On Range Summary Queries

Peyman Afshani
Aarhus University, Denmark

Pingan Cheng
Aarhus University, Denmark

Aniket Basu Roy
Aarhus University, Denmark

Zhewei Wei
Renmin University of China, Beijing, China

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 [23] but a few sub-optimal methods were available in higher dimensions [4, 6].

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.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Computational Geometry, Range Searching, Random Sampling, Geometric Approximation, Data Structures and Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.7

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.03180

Funding *Peyman Afshani, Pingan Cheng*: Supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Research under grant ID DFF−7014−00404.
Aniket Basu Roy: Supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Research under grant ID DFF−7014−00404 and the Independent Research Fund Denmark (DFF) under a Sapere Aude Research Leader grant No. 1051-00106B.
Zhewei Wei: Supported by National Natural Science Foundation of China (No. U2241212).

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis; Article No. 7; pp. 7:1–7:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 $w_1 \le \cdots \le w_k$, a δ-quantile, for $0 \le \delta \le 1$, 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 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.

[1] For $a \le 0$ (resp. $a \ge k$), we define the a-quantile to be the 0-quantile (resp. k-quantile).

1.1 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 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 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 [5] 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 [20, 10, 18]. 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 [10]. For higher dimensions, to the best of our knowledge, the only known technique is to reduce the problem to several range counting instances [10, 13], 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 [9] has been studied.

Data summary queries have also received some amount of attention, especially in the context of geometric queries. Agarwal et al. [6] 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 [17] 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 [17] 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 [24]. 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 $|P \cap \gamma|$ [3, 4, 16]. 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 [3]. 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. [15] 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., [11].

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 [19] studied the problem of finding the most frequent colors in a given (orthogonal) query range. This problem has received further attention in the community [8, 7, 14]. 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.

1.2 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 [6, 4] 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.

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 the full version.

2.1 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 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 $(0, 0, -\infty)$).*

Table 1 Our main results compared with Mergeability-based [6] and Independent Range Sampling (IRS)-based [4] 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	$O(n)$ $O(n)$ $O(n)$ $\boldsymbol{O(n\log_w\frac{1}{\varepsilon})}$	$O(\log n+\frac{1}{\varepsilon}n^{2/3})$ $\tilde{O}(n^{2/3}+\frac{1}{\varepsilon^{3/2}}n^{1/6})$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon})}$	Mergeability-based [6] Monte Carlo [17] IRS-based [4] **New**
3D AHHS Dominance	$O(n)$ $O(n)$ $\boldsymbol{O(n)}$	$O(\log n+\frac{1}{\varepsilon}\log^3 n)$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon})}$	Mergeability-based [6] IRS-based [4] **New†**
3D AQS Halfspace	$O(n)$ $O(n)$ $\boldsymbol{O(n\log^2\frac{1}{\varepsilon}\log_w\frac{1}{\varepsilon})}$	$O(\log n+\frac{1}{\varepsilon}n^{2/3}\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})}$	Mergeability-based [6] IRS-based [4] **New**
3D AQS Dominance	$O(n)$ $O(n)$ $\boldsymbol{O(n)}$	$O(\log n+\frac{1}{\varepsilon}\log^3 n\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon})}$	Mergeability-based [6] IRS-based [4] **New†**

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 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. [12], dominance queries can be simulated by a halfspace queries and thus Lemma 1 applies to them. See the full version for details.

We obtain the approximate version of the range counting result using shallow cuttings.

▶ **Theorem 2** (Approximate Range Counting [2]). *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 $\alpha>0$.*

2.2 ε-approximation

Another tool we will use is ε-approximation, which is a useful sampling technique:

▶ **Definition 3.** *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 $\gamma\in\Gamma$, $\left|\frac{|\gamma\cap A|}{|A|}-\frac{|\gamma\cap P|}{|P|}\right|\le\varepsilon$.*

The set A above allows us to approximate the number of points of $\gamma\cap P$ with additive error $\varepsilon|P|$ 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 4** (ε-approximation [21, 22]). *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.*

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 the full version for details. For the clarity of description, we use ε_0 to denote the target error for the AHHS queries. We will reserve ε as a general error parameter. We show the following.

