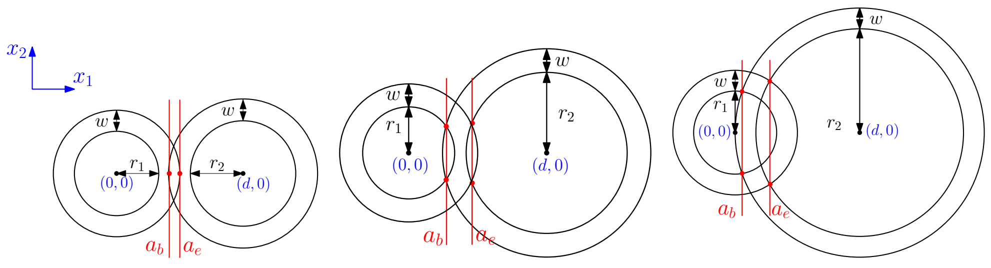
So for the intersection to be nonempty, the first coordinates of all points in the intersection region must satisfy

$$\begin{cases} r_1^2 - x_1^2 \le (r_2+w)^2 - (x_1-d)^2 \\ r_2^2 - (x_1-d)^2 \le (r_1+w)^2 - x_1^2 \end{cases}.$$

They together imply that

$$\frac{r_1^2 + d^2 - (r_2+w)^2}{2d} \le x_1 \le \frac{(r_1+w)^2 + d^2 - r_2^2}{2d},$$

which gives us the values for a_b and a_e. It suffices to bound the D-measure of the part of $\mathscr{S}_1$ between $x_1 = a_b$ and $x_1 = a_e$. Since $d > w > 0$, $a_e - a_b \le w(r_1 + r_2 + w)/d <$



(a) Case 1. (b) Case 2. (c) Case 3.

Figure 8.E.1: Project Two Sphere Shells to the $x_1 - x_2$ Plane.

$r_1 + r_2 + w$. There are five cases. First, when $r_1 \le a_b < r_1 + w = a_e$, see Figure 8.E.1a for an example, the intersection measure is easily bounded by

$$
\begin{aligned}
\mu^D(I) &\le \int_{a_b}^{r_1+w} c\left(\sqrt{(r_1+w)^2 - x_1^2}\right)^{D-1} \mathrm{d}x_1 \\
&\le \int_{a_b}^{r_1+w} c\left(\sqrt{(r_1+w)^2 - r_1^2}\right)^{D-1} \mathrm{d}x_1 \\
&= O((r_1 w)^{\frac{D-1}{2}} \cdot \frac{r_1 w}{d}) \\
&= O(\frac{(r_1 w)^{\frac{D+1}{2}}}{d}) = O(\frac{r_1^{D-1} w^2}{d}) = O(\frac{n^{D-1} w^2}{d}),
\end{aligned}
$$

where the first inequality follows from the $(D-1)$-measure of a $(D-1)$-dimensional ball and c is the positive constant in the measure formula, the second inequality follows from $r_1 \le a_b \le x_1$, and the last equality follows from $D \ge 3$, $r_1 = \Theta(n)$, and $w = o(n)$.

Second, when $a_b < r_1 < a_e < r_1 + w$, see Figure 8.E.1b for an example, the intersection measure is bounded by

$$\begin{aligned}
\mu^D(I) &\le \int_{a_b}^{r_1} c\left(\left(\sqrt{(r_1+w)^2-x_1^2}\right)^{D-1} - \left(\sqrt{r_1^2-x_1^2}\right)^{D-1}\right)\mathrm{d}x_1 \\
&+ \int_{r_1}^{a_e} c\left(\left(\sqrt{(r_1+w)^2-x_1^2}\right)^{D-1}\right)\mathrm{d}x_1 \\
&\le \int_{a_b}^{r_1} \frac{D-1}{2} c\left((r_1+w)^2-x_1^2\right)^{\frac{D-1}{2}-1}\left(2r_1w+w^2\right)\mathrm{d}x_1 \\
&+ \int_{r_1}^{a_e} c\left(\left(\sqrt{(r_1+w)^2-r_1^2}\right)^{D-1}\right)\mathrm{d}x_1 \\
&\le \int_{a_b}^{r_1} \frac{D-1}{2} c\,(r_1+w)^{D-3}\left(2r_1w+w^2\right)\mathrm{d}x_1 \\
&+ \int_{r_1}^{a_e} c\left(\sqrt{2r_1w+w^2}\right)^{D-1}\mathrm{d}x_1 \\
&= O\left(r_1^{D-1}\frac{w^2}{d}\right) + O\left((\sqrt{r_1w})^{D-1}w\right) = O\left(n^{D-1}\frac{w^2}{d}\right),
\end{aligned}$$

where c is a positive constant and the second inequality follows from $a^n - b^n \le na^{n-1}(a-b)$ for $a,b,c \in \mathbb{R}$ and $a \ge b$, $n \ge 1$.

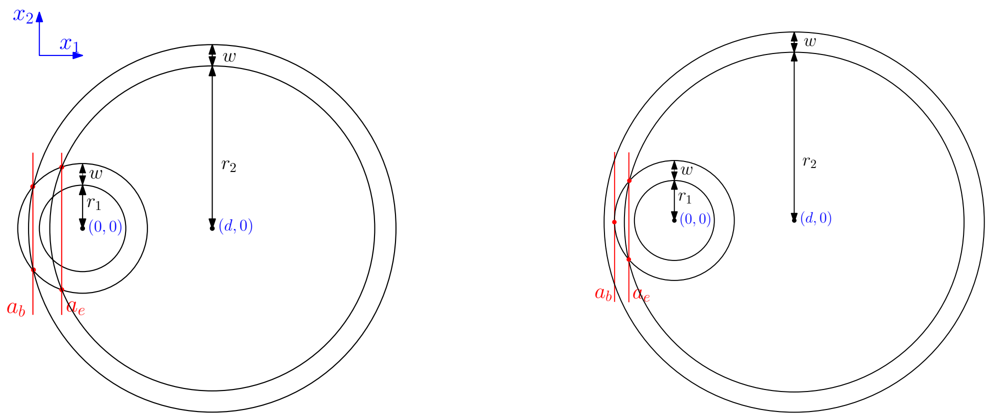
Third, when $-r_1 < a_b < a_e \le r_1$, see Figure 8.E.1c for an example, we bound the intersection measure by

$$\begin{aligned}
\mu^D(I) &\le \int_{a_b}^{a_e} c\left(\left(\sqrt{(r_1+w)^2-x_1^2}\right)^{D-1} - \left(\sqrt{r_1^2-x_1^2}\right)^{D-1}\right)\mathrm{d}x_1 \\
&\le \int_{a_b}^{a_e} \frac{D-1}{2} c\left((r_1+w)^2-x_1^2\right)^{\frac{D-1}{2}-1}\left(2r_1w+w^2\right)\mathrm{d}x_1 \\
&\le \int_{a_b}^{a_e} \frac{D-1}{2} c\,(r_1+w)^{D-3}\left(2r_1w+w^2\right)\mathrm{d}x_1 \\
&= O\left(r_1^{D-1}\frac{w^2}{d}\right), \\
&= O\left(n^{D-1}\frac{w^2}{d}\right),
\end{aligned}$$

where c is a positive constant.

Fourth, when $-r_1 - w \le a_b \le -r_1 < a_e$, see Figure 8.E.2a for an example, note that it is symmetric to the second case and so the bound follows. Finally, when $a_b = -r_1 - w < a_e \le -r_1$, see Figure 8.E.2b for an example, note that it is symmetric to the first case and so the bound follows. and the bound for this case is symmetric to the second case. Therefore we get $\mu^D(I) = O\left(n^{D-1}\frac{w^2}{d}\right)$ for all cases.

□



(a) Case 4. (b) Case 5.

Figure 8.E.2: Project Two Sphere Shells to the $x_1 - x_2$ Plane.

We again use Framework 9.2.1 to show a lower bound. We consider the following construction. Let $\mathscr{C}_1^D = \prod_{i=1}^D [0,n]$ be a hypercube in D dimensions. Let $\mathscr{C}_2^D = [-(D+1)n, -Dn] \times \prod_{i=2}^D [0,n]$ be a D dimensional hypercube to the left of $\mathscr{C}_1^D$ with respect to the x_1 axis.

For each of the D dimensions, we cut $\mathscr{C}_1^D$ into n/d evenly spaced slices. This gives us a grid consisting of $\Theta((n/d)^D)$ grid points. For each grid point p with x_1 being kd for $k = 0,1,2,\cdots,\lfloor n/d \rfloor$, we construct $\Theta(n/w)$ D-sphere shells as follows. We set its beginning radius $r = r_b = (\sqrt{(kd/n+D)^2+D-1})n$ and construct a series of $(D-1)$-spheres with p being the center and r being the radius, and then increase the radius by w each time until reaching $r = r_e = (kd/n+D+1)n$. The region between two consecutive $(D-1)$-spheres forms a D-sphere shell. Clearly, this construction is valid since $r_b < r_e$. The following lemma bounds the D-measure of the intersection of $\mathscr{C}_2^D$ and a slab.

Lemma 8.E.2. *Any sphere shell intersects $\mathscr{C}_2^D$ with D-measure $\Omega(n^{D-1}w)$.*

Proof. To see this, consider any grid point $p = (kd, x_2, \cdots, x_D)$ in $\mathscr{C}_1^D$. The distance between any point q in $\mathscr{C}_2^D$ with its first coordinate being $-Dn$ and p is upper bounded by

$$\|p-q\|_2 = \sqrt{(kd+Dn)^2 + \sum_{j=2}^{D} (x_j - x_j')^2} \le (\sqrt{(kd/n+D)^2+D-1})n. \tag{8.1}$$

Similarly, the distance between any point q' in $\mathscr{C}_2^D$ with first coordinate being $-(D+1)n$ and p is lower bounded by

$$\|p-q'\|_2 = \sqrt{(kd+(D+1)n)^2 + \sum_{j=2}^{D} (x_j - x_j')^2} \ge (kd/n+D+1)n. \tag{8.2}$$

Inequalities (8.1) and (8.2) together imply that a $D-1$ dimensional sphere with p being the center and r being the distance, $r_b \le r \le r_e$, intersects $\mathscr{C}_2^D$ with $(D-1)$-measure $\Omega(n^{D-1})$. So a sphere shell formed by two such spheres with distance w has D-measure $\Omega(n^{D-1}w)$. See Figure 8.E.3 for an example. □

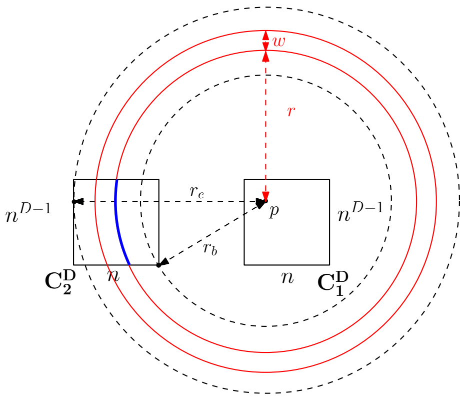


Figure 8.E.3: Generating Sphere Shells (Projection to the $x-y$ plane): Each sphere generated intersect $\mathscr{C}_2^D$ with $(D-1)$-measure $\Omega(n^{D-1})$ (the blue part). The D-measure of the intersection of $\mathscr{C}_2^D$ and a sphere shell is $\Omega(n^{D-1}w)$.

In total, we have constructed $\Theta((n/d)^D \cdot (r_e - r_b)/w) = \Theta(n^{D+1}/(d^D w))$ D-sphere shells. We set $w = c_w Q(n)$ for some sufficiently large constant c_w and $d = w^2 2^{\sqrt{\log n}}$. By Lemma 8.E.2, the intersection D-measure of any sphere shell and $\mathscr{C}_2^D$ is at least $4(D+1)n^{D-1}Q(n)$ for large enough constant c_w and that of two shells is upper bounded by $O(n^{D-1}/2^{\sqrt{\log n}})$. Then by Theorem 9.2.1, $S(n) = \Omega\left(\frac{n^{D+1}Q(n)}{d^D w}\right) = \overset{o}{\Omega}\left(\frac{n^{D+1}}{Q(n)^{2D}}\right)$.

So we obtain the following lower bound.

Theorem 8.3.2. *Any data structure that solves sphere shell reporting problems in D dimensions on a point set of size n with query time $Q(n) + O(k)$, where $D \ge 3$ and k is the output size, must use $S(n) = \overset{o}{\Omega}(n^{D+1}/Q(n)^{2D})$ space.*

8.F Total Absolute Curvature of the Zero Set of Irreducible Polynomials

In this section, we prove the following lemma.

Lemma 8.F.1. *Let P be an irreducible bivariate polynomial of constant degree. Then $Z(P)$ has total absolute curvature $O(1)$.*

We first show for any value $v \in \mathbb{R} \cup \{\pm\infty\}$, the number of points on $Z(P)$ whose derivative achieves this value is $O(1)$.

We will use Bézout's Thoerem.

Theorem 8.F.1 (Bézout's Theorem)**.** *Given 2 polynomials $P(x,y)$ and $Q(x,y)$ of degree Δ_p and Δ_q respectively, either the number of common zeroes of P and Q is at most $\Delta_p \cdot \Delta_q$ or they have a common factor.*

Now we show any irreducible polynomial has $O(1)$ points achieving the same derivative.

Lemma 8.F.2. *Let $P(x,y)$ be an irreducible bivariate polynomial of degree $\Delta > 1$. Then the number of points on $Z(P(x,y))$ which have a fixed derivative c is bounded by $O(\Delta^2)$.*

Proof. For simplicity, we first rotate $P(x,y)$ such that the fixed derivative is 0. Let us denote the new polynomial with $Q(x,y)$ and it is easy to see that Q is also irreducible since if Q could be written as $Q(x,y) = R(x,y)S(x,y)$, then $P(x,y)$ would also have a similar decomposition.

By differentiating Q, we get that $\frac{\mathrm{d}y}{\mathrm{d}x} = -\frac{Q_x(x,y)}{Q_y(x,y)} = 0$, and thus $Q_x(x,y) = 0$. As a result, any point (x,y) with derivative 0, lies on the zero set of Q and Q_x.

Both Q and Q_x have degree $O(\Delta)$ and since Q is irreducible and degree of Q_x is at least one, they cannot have a common factor. By Bézout's Theorem, this implies that they have $O(\Delta^2)$ common zeroes. □

We now prove Lemma 8.F.1. More specifically, we prove the following:

Lemma 8.F.3. *Consier a smooth curve C such that for any value v, there are at most k points p on C such that the tangent line at p has slope v. Then C has total absolute curvature $O(k^2)$.*

Proof. We parametrize $P(x,y) = 0$ by its arc length s over an interval I and then consider the function $\alpha : \mathbb{R} \to \mathbb{R}$ be a function that maps the arc length of the curve to the angle of the curve. Note that $\alpha(s)$ is allowed to increase beyond 2π. Let α_1 and α_2 be the infimum and surpremum of $\alpha(s)$ over $s \in I$. Note that we must have $\alpha_2 - \alpha_1 \le k2\pi$ as otherwise we can find more then k points with the same slope on C. $\alpha'(s)$ determines the curvature of the curve at point s and its total curvature is

$$\int_I |\alpha'(s)| ds \le 2\pi k^2$$

where the inequality follows from the observation that the equation $\alpha(s) = v$ for every v has at most k solutions and thus the total change in $\alpha(s)$ is bounded by $k \cdot |\alpha_2 - \alpha_1| \le 2\pi k^2$. □

Lemma 8.F.1 then follows easily by Lemma 8.F.2 and 8.F.3.

8.G Proof of Lemma 8.4.1

Lemma 8.4.1. *We can cut $\partial\mathscr{R}$ into a set $\mathscr{S}$ of $O(Q(n))$ sub-curves such that for each sub-curve σ, we can find a set S_σ of slabs that together cover σ. Let P_σ be the subset of the input that lies inside the query and inside the slabs, i.e., $P_\sigma = \mathscr{R} \cap \mathscr{P} \cap (\cup_{s \in S_\sigma} s)$. P_σ can be reported in time $Q(n)\tilde{O}(\kappa_\sigma + 1/Q(n)) + O(|P_\sigma|)$, where κ_σ is the total absolute curvature of σ. Furthermore, for any two distinct $\sigma_1, \sigma_2 \in \mathscr{S}$, $s_1 \cap s_2 = \emptyset$ for all $s_1 \in S_{\sigma_1}, s_2 \in S_{\sigma_2}$.*

Now suppose we have a sub-curve $\sigma \subset \partial\mathscr{R}$ in C that contains no singular points (points with undefined derivatives) except for possible the two boundaries, if the total absolute curvature is between 0 and $\pi/4$, then we can efficiently find $O(1)$ slabs to cover it as shown in the following lemma.

Lemma 8.G.1. *Let σ be any differentiable sub-curve in a cell C with total absolute curvature κ_σ such that $0 \leq \kappa_\sigma \leq \pi/4$. We can find a set of $O(1)$ slabs of width $O(\kappa_\sigma/Q(n) + 1/Q(n)^2)$ that together cover σ and these slabs can be found in time $\tilde{O}(1)$.*

Proof. Let p and q be the end points of the curve σ. Consider the point r furthest away from the line pq on the curve. See Figure 8.G.1 for an example. Observe that we can use the mean value theorem between p and r and also between r and q. This yields that the sum of the angles $\angle rpq + \angle rqp$ is at most the total absolute curvature of σ. Since p, q are in C, $|\overline{pq}| = O(1/Q(n))$ and since $\angle rpq, \angle rqp \leq \kappa_\sigma \leq \pi/4$, it follows that the distance between the line tangent to r and $\overline{pq}$ is $O(\kappa_\sigma/Q(n))$. Finally, notice that in our construction, we have created slabs of orientation $i/Q(n)$ for every integer i. As a result, we can cover σ with $O(1)$ slabs of width $O(\kappa_\sigma/Q(n) + 1/Q(n)^2)$. To find the slabs, we can use any of the previous techniques in semialgebraic range searching since the input size (i.e., the number of slabs) in our construction is $Q(n)^{O(1)}$. □

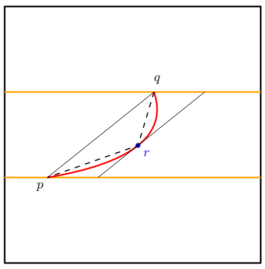


Figure 8.G.1: Covering a Sub-curve Using Slabs

We now show how to decompose $\partial\mathscr{R}$. Observe that $\partial\mathscr{R}$ intersects $O(Q(n))$ cells in G because otherwise $\partial\mathscr{R}$ will have $\omega(1)$ tangents, which contradicts Bézout's theorem. We cut $\partial\mathscr{R}$ using these $O(Q(n))$ cells to get $\mathscr{S}$.

Let $\sigma \subset \partial\mathscr{R}$ be the sub-curve in a cell $C \in G$. To find slabs to cover σ, we refine σ to be smaller pieces of curves to use Lemma 8.G.1. We simply cut σ into pieces such that each piece has total absolute curvature $\leq \pi/4$ and contains no singular points. Recall that the singular points of the zero set of a bivariate polynomial is a point where both partial derivatives are 0. By Bézout's theorem, there are $O(1)$ singular points. Since the total curvature of $\partial\mathscr{R}$ is $O(1)$, we will get $O(1)$ refined sub-curves. This part is easy with the assumption of our model of computation and so we omit the details about how to cut σ.

Now for each (refined) sub-curve σ_r, by Lemma 8.G.1 we can find $O(1)$ slabs to cover it. We report points close to σ_r as follows. First we sort the slabs in some order. Let s be a slab we find for σ_r. When we examine s, we use the data structure built in s to find the points in $\mathscr{R}$. The query time will be $Q(n)\tilde{O}(\kappa_{\sigma_r} + 1/Q(n)) + O(k)$ by Lemma 8.G.1. Before reporting the point, we check if the point has been reported in slabs we have examined before. This is because the slabs we found may intersect. But since we have $O(1)$ refined sub-curves for σ and each refined sub-curve requires $O(1)$ slabs to cover, it takes only $O(1)$ time to check for duplicates. Summing up the query cost for all refined sub-curves for σ, the total query time is $Q(n)\tilde{O}(\kappa_\sigma + 1/Q(n)) + O(t_\sigma)$. Since cells in G are disjoint and each slab is built only for a specific cell, the slabs we find for two distinct sub-curves will have zero intersection. This proves Lemma 8.4.1.

8.H Proof of Lemma 8.4.2

Lemma 8.4.2. *There are $O(Q(n))$ remaining regions and each region can be expressed using $O(1)$ linear inequalities. These regions can be found in time $O(Q(n))$.*

There are two types of remaining regions. First, cells fully contained in $\mathscr{R}$ but do not intersect $\partial\mathscr{R}$. Second, the regions in a cell intersected by $\partial\mathscr{R}$ but not covered by slabs.

We first handle the first type. For any two adjacent vertical lines l_1, l_2 in the grid G, we find all the cells between them intersected by $\partial\mathscr{R}$ in decreasing order with respect to their y-coordinates. For two consecutive cells C_1, C_2 we find, all the cells between C_1, C_2 must be all contained or all not contained in $\mathscr{R}$ because otherwise C_1, C_2 are not adjacent. We then express the union of cells in between C_1, C_2 using four linear inequalities. By this, we can find all the cells intersecting $\partial\mathscr{R}$ and all the chunks of cells fully contained in $\mathscr{R}$ between l_1, l_2. We do this for every consecutive pair of vertical lines. The number of chunks is linear to the number of cells intersecting $\partial\mathscr{R}$ which is $O(Q(n))$ by Bézout's theorem, so we have $O(Q(n))$ chunks as well. See Figure 8.H.1 (a) for an example.

For the second type, observe that each such region is defined by the boundaries of C (and/or) the outermost boundaries of slabs we used to cover sub-curves. Since by the analysis of Lemma 8.4.1, the sub-curve in a cell C requires only $O(1)$ slabs to cover. The outmost boundaries of these $O(1)$ slabs form a subdivision of complexity $O(1)$. Since each face in the subdivision is either fully contained in $\mathscr{R}$ or not contained in $\mathscr{R}$, it suffices to check an arbitrary point in the face. We omit the details here. In one cell, we have $O(1)$ remaining regions (faces in the subdivision) and it takes $O(1)$

time to find it. Since $\partial\mathscr{R}$ intersects $O(Q(n))$ regions, there are $O(Q(n))$ regions in total and it takes $O(Q(n))$ time to find them. See Figure 8.H.1 (b) for an example. This proves Lemma 8.4.2.

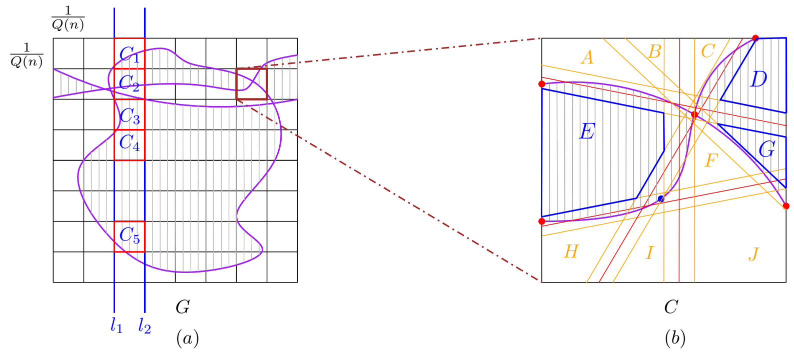


Figure 8.H.1: To Answer a Query (Shaded Region): (a): Finding cells fully contained in $\partial\mathscr{R}$. We have a chunk of zero cell between pairs C_1, C_2 as well as pairs C_2, C_3 and C_3, C_4, and a chunk of two cells between C_4, C_5. (b): Covering a sub-curve σ in a cell. Red dots are singular points of $\partial\mathscr{R}$ and its intersections with C. The blue dots is used to make sure each refined sub-curve has total absolute curvature $\leq \pi/4$. We use slabs (denoted by orange/red line segments) to cover the boundaries of σ. There are 10 regions in the subdivision formed by the outmost boundaries of slabs. Three of them (D, E, G) are fully contained in $\mathscr{R}$.

8.I An $S(n) = \tilde{O}(\frac{n^{\mathbf{m}}}{Q(n)^{3\mathbf{m}-4}})$ Derivative-based Data Structure

The data structure is similar to the curvature-based one. We also build a two-level data structure. For each cell C, we "guess" $Q(n)$ first derivatives $\alpha_1 = -c, -c+t, -c+2t, \cdots, c$, for a big enough constant c and $t = 2c/Q(n)$. For each guess α_1, we generate a series of disjoint parallel slabs each of (vertical) width $w_v = 1/Q(n)^2$ that together cover C such that the boundary of each slab has derivative α_1. Since $|\alpha_1| = O(1)$, the angle γ between any slab and the x-axis is also $O(1)$, so the width of each slab is $w = w_v \cdot \cos\gamma = \Theta(w_v)$. Therefore the total number of slabs we generate for each α_1 in a cell is $\frac{\Theta(1/Q(n))}{\Theta(w_v)} = O(Q(n))$. For each slab, we collect the points in it and build an $\tilde{O}(1) + O(k)$ query time and $\tilde{O}(n^{\mathbf{m}})$ space data structure. This is our top-level data structure. For the bottom-level data structure, we still use a multilevel simplex range reporting data structure with $\tilde{O}(n^2)$ space and $\tilde{O}(1) + O(k)$ query time.

The space usage for the top level data structure is easily bounded to be

$$S_1(n) = \tilde{O}\left(Q(n)^2 \cdot \frac{2c}{2c/Q(n)} \cdot O(Q(n)) \cdot \left(\frac{1}{Q(n)^2} \cdot \frac{1}{Q(n)} \cdot n\right)^{\mathbf{m}}\right) = \tilde{O}\left(\frac{n^{\mathbf{m}}}{Q(n)^{3\mathbf{m}-4}}\right).$$

Since the bottom level data structure takes up $\tilde{O}(n^2)$ space. The total space usage is $\tilde{O}(\frac{n^{\mathbf{m}}}{Q(n)^{3\mathbf{m}-4}})$ for $\mathbf{m} \geq 3$.

For the query answering, we prove a lemma similar to Lemma 8.G.1.

Lemma 8.I.1. *In our construction, if some differentiable sub-curve σ is contained in some cell C, then we can find $O(1)$ slabs that together cover σ. The time needed to find all these slabs is $\tilde{O}(1)$.*

Proof. Let $(p_x, p_x), (q_x, q_y)$ be the left and right endpoints of σ and $\frac{\mathrm{dy}}{\mathrm{dx}}(p_x, p_y) = \alpha_1^*$. Let $f(x)$ be the implicit function defined by σ between (p_x, p_y) and (q_x, q_y). Let $g(x) = \alpha_1(x - p_x) + p_y$ be the line passing through (p_x, p_y) with slope α_1. Define the vertical distance between $f(x)$ and $g(x)$ in $[p_x, q_x]$ to be $d(x) = f(x) - g(x)$. Since we guess $\alpha_1 = \frac{\mathrm{dy}}{\mathrm{dx}}(p_x, p_y)$ with step size $2c/Q(n)$,

$$\begin{aligned} d(x) &= f(x) - (\alpha_1(x - p_x) + p_y) \\ &\le f(x) - ((\alpha_1^* \pm 2c/Q(n))(x - p_x) + p_y) \\ &= (f(x) - \alpha_1^*(x - p_x) - p_y) \pm 2c/Q(n)(x - p_x) \\ &= \frac{f^{(2)}(\xi)}{2!}(x - p_x)^2 \pm 2c/Q(n)(x - p_x), \end{aligned}$$

for some constant ξ between p_x and x, where the last equality follows from Taylor's theorem. Since $x \in [p_x, q_x]$ and $|q_x - p_x| \le 1/Q(n)$ as they are in C and all the derivatives are bounded, $|d(x)| = O(1/Q(n)^2)$. Since each slab has vertical width $w_v = 1/Q(n)^2$, we only need $O(1)$ slabs to cover σ.

To find these slabs, by a similar analysis as in Lemma 8.G.1, since there are only $Q(n)^{O(1)}$ slabs in total, we can build a simple $Q(n)^{O(1)}$ size searching data structure to find the $O(1)$ slabs in time $\tilde{O}(1)$. □

Having Lemma 8.I.1 in hand, the query process is essentially the same as the one for the curvature-based solution and the analysis is also the same by replacing Lemma 8.G.1 by Lemma 8.I.1. We omit the deials and present the following theorem.

Theorem 8.I.1. *Let $\mathscr{R}$ be the set of semialgebraic ranges formed by degree-Δ bivariate polynomials with bounded derivatives up to the Δ-th order. For any $\log^{O(1)} n \le Q(n) \le n^{\varepsilon}$ for some constant ε, and a set $\mathscr{P}$ of n points distributed uniformly randomly in $\mathscr{U} = [0,1] \times [0,1]$, we can build a data structure of space $\tilde{O}(n^{\mathbf{m}}/Q(n)^{3\mathbf{m}-4})$ such that for any $\mathscr{R} \in \mathscr{R}$, we can report $\mathscr{R} \cap \mathscr{P}$ in time $\tilde{O}(Q(n)) + O(k)$ in expectation, where $\mathbf{m}$ is the number of parameters needed to define a degree-Δ bivariate polynomial and k is the output size.*

Remark 8.I.1. *Note that we actually only need bounded derivatives up to the second order in Theorem 8.I.1.*

8.J An $S(n) = \tilde{O}(\frac{n^{\mathbf{m}}}{Q(n)^{((2\mathbf{m}-\Delta)(\Delta+1)-2)/2}})$ Derivative-based Data Structure

Now we improve the results in Appendix 8.I. The main idea is to use slabs formed by higher degree polynomial equalities. These slabs work as finer and finer approximations to the boundaries of query ranges. We first introduce some notations.

Definition 8.J.1. *Let $I_x = [x_l, x_r]$ be an interval in the x-axis. Let $U(x)$ and $L(x)$ be two degree-i polynomials in x such that $\forall x \in I_x, U(x) > L(x)$. We say that the region enclosed by $U(x)$, $L(x)$, $x = x_l$ and $x = x_r$ is an i-slab s. We also say the x-range of s is $[x_l, x_r]$. Furthermore, if for all $x \in I_x$, $U(x) - L(x) = w$, we say s is a uniform slab with width w.*

In our application, $L(x), U(x)$ will be two degree-i polynomial functions that differ only in their constant terms. It is not hard to see that in this case, all the slabs are in fact uniform.

In a nutshell, our data structure Ψ_Δ for degree-Δ polynomial inequalities is still a two-level data structure. The top-level structure is similar to that we described in Appendix 8.I but instead of using 1-slabs, we use $(\Delta-1)$-slabs. These $(\Delta-1)$-slabs will have width $1/Q(n)^\Delta$ and we build data structures of size $\tilde{O}(n^{\mathbf{m}})$ for the points in each slab that can answer semialgebraic queries defined by degree-Δ polynomial inequalities in $\tilde{O}(1) + O(k)$ time. The second part is a data structure built for the entire input points and it can answer degree-$(\Delta-1)$ polynomial inequality queries in time $\tilde{O}(1) + O(k)$ with space usage $\tilde{O}(n^{\mathbf{m}'})$, where $\mathbf{m}' = \mathbf{m}_{2,\Delta-1}$. The overall idea of our data structure is the following: given $\mathscr{R}$, we use $(\Delta-1)$-slabs to cover its boundary. Then the remaining parts will be defined by degree-$(\Delta-1)$ polynomial inequalities. So we can use the bottom-level data structure to solve them.

Now we describe the details. We first describe how to generate i-slabs for $i = 1, 2, \cdots, \Delta-1$. The base 1-slabs are what we have described in Appendix 8.I. Now assume we already have an $(i-1)$-slab s_{i-1}, we generate i-slabs as follows. Let the x-range of s_{i-1} be $[x_l, x_r]$. Let $\alpha_j^l = \frac{\mathrm{d}^j y}{\mathrm{d}x^j}(x_l)$ for $j = 1, 2, \cdots, i-1$ be the j-th order derivatives of $L(x)$ of s_{i-1} at $x = x_l$. Now to construct $L(x)$ of an i-slab s_i, we make $Q(n)$ finer guesses for each $\frac{\mathrm{d}^j y}{\mathrm{d}x^j}(x_l)$. Specifically, $\frac{\mathrm{d}^j y}{\mathrm{d}x^j}(x_l) = \alpha_j^l, \alpha_j^l + \frac{2c}{Q(n)^{i-j+1}}, \alpha_j^l + 2 \cdot \frac{2c}{Q(n)^{i-j+1}}, \cdots, \alpha_j^l + \frac{2c}{Q(n)^{i-j}}$, for $j = 1, 2, \cdots, i-1$, and $\frac{\mathrm{d}^i y}{\mathrm{d}x^i}(x_l) = -c + \frac{2c}{Q(n)}, -c + 2 \cdot \frac{2c}{Q(n)}, \cdots, c$. We then place "anchor" points evenly spaced with distance $1/Q(n)^{i+1}$ on the left boundary of s_{i-1}. Every two degree-i polynomials passing through adjacent anchor points having the same $\frac{\mathrm{d}^j y}{\mathrm{d}x^j}(x_l)$ for $j = 1, 2, \cdots, i$ defines an i slab. If any two degree-i polynomials $P(x), Q(x)$ have the same k-th derivatives for all $k = 1, 2, \cdots, i$ at two points (x_l, y_1), (x_l, y_2), it is elementary to show that for all x, $|P(x) - Q(x)| = |y_1 - y_2|$. So every i-slab is uniform and its width is $1/Q(n)^{i+1}$.

To build Ψ_Δ, we first build 1-slabs as we did in Appendix 8.I, and then repeatedly applying the process described in the previous paragraph to get degree-$(\Delta-1)$ slabs. Then we build the $\tilde{O}(n^{\mathbf{m}})$ space data structure in each slab as the top-level data structure, and then build the $\tilde{O}(n^{\mathbf{m}'})$ space data structure for all input points as the bottom-level data structure.

Now we bound the space usage. By the above procedure, for each $(i-1)$-slab, $i \geq 3$ we generate $Q(n)^{i-2}$ guesses for derivatives for the first $i-2$ derivatives, and

$Q(n)$ guesses for the $(i-1)$-th derivative. We have $\frac{1/Q(n)^{i-1}}{1/Q(n)^i} = Q(n)$ anchor points for the lower boundaries of slabs to pass through. So in total, we generate $Q(n)^{i-2} \cdot Q(n) \cdot Q(n) = Q(n)^i$ many $(i-1)$-slabs in an $(i-2)$-slab. We know from Appendix 8.I that the number of 1-slabs is upper bounded by $O(Q(n)^4)$. Since we only build fast-query data structures in $(i-1)$-slabs, the total space usage of all the structures built on $(i-1)$-slabs is then bounded by

$$\begin{aligned} S_1(n) &= O\left(Q(n)^4 \cdot \left(\prod_{j=3}^{\Delta} Q(n)^j \cdot\right) \cdot \left(\frac{1}{Q(n)^{\Delta}} \cdot \frac{1}{Q(n)} \cdot n\right)^{\mathbf{m}}\right) \\ &= O\left(Q(n)^{(\Delta+1)\Delta/2+1} \cdot \frac{n^{\mathbf{m}}}{Q(n)^{\mathbf{m}(\Delta+1)}}\right) \\ &= O\left(\frac{n^{\mathbf{m}}}{Q(n)^{((2\mathbf{m}-\Delta)(\Delta+1)-2)/2}}\right). \end{aligned}$$

As mentioned before, the space usage of the bottom-level data structure for Ψ_Δ is $\tilde{O}(n^{\mathbf{m}'})$. Then for query time $Q(n) = n^{\varepsilon}$ where ε is some small constant, the space usage of our entire data structure Ψ_Δ is bounded by $\tilde{O}(n^{\mathbf{m}}/Q(n)^{((2\mathbf{m}-\Delta)(\Delta+1)-2)/2})$.

For query answering, we first show the following lemma, which is a generalization of Lemma 8.I.1. The proof idea is similar to Lemma 8.I.1, the only difference is now we consider a Taylor polynomial of degree-$(\Delta-1)$ instead of 1.

Lemma 8.J.1. *In our construction, if some differentiable sub-curve σ is contained in some cell C, then we can find up to $O(1)$ $(\Delta-1)$-slabs to cover σ. The time needed to find these slabs is $\tilde{O}(1)$.*

Proof. Let $(p_x, p_y), (q_x, q_y)$ be the left and right endpoints of σ and $\frac{\mathrm{d}^i y}{\mathrm{d}x^i}(p_x, p_y) = \alpha_i^*$ for $i = 1, 2, \cdots, \Delta-1$. Let $f(x)$ be the implicit function defined by σ in $[p_x, q_x]$ and let $g(x)$ be a degree-Δ polynomial whose first Δ derivatives agree with those of $f(x)$ at point (p_x, p_y). By Taylor's theorem, the vertical distance between $f(x)$ and $g(x)$ is easily calculated to be bounded by $O(1/Q(n)^{\Delta+1})$ in $[p_x, q_x]$. Next we bound the vertical distance between $g(x)$ and the best fitting polynomial in our construction. Let (a, b) be the intersection of $g(x)$ with the line containing the left boundary of C. Let $h(x) = \sum_{i=1}^{\Delta-1} \frac{\alpha_i}{i!}(x-a)^i + b$ be a degree-$(\Delta-1)$ polynomial passing through (a, b) with the i-th order derivative being α_i at $x = a$. We define the vertical distance between $g(x)$ and $h(x)$ in this range to be $d(x) = g(x) - h(x)$.