▶ **Theorem 5.** *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 6.** *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 7.** *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.*

3.1 Base Solution

The above results are built from a *base solution*, which solves the following problem:

▶ **Problem 3** (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 3, 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 the full version for details.

▶ **Theorem 8.** *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 8, 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 9.** *For $d \geq 3$, Problem 3 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 8. 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.

3.2 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, we build the previous base structure for the conflict list $\mathcal{S}_\Delta$ 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.

4 Approximate Quantile Summary Queries

In this section, we solve Problem 1. We first show a general technique that uses our solution to AHHS queries solution to obtain an efficient solution for AQS queries. We show that for 3D halfspace and 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 4.2, we present an optimal solution for dominance ranges based on a different idea.

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 5. 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.

4.1 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 $\mathit{\Delta}$ in the i-th shallow cutting and its conflict list $\mathcal{S}_\Delta$. Let $\epsilon = \frac{\varepsilon_0}{c}$ for a big enough constant c. We partition $\mathcal{S}_\Delta$ into $t = \frac{1}{\epsilon}$ 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 $\mathcal{T}$ and let $V(\mathcal{T})$ be the set of vertices of $\mathcal{T}$. Next, we build the following set A_Δ of colored hyperplanes, associated with Δ: Let $\varepsilon' = \frac{\epsilon}{\log^2 t}$. For every vertex $v \in V(\mathcal{T})$, 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 5, 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 2, 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 $k_{i-1} < k \le k_i$. The cell $\mathit{\Delta} \in \mathscr{C}_i$ containing q can also be found in $O(\log n)$ time using a standard point location data structure (e.g., see [1]).

The interesting part of the query is how to handle the query after finding the cell $\mathit{\Delta}$. Let H_q be the subset of H that lies below q. Recall that $\mathcal{S}_\Delta$ is the subset of H that intersects Δ. The important property of $\mathit{\Delta}$ is that $H_q \subset \mathcal{S}_\Delta$ and also $|\mathcal{S}_\Delta| = O(|H_q|) = O(k)$.

We query the data structure Ψ_Δ built for $\mathit{\Delta}$ to obtain a list of colors and their approximate counts where the additive error in the approximation is at most $\varepsilon'|A_\Delta|$. 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, 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 $\mathcal{T}$ to visualize this process. Recall that in Ψ_Δ, every vertex $v \in V(\mathcal{T})$ represents a unique color in the data structure Ψ_Δ 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 $\varepsilon'|A_\Delta|$ 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 $2\varepsilon'|A_\Delta|$. Consider the leaf node that represents $g_j \subset G_j$ and the path π that connects it to the root of $\mathcal{T}$. 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 $2\varepsilon'|A_\Delta|$ 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 $2\varepsilon'|A_\Delta|$ and in this case, we add this estimate to r_j. In both cases, we are off by an additive error of $2\varepsilon'|A_\Delta|$. 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 $2\varepsilon'|A_\Delta|\log t$. Now observe that

$$2\varepsilon'|A_\Delta|\log t = 2\frac{\epsilon}{\log^2 t}\cdot\log t|\mathcal{S}_\Delta|\cdot\log t = O(\varepsilon k) = O\left(\frac{\varepsilon_0 k}{c}\right) \leq \varepsilon_0 k$$

which follows by setting c large enough and observing the fact that $|A_\Delta| \leq \log t|\mathcal{S}_\Delta|$ since every hyperplane in $\mathcal{S}_\Delta$ is duplicated $\log t$ times.

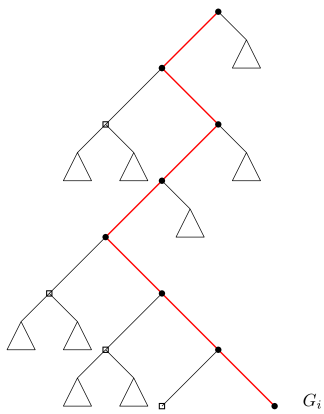

Figure 1 Compute the Approximate Rank of a Group: The approximated rank of G_i is calculated as the sum of all the approximate counts of square nodes.

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 $\mathcal{T}$ 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., $r_j \geq j'\varepsilon_0 k^*$, 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 built for cell Δ from a k_i-shallow cutting $\mathscr{C}_i$. Observe that $\sum_{v\in V(\mathcal{T})}|G_v| = |\mathcal{S}_\Delta|\log t$ since in the sum every hyperplane will be counted $\log t$ times. E_v is an ε'-approximation of G_v and thus

$$|E_v| \leq \min\left\{\varepsilon'^{-3/2}, G_v\right\} \tag{1}$$

which implies

$$|A_\Delta| = \sum_{v \in V(\mathcal{T})} |E_v| \le \min\left\{\varepsilon'^{-3/2} 2t, |\mathcal{S}_\Delta| \log t\right\} \tag{2}$$

where the first part follows as there are at most $2t$ vertices in $\mathcal{T}$ and the second part follows from (1). We build an instance of Theorem 5 on the set A_Δ which by Theorem 5 uses $O(|A_\Delta| \log_w \frac{1}{\varepsilon'})$ space. Assuming Δ belongs to a k_i-shallow cutting $\mathscr{C}_i$, we have $|\mathcal{S}_\Delta| = O(k_i)$ and there are $O(n/k_i)$ cells in $\mathscr{C}_i$. Observe that

$$\sum_{\Delta \in \mathscr{C}_i} |A_\Delta| = \sum_{\Delta \in \mathscr{C}_i} \min\left\{\varepsilon'^{-3/2} 2t, |\mathcal{S}_\Delta| \log t\right\} = \sum_{\Delta \in \mathscr{C}_i} O\left(\min\left\{\varepsilon'^{-3/2} t, k_i \log t\right\}\right) =$$
$$O\left(\min\left\{\frac{n}{k_i}\varepsilon_0^{-3}, n \log \frac{1}{\varepsilon_0}\right\}\right). \tag{3}$$

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 1, we can find the desired cutting cell in time $O(\log n)$. Next, we query the data structure Ψ_Δ which by Theorem 5 uses $O(\log n + \varepsilon'^{-1}) = O(\log n + \frac{\epsilon}{\log^2 t}) = O(\log n + \frac{1}{\varepsilon_0} \log^2 \frac{1}{\varepsilon_0})$ query time. Scanning the groups and pruning the output of the data structure Ψ_Δ takes asymptotically smaller time and thus it can be absorbed in the above expression. Therefore, we obtain the following result.

▶ **Theorem 10.** *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 with the structure in Theorem 6, and we immediately get the following:

▶ **Theorem 11.** *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 $O(\log n + \frac{1}{\varepsilon_0} \log^2 \frac{1}{\varepsilon_0})$.*

4.2 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.

4.2.1 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

$$\left| \frac{r_{P \cap \mathcal{D}}(p)}{|P|} - \frac{r_{A \cap \mathcal{D}}(p)}{|A|} \right| \leq \epsilon \tag{4}$$

for any point $p \in P$ and any range $\mathcal{D} \in \mathscr{D}$. 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 $\mathcal{D} \in \mathscr{D}$, $r_{P \cap \mathcal{D}}(p)$ (resp. $r_{A \cap \mathcal{D}}(p)$) is equal to the number of points in P' (resp. A') contained in 4D dominance range $\mathcal{D} \times (-\infty, w_p)$. By the definition of ϵ-approximation, property (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 $k \leq \frac{1}{\varepsilon_0} \log^\kappa \frac{1}{\varepsilon_0}$, for each cell Δ in the cutting $\mathscr{C}$, we collect the points in its conflict list $\mathcal{S}_\Delta$ and divide them into $t = \frac{1}{\epsilon}$ 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 $\epsilon = \frac{\varepsilon_0}{c}$ 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 $\mathcal{S}_\Delta$ first, and then divide the approximation into $t = \frac{1}{\epsilon}$ 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.