Since we guess $\alpha_i = \frac{\mathrm{d}^i y}{\mathrm{d}x^i}$ at $x = a$ with step size $2c/Q(n)^{\Delta-i}$ in our construction,

$$\begin{aligned}
d(x) &= g(x) - \left(\sum_{i=1}^{\Delta-1} \frac{\alpha_i}{i!}(x-a)^i + b\right) \\
&\le g(x) - \left(\sum_{i=1}^{\Delta-1} \left(\frac{\alpha_i^* \pm 2c/Q(n)^{\Delta-i}}{i!}\right)(x-a)^i + b\right) \\
&= \left(g(x) - \left(\sum_{i=1}^{\Delta-1} \frac{\alpha_i^*}{i!}(x-a)^i + b\right)\right) \pm \sum_{i=1}^{\Delta-1} \frac{2c/Q(n)^{\Delta-i}}{i!}(x-a)^i \\
&= \frac{g^{(\Delta)}(\xi)}{\Delta!}(x-a)^\Delta \pm \sum_{i=1}^{\Delta-1} \frac{2c/Q(n)^{\Delta-i}}{i!}(x-a)^i
\end{aligned}$$

for some constant ξ between a and x, where the last equality follows from Taylor's theorem. Since $x \in [a, q_x]$ and $|q_x - a| \le 1/Q(n)$ and all the derivatives of $g(x)$ are bounded in $\mathscr{U}$, $|d(x)| = O(1/Q(n)^\Delta)$. Then the distance between $f(x)$ and $h(x)$ is bounded by $O(1/Q(n)^{\Delta+1}) + |d(x)| = O(1/Q(n)^\Delta)$ in $[p_x, q_x]$. Since each $(\Delta-1)$-slab has width $1/Q(n)^\Delta$, so it takes $O(1)$ $(\Delta-1)$-slabs to cover σ. To find these slabs, by a similar analysis as in Lemma 8.G.1, since there are only $Q(n)^{O(1)}$ slabs in total, we can build a simple $Q(n)^{O(1)}$ size searching data structure to find the $O(1)$ slabs in time $\tilde{O}(1)$. $\square$

With Lemma 8.J.1 in hand, the query algorithm is essentially the same as the data structure described in Appendix 8.G except for one minor difference: here when we answer query in some cell, we find $(\Delta-1)$-slabs and use the fast query data structure in it. But now since the boundaries of slabs are degree-$(\Delta-1)$ polynomials, we need to handle ranges defined by $(\Delta-1)$ polynomial inequalities instead of linear inequalities. This can be handled by our bottom-level data structure. By a similar analysis as in Appendix 8.G, we can find $O(Q(n))$ $(\Delta-1)$-slabs to cover $\partial\mathscr{R}$. We can then report all the points close to $\partial\mathscr{R}$ in time $\tilde{O}(Q(n)) + O(k)$. The remaining regions of $\mathscr{R}$ are defined by $O(Q(n))$ boundaries of the slabs we used and G by a similar analysis as in Appendix 8.H. We use the bottom-level data structure for this part and again we need $\tilde{O}(Q(n)) + O(k)$ time to report the points. In total, the query time is bounded by $\tilde{O}(Q(n)) + O(k)$. This proves Theorem 8.4.2.

Specifically, for polynomial inequalities of form $y + \sum_{a_i} x^i \le 0$ or $x + \sum_{a_i} y^i \le 0$, where $a_i \in \mathbb{R}$ and $0 \le i \le \Delta$ is an integer, we have:

Theorem 8.J.1. *For Semialgebraic range $\mathscr{R}$ formed by polynomial inequalities of form $y + \sum_{a_i} x^i \le 0$ or $x + \sum_{a_i} y^i \le 0$, where $a_i \in \mathbb{R}$ and $0 \le i \le \Delta$ is an integer, and any $\log^{O(1)} n \le Q(n) \le n^\varepsilon$ for some constant ε, if the n input points are distributed uniformly randomly in a unit square $\mathscr{U} = [0,1] \times [0,1]$, we can build a data structure of space $\tilde{O}(n^{\Delta+1}/Q(n)^{(\Delta+3)\Delta/2})$ that answers range reporting queries with $\mathscr{R}$ in time $\tilde{O}(Q(n)) + O(k)$ in expectation, where k is the number of points to report.*

Chapter 9

2D Generalization of Fractional Cascading on Axis-aligned Planar Subdivisions

Abstract

Recently, Ezra and Sharir [ES22b] showed an $O(n^{3/2+\sigma})$ space and $O(n^{1/2+\sigma})$ query time data structure for ray shooting among triangles in $\mathbb{R}^3$. This improves the upper bound given by the classical $S(n)Q(n)^4 = O(n^{4+\sigma})$ space-time tradeoff for the first time in almost 25 years and in fact lies on the tradeoff curve of $S(n)Q(n)^3 = O(n^{3+\sigma})$. However, it seems difficult to apply their techniques beyond this specific space and time combination. This pheonomenon appears persistently in almost all recent advances of flat object intersection searching, e.g., line-tetrahedron intersection in $\mathbb{R}^4$ [ES22a], triangle-triangle intersection in $\mathbb{R}^4$ [ES22a], or even among flat semialgebraic objects [AAE$^+$22].

We give a timely explanation to this phenomenon from a lower bound perspective. We prove that given a set $\mathscr{S}$ of $(d-1)$-dimensional simplicies in $\mathbb{R}^d$, any data structure that can report all intersections with a query line in small ($n^{o(1)}$) query time must use $\Omega(n^{2(d-1)-o(1)})$ space. This dashes the hope of any significant improvement to the tradeoff curves for small query time and almost matches the classical upper bound. We also obtain an almost matching space lower bound of $\Omega(n^{6-o(1)})$ for triangle-triangle intersection reporting in $\mathbb{R}^4$ when the query time is small. Along the way, we further develop the previous lower bound techniques by Afshani and Cheng [AC23b, AC22].

9.1 Introduction

Given a set $\mathscr{S}$ of triangles in $\mathbb{R}^3$, how to preprocess $\mathscr{S}$ such that given any query ray γ, we can efficiently determine the first triangle intersecting γ or report no such triangle exists? This problem, known as ray shooting, is one of the most important problems in computational geometry with countless papers published over the last three decades [Pel90, AM93, MS93, Pel93, dBHO$^+$94, AS96, Ram99, AdBG08, dBG08, ES22b, AAE$^+$22]. For a comprehensive overview of this problem, we refer the readers to an excellent recent survey [Pel17].

Recently, there have been considerable and significant advances on ray shooting and a number of problems related to intersection searching on the upper bound side. We complement these attempts by giving lower bounds for a number of intersection searching problems; these also settle a recent open question asked by Ezra and Sharir [ES22b].

Background and Previous Results

In geometric intersection searching, the input is a set $\mathscr{S}$ of geometric objects and the goal is to preprocess $\mathscr{S}$ into a data structure such that given a geometric object γ at the query time, one can find all the objects in $\mathscr{S}$ that intersect γ. In the reporting variant of such a query, the output should be the list of all the intersecting objects in $\mathscr{S}$. Intersection searching is a generalization of range searching, a fundamental and core area of computational geometry [Aga17]. This captures many natural classic problems e.g., simplex range reporting where the inputs are points (0-flats) and the queries are simplices (subsets of d-flats), ray shooting reporting among triangles in $\mathbb{R}^3$ where the inputs are triangles (subsets of 2-flats) and the queries are rays (subsets of 1-flats) and so on. See [Aga17, Pel17] for more information.

Without going too much in-depth, it suffices to say that by now, the simplex range searching problem is more or less well-understood. There are classical solutions that offer the space and query time trade-off of $S(n)Q^d(n) = \tilde{O}(n^d)$ where $S(n)$ and $Q(n)$ are the space and query time of the data structure [Cha18, Mat93, Cha12] and there are a number of almost matching lower bounds that show these are essentially tight [Afs13, Cha89, CR96].

However, intersection searching in higher dimensions is less well-understood. The classical technique is to lift the problem to the parametric space of the input or the query, reducing the problem to semialgebraic range searching, a generalized version of simplex range searching, where queries are semialgebraic sets of constant description complexity. In mid-1990s, semialgebraic range searching could only be solved efficiently in four and lower dimensions by classical tools developed for simplex range searching [AM94], resulting in a space-time trade-off bound of $S(n)Q(n)^4 = O(n^{4+\sigma})$ for line-triangle intersection searching in $\mathbb{R}^3$, where $\sigma > 0$ can be any small constant.

Recently, using polynomial techniques [GK15, Gut15], several major advances have been made on semialgebraic range searching. For example, near optimal small linear space and fast query data structure were developed [AMS13, MP15, AAEZ21]. These almost match the newly discovered lower bound bounds [AC23b, AC22]. However, these polynomial techniques also have led to significant advances in intersection searching. For ray-triangle intersection reporting in $\mathbb{R}^3$, Ezra and Sharir [ES22b] showed that using algebraic techniques, it is possible to build a data structure of space $S(n) = O(n^{3/2+\sigma})$ and query time $Q(n) = O(n^{1/2+\sigma})$ for ray shooting among triangles. The significance of this result is that it improves the upper bound given by the trade-off curve of $S(n)Q(n)^4 = O(n^{4+\sigma})$ for the first time in almost 25 years and in fact it lies on the trade-off curve of $S(n)Q(n)^3 = O(n^{3+\sigma})$. This leads to the following very interesting question asked by Ezra and Sharir. To quote them directly: *"There are several open questions that our work raises. First, can we improve our trade-off for all values of storage, beyond the special values of $O(n^{3/2+\varepsilon})$ storage and $O(n^{1/2+\varepsilon})$ query time? Ideally, can we obtain query time of $O(n^{1+\varepsilon}/s^{1/3})$, with s storage, as in the case of ray shooting amid planes? Alternatively, can one establish a lower-bound argument that shows the limitations of our technique?"*

Inspired by [ES22b], additional results for flat intersection searching were discovered during the last two years, e.g., triangle-triangle intersection searching in $\mathbb{R}^4$ [ES22a], line-tetrahedron intersection searching in $\mathbb{R}^4$ [ES22a], curve-disk intersection searching in $\mathbb{R}^3$ [AAE+22], and even more general semialgebraic flat intersection searching [AAE+22]. Similar to the result in [ES22b], the improved results are only observed for a special space-time combination and the improvement to the entire trade-off curve is limited. This once again raises the question of whether it is possible to obtain the trade-off curve of $S(n)Q(n)^d = O(n^{d+\sigma})$ for intersection searching in $\mathbb{R}^d$.

Our Results

We give a negative answer to this question. We show that answering intersection searching queries in polylogarithmic time when the queries are lines in $\mathbb{R}^d$ and input objects are subsets of $(d-1)$-flats (that we call hyperslabs) requires $\overset{o}{\Omega}(n^{2(d-1)})$ space[1]. Our lower bound in fact applies to "thin" $(d-1)$-dimensional slabs (e.g., in 3D, that would be the intersection of the region between two parallel hyperplanes with another hyperplane). This almost matches the current upper bound for the problem and shows that the improvement in [ES22b] cannot significantly improve the trade-off curve when the query time is small. To be specific, we obtain a lower bound of

$$S(n) = \overset{o}{\Omega}\left(\frac{n^{2(d-1)}}{Q(n)^{4(3d-1)(d-1)-1}}\right)$$

for line-hyperslab intersection reporting in $\mathbb{R}^d$ and a lower bound of

$$S(n) = \overset{o}{\Omega}\left(\frac{n^6}{Q(n)^{125}}\right)$$

for triangle-triangle intersection reporting in $\mathbb{R}^4$. Here, $S(n)$ and $Q(n)$ are the space and query time of the data structure. Similar to the other semialgebraic range reporting lower bounds [AC23b, AC22], these lower bounds have a much larger exponent on $Q(n)$ than on n which does allow for substantial improvements when $Q(n)$ is no longer too small; we have not opted for optimizing the exponent of $Q(n)$ in our bounds and using tighter arguments, these exponents can be improved but they cannot match the exponent of n.

[1]In this paper, $\overset{o}{\Omega}(\cdot), \overset{o}{\Theta}(\cdot), \overset{o}{O}(\cdot)$ hides $n^{o(1)}$ factors; $\tilde{\Omega}(\cdot), \tilde{\Theta}(\cdot), \tilde{O}(\cdot)$ hides $\log^{O(1)} n$ factors.

We believe our results are timely as flat intersection searching is a hotly investigated field recently, and as mentioned, with many open questions that need to be answered from a lower bound point of view.

Technical Contributions

From a technical point of view, our results require going beyond the previous attempts [AC23b, AC22]. To elaborate, the previous general technique assumed a particular form for the polynomials involved in defining the query semialgebraic ranges, namely, of the form $X_1 = X_2^\Delta + P(X_1, \cdots, X_d)$ where the coefficients of P had to be independent and thus could be set arbitrarily small. Unfortunately, the problems in intersection searching cannot fit this framework and there seems to be no easy fix for the following reason. The previous technique relies heavily on the fact that if the coefficients of P is small enough, then one can approximate X_1 with X_2^Δ and for the technique to work both conditions must hold (i.e., small coefficients for P and having degree Δ on X_2).

Generally speaking, the previous techniques do not say anything about problems in which the polynomials involved have a specific form; the only exception is the lower bound for annuli [AC23b] where specific approaches had to be created that could only be applied to the specific algebraic form of circles.

The issue is very prominent in intersection searching where we are dealing with polynomials where the coefficients of the monomials are no longer independent and the polynomials involved have specific forms; for instance, the coefficient of X_2^Δ is zero. We introduce techniques that allows us circumvent these limitations and obtain lower bounds for some broader class of problems that involve polynomials with some specific forms.

9.2 Preliminaries

The Geometric Range Reporting Lower Bound Framework in the Pointer Machine

We use the pointer machine lower bound framework that was also used in the latest proofs [AC22]. This is a streamlined version of the one originally proposed by Chazelle [Cha90a] and Chazelle and Rosenberg [CR96]. In the pointer machine model, the memory is represented as a directed graph where each node stores one point as well as two pointers pointing to two other nodes in the graph. Given a query, the algorithms starts from a special "root" node, and then explores a subgraph which contains all the input points to report. The size of the directed graph is then a lower bound for the space usage and then minimum subgraph needed to explore to answer any query is a lower bound for the query time.

Intuitively, to answer a range reporting query efficiently, we need to store the output points to the query close to each other. If the answer to any query contains many points and two queries share very few points in common, many points must be stored multiple times, leading to a big space usage.

The streamlined version of the framework is the following [AC22].

Theorem 9.2.1. *Suppose a d-dimensional geometric range reporting problem admits an $S(n)$ space and $Q(n)+O(k)$ query time data structure, where n is the input size and k is the output size. Let $Vol(\cdot)$ denote the d-dimensional Lebesgue measure. Assume we can find $m = n^c$, for a positive constant c, ranges $\mathscr{R}_1, \mathscr{R}_2, \cdots, \mathscr{R}_m$ in a d-dimensional hyperrectangle R such that*

1. *$\forall i = 1,2,\cdots,m, Vol(\mathscr{R}_i \cap R) \geq 4cVol(R)Q(n)/n$;*
2. *$Vol(\mathscr{R}_i \cap \mathscr{R}_j) = O(Vol(R)/(n2^{\sqrt{\log n}}))$ for all $i \neq j$.*

Then, we have $S(n) = \overset{o}{\Omega}(mQ(n))$.

Notations and Definitions for Polynomials

In this paper, we only consider polynomials on the reals. Let $P(X_1,\cdots,X_d)$ be a polynomial on d indeterminates of degree Δ. Sometimes we will use the notation X to denote the set of d interminates $X_1,\cdots,X_d$ and so we can write P as $P(X)$. We denote by $I_{d,\Delta}$ a set of d-tuples of non-negative integers $(i_1,\cdots,i_d)$ whose sum is at most Δ. We might omit the subscripts d and Δ if they are clear from the context. For an $\mathbf{i} \in I$, we use the notation $X^{\mathbf{i}}$ to represent the monomial $\Pi_{j=1}^{d} X_j^{i_j}$ where $\mathbf{i} = (i_1,\cdots,i_d)$. Thus, given real coefficients $A_{\mathbf{i}}$, for $\mathbf{i} \in I$, we can write P as $\sum_{\mathbf{i}\in I} A_{\mathbf{i}} X^{\mathbf{i}}$.

Geometric Lemmas

We introduce and generalize some geometric lemmas about the intersection of polynomials used in [AC23b]. We first generalize the core Lemma in [AC23b] for univariate polynomials, using a proof similar to [AC22]. We refer the readers to Appendix 9.A for details.

Lemma 9.2.1. *Let $P(x) = \sum_{i=0}^{\Delta} a_i x^i$ and $Q(x) = \sum_{i=0}^{\Delta} b_i x^i$ be two univariate (constant) degree-Δ polynomials in $\mathbb{R}[x]$ and $|a_i - b_i| \geq \eta$ for some $0 \leq i \leq \Delta$.*

Suppose there is an interval $\mathscr{I}$ of x such that for every $x_0 \in \mathscr{I}$ we have $|P(x_0) - Q(x_0)| \leq w$, then the length of $\mathscr{I}$ is upper bounded by $O((w/\eta)^{1/\mathscr{U}})$, where $\mathscr{U} = \binom{\Delta+1}{2}$ and the $O(\cdot)$ notation hides constant factors that depend on Δ.

Using Lemma 9.2.1, we can show the following; See Appendix 9.B for details.

Lemma 9.2.2. *Let $P_1(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} A_{\mathbf{i}} X^{\mathbf{i}}$ and $P_2(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} B_{\mathbf{i}} X^{\mathbf{i}}$ be two d-variate degree-Δ polynomials in $\mathbb{R}[X]$ and $|A_{\mathbf{i}} - B_{\mathbf{i}}| \geq \eta_d$ for some $\mathbf{i} \in I_{d,\Delta}$.*

Suppose for each assignment $X_d \in \mathscr{I}_d$ to P_1, P_2, where $\mathscr{I}_d$ is an interval for X_d, all the coefficients of the resulting $(d-1)$-variate polynomial $Q_1(X_1,\cdots X_{d-1})$ and $Q_2(X_1,\cdots X_{d-1})$ differ by at most η_{d-1}, then $|\mathscr{I}_d| = O((\eta_{d-1}/\eta_d)^{1/\mathscr{U}})$.

We can use Lemma 9.2.2 $d-2$ times, and obtain the following corollary.

Corollary 9.2.1. *Let $P_1(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} A_{\mathbf{i}}X^{\mathbf{i}}$ and $P_2(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} B_{\mathbf{i}}X^{\mathbf{i}}$ be two d-variate degree-Δ polynomials in $\mathbb{R}[X]$ and $|A_{\mathbf{i}} - B_{\mathbf{i}}| \geq \eta_d$ for some $\mathbf{i} \in I_{d,\Delta}$ for $d \geq 3$.*

Suppose for each assignment $X_i \in \mathscr{I}_i$ to P_1, P_2, where $\mathscr{I}_i$ is an interval for X_i, for $i = 3,4,\cdots,d$, all the coefficients of the resulting bivariate polynomial $Q_1(X_1,X_2)$ and $Q_2(X_1,X_2)$ differ by at most η_2, then $|\mathscr{I}_i| = O((\eta_{i-1}/\eta_i)^{1/\mathscr{U}})$ for all $i = 3,4,\cdots,d$.

To get the final corollary, we would like the set each η_i such that the length of all each interval $\mathscr{I}_i$ is bounded by some parameter ϑ for $i = 3,\cdots,d$. We thus set $\eta_{d-i} = \eta_{d-i+1}\vartheta^{\mathscr{U}}$.

Corollary 9.2.2. *Let $P_1(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} A_{\mathbf{i}}X^{\mathbf{i}}$ and $P_2(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} B_{\mathbf{i}}X^{\mathbf{i}}$ be two d-variate degree-Δ polynomials in $\mathbb{R}[X]$ and $|A_{\mathbf{i}} - B_{\mathbf{i}}| \geq \eta_d$ for some $\mathbf{i} \in I_{d,\Delta}$ for $d \geq 3$.*

Suppose for each assignment $X_i \in \mathscr{I}_i$ to P_1, P_2, where $\mathscr{I}_i$ is an interval for X_i, for $i = 3,4,\cdots,d$, all the coefficients of the resulting bivariate polynomial $Q_1(X_1,X_2)$ and $Q_2(X_1,X_2)$ differ by at most $\eta_d\vartheta^{\mathscr{U}(d-2)}$, then $|\mathscr{I}_i| = O(\vartheta)$ for all $i = 3,4,\cdots,d$.

Algebra Preliminaries

In this section, we review some tools from algebra. The first tool we will use is the linearity of determinants from linear algebra.

Theorem 9.2.2 (Linearity of Determinants). *Let $A = \begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_n\end{bmatrix}$ be an $n \times n$ matrix where each $\mathbf{a}_i \in \mathbb{R}^n$ is a vector. Suppose $\mathbf{a}_j = r \cdot \mathbf{w} + \mathbf{v}$ for some $r \in \mathbb{R}$ and $\mathbf{w}, \mathbf{v} \in \mathbb{R}^n$, then the determinant of A, denoted by $\det(A)$, is*

$$\begin{aligned}\det(A) &= \det\left(\begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_{j-1} & \mathbf{a}_j & \mathbf{a}_{j+1} & \cdots & \mathbf{a}_n\end{bmatrix}\right)\\ &= r\cdot\det\left(\begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_{j-1} & \mathbf{w} & \mathbf{a}_{j+1} & \cdots & \mathbf{a}_n\end{bmatrix}\right) + \det\left(\begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_{j-1} & \mathbf{v} & \mathbf{a}_{j+1} & \cdots & \mathbf{a}_n\end{bmatrix}\right).\end{aligned}$$

We will use two types of special matrices in the paper. The first is Vandermonde matrices.

Definition 9.2.1 (Vandermonde Matrices). *An $n \times n$ Vandermonde matrix is defined by n values $x_1,\cdots,x_n$ such that each entry $e_{ij} = x_i^{j-1}$ for $1 \leq i,j \leq n$.*

We can compute the determinant of Vandermonde matrices easily.

Theorem 9.2.3 (Determinant of Vandermonde Matrices). *Let V be a Vandermonde matrix defined by parameters $x_1,\cdots,x_n$. Then $\det(V) = \prod_{1\leq i<j\leq n}(x_j - x_i)$.*

We also need Sylvester matrices.

Definition 9.2.2 (Sylvester Matrices). *Let $P = \sum_{i=0}^{\Delta_1} a_i x^i$ and $Q = \sum_{i=0}^{\Delta_2} b_i x^i$ be two univariate polynomials over $\mathbb{R}[x]$ of degrees Δ_1, Δ_2 respectively . Then the Sylvester* *matrix of P and Q, denoted by Syl(P,Q), is a* $(\Delta_1+\Delta_2)\times(\Delta_1+\Delta_2)$ *matrix of the following form*

$$\begin{bmatrix} a_{\Delta_1} & a_{\Delta_1-1} & \cdots & a_0 & 0 & \cdots & 0 & 0 \\ 0 & a_{\Delta_1} & a_{\Delta_1-1} & \cdots & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{\Delta_1} & a_{\Delta_1-1} & \cdots & a_1 & a_0 \\ b_{\Delta_2} & b_{\Delta_2-1} & \cdots & b_0 & 0 & \cdots & 0 & 0 \\ 0 & b_{\Delta_2} & b_{\Delta_2-1} & \cdots & b_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & b_{\Delta_2} & b_{\Delta_2-1} & \cdots & b_1 & b_0 \end{bmatrix}.$$

The Sylvester matrix has Δ_2 rows with entries from P and Δ_1 rows with entries from Q. For example, the Sylvester matrx of two polynomials $P = p_1x + p_2$ and $Q = q_1x + q_2$ is

$$\text{Syl}(P,Q) = \begin{bmatrix} p_1 & p_2 \\ q_1 & q_2. \end{bmatrix}$$

One application of Sylvester matrices is to compute the resultant, which is one of the important tools in algebraic geometry. One significance of the resultant is that it equals zero if and only if P and Q have a common factor.

Definition 9.2.3. *Let P,Q be two univariate polynomials over $\mathbb{R}$. The resultant of P and Q, denoted by Res(P,Q), is defined to be the determinant of the Sylvester matrix of P and Q, i.e., Res$(P,Q) = \det($Syl$(P,Q))$.*

9.3 An Algebraic Geometry Lemma

In this section, we prove an important algebraic geometry lemma that will later be used in our lower bound proof.

Lemma 9.3.1. *Let F and G be two univariate polynomials on x of degree Δ_F and Δ_G respectively and the leading coefficient of G is 1. Let $P(x,y) \equiv yG(x) - F(x)$.*

Let L be a set of $\ell = \Delta_1 + \Delta_G + 1$ points (x_k, y_k) where $\Delta_1 \geq \Delta_F - 1$ and each $x_k = \Theta(1)$ such that $|P(x_k,y_k)| \leq \varepsilon < 1$ for a parameter ε, and $G(x_k) = \Theta(1)$.

Let V be a vector of ℓ monomials consisting of monomials x^i for $0 \leq i \leq \Delta_1$ and monomials yx^i for $0 \leq i \leq \Delta_G - 1$.

If A is an $\ell \times \ell$ matrix where the k-th row of A is the evaluation of the vector V on point (x_k,y_k), then $|\det(A)| \geq \Omega(\text{Res}(G,F)\lambda^{\ell^2}) - O(\varepsilon)$ where $\lambda = \min_{1\leq k_1<k_2\leq \ell} |x_{k_1} - x_{k_2}|$.

Proof. Note that if $\text{Res}(G,F) = 0$, then there is nothing to prove and thus we can assume this is not the case. Now observe that since $G(x_k) = \Theta(1)$, we can write $y_k = \frac{F(x_k)}{G(x_k)} + \gamma_k$ where $|\gamma_k| = O(\varepsilon)$.

Now consider the matrix A and plug in this value of y_k. An entry of A is in the form of a monomial yx^i being evaluated on a point (x_k, y_k) and thus we have:

$$y_k x_k^i = \left(\frac{F(x_k)}{G(x_k)} + \gamma_k\right) x_k^i = \frac{F(x_k)}{G(x_k)} x_k^i + \gamma_{i,k} \tag{9.1}$$

where $|\gamma_{i,k}| = O(\varepsilon)$. We use the linearity of determinants (see Theorem 9.2.2) in a similar fashion that was also used in [AC22]. In particular, consider a column of the matrix A; it consists of the evaluations of a monomial yx^i on all the points $(x_1, y_1), \cdots, (x_\ell, y_\ell)$. Using Eq. (9.1), we can write this column as the addition of a column C_i that consists of the evaluation of the rational function $\frac{F(x)}{G(x)}x^i$ on the points $x_1, \cdots, x_\ell$ and a column Γ_i that consists of all the values $\gamma_{i,k}$ for $1 \le k \le \ell$. By the linearity of determinants, we can write the determinant of A as the sum of determinants of two matrices where one matrix includes the column C_i and the other has Γ_i; observe that the magnitude of the determinant of the latter matrix can be upper bounded by $O(\varepsilon)$, with hidden constants that depend on Δ. By performing this operation on all the columns, we can separate all the entries involving $\gamma_{i,k}$ into separate matrices and the magnitude of sum of the determinants can be bounded by $O(\varepsilon)$.

Let B be the matrix that remains after removing all the $\gamma_{i,k}$ terms. We bound $|\det(B)|$. Note that B consists of row vectors

$$U = (1 \quad x \quad \cdots \quad x^{\Delta_1} \quad y \quad yx \quad \cdots \quad yx^{\Delta_G - 1}).$$

evaluated at some value $x = x_k$ and $y = \frac{F(x_k)}{G(x_k)}$ at its k-th row. This is equivalent to the evaluation of the following vector:

$$(1 \quad x \quad \cdots \quad x^{\Delta_1} \quad \tfrac{F}{G} \quad \tfrac{F}{G}x \quad \cdots \quad \tfrac{F}{G}x^{\Delta_G - 1}).$$

Observe that row k of matrix B will be evaluating U on the point x_k. Since $G(x_k) = \Theta(1) \neq 0$, we can multiply row k by $G(x_k)$ and this will only change the determinant by a constant factor. With a slight abuse of the notation, let B denote the matrix after this multiplication step. Thus, the columns of B now correspond to the evaluation of the following vector.

$$(G \quad Gx \quad \cdots \quad Gx^{\Delta_1} \quad F \quad Fx \quad \cdots \quad Fx^{\Delta_G - 1}).$$

Note that we can exchange columns and it will only flip the signs of the determinant of a matrix. We will focus on bounding the determinant of

$$(Gx^{\Delta_1} \quad Gx^{\Delta_1 - 2} \quad \cdots \quad G \quad Fx^{\Delta_G - 1} \quad Fx^{\Delta_G - 2} \quad \cdots \quad F).$$

The key observation is that there is a strong connection between the Sylvester matrix of G, F and matrix B. Recall that the Sylvester matrix of G and F is of the form

$$\mathrm{Syl}(G,F) = \begin{bmatrix} G_{\Delta_G} & G_{\Delta_G-1} & \cdots & G_0 & 0 & \cdots & 0 & 0 \\ 0 & G_{\Delta_G} & G_{\Delta_G-1} & \cdots & G_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & G_{\Delta_G} & G_{\Delta_G-1} & \cdots & G_1 & G_0 \\ F_{\Delta_F} & F_{\Delta_F-1} & \cdots & F_0 & 0 & \cdots & 0 & 0 \\ 0 & F_{\Delta_F} & F_{\Delta_F-1} & \cdots & F_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & F_{\Delta_F} & F_{\Delta_F-1} & \cdots & F_1 & F_0 \end{bmatrix},$$

where G_i (resp. F_i) is the coefficient of x^i in G (resp. F). Observe that

$$\begin{aligned}(Gx^{\Delta_F-1} \quad Gx^{\Delta_F-2} \quad \cdots \quad G \quad Fx^{\Delta_G-1} \quad Fx^{\Delta_G-2} \quad \cdots \quad F) = \\ \mathrm{Syl}(G,F)\cdot(x^{\Delta_F+\Delta_G-1} \quad x^{\Delta_F+\Delta_G-2} \quad \cdots \quad x \quad 1)^T,\end{aligned}$$

which means that by the linear transformation described by $\mathrm{Syl}(G,F)^{-1}$, which exists as $\mathrm{Res}(G,F) = \det(\mathrm{Syl}(G,F)) \neq 0$, we can turn the last $\Delta_F + \Delta_G$ columns in B to

$$(x^{\Delta_F+\Delta_G-1} \quad x^{\Delta_F+\Delta_G-2} \quad \cdots \quad x \quad 1).$$

Since the remaining columns are all polynomials in x and the highest degree in column i is $\Delta_G + \Delta_1 - i$ for $i = 0,1,\cdots,\Delta_F$, by using column operations, we can eliminate all lower degree terms for each column and the only term left for column i is $G_{\Delta_G}x^{\Delta_G+\Delta_1-i}$. Note that column operations do not change the determinant.

By assumption, the leading coefficients of G is 1, i.e., $G_{\Delta_G} = 1$. Thus, this transforms B into a Vandermonde matrix V_B of size $\ell \times \ell$. By Theorem 9.2.3, $|\det(V_B)| = \Omega(\lambda^{\ell^2})$. Since multiplying the inverse of $\mathrm{Syl}(G,F)$ scales $\det(B)$ by a factor of $\Theta(|\det(\mathrm{Syl}(G,F)^{-1})|) = \Theta(|\mathrm{Res}(G,F)^{-1}|)$, we bound $|\det(B)| = |\det(V_B)|/(1/|\mathrm{Res}(G,F)|) = \Omega(|\mathrm{Res}(G,F)|\lambda^{\ell^2})$. The claim then follows from this. □

9.4 Lower Bounds for Flat Intersection Reporting

We are now ready to show lower bounds for flat intersection reporting. We first establish a reduction from special polynomial slab reporting problems to flat intersection reporting.

A Reduction from Polynomial Slab Range Reporting to Flat-hyperslab Intersection Reporting

We study the following flat intersection reporting problem.

Definition 9.4.1 (Flat-hyperslab Intersection Reporting). *In the t-flat-hyperslab intersection reporting problem, we are given a set $\mathscr{S}$ of n $(d-t)$-dimensional hyperslabs* *in $\mathbb{R}^d$, i.e., regions created by a linear translation of $(d-t-1)$-flats, where $0 \le t < d$, as the input, and the goal is to preprocess $\mathscr{S}$ into a data structure such that given any query t-flat γ, we can output $\mathscr{S} \cap \gamma$, i.e., the set of $(d-t)$-hyperslabs intersecting the query t-flat, efficiently.*

First, observe that any t-flat that is not parallel to any of the axes can be formulated as

$$\begin{bmatrix} a_{0,1} & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ a_{1,1} & a_{1,2} & \cdots & a_{1,t} & a_{1,t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{d-t,1} & a_{d-t,2} & \cdots & a_{d-1,t} & a_{d-t,t+1} \end{bmatrix} \cdot \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_t \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix},$$

where $a_{i,j}$'s are the parameters defining the t-flat, and $\tau_1, \cdots, \tau_t$ are the free variables that generate points in the t-flat. Note that we only need $(d-t)(t+1)$ independent $a_{i,j}$'s to define a t-flat.

On the other hand, we consider $(d-t)$-hyperslabs of form

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & b_{1,1} & b_{1,2} & \cdots & b_{1,d-t} \\ 0 & 1 & \cdots & 0 & b_{2,1} & b_{2,2} & \cdots & b_{2,d-t} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{t,1} & b_{t,2} & \cdots & b_{t,d-t} \\ 0 & 0 & \cdots & 0 & b_{t+1,1} & b_{t+1,2} & \cdots & b_{t+1,d-t} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{d-1} \\ x_d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1+w \end{bmatrix},$$

where $b_{i,j}$'s are the parameters defining a $(d-t-1)$-flat, and parameter $w \in [0, w_0]$ adds one extra dimension to the flat to make it $(d-t)$-dimensional; in essence, we will be considering all the $(d-t-1)$-flats for all $w \in [0, w_0]$ which will turn it into a $(d-t)$-hyperslab.

Therefore, the intersection of a t-flat and a $(d-t)$-hyperslab must be a solution to

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & b_{1,1} & b_{1,2} & \cdots & b_{1,d-t} \\ 0 & 1 & \cdots & 0 & b_{2,1} & b_{2,2} & \cdots & b_{2,d-t} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{t,1} & b_{t,2} & \cdots & b_{t,d-t} \\ 0 & 0 & \cdots & 0 & b_{t+1,1} & b_{t+1,2} & \cdots & b_{t+1,d-t} \end{bmatrix} \begin{bmatrix} a_{0,1} & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ a_{1,1} & a_{1,2} & \cdots & a_{1,t} & a_{1,t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{d-t,1} & a_{d-t,2} & \cdots & a_{d-1,t} & a_{d-t,t+1} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \vdots \\ \tau_t \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1+w \end{bmatrix}.$$

Multiplying the two matrices, we obtain the following system

$$\begin{bmatrix} a_{0,1} + \sum_{i=1}^{d-t} a_{i,1} b_{1,i} & \sum_{i=1}^{d-t} a_{i,2} b_{1,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1} b_{1,i} \\ \sum_{i=1}^{d-t} a_{i,1} b_{2,i} & 1 + \sum_{i=1}^{d-t} a_{i,2} b_{2,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1} b_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d-t} a_{i,1} b_{t+1,i} & \sum_{i=1}^{d-t} a_{i,2} b_{t+1,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1} b_{t+1,i} \end{bmatrix} \cdot \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_t \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1+w \end{bmatrix}.$$

We denote this linear system by $A\mathbf{m}\tau = \mathbf{s}$ and assume

$$\det(A) \neq 0 \tag{9.2}$$

which is the case when the t-flat and the $(d-t)$-hyperslab properly intersect, and this system has a solution iff the last entry of the solution vector is 1. So by Cramer's rule, we have

$$1 = \frac{\begin{vmatrix} a_{0,1}+\sum_{i=1}^{d-t} a_{i,1}b_{1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{1,i} & \cdots & 0 \\ \sum_{i=1}^{d-t} a_{i,1}b_{2,i} & 1+\sum_{i=1}^{d-t} a_{i,2}b_{2,i} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d-t} a_{i,1}b_{t+1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{t+1,i} & \cdots & -1+w \end{vmatrix}}{\begin{vmatrix} a_{0,1}+\sum_{i=1}^{d-t} a_{i,1}b_{1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{1,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1}b_{1,i} \\ \sum_{i=1}^{d-t} a_{i,1}b_{2,i} & 1+\sum_{i=1}^{d-t} a_{i,2}b_{2,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1}b_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d-t} a_{i,1}b_{t+1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{t+1,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1}b_{t+1,i} \end{vmatrix}}.$$

By the linearity of determinants, we have

$$0 = \begin{vmatrix} a_{0,1}+\sum_{i=1}^{d-t} a_{i,1}b_{1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{1,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1}b_{1,i} \\ \sum_{i=1}^{d-t} a_{i,1}b_{2,i} & 1+\sum_{i=1}^{d-t} a_{i,2}b_{2,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t+1}b_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d-t} a_{i,1}b_{t+1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{t+1,i} & \cdots & 1+\sum_{i=1}^{d-t} a_{i,t+1}b_{t+1,i}-w \end{vmatrix}. \tag{9.3}$$

Consider the value of the above determinant using Leibniz formula for determinants, which is the sum of $(t+1)!$ terms. Consider the terms that have at most 1 factor of $b_{i,j}$; these can only come from the diagonals. Thus, any t-flat parameterized by $\mathbf{a} = (a_{i,j})$ intersects a query $(d-t)$-hyperslab parameterized by $\mathbf{b} = (b_{i,j})$ if and only if

$$0 = a_{0,1} + a_{0,1}\sum_{j=2}^{t+1}\sum_{i=1}^{d-1} a_{i,j}b_{j,i} + \sum_{i=1}^{d-1} a_{i,1}b_{1,i} + E(\mathbf{a},\mathbf{b}) + f(\mathbf{a},\mathbf{b},w) \quad = P(\mathbf{a},\mathbf{b}) + f(\mathbf{a},\mathbf{b},w),$$

where $E(\mathbf{a},\mathbf{b})$ contains the sum of products of at least two distinct $a_{i_1,i_2}b_{i_3,i_1}$ and $f(\mathbf{a},\mathbf{b},w)$ is a polynomial with factor w.

Note that after fixing $\mathbf{a},\mathbf{b}$, $f(\mathbf{a},\mathbf{b},w)$ is a polynomial in w and we assume that

$$\frac{\partial f(\mathbf{a},\mathbf{b},w)}{\partial w} = -\begin{vmatrix} a_{0,1}+\sum_{i=1}^{d-t} a_{i,1}b_{1,i} & \sum_{i=1}^{d-t} a_{i,2}b_{1,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t}b_{1,i} \\ \sum_{i=1}^{d-t} a_{i,1}b_{2,i} & 1+\sum_{i=1}^{d-t} a_{i,2}b_{2,i} & \cdots & \sum_{i=1}^{d-t} a_{i,t}b_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d-t} a_{i,1}b_{t,i} & \sum_{i=1}^{d-t} a_{i,2}b_{t,i} & \cdots & 1+\sum_{i=1}^{d-t} a_{i,t}b_{t,i} \end{vmatrix} < 0. \tag{9.4}$$

This implies the following lemma.

Lemma 9.4.1. *Assuming* $\mathbf{a},\mathbf{b}$ *satisfying Assumptions (9.2) and (9.4), for any fixed* $\mathbf{a}$*, there is a* $\mathbf{b}$ *such that* $0 \le P(\mathbf{a},\mathbf{b}) \le -f(\mathbf{a},\mathbf{b},w_0)$ *if and only if there is some* $w \in [0,w_0]$ *such that* $P(\mathbf{a},\mathbf{b}) + f(\mathbf{a},\mathbf{b},w) = 0$.

Proof. Since $f(\mathbf{a},\mathbf{b},w)$ is a polynomial in w and $\frac{\partial f}{\partial w} < 0$, $f(\mathbf{a},\mathbf{b},w)$ is continuous and decreasing in $[0,w_0]$. Furthermore $f(\mathbf{a},\mathbf{b},0) = 0$ as w is a factor of f. The lemma follows. □

Fixing $\mathbf{a}$ in $P(\mathbf{a},\mathbf{b})$, we obtain a polynomial in $\mathbf{b}$. Let $(P(\mathbf{a},\mathbf{b}), f(\mathbf{a},\mathbf{b},w_0)) = \{\mathbf{b} : 0 \le P(\mathbf{a},\mathbf{b}) \le -f(\mathbf{a},\mathbf{b},w_0)\}$ be a polynomial slab. This essentially establishes a reduction between polynomial slab reporting and flat intersection reporting.

Corollary 9.4.1. *Assuming* $\mathbf{a},\mathbf{b}$ *satisfying Assumptions (9.2) and (9.4), for any fixed* $\mathbf{a}$*, there is a* $\mathbf{b}$ *such that* $\mathbf{b} \in (P(\mathbf{a},\mathbf{b}), f(\mathbf{a},\mathbf{b},w_0))$ *if and only if a t-flat parameterized by* $\mathbf{a}$ *intersects a* $(d-t)$*-hyperslab of width* w_0 *parameterized by* $\mathbf{b}$.

Lower Bounds for Flat-hyperslab Intersection Reporting

We are now ready to prove the lower bounds. We show lower bounds for 1-flat-hyperslab intersection reporting in $\mathbb{R}^d$ and 2-flat-hyperslab intersection reporting in $\mathbb{R}^4$.

First observe that by setting $t = 1$ in Eq. (9.3) and using Corollary 9.4.1 a polynomial slab reporting problem with polynomial

$$\begin{aligned} P_1(\mathbf{a},\mathbf{b}) &= a_{0,1} + a_{0,1}\sum_{i=1}^{d-1} a_{i,2}b_{2,i} + \sum_{i=1}^{d-1} a_{i,1}b_{1,i} + \sum_{i,j=1\wedge i\neq j}^{d-1} (a_{i,1}a_{j,2} - a_{j,1}a_{i,2})b_{1,i}b_{2,j} \\ &= b_{1,1}G_1(b_{2,2}) + F_1(b_{2,2}), \end{aligned} \tag{9.5}$$

reduces to a line-hyperslab intersection reporting problem, where to get G_1, we have collected all the monomials that have $b_{1,1}$ in them and then we have factored $b_{1,1}$ out and we are considering it as a polynomial of $b_{2,2}$ (all the other variables are considered "constant"). F_1 is defined similarly by considering the remaining terms as a function of $b_{2,2}$. Observe that the polynomial does not have any term with degree 3. Let $G_1 = g_{1,1}b_{2,2} + g_{1,0}$ and $F_1 = f_{1,1}b_{2,2} + f_{1,0}$.

Similarly, polynomial slab reporting with

$$\begin{aligned} P_2(\mathbf{a},\mathbf{b}) &= a_{0,1} + a_{0,1}\sum_{j=1}^{2}\sum_{i=2}^{3} a_{j,i}b_{i,j} + \sum_{j=1}^{2} a_{j,1}b_{1,j} \\ &\quad + a_{0,1}\sum_{j,l=1\wedge j\neq l}^{2} (a_{j,2}a_{l,3} - a_{j,3}a_{l,2})b_{2,j}b_{3,l} + \sum_{j,l=1\wedge j\neq l}^{2}\sum_{k=2}^{3} (a_{j,1}a_{l,k} - a_{j,k}a_{l,1})b_{1,j}b_{k,l} \\ &= b_{1,1}G_2(b_{2,2}) + F_2(b_{2,2}) \end{aligned} \tag{9.6}$$

reduces to 2-flat-hyperslab intersection reporting in $\mathbb{R}^4$ where G_2, F_2 are defined similarly as G_1, F_1.

For the moment, we focus on the case of line-hyperslab intersection reporting but the same applies also to 2-flat-hyperslab intersection reporting in $\mathbb{R}^4$ since the polynomials F_2 and G_2 involved in the definition of Eq. (9.6) are quite similar to Eq. (9.5).

Here, we will use our techniques from Section 9.3. The general idea is that we will use Corollary 9.2.2, to reduce the $2(d-1)$-variate polynomials P_1 and P_2 into bivariate polynomials on $b_{1,1}$ and $b_{2,2}$. Then, the variable $b_{1,1}$ will be our y variable and $b_{2,2}$ will be the x variable in Section 9.3, and G_1 and F_1 here will play the same role as in that section. We will set

$$a_{1,1} = \frac{1 + a_{1,2}a_{2,1}}{a_{2,2}} \tag{9.7}$$

which will ensure that the leading coefficient of G_1 is 1. This is our normalization step, since we can divide the equations defining the intersection (and thus polynomials P_1 and P_2) by any constant. Eventually, the resultant of the polynomials F_1 and G_1 will play an important role. Observe that the resultant is

$$\operatorname{Res}(G_1, F_1) = \begin{vmatrix} 1 & g_0 \\ f_1 & f_0 \end{vmatrix} = f_0 - g_0 f_1. \tag{9.8}$$

Construction of Input Points and Queries

Now we are ready to describe our input and query construction. Assume we have a data structure that uses $S(n)$ space and has the query time $Q(n) + O(k)$ where k is the output size; for brevity we use $Q = Q(n)$.

We will start with a fixed line and a fixed hyperslab and then build the queries and inputs very close to these two fixed objects. However, we require a certain "general position" property with respect to these two fixed objects.

Recall that Eq. (9.5) refers to the condition of whether a (query) line described by $\mathbf{a}$ variables intersects a $(d-2)$-dimensional flat described by the $\mathbf{b}$ variables (which corresponds to setting the variable w to zero). Consider a fixed flat and a fixed line. To avoid future confusion, let $\mathbf{A}$ and $\mathbf{B}$ refer to this fixed line and flat. We require the following.

- $\mathbf{A}$ and $\mathbf{B}$ must intersect properly (i.e., the line is not contained in the flat). Observe that it implies that when we consider $P_1(\mathbf{A}, \mathbf{b})$ as a polynomial in $\mathbf{b}$ variables, $\mathbf{B}$ does not belong to the zero set of $P_1(\mathbf{A}, \mathbf{b})$. Note that this satisfies Assumption (9.2).
- The polynomial $P_1(\mathbf{A}, \mathbf{b})$ (as a polynomial in $\mathbf{b}$) is irreducible. This is true as long as $\mathbf{A}$ is chosen so that no coefficient in P_1 is zero. To see this, note that P_1 is a polynomial in $\mathbf{b}$ and any variable $b_{i,j}$ has degree 1. Suppose for the sake of contradiction that P_1 is reducible, then the factorization must be of the form

$$P_1(\mathbf{A}, \mathbf{b}) = \left(c_{10} + \sum_{i=1}^{d-1} c_{1i} b_{1i}\right) \cdot \left(c_{20} + \sum_{i=1}^{d-1} c_{2i} b_{2i}\right),$$

for nonzero coefficients $c_{10}, c_{20}, c_{1i}, c_{2i}$. Then by Eq. (9.5),

1. $a_{0,1} = c_{10}c_{20}$,
2. $\forall i = 1, \cdots, d-1 : a_{0,1}a_{i,2} = c_{10}c_{2i}$,
3. $\forall i = 1, \cdots, d-1 : a_{i,1} = c_{1i}c_{20}$,
4. $\forall i, j = 1, 2, \cdots, d-1 : a_{i,1}a_{j,2} - a_{j,1}a_{i,2} = c_{1i}c_{2i}$.

However, for these conditions to hold, all coefficients of P_l must be zero, a contradiction.

- Observe that the irreducibility of $P_l(\mathbf{A}, \mathbf{b})$ as a polynomial in $\mathbf{b}$ implies that it has only finitely many points where the tangent hyperplane at those points is parallel to some axis. We assume $\mathbf{B}$ is not one of those points.
- The irreducibility of $P_l(\mathbf{A}, \mathbf{b})$ as a polynomial in $\mathbf{b}$ can be used to satisfy Assumption (9.4) since the corresponding polynomial of the determinant involved in Assumption (9.4) can only have $\Theta(1)$ many common roots with $P_l(\mathbf{A}, \mathbf{b})$.
- Finally, since the polynomial $P_l(\mathbf{A}, \mathbf{b})$ is irreducible and since $\mathrm{Res}(G_1, F_1)$ is also of degree 2 in $\mathbf{b}$ variables, it follows that $\mathrm{Res}(G_1, F_1)$ is algebraically independent of $P_l(\mathbf{A}, \mathbf{b})$. This means that there are only finitely many places where both polynomials are zero, meaning, we can additionally assume that Eq. (9.8) is non-zero (when evaluated at $\mathbf{B}$).

Consider two parameters ε_p and $\varepsilon_q = \varepsilon_p / C$ where C is a large enough constant and ε_p is a parameter to be set later. Consider the parametric space of the input objects, where the variable $\mathbf{b}$ defines a single point. In such a space, $\mathbf{B}$ defines a single point. Place an axis-aligned cube R of side-length ε_p centered around $\mathbf{B}$. The input slabs are defined by placing a set of n random points inside R. Each point in R defines a $(d-2)$-dimensional flat. We set $w = \Theta(\frac{Q}{n})$ which in turn defines a "narrow $(d-1)$-hyperslab".

We now define the set of queries. Notice that P_l has exactly $2(d-1)$ algebraically independent coefficients; these are the coefficients of linear terms involved plus $a_{0,1}$; recall that by Eq. (9.7), $a_{1,1}$ was fixed as a function of $a_{1,2}a_{2,1}$ and $a_{2,2}$ but we still have $a_{0,1}$ as a free parameter. These $2(d-1)$ coefficients define another parametric space, where $\mathbf{A}$ denotes a single point. Place a $2(d-1)$-dimensional hypercube of side length ε_q and then subdivide it into a grid where the side-length of every cell is τ. Every grid point now defines a different query. Let $\mathcal{Q}$ be the set of all the queries we have constructed.

Notice that a query defined by a point $\mathbf{a} \in \mathcal{Q}$ defines a line in the primal space, but when considered in the parametric space R, it corresponds to a manifold (zeroes of a degree two multilinear polynomial) that includes the set of points that correspond to $(d-2)$-dimensional flats that pass through the line in the primal space. The variable w allows us to turn it to a range reporting problem where we need to output any $(d-2)$-dimensional flat that passes within w vertical distance of the query line. The following observations and lemmas are the important geometric properties that we require out of our construction.

Observation 9.4.1. *For two different queries* $\mathbf{a}_1$ *and* $\mathbf{a}_2$*, the polynomials* $P_1(\mathbf{a}_1, \mathbf{b})$ *and* $P_1(\mathbf{a}_2, \mathbf{b})$ *differ by at least* τ *in at least one of their coefficients.*

Observation 9.4.2. *Consider a line* f *parallel to an axis. For small enough* ε_p*, and any* $\mathbf{a} \in \mathscr{Q}$*, the function* $P_1(\mathbf{a}, \mathbf{b})$ *evaluated on the line* f *is such that the magnitude of its derivative is bounded by* $\Omega(1)$*.*

Proof. Recall that $\mathbf{B}$ was chosen such that the manifold corresponding to $\mathbf{A}$ does not have a tangent parallel to any of the axes at point $\mathbf{B}$ and thus the derivate of the function $P_1(\mathbf{A}, \mathbf{B})$ is non-zero at $\mathbf{B}$. The lemma then follows since ε_p and ε_q are small enough and $P_1(\mathbf{A}, \mathbf{B})$ is a continuous function w.r.t any of its variables. $\square$

Let $\mathrm{Vol}'(R)$ be the $(d-1)$-dimensional volume of R, i.e., the volume of the projection of R to any of its $(d-1)$-dimensional subspace.

Observation 9.4.3. *The intersection volume of the range defined by a query* $\mathbf{a}$ *and* R *is* $\Theta(w\mathit{Vol}'(R))$ *if* C *in the definition of* ε_q *is large enough, for* $w \le \varepsilon_p$*.*

Proof. Observe that the query manifold defined by $\mathbf{A}$ passes through the center, $\mathbf{B}$, of R by construction. Since each coordinate of $\mathbf{a}$ differs from $\mathbf{A}$ by at most ε_q, it thus follows that by setting C large enough, we can ensure that the distance between $\mathbf{B}$ and $\mathbf{a}$ is less than $\varepsilon_p/2$. Also observe that the width of the range along any axis will be $\Theta(w)$. The claim now follows by integrating the volume over vertical lines using Observation 9.4.2. $\square$

Lemma 9.4.2. *Consider a query* $\mathbf{a} \in \mathscr{Q}$ *and let* r *be the range that represents* $\mathbf{a}$ *in the parametric space defined by* R*. Consider an interval* $\mathscr{I}$ *on the* i*-th side of* R*, for some* i*. Let* $\mathsf{r}_{\mathscr{I}}$ *be the subset of* r *whose projection on the* i*-th side of* R *falls inside* $\mathscr{I}$*. Then, the volume of* $\mathsf{r}_{\mathscr{I}}$ *is* $O(\mathit{Vol}'(R)w|\mathscr{I}|/\varepsilon_p)$*.*

Proof. Both claims follow through Observation 9.4.2 by integrating the corresponding volumes over lines parallel to axes. $\square$

Using the Framework

Observe that by the above Observation 9.4.3, setting $w = \Theta(\frac{Q}{n}\varepsilon_p)$ satisfies Condition 1 of the lower bound framework in Theorem 9.2.1.

Satisfying Condition 2 requires a bit more work however. To do that, consider two queries defined by points $\mathbf{a}_1$ and $\mathbf{a}_2$. Let r_1 and r_2 be the two corresponding ranges in the parametric space of R.

To satisfy Condition 2, assume for contradiction that the volume of $\mathsf{r}_1 \cap \mathsf{r}_2$ is large, i.e., $\omega(\mathrm{Vol}(R)/(n\psi))$ where $\psi = 2^{\sqrt{\log n}}$. We now combine Observation 9.4.1, and Corollary 9.2.2 with parameter ϑ set to $\varepsilon_0 \frac{\varepsilon_p}{Q\psi}$ where ε_0 is a small enough constant and where X_1 represents $b_{1,1}$, X_2 represents $b_{2,2}$ and the remaining indeterminates represent the rest of variables in $\mathbf{b}$; note that the value of d in Corollary 9.2.2 is $\beta = 2(d-1)$ and $\mathscr{U} = \binom{2+1}{2} = 3$. Observe that each interval $\mathscr{I}_i$ determined by Corollary 9.2.2 defines a slab parallel to the i-th axis in R; let R_{bad} be the union of these slabs. By Lemma 9.4.2, and choice of small enough ε_0, a positive fraction of the intersection volume of r_1 and r_2 must lie outside R_{bad}. In addition, Corollary 9.2.2 allows us to pick some fixed values for all variables in $\mathbf{b}$, except for $b_{1,1}$ and $b_{2,2}$ with the property the final polynomials H_1 and H_2 (on indeterminates $b_{1,1}$ and $b_{2,2}$) that we obtain have the property that they have at least one coefficient which differs by

$$\Omega\left(\tau\left(\varepsilon_0\frac{\varepsilon_p}{Q\psi}\right)^{3(\beta-2)}\right) \tag{9.9}$$

between them; we call this operation of plugging values for all $\mathbf{b}$ except for $b_{1,1}$ and $b_{2,2}$ *slicing*. After slicing, we are reduced to the bivariate case; consider the set of points on which both H_1 and H_2 have value $O(w)$. If the 1D interval length of such points is $O(\varepsilon_p/(Q\psi)$, we call this a *good* slice, otherwise a *bad* slice. By Lemma 9.4.2, there must be bad slices since if all the slices are good, by integration of the intersection area of r_1 and r_2 over all the remaining variables in $\mathbf{b}$, r_1 and r_2 intersect with volume $O(\text{Vol}(R)/(n\psi))$, a contradiction.

We now show that we can arrive at a contradiction, assuming the existence of a bad slice. Given a bad slice, and any constant ℓ, we can find ℓ points $(x_1,y_1),\ldots,(x_\ell,y_\ell)$ such that $|x_{k_1}-x_{k_2}| = \omega(\varepsilon_p/(Q\psi))$ for all $1 \le k_1 < k_2 \le \ell$ and that $H_1(x_k,y_k), H_2(x_k,y_k) = O(w)$ for all $k \in \{1,2\cdots,\ell\}$. Observe that $H_i(x,y)$ has only monomials y, x, xy and a constant term. The critical observation here is that the coefficient of the monomial xy is always 1 since the coefficient of the monomial $b_{1,1}b_{2,2}$ was 1 and there was no monomial of degree three in P_1, meaning, after slicing this coefficient will not change. We pick $\ell = 3$ and thus we tweak all the three other coefficients of H_1. Tweaking H_1 such that $\tilde{H}_1(x_k,y_k) = H_2(x_k,y_k)$ corresponds to solving a linear system of equations that come from evalutions of monomials X, Y, and a constant term at points (x_k,y_k). We can thus use Lemma 9.3.1 with $\Delta_1 = \Delta_F = 1$, $\lambda = \omega(\varepsilon_p/(Q\psi))$. Observe that $\text{Res}(G,F)$ here is a constant by the properties of our construction. Also observe that by Lemma 9.3.1, the magnitude of the determinant of matrix A defined in Lemma 9.3.1 is

$$\omega\left((\varepsilon_p/(Q\psi)))^9\right).$$

By the same argument in [AC22], this means that the tweaking operation can be done such that each coefficient of H_1 is changed by

$$o\left((\varepsilon_p/(Q\psi)))^{-9}w\right). \tag{9.10}$$

We observe that after tweaking, $\tilde{H}_1$ and H_2 must coincide since by Lemma 9.3.1, the determinant of the relevant monomials is non-zero and thus there's a unique polynomial that passes through points $(x_1,y_1),\cdots,(x_\ell,y_\ell)$. Finally, to get a contradiction, we simply need to ensure that Eq. (9.10) is asymptotically smaller than Eq. (9.9). This yields a bound for the value of τ,

$$\tau = \Theta\left(w(Q\psi)^{3(\beta-2)+9}\right) = \Theta\left(w(Q\psi)^{3\beta+3}\right) \tag{9.11}$$

where we have assumed that ε_p, and ε_0 are small enough constants that have been absorbed in the $\Theta(\cdot)$ notation. Thus, this choice of τ will make sure that Condition 2 of the framework is also satisfied. It remains to calculate the number of queries that have been generated. Observe that τ was the side-length of a small enough grid around the point $\mathbf{A}$ in a β-dimensional space. Thus, the number of queries we generated is

$$m = \overset{o}{\Omega}\left(\left(\frac{1}{\tau}\right)^{\beta}\right) = \overset{o}{\Omega}\left(\frac{n^{\beta}}{Q^{\beta(3\beta+4)}}\right). \tag{9.12}$$

Applying Theorem 9.2.1 yields a space lower bound of

$$S(n) = \overset{o}{\Omega}(mQ) = \overset{o}{\Omega}\left(\frac{n^{2(d-1)}}{Q^{4(3d-1)(d-1)-1}}\right) \tag{9.13}$$

for line-hyperslab intersection reporting since $\beta = 2(d-1)$. One can verify that the same argument works for triangle-triangle intersection reporting in $\mathbb{R}^4$, since P_2 is also a multilinear polynomial of degree two. In this case, $\beta = 6$ which yields a space lower bound of

$$S(n) = \overset{o}{\Omega}\left(\frac{n^6}{Q^{125}}\right). \tag{9.14}$$

To sum up, we obtain the following results:

Theorem 9.4.1. *Any data structure that solves line-hyperslab intersection reporting in $\mathbb{R}^d$ must satisfy a space-time tradeoff of $S(n) = \overset{o}{\Omega}\left(\frac{n^{2(d-1)}}{Q(n)^{(4(3d-1)(d-1)-1}}\right)$.*

Theorem 9.4.2. *Any data structure that solves triangle-triangle intersection reporting in $\mathbb{R}^4$ must satisfy a space-time tradeoff of $S(n) = \overset{o}{\Omega}\left(\frac{n^6}{Q(n)^{125}}\right)$.*

9.5 Conclusion and Open Problems

We study line-hyperslab intersecting reporting in $\mathbb{R}^d$ and triangle-triangle intersecting reporting in $\mathbb{R}^4$. We show that any data structure with $n^{o(1)} + O(k)$ query time must use space $\overset{o}{\Omega}(n^{2(d-1)})$ and $\overset{o}{\Omega}(n^6)$ for the two problems respectively. This matches the classical upper bounds for the small $n^{o(1)}$ query time case for the two problems and answer an open problem for lower bounds asked by Ezra and Sharir [ES22b]. Along the way, we generalize and develop the lower bound technique used in [AC23b, AC22].

The major open problem is how to show a lower bound for general intersection reporting between objects of t and $(d-t)$ dimensions or for flat semialgebraic objects as studied recently in [AAE$^+$22]. Many of our techniques work, however, one big challenge is that after applying Corollary 9.2.2, the leading coefficient changes and thus we can no longer guarantee big gaps between coefficients.

Appendices

9.A Proof of Lemma 9.2.1

Lemma 9.2.1. *Let $P(x) = \sum_{i=0}^{\Delta} a_i x^i$ and $Q(x) = \sum_{i=0}^{\Delta} b_i x^i$ be two univariate (constant) degree-Δ polynomials in $\mathbb{R}[x]$ and $|a_i - b_i| \geq \eta$ for some $0 \leq i \leq \Delta$.*

Suppose there is an interval $\mathscr{I}$ of x such that for every $x_0 \in \mathscr{I}$ we have $|P(x_0) - Q(x_0)| \leq w$, then the length of $\mathscr{I}$ is upper bounded by $O((w/\eta)^{1/\mathscr{U}})$, where $\mathscr{U} = \binom{\Delta+1}{2}$ and the $O(\cdot)$ notation hides constant factors that depend on Δ.

Proof. The proof is by contradiction. Assume for the sake of contradiction that $|\mathscr{I}| = \omega((w/\eta)^{1/\mathscr{U}})$. We pick $\Delta+1$ points $(u_1, v_1), \cdots, (u_{\Delta+1}, v_{\Delta+1})$ from $y = P(x)$ for $u_1, \cdots, u_{\Delta+1} \in \mathscr{I}$ such that $|u_{k_1} - u_{k_2}| = \Omega(|\mathscr{I}|)$ for $k_1 \neq k_2$. Let $(u_1, v_1 + \xi_1), \cdots, (u_{\Delta+1}, v_{\Delta+1} + \xi_{\Delta+1})$ be $\Delta+1$ points on $y = Q(x)$. By definition $|\xi_k| \leq w$ for all $k = 1, 2, \cdots, \Delta+1$. We would like to tweak coefficient a_i of $P(x)$ by δ_i for $i = 0, 1, \cdots, \Delta$ to obtain a polynomial $P'(x)$ such that $y = P'(x)$ and $y = Q(x)$ agree on $x = u_1, \cdots, u_{\Delta+1}$. Observe that to do that, for each k, we would like to have

$$v_k + \xi_k = \sum_{i=0}^{\Delta} (a_i + \delta_i) u_k^i \implies \xi_k = \sum_{i=0}^{\Delta} \delta_i u_k^i$$

where the last follows from $v_k = \sum_{i=0}^{\Delta} a_i u_k^i$. Thus, to perform the tweaking, each δ_i should satisfy the following system

$$\begin{bmatrix} 1 & u_1 & \cdots & u_1^{\Delta} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & u_{\Delta+1} & \cdots & u_{\Delta+1}^{\Delta} \end{bmatrix} \cdot \begin{bmatrix} \delta_0 \\ \vdots \\ \delta_{\Delta} \end{bmatrix} = \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_{\Delta+1} \end{bmatrix},$$

or $A \cdot \mathbf{m}\delta = \mathbf{m}\xi$. Note that A is a Vandermonde matrix. So by Theorem 9.2.3, $\det(A) = \Omega(|\mathscr{I}|^{\mathscr{U}})$, since $|u_{k_1} - u_{k_2}| = \Omega(|\mathscr{I}|)$. This shows that there exists a unique solution for δ_i and thus we can perform the tweaking. In addition, it follows that $|\delta_i| = O(\frac{\xi}{\det(A)}) = O(\frac{w}{\det(A)})$. Now observe that if we assume $|\mathscr{I}| = \omega((w/\eta)^{1/\mathscr{U}})$, it follows that $|\delta_i| = o(\eta)$. Since P', Q have $\Delta+1$ points in common, they must be equivalent. This mean $|a_i - b_i| = o(\eta)$ for all $0 \leq i \leq \Delta$, a contradiction. □

9.B Proof of Lemma 9.2.2

Lemma 9.2.2. *Let $P_1(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} A_{\mathbf{i}}X^{\mathbf{i}}$ and $P_2(X) = \sum_{\mathbf{i}\in I_{d,\Delta}} B_{\mathbf{i}}X^{\mathbf{i}}$ be two d-variate degree-Δ polynomials in $\mathbb{R}[X]$ and $|A_{\mathbf{i}} - B_{\mathbf{i}}| \geq \eta_d$ for some $\mathbf{i} \in I_{d,\Delta}$.*

Suppose for each assignment $X_d \in \mathscr{I}_d$ to P_1, P_2, where $\mathscr{I}_d$ is an interval for X_d, all the coefficients of the resulting $(d-1)$-variate polynomial $Q_1(X_1,\cdots X_{d-1})$ and $Q_2(X_1,\cdots X_{d-1})$ differ by at most η_{d-1}, then $|\mathscr{I}_d| = O((\eta_{d-1}/\eta_d)^{1/\mathscr{U}})$.

Proof. Note that

$$P_1 = \sum_{\mathbf{i}\in I} A_{\mathbf{i}}X^{\mathbf{i}} = \sum_{\mathbf{j}\in I_{:d-1}} h_{\mathbf{j}}(X_d)X_{:d-1}^{\mathbf{j}},$$

and

$$P_2 = \sum_{i\in I} B_{\mathbf{i}}X^{\mathbf{i}} = \sum_{\mathbf{j}\in I_{:d-1}} \hat{h}_{\mathbf{j}}(X_d)X_{:d-1}^{\mathbf{j}},$$

where $I_{:d-1}$ stands for the set of $(d-1)$-tuples formed by taking the $(d-1)$ entries of every d-tuple in I and $X_{:d-1} = (X_1,\cdots,X_{d-1})$, and the coefficients of $X_{:d-1}$ are polynomials in X_d of form

$$h_{\mathbf{j}}(X_d) = \sum_{i=0}^{\Delta} A_{\mathbf{j}\oplus i}X_d^i,$$

and

$$\hat{h}_{\mathbf{j}}(X_d) = \sum_{i=0}^{\Delta} B_{\mathbf{j}\oplus i}X_d^i,$$

where $\mathbf{j}\oplus i$ stands for a d-tuple formed by appending i to $\mathbf{j}$. Since $P_1 \not\equiv P_2$, there must exists one $\mathbf{j}$ such that $h_{\mathbf{j}}(X_d) - \hat{h}_{\mathbf{j}}(X_d) \not\equiv 0$. Then by Lemma 9.2.1, the interval length for X_d in which $|h_{\mathbf{j}}(X_d) - \hat{h}_{\mathbf{j}}(X_d)| \leq \eta_{d-1}$ is upper bounded by $O(\eta_{d-1}/\eta_d)^{1/\mathscr{U}})$. By a union bound over all monomials, the lemma follows. $\square$

Chapter 10

On Range Summary Queries

Abstract

We study the query version of the approximate heavy hitter and quantile problems. In the former problem, the input is a parameter ε and a set P of n points in $\mathbb{R}^d$ where each point is assigned a color from a set C, and the goal is to build a structure such that given any geometric range γ, we can efficiently find a list of approximate heavy hitters in $\gamma \cap P$, i.e., colors that appear at least $\varepsilon|\gamma \cap P|$ times in $\gamma \cap P$, as well as their frequencies with an additive error of $\varepsilon|\gamma \cap P|$. In the latter problem, each point is assigned a weight from a totally ordered universe and the query must output a sequence S of $1 + 1/\varepsilon$ weights such that the i-th weight in S has approximate rank $i\varepsilon|\gamma \cap P|$, meaning, rank $i\varepsilon|\gamma \cap P|$ up to an additive error of $\varepsilon|\gamma \cap P|$. Previously, optimal results were only known in 1D [WY11] but a few sub-optimal methods were available in higher dimensions [AW17, ACH+13].

We study the problems for two important classes of geometric ranges: 3D halfspace and 3D dominance queries. It is known that many other important queries can be reduced to these two, e.g., 1D interval stabbing or interval containment, 2D three-sided queries, 2D circular as well as 2D k-nearest neighbors queries. We consider the real RAM model of computation where integer registers of size w bits, $w = \Theta(\log n)$, are also available. For dominance queries, we show optimal solutions for both heavy hitter and quantile problems: using linear space, we can answer both queries in time $O(\log n + 1/\varepsilon)$. Note that as the output size is $\frac{1}{\varepsilon}$, after investing the initial $O(\log n)$ searching time, our structure takes on average $O(1)$ time to find a heavy hitter or a quantile! For more general halfspace heavy hitter queries, the same optimal query time can be achieved by increasing the space by an extra $\log_w \frac{1}{\varepsilon}$ (resp. $\log\log_w \frac{1}{\varepsilon}$) factor in 3D (resp. 2D). By spending extra $\log^{O(1)} \frac{1}{\varepsilon}$ factors in both time and space, we can also support quantile queries.

We remark that it is hopeless to achieve a similar query bound for dimensions 4 or higher unless significant advances are made in the data structure side of theory of geometric approximations.

10.1 Introduction

Range searching is an old and fundamental area of computational geometry that deals with storing an input set $P \subset \mathbb{R}^d$ of n (potentially weighted) points in a data structure such that given a query range γ, one can answer certain questions about the subset of points inside γ. Range searching is often introduced within a general framework that allows a very diverse set of questions to be answered. For instance, if the points in P have been assigned integer or real weights, then one can count the points in γ (range counting), sum the total weights of the points in γ (weighted range counting), or find the maximum or minimum weight in γ (range max or min queries).

However, there are some important questions that cannot be answered within this general framework. Consider the following motivating example: our data includes the locations of houses in a city as well as their estimated values and given a query range γ, we are interested in the distribution of the house values within γ, for example, we might be interested to see if there's a large inequality in house values or not. Through classical results, we can find the most expensive and the least expensive houses (max and min queries), and the average value of the houses (by dividing the weighted sum of the values by the total number of houses in γ). Unfortunately, this information does not tell us much about the distribution of the house values within γ, e.g., one cannot compute the Gini index which is a widely-used measure of inequality of the distribution. Ideally, to know the exact distribution of values within γ, one must have all the values inside γ, which in the literature is known as a *range reporting* query which reports all the points inside the query range γ. However, this could be an expensive operation, e.g., it can take $\Omega(n)$ time if the query contains a constant fraction of the input points. A reasonable alternative is to ask for a "summary" query, one that can summarize the distribution. In fact, the streaming literature is rich with many important notions of summary that are used to concisely represent a large stream of data approximately but with high precision. Computing ε-quantiles can be considered as one of the most important concepts for a succinct approximation of a distribution and it also generalizes many of the familiar concepts, e.g., 0-quantile, 0.5-quantile, and 1-quantile that are also known as the minimum, the median, and the maximum of S. We now give a formal definition below.

Quantile summaries. Given a sequence of values $w_1 \leq \cdots \leq w_k$, a δ-quantile, for $0 \leq \delta \leq 1$, is the value with rank $\lfloor \delta k \rfloor$. By convention, 0-quantile and 1-quantiles are set to be the minimum and the maximum, i.e., w_1 and w_k respectively. An ε-quantile summary is then defined as the list of $1+\varepsilon^{-1}$ values where the i-th value is the $i\varepsilon$-quantile, for $i = 0, \cdots, \varepsilon^{-1}$. As we will review shortly, computing exact quantiles is often too expensive so instead we focus on approximations. We define an approximate ε-quantile summary (AQS) to be a sequence of $1+\varepsilon^{-1}$ values where the i-th value is between the $(i-1)$-quantile and the $(i+1)$-quantile[1], for $i = 0, \cdots, \varepsilon^{-1}$. An approximate quantile summary with a reasonably small choice of ε can give a very

[1]For $a \leq 0$ (resp. $a \geq k$), we define the a-quantile to be the 0-quantile (resp. k-quantile).

good approximation of the distribution. It also has the benefit that the query needs to output only $O(\varepsilon^{-1})$ values, regardless of the number of points inside the query range.

To obtain a relatively precise approximation of the distribution, ε needs to be chosen sufficiently small, and thus we consider it an additional parameter (and thus not a constant). This is also similar to the literature on streaming where the dependency on ε is important.

Problem Definition, Previous Work, and Related Results

One of our main problems is the problem of answering approximate quantile summary (AQS) queries which is defined as follows.

Problem 10.1.1 (Approximate quantile summaries)**.** *Consider an input set P of n points in $\mathbb{R}^d$ where each point $p \in P$ is assigned a weight w_p from a totally ordered universe. Given a value ε, we are asked to build a structure such that given a query range γ, it can return an AQS of $P \cap \gamma$ efficiently.*

It turns out that another type of "range summary queries" is extremely useful for building data structures for AQS queries.

Heavy hitter summaries. Consider a set P of k points where each point in P is assigned a color from the set $[n]$. Let f_i be the frequency of color i in P, i.e., the number of times color i appears among the points in P. A *heavy hitter summary (HHS)* with parameter ε, is the list of all the colors i with $f_i \geq \varepsilon k$ together with the value f_i. As before, working with exact HHS will result in very inefficient data structures and thus once again we turn to approximations. An *approximate heavy hitter summary (AHHS)* with parameter ε is a list, L, of colors such that every color i with $f_i \geq \varepsilon k$ is included in L and furthermore, every color $i \in L$ is also accompanied with an approximation, f'_i, of its frequency such that $f_i - \varepsilon k \leq f'_i \leq f_i + \varepsilon k$.

Problem 10.1.2 (Approximate heavy hitters summaries)**.** *Consider an input set P of n points in $\mathbb{R}^d$ where each point in P is assigned a color from the set $[n]$. Given a parameter ε, we are asked to build a structure such that given a query γ, it can return an AHHS of the set $P \cap \gamma$.*

Observe that in both problems, the output size of a query is $O(1/\varepsilon)$ in the worst-case. Our main focus is to obtain data structures with the optimal worst-case query time of $O(\log n + \varepsilon^{-1})$. Note that it makes sense to define an *output-sensitive* variant where the query time is $O(\log n + k)$ where k is the output size. E.g., it could be the case for a AHHS query that the numbrer of heavy hitters is much fewer than ε^{-1}. This makes less sense for AQS queries, since unless the distribution of weights inside the query range γ is almost constant, an AQS will have $\Omega(\varepsilon^{-1})$ distinct values. As our main focus is on AQS, we only consider AHHS data structures with the worst-case query time of $O(\log n + \varepsilon^{-1})$.

A note about the notation. To reduce the clutter in the expressions of query time and space, we adopt the convention that $\log(\cdot)$ function is at least one, e.g., we define $\log_a b$ to be $\max\{1, \frac{\ln b}{\ln a}\}$ for any positive values a, b.

Previous Results

As discussed, classical range searching solutions focus on rather simple queries that can return sum, weighted sum, minimum, maximum, or the full list of points contained in a given query range. This is an extensively researched area with numerous results to cite and so we refer the reader to an excellent survey by Agarwal [Aga17] that covers such classical results.

However, classical range searching data structures cannot give detailed statistical information about the set of points contained inside the query region, unless one opts to report the entire subset of points inside the query range, which could be very expensive if the set is large. Because of this, there have been a number of attempts to answer more informative queries. For example, "range median" queries have received quite a bit of attention [BKMT05, BGJrS11, JrL11]. Note that the median is the same as 0.5-quantile and thus these can be considered the first attempts at answering quantile queries. However, optimal solution (linear space and logarithmic query time) to exact range median queries has only be found in 1D [BGJrS11]. For higher dimensions, to the best of our knowledge, the only known technique is to reduce the problem to several range counting instances [BGJrS11, CZ15], and it is a major open problem in the range searching field to find efficient data structures for exact range counting. Due to this barrier, the approximate version of the problem [BKMT05] has been studied.

Data summary queries have also received some amount of attention, especially in the context of geometric queries. Agarwal et al. [ACH$^+$13] showed that the heavy hitters summary (as well as a few other data summaries) are "mergeable" and this gives a baseline solution for a lot of different queries in higher dimensions, although a straightforward application of their techniques gives sub-optimal dependency on ε. In particular, for $d = 2$ and for halfspace (or simplex) queries it yields a linear-space data structure with $O(\frac{\sqrt{n}}{\varepsilon})$ query time. For $d = 3$ the query time will be $O(n^{2/3}/\varepsilon)$. In general, in the naive implementation, the query time will be $O(f(n)/\varepsilon)$ where $f(n)$ is the query time of the corresponding "baseline" range searching query (see Table 1 for more information). A more efficient approach towards merging of summaries was taken by [HY17] where they study the problem in a communication complexity setting, however, it seems possible to adopt their approach to a data structure as well, in combination with standard application of partition trees; after building an optimal partition tree, for any node v in the tree, consider it as a player in the communication problem with the subset of points in the subtree of v as its input. At the query time, after identifying $O(n^{2/3})$ subsets that cover the query range, the goal would be to merge all the summaries involved. By plugging the results in [HY17] this can result in a linear-space data structure with query time of $\tilde{O}(n^{2/3} + n^{1/6}\varepsilon^{-3/2})$.

The issue of building optimal data structures for range summary queries was only tackled in 1D by Wei and Yi [YWW14]. They built a data structure for answering a number of summary queries, including heavy hitters queries, and showed it is possible to obtain an optimal data structure with $O(n)$ space and $O(\log n + 1/\varepsilon)$ query time. Beyond this, only sub-optimal solutions are available. Recently, there have been efforts to tackle "range sampling queries" where the goal is to extract k random samples from the set $|P \cap \gamma|$ [AP19, AW17, HQT14]. In fact, one of the main motivations to consider range sampling queries was to gain information about the distribution of the point set inside the query [AP19]. In particular, range sampling provides a general solution for obtaining a "data summary" and for example, it is possible to solve the heavy hitters query problem. However, it has a number of issues, in particular, it requires sampling at least $1/\varepsilon^2$ points from the set $|P \cap \gamma|$, and even then it will only provide a Monte Carlo type approximation which means to boost the probabilistic guarantee, even more points need to be sampled. For example, to get a high probability guarantee, $\Omega(\varepsilon^{-2} \log n)$ samples are required.

Type-2 Color Counting. These queries were introduced in 1995 by Gupta et al. [GJS95] within the area of "colored range counting". In this problem, given a set of colored points, we want to report the frequencies of all the colors that appeared in a given query range. This is a well-studied problem, but mostly in the orthogonal setting, see e.g., [CHN20].

AHHS queries can be viewed as approximate type-2 color counting queries but with an additive error. Consider a query with k points. If we allow error εk in type-2 counting, then we can ignore colors with frequencies fewer than εk but otherwise we have to report frequencies with error εk, which is equivalent to answering an AHHS query.

Other Related Problems. Karpinski and Nekrich [KN08] studied the problem of finding the most frequent colors in a given (orthogonal) query range. This problem has received further attention in the community [BGN13, BGM+21, DHM+13]. But the problem changes fundamentally when we introduce approximations.

The Model of Computation. Our model of computation is the real RAM where we have access to real registers that can perform the standard operations on real numbers in constant time, but we also have access to $w = \Theta(\log n)$ bits long integer registers that can perform the standard operations on integers and extra nonstandard operations which can be implemented by table lookups since we only need binary operations on fewer than $\frac{1}{2} \log n$ bits. Note that our data structure works when the input coordinates are real numbers, however, at some point, we will make use of the capabilities of our model of computation to manipulate the bits inside its integer registers.

Our Contributions

Our main results and a comparison with the previously known results are shown in Table 1.

Overall, we obtain a series of new results for 3D AHHS and AQS query problems which improve the current results via mergeability and independent range sampling [ACH$^+$13, AW17] by up to a huge multiplicative $n^{\Omega(1)}$ factor in query time with almost the same linear-space usage. This improvement is quite nontrivial and requires an innovative combination of known techniques like the shallow cutting lemma, the partition theorem, ε-approximations, as well as some new ideas like bit-packing for nonorthogonal queries, solving AQS query problem using AHHS instances, rank-preserving geometric sampling and so on.

For dominance queries, we obtain the first optimal results. When $\varepsilon^{-1} = O(\log n)$ our halfspace AHHS results are also optimal. Note that for small values of ε, our halfspace AHHS results yield significant improvements in the query time over the previous approaches. Along the way, we also show improved results of the above problems for 2D as well as a slightly improved exact type-2 simplex color counting result.

10.2 Preliminaries

In this section, we introduce the main tools we will use in our results. For a comprehensive introduction to the tools we use, see Appendix 10.A and Appendix 10.B.

Shallow Cuttings and Approximate Range Counting

Given a set H of n hyperplanes in $\mathbb{R}^3$, the level of a point $q \in \mathbb{R}^3$ is the number of hyperplanes in H that pass below q. We call the locus of all points of level at most k the $(\leq k)$-level and the boundary of the locus is the k-level. A shallow cutting $\mathscr{C}$ for the $(\leq k)$-level of H (or a k-shallow cutting for short) is a collection of disjoint cells (tetrahedra) that together cover the $(\leq k)$-level of H with the property that every cell $C \in \mathscr{C}$ in the cutting intersects a set H_C, called the conflict list of C , of $O(k)$ hyperplanes in H. The shallow cutting lemma is the following.

Lemma 10.2.1. *For any set of n hyperplanes in $\mathbb{R}^3$ and a parameter k, there exists an $O(k/n)$-shallow cutting of size $O(n/k)$ that covers the $(\leq k)$-level. The cells in the cutting are all vertical prisms unbounded from below (tetrahedra with a vertex at $(0,0,-\infty)$).*

Furthermore, we can construct these cuttings for all k of form a^i simultaneously in $O(n\log n)$ time for any $a > 1$. Given any point $q \in \mathbb{R}^3$, we can find the smallest level k that is above q as well the cell containing q in $O(\log n)$ time.

The above can also be applied to dominance ranges, which are defined as below. Given two points p and q in $\mathbb{R}^d$, p dominates q if and only if every coordinate of p is larger or equal to that of q. The subset of $\mathbb{R}^d$ dominated by p is known as a *dominance*

Table 1: Our main results compared with Mergeability-based [ACH$^+$13] and Independent Range Sampling (IRS)-based [AW17] solution. The IRS-based solutions are randomized with success probability $1-\delta$ for a parameter $0<\delta<1$. F is the number of colors of the input. $w=\Theta(\log n)$ is the word size of the machine. † indicates optimal solutions.

Summary Query Types	**Space**	**Query Time**	**Remark**
Type-2 Simplex Color Counting	$O(n)$	$O\left(n^{1-\frac{1}{d}}+\frac{n^{1-\frac{1}{d}}F^{\frac{1}{d}}}{w^{\alpha}}\right)$	New
3D AHHS Halfspace	$O(n)$ $O(n)$ $O(n)$ $O(n\log_w\frac{1}{\varepsilon})$	$O(\log n+\frac{1}{\varepsilon}n^{2/3})$ $\tilde{O}(n^{2/3}+\frac{1}{\varepsilon^{3/2}}n^{1/6})$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon})$	Mergeability-based [ACH$^+$13] Monte Carlo [HY17] IRS-based [AW17] **New**
3D AHHS Dominance	$O(n)$ $O(n)$ $O(n)$	$O(\log n+\frac{1}{\varepsilon}\log^3 n)$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon})$	Mergeability-based [ACH$^+$13] IRS-based [AW17] **New†**
3D AQS Halfspace	$O(n)$ $O(n)$ $O(n\log^2\frac{1}{\varepsilon}\log_w\frac{1}{\varepsilon})$	$O(\log n+\frac{1}{\varepsilon}n^{2/3}\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})$	Mergeability-based [ACH$^+$13] IRS-based [AW17] **New**
3D AQS Dominance	$O(n)$ $O(n)$ $O(n)$	$O(\log n+\frac{1}{\varepsilon}\log^3 n\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $O(\log n+\frac{1}{\varepsilon})$	Mergeability-based [ACH$^+$13] IRS-based [AW17] **New†**

range. When the query range in a range searching problem is a dominance range, we refer to it as a *dominance query*.

As observed by Chan et al. [CLP11], dominance queries can be simulated by a halfspace queries and thus Lemma 10.2.1 applies to them. See Appendix 10.B for details.

We obtain the approximate version of the range counting result using shallow cuttings.

Theorem 10.2.1 (Approximate Range Counting [AHZ10]). *Let P be a set of n points in $\mathbb{R}^3$. One can build a data structure of size $O(n)$ for halfspace or dominance ranges such that given a query range γ, one can report $|\gamma\cap P|$ in $O(\log n)$ time with error $\alpha|\gamma\cap P|$ for any constant $\alpha>0$.*

ε-approximation

Another tool we will use is ε-approximation, which is a useful sampling technique:

Definition 10.2.1. *Let (P,Γ) be a finite set system. Given any $0 < \varepsilon < 1$, a set $A \subseteq P$ is called an ε-approximation for (P,Γ) if for any $\gamma \in \Gamma$, $\left|\frac{|\gamma \cap A|}{|A|} - \frac{|\gamma \cap P|}{|P|}\right| \leq \varepsilon$.*

The set A above allows us to approximate the number of points of $\gamma \cap P$ with additive error of $\varepsilon|P|$ by computing $|\gamma \cap A|$ exactly; essentially, ε-approximations reduce the approximate counting problem on the (big) set P to the exact counting problem on the (small) set A.

It has been shown that small-sized ε-approximations for set systems formed by points and halfspaces/dominance ranges exist:

Theorem 10.2.2 (ε-approximation [Mat10, Phi16])**.** *There exist ε-approximations of size $O(\varepsilon^{-\frac{2d}{d+1}})$ and $O(\varepsilon^{-1}\log^{d+1/2}\varepsilon^{-1})$ for halfspace and dominance ranges respectively.*

10.3 Approximate Heavy Hitter Summary Queries

We solve approximate quantile summary (AQS) queries using improved results for approximate heavy hitter summary (AHHS) queries. We sketch the main ideas of our new AHHS solutions in this section and refer the readers to Appendix 10.D for details. We show the following.

Theorem 10.3.1. *For $d = 3$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

Theorem 10.3.2. *For $d = 2$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

Theorem 10.3.3. *For $d = 2,3$, the approximate dominance heavy hitter summary queries can be answered using the optimal $O(n)$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

Base Solution

The above results are built from a *base solution*, which solves the following problem:

Problem 10.3.1. *[Coarse-Grained AHHS Queries] Let P be a set of points in $\mathbb{R}^d$, each associated with a color. The problem is to store P in a structure such that given a query range q, one can estimate the frequencies of colors in $q \cap P$ with an additive error up to $\varepsilon|P|$ efficiently for some parameter $0 < \varepsilon < 1$.*

Note that here we allow more error (since the error is defined in the entire point set). To solve Problem 10.3.1, one crucial component we need is a better (exact) type-2 color counting structure for halfspaces. We combine several known techniques in a novel way with bit-packing to get the following theorem. See Appendix 10.C for details.

Theorem 10.3.4. *Given an integer parameter F, a set P of n points in $\mathbb{R}^d$ where each point is assigned a color from the set $[F]$, one can build a linear-sized data structure, such that given a query simplex q, it can output the number of times each color appears in $P \cap q$ in total time* $\max\{O(n^{(d-1)/d}), O(n^{(d-1)/d}F^{1/d}/w^{\alpha})\}$, *for some appropriate constant α and word size w.*

The main idea for getting a base solution is relatively straightforward. We group colors according to their frequencies where each group contains colors of roughly equal frequencies. However, we have to be careful about the execution and the analysis is a bit tricky. For example, if we place all the points in one copy of the data structure of Theorem 10.3.4, then we will get a sub-optimal result. However, by grouping the points correctly, and being stringent about the analysis, we can obtain the following.

Theorem 10.3.5. *For $d \geq 3$, Problem 10.3.1 for simplex queries (the intersection of $d+1$ halfspaces) can be solved with $O(X)$ space for $X = \min\{|P|, \varepsilon^{-\frac{2d}{d+1}}\}$ and a query time of*

$$O\left(\frac{|P|^{1-\frac{2}{d-1}}}{w^{\alpha}\varepsilon^{\frac{2}{d-1}}}\right) + O\left(X^{\frac{d-1}{d}}\right)$$

where w is the word-size of the machine and α is some positive constant.

The main challenge is that we have two cases for the size of an ε-approximation on n points since it is bounded by $\min\left\{n, O(\varepsilon^{-\frac{2d}{d+1}})\right\}$ and also two cases for the query time of Theorem 10.3.4. However, the main idea is that since the total error budget is $\varepsilon|P|$, we can afford to pick a larger error parameter $\varepsilon_i = \frac{\varepsilon|P|}{|P_i|}$, where P_i is the set of points with color i. The details are presented in the full version.

Solving AHHS Queries

We first transform the problem into the dual space. So the point set P becomes a set H of hyperplanes and any query halfspace becomes a point q. We want to find approximate heavy hitters of hyperplanes of H below q. Here, we remark that obtaining a data structure with $O(n \log \frac{1}{\varepsilon_0})$ space is not too difficult: build a hierarchy of shallow cuttings covering level $2^i/\varepsilon_0$ for $i = 0, 1, \cdots, \log(\varepsilon_0 n)$ of the arrangement of H. For each shallow cutting cell Δ_C, we build the previous base structure for the conflict list $\mathscr{S}_{\Delta_C}$ for a parameter $\varepsilon = \varepsilon_0/c$ for a big enough constant c. Then, observe that for queries below level ε_0^{-1}, we can spend $O(\log n + \frac{1}{\varepsilon_0})$ time to find all the hyperplanes passing below the query and answer the AHHS queries explicitly and also for shallow cutting levels above level $\varepsilon_0^{-3/2}$, the total amount of space used by the base solution is $O(n)$. Thus, it turns out that the main difficulty lies in handling the levels between ε_0^{-1} and $\varepsilon_0^{-3/2}$.

To reduce the space to $O(\log_w \frac{1}{\varepsilon_0})$, recall that in the query time of the base structure, we have two terms $O(1/(\varepsilon_0 w^\alpha))$ and $O(X^{2/3})$.

Observe that we can afford to set ε to be roughly ε_0/w^α and the first term will still be $O(\varepsilon_0^{-1})$ because we are at level below $\varepsilon_0^{-3/2}$, we have $X < \varepsilon_0^{-3/2}$ and so the second term will always be $O(\varepsilon_0^{-1})$! The effect of setting $\varepsilon = \varepsilon_0/w^\alpha$ is that now the base structure we built for a cell can output frequencies with a factor of w^α more precision, meaning it can be used for a factor of w^α many more levels. So we only need to build the base structure for shallow cuttings built for a factor of w^α! This gives us the $O(n \log_w \frac{1}{\varepsilon_0})$ space bound. Of course, here the output has size $O(\varepsilon^{-1}) = O(w^\alpha \varepsilon_0^{-1})$ and we cannot afford to examine all these colors. The final ingredient here is that we can maintain a list of $O(\varepsilon_0^{-1})$ candidate colors using shallow cuttings built for a factor of 2.

We remark that although the tools are standard, the combination of the tools and the analysis are quite nontrivial. Also when we have $\Theta(1/\varepsilon_0)$ heavy hitters, our query time is optimal. It is an interesting open problem if the query time can be made output sensitive.

10.4 Approximate Quantile Summary Queries

In this section, we solve Problem 10.1.1. We first show a general technique that uses our solution AHHS queries to obtain an efficient solution for AQS queries. We show that for halfspace and 3D dominance ranges we can convert the solution for AHHS queries to a solution for AQS queries with an $O(\log^2 \frac{1}{\varepsilon})$ blow up in space and time. Then in Section 10.4, we present an optimal solution based on a different idea for dominance ranges.

First, we show how to solve AQS queries using the AHHS query solution. We describe the data structure for halfspaces, since as we have mentioned before, the same can be applied to dominance ranges in 3D as well. The high level idea of our structure is as follows: We first transform the problem into the dual space. This yields the problem instance where we have n weighted hyperplanes and given a query point q, we would like to extract an approximate quantile summary for the hyperplanes that pass below q. To do this, we build hierarchical shallow cuttings. For each cell in each cutting, we collect the hyperplanes in its conflict list and then divide them into $O(\frac{1}{\varepsilon_0})$ groups according to the increasing order of their weights. Given a query point in the dual space, we first find the cutting and the cell containing it, and then find an approximated rank of each group, within the subset below the query. This is done by generating an AHHS problem instance and applying Theorem 10.3.1. We construct the instance in a way such that the rank approximated will only have error small enough such that we can afford to scan through the groups and pick an arbitrary hyperplane in corresponding groups to form an approximate ε_0-quantile summary.

The Data Structure and the Query Algorithm

We dualize the set P of n input points which gives us a set $H = \overline{P}$ of n hyperplanes. We then build a hierarchy of shallow cuttings where the i-th shallow cutting, $\mathscr{C}_i$, is a k_i-shallow cutting where $k_i = \frac{2^i}{\varepsilon_0}$, for $i = 0, 1, 2, \cdots, \log(\varepsilon_0 n)$. Consider a cell Δ_C in the i-th shallow cutting and its conflict list $\mathscr{S}_{\Delta_C}$. Let $\varepsilon = \frac{\varepsilon_0}{c}$ for a big enough constant c. We partition $\mathscr{S}_{\Delta_C}$ into $t = \frac{1}{\varepsilon}$ groups $G_1, G_2, \cdots, G_t$ sorted by weight, meaning, the weight of any hyperplane in G_j is no larger than that of any hyperplane in G_{j+1} for $j = 1, 2, \cdots, t-1$.

For each group G_j, we store the smallest weight among the hyperplanes it contains, as its representative. To make the description shorter, we make the simplifying assumption that t is a power of 2 (if not, we can add some dummy groups). We arrange the groups G_j as the leaves of a balanced binary tree $\mathscr{T}$ and let $V(\mathscr{T})$ be the set of vertices of $\mathscr{T}$. Next, we build the following set A_Δ of colored hyperplanes, associated with Δ: Let $\varepsilon' = \frac{\varepsilon}{\log^2 t}$. For every vertex $v \in V(\mathscr{T})$, let G_v to be the set of all the hyperplanes contained in the subtree of v; we add an ε'-approximation, E_v, of G_v to A_Δ with color v. Using Theorem 10.3.1, we store the points dual to hyperplanes in A_Δ in a data structure Ψ_Δ for AHHS queries with error parameter ε'. This completes the description of our data structure.

The query algorithm. A given query q is answered as follows. Let us quickly go over the standard parts: We consider the query in the dual space and thus q is considered to be a point. Let k be the number of hyperplanes passing below q. Observe that by Theorem 10.2.1, we can find a $(1+\alpha)$ factor approximation, k^*, of k in $O(\log n)$ time for any constant α, using a data structure that consumes linear space. This allows us to find the first k_i-shallow cutting $\mathscr{C}_i$ with $k_{i-1} < k \leq k_i$. The cell $\Delta_C \in \mathscr{C}_i$ containing q can also be found in $O(\log n)$ time using a standard point location data structure (e.g., see [AC09]).

The interesting part of the query is how to handle the query after finding the cell Δ_C. Let H_q be the subset of H that lies below q. Recall that $\mathscr{S}_{\Delta_C}$ is the subset of H that intersects Δ_C. The important property of Δ_C is that $H_q \subset \mathscr{S}_{\Delta_C}$ and also $|\mathscr{S}_{\Delta_C}| = O(|H_q|) = O(k)$.

We query the data structure Ψ_{Δ_C} built for Δ_C to obtain a list of colors and their approximate counts where the additive error in the approximation is at most $\varepsilon'|A_{\Delta_C}|$. To continue with the description of the query algorithm, let us use the notation g_j to denote the subset of G_j that lies below q, and let $g = \cup_{j=1}^{t} g_j$ and thus $|g| = k$.

Note that while the query algorithm does not have direct access to g, or k, we claim that using the output of the data structure Ψ_{Δ_C}, we can calculate the approximate rank of the elements of g_i within g up to an additive error of $\varepsilon_0 k$. Again, we can use tree $\mathscr{T}$ to visualize this process. Recall that in Ψ_{Δ_C}, every vertex $v \in V(\mathscr{T})$ represents a unique color in the data structure Ψ_{Δ_C} and the data structure returns an AHHS summary with error parameter ε'. This allows us to estimate the number of elements of E_v that pass below q with error $\varepsilon'|A_{\Delta_C}|$ and since E_v is an ε'-approximation of G_v, this allows us to estimate the number of elements of G_v that pass below q with error at most $2\varepsilon'|A_{\Delta_C}|$. Consider the leaf node that represents $g_j \subset G_j$ and the path π that connects it to the root of $\mathscr{T}$. The approximate rank, r_j, of g_j is calculated as follows. Consider a subtree with root u that hang to the left of the path π (as shown in Figure 1). If color u does not appear in the output of the AHHS query, then we can conclude that at most $2\varepsilon'|A_{\Delta_C}|$ of its hyperplanes pass below q and in this case we do nothing. If it does appear in the output of the AHHS query, then we know the number of hyperplanes in its subtree that pass below q up to an additive error of $2\varepsilon'|A_{\Delta_C}|$ and in this case, we add this estimate to r_j. In both cases, we are off by an additive error of $2\varepsilon'|A_{\Delta_C}|$. We repeat this for every subtree that hangs to the left of π. The number of such subtree is at most $\log t$ and thus the total error is at most $2\varepsilon'|A_{\Delta_C}|\log t$. Now observe that

$$2\varepsilon'|A_{\Delta_C}|\log t = 2\frac{\varepsilon}{\log^2 t} \cdot \log t|\mathscr{S}_{\Delta_C}| \cdot \log t = O(\varepsilon k) = O\left(\frac{\varepsilon_0 k}{c}\right) \le \varepsilon_0 k$$

which follows by setting c large enough and observing the fact that $|A_{\Delta_C}| \le \log t|\mathscr{S}_{\Delta_C}|$ since every hyperplane in $\mathscr{S}_{\Delta_C}$ is duplicated $\log t$ times.

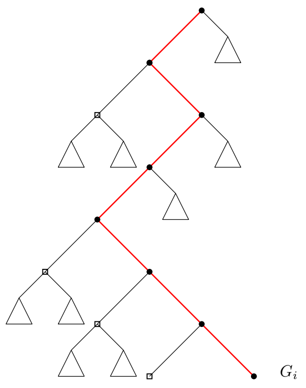


Figure 1: Compute the Approximate Rank of a Group: The approximated rank of G_i is calculated as the sum of all the approximate counts of square nodes.

We are now almost done. We just proved that in each g_i, we know the rank of its elements within g up to an additive error of $\varepsilon_0 k$. This means that picking one element from each G_i gives us a super-set of an AQS; in the last stage of the query algorithm we simply prune the unnecessary elements as follows: We scan all the leave in $\mathscr{T}$ from left to right, i.e., consider the group G_j for $j = 1$ to t and compute the quantile summary in a straightforward fashion. To be specific, we initialize a variable $j' = 0$ and then consider G_j, for $j = 1$ to t. The first time r_j exceeds a quantile boundary, i.e., $r_j \ge j'\varepsilon_0 k^*$, we add the hyperplane with the lowest weight in G_j to the approximate ε_0-quantile summary, and then increment j'.

Analysis

Based on the previous paragraph, the correctness is established. Thus, it remains to analyze the space and query complexities. We start with the former.

Space Usage. Consider the structure Ψ_{Δ_C} built for cell Δ_C from a k_i-shallow cutting $\mathscr{C}_i$. Observe that $\sum_{v\in V(\mathscr{T})}|G_v| = |\mathscr{S}_{\Delta_C}|\log t$ since in the sum every hyperplane will be counted $\log t$ times. E_v is an ε'-approximation of G_v and thus

$$|E_v| \le \min\left\{\varepsilon'^{-3/2}, G_v\right\} \tag{10.1}$$

which implies

$$|A_{\Delta_C}| = \sum_{v\in V(\mathscr{T})} |E_v| \le \min\left\{\varepsilon'^{-3/2}2t, |\mathscr{S}_{\Delta_C}|\log t\right\} \tag{10.2}$$

where the first part follows as there are at most $2t$ vertices in $\mathscr{T}$ and the second part follows from (10.1). We build an instance of Theorem 10.3.1 on the set A_{Δ_C} which by Theorem 10.3.1 uses $O(|A_{\Delta_C}|\log_w \frac{1}{\varepsilon'})$ space. Assuming Δ_C belongs to a k_i-shallow cutting $\mathscr{C}_i$, we have $|\mathscr{S}_{\Delta_C}| = O(k_i)$ and there are $O(n/k_i)$ cells in $\mathscr{C}_i$. Observe that

$$\sum_{\Delta_C\in\mathscr{C}_i}|A_{\Delta_C}| = \sum_{\Delta_C\in\mathscr{C}_i}\min\left\{\varepsilon'^{-3/2}2t, |\mathscr{S}_{\Delta_C}|\log t\right\} = \sum_{\Delta_C\in\mathscr{C}_i} O\left(\min\left\{\varepsilon'^{-3/2}t, k_i\log t\right\}\right) =$$
$$O\left(\min\left\{\frac{n}{k_i}\varepsilon_0^{-3}, n\log\frac{1}{\varepsilon_0}\right\}\right). \tag{10.3}$$

Thus, the total space used for $\mathscr{C}_i$ is

$$O\left(\min\left\{\frac{n}{k_i}\varepsilon_0^{-3}\log_w\frac{1}{\varepsilon_0}, n\log_w\frac{1}{\varepsilon_0}\log\frac{1}{\varepsilon_0}\right\}\right).$$

Finally, observe that there can be at most $O(\log\frac{1}{\varepsilon_0})$ levels where the second term dominates; to be specific, at least when k_i exceeds ε_0^{-4}, the first term dominates and the total space used by those levels is $O(n)$ as k_i's form a geometric series. So the total space usage of our structure is $O(n\log^2\frac{1}{\varepsilon_0}\log_w\frac{1}{\varepsilon_0})$.

Query Time. By Lemma 10.2.1, we can find the desired cutting cell in time $O(\log n)$. Next, we query the data structure Ψ_{Δ_C} which by Theorem 10.3.1 uses $O(\log n + \varepsilon'^{-1}) = O(\log n + \frac{\varepsilon}{\log^2 t}) = O(\log n + \frac{1}{\varepsilon_0}\log^2\frac{1}{\varepsilon_0})$ query time. Scanning the groups and pruning the output of the data structure Ψ_{Δ_C} takes asymptotically smaller time and thus it can be absorbed in the above expression. Therefore, we obtain the following result.

Theorem 10.4.1. *Given an input consisting of an error parameter ε_0, and a set P of n points in $\mathbb{R}^3$ where each point $p\in P$ is associated with a weight w_p from a totally ordered universe, one can build a data structure that uses $O(n\log^2\frac{1}{\varepsilon_0}\log_w\frac{1}{\varepsilon_0})$ space such that given any query halfspace h, it can answer an AQS query with parameter ε_0 in time $O(\log n + \frac{1}{\varepsilon_0}\log^2\frac{1}{\varepsilon_0})$.*

For the case of 2D, we can just replace Ψ_{Δ_C} with the structure in Theorem 10.3.2, and we immediately get the following:

Theorem 10.4.2. *Given an input consisting of an error parameter ε_0, and a set P of n points in $\mathbb{R}^2$ where each point $p \in P$ is associated with a weight w_p from a totally ordered universe, one can build a data structure that uses $O(n \log^2 \frac{1}{\varepsilon_0} \log \log_w \frac{1}{\varepsilon_0})$ space such that given any query halfspace h, it can answer an AQS query with parameter ε_0 in time $O(\log n + \frac{1}{\varepsilon_0} \log^2 \frac{1}{\varepsilon_0})$.*

Dominance Approximate Quantile Summary Queries

Now we turn our attention to dominance ranges. We will show a structure similar to that for halfspace queries. The main difference is that we now use exact type-2 color counting as an auxiliary structure to estimate the rank of each group. This saves us roughly $\log^2 \frac{1}{\varepsilon_0}$ factors for both space and query time and so we can answer quantile queries in the optimal $O(\log n + \frac{1}{\varepsilon_0})$ time. To reduce the space to linear, we need more ideas. We first present a suboptimal but simpler structure to demonstrate our main idea. Then we modify this structure to get the desired optimal structure. We use shallow cuttings in the primal space.

A Suboptimal $O(n \log \log \frac{1}{\varepsilon_0})$ Space Solution

We first describe a data structure that solves the dominance AQS problem with $O(n \log \log \frac{1}{\varepsilon_0})$ space and the optimal $O(\log n + \frac{1}{\varepsilon_0})$ query time.

Rank-Preserving Approximation for Weighted Points. Let S be a weighted point set where every point has been assigned a weight from a totally ordered universe. Let $r_S(p)$ be the rank of a point p in the set S. Consider a geometric set system $(P, \mathscr{D})$, where P is a set of weighted points in $\mathbb{R}^3$ and $\mathscr{D}$ is a family of subsets of P induced by 3D dominance ranges. We mention a way to construct a sample A for P and a parameter ε such that

$$\left| \frac{r_{P \cap \mathscr{D}}(p)}{|P|} - \frac{r_{A \cap \mathscr{D}}(p)}{|A|} \right| \leq \varepsilon \qquad (10.4)$$

for any point $p \in P$ and any range $\mathscr{D} \in \mathscr{D}$. First note that taking an ε-approximation for P does **not** work since it does not take the weights of P into consideration. Our simple but important observation is that we can lift the points P into 4D by adding their corresponding weights as the fourth coordinate. Let us call this new point set P' and let $(P', \mathscr{D}')$ be the set system in 4D induced by 4D dominance ranges. Consider an ε-approximation A' for P' and let A be the projection of A' into the first three dimensions (i.e., by removing the weights again). A will be our sample for P and to distinguish it from an unweighted approximation, we call it *rank-preserving ε-approximation*. Indeed, for any point $p \in P$ with weight w_p and any $\mathscr{D} \in \mathscr{D}$, $r_{P \cap \mathscr{D}}(p)$ (resp. $r_{A \cap \mathscr{D}}(p)$) is equal to the number of points in P' (resp. A') contained in 4D dominance range $\mathscr{D} \times (-\infty, w_p)$. By the definition of ε-approximation, property (10.4) holds.

We now turn our attention to the AQS for 3D dominance queries.

The Data Structure and The Query Algorithm. Similar to the structure we presented for halfspace queries, we build $\frac{2^i}{\varepsilon_0}$-shallow cuttings for $i = 0, 1, \cdots, \log(\varepsilon_0 n)$. Let $\kappa = O(1)$ be the constant such that $O(\frac{1}{\varepsilon_0} \log^\kappa \frac{1}{\varepsilon_0})$ is the size of the ε_0-approximation for dominance ranges in 4D. Consider one k-shallow cutting $\mathscr{C}$. We consider two cases:

- If $k \le \frac{1}{\varepsilon_0} \log^\kappa \frac{1}{\varepsilon_0}$, for each cell Δ_C in the cutting $\mathscr{C}$, we collect the points in its conflict list $\mathscr{S}_{\Delta_C}$ and divide them into $t = \frac{1}{\varepsilon}$ groups $G_1, G_2, \cdots, G_t$ according to their weights (meaning, the weights in G_i are no larger than weights in group G_{i+1}) where $\varepsilon = \frac{\varepsilon_0}{c}$ for a big enough constant c as we did for halfspace queries.

- For $k > \frac{1}{\varepsilon_0} \log^\kappa \frac{1}{\varepsilon_0}$, we take a rank-preserving ε-approximation of $\mathscr{S}_{\Delta_C}$ first, and then divide the approximation into $t = \frac{1}{\varepsilon}$ groups, just like the above case. Again, for each group, we store the smallest weight among the points it contains.

We build the following structure for each cell Δ_C.

Let N be the number of points in all the t groups we generated for a cell Δ_C. We collect groups $G_{i\cdot\alpha+1}, G_{i\cdot\alpha+2}, \cdots, G_{(i+1)\cdot\alpha}$ into a cluster $\mathscr{C}_i$ for each $i = 0, 1, \cdots, t/\alpha - 1$ where $\alpha = (\log\log \frac{1}{\varepsilon_0})^3$. For each group j in cluster $\mathscr{C}_i$ for $j = 1, 2, \cdots, \alpha$, we color the points in the group with color j. Then we build the following type-2 color counting structure Ψ_i for $\mathscr{C}_i$. Let N_i be the total number of points in $\mathscr{C}_i$:

- First, we store three predecessor search data structures, one for each coordinate. This allows us to map the input coordinates as well as the query coordinates to rank space.

- Next, we build a grid of size $\sqrt[3]{N_i} \times \sqrt[3]{N_i} \times \sqrt[3]{N_i}$ such that each slice contains $\sqrt[3]{N_i^2}$ points. For each grid point, we store the points it dominates in a frequency vector using the compact representation.

- Finally, we recurse on each grid slab (i.e., three recursions, one for each dimension). The recursion stops when the number of points in the subproblem becomes smaller than $N_* = N_i^\eta$ for some small enough constant η.

- For these "leaf" subproblems, note that the total number of different answers to queries is bounded by $O(N_i^{3\eta})$. We build a lookup table which records the corresponding frequency vectors for these answers. Note that since at every step we do a rank space reduction, the look up can be simply done in $O(1)$ time, after reducing the coordinates of the query to rank space.

The query algorithm. Given a query q, we first locate the grid cell C containing q and this gives us three ranks. Using the ranks for x and y, we obtain an entry and using the rank of z, we find the corresponding word and the corresponding frequency vector stored in the lower corner of C. We get three more frequency vectors by recursing to three subproblems. We merge the three frequency vectors to generate the final answer. This completes the description of the structure we build for each family $\mathscr{C}_i$.

To answer a query q, we first find the first shallow cutting level above q and the corresponding cutting cell Δ_C. We then query the data structure described above to get the count the number of points dominated by q in each of the t groups. Then by maintaining a running counter, we scan through the t groups from left to right to construct the approximate ε_0-quantile summary.

Space Usage. For the space usage, note that there are N_i grid points in each recursive level and the recursive depth is $O(1)$. There are α colors and the frequency of a color is no more than N_i. So the total number of words needed to store frequency vectors is $O(N_i \frac{\alpha \log N_i}{w})$. When the problem size is below N_i^{η}, for each subproblem, we store a lookup table using $O(N_i^{3\eta} \frac{\alpha \log N_i}{w})$ words. So the total number of words used for the bottom level is $O(\frac{N_i}{N_i^{\eta}}) \cdot O(N_i^{3\eta} \frac{\alpha \log N_i}{w}) = O(N_i \frac{N_i^{2\eta} \alpha \log N_i}{w})$. Note that by our construction and $\varepsilon_0 \geq \frac{1}{n}$, $N = O(\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0})$, $\alpha = (\log\log \frac{1}{\varepsilon_0})^3 \leq (\log\log n)^3$ and $N_i = O(\alpha \frac{N}{1/\varepsilon_0}) = O(\alpha \log^{\kappa} \frac{1}{\varepsilon_0}) = O(\alpha \log^{\kappa} n)$. Since by assumption, $w = \Omega(\log n)$, by picking η in $N_* = N_i^{\eta}$ to be a small enough constant, the space usage for frequency vectors satisfy

$$f(N_i) = \begin{cases} 3\sqrt[3]{N_i} f(\sqrt[3]{N_i^2}) + O(\frac{N_i}{w^{1-o(1)}}), \text{for } N_i \geq N_* \\ O(\frac{N_i}{w^{1-\beta}}), \text{otherwise} \end{cases},$$

for some constant $0 < \beta < 1$, which solves to $O(\frac{N_i (\log N_i)^3}{w^{1-\beta}}) = O(\frac{N_i}{w^{1-\tau}})$ for some constant $0 < \tau < 1$. Since the recursive depth is $O(1)$, the space usage for all the predecessor searching structures is $O(N_i)$. Therefore the space usage of Ψ_i is $O(N)$. So the total space for each shallow cutting cell Δ_C is bounded by $\frac{N}{N_i} \cdot O(N_i) = O(N)$.

For $k_i \geq \frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$, $N = O(\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0})$. So the total space usage for them is bounded by

$$\sum_{i=\kappa \log\log \frac{1}{\varepsilon_0}}^{\varepsilon_0 n} O\left(\frac{n}{k_i}\right) \cdot O(N) = \sum_{i=\kappa \log\log \frac{1}{\varepsilon_0}}^{\varepsilon_0 n} O\left(\frac{n\varepsilon_0}{2^i}\right) \cdot O\left(\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}\right) = O(n).$$

For $k_i < \frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$, $N = k_i$ and so we have space bound

$$\sum_{i=0}^{\kappa \log\log \frac{1}{\varepsilon_0}} O\left(\frac{n}{k_i}\right) \cdot O(N) = \sum_{i=0}^{\kappa \log\log \frac{1}{\varepsilon_0}} O\left(\frac{n}{k_i}\right) \cdot O(k_i) = O\left(n \log\log \frac{1}{\varepsilon_0}\right).$$

This completes our space bound proof.

Query Time. For the query time, we first spend $O(\log n)$ time to find an appropriate shallow cutting level and the corresponding cell by the property of shallow cuttings.

Then we query Ψ_i for $i = 0, 1, \cdots, t/\alpha - 1$ to estimate the count for each group in the cell. For each Ψ_i, note that each predecessor searching takes $O(\log N_i)$ time. Also each frequency vector can fit in one word and so we can merge two frequency vectors in time $O(1)$. This gives us the following recurrence relation for the query time

$$g(N_i) = \begin{cases} 3g(\sqrt[3]{N_i^2}) + O(\log N_i), \text{for } N_i \geq N_* \\ O(\log N_i), \text{otherwise} \end{cases},$$

which solves to $O((\log N_i)^3) = O((\log\log\frac{1}{\varepsilon_0})^3) = O(\alpha)$. Since we need to query t/α such data structures to get the count for all groups, the total query time for count estimation is $O(t) = O(1/\varepsilon_0)$. Then we scan through the groups and report the approximate quantiles which takes again $O(1/\varepsilon_0)$ time. So the total query time is $O(\log n + \frac{1}{\varepsilon_0})$.

Correctness. Given a query q, let k be the actual number of points dominated by q. By the property of shallow cuttings, we find a cell Δ_C containing q in the shallow cutting level k_i above it such that $k \leq k_i \leq 2k$. When $k_i < \frac{1}{\varepsilon_0}\log^{\kappa}\frac{1}{\varepsilon_0}$, after we estimate the count in each group, since the estimation is exact and each group has size $\frac{|\mathscr{S}_{\Delta_C}|}{t} = \frac{\varepsilon_0|\mathscr{S}_{\Delta_C}|}{c}$, each quantile we output will have error at most $\frac{\varepsilon_0|\mathscr{S}_{\Delta_C}|}{c}$. For $k_i \geq \frac{1}{\varepsilon_0}\log^{\kappa}\frac{1}{\varepsilon_0}$, we introduce error $\varepsilon|\mathscr{S}_{\Delta_C}|$ in the ε-approximation, but since we use exact counting for each group, the total error will not increase as we add up ranks of groups. So the total error is at most $\frac{2\varepsilon_0|\mathscr{S}_{\Delta_C}|}{c}$. In both cases, the total error is at most $\varepsilon_0 k$ for a big enough c.

An Optimal Solution for 3D Dominance AQS

In this section, we modify the data structure in the previous section to reduce the space usage to linear. It can be seen from the space analysis that the bottleneck is shallow cuttings with $k_i \leq \frac{1}{\varepsilon_0}\log^{\kappa}\frac{1}{\varepsilon_0}$. For the structures built for these levels, the predecessor searching structures take linear space at each level which leads to a super linear space usage in total. To address this issue, we do a rank space reduction for points in the cells of these levels before constructing Ψ_i's so that we can use the integer register to spend sublinear space for the predecessor searching structures.

Rank Space Reduction Structure. We consider the cells in the $\frac{1}{\varepsilon_0}\log^{\kappa}\frac{1}{\varepsilon_0}$-shallow cutting. Let $A = \log^{\kappa+1}\frac{1}{\varepsilon_0}$. For the points in the conflict list $\mathscr{S}_{\Delta_C}$ of a shallow cutting cell $\Delta_C \in \mathscr{C}$, we build a grid of size $A \times A \times A$ such that each slice of the grid contains $O(1/(\varepsilon_0 \log\frac{1}{\varepsilon_0}))$ points. The coordinate of each grid point consists of the ranks of its three coordinates in the corresponding dimensions. For each of the $O(\frac{A}{\varepsilon_0 \log(1/\varepsilon_0)})$ points in $\mathscr{S}_{\Delta_C}$, we round it down to the closest grid point dominated by it. This reduces the coordinates of the points down to $O(\log\log\frac{1}{\varepsilon_0})$ bits and now we can apply the sub-optimal solution from the previous subsection which leads to an $O(n)$ space solution. To be more specific, we build the hierarchical shallow cuttings for

$k_i \leq \frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$ locally for hyperplanes in $\mathscr{S}_{\Delta_C}$ and apply the previous solution with a value $\varepsilon' = \varepsilon/c$ for a large enough constant c.

Query Algorithm and the query time. The query algorithm is similar to that for the previous suboptimal solution. The only difference is that when the query q is in a shallow cutting level smaller than $\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$, we use the rank space reduction structure to reduce q to the rank space. Let q' be the grid point obtained after reducing q to rank space. Observe that the set of points dominated by q can be written as the union of the points dominated by q' and the subset of points dominated by q in three grid slabs of A that contain q. We get an ε'-quantile for the former set using the data structure implemented on the grid points. The crucial observation is that there are $O(\varepsilon_0^{-1}/\log \frac{1}{\varepsilon_0})$ points in the slabs containing q and thus we can afford to build an approximate ε'-quantile summary of these points in $O(\frac{1}{\varepsilon_0})$ time. We can then merge these two quantiles and return the answer as the result. By setting c in the definition of ε' small enough, we make sure that the result is a valid ε_0 quantile summary. This also yields a query time (after locating the correct cell Δ_C in the shallow cutting) of $O(\frac{1}{\varepsilon_0})$.

Correctness. Since we build shallow cutting $k_i \leq \frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$ inside each cell in $\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$-shallow cutting, the transformed coordinates are consistent. As we described above, this introduces error to the counts Ψ_i's outputs, but since we correct the error explicitly afterwards, the counts we get are still exact. The remaining is the same as the suboptimal solution and so our structure finds ε_0-quantile properly.

Space Usage. For the rank space reduction structure, we need to store a predecessor searching structure for the query, which takes space linear in the number of slices which is $O(A)$. We build this structure for each cell in the A/ε_0-shallow cutting level and there are $O(n\varepsilon_0/A)$ cells in total, and so the space usage is $O(n\varepsilon_0)$. Building shallow cuttings inside each cell will only increase the space by a constant factor by the property of shallow cuttings.

For each Ψ_i, by our analysis in the suboptimal solution, the frequency vectors will take $O(\frac{N}{w^{1-\tau}})$ space. Now since the coordinates of the points and queries are integers of size at most A, it takes $O(\log A) = O(\log\log \frac{1}{\varepsilon_0})$ bits to encode a coordinate. Since the word size is $w = \Omega(\log n)$, we need only $O(\frac{N_i \log A}{w})$ space to build the predecessor searching structures for Ψ_i. In total, we spend $O(\frac{N}{w^{1-o(1)}})$ space for each shallow cutting level less than A/ε_0. So, the total space usage is $O(n)$. We conclusion this section by the following theorem.

Theorem 10.4.3. *Given an input consisting of a parameter $\varepsilon_0 > 0$, and a set P of n points in $\mathbb{R}^3$ where each point $p \in P$ is associated with a weight w_p from a totally ordered universe, one can build a data structure that uses the optimal $O(n)$ space such that given any dominance query γ, the data structure can answer an AQS query with parameter ε_0 in the optimal query time of $O(\log n + \frac{1}{\varepsilon_0})$.*

10.5 Open Problems

Our results bring many interesting open problems. First, for type-2 color counting problems, we showed a linear-sized structure for simplex queries. It is not clear if the query time can be reduced with more space. It is an intriguing open problem to figure out the correct space-time tradeoff for the problem. Note that our query time in Theorem 10.3.4 depends on the number of colors in total. It is unclear if the query time can be made output-sensitive. This seems difficult and unfortunately there seems to be no suitable lower bound techniques to settle the problem. Furthermore, since improving exact simplex range counting results is already very challenging, it makes sense to consider the approximate version of the problem with multiplicative errors.

Second, for heavy-hitter queries, there are two open problems. In our solution, the space usage is optimal with up to some extra polylogarithmic factor (in $\frac{1}{\varepsilon}$). An interesting challenging open problem is if the space usage can be made linear. On the other hand, our query time is not output-sensitive. Technically speaking, there can be less than $1/\varepsilon$ heavy hitters, and in this case, it would be interesting to see if $O(\log n + k)$ query time can be obtained for k output heavy hitters with (close to) linear space[2].

Third, for dominance queries, our data structure for halfspace ranges is suboptimal. The main reason is that we need a type-2 range counting solution as a subroutine. For halfspace ranges, our exact type-2 solution is too costly, and so we have to switch to an approximate version. This introduces some error and as a result, we need to use a smaller error parameter, which leads to extra polylogarithmic factors in both time and space. In comparison, we obtain an optimal solution for dominance quantile queries through exact type-2 counting. Currently, it seems quite challenging to improve the exact type-2 result for halfspace queries and some different ideas probably are needed to improve our results.

Finally, it is also interesting to investigate approximate quantile summaries, or heavy hitter summaries (or other data summaries or data sketches used in the streaming literature) for a broader category of geometric ranges. In this paper, our focus has been on very fast data structures, preferably those with optimal $O(\log n + \frac{1}{\varepsilon})$ query time, but we know such data structures do not exist for many important geometric ranges. For example, with linear space, simplex queries require $O(n^{(d-1)/d})$ time and there are some matching lower bounds. Nonetheless, it is an interesting open question whether approximate quantile or heavy hitter summary can be built for simplex queries in time $O(n^{(d-1)/d} + \frac{1}{\varepsilon})$ using linear or near-linear space; as we review in the introduction, the general approaches result in sub-optimal query times of $O(n^{(d-1)/d} \cdot \frac{1}{\varepsilon})$ or $O(n^{(d-1)/d} + \frac{1}{\varepsilon^2})$.

[2]We thank an anonymous referee for suggesting the "output-sensitive" version.

Appendices

10.A The Partition Theorem, Cuttings, and Exact Range Counting

The partition theorem is a standard tool for simplex range searching. It is originally proved by Matoušek [Mat92a] in early 1990s in the following form.

Theorem 10.A.1 (Matoušek's Partition Theorem [Mat92a])**.** *Let P be a point set of size n in $\mathbb{R}^d$ ($d \geq 2$), and s, $2 \leq s \leq n$, be an integer parameter and set $t = \frac{n}{2s}$. One can partition P into t subsets $P_1, \cdots, P_t$ such that for every $1 \leq i \leq s$, $s \leq |P_i| \leq 2s$, and P_i is enclosed in a simplex Δ_i such that any hyperplane in $\mathbb{R}^d$ has crossing number $O(t^{1-1/d})$, meaning, it intersects $O(t^{1-1/d})$ of the simplices.*

By directly applying this theorem, we can solve simplex range searching problems in near optimal time (up to $\log^{O(1)} n$ factors). The main reason is that the constant hidden in the crossing number will explode if we build a partition tree of super constant levels using this theorem. To get the optimal query time, Chan [Cha12] refined and improved the partition theorem and proved the following version.

Theorem 10.A.2 (Chan's Partition Refinement Theorem [Cha12])**.** *Let P be a point set of size n in $\mathbb{R}^d$ ($d \geq 2$), and H be a set of m hyperplanes in $\mathbb{R}^d$. Suppose we are given τ disjoint cells covering P, such that each cell contains at most $2n/\tau$ points of P and each hyperplane in H crosses at most ℓ cells. Then for any b, we can subdivide every cell into $O(b)$ disjoint subcells, for a total of at most $b\tau$ subscells, such that each subcell contains at most $2n/(b\tau)$ points of P, and each hyperplane in H crosses at most the following total number of cells:*

$$O((b\tau)^{1-1/d} + b^{1-1/(d-1)}\ell + b\log\tau\log m).$$

Note that the crossing number in Chan's version has some extra terms ($b^{1-1/(d-1)}l$ and $b\log\tau\log m$). These terms are dominated by the first term ($(b\tau)^{1-1/d}$) when $\tau = \log^{\Omega(1)} n$ but otherwise, the crossing number bound in Chan's bound can be worse than Matoušek's. For most applications of the partition theorem, we will build a partition tree of level $\Omega(\log\log n)$ and in that case, these extra terms can be safely ignored. Specifically, for the simplex range counting problem, by applying Chan's method, we obtain the following optimal result for linear space structures.

Theorem 10.A.3 (Small-Space Exact Simplex Range Counting [Mat93])**.** *Let P be a set of n points in $\mathbb{R}^d$. One can build a data structure of size $O(n)$ for simplex ranges such that given a query range γ, one can report $|\gamma \cap P|$ in $O(n^{(d-1)/d})$ time.*

Given two points $p, q \in \mathbb{R}^d$, we say p dominates q, denoted $q \geq p$, if all coordinates of p is no smaller than those of q. A dominance range γ is specified by a point p_γ and is defined to be $\{q \in \mathbb{R}^d : q \leq p_\gamma\}$. When the context is clear, we use p_γ and γ interchangeably. We will be interested in range counting for halfspace and dominance ranges in this paper. But since these two types of ranges are special cases of simplex ranges and so Theorem 10.A.3 applies.

If we want to answer queries faster in polylogarithmic time, one way is to spend more space. This is done by using point-line duality [dBCvKO08] and applying the technique of cuttings.

Given a simplex $\Delta_C \subset \mathbb{R}^d$ and a set H of hyperplanes, we denote the set of hyperplanes of H intersecting the interior of Δ_C by $\mathscr{S}_{\Delta_C}$, which is called the conflict list of Δ_C. An $(1/r)$-cutting for H is defined to be a collection $\mathscr{C}$ of pairwise interior disjoint d-dimensional closed simplicies that together cover $\mathbb{R}^d$ such that $|\mathscr{S}_{\Delta_C}| \leq n/r$ for all $\Delta_C \in \mathscr{C}$. The size of a cutting $\mathscr{C}$, denoted by $|\mathscr{C}|$, is defined to be the number of simplicies it has. The optimal size bound and construction algorithm of cuttings were developed after a series of work by the pioneers in the computational geometry community [Cla87, HW87, Mat92a, Aga91, CF90] and culminated in the work of Chazelle [Cha18].

Lemma 10.A.1 (Cutting Lemma [Cha18])**.** *Given a set H of n hyperplanes in $\mathbb{R}^d$, for any $r \geq 1$, there exists an $(1/r)$-cutting for H of size $O(r^d)$. The cutting as well as the conflict lists can be constructed in deterministically $O(nr^{d-1})$ time.*

Using Theorem [Cha18], we can obtain the following fast query time result for simplex range counting.

Theorem 10.A.4 (Fast-Query Exact Simplex Range Counting [Mat93])**.** *Let P be a set of n points in 2D (resp. 3D). One can build a data structure of size $O(n^d)$ for simplex ranges such that given a query range γ, one can report $|\gamma \cap P|$ in $O(\log n)$ time.*

10.B Shallow Cuttings and Approximate Range Counting

To define shallow cuttings more formally. We borrow the definition of shallow cuttings for general algebraic surfaces in [AHZ10]. Let $\mathscr{F}$ be a collection of continuous and totally defined algebraic functions $f : \mathbb{R}^d \to \mathbb{R}$ for constant dimension d and degree Δ. Each function $f \in \mathscr{F}$ defines a continuous surface in $\mathbb{R}^{d+1}$. Given any point $p = (p_1, p_2, \cdots, p_{d+1}) \in \mathbb{R}^{d+1}$, we say that f passes below p if $f(p_1, p_2, \cdots, p_d) \leq p_{d+1}$.

The collection of all surfaces in $\mathscr{F}$ partitions $\mathbb{R}^{d+1}$ into a subdivision consisting of disjoint cells (faces of dimensions $0, 1, \cdots, d+1$) that together cover the entire $\mathbb{R}^{d+1}$. We call this subdivision the arrangement of $\mathscr{F}$. Given any point $p \in \mathbb{R}^{d+1}$, its level is defined by the number of functions $f \in \mathscr{F}$ that passes below it. We define the $(\leq k)$-level of $\mathscr{F}$ to be the closure of all points in $\mathbb{R}^{d+1}$ with level at most k.

A shallow cutting $\mathscr{C}$ for the $(\leq k)$-level of $\mathscr{F}$ (or a k-shallow cutting for short) is a collection of disjoint cells that together cover the $(\leq k)$-level of $\mathscr{F}$ with the property that every cell $C \in \mathscr{C}$ in the cutting intersects a set $\mathscr{F}_C$ of $O(k)$ functions in $\mathscr{F}$. We call $\mathscr{F}_C$ the conflict list of C.

For a set $\mathscr{F}$, we say it is shallow cuttable if it has the following properties:

1. For each $k \in \mathbb{N}$, there exists a $(\leq k)$-shallow cutting for $\mathscr{F}$ and the number of cells the shallow cutting has is bounded by $O(|\mathscr{F}|/k)$.

2. The shallow cutting can be constructed in time $O(|\mathscr{F}|\log|\mathscr{F}|)$.

3. Given a hierarchy of α^i-shallow cuttings, there exists a linear sized structure such that given any query point q, we can find the α^i-shallow cutting $\mathscr{C}$ and the cell $C \in \mathscr{C}$ that contains q with the smallest i in time $O(\log|\mathscr{F}|)$.

For halfspace and dominance range searching problems, shallow cuttings are often used after translating the problem to the dual space, in which the roles of inputs and outputs "switch". It has been shown that both halfspace and dominance ranges are shallow cuttable in 2D and 3D [AHZ10]. More generally, we can count the number of points approximately in linear space and logarithmic time for shallow cuttable ranges.

Theorem 10.2.1 (Approximate Range Counting [AHZ10])**.** *Let P be a set of n points in $\mathbb{R}^3$. One can build a data structure of size $O(n)$ for halfspace or dominance ranges such that given a query range γ, one can report $|\gamma \cap P|$ in $O(\log n)$ time with error $\alpha|\gamma \cap P|$ for any constant $\alpha > 0$.*

We mention that the concept of shallow cuttings can also be defined in the primal space, i.e., for a set P of points. For example, for dominance ranges, given any point $p \in P$, we can define its level to be the number of points in P dominated by p. A (primal) k-shallow cutting for P is a set $\mathscr{C}$ of points in $\mathbb{R}^d$ satisfying that each point in P with level at most k is dominated by a point in $\mathscr{C}$ and each point in $\mathscr{C}$ dominates $O(k)$ points in P. It has been shown that such shallow cuttings can be constructed efficiently and we can find a shallow cutting level and a cell in a shallow cutting hierarchy efficiently as in the shallow cutting in the dual space [AT18].

10.C Type-2 Colored Simplex Range Counting

In this section, we study the "type-2" colored range counting problem for simplex queries in $\mathbb{R}^d$. In the problem, the input is a set of n points in $\mathbb{R}^d$, and each point is assigned one color from a set of F colors. We want to build a data structure such that given a query simplex q, we can report the number of points appearing in q for each color. In other words, we want to output a "frequency vector" for the colors. Formally, we define frequency vectors as follows.

Definition 10.C.1 (Frequency Vector). *Given a point set P with each point being assigned one of F colors, the frequency vector of P is defined to be $\mu(P) = (\mu_1, \dots, \mu_F)$ where μ_i is the frequency of the i-th color in P.*

Before giving the full details, we describe the overall idea of our data structure. Depending on different values of F/w, we will build different structures. When $F/w > \frac{n}{\log^4 n}$, we build a partition tree T using Matoušek's Theorem 10.A.1. Otherwise, we build a partition tree T using Chan's Theorem 10.A.2. For every vertex v of T, we denote the partitioning simplex corresponding to v by $v(\mathscr{R})$. We store the colored points at the leaves of T explicitly. Whereas, for every internal vertex v, we only store the frequency vector of the points in $v(\mathscr{R})$. The frequency vectors are represented in a compact way that we explain later.

The query algorithm is as follows. Given a query q, we query the corresponding partition tree T. During the process, we maintain a running frequency vector, i.e., the total of all the different colors seen so far (in a compact representation). We start from the root r of T. If q fully contains $v(\mathscr{R})$ then we add the frequency vector of $v(\mathscr{R})$ to the running frequency vector. If q does not contain $v(\mathscr{R})$ but has a non-empty intersection with it then we traverse its children. If q and $v(\mathscr{R})$ have empty intersection we do nothing. In case v is a leaf of T, we explicitly count the frequency of every color in $v(\mathscr{R})$ and update the frequency vector.

The total query time amounts to accessing the *compact representations* of the frequency vectors for the internal vertices and the *explicit counting* for the leaves which are traversed.

We now present the details. We start with the notion of compact representation.

Compact representations. Consider a frequency vector $\mu(\mu_1, \dots, \mu_F)$ and let $s = \sum_{i=1}^{F} \mu_i$. We would like to store μ using $O(F \log(2(s+2F)/F))$ bits while supporting the following two operations: one, an "add operation" to add the representations of two such vectors to obtain a representation of the addition, and two, an "extract" operation to retrieve the i-th frequency, μ_i, for a given i in constant time.

We first describe the encoding using a third character # besides 0 and 1: We encode the μ_i's in binary and place the character # between them. The representation can be easily made binary by simply re-encoding the characters 0, 1, and # in binary which increase the size of the encoding by a factor of 2. Let $L = 2\sum_{i=1}^{F}(\log(\mu_i + 2) + 1)$, i.e., the length of the representation in bits. By the arithmetic mean geometric mean inequality and $\sum_{i=1}^{F} \mu_i = s$,

$$L = 2\sum_{i=1}^{F} \log(\mu_i + 2) + 1 = 2F + 2\log\prod_{i=1}^{F}(\mu_i + 2) \le 2F + 2F\log\frac{s+2F}{F} = 2F\log\frac{2(s+2F)}{F}.$$

We pack the representation in $O(\lceil L/w \rceil)$ words and for each packed word, we store the number of the μ_i's it contains. Let c_j be the number of frequencies encoded in the j-th word. Observe that $c_j = O(w)$ and thus, we can store them in a data structure for prefix sums with constant query time. This concludes the description of the packed representation.

Note that two representations can be added in a straightforward way using non-standard word operations or using tabulations[3].

Now consider the extract operation with index i. We need to find the smallest index j such that the prefix sum up to j exceeds i. However, as each c_j is at most w, it can be encoded in unary and the extraction problem becomes equivalent to the `select` operation in bit vectors which can be performed in $O(1)$ time after building an index that takes linear time [RRS07]. To summarize, we have shown the following.

Lemma 10.C.1. *For any set of at most s points with each point being assigned one of F colors, we can encode its frequency vector with a compact representation of $O(1+\frac{F}{w}\log\frac{2(s+2F)}{F})$ words, where w is the word size. Furthermore, we can add two compact representations in time $O(1+\frac{F}{w}\log\frac{2(s+2F)}{F})$ and extract the frequency of the i-th color in time $O(1)$ for $1 \le i \le F$.*

The Data Structure. Now we describe the details of our data structure. As mentioned before, we build either partition trees based on an earlier version of Matoušek (Theorem 10.A.1) and a more recent (refined and simplier) version of Chan (Theorem 10.A.2). The main reason why we need both structures is although Chan's version is superior in many aspects, it is not able to handle the case when the partition tree is very shallow (which can happen in our application). Fortunately, Matoušek's early version is able to give the optimal bound in this case. So we will use different partition trees for different cases.

Theorem 10.3.4. *Given an integer parameter F, a set P of n points in $\mathbb{R}^d$ where each point is assigned a color from the set $[F]$, one can build a linear-sized data structure, such that given a query simplex q, it can output the number of times each color appears in $P \cap q$ in total time* $\max\{O(n^{(d-1)/d}), O(n^{(d-1)/d}F^{1/d}/w^{\alpha})\}$, *for some appropriate constant α and word size w.*

Proof. We build two partition trees based on the value of $\frac{F}{w}$. The base case for both cases is the same: When the size of the subproblem reaches below F/w we stop and examine all points by brute force.

When $\frac{F}{w} \ge \frac{n}{\log^4 n}$, we build a partition tree T of Matoušek's version (Theorem 10.A.1) with fanout $t = w^{\delta}$ for some small enough $\delta > 0$. For each node in T, we store a frequency vector of the points rooted at that node in the compact representation. The query time $Q(n)$ satisfies the following recurrence relation

$$Q(n) = O(t^{\frac{d-1}{d}})Q\left(\frac{n}{t}\right) + O(t)\cdot O\left(1+\frac{F}{w}\log\frac{2(n/t+2F)}{F}\right).$$

After running for k steps we have,

$$Q(n) \le c_0^k t^{k\frac{d-1}{d}} Q\left(\frac{n}{t^k}\right) + \sum_{i=1}^{k} c_1 t^{\frac{1}{d}} t^{i(\frac{d-1}{d})}\cdot\left(1+\frac{F}{w}\log\frac{2(n/t^i+2F)}{F}\right), \tag{10.5}$$

[3]E.g., pack into words of size $\frac{\log n}{2}$ and then build a table of size n which supports the add operation in constant time.

for some constants c_0, c_1.

Note that when $\frac{n}{t^i} \geq 2F$, $\log \frac{2(n/t^i+2F)}{F} \leq \log \frac{4n/t^i}{F}$ and so the summation in Inequality 10.5 is upper bounded by

$$\sum_{i=1}^{k} c_1 t^{\frac{1}{d}} t^{i(\frac{d-1}{d})} \cdot \left(1 + \frac{F}{w} \log \frac{4n}{Ft^i}\right) = c_1 \sum_{i=1}^{k} (t^{\frac{1}{d}} t^{i(\frac{d-1}{d})}) + t^{\frac{1}{d}} t^{i(\frac{d-1}{d})} \cdot \left(\frac{F}{w} \log \frac{4n}{Ft^i}\right).$$

Let $f(i) = t^{i(\frac{d-1}{d})} \log \frac{4n}{Ft^i}$, and we have

$$\frac{f(i+1)}{f(i)} = t^{\frac{d-1}{d}} \left(1 + \frac{\log \frac{1}{t}}{\log \frac{4n}{Ft^i}}\right) = t^{\frac{d-1}{d}} \left(1 + \frac{\log t}{\log \frac{Ft^i}{4n}}\right) \geq t^{\frac{d-1}{d}} \left(1 + \frac{\log t}{\log \frac{w}{4}}\right) \geq w^{\delta(\frac{d-1}{d})} = \omega(1),$$

where the first inequality follows from $\frac{n}{t^i} \geq \frac{F}{w}$ and the last inequality follows from $t = w^{\delta}$ for some constant δ. When $\frac{n}{t^i} < 2F$, the summation in Inequality 10.5 is upper bounded by

$$\sum_{i=1}^{k} c_1 t^{\frac{1}{d}} t^{i(\frac{d-1}{d})} \cdot \left(1 + \frac{F}{w} \log \frac{2(2F+2F)}{F}\right) \leq \sum_{i=1}^{k} c_1 t^{\frac{1}{d}} t^{i(\frac{d-1}{d})} \cdot \left(1 + \frac{F}{w} \log 8\right).$$

In both cases, the summation is a geometric series and it is dominated by the last term, i.e., when i achieves the maximum value.

Since $\frac{F}{w} \geq \frac{n}{\log^4 n}$, we observe that after constant levels of applying Matoušek's scheme we will reach the base case because we stop as soon as the subproblem size goes below $\frac{F}{w}$. So we have $\frac{F}{w^{1+\delta}} \leq \frac{n}{t^k} \leq \frac{F}{w}$ which means $t^k \leq \frac{nw^{1+\delta}}{F}$ and $t^k \geq \frac{nw}{F}$. Plugging in these inequalities in Equation 10.5, and using that $Q(n/t^k) = O(\frac{F}{w})$, we obtain the following bound

$$Q(n) = O\left(\frac{n^{\frac{d-1}{d}} F^{\frac{1}{d}}}{w^{\frac{1}{d}-\delta}}\right). \tag{10.6}$$

When $\frac{F}{w} < \frac{n}{\log^4 n}$, we use Chan's version (Theorem 10.A.2) to build the partition tree. The observation here is that we can partition the point set into at least $\tau = \log^4 n$ subsets. Now the crossing number is at least $\log^2 n$ and thus the main term $O((b\tau)^{1-1/d})$ dominates.

We build a partition tree based on Chan's method (Theorem 10.A.2) with parameter $b = \Theta(1)$. Again, we attach the corresponding frequency vectors to internal nodes. The query time $Q(n)$ in this case is bounded by

$$Q(n) = O(n^{1-1/d}) + O\left(\sum_{i=0}^{\beta-1} bl(b^i) \cdot \left(1 + \frac{F}{w} \log \frac{2(\frac{n}{b^i} + 2F)}{F}\right)\right),$$

where the first term is the standard cost of traversing the partition tree in Chan's version and the second term is the cost of examining all the disjoint cells fully contained in q, and the parameters $\beta = \log_b \frac{n}{F/w} = \Omega(\log\log n)$ and $l(u)$ is the crossing number of u

cells. Note that by a similar analysis as we did before, the summation is dominated by its last term, i.e.,

$$\begin{aligned}\sum_{i=0}^{\beta-1} bl(b^i)\cdot\left(1+\frac{F}{w}\log\frac{2(\frac{n}{b^i}+2F)}{F}\right) &= O\left(b\left(\frac{nw}{bF}\right)^{\frac{d-1}{d}}\cdot\left(1+\frac{F}{w}\log\frac{2(\frac{F}{w}+2F)}{F}\right)\right)\\ &= O\left(\frac{n^{\frac{d-1}{d}}F^{\frac{1}{d}}}{w^{\frac{1}{d}}}\right).\end{aligned}$$

So $Q(n) = O\left(\frac{n^{\frac{d-1}{d}}F^{\frac{1}{d}}}{w^{\frac{1}{d}}}\right)$ in this case. Note that the bound in Equation 10.6 is slightly larger and thus our query time is bounded by Equation 10.6.

□

10.D Approximate Heavy Hitter Summary Queries

In this section, we prove the following approximate heavy hitter summary query results for halfspace and dominance ranges.

Theorem 10.3.1. *For $d = 3$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

Theorem 10.3.2. *For $d = 2$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

Theorem 10.3.3. *For $d = 2,3$, the approximate dominance heavy hitter summary queries can be answered using the optimal $O(n)$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

We start with the description of a base data structure for more general simplex ranges which will be used as a building block for our final AHHS solution.

Base Solution for AHHS Queries

Our base data structure is for the following problem:

Problem 10.3.1. *[Coarse-Grained AHHS Queries] Let P be a set of points in $\mathbb{R}^d$, each associated with a color. The problem is to store P in a structure such that given a query range q, one can estimate the frequencies of colors in $q\cap P$ with an additive error up to $\varepsilon|P|$ efficiently for some parameter $0 < \varepsilon < 1$.*

Note that this is a coarser problem compared to the original AHH query Problem 10.1.2 since the error is fixed to be $\varepsilon|P|$ and is independent of the query range. We prove the following theorem for this base problem:

Theorem 10.3.5. *For $d \geq 3$, Problem 10.3.1 for simplex queries (the intersection of $d+1$ halfspaces) can be solved with $O(X)$ space for $X = \min\{|P|, \varepsilon^{-\frac{2d}{d+1}}\}$ and a query time of*

$$O\left(\frac{|P|^{1-\frac{2}{d-1}}}{w^{\alpha}\varepsilon^{\frac{2}{d-1}}}\right) + O\left(X^{\frac{d-1}{d}}\right)$$

where w is the word-size of the machine and α is some positive constant.

We will handle colors with different frequencies differently. To simplify the exposition, we make the following definition:

Definition 10.D.1. *Let P be a set of points with colors. Let P_i be the set of points with color i. We say a color i is **big** if $|P_i| \geq (\varepsilon|P|)^{\frac{2d}{d-1}}$; otherwise we say it is **small**. For a small color i, we say it is **δ-small** if $\delta(\varepsilon|P|)^{\frac{2d}{d-1}} \leq |P_i| \leq 2\delta(\varepsilon|P|)^{\frac{2d}{d-1}}$.*

Note that a simple calculation shows that if $|P| > \varepsilon^{-\frac{2d}{d+1}}$, then there are no big colors since the bound in the definition of a big color exceeds $|P|$.

The data structure. The idea is to count big colors *exactly* and count small colors *approximately* through ε-approximations. We collect the points with big colors in a set P_{Big} and build a data structure Ψ_{Big} on P_{Big} using Theorem 10.3.4. For each small color i, we first compute an ε_i-approximation P_i' for P_i where $\varepsilon_i = \varepsilon|P|/|P_i|$. For a given value $\delta \leq 1$, let P_δ be the set of points whose colors are δ-small and let P_δ' be the union of the approximations built on the δ-small colors. We build P_δ' for every $\delta = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \cdots, \frac{1}{2}(\varepsilon|P|)^{-\frac{d+1}{d-1}}$. We can bound the number of colors that are δ-small to be $F_\delta = \Theta(|P_\delta|/(\delta(\varepsilon|P|)^{\frac{2d}{d-1}}))$. We also store P_δ' in a data structure Ψ_δ using Theorem 10.3.4.

The query algorithm. The query is answered as follows. Given a query simplex range q, we query the data structures Ψ_{Big} as well as all the data structure Ψ_δ, for every $\delta = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \cdots, \frac{1}{2}(\varepsilon|P|)^{-\frac{d+1}{d-1}}$. We examine each output frequency vector of colors and output a color if its frequency is more than $\varepsilon|P|$.

The correctness is trivial since we are either counting the colors exactly or we are using an ε_i-approximation. In the latter case, the error is at most $\varepsilon_i|P_i| = \varepsilon|P|$. Since we can ignore colors of frequencies no more than $\varepsilon|P|$, we only need to consider δ-small colors for $\delta \geq \frac{1}{2}(\varepsilon|P|)^{-\frac{d+1}{d-1}}$.

Space bound. Let $Y = |P_{\text{Big}} \cup \bigcup_{i \in small} P_i'|$. The space usage of our data structure is clearly bounded by $O(Y)$. We now show that $Y = O(X)$ which would prove our claim on the space bound.

First we bound the size of the ε_i-approximation for each small color i. Consider a δ-small color i and the set P_i. By construction, we are using an $\varepsilon_i = \varepsilon|P|/|P_i|$

approximation which has size

$$O\left(\left(\frac{1}{\varepsilon_i}\right)^{\frac{2d}{d+1}}\right) = O\left(\left(\frac{|P_i|}{\varepsilon|P|}\right)^{\frac{2d}{d+1}}\right) = O\left(\left(\frac{2\delta(\varepsilon|P|)^{\frac{2d}{d-1}}}{\varepsilon|P|}\right)^{\frac{2d}{d+1}}\right)$$
$$= O\left(\left(\delta(\varepsilon|P|)^{\frac{d+1}{d-1}}\right)^{\frac{2d}{d+1}}\right) = O\left(\delta^{\frac{d-1}{d+1}}|P_i|\right) = O(|P_i|), \qquad (10.7)$$

where the second last equality follows from $\delta(\varepsilon|P|)^{\frac{2d}{d-1}} \le |P_i|$. Thus, each ε_i-approximation has size at most $O(|P_i|)$ since $\delta \le 1/2$, and so $Y \le |P|$.

When $|P| > \varepsilon^{-\frac{2d}{d+1}}$, as we argued, there is no big color and all the colors are small. In this case, we can bound

$$Y \le \left|\bigcup_{i\in small} P_i'\right| = \sum_i O\left(\left(\frac{1}{\varepsilon_i}\right)^{\frac{2d}{d+1}}\right) = O\left(\sum_i \left(\frac{|P_i|}{\varepsilon|P|}\right)^{\frac{2d}{d+1}}\right) = O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{2d}{d+1}}\right), \qquad (10.8)$$

where the last equality follows from the convexity of the function $x^{\frac{2d}{d+1}}$ and thus the sum is maximized when all the values of $|P_i|$ are 0 and one of them equals $|P|$. We have thus shown that $Y \le X = \min\{|P|, \varepsilon^{-\frac{2d}{d+1}}\}$ and so $O(X)$ is the space bound.

Query time bound. It remains to analyze the query time. For P_{Big}, note that the number of colors is no more than $F_{Big} = |P|/(\varepsilon|P|)^{\frac{2d}{d-1}}$. By Theorem 10.3.4, the data structure we built for P_{Big} will have query time

$$O(|P_{Big}|^{(d-1)/d} + |P_{Big}|^{(d-1)/d} F_{Big}^{1/d}/w^{\alpha}) = O\left(|P|^{(d-1)/d}\right) + O\left(\frac{|P|^{\frac{d-1}{d}}|P|^{\frac{1}{d}}}{w^{\alpha}(\varepsilon|P|)^{\frac{2}{d-1}}}\right)$$
$$= O\left(X^{(d-1)/d}\right) + O\left(\frac{|P|^{1-\frac{2}{d-1}}}{w^{\alpha}\varepsilon^{\frac{2}{d-1}}}\right),$$

for some constant α, where the first equality follows from $P_{Big} \subset P$ and the last equality follows from $|P| \le \varepsilon^{-\frac{2d}{d+1}}$ for big colors to exist in which case $X = |P|$ by definition. We now bound the query time for small colors. By Equation 10.7, the total size of the ε-approximations stored for the δ-small colors is $n_\delta = |P_\delta'| = O(\delta^{\frac{d-1}{d+1}}|P_\delta|)$. By Theorem 10.3.4, the query time of the data structure built on P_δ' is

$$O\left(\max\left\{n_\delta^{(d-1)/d}, n_\delta^{(d-1)/d} F_\delta^{1/d}/w^{\alpha}\right\}\right) = O(n_\delta^{(d-1)/d}) + O(n_\delta^{(d-1)/d} F_\delta^{1/d}/w^{\alpha}). \qquad (10.9)$$

The total query time for small colors is the sum over all δ of the expression in Equation 10.9. We can bound each of the two terms separately.

First observe that

$$\sum_{\delta}\left(\delta^{\frac{d-1}{d+1}}\left(\frac{|P_\delta|}{|P|}\right)\right)^{(d-1)/d} \le \sum_{\delta}\delta^{\frac{(d-1)^2}{d(d+1)}} = O(1),$$

since the summation is over geometrically decreasing values of δ and $|P_\delta| \le |P|$ by definition. When $X = \min\{|P|, \varepsilon^{-\frac{2d}{d+1}}\} = |P|$, this implies

$$\sum_{\delta} n_\delta^{\frac{d-1}{d}} = \sum_{\delta}(\delta^{\frac{d-1}{d+1}}|P_\delta|)^{(d-1)/d} \le |P|^{\frac{d-1}{d}} \le X^{\frac{d-1}{d}}.$$

When $X = \min\{|P|, \varepsilon^{-\frac{2d}{d+1}}\} = \varepsilon^{-\frac{2d}{d+1}}$, by definition,

$$\begin{aligned}\sum_{\delta} n_\delta^{\frac{d-1}{d}} &= \sum_{\delta}\left(\sum_{i:\delta\text{-small}}\left(\frac{1}{\varepsilon_i}\right)^{\frac{2d}{d+1}}\right)^{\frac{d-1}{d}} = \sum_{\delta}\left(\sum_{i:\delta\text{-small}}\left(\frac{|P_i|}{\varepsilon|P|}\right)^{\frac{2d}{d+1}}\right)^{\frac{d-1}{d}} \le \sum_{\delta}\left(\frac{|P_\delta|}{\varepsilon|P|}\right)^{\frac{2(d-1)}{d+1}}\\ &\le \left(\frac{|P|}{\varepsilon|P|}\right)^{\frac{2(d-1)}{d+1}} = \left(\left(\frac{1}{\varepsilon}\right)^{\frac{2d}{d+1}}\right)^{\frac{d-1}{d}} = X^{\frac{d-1}{d}},\end{aligned}$$

where the first inequality follows from $\frac{2d}{d+1} \ge 1$ for $d \ge 1$ and the second inequality follows from $\frac{2(d-1)}{d+1} \ge 1$ when $d \ge 3$.

For the second term, since $F_\delta = O(|P_\delta|/(\delta(\varepsilon|P|)^{\frac{2d}{d-1}}))$,

$$\begin{aligned}\sum_{\delta} O\left(\frac{n_\delta^{\frac{d-1}{d}} F_\delta^{\frac{1}{d}}}{w^\alpha}\right) &= \sum_{\delta} O\left(\frac{(\delta^{\frac{d-1}{d+1}}|P_\delta|)^{\frac{d-1}{d}}}{w^\alpha}\cdot\left(\frac{|P_\delta|}{\delta(\varepsilon|P|)^{\frac{2d}{d-1}}}\right)^{\frac{1}{d}}\right) = \sum_{\delta} O\left(\frac{\delta^{\frac{d-3}{d+1}}|P_\delta|}{w^\alpha(\varepsilon|P|)^{\frac{2}{d-1}}}\right)\\ &= O\left(\frac{(\sum_\delta \delta^{\frac{d-3}{d+1}})|P|}{w^\alpha(\varepsilon|P|)^{\frac{2}{d-1}}}\right) \le O\left(\frac{|P|^{1-\frac{2}{d-1}}}{w^\alpha\varepsilon^{\frac{2}{d-1}}}\right),\end{aligned}$$

where the third equality follows from the observation that the sum of all n_δ is at most $|P|$ and the last equality follows from $d \ge 3$ and $\delta \le \frac{1}{2}$ forms a decreasing geometric series.

3D AHHS Queries

Now we proceed to show how to answer 3D heavy hitter queries for halfspace and dominance ranges. We show the following two results.

Theorem 10.3.1. *For $d = 3$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

Theorem 10.3.3. *For $d = 2, 3$, the approximate dominance heavy hitter summary queries can be answered using the optimal $O(n)$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

The Data Structure for 3D AHHS Queries

Let h be the query halfspace (or dominance range) and let k be the number of points in h (i.e., $k = |h \cap P|$). We consider the dual space. Let H be the set of hyperplanes dual to P and let q be the dual of h. Using Theorem 10.2.1, we can find a $(1+\alpha)$factor approximation, k^*, of k in $O(\log n)$ time and using linear space. In our data structure, we will build a series of shallow cuttings, using parameters β, λ. Depending on whether we are dealing with halfspace or dominance ranges, the values of these parameters will be different. Nonetheless, since the value of k^* is known, we can choose the correct level to query. To give more details, we consider the following cuttings.

- Base cutting: We build a *base shallow cutting* for the ε_0^{-1} level.
- Lower level cuttings: These are built for levels between ε_0^{-1} and $\varepsilon_0^{-1}w^{\beta\lambda}$, where $w = \Theta(\log n)$ is the word-size, and β and λ are parameters to be set later. To be specific, we build $\varepsilon_0^{-1}w^{\beta i}$-shallow cuttings for $i = 1, \cdots, \lambda$.
- Higher level cuttings: We build $\varepsilon_0^{-1}w^{\beta\lambda}2^i$-shallow cuttings for $i = 1, \cdots, \log\left(n\varepsilon_0/w^{\beta\lambda}\right)$.

The data structure stores different auxiliary data structures depending on the shallow cutting. As a result, since the data structure knows k^*, it can choose the correct level to query. For instance, if $k^* < \varepsilon_0^{-1}$, then we use the query is answered using the auxiliary data structures built for the base cutting. Similarly, when $\varepsilon_0^{-1} \le k^* \le \varepsilon_0^{-1}w^{\beta\lambda}$, the data structure uses lower level cuttings, and otherwise it uses higher level cuttings.

We describe each of them in turn. We start with the higher level cuttings.

Higher level cuttings. The data structure for these cuttings is simple. For each cell Δ_C in a higher level cutting, we build a $c_1\varepsilon_0$-approximation E_{Δ_C} for its conflict list $\mathscr{S}_{\Delta_C}$ and store it in the base structure with parameter $c_1\varepsilon_0$.

The query algorithm. Recall that the query probes the higher level cuttings when k^* exceeds $\varepsilon_0^{-1}w^{\beta\lambda}$. In this case, it can find the cell $\Delta_C \in \mathscr{C}_i$ such that $|\mathscr{S}_{\Delta_C}| = O(k^*)$. We query the base data structure built on $\mathscr{S}_{\Delta_C}$ and return the result as the AHHS. The query time here is $O(\varepsilon_0^{-1})$ which follows directly from the base structure.

Correctness. Consider a color c that has frequency $j\varepsilon_0 k$ among the hyperplanes that pass below q, for $j \ge 1$. Since E_{Δ_C} is a $c_1\varepsilon_0$-approximation of $\mathscr{S}_{\Delta_C}$, it follows that c appears $j\varepsilon_0|E_{\Delta_C}| \pm c_1\varepsilon_0|E_{\Delta_C}|$ times among the subset of E_{Δ_C} that passes below q. If c_1 is chosen to be a small enough constant (e.g., $c_1 < 0.5$), c appears frequent enough in E that the base structure would report c as well as its frequency with error at most $c_1\varepsilon_0|\mathscr{S}_{\Delta_C}|$. This in turn allows us to approximate the frequency of c within hyperplanes below q up to an additive error of $2c_1\varepsilon_0 k$.

Space usage. Here, we need to distinguish between halfspaces and dominance ranges. In the case of halfspaces, the size of E_{Δ_C} is bounded by $O(\varepsilon_0^{-3/2})$ whereas for dominance ranges this is bounded by $O(\varepsilon_0^{-1}\log^4\frac{1}{\varepsilon_0})$ by Theorem 10.2.2. Consequently, for halfspaces we have the following situation: for a cell $\Delta_C \in \mathscr{C}_i$ where $\mathscr{C}_i$ is a $\varepsilon_0^{-1}w^{\beta\lambda}2^i$-shallow cuttings, according to Theorem 10.3.5, the space of the base structure built for E_{Δ_C} is bounded by $O(\varepsilon_0^{-3/2})$ regardless and thus the total space consumption is bounded by

$$\sum_{i=1}^{\infty} O\left(\varepsilon_0^{-3/2}\frac{n}{\varepsilon_0^{-1}w^{\beta\lambda}2^i}\right) = O(n)$$

which follows by picking λ such that $w^{\beta\lambda} = \varepsilon_0^{-1/2}$ for a small enough value β.

For dominance ranges, it is sufficient to pick λ such that $\log^4\frac{1}{\varepsilon_0} \le w^{\beta\lambda} \le \varepsilon_0^{-1/2}$. We will need to pick β to be a small enough constant. However, this implies that λ can also be chosen to be a constant for dominance cases. The total space consumption here is bounded by

$$\sum_{i=1}^{\infty} O\left(\varepsilon_0^{-1}\log^4\frac{1}{\varepsilon_0}\cdot\frac{n}{\varepsilon_0^{-1}\log^4\frac{1}{\varepsilon_0}\cdot 2^i}\right) = O(n).$$

Lower level cuttings. For each cell Δ_C in a lower level cutting, we store the list of *frequent* colors that appear at least $\frac{c_1\varepsilon_0}{w^\beta}|\mathscr{S}_{\Delta_C}|$ times in the conflict list $\mathscr{S}_{\Delta_C}$ of Δ_C. So for each cutting cell, we store $\frac{w^\beta}{c_1\varepsilon_0}$ frequent colors in the base structure in Theorem 10.3.5, and so the total space needed for all lower cuttings is

$$\sum_{i=1}^{\lambda} O\left(\frac{n}{w^{\beta i}/\varepsilon_0}\cdot\frac{w^\beta}{c_1\varepsilon_0}\right) = O(n).$$

This also enables us to *re-number* the colors: consider a Δ_C in the lower level cuttings. We re-number the candidate colors in Δ_C from 1 up to $w^\beta\varepsilon_0^{-1}$ and store them in a dictionary such that given any (global) color C, we can fetch its index in our new re-numbering in $O(1)$ time. Re-numbering is crucial for using the base data structure as it returns a compact representation. Note that the set of frequent colors is a super set of the actual heavy hitters. Indeed, the base structures we built for lower levels can output a frequency vector of $\frac{w^\beta}{c_1\varepsilon_0}$ colors (in a compact representation) and thus we cannot afford to check all these colors.

We use an auxiliary structure to generate a list of $O(\frac{1}{\varepsilon_0})$ *candidate* colors for each query and as we will show, we only need to check these candidate colors. We call it the *testing* structure. This is also done via shallow cuttings: We build shallow cuttings for level $\varepsilon_0^{-1}2^i$ for $i = 1,2,\cdots,\log\left(w^{\beta\lambda}\right)$. We call these *testing* cuttings. For each cell Δ_C in the testing cuttings, we collect a list of colors that appear at least $c_1\varepsilon_0|\mathscr{S}_{\Delta_C}|$ times in the conflict list $\mathscr{S}_{\Delta_C}$ of Δ_C. We call these colors *candidate* colors and clearly, for each cell Δ_C, we store at most $\frac{1}{c_1\varepsilon_0}$ candidate colors. The total space, over all testing cuttings, needed to store the candidate colors is at most

$$\sum_{i=1}^{\infty} O\left(\frac{n}{\varepsilon_0^{-1}2^i}\cdot\frac{1}{c_1\varepsilon_0}\right) = O(n).$$

The query algorithm. Recall that the query probes the lower level cuttings when k^* is between ε_0^{-1} and $\varepsilon_0^{-1}w^{\beta\lambda}$. In this case, we find the smallest index i such that $\varepsilon_0^{-1}w^{\beta i}$ exceeds k^*. We find the cell Δ_C in $\mathscr{C}_i$ that contains the query. It thus follows that $|\mathscr{S}_{\Delta_C}| = O(w^\beta k)$. We query the base data structure implemented on $\mathscr{S}_{\Delta_C}$ with q. However, since we have built the base structure with error parameter ε_0/w^β, the base data structure may return a list of $O(\varepsilon_0^{-1}w^\beta)$ colors, potentially in packed representation. We pick β small enough such that the result of the base data structure fits in ε_0^{-1} words. On the other hand, since $|\mathscr{S}_{\Delta_C}| = O(\varepsilon_0^{-1}w^{\beta\lambda}) = O(\varepsilon_0^{3/2})$, the query time of the base data structure is $O(\varepsilon_0^{-1})$ by Theorem 10.3.5. In addition, we query the testing structure to obtain a list of $O(\varepsilon_0^{-1})$ candidate colors. For each candidate color, we can use the stored dictionary to find its frequency in the packed representation, if it exists, we report it using the extract operation. This concludes the query algorithm.

Correctness. The base data structure finds all the colors that appear at least $\frac{c_1\varepsilon_0|\mathscr{S}_{\Delta_C}|}{w^\beta} = \frac{O(c_1\varepsilon_0 k w^\beta)}{w^\beta} \leq \varepsilon_0 k$ times below q, if c_1 is chosen to be a small enough constant. Thus, any color that appears $\varepsilon_0 k$ times is reported correctly by the base data structure.

Space analysis. By Theorem 10.3.5, each level in a lower level shallow cutting $\mathscr{C}_i$ will consume $O(n)$ space. There are λ lower level shallow cuttings and thus the total space used by lower level cuttings is $O(n\lambda)$.

For halfspace ranges, recall that we needed to pick λ such that $w^{\beta\lambda} = \varepsilon_0^{-1/2}$. As β is chosen to be a small value, it follows that for halfspaces we can choose $\lambda = O(\log_w \frac{1}{\varepsilon_0})$, leading to an $O(n\log_w \frac{1}{\varepsilon_0})$ space bound.

For dominance ranges, it is sufficient to pick λ such that $w^{\beta\lambda} \geq \log^4 \frac{1}{\varepsilon_0}$. In this case, as we have shown before, by picking β to be a small enough constant, it suffices to pick λ to be a large enough constant, leading to an $O(n)$ space bound.

The base cutting. The base cutting, $\mathscr{C}_0$, is a ε_0^{-1}-shallow cutting. We store the conflict list of the cells in $\mathscr{C}_0$ explicitly. By Lemma 10.2.1, in this case, we find $\mathscr{C}_0$ and the cell $\Delta_C \in \mathscr{C}_0$ that contains q in time $O(\log n)$. Here, we can explicitly access $\mathscr{S}_{\Delta_C}$ and answer the AHHS query directly, in time $O(\log n + \varepsilon_0^{-1})$ with no error.

Putting It Together

Over all three different shallow cutting levels, the total space complexity was $O(n\lambda)$. For dominance ranges, we saw that we can afford to pick λ to be a constant. This yields a data structure with $O(n)$ space. For halfspaces, we picked $\lambda = O(\log_w \frac{1}{\varepsilon_0})$ and thus we get a data structure with $O(n \log_w \frac{1}{\varepsilon_0})$ space.

2D AHHS Queries

In the plane, we can actually improve the space complexity even further. However, this requires modifying the base structure. The main idea here is that the size of ε-approximation in 2D is small enough that we can try more aggressive approximations.

Base Solution for 2D AHHS Queries

This subsection is devoted to the proof of the specialized base structure for 2D.

Theorem 10.D.1. *Consider a colored point set P in $\mathbb{R}^2$ and let ε be parameter. Let $B = \varepsilon|P|$. We can store P in a data structure of linear size such that given a set q_c of $F = \frac{1}{\varepsilon B^{1/3}}$ colors, one can estimate the frequency of the colors in q_c up to additive error of $\varepsilon|P|$ in $O(F)$ time.*

The main idea and a summary. The main observation is that in 2D, given a point set P and a value ε, there exists ε-approximations of size $O(\varepsilon^{-4/3})$ and thus if we store them in a data structure for halfspace range counting queries, the query time will be reduced to $O(\varepsilon^{-2/3})$ i.e., smaller than ε^{-1}. Intuitively, this means that the difficult case of the problem is when there a lot of different colors with low frequency. However, proving the base theorem is still quite non-trivial since we are aiming for very fast query times. Our main observation is that when there are few points, the size of the ε-approximations are small enough that we can afford to spend slightly higher space and lower the query time in return. By picking the parameters carefully, we arrive at the claimed lemma.

Dealing with "frequent" colors. We define a color to be frequent if it appears at least $\varepsilon B|P|$ times. Clearly, the number of frequent colors is at most $|P|/(\varepsilon B|P|) = (\varepsilon B)^{-1}$. Consider a frequent color j and assume it appears n_j times in the point set P. We store an ε_j-approximation $\mathscr{E}_j$ for this color where $\varepsilon_j = \frac{\varepsilon|P|}{n_j}$. The size of $\mathscr{E}_j$ is $\min\left\{n_j, O\left(\frac{n_j}{\varepsilon|P|}\right)^{4/3}\right\}$ and we store $\mathscr{E}_j$ in a data structure in Theorem 10.A.3.

Now consider a query and consider dealing with the frequent colors in q_c. For every frequent color $j \in q_c$, we simply count the number of points from $\mathscr{E}_j$ and use it to get an estimate with the correct additive error. The query time is $O(\sqrt{|\mathscr{E}_j|}) = O\left(\frac{n_j}{\varepsilon|P|}\right)^{2/3}$. The total query time spent on all the frequent colors is thus

$$\sum_{j\in q_c,\ j \text{ is frequent}} O\left(\frac{n_j}{\varepsilon|P|}\right)^{2/3} \le \frac{1}{\varepsilon B} O\left(\frac{\varepsilon|P|B}{\varepsilon|P|}\right)^{2/3} = O\left(\frac{1}{\varepsilon B^{1/3}}\right)$$

where the inequality follows from the observation that $\sum n_j \leq |P|$ and since the exponent $2/3$ is less than one, the function $x^{2/3}$ is a concave function and thus the expression on the left is maximized when all the n_j's are equal. Since there are at most $(\varepsilon B)^{-1}$ colors the maximum is when each $n_j = |P|/(\varepsilon B)^{-1} = \varepsilon B|P|$. As a result, we can deal with the frequent colors within our claimed query time. The space is also clearly linear.

Dealing with infrequent colors. This is the more tricky part of the data structure. Consider an infrequent color j that appears n_j times. As before, we build an ε_j-approximation $\mathscr{E}_j$ for this color where $\varepsilon_j = \frac{\varepsilon|P|}{n_j}$ and the size of $\mathscr{E}_j$ is $O\left(\frac{n_j}{\varepsilon|P|}\right)^{4/3}$. Let X be the collection of $\mathscr{E}_j$ for all the infrequent colors j. We can bound the size of X asymptotically by

$$\sum_{j \text{ is infrequent}} \left(\frac{n_j}{\varepsilon|P|}\right)^{4/3} \leq \frac{1}{\varepsilon B}\left(\frac{\varepsilon B|P|}{\varepsilon|P|}\right)^{4/3} = \frac{B^{1/3}}{\varepsilon}$$

where the inequality follows from Jensen's inequality, i.e., that the function $x^{4/3}$ is a convex function that thus the expression on the left is maximized when each n_j is either maximized or minimized; however, as each infrequent color appears at most $\varepsilon B|P|$ times, the maximum is achieved when $\frac{1}{\varepsilon B}$ of the n_j's are set to $\varepsilon B|P|$ and the rest are set to 0.

We then transform X into the dual space using point-line duality. We build a $B^{-2/3}$-cutting Z in the dual space. By Lemma 10.A.1, Z has $O(B^{4/3})$ triangles that cover the entire plane where each triangle is intersected by $O(X/B^{2/3}) = O(1/(\varepsilon B^{1/3}))$ lines. For each triangle, we store the number of lines of each color that passes below it ($F = O(1/(\varepsilon B^{1/3}))$ frequencies), as well as the set of lines intersecting it ($O(1/(\varepsilon B^{1/3}))$ lines). In total we will asymptotically use

$$\frac{1}{\varepsilon B^{1/3}} \cdot B^{4/3} = \frac{B}{\varepsilon} \leq |P|$$

space. To bound the query time, consider a query point q which is dual to the query halfspace. Now observe that after locating the triangle in the cutting that contains q, we simply need to look at the stored frequencies in the triangle, as well as the set of lines that intersect the triangle. In total this will take $O(1/(\varepsilon B^{1/3}))$ time.

A Data Structure for 2D AHHS Queries

In our data structure for 2D queries, we use most of the ingredients that we developed for 3D. In particular, we can use $O(n)$ space such that for a given halfplane h, we can find a list of $O(\varepsilon_0^{-1})$ candidate colors in $O(\log n + \varepsilon_0^{-1})$ time. Similarly, we also build a level 0 shallow cutting $\mathscr{C}_0$ that allows us answer the query when h contains at most ε_0^{-1} points. However, we only build the first lower level shallow cutting $\mathscr{C}_1$ from Subsection 10.D. Consequently, if h contains up to $\varepsilon_0^{-1} w^{\beta}$ points, we can answer the query in optimal running time. Handling the remaining queries is where we deviate from the 3D data structure.

We now build a different hierarchy of shallow cuttings. Let $g_0 = \varepsilon_0^{-1} w^{\beta}$. We build a series of shallow cuttings using parameters $g_0, g_1, \cdots$ that will be determined shortly. In particular, we build a g_i-shallow cutting $\mathscr{C}'_i$ until g_i exceeds ε_0^{-3}; we then switch back to the 3D solution and as argued there, the total space of this part of the structure will be $O(n)$. For each cell Δ_C in $\mathscr{C}'_i$, we store the conflict list of Δ_C in the base structure of Theorem 10.D.1, with the following parameters: we have $|P| = \Theta(g_i)$, we want to set ε such that the query time equals $O(1/\varepsilon_0)$ and thus we must satisfy

$$\frac{1}{\varepsilon B^{1/3}} \leq \frac{1}{\varepsilon_0} \Leftrightarrow \varepsilon B^{1/3} \geq \varepsilon_0 \Leftrightarrow \varepsilon \left(\varepsilon|P|\right)^{1/3} \geq \varepsilon_0 \Leftarrow \varepsilon \left(\varepsilon g_i\right)^{1/3} \geq \varepsilon_0 \tag{10.10}$$

Thus, setting our parameters in way to satisfy Inequality 10.10 makes sure that the query time of the base structure we are using is $O(\varepsilon_0^{-1})$.

To guarantee the approximation factor, observe that we know that the query is outside the shallow cutting $\mathscr{C}'_{i-1}$ which implies there are at least g_{i-1} points in the query. Consequently, if we can ensure the following, then our approximation factor is also as desired:

$$\varepsilon|P| \leq \varepsilon_0 g_{i-1} \Leftarrow \varepsilon\Theta(g_i) \leq \varepsilon_0 g_{i-1}. \tag{10.11}$$

Inequalities 10.10 and 10.11 give us a recursion for g_i. To simplify the exposition, we can simply rescale the constant ε_0 to get rid of the constant in the $\Theta(\cdot)$ notation in Inequality 10.11 and turn the inequality into an equality. This yields $\varepsilon = \frac{\varepsilon_0 g_{i-1}}{g_i}$ which in turn yields

$$g_i \leq g_{i-1}\left(\varepsilon_0 g_{i-1}\right)^{\frac{1}{3}} \qquad \text{and recall that we have } g_0 = \frac{w^{\beta}}{\varepsilon_0}. \tag{10.12}$$

Now a simple induction yields that

$$g_i = \frac{1}{\varepsilon_0} \cdot \left(w^{\beta}\right)^{(4/3)^i}. \tag{10.13}$$

Recall that as soon as g_i reaches ε_0^{-3}, we switch to the 3D solution and from this point on, the remaining levels will consume $O(n)$ space. As a result, the space complexity of the data structure is $O(nj)$ where j is the smallest index such that $g_j \geq \varepsilon_0^{-3}$. From Equation 10.13, it is clear that $j = O(\log\log_w \varepsilon_0^{-1})$ and thus the space bound is $O(n \log\log_w \varepsilon_0^{-1})$.

Chapter 11

2D Generalization of Fractional Cascading on Axis-aligned Planar Subdivisions

Abstract

Fractional cascading is one of the influential and important techniques in data structures, as it provides a general framework for solving a common important problem: the iterative search problem. In the problem, the input is a graph G with constant degree. Also as input, we are given a (different) set of values for every vertex of G. The goal is to preprocess G such that when we are given a query value q, and a connected subgraph π of G, we can find the predecessor of q in all the sets associated with the vertices of π. The fundamental result of fractional cascading, by Chazelle and Guibas, is that using linear space, queries can be answered in $O(\log n + |\pi|)$ time, at essentially constant time per predecessor [CG86a]. While this technique has received plenty of attention in the past decades, an almost quadratic space lower bound for "two-dimensional fractional cascading" by Chazelle and Liu in STOC 2001 [CL04] has convinced the researchers that fractional cascading is fundamentally a one-dimensional technique.

In two-dimensional fractional cascading, the input includes a planar subdivision for every vertex of G and the query is a point q and a subgraph π and the goal is to locate the cell containing q in all the subdivisions associated with the vertices of π. In this paper, we show that it is actually possible to circumvent the lower bound of Chazelle and Liu for axis-aligned planar subdivisions.

We present a number of upper and lower bounds which reveal that in two-dimensions, the problem has a much richer structure. When G is a tree and π is a path, then queries can be answered in $O(\log n + |\pi| + \min\{|\pi|\sqrt{\log n}, \alpha(n)\sqrt{|\pi|}\log n\})$ time using linear space where α is an inverse Ackermann function; surprisingly, we show both branches of this bound are tight, up to the inverse Ackermann factor. When G is a general graph or when π is a general subgraph, then the query bound becomes $O(\log n + |\pi|\sqrt{\log n})$ and this bound is once again tight in both cases.

11.1 Introduction

Fractional cascading [CG86a] is one of the widely used tools in data structures as it provides a general framework for solving a common important problem: the iterative search problem, i.e., the problem of finding the predecessor of a single value q in multiple data sets. In the problem, we are to preprocess a degree-bounded "catalog" graph G where each vertex represents an input set of values from a totally ordered universe U; the input sets of different vertices of G are completely unrelated. Then, at the query time, given a value $q \in U$ and a connected subgraph π of G, the goal is to find the predecessor of q in the sets that correspond to the vertices of π. The fundamental theorem of fractional cascading is that one can build a data structure of linear size such that the queries can be answered in $O(\log n + |\pi|)$ time, essentially giving us constant search time per predecessor after investing an initial $O(\log n)$ search time [CG86a]. Many problems benefit from this technique [CG86b] since they need to solve the iterative search problem as a base problem.

Given its importance, it is not surprising that many have attempted to generalize this technique: The first obvious direction is to consider the dynamic version of the problem by allowing insertions or deletions into the sets of the vertices of G. In fact, Chazelle and Guibas themselves consider this [CG86a] and they show that with $O(\log n)$ amortized time per update, one can obtain $O(\log n + |\pi| \log\log n)$ query time. Later, Mehlhorn and Näher improve the update time to $O(\log\log n)$ amortized time [MN90] and then Dietz and Raman [DR91] remove the amortization. There is also some attention given to optimize the dependency of the query time on the maximum degree of graph G [GK09].

The next obvious generalization is to consider the higher dimensional versions of the problem. Here, each vertex of G is associated with an input subdivision and the goal is to locate a given query point q on every subdivision associated with the vertices of π. Unfortunately, here we run into an immediate roadblock already in two dimensions: After listing a number of potential applications of two-dimensional fractional cascading, Chazelle and Liu [CL04] "dash all such hopes" by showing an $\tilde{\Omega}(n^2)$ [1] space lower bound in the pointer-machine model for any data structure that can answer queries in $O(\log^{O(1)} n + |\pi|)$ time. Note that this lower bound can be generalized to also give a $\Omega(n^{2-\varepsilon})$ space lower bound for data structures with $O(\log^{O(1)} n) + o(|\pi| \log n)$ query time. As far as we can tell, progress in this direction was halted due to this negative result since the trivial solution already gives the $O(|\pi| \log n)$ query time, by just building individual point location data structures for each subdivision.

We observe that the lower bound of Chazelle and Liu does not apply to orthogonal subdivisions, a very important special case of planar point location problem. Many geometric problems need to solve this base problem, e.g., 4D orthogonal dominance range reporting [AAL12, ACT14], 3D point location in orthogonal subdivisions [Rah15], some 3D vertical ray-shooting problems [dBvKS95]. In geographic

[1]The $\tilde{\Omega}$ notation hides polylogarithmic factors.

information systems, it is very common to overlay planar subdivisions describing different features of a region to generate a complete map. Performing point location queries on such maps corresponds to iterative point locations on a series of subdivisions.

Motivated by this observation, we systematically study the generalization of fractional cascading to two dimensions, when restricted to orthogonal subdivisions. We obtain a number of interesting results, including both upper and lower bounds which show most of our results are tight except for the general path queries of trees where the bound is tight up to a tiny inverse Ackermann factor [CLRS22].

The problem definition The formal definition of the problem is as follows. The input is a degree-bounded connected graph $G = (V, E)$ where each vertex $v \in V$ is associated with an axis-aligned planar subdivision. Let n be the total number of vertices, edges, and faces in the subdivisions, which we call the *graph subdivision complexity*. We would like to build a data structure such that given a query (q, π), where q is a query point and π is a connected subgraph of G, we can locate q in all the subdivisions induced by vertices of π efficiently. We call this problem 2D Orthogonal Fractional Cascading (2D OFC).

Related Work

While the negative result of Chazelle and Liu [CL04] stops any progress on the general problem of two-dimensional fractional cascading, there have been other results that can be seen as special cases of two-dimensional fractional cascading. For example, Chazelle et al. [CEGea94] improved the result of ray shooting in a simple polygon by a $\log n$ factor. In a "geodesically triangulated" subdivision of n vertices, they showed it is possible to locate all the triangles crossed by a ray in $O(\log n)$ time instead of $O(\log^2 n)$, which resembles 2D fractional cascading. However, their solution relies heavily on the characteristic of geodesic triangulation and cannot be generalized to other problems. Chazelle's data structure for the rectangle stabbing problem [Cha86] can also be viewed as a restricted form of two-dimensional fractional cascading where $\pi = G$.

In recent years, interestingly, a technique similar to 2D fractional cascading has been used to improve many classical computational geometry data structures. While working on the 4D dominance range reporting problem, Afshani et al. [AAL12] are implicitly performing iterative point location queries along a path of a balanced binary tree on somewhat specialized subdivsions in $O(\log^{3/2} n)$ total time. Later Afshani et al. [ACT14] studied an offline version of tree point location problem and gave an optimal $O(n + k \log n)$ query time and $O(n)$ space data structure, where k is the number of query points and n is the subdivision size of a tree of height $O(\log n)$. The same idea is used to improve the result of 3D point location in orthogonal subdivisions. In that probelm, Rahul [Rah15] obtained another data structure with $O(\log^{3/2} n)$ query time.

Another related problem is the “unrestricted” version of fractional cascading where essentially π can be an arbitrary subgraph of G, instead of a connected subgraph. In one variant, we are given a set L of categories and a set S of n points in d dimensional space where each point belongs to one of the categories. The query is given by a d-dimensional rectangle r and a subset $Q \subset L$ of the categories. We are asked to report the points in S contained in r and belonging to the categories in Q. In 1D, Chazelle and Guibas [CG86b] provided a $O(|Q| \log \frac{|L|}{|Q|} + \log n + k)$ query time and linear size data structure, where k is the output size, together with a restricted lower bound. Afshani et al. [ASTW14] strenghtened the lower bound and presented several data structures for three-sided queries in two-dimensions. Their data structures match the lower bound within an inverse Ackermann factor for the general case.

Our Results

We study 2D OFC in a pointer machine model of computation. Some of our bounds involve inverse Ackermann functions. The particular definition that we use is the following. We define $\alpha_2(n) = \log n$ and then we define $\alpha_i(n) = \alpha^*_{i-1}(n)$, meaning, it’s the number of times we need to apply the $\alpha_{i-1}(\cdot)$ function to n until we reach a fixed constant. $\alpha(n)$ corresponds to the value of i such that $\alpha_i(n)$ is at most a fixed constant. Our results are summarized in Table 1.

Table 1: Our Results

Graph	**Query**	**Space**	**Query Time**	**Tight?**
Tree	Path	$O(n\alpha_c(n))$	$O(\min\{\lvert\pi\rvert\sqrt{\log n}, c\sqrt{\lvert\pi\rvert}\log n\} + \log n + \lvert\pi\rvert)$	Up to $\alpha_c(n)$ factor
Tree	Path	$O(n)$	$O(\min\{\lvert\pi\rvert\sqrt{\log n}, \alpha(n)\sqrt{\lvert\pi\rvert}\log n\} + \log n + \lvert\pi\rvert)$	Up to $\alpha(n)$ factor
Tree	Subtree	$O(n)$	$O(\log n + \lvert\pi\rvert\sqrt{\log n})$	yes
Graph	Path / Subgraph	$O(n)$	$O(\log n + \lvert\pi\rvert\sqrt{\log n})$	yes

Our results show some very interesting behavior. First, by looking at the last two rows of Table 1, we can see that we can always do better than the naïve solution by a $\sqrt{\log n}$ factor. Furthermore, this is tight. We show matching query lower bounds both when G can be an arbitrary graph but with π being restricted to a path and also when G is a tree but π is allowed to be any subtree of G. Second, when G is a tree and π is a path we get some variation depending on the length of the query path. When π is of length at most $\frac{\log n}{2}$, then we can answer queries in $O(|\pi|\sqrt{\log n})$ time, but when π is longer than $\frac{\log n}{2}$, we obtain the query bound of $O(\sqrt{|\pi|}\log n)$ (ignoring some inverse Ackermann factors). Furthermore, we give two lower bounds that show both of these branches are tight! When π is very long, longer than $\frac{\log^2 n}{2}$, then the query bound becomes $O(|\pi|)$ which is also clearly optimal.

11.2 Preliminaries

In this section, we introduce some geometric preliminaries and present the tools we will use to build the data structures and to prove the lower bounds.

Geometric Preliminaries

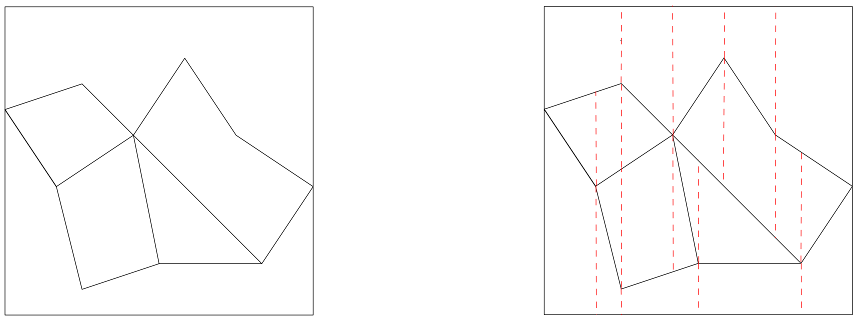
First we review the definition of planar subdivisions.

Definition 11.2.1. *A graph is said to be a* planar graph *if it can be embedded in the plane without crossings. A planar subdivision is a planar embedding of a planar graph where all the edges are straight line segments. The* complexity of a planar subdivision *is the sum of the vertices, edges, and faces of the subdivision.*

Planar point location, defined below, is one classical problem related to planar subdivisions:

Definition 11.2.2. *Given a planar subdivision S of complexity n, in the* planar point location *problem, we are asked to preprocess S such that given any query point q in the plane, we can find the face f in S containing q efficiently.*

Note that we can assume that the subdivision is enclosed by a bounding box. There are several different ways to solve the planar point location problem optimally in $O(\log n)$ query time and $O(n)$ space, see [GOT18] for detail. One simple solution uses trapezoidal decomposition, see [dBCvKO08] for a detailed introduction. Roughly speaking, given a planar subdivision S enclosed by a bounding box R, we construct a trapezoidal decomposition of the subdivision by extending two rays from every vertex of S, one upwards and one downwards. The rays stop when they hit an edge in S or the boundary of R. The faces of the subdivision we obtain after this transform will be only trapezoids. Figure 1 gives an example of trapezoidal decomposition. A crucial property of trapezoidal decomposition is that it increases the complexity of the subdivision by only a constant factor.



(a) A Planar Subdivision (b) After Trapezoidal Decomposition

Figure 1: Example of Trapezoidal Decomposition

We also review some concepts related to cuttings.

Definition 11.2.3. *Given a set H of n hyperplanes in the plane, a $(1/r)$-cutting, $1 \leq r \leq n$, is a set of (possibly open) disjoint simplices that together cover the entire plane such that each simplex intersects $O(n/r)$ hyperplanes of H. For each simplex in the cutting, the set of all hyperplanes of H intersecting it is called the* conflict list *of that simplex.*

$(1/r)$-cuttings are important in computational geometry as they enable us to apply the divide-and-conquer paradigm in higher dimensions. The following theorem by Chazelle [Cha93], after a series of work in the computational geometry community [Mat91, Mat95, Aga90, Aga91, CF90], shows the existence of $(1/r)$-cuttings of small size and an efficient deterministic algorithm computing $(1/r)$-cuttings.

Theorem 11.2.1 (Chazelle [Cha93])**.** *Given a set H of n hyperplanes in the plane, there exists a $(1/r)$-cutting, $1 \leq r \leq n$, of size $O(r^2)$, which is optimal. We can find the cutting and the corresponding conflict lists in $O(nr)$ time.*

In this paper, we will use intersection sensitive $(1/r)$-cuttings which is a generalization of $(1/r)$-cuttings. The following theorem is given by de Berg and Schwarzkopf [dBS95].

Theorem 11.2.2 (de Berg and Schwarzkopf [dBS95])**.** *Given a set H of n line segments in the plane with A intersections, we can construct a $(1/r)$-cutting, $1 \leq r \leq n$, of size $O(r + Ar^2/n^2)$. We can find the cutting and the corresponding conflict lists in time $O(n \log r + Ar/n)$ using a randomized algorithm.*

Note that by the construction of generalized cuttings, see [dBS95] for detail, the following corollary follows directly from Theorem 11.2.2,

Corollary 11.2.1. *Given an axis-aligned planar subdivision of complexity n, we can construct a $(1/r)$-cutting, $1 \leq r \leq n$, of size $O(r)$. More specifically, each cell of the cutting is an axis-aligned rectangle and the size of the conflict list of every cell is bounded by $O(n/r)$. We can find the cutting and the corresponding conflict lists in time $O(n \log n)$ using a randomized algorithm.*

Rectangle Stabbing

In d dimensional rectangle stabbing problem, we are given a set of n d-dimensional axis-parallel rectangles, our task is to build a data structure such that given a query point q, we can report the rectangles containing the query point efficiently. As noted earlier, Chazelle [Cha86] provides an optimal solution in two-dimensions, a linear-sized data structure that can answer queries in $O(\log n + t)$ time where t is the output size. The following lemma by Afshani et al. [AAL12] establishes an upper bound of this problem and it is obtained by a basic application of range trees [Ben79] with large fan-out and Chazelle's data structure.

Lemma 11.2.1 (Afshani et al. [AAL12])**.** *We can answer d dimensional rectangle stabbing queries in time $O(\log n \cdot (\log n / \log H)^{d-2} + t)$ using space $O(nH \log^{d-2} n)$, where n is the number of rectangles, t is the output size, and $H \geq 2$ is any parameter.*

A Pointer Machine Lower Bound Framework

We will use the pointer machine lower bound framework of Afshani [Afs13]. The framework deals with an abstract "geometric stabbing problem" which is defined by a set $\mathbb{R}$ of "ranges" and a set U of queries. An instance of the geometric stabbing problem is given by a set $R \subset \mathbb{R}$ of n "ranges" and the goal is to preprocess R to answer queries q. Given R, an element $q \in U$ (implicitly) defines a subset $R_q \subset R$ and the data structure is to output the elements of R_q. However, the data structure is restricted to operate in the (strengthened) pointer machine model of computation where the memory is a directed graph M consisting of "cells" where each cell can store an element of R as well as two pointers to other memory cells. At the query time, the algorithm must find a connected subgraph M_q of M where each element of R_q is stored in at least one memory cell of M_q. The size of M is a lower bound on the space complexity of the data structure and the size of R_q is a lower bound on the query time. However, the lower bound model allows for unlimited computation and allows the data structure to have complete information about the problem instance; the only bottleneck is being able to navigate to the cells storing the output elements. In addition, the framework assumes that we have a measure μ such that $\mu(U) = 1$. We need a slightly more precise version of the lower bound framework where the dependency on a certain constant is made explicit.

Theorem 11.2.3. *Assume, we have an algorithm that given any input instance $R \subset \mathbb{R}$ of n ranges, it can store R in a data structure of size $S(n)$ such that given any query $q \in U$, it can answer the query in $Q(n) + \gamma|R_q|$ time.*

Then, suppose we can construct an input set $R \subset \mathbb{R}$ of n ranges such that the following two conditions are satisfied: (i) every query point $q \in U$ is contained in exactly $|R_q| = t$ ranges and $\gamma t \geq Q(n)$; (ii) there exists a value v such that for any two ranges $r_1, r_2 \in R$, $\mu(\{q \in U | r_1, r_2 \in R_q\})$ is well-defined and is upper bounded by v. Then, we must have $S(n) = \Omega(tv^{-1}/2^{O(\gamma)}) = \Omega(Q(n)v^{-1}/2^{O(\gamma)})$.

For the proof of this theorem, we refer the readers to section 11.A. In our applications, μ will basically be the Lebesgue measure and U will be the unit cube.

11.3 Queries on Catalog Paths

In this section, we give a simple solution for when the catalog graph is a path. It will be used as a building block for later data structures.

Theorem 11.3.1. *Consider a catalog path G, in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can construct a data structure using $O(n)$ space such that given any query (q, π), where q is a query point and π is a subpath, all regions containing q along π can be reported in time $O(\log n + |\pi|)$.*

Proof. We can convert each subdivision into a set of disjoint rectangles of total size $O(n)$ using trapezoidal decomposition [dBCvKO08]. Then, we partition G into

$m = \lceil |G|/\log n \rceil$ paths, $G_1, \cdots, G_m$ where each path except potentially for G_m has size $\log n$ and G_m has size at most $\log n$.

Now we use an observation that was also made in previous papers [AAL10, AAL12, Rah15]: when $H = G$, the two-dimensional fractional cascading can be reduced to rectangle stabbing. As a result, for each G_i, $1 \leq i \leq m$, we collect all the rectangles of its subdivisions and build a 2D rectangle stabbing data structure on them. By Lemma 11.2.1 this requires $O(n)$ space. Now given a query subpath of length $|\pi|$, we use the rectangle stabbing data structures on the subdivisions of each G_i as long as $|G_i \cap \pi| > 0$. Since π is a path, for at most two indices i we will have $0 < |G_i \cap \pi| < \log n$ and for the rest $|G_i \cap \pi| = |G_i| = \log n$. This gives us $O(\log n + |\pi|)$ query time. □

11.4 Path Queries on Catalog Trees

Now we consider answering path queries on catalog trees. We first show optimal data structures for trees of different heights. It turns out we need different data structures to achieve optimality when heights differ. We then present a data structure using $O(n\alpha_c(n))$ space that can answer path queries in $O(\log n + |\pi| + \min\{|\pi|\sqrt{\log n}, \sqrt{|\pi| \log n}\})$ time and a data structure using $O(n)$ space answering path queries in $O(\log n + |\pi| + \min\{|\pi|\sqrt{\log n}, \alpha(n)\sqrt{|\pi| \log n}\})$ time, where $c \geq 3$ is any constant and $\alpha_k(n)$ is the k-th function in the inverse Ackermann hierarchy [CLRS22] and $\alpha(n)$ is the inverse Ackermann function [CLRS22] . We also present lower bounds for our data structures. Without loss of generality, we assume the tree is a binary tree.

Trees of height $\leq \frac{\log n}{2}$

The Upper Bound

For trees of this height, we present the following upper bound. The main idea is to use the sampling idea that is employed previously [CLP11, AAL12], however, there are some main differences. Instead of random samples or shallow cuttings, we use intersection sensitive cuttings [dBS95] and more notably, the fractional cascading on an arbitrary tree cannot be reduced to a geometric problem such as 3D rectangle stabbing, so instead we do something else.

Lemma 11.4.1. *Consider a catalog tree of height $h \leq \frac{\log n}{2}$ in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n)$ space such that given any query (q, π), where q is a query point and π is a path, all regions containing q along π can be reported in time $O(\log n + |\pi|\sqrt{\log n})$.*

Proof. Let r be a parameter to be determined later. Consider a planar subdivision A_i and let n_i be the number of rectangles in A_i. We create an intersection sensitive (r^2/n_i)-cutting C_i on A_i. By Corollary 11.2.1, C_i contains $O(n_i/r^2)$ cells and each cell of C_i is an axis-aligned rectangle. Furthermore, the conflict list size of each rectangle is $O(r^2)$. For each cell in C_i, we build an optimal point location data structure on its conflict list. The total space usage is linear, since total size of the conflict lists is linear.

Then, we consider every path of length at most $\log r$ in the catalog graph, and we call them subpaths. For every subpath, we collect all the cells of the cuttings belonging to the vertices of the subpath and build a 2D rectangle stabbing data structure on them. Since the degree of any vertex is bounded by 3, each vertex is contained in at most

$$\sum_{j=0}^{\log r}\sum_{i=0}^{j} 3^i \cdot 3^{j-i} = \Theta(r^{\log 3}\log r)$$

many subpaths. Then the total space usage of the 2D rectangle stabbing data structures is bounded by $O(n\log r/r^{2-\log 3}) = O(n)$. Given any query path π, it can be covered by $|\pi|/\log r$ subpaths. For each subpath, we can find all the cells of the cuttings containing the query point in $O(\log n)$ time and then perform an additional point location query on its conflict list, for a total of $O(\log n + (\log r)^2)$ query time per subpath. Thus, the query time of this data structure is bounded by

$$\frac{|\pi|}{\log r}(\log n + \log r \cdot \log r).$$

We pick $r = 2^{\sqrt{\log n}}$, then we obtain the desired $O(\log n + |\pi|\sqrt{\log n})$ query time. □

The Lower Bound

We now present a matching lower bound. We show the following:

Lemma 11.4.2. *Assume, given any catalog tree of height $\sqrt{\log n} \le h \le \frac{\log n}{2}$ in which each vertex is associated with a planar subdivision with n being the total complexity of the subdivisions, we can build a data structure that satisfies the following: it uses at most $n2^{\varepsilon\sqrt{\log n}}$ space, for a small enough constant ε, and it can answer 2D OFC queries (q,π). Then, its query time must be $\Omega(|\pi|\sqrt{\log n})$.*

Proof. We will use the following idea: We consider a special 3D rectangle stabbing problem and show a lower bound using Theorem 11.2.3. We will use the 3D Lebesgue measure, denoted by $V(\cdot)$. Then we show a reduction from this problem to a 2D OFC problem on trees to obtain the desired lower bound.

We consider the following instance of 3D rectangle stabbing problem. The input n rectangles are partitioned into h sets of size n/h each. The rectangles of each set are pairwise disjoint and they tile the unit cube in 3D. The depth (i.e., the length of the side parallel to the z-axis) of rectangles in set i is $1/2^i$. Figure 2 is an example of rectangles in set 2. Note that the depth of each rectangle is $\frac{1}{2^2}$, while the other two dimensions can be arbitrary as long as they together tile the unit cube and the number of them is n/h. Note that the rectangles in this set can be viewed as four subsets of rectangles by cutting the z-dimension of the unit cube into four even intervals. The projection of each subset into the xy-plane gives us an axis-aligned planar subdivision.

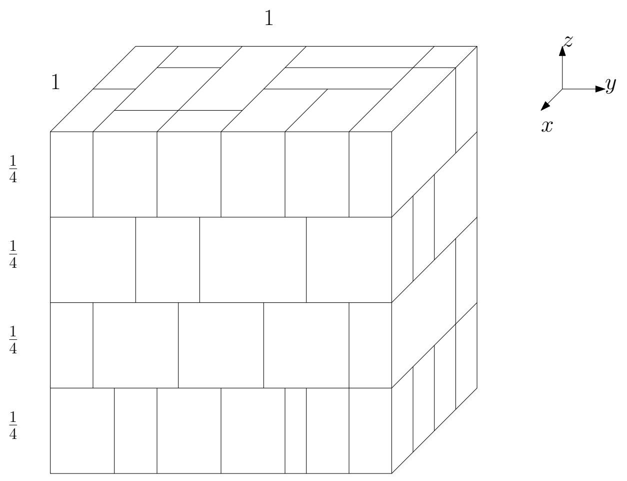


Figure 2: An example of rectangles in set 2

We first show the reduction: assume, we are given an instance of special 3D rectangle stabbing problem. We build a balanced binary tree of height h on the z-axis as the catalog graph. Note that the number of vertices at layer i of the tree is the same as the number of different z-intervals rectangles at set i cover. We project the rectangles having the same z-interval to the xy-plane and obtain a 2D axis-aligned planar subdivision. We attach each of the subdivisions to the corresponding vertices. By construction, the regions reported by a point location query from the root to a leaf correspond to all the rectangles containing the query point after lifting it to 3D appropriately.

Now we describe a hard instance of the rectangle stabbing problem to establish a lower bound. It will have rectangles of h different shapes. For each shape, we tile (disjointly cover) the unit cube using isometric copies of the shape to obtain a set of rectangles. We collect every r different shapes into a class and obtain h/r classes, where r is a parameter to be determined later. We say that the i-th rectangle in a class has group number i, $1 \leq i \leq r$. Now we specify the dimensions (i.e., side lengths) of the rectangles. For a rectangle in class $i = 0, \cdots, h/r - 1$, with group number $j = 0, \cdots, r-1$, its dimensions are

$$[\frac{1}{K^j} \times K^j \cdot 2^{ir+j} \cdot V \times \frac{1}{2^{ir+j}}],$$

where K, V are parameters to be determined later and the $[W \times H \times D]$ notation denotes an axis-aligned rectangle with width W, height H, and depth D. Observe that every rectangle has volume V and thus we need $1/V$ copies to tile the unit cube. By setting $V = h/n$, the total number of rectangles we generate is n. Also note that all the rectangles in the same group are pairwise disjoint and they together cover the whole unit cube. This implies for any query point q in the unit cube, it is contained in exactly $h = |\pi|$ rectangles.

Now we analyze the intersection of any two rectangles. First, observe that given two axis-aligned rectangles with dimensions $[W_1 \times H_1 \times D_1]$ and $[W_2 \times H_2 \times D_2]$, their intersection is an axis-aligned rectangle with dimensions at most $[\min\{W_1, W_2\} \times \min\{H_1, H_2\} \times \min\{D_1, D_2\}]$. Second, by our construction, the rectangles that have identical width, depth, and height are disjoint. As a result, either the width of the two rectangles will differ by a factor K or their depth will differ by a factor 2^r. This means that, the maximum intersection volume of any two rectangles R_1, R_2 in class i_1, i_2, group j_1, j_2 can be achieved only in one of the following two cases:

$$V(R_1 \cap R_2) = \begin{cases} \frac{V}{2K} & i_1 = i_2 \text{ and } j_1 = j_2 + 1, \\ \frac{V}{2^r} & i_1 = i_2 + 1 \text{ and } j_1 = j_2. \end{cases}$$

We set $K = 2^r$, then the intersection volume of any two rectangles is bounded by $v = V/2^r$. However, for the construction to be well-defined, the side length of the rectangles cannot exceed 1 as otherwise, they do not fit in the unit cube. The largest height of the rectangles is obtained for $j = r-1$ and $i = \frac{h}{r} - 1$. Thus, we must have,

$$K^{r-1} 2^{r(\frac{h}{r}-1)+r-1} V \leq 1.$$

By plugging the values $V = h/n$ and $K = 2^r$ we get that we must have

$$2^{r^2-r} 2^{h-1} h < n \tag{11.1}$$

Since by our assumptions $h \leq \frac{\log n}{2}$, it follows that by setting $r = \frac{\sqrt{\log n}}{4}$, the inequality (11.1) holds.

If $\gamma h \geq Q(n)$ holds, then we satisfy the first condition of Theorem 11.2.3 and thus we obtain the space lower bound of

$$S(n) = \Omega\left(\frac{Q(n)v^{-1}}{2^{O(\gamma)}}\right) = \Omega\left(\frac{n2^r}{\log n 2^{O(\gamma)}}\right). \tag{11.2}$$

Now observe that if we set $\gamma = \delta\sqrt{\log n}$, for a sufficiently small $\delta > 0$, then it follows that the data structure must use more than $\Omega(n2^{\Omega(\sqrt{\log n})})$ space. However, by the statement of our lemma, we are not considering such data structures. As a result, when $\gamma = \delta\sqrt{\log n}$, the query time must be large enough that the first condition of the framework does not hold, meaning, we must have $Q(n) \geq \gamma h = \delta\sqrt{\log n} h = \Omega(|\pi|\sqrt{\log n})$. □

Trees of height $h > \frac{\log n}{2}$ and $h \leq \frac{\log^2 n}{2}$

The Upper Bound

We start with the following lemma which gives us a data structure that can only answer query paths that start from the root and finish at a leaf. The main idea here is used previously in the context of four-dimensional dominance queries [AAL12, CLP11] and it uses the observation that such "root to leaf" queries can be turned into a geometric problem, the 3D rectangle stabbing problem.

Lemma 11.4.3. *Consider a balanced catalog tree of height $h > \frac{\log n}{2}$ and $h \le \frac{\log^2 n}{2}$ in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n)$ space such that given any query (q,π), where q is a query point and π is a path starting from the root to a leaf, all regions containing q along π can be reported in time $O(\sqrt{|\pi|}\log n)$.*

Proof. Let r be a parameter to be determined later. For each subdivision A_i, we create an intersection sensitive (r/n_i)-cutting C_i on A_i. By the same argument as Lemma 11.4.1, all the cells in the cuttings are axis-aligned rectangles satisfying (i) the conflict set size of any cell in C_i is bounded by $O(r)$ and (ii) the total number of cells in C_i is $O(n_i/r)$.

Now we lift each cell in the cuttings to 3D rectangles and collect all the 3D rectangles to construct a 3D rectangle stabbing data structure for it. This is done as follows. We assign a z range for each vertex in the catalog tree; Let m be the number of leaves. Order the leaves of the catalog tree from left to right and for the i-th leaf $l_i, i \in \{1,2,\cdots,m\}$, we assign the range $[i-1,i)$ as its z range. For any internal vertex, its z range is the union of the z ranges of its children. Then, we lift the 2D rectangles induced by the subdivision of a vertex to a 3D rectangle using the z range (i.e., by forming the Cartesian product of the rectangle and the z range). We store the 3D rectangles in a rectangle stabbing data structure. Given a query point $q = (x_q, y_q)$ and a query path π, we first lift q to be (x_q, y_q, z_q), where z_q is any z value in the z range of the deepest vertex in π, and then query the 3D rectangle stabbing data structure.

In addition, for each cell in a cutting, we build an optimal point location data structure on its conflict set. All these point location data structures take space $O(\sum_i n_i) = O(n)$ in total and each of them can answer a point location query in time $O(\log r)$.

To achieve space bound $O(n)$ for the 3D rectangle stabbing data structure, it suffices to choose $H = \frac{r}{\log(n/r)}$. We then balance the query time for 3D rectangle stabbing and 2D point locations to achieve the optimal query time

$$\log n \cdot \frac{\log n}{\log \frac{r}{\log(n/r)}} = h \cdot \log r.$$

We pick $r = 2^{\log n/\sqrt{h}}$ and the query time is bounded by $O(\sqrt{h}\log n) = O(\sqrt{|\pi|}\log n)$. □

The above data structure is not a true fractional cascading data structure because it can only support restricted queries. To be able to answer query paths of arbitrary lengths $> \frac{\log n}{2}$ and $\le \frac{\log^2 n}{2}$, we need the following result.

Lemma 11.4.4. *Consider a catalog tree in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions and let h_1 and h_2, $h_1 < h_2$, be two fixed parameters. We can build a data structure using $O(n\log(h_2/h_1))$ space such that given any query (q,π), where q is a query point and π is a path whose* *length obeys $h_1 \leq |\pi| \leq h_2$, all regions containing q along π can be reported in time $O(\sqrt{|\pi|}\log n)$.*

Proof. First, observe that w.l.o.g, we can assume that the height of the catalog tree is at most h_2: we can partition the catalog tree into a forest by cutting off vertices whose depth is a multiple of h_2. Since the length of π is at most h_2, it follows that π can only contain vertices from at most two of the trees in the resulting forest, meaning, answering π can be reduced to answering at most two queries on trees of height at most h_2.

Thus, w.l.o.g., assume v is the root of the catalog tree of height h_2 and π is a path of length at least h_1 in this catalog tree. We build the following data structures. Let $v_1, \cdots, v_m$ be the vertices at height $h_2/2$. Let T_0 be the tree rooted at v and cut off at height $h/2$ with $v_1, \cdots, v_m$ being leafs and T_i be the tree rooted at v_i, $1 \leq i \leq m$. We build $m+1$ data structures of Lemma 11.4.3 on $T_0, \cdots, T_m$ and then we recurse on each of the $m+1$ trees. The recursion stops once we reach subproblems on trees of height at most h_1.

Since the data structure of Lemma 11.4.3 uses $O(n)$ space, at each recursive level, the total space usage of data structures we constructed is $O(n)$. Over the $O(\log(h_2/h_1))$ recursion levels, this sums up to $O(n\log(h_2/h_1))$ space.

Now we analyze the query time. Given a query (q, π), we may query several data structures that together cover the whole path of π. Let u be the highest vertex on π. We can decompose π into two disjoint parts π_1 and π_2, that start from u and end at vertices u_1 and u_2 respectively, with u_1 and u_2 being descendants of u. It thus suffices to only answer π_1, as the other path can be answered similarly. The first observation is that we can find a series of data structures that can be used to answer disjoint parts of π_1. The second observation is that we can afford to make the path a bit longer to truncate the recursion. We now describe the details.

Consider the trees $T_0, \cdots, T_m$ defined at the top level of the recursion. If π_1 is entirely contained in one of the trees, then we recurse on that tree. Otherwise, u is contained in T_0 and u_1 is contained in some subtree T_i. Now, π_1 can be further subdivided into two smaller "anchored" paths: one from u to v_i ("anchored" at u) and another from v_i to u_1 ("anchored" at v_i) and each smaller path can be answered recursively in the corresponding tree. Thus, it suffices to consider answering the query q along an anchored path.

Thus, consider the case of answering an anchored path π' in the data structure. To reduce the notation and clutter, assume π' is an anchored path, starting from the root of T and ending at a vertex u. Assume the vertices $v_1, \cdots, v_m$ and trees $T_0, \cdots, T_m$ are defined as above. First, consider the case when the height of T is at most h_1; in this case, we have built an instance of the data structure of Lemma 11.4.3 on T but not on the trees $T_0, \cdots, T_m$. In this case, we simply answer q on a root of leaf path in T that includes π', e.g., by a picking a leaf in the subtree of u. In this case, we will be performing a number of "useless" point location queries, in particular those on the descendants of u. However, as the height of T is at most h_1, it follows that the query bound stays asymptotically the same: $O(\sqrt{h_1}\log n)$. Furthermore, there is no recursion in this case and thus this cost is paid only once per anchored path. The second case is when the height of T is greater than h_1. In this case, if u lies in T_0 we simply recurse on T_0 but if u lies in a tree T_i, we first query the data structure of Lemma 11.4.3 using the path from the root of T until v_i, and then we recurse on T_i. As a result, answering the anchored path query reduces to answering at most one query on an instance of data structure Lemma 11.4.3 and another recursive "anchored" on a tree of half the height. Thus, the i-th instance of the data structure Lemma 11.4.3 that we query covers at most $1/2^i$ fraction of the anchored path. Thus, if k is the length of the anchored path, it follows that the total query time of all the data structures we query is bounded by

$$\sum_{i=1}^{\infty} \frac{\sqrt{k}\log n}{2^i} = O(\sqrt{k}\log n) = O(\sqrt{|\pi|}\log n).$$

□

We now reduce the space of the above lemma dramatically. We will repeatedly use a "bootstrapped" data structure. The following lemma establishes how we can bootstrap a base data structure to obtain a more efficient one.

Lemma 11.4.5. *Consider a catalog tree of height $h > \frac{\log n}{2}$ and $h \leq \frac{\log^2 n}{2}$ in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. Assume, for any fixed value Δ, $\omega(1) \leq \Delta \leq \log n$, we can build a "base" data structure that can answer a 2D OFC query (q,π) in $Q_b(n) = O(\sqrt{|\pi|}\log n)$ time as long as π is path of length between $\frac{\log^2 n}{2\Delta}$ and $\frac{\log^2 n}{2}$. Furthermore, assume it uses $S_b(\Delta,n) = O(nf(\Delta))$ space, for some function f which is monotone increasing in Δ and for $\Delta = \omega(1)$ we have $f(\Delta) = \omega(1)$.*

Then, for any given fixed value Δ, $\omega(1) \leq \Delta \leq \log n$, we can build a "bootstrapped" data structure that can answer a 2D OFC query (q,π) in $Q_b(n) + O(\sqrt{|\pi|}\log n)$ time as long as π is path of length between $\frac{\log^2 n}{2\Delta}$ and $\frac{\log^2 n}{2}$. Furthermore, it uses $O(nf^*(\Delta))$ space, where $f^*(\cdot)$ is the iterative $f(\cdot)$ function which denotes how many times we need to apply $f(\cdot)$ function to Δ to reach a constant value.

Proof. We construct an intersection sensitive $(f(\Delta)/n_i)$-cutting C_i for each planar subdivision A_i attached to the tree. Call these the "first level" cuttings. Similar to the analysis in Lemma 11.4.1, we obtain $O(n_i/f(\Delta))$ cells, which are disjoint axis-aligned rectangles, for each C_i and thus $n' = O(n/f(\Delta))$ cells in total. Each cell in the cutting has a conflict list of size $O(f(\Delta))$ and on that we build a point location data structure. This takes $O(n)$ space in total. We store the cells of the cutting in an instance of the base data structure with parameter Δ. Call this data structure $\mathscr{A}_1$. The space usage of $\mathscr{A}_1$ is

$$S_b(\Delta,n') = O(n'f(\Delta)) = O(n).$$

Now we consider a query (q,π). Let $\delta_1 = \log(f(\Delta))$. Consider the case when $\frac{1}{2}\cdot\left(\frac{\log^2 n}{\Delta}\right) \leq |\pi| \leq \frac{1}{2}\cdot\left(\frac{\log n}{\delta_1}\right)^2$. In this case, as $\mathscr{A}_1$ is built with parameter Δ, we can query it with (q,π). Thus, in $Q_b(n)$ time, for every subdivision on path π, we find the cell of the cutting that contains q. Then, we use the point location data structure on the conflict lists of the cells to find the original rectangle containing q. This takes an additional $O(\log(f(\Delta)))$ as the size of each conflict is $O(f(\Delta))$. Thus, the query time in this case is

$$Q_b(n)+O(|\pi|\log(f(\Delta))) = Q_b(n)+O(\sqrt{|\pi|}\log n)$$

since we have $|\pi| \le \frac{1}{2}\cdot\left(\frac{\log n}{\delta_1}\right)^2 = \frac{1}{2}\cdot\left(\frac{\log n}{\log(f(\Delta))}\right)^2$.

Thus, the only paths we cannot answer yet are those when $\frac{1}{2}\cdot\left(\frac{\log n}{\delta_1}\right)^2 \le |\pi| \le \frac{\log^2 n}{2}$. In this case, we can bootstrap. First, observe that we can build a data structure $\mathscr{A}'$ on the the original rectangles, where $\mathscr{A}'$ is an instance of the base data structure but this time with parameter Δ set to δ_1^2.

This will take $S_b(\delta_1^2,n) = O(nf(\delta_1^2))$ space. Thus, the total space consumption is

$$O(n)+S_b(\delta_1^2,n) = O(n)+O(nf(\log^2(f(\Delta)))) = O(n)+O(nf(f(\Delta))) \qquad (11.3)$$

where the last inequality follows since $f(\cdot)$ is a monotone increasing function and $\log^2(f(\Delta)) < f(\Delta)$ as $f(\Delta)=\omega(1)$. By construction, the data structure $\mathscr{A}'$ is built to handle exactly paths of this lenghth but it is using too much space. The idea here is that we can repeat the previous technique using "second level" cuttings to obtain a data structure $\mathscr{A}_2$: for a subdivision of size n_i, build a $(f(f(\Delta))/n_i)$-cutting, called the "second level" cutting. By repeating the same idea we used for the first level cuttings, we can spend additional $O(n)$ space to build a data structure $\mathscr{A}_2$ which can answer queries (q,π) as long as $\frac{1}{2}\cdot\left(\frac{\log^2 n}{\Delta}\right) \le |\pi| \le \frac{1}{2}\cdot\left(\frac{\log n}{\delta_2}\right)^2$ where $\delta_2 = \log(f(f(\Delta)))$. By repeating this process for $f^*(\Delta)$ steps, we can obtain the claim data structure. $\square$

We will essentially begin with the data structure in Lemma 11.4.4 and use Lemma 11.4.5 to bootstrap. To facilitate the description, we define two useful functions first. Let $\log^* n$ be the iterated log function, i.e., the number of times we need to apply log function to n until we reach 2. We define $\log^{*(i)} n$ as follows

$$\log^{*(i)} n = \begin{cases} \log^{*(i)}(\log^{*(i-1)} n)+1 & n>1,\\ 0 & n\le 1.\end{cases}$$

In other words, it is the number of times we need to apply $\log^{*(i-1)}$ function to n until we reach 2. We also defined the following function

$$\tau(n) = \{\min i : \log^{*(i)} n \le 3\}.$$

In fact, $\log^{*(i)} n$ is the $(i+2)$-th function of the inverse Ackermann hierarchy [CLRS22] and $\tau(n)=\alpha(n)-3$, where $\alpha(n)$ the inverse Ackermann function [CLRS22].

Lemma 11.4.6. *Consider a catalog tree of height $h > \frac{\log n}{2}$ and $h \le \frac{\log^2 n}{2}$ in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n\alpha_c(n))$ space, where $c \ge 3$ is any constant and $\alpha_c(n)$ is the c-th function of the inverse Ackermann hierarchy, such that given any query (q,π), where q is a query point and π is a path of length $> \frac{\log n}{2}$ and $\le \frac{\log^2 n}{2}$, all regions containing q along π can be reported in time $O(\sqrt{|\pi|}\log n)$. Furthermore, we can also build a data structure using $O(n)$ space answering queries in time $O(\alpha(n)\sqrt{|\pi|}\log n)$, where $\alpha(n)$ is the inverse Ackermann function.*

Proof. By Lemma 11.4.4, if we set $h_1 = \frac{\log n}{2}$ and $h_2 = \frac{\log^2 n}{2}$, we obtain a data structure using $O(n\log\log n)$ answering queries in time $O(\sqrt{|\pi|}\log n)$. By picking $\Delta = \log n$, $f = \log n$, we can apply Lemma 11.4.5 to reduce the space to $O(n\log^*(\log n)) = O(n\log^* n)$ while achieving the same query time. If we again pick $\Delta = \log n$, but $f = \log^* n$, by applying Lemma 11.4.5 again, the space is further reduced to $O(n\log^{**}(\log n)) = O(n\log^{**} n)$. We continue this process until $\log^{*(i)} n$ is less than three. Note that we will need to pay $O(\sqrt{|\pi|}\log n)$ extra query time each time we apply Lemma 11.4.5. We will end up with a linear-sized data structure with query time $O(\tau(n)\sqrt{|\pi|}\log n) = O(\alpha(n)\sqrt{|\pi|}\log n)$. On the other hand, if we stop applying Lemma 11.4.5 after a constant c many rounds, we will end up with a $O(n\log^{*(c)} n) = O(n\alpha_{c+2}(n))$ sized data structure with the original $O(\sqrt{|\pi|}\log n)$ query time. □

The Lower Bound

We show a matching lower bound in this section.

Lemma 11.4.7. *Assume, given any catalog tree of height $\frac{\log n}{2} < h \le \frac{\log^2 n}{2}$ in which each vertex is associated with a planar subdivision with n being the total complexity of the subdivisions, we can build a data structure that satisfies the following: it uses at most $n2^{\varepsilon \log n/\sqrt{h}}$ space, for a small enough constant ε, and it can answer 2D OFC queries (q,π). Then, its query time must be $\Omega(\sqrt{|\pi|}\log n)$.*

Proof. We first describe a hard input instance for a 3D rectangle stabbing problem and later we show that this can be embedded as an instance of 2D OFC problem on a a tree of height h. Also, we actually describe a tree of height $h' = \frac{h+(\log n)/4}{2} \le h$. This is not an issue as we can add dummy vertices to the root to get the height to exactly h.

We begin by describing the set of rectangles. Each rectangle is assigned a "class number" and a "group number". The number of classes is $\frac{\sqrt{h}}{2}$ and the number of groups is $\frac{\log n}{4\sqrt{h}} + \sqrt{h}$. The rectangles with the same class number and group number will be disjoint, isometric and they would tile the unit cube. Rectangles with class $i = 0,\cdots,\frac{\sqrt{h}}{2}-1$ and group $j = 0,\cdots,\frac{\log n}{4\sqrt{h}}-1$ will be of shape

$$[\frac{1}{K^j} \times K^j \cdot 2^{ir+j} \cdot V \times \frac{1}{2^{ir+j}}],$$

where V, K are some parameters to be determined later and $r = \frac{\sqrt{\log n}}{4\sqrt{h}}$. Similarly, rectangles with class $i = 0, \cdots, \frac{\sqrt{h}}{2} - 1$ and group $j = \frac{\log n}{4\sqrt{h}}, \cdots, \frac{\log n}{4\sqrt{h}} + \sqrt{h} - 1$ will be of shape

$$[\frac{1}{K^j} \times K^j \cdot 2^{ir+r} \cdot V \times \frac{1}{2^{ir+r}}].$$

The total number of different shapes is $\frac{\sqrt{h}}{2} \cdot (\frac{\log n}{4\sqrt{h}} + \sqrt{h}) = h'$. Note that each rectangle has volume V, so the total number of rectangles we use in all the tilings is n by setting $V = h'/n$. By our construction any query point is contained in $t = h' = |\pi|$ rectangles. Now we analyze the maximal intersection volume of two rectangles. By the same argument as in the proof of Lemma 11.4.2 the maximal intersection volume can only be achieved by two rectangles when they are in the same class and adjacent groups or in the same group of adjacent classes. For two rectangles R_1 and R_2 in group j_1, j_2 of class i_1, i_2, we have

$$V(R_1 \cap R_2) = \begin{cases} \frac{V}{2K} & i_1 = i_2 \text{ and } j_1 = j_2 + 1 \leq \frac{\log n}{4\sqrt{h}}, \\ \frac{V}{K} & i_1 = i_2 \text{ and } \frac{\log n}{4\sqrt{h}} \leq j_1 = j_2 + 1, \\ \frac{V}{2^r} & i_1 = i_2 + 1 \text{ and } j_1 = j_2. \end{cases}$$

We set $K = 2^r$, then the intersection of any two rectangle is no more than $v = V/2^r$.

We also need to make sure no side length of any rectangle exceeds the side length of the unit cube. The maximum side length can only be obtained when $i = \frac{\sqrt{h}}{2} - 1$ and $j = \frac{\log n}{4\sqrt{h}} + \sqrt{h} - 1$ in the second dimension. We must have

$$K^{\frac{\log n}{4\sqrt{h}} + \sqrt{h} - 1} \cdot 2^{r(\frac{\sqrt{h}}{2} - 1) + \frac{\log n}{4\sqrt{h}}} \cdot V \leq 1$$

Plugging $K = 2^r$ and $V = h'/n$ in, we must have

$$2^{r\frac{\log n}{4\sqrt{h}} + r(\sqrt{h} - 1)} \cdot 2^{r(\frac{\sqrt{h}}{2} - 1) + \frac{\log n}{4\sqrt{h}}} \cdot h' \leq n \tag{11.4}$$

Since $\frac{\log n}{2} \leq h \leq \frac{\log^2 n}{2}$ and $r = \frac{\log n}{4\sqrt{h}}$, (11.4) holds.

Suppose $\gamma h \geq Q(n)$, then the first condition of Theorem 11.2.3 is satisfied and we get the lower bound of

$$S(n) = \Omega(\frac{tv^{-1}}{2^{O(\gamma)}}) = \Omega(\frac{n2^r}{2^{O(\gamma)}}).$$

Observe that by setting $\gamma = \frac{\delta \log n}{\sqrt{h}}$ for a sufficiently small $\delta > 0$, the data structure must use $\Omega(n2^{\Omega(\log n/\sqrt{h})})$ space, which contradicts the space usage in our theorem. Therefore, $Q(n) \geq \gamma h = \frac{\delta \log n}{\sqrt{h}} h = \Omega(\sqrt{h} \log n) = \Omega(\sqrt{|\pi|} \log n)$. It remains to show that this set of rectangles can actually be embedded into an instance of the 2D OFC problem. To do that, we describe the tree T that can be used for this embedding. See Figure 3. We hold the convention that the root of T has depth 0. Starting from the root,

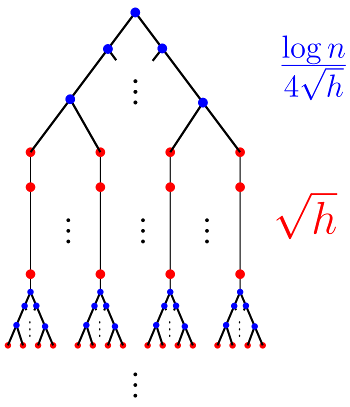


Figure 3: A difficult tree for fractional cascading.

until depth $\frac{\log n}{4\sqrt{h}}$, every vertex will have two children (blue vertices in Figure 3) then we will have $\sqrt{h}$ vertices with one child (red vertices in Figure 3). Then this pattern continues for $\frac{\sqrt{h}}{2}$ steps. The first set of blue and red vertices correspond to class 1, the next to class 2 and so on. Within each class, the top level corresponds to group 1 and so on. To be specific, vertices at depth $(\frac{\log n}{4\sqrt{h}} + \sqrt{h})i + j$ of the tree have rectangles of class i and group j. Now, it can be seen that the rectangles can be assigned to the vertices of T, similar to how it was done in Lemma 11.4.2. The notable difference here is that the depth of the rectangles decreases as the group number increases from 0 to $\frac{\log n}{4\sqrt{h}} - 1$ but then it stays the same from $\frac{\log n}{4\sqrt{h}}$ until $\frac{\log n}{4\sqrt{h}} + \sqrt{h} - 1$ but this exactly corresponds to the structure of the tree T. $\square$

Trees of height $> \frac{\log^2 n}{2}$

For trees of this height, we have:

Lemma 11.4.8. *Consider a catalog tree of height $> \frac{\log^2 n}{2}$ in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n)$ space such that given any query (q, π), where q is a query point and π is a path of length $|\pi| > \frac{\log^2 n}{2}$, all regions containing q along π can be reported in time $O(|\pi|)$.*

Proof. We combine the classical heavy path decomposition by Sleator and Tarjan [ST83] and the data structure for catalog paths to achieve the desire query time. We first apply the heavy path decomposition to the tree and then for every heavy path created we build a 2D OFC data structure to answer queries along the path. Clearly, we only spend linear space in total. Then by the property of the heavy path decomposition, we only need to query $O(\log n)$ heavy paths to answer a query, which leads to a query time of $O(\log^2 n + |\pi|) = O(|\pi|)$. $\square$

By combining Lemma 11.4.1, Lemma 11.4.6, Lemma 11.4.8, we immediately get the following corollary.

Corollary 11.4.1. *Consider a catalog tree in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n\alpha_c(n))$ space, where $c \geq 3$ is any constant and $\alpha_c(n)$ is the c-th function of the inverse Ackermann hierarchy, such that given any query (q, π), where q is a query point and π is a path, all regions containing q along π can be reported in time $O(\log n + |\pi| + \min\{|\pi|\sqrt{\log n}, \sqrt{|\pi|}\log n\})$. Furthermore, we can also build a data structure using $O(n)$ space answering queries in time $O(\log n + |\pi| + \min\{|\pi|\sqrt{\log n}, \alpha(n)\sqrt{|\pi|}\log n\})$, where $\alpha(n)$ is the inverse Ackermann function.*

11.5 Queries on Catalog Graphs and Subgraph Queries

In this section, we consider general catalog graphs as well as subgraph queries on catalog trees. Our result shows that it is possible to build a data structure of space $O(n)$ such that we can save a $\sqrt{\log n}$ factor from the naïve query time of iterative point locations. We also present a matching lower bound.

We begin by presenting a basic reduction.

Lemma 11.5.1. *Given a catalog graph G of m vertices with graph subdivision complexity n and maximum degree d, we can generate a new catalog graph G' with $\Theta(md)$ vertices with graph subdivision complexity $\Theta(n)$ and bounded degree $O(d^2)$ such that the following holds: given any connected subgraph $\pi \subseteq G$, in time $O(|\pi|)$, we can find a path π' in G' such that the answer to any query $Q_1 = (q, \pi)$ in G equals the answer to query $Q_2 = (q, \pi')$ in G'.*

Proof. The main idea is that we can add a number of dummy vertices to the graph such that we can turn a subgraph query to a path query.

We can obtain G' in the following way. For every vertex in G, place $2d$ copies of the vertex in G'. All the copies of a vertex are connected in G'. Furthermore, every copy of a vertex v_i is connected to every copy of vertex v_j if and only if v_i and v_j are connected in G. The maximum degree of G' is thus $O(d^2)$.

Now consider a subraph query π in G. By definition, π is a connected subgraph of G and w.l.o.g., we can assume π is a tree. We can form a walk W from π by following a DFS ordering of π such that W traverses every edge of π at most twice and visits every vertex of π. We observe that we can realize W as a path in G' by utilizing the dummy vertices; as each vertex has $2d$ dummy vertices, every visit to a vertex in G can be replaced by a visit to a distinct dummy vertex. □

The Upper Bound

Formally, we have the following result.

Lemma 11.5.2. *Consider a degree-bounded catalog graph in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n)$ space such that given any query (q,π), where q is a query point and π is a path, all regions containing q along π can be reported in time $O(\log n+|\pi|\sqrt{\log n})$.*

Proof. We build the data structure essentially the same way as in the proof of Lemma 11.4.1. The only difference is that the degree of a node is bounded by a $d \geq 2$ which can be any constant. By the same argument in the proof of Lemma 11.4.1, any node is stored at most $\Theta(r^{\log d}\log r)$ times and we can obtain a $O(n)$ space bounded data structure achieving $O(\log n+|\pi|\sqrt{\log n})$ query time by creating an intersection sensitive $(r^{2\log d}/n_i)$-cutting for each planar subdivision A_i and balancing the query time of cutting cells and conflict lists. □

By Lemma 11.5.1, we can also obtain the following two corollaries.

Corollary 11.5.1. *Consider a catalog graph in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n)$ space such that given any query (q,π), where q is a query point and π is a connected subgraph, all regions containing q along π can be reported in time $O(\log n+|\pi|\sqrt{\log n})$.*

Specifically, for catalog trees we have the following:

Corollary 11.5.2. *Consider a catalog tree in which each vertex is associated with a planar subdivision. Let n be the total complexity of the subdivisions. We can build a data structure using $O(n)$ space such that given any query (q,π), where q is a query point and π is a subtree, all regions containing q along π can be reported in time $O(\log n+|\pi|\sqrt{\log n})$.*

The Lower Bound

In this section, we show that the $\sqrt{\log n}$ factor that exists in Lemma 11.5.2, Corollary 11.5.1, and Corollary 11.5.2 is tight. Like the proof of path queries for catalog trees, we need a reduction from a rectangle stabbing problem to a 2D OFC subtree query problem on catalog trees. But unlike previous proofs, we use an instance of the rectangle stabbing problem in a much higher dimension.

We show the lower bound for subtree queries of a catalog tree. By Lemma 11.5.1, this also gives a lower bound for path queries in general catalog graphs.

Lemma 11.5.3. *Assume, given any catalog tree of height $\sqrt{\log n} \leq h \leq \frac{\log n}{2}$ in which each vertex is associated with a planar subdivision with n being the total complexity of the subdivisions, we can build a data structure that satisfies the following: it uses at most $n2^{\varepsilon\sqrt{\log n}}$ space, for a small enough constant ε, and it can answer 2D OFC queries (q,π), where q is a query point and π is a subtree containing $b=\sqrt{n}/\log n$ leaves. Then, its query time must be $\Omega(|\pi|\sqrt{\log n})$.*

Proof. We define the following special $(2+b)$-dimensional rectangle stabbing problem. The input consists of n rectangles in $(2+b)$ dimensions. According to their shapes, rectangles are divided into $h' = h - r$ sets of size n/h' each where r is a parameter to be determined later. The rectangles in each set are further divided into b groups of size $n/(h'b)$ each. All the rectangles in the same group are pairwise disjoint and they together tile the $(2+b)$-dimensional unit cube. We put restrictions on the shapes of the input rectangles to make this problem special. For a rectangle in set i, $i = 0, 1, \cdots, h'-1$, group j, $j = 0, 1, \cdots, b-1$, except for the first two and the $(2+j)$-th dimensions, its other side lengths are all set to be 1. The side length of the $(2+j)$-th dimension is set to be $1/2^{i+r}$. We put restriction on the side lengths of the first two dimensions in set i as follows: First for the first group $j = 0$ in this set, we put no restrictions of the first two dimensions as long as they tile the unit cube and the total number of rectangles used for this group is $n/(h'b)$. For an arbitrary group j, we cut the range of the unit cube in the $(2+j)$-th dimension into 2^{i+r} equal length pieces. This partitions the unit cube into 2^{i+r} parts. Note that each part of the unit cube is also tiled by rectangles since we require the side length of the $(2+j)$-th dimension of the rectangles to be $1/2^{i+r}$. If we project the rectangles in each part of the unit cube into the first two dimensions, we obtain 2^{i+r} axis-aligned planar subdivisions. The planar subdivisions we generated for the first group is used as a blueprint for the shape of rectangles in other groups. More specifically, for the remaining groups in set i, we require that the choices of the first two dimensions to give the same set of 2^{i+r} planar subdivisions as the first group. The problem is as follows: Given a point in $(2+b)$-dimensions, find all the rectangles containing this query point.

We now describe a reduction from this problem to 2D OFC subtree queries on catalog trees. We consider a complete balanced binary tree of height $h = h' + r$. Note that the number of nodes at layer $i+r$ of the tree is the same as the number of different subdivisions we get by projecting a group in set i to the first two dimensions. Since we require all the groups in the same set yield the same set of subdivisions, we can simply attach the subdivisions to the nodes starting from layer r. For nodes in layer smaller than r, we attach them with empty subdivisions.

Now let us analyze a rectangle stabbing query q on the rectangle stabbing problem. Consider rectangles in set i, we need to find the rectangle containing q in each of the b groups. In this special rectangle stabbing problem, to find the rectangle in group j containing q, we can find the rectangle by first use the $(2+j)$-th coordinate of q to find the part of the unit cube where q is in, and then find the output rectangle by a simple planar point location on the projection of the part using the first two coordinates of q. By our construction this is equivalent to choose a node in layer $i+r$ of the binary tree and to perform a point location query on the subdivision attached to it. Note that the node in layer $i+r+1$ we choose must be one of the children of the chosen node in layer $i+r$. So if we only focus on one specific group j of all sets, the rectangle stabbing query corresponds to a series of point location queries from the root to a leaf in the binary tree we constructed. Similarly, we obtain b such paths if we consider all groups and they together form a subtree of b leaves. The answer to the point location queries along the subtree gives the answer to the rectangle stabbing problem.

We describe a hard high dimensional rectangle stabbing problem instance. As before, we create rectangles of different shapes to tile the unit cube. But this time, we will consider a $(2+b)$-dimensional rectangle stabbing problem. For rectangles in class $i = 0, \cdots, h'/r - 1$, supergroup $j = 0, \cdots, r-1$, we create the following shapes:

$$\left.\begin{array}{ll}
[\frac{1}{K^j} \times K^j \cdot 2^{ir+j+r} \cdot V & \times \frac{1}{2^{ir+j+r}} \times 1 \times 1 \times \cdots \times 1] \\
[\frac{1}{K^j} \times K^j \cdot 2^{ir+j+r} \cdot V & \times 1 \times \frac{1}{2^{ir+j+r}} \times 1 \times \cdots \times 1] \\
 & \vdots \\
[\frac{1}{K^j} \times K^j \cdot 2^{ir+j+r} \cdot V & \times 1 \times 1 \times \cdots \times 1 \times \frac{1}{2^{ir+j+r}}]
\end{array}\right\} b \text{ shapes}$$

where K, V are parameters to be determined later. Note that all the rectangles are in $(2+b)$ dimensions.

We use each of the shape to tile a unit cube. Since the volume of any rectangle is V, we need $1/V$ rectangles of the same shape to tile the cube. We call it a group. Note that the rectangles in the same group are pairwise disjoint. We generated $h'b/V$ rectangles in total. By setting $V = h'b/n$, the total number of rectangles is n. Note that any point in the unit cube is contained in exactly $t = h'b$ rectangles.

Now we shall analyze the volume of the intersection between any two $(2+b)$-dimensional rectangles. Note that if two rectangles have the same side lengths for $b-1$ out of the last b dimensions, then it is the case we have analyzed in the proof of, e.g., Lemma 11.4.2, and the volume of the intersection of any two rectangles is bounded by V/K if we set $K = 2^r$. Now we analyze the other case. By our construction, two rectangles can only have at most two different side lengths in the last b dimensions. We consider two rectangles in class i_1 supergroup j_1, and class i_2 supergroup j_2 respectively. Without loss of generality, we assume $j_1 \geq j_2$. The case for $j_1 \leq j_2$ is symmetric. Then there are two possible expressions for the intersection volume depending on the values of i_1 and i_2. The first one is

$$\frac{1}{K^{j_1}} \times K^{j_1} \cdot 2^{i_1 r + j_1 + r} \cdot V \times \frac{1}{2^{i_1 r + j_1 + r}} \times \frac{1}{2^{i_2 r + j_2 + r}} = \frac{V}{2^{i_2 r + j_2 + r}} \leq \frac{V}{K}.$$

The second possible expression is

$$\frac{1}{K^{j_1}} \times K^{j_2} \cdot 2^{i_2 r + j_2 + r} \cdot V \times \frac{1}{2^{i_1 r + j_1 + r}} \times \frac{1}{2^{i_2 r + j_2 + r}} = \frac{V}{K^{j_1 - j_2}} \times \frac{1}{2^{i_1 r + j_1 + r}} \leq \frac{V}{K}.$$

The last inequality holds because $j_1 \geq j_2$.

To make this construction well-defined, no side length of the rectangles can exceed 1. The largest side length can only be obtained in the second dimension when $i = h'/r - 1$ and $j = r - 1$. We must have

$$K^{r-1} 2^{h'+r-1} V \leq 1.$$

By plugging in the values $V = h'b/n$ and $K = 2^r$ we get that we must have

$$2^{r^2 - r} 2^{h'+r-1} h'b < n \tag{11.5}$$

Since by our assumptions $h' \le \frac{\log n}{2}$, $b \le \frac{\sqrt{n}}{\log n}$, it follows that by setting $r = \frac{\sqrt{\log n}}{4}$, the inequality (11.5) holds.

If $\gamma h' b \ge Q(n)$ holds, then the first condition of Theorem 11.2.3 is satisfied and we obtain the lower bound of

$$S(n) = \Omega(\frac{tv^{-1}}{2^{O(\gamma)}}) = \Omega(\frac{n2^r}{2^{O(\gamma)}}).$$

Now if we set $\gamma = \delta\sqrt{\log n}$ for a sufficiently small $\delta > 0$, the data structure must use $\Omega(n2^{\Omega(\sqrt{\log n})})$ space, which contradicts the space usage stated in our lemma. Note that $|\pi| = (h'+r)b = \Theta(h'b)$. Then $Q(n) \ge \gamma h' b = \Omega(|\pi|\sqrt{\log n})$. □

Remark 11.5.1. *Note that the lower bound holds even when the query path is of length* $\ge \sqrt{\log n}$ *and* $\le \frac{\log n}{2}$. *We have already established this lower bound in Lemma 11.4.2.*

Combining Lemma 11.5.1 and Lemma 11.5.3, we immediately have the following corollary:

Corollary 11.5.3. *Assume, given any degree-bounded catalog graph in which each vertex is associated with a planar subdivision with n being the total complexity of the subdivisions, we can build a data structure that satisfies the following: it uses at most* $n2^{\varepsilon\sqrt{\log n}}$ *space, for a small enough constant* ε*, and it can answer 2D OFC queries* (q, π)*, where q is a query point and* π *is a path. Then, its query time must be* $\Omega(|\pi|\sqrt{\log n})$.

11.6 Open Problems

For the linear space data structure we obtained for general path queries of trees Corollary 11.4.1, there is a tiny inverse Ackermann gap between the query time we obtain and the lower bound. It is an interesting problem whether we can get rid of that term or improve the lower bound.

The problem we consider is very general in the sense that the only restriction we place on the input instance is that the graph subdivision complexity is n. Some special cases admit better solutions. For example, if we require the subdivision complexity of each vertex of the graph to be asymptotically the same, we can obtain an $O(n)$ space and $O(\log n + |\pi| \log\log n)$ query time data structure for path queries on catalog trees of height $\le \frac{\log n}{2}$, while we can only achieve $O(\log n + |\pi|\sqrt{\log n})$ query time given linear space in the general case Lemma 11.4.1. This is done by creating an intersection sensitive $(\log n/n_i)$-cutting C_i for each subdivision A_i in the tree and then storing all cutting cells on each path using the data structure in Theorem 11.3.1 and building point location data structures on the conflict list of each cell.

Higher dimensional generalization of our results is another direction. In 2D, we can transform an axis-aligned planar subdivision to a subdivision consisting of only rectangles by increasing the subdivision complexity by only a constant factor; however it is not the case for 3D. On the other hand, for 3D point locations on orthogonal subdivisions, we have Rahul's $O(\log^{3/2} n)$ query time and linear space data structure [Rah15] in the standard pointer machine model. Recently, the query time is improved to $O(\log n)$ by Chan et al. [CNRT22], but they use a stronger arithmetic pointer machine model. Given that the higher dimensional counterparts of the tools we use for 2D are suboptimal, it is a challenging and interesting problem to see how the results will be in higher dimensions.

Other open problems include considering the dynamization of our results, i.e., to support insertion and deletion dynamically, and other computational models, e.g., RAM and I/O model.

Appendices

11.A Proof of Theorem 11.2.3

Theorem 11.2.3. *Assume, we have an algorithm that given any input instance $R \subset \mathbb{R}$ of n ranges, it can store R in a data structure of size $S(n)$ such that given any query $q \in U$, it can answer the query in $Q(n) + \gamma|R_q|$ time.*

Then, suppose we can construct an input set $R \subset \mathbb{R}$ of n ranges such that the following two conditions are satisfied: (i) every query point $q \in U$ is contained in exactly $|R_q| = t$ ranges and $\gamma t \geq Q(n)$; (ii) there exists a value v such that for any two ranges $r_1, r_2 \in R$, $\mu(\{q \in U | r_1, r_2 \in R_q\})$ is well-defined and is upper bounded by v. Then, we must have $S(n) = \Omega(tv^{-1}/2^{O(\gamma)}) = \Omega(Q(n)v^{-1}/2^{O(\gamma)})$.

To prove this theorem, we first show a special property of the subgraph M_q explored to answer a query $q \in U$. Note that in the pointer machine model, we begin the exploration with a special cell, called the root. If we consider only the first in-edge to any cell in M_q, we obtain a tree.

Lemma 11.A.1. *Let M_q be the explored subgraph corresponds to a query $q \in U$. We call the memory cells in M_q containing reported ranges marked cells. Let a fork be a subtree of M_q of size at most $c\gamma$ containing two marked cells, where c is a large enough constant and γ is the parameter in Theorem 11.2.3. Then M_q can be decomposed into $\Omega(|R_q|)$ many forks, where R_q is the set of ranges containing q.*

Proof. The proof we present is very similar to the one described in [Afs13]. We generate the forks using the following method. For every cell in M_q, we assign two values mark and size to it. For marked cell, we initialize its mark value to be one. Other cells will have mark value zero. For any cell in M_q, we assign one to its size value. Without loss of generality, we assume M_q to be a tree. At every step, we choose an arbitrary leaf and add its mark value and size value to the corresponding values of its parent. Then we remove this cell. If its parent has another child, we repeat this process until its parent becomes a leaf. If after this process its parent has mark value two and size value more than $c\gamma$, we remove its parent as well and do nothing. We call this situation "wasted". If its parent has mark value two and size value no more than $c\gamma$, we find a fork. We add it to the fork set and remove the subtree. If its parent has mark value less than two, we do nothing. Note that its parent cannot have mark value more than two because that will indicate one of its child has mark value at least two but not being added to a fork or wasted. We go on to the next step until reaching the root.

Let us consider how many marks will be wasted. We only waste marks when we find a subtree containing two marks but of size more than $c\gamma$ and when we reach the root with only one mark. Since M_q contains $Q(n) + \gamma|R_q| \leq 2\gamma|R_q|$ cells, the number of marks wasted is bounded by $4\gamma|R_q|/(c\gamma) + 1 = 4|R_q|/c + 1$. Other marks are all stored in forks, so the number of forks is more than $(|R_q| - 4|R_q|/c - 1)/2 = \Omega(|R_q|)$ for a sufficiently large c. □

We also need another lemma, which follows directed from Lemma 1 in Afshani [Afs13].

Lemma 11.A.2. *The number of forks of size $O(\gamma)$ is $O(S(n)2^{O(\gamma)})$.*

Now we prove Theorem 11.2.3.

Proof. Consider any query point $q \in U$. By definition, it is contained in a set R_q of ranges. Consider the explored subgraph M_q when answering q. By Lemma 11.A.1, we can decompose M_q into a set F_q of $\Omega(|R_q|)$ forks such that each fork contains two output ranges. Note that for the two ranges to be output, q must lie in the intersection of the two ranges. Similarly, q must lie in all the intersection of the two ranges for every fork in M_q. This implies that q is covered by these intersections $\Omega(|R_q|)$ times.

Since we can answer queries for all $q \in U$ and by assumption (i) each q is contained in t ranges, it implies that the intersections of two ranges in all possible forks cover U $\Omega(t)$ times. By Lemma 11.A.2, the number of possible forks of size $O(\gamma)$ is $O(S(n)2^{O(\gamma)})$. Each fork has $\binom{O(\gamma)}{2} = O(\gamma^2)$ ways to choose two ranges. By assumption (ii), the measure of any two ranges is bounded by v. So by a simple measure argument,

$$O(\gamma^2)S(n)2^{O(\gamma)}v = \Omega(t).$$

This gives us

$$S(n) = \Omega(\frac{t}{v2^{O(\gamma)}}).$$

By our assumption (i), $\gamma t \geq Q(n)$, we also obtain

$$S(n) = \Omega(\frac{Q(n)}{v2^{O(\gamma)}}).$$

□

Bibliography

[AAE$^+$22] Pankaj K. Agarwal, Boris Aronov, Esther Ezra, Matthew J. Katz, and Micha Sharir. Intersection queries for flat semi-algebraic objects in three dimensions and related problems. In *38th International Symposium on Computational Geometry*, volume 224 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 4, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022. 34, 137, 138, 139, 153

[AAEZ21] Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl. Efficient algorithm for generalized polynomial partitioning and its applications. *SIAM J. Comput.*, 50(2):760–787, 2021. 19, 24, 26, 31, 33, 34, 75, 76, 77, 104, 105, 106, 138

[AAL10] Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting: query lower bounds, optimal structures in 3-d, and higher-dimensional improvements. In *Computational geometry (SCG'10)*, pages 240–246. ACM, New York, 2010. 5, 200

[AAL12] Peyman Afshani, Lars Arge, and Kasper Green Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *Computational geometry (SCG'12)*, pages 323–332. ACM, New York, 2012. 5, 53, 54, 56, 194, 195, 198, 200, 203

[ABR00] Stephen Alstrup, Gerth Stø lting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 198–207. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000. 48

[AC07] Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. In *Computational geometry (SCG'07)*, pages 337–343. ACM, New York, 2007. 44

[AC09] Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Twentieth Annual*

ACM-SIAM Symposium on Discrete Algorithms, pages 180–186. SIAM, Philadelphia, PA, 2009. 6, 167

[AC20] Peyman Afshani and Pingan Cheng. 2D generalization of fractional cascading on axis-aligned planar subdivisions. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science—FOCS 2020*, pages 716–727. IEEE Computer Soc., Los Alamitos, CA, [2020] ©2020. 10, 53, 54, 56, 58, 60

[AC22] Peyman Afshani and Pingan Cheng. On semialgebraic range reporting. In *38th International Symposium on Computational Geometry*, volume 224 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 3, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022. 10, 23, 24, 137, 138, 139, 140, 141, 144, 152, 153

[AC23a] Peyman Afshani and Pingan Cheng. Lower bounds for intersection reporting among flat objects. In *39th International Symposium on Computational Geometry*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Paper No. 3, 16. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2023. 11, 35

[AC23b] Peyman Afshani and Pingan Cheng. Lower bounds for semialgebraic range searching and stabbing problems. *J. ACM*, 70(2):Art. 16, 26, 2023. 10, 23, 103, 104, 105, 106, 107, 108, 109, 115, 118, 119, 137, 138, 139, 140, 141, 153

[AC23c] Peyman Afshani and Pingan Cheng. An optimal lower bound for simplex range reporting. In *2023 Symposium on Simplicity in Algorithms (SOSA)*, pages 272–277. SIAM, Philadelphia, PA, 2023. 10, 22, 23, 73

[ACBRW23] Peyman Afshani, Pingan Cheng, Aniket Basu Roy, and Zhewei Wei. On range summary queries. In *50th International Colloquium on Automata, Languages, and Programming*, volume 261 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Paper No. 7, 17. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2023. 11, 45

[ACH$^+$13] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):Art. 26, 28, 2013. 41, 42, 46, 157, 160, 162, 163

[ACT14] Peyman Afshani, Timothy M. Chan, and Konstantinos Tsakalidis. Deterministic rectangle enclosure and offline dominance reporting on the RAM. In *Automata, languages, and programming. Part I*, volume 8572 of *Lecture Notes in Comput. Sci.*, pages 77–88. Springer, Heidelberg, 2014. 194, 195

[AD18] Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 898–917. SIAM, Philadelphia, PA, 2018. 74

[AdBG08] Boris Aronov, Mark de Berg, and Chris Gray. Ray shooting and intersection searching amidst fat convex polyhedra in 3-space. *Comput. Geom.*, 41(1-2):68–76, 2008. 137

[Afs13] Peyman Afshani. Improved pointer machine and I/O lower bounds for simplex range reporting and related problems. *Internat. J. Comput. Geom. Appl.*, 23(4-5):233–251, 2013. 6, 22, 23, 58, 63, 64, 65, 73, 77, 78, 106, 138, 199, 217, 218

[Afs19] Peyman Afshani. A new lower bound for semigroup orthogonal range searching. In *35th International Symposium on Computational Geometry*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 3, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019. 5, 106

[Afs21] Peyman Afshani. A lower bound for dynamic fractional cascading. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2229–2248. [Society for Industrial and Applied Mathematics (SIAM)], Philadelphia, PA, 2021. 51

[Aga90] Pankaj K. Agarwal. Partitioning arrangements of lines. II. Applications. *Discrete Comput. Geom.*, 5(6):533–573, 1990. 18, 198

[Aga91] Pankaj K. Agarwal. Geometric partitioning and its applications. In *Discrete and computational geometry (New Brunswick, NJ, 1989/1990)*, volume 6 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 1–37. Amer. Math. Soc., Providence, RI, 1991. 18, 178, 198

[Aga16] Pankaj K. Agarwal. Range searching. In J. E. Goodman, J. O'Rourke, and C. Toth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., 2016. 4

[Aga17] Pankaj K. Agarwal. Simplex range searching and its variants: a review. In *A journey through discrete mathematics*, pages 1–30. Springer, Cham, 2017. 64, 75, 116, 138, 160

[AHZ10] Peyman Afshani, Chris Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Comput. Geom.*, 43(8):700–712, 2010. 44, 163, 178, 179

[AM93] Pankaj K. Agarwal and Jiří Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993. 137

[AM94] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11(4):393–418, 1994. 7, 8, 31, 74, 77, 138

[AMM06] Sunil Arya, Theocharis Malamatos, and David M. Mount. On the importance of idempotence. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 564–573. ACM, New York, 2006. 106

[AMS13] Pankaj K. Agarwal, Jiří Matoušek, and Micha Sharir. On range searching with semialgebraic sets. II. *SIAM J. Comput.*, 42(6):2039–2062, 2013. 7, 16, 31, 33, 71, 74, 75, 77, 103, 105, 138

[AMX12] Sunil Arya, David M. Mount, and Jian Xia. Tight lower bounds for halfspace range searching. *Discrete Comput. Geom.*, 47(4):711–730, 2012. 6, 106

[AP19] Peyman Afshani and Jeff M. Phillips. Independent range sampling, revisited again. In *35th International Symposium on Computational Geometry*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 4, 13. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019. 42, 161

[AS96] Pankaj K. Agarwal and Micha Sharir. Ray shooting amidst convex polyhedra and polyhedral terrains in three dimensions. *SIAM J. Comput.*, 25(1):100–116, 1996. 137

[AS07] Roel Apfelbaum and Micha Sharir. Large complete bipartite subgraphs in incidence graphs of points and hyperplanes. *SIAM J. Discrete Math.*, 21(3):707–725, 2007. 66

[ASTW14] Peyman Afshani, Cheng Sheng, Yufei Tao, and Bryan T. Wilkinson. Concurrent range reporting in two-dimensional space. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 983–994. ACM, New York, 2014. 196

[AT18] Peyman Afshani and Konstantinos Tsakalidis. Optimal deterministic shallow cuttings for 3-d dominance ranges. *Algorithmica*, 80(11):3192–3206, 2018. 179

[Avi84] David Avis. Non-partitionable point sets. *Inf. Process. Lett.*, 19(3):125–129, 1984. 64

[AW17] Peyman Afshani and Zhewei Wei. Independent range sampling, revisited. In *25th European Symposium on Algorithms*, volume 87 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 3, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017. 42, 46, 157, 161, 162, 163

[BCP93] Hervé Brönnimann, Bernard Chazelle, and János Pach. How hard is half-space range searching? *Discrete Comput. Geom.*, 10(2):143–155, 1993. 106

[BCV19] Martin Balko, Josef Cibulka, and Pavel Valtr. Covering lattice points by subspaces and counting point-hyperplane incidences. *Discrete Comput. Geom.*, 61(2):325–354, 2019. 66

[Ben79] Jon Louis Bentley. Decomposable searching problems. *Inf. Process. Lett.*, 8(5):244–251, 1979. 5, 198

[BGJrS11] Gerth Stø lting Brodal, Beat Gfeller, Allan Grø nlund Jø rgensen, and Peter Sanders. Towards optimal range medians. *Theoret. Comput. Sci.*, 412(24):2588–2601, 2011. 39, 160

[BGM$^+$21] Djamal Belazzougui, Travis Gagie, J. Ian Munro, Gonzalo Navarro, and Yakov Nekrich. Range majorities and minorities in arrays. *Algorithmica*, 83(6):1707–1733, 2021. 161

[BGN13] Djamal Belazzougui, Travis Gagie, and Gonzalo Navarro. Better space bounds for parameterized range majority and minority. In *Algorithms and data structures*, volume 8037 of *Lecture Notes in Comput. Sci.*, pages 121–132. Springer, Heidelberg, 2013. 161

[BK03] Peter Braß and Christian Knauer. On counting point-hyperplane incidences. *Comput. Geom.*, 25(1-2):13–20, 2003. 66

[BKMT05] Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Approximate range mode and range median queries. In *STACS 2005*, volume 3404 of *Lecture Notes in Comput. Sci.*, pages 377–388. Springer, Berlin, 2005. 39, 160

[CEGea94] B. Chazelle, H. Edelsbrunner, M. Grigni, and et al. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994. 195

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990. 18, 73, 178, 198

[CG86a] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986. 9, 49, 50, 193, 194

[CG86b] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: II. applications. *Algorithmica*, 1(2):163–191, 1986. 10, 49, 50, 51, 194, 196

[Cha86] Bernard Chazelle. Filtering search: a new approach to query-answering. *SIAM J. Comput.*, 15(3):703–724, 1986. 4, 5, 195, 198

[Cha88] Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17(3):427–462, 1988. 5, 20

[Cha89] Bernard Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2(4):637–666, 1989. 5, 6, 64, 73, 106, 138

[Cha90a] Bernard Chazelle. Lower bounds for orthogonal range searching. I. The reporting case. *J. Assoc. Comput. Mach.*, 37(2):200–212, 1990. 4, 5, 20, 21, 65, 66, 69, 77, 78, 106, 107, 119, 140

[Cha90b] Bernard Chazelle. Lower bounds for orthogonal range searching. II. The arithmetic model. *J. Assoc. Comput. Mach.*, 37(3):439–463, 1990. 5, 106

[Cha93] Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discret. Comput. Geom.*, 9:145–158, 1993. 6, 18, 64, 198

[Cha00] Bernard Chazelle. *The discrepancy method*. Cambridge University Press, Cambridge, 2000. Randomness and complexity. 69

[Cha12] Timothy M. Chan. Optimal partition trees. *Discrete Comput. Geom.*, 47(4):661–690, 2012. 6, 15, 22, 43, 63, 64, 71, 73, 74, 138, 177

[Cha18] Bernard Chazelle. Cuttings. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2018. Second edition. 73, 138, 178

[CHN20] Timothy M. Chan, Qizheng He, and Yakov Nekrich. Further results on colored range searching. In *36th International Symposium on Computational Geometry*, volume 164 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 28, 15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020. 161

[CL04] Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *J. Comput. System Sci.*, 68(2):269–284, 2004. 10, 52, 53, 65, 193, 194, 195

[Cla87] Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2(2):195–222, 1987. 18, 73, 178

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015. An introduction to computational algebraic geometry and commutative algebra. 7

[CLP11] Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Computational geometry (SCG'11)*, pages 1–10. ACM, New York, 2011. 44, 163, 200, 203

[CLRS22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022. 195, 200, 207

[CNRT22] Timothy M. Chan, Yakov Nekrich, Saladi Rahul, and Konstantinos Tsakalidis. Orthogonal point location and rectangle stabbing queries in 3-D. *J. Comput. Geom.*, 13(1):399–428, 2022. 216

[Col85] Richard Cole. Partitioning point sets in 4 dimensions. In *Automata, languages and programming (Nafplion, 1985)*, volume 194 of *Lecture Notes in Comput. Sci.*, pages 111–119. Springer, Berlin, 1985. 64

[CR96] Bernard Chazelle and Burton Rosenberg. Simplex range reporting on a pointer machine. *Comput. Geom.*, 5(5):237–247, 1996. 4, 20, 21, 22, 63, 64, 65, 66, 73, 77, 78, 106, 107, 119, 138, 140

[CSW92] Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8(5&6):407–429, 1992. 15, 64

[CW89] Bernard Chazelle and Emo Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4(5):467–489, 1989. 15, 64, 73

[CW16] Timothy M. Chan and Bryan T. Wilkinson. Adaptive and approximate orthogonal range counting. *ACM Trans. Algorithms*, 12(4):Art. 45, 15, 2016. 5

[CZ15] Timothy M. Chan and Gelin Zhou. Multidimensional range selection. In *Algorithms and computation*, volume 9472 of *Lecture Notes in Comput. Sci.*, pages 83–92. Springer, Heidelberg, 2015. 160

[CZ22] Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2D fractional cascading, and decision trees. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210. [Society for Industrial and Applied Mathematics (SIAM)], Philadelphia, PA, 2022. 53

[CZ23] Timothy M. Chan and Da Wei Zheng. Simplex range searching revisited: how to shave logs in multi-level data structures. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1493–1511. SIAM, Philadelphia, PA, 2023. 19, 71, 73

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. 6, 44, 49, 54, 75, 178, 197, 199

[dBG08] Mark de Berg and Chris Gray. Vertical ray shooting and computing depth orders for fat objects. *SIAM J. Comput.*, 38(1):257–275, 2008. 138

[dBHO$^+$94] M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12(1):30–53, 1994. 137

[dBS95] Mark de Berg and Otfried Schwarzkopf. Cuttings and applications. *Int. J. Comput. Geom. Appl.*, 5(4):343–355, 1995. 55, 64, 198, 200

[dBvKS95] Mark de Berg, Marc J. van Kreveld, and Jack Snoeyink. Two- and three-dimensional point location in rectangular subdivisions. *J. Algorithms*, 18(2):256–277, 1995. 194

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. 39

[DHM$^+$13] Stephane Durocher, Meng He, J. Ian Munro, Patrick K. Nicholson, and Matthew Skala. Range majority in constant time and linear space. *Inform. and Comput.*, 222:169–179, 2013. 161

[DR91] Paul F. Dietz and Rajeev Raman. Persistence, amortization and randomization. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1991)*, pages 78–88. ACM, New York, 1991. 194

[Dvi09] Zeev Dvir. On the size of Kakeya sets in finite fields. *J. Amer. Math. Soc.*, 22(4):1093–1097, 2009. 74

[ES22a] Esther Ezra and Micha Sharir. Intersection searching amid tetrahedra in 4-space and efficient continuous collision detection. In *30th annual*

European Symposium on Algorithms, volume 244 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 51, 17. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022. 34, 35, 137, 139

[ES22b] Esther Ezra and Micha Sharir. On ray shooting for triangles in 3-space and related problems. *SIAM J. Comput.*, 51(4):1065–1095, 2022. 8, 34, 137, 138, 139, 153

[EW86] Herbert Edelsbrunner and Emo Welzl. Halfplanar range search in linear space and o(n^(0.695)) query time. *Inf. Process. Lett.*, 23(6):289–293, 1986. 15, 64

[FC20] Luciano Floridi and Massimo Chiriatti. GPT-3: its nature, scope, limits, and consequences. *Minds Mach.*, 30(4):681–694, 2020. 39

[GJS95] Prosenjit Gupta, Ravi Janardan, and Michiel Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization. *J. Algorithms*, 19(2):282–317, 1995. 161

[GK09] Yoav Giora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Trans. Algorithms*, 5(3):Art. 28, 51, 2009. 194

[GK15] Larry Guth and Nets Hawk Katz. On the erdős distinct distances problem in the plane. *Ann. of Math. (2)*, 181(1):155–190, 2015. 7, 16, 74, 105, 138

[GOT18] Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors. *Handbook of discrete and computational geometry*. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, 2018. Third edition of [MR1730156]. 72, 105, 197

[Gut15] Larry Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Cambridge Philos. Soc.*, 159(3):459–469, 2015. 19, 34, 75, 138

[HQT14] Xiaocheng Hu, Miao Qiao, and Yufei Tao. Independent range sampling. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 246–255. ACM, 2014. 42, 161

[HW87] David Haussler and Emo Welzl. epsilon-nets and simplex range queries. *Discret. Comput. Geom.*, 2:127–151, 1987. 15, 18, 64, 73, 178

[HY17] Zengfeng Huang and Ke Yi. The communication complexity of distributed epsilon-approximations. *SIAM J. Comput.*, 46(4):1370–1394, 2017. 160, 163

[Jac41] C. G. J. Jacobi. De functionibus alternantibus earumque divisione per productum e differentiis elementorum conflatum. *J. Reine Angew. Math.*, 22:360–371, 1841.

[JrL11] Allan Grø nlund Jø rgensen and Kasper Green Larsen. Range selection and median: tight cell probe lower bounds and adaptive data structures. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–813. SIAM, Philadelphia, PA, 2011. 39, 160

[KMS12] Haim Kaplan, Jiří Matoušek, and Micha Sharir. Simple proofs of classical theorems in discrete geometry via the Guth-Katz polynomial partitioning technique. *Discrete Comput. Geom.*, 48(3):499–517, 2012. 16

[KN08] Marek Karpinski and Yakov Nekrich. Searching for frequent colors in rectangles. In *Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008*, 2008. 161

[Knu97] Donald E. Knuth. *The art of computer programming. Vol. 1.* Addison-Wesley, Reading, MA, 1997. Fundamental algorithms, Third edition [of MR0286317]. 20

[KU58] Andrey Nikolaevich Kolmogorov and Vladimir Andreyevich Uspensky. On the definition of an algorithm. *Uspehi Mat. Nauk*, 13:3–28, 1958. English translation in *AMS transl.* II Vol. 29 (1963), 217-245. 20

[Lan93] Serge Lang. *Algebra (3. ed.).* Addison-Wesley, 1993. 3

[Mat90] Jiří Matoušek. Construction of epsilon-nets. *Discret. Comput. Geom.*, 5:427–448, 1990. 18

[Mat91] Jiří Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6(5):385–406, 1991. 18, 64, 73, 198

[Mat92a] Jiří Matoušek. Efficient partition trees. *Discret. Comput. Geom.*, 8:315–334, 1992. 6, 15, 43, 177, 178

[Mat92b] Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2(3):169–186, 1992. 6

[Mat93] Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993. 6, 15, 22, 64, 71, 73, 77, 105, 116, 117, 138, 178

[Mat94] Jiří Matoušek. Geometric range searching. *ACM Comput. Surv.*, 26(4):421–461, 1994. 4, 105

[Mat95] Jiří Matoušek. Approximations and optimal geometric divide-an-conquer. *J. Comput. Syst. Sci.*, 50(2):203–208, 1995. 18, 198

[Mat99] Jiří Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer Berlin, Heidelberg, 1999. 13

[Mat02] Jiří Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002. 43

[Mat10] Jiří Matoušek. *Geometric discrepancy*, volume 18 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2010. An illustrated guide, Revised paperback reprint of the 1999 original. 164

[Meg83] Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach.*, 30(4):852–865, 1983. 43

[MN90] Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(2):215–241, 1990. 51, 194

[MP15] Jiří Matoušek and Zuzana Patáková. Multilevel polynomial partitions and simplified range searching. *Discrete Comput. Geom.*, 54(1):22–41, 2015. 17, 33, 74, 103, 104, 106, 138

[MS93] Jiří Matoušek and Otfried Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10(2):215–232, 1993. 137

[Mut05] S. Muthukrishnan. Data streams: algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, 2005. 39

[MV17] Nabil H. Mustafa and Kasturi R. Varadarajan. Epsilon-approximations and epsilon-nets. *CoRR*, abs/1702.03676, 2017. 46

[Ope23] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. 39

[Pel90] Marco Pellegrini. Stabbing and ray shooting in 3 dimensional space. In Raimund Seidel, editor, *Proceedings of the Sixth Annual Symposium on Computational Geometry, Berkeley, CA, USA, June 6-8, 1990*, pages 177–186. ACM, 1990. 137

[Pel93] M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9(5):471–494, 1993. 137

[Pel17] Marco Pellegrini. Ray shooting and lines in space. In *Handbook of discrete and computational geometry (3rd Edition)*, CRC Press Ser. Discrete Math. Appl., pages 1093–1112. CRC, Boca Raton, FL, 2017. 138

[Phi16] Jeff M. Phillips. Coresets and sketches. *CoRR*, abs/1601.00617, 2016. 164

[PS22] Zuzana Patáková and Micha Sharir. Covering points by hyperplanes and related problems. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 57:1–57:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 66, 67

[Rah15] Saladi Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 200–211. SIAM, Philadelphia, PA, 2015. 194, 195, 200, 216

[Ram99] Edgar A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry (Miami Beach, FL, 1999)*, pages 390–399. ACM, New York, 1999. 137

[RDGF15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 39

[Rot76] K. F. Roth. Developments in Heilbronn's triangle problem. *Advances in Math.*, 22(3):364–385, 1976. 65

[RRS07] Rajeev Raman, Venkatesh Raman, and Srinivasa Rao Satti. Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets. *ACM Trans. Algorithms*, 3(4):Art. 43, 25, 2007. 181

[Sch79] Arnold Schönhage. Storage modification machines. In Klaus Weihrauch, editor, *Theoretical Computer Science, 4th GI-Conference, Aachen, Germany, March 26-28, 1979, Proceedings*, volume 67 of *Lecture Notes in Computer Science*, pages 36–37. Springer, 1979. 20

[Sen95] Sandeep Sen. Fractional cascading revisited. *J. Algorithms*, 19(2):161–172, 1995. 51

[She16] Adam Sheffer. Lower bounds for incidences with hypersurfaces. *Discrete Anal.*, pages Paper No. 16, 14, 2016. 66

[ST83] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. System Sci.*, 26(3):362–391, 1983. 55, 210

[Tao22] Yufei Tao. Algorithmic techniques for independent query sampling. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 129–138. ACM, 2022. 42

[Tar79] Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *J. Comput. System Sci.*, 18(2):110–127, 1979. 20

[VC15] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, Cham, 2015. Reprint of Theor. Probability Appl. **16** (1971), 264–280. 42, 43

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 39

[Wel88] Emo Welzl. Partition trees for triangle counting and other range searching problems. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana, IL, 1988)*, pages 23–33. ACM, New York, 1988. 64, 73

[Wel92] Emo Welzl. On spanning trees with low crossing numbers. In *Data structures and efficient algorithms (Berlin, 1991)*, volume 594 of *Lecture Notes in Comput. Sci.*, pages 233–249. Springer, Berlin, 1992. 15

[Wil82] Dan E. Willard. Polygon retrieval. *SIAM J. Comput.*, 11(1):149–165, 1982. 13, 15, 64, 73

[Wil85] Dan E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.*, 14(1):232–253, 1985. 5

[WY11] Zhewei Wei and Ke Yi. Beyond simple aggregates: indexing for summary queries. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 117–128. ACM, 2011. 157

[Yao83] F. Frances Yao. A 3-space partition and its applications (extended abstract). In David S. Johnson, Ronald Fagin, Michael L. Fredman, David

Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 258–263. ACM, 1983. 15, 64

[YDEP89] F. Frances Yao, David P. Dobkin, Herbert Edelsbrunner, and Michael S. Paterson. Partitioning space for range queries. *SIAM J. Comput.*, 18(2):371–384, 1989. 15, 64

[You01] A. Young. On Quantitative Substitutional Analysis. *Proc. Lond. Math. Soc.*, 33:97–146, 1901. 108

[YWW14] Ke Yi, Lu Wang, and Zhewei Wei. Indexing for summary queries: theory and practice. *ACM Trans. Database Syst.*, 39(1):Art. 2, 39, 2014. 41, 161

[YY85] Andrew Chi-Chih Yao and F. Frances Yao. A general approach to d-dimensional geometric queries (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 163–168. ACM, 1985. 19, 64, 74, 105, 117