Let N be the number of points in all the t groups we generated for a cell Δ. We collect groups $G_{i \cdot \alpha + 1}, G_{i \cdot \alpha + 2}, \cdots, G_{(i+1) \cdot \alpha}$ into a cluster $\mathcal{C}_i$ for each $i = 0, 1, \cdots, t/\alpha - 1$ where $\alpha = (\log\log \frac{1}{\varepsilon_0})^3$. For each group j in cluster $\mathcal{C}_i$ 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 $\mathcal{C}_i$. Let N_i be the total number of points in $\mathcal{C}_i$:

- 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 $\mathcal{C}_i$.

To answer a query q, we first find the first shallow cutting level above q and the corresponding cutting cell Δ. 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 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 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 $\frac{|\mathcal{S}_\Delta|}{t} = \frac{\varepsilon_0 |\mathcal{S}_\Delta|}{c}$, each quantile we output will have error at most $\frac{\varepsilon_0 |\mathcal{S}_\Delta|}{c}$. For $k_i \geq \frac{1}{\varepsilon_0} \log^\kappa \frac{1}{\varepsilon_0}$, we introduce error $\epsilon |\mathcal{S}_\Delta|$ 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 $\frac{2\varepsilon_0 |\mathcal{S}_\Delta|}{c}$. In both cases, the total error is at most $\varepsilon_0 k$ for a big enough c.

4.2.2 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 $\mathcal{S}_\Delta$ of a shallow cutting cell $\Delta \in \mathscr{C}$, 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 $\mathcal{S}_\Delta$, 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 $\mathcal{S}_\Delta$ 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 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 12.** *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 $O(\log n + \frac{1}{\varepsilon_0})$.*

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 8 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 AQS 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 AQS 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 $O(n^{(d-1)/d} + \frac{1}{\varepsilon^2})$.

References

1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009.

2 Peyman Afshani, Chris Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry: Theory and Applications*, 43:700–712, 2010. preliminary version at SoCG'09.

3 Peyman Afshani and Jeff M. Phillips. Independent Range Sampling, Revisited Again. In *Symposium on Computational Geometry (SoCG)*, pages 4:1–4:13, 2019.

4 Peyman Afshani and Zhewei Wei. Independent Range Sampling, Revisited. In *Proceedings of European Symposium on Algorithms (ESA)*, volume 87, pages 3:1–3:14, 2017.

5 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.

6 Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 23–34, 2012.

7 Djamal Belazzougui, Travis Gagie, J. Ian Munro, Gonzalo Navarro, and Yakov Nekrich. Range majorities and minorities in arrays. *Algorithmica*, 83(6):1707–1733, 2021.

8 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.

9 Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Approximate range mode and range median queries. In *Proceedings of Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 2005.

[2] We thank an anonymous referee for suggesting the "output-sensitive" version.

10 Gerth Stølting Brodal, Beat Gfeller, Allan Grønlund Jørgensen, and Peter Sanders. Towards optimal range medians. *Theoretical Computer Science*, 412(24):2588–2601, 2011.

11 Timothy M. Chan, Qizheng He, and Yakov Nekrich. Further Results on Colored Range Searching. In *Symposium on Computational Geometry (SoCG)*, volume 164, pages 28:1–28:15, 2020.

12 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

13 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.

14 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.

15 P. Gupta, R. Janardan, and M. Smid. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *Journal of Algorithms*, 19(2):282–317, 1995.

16 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. `doi:10.1145/2594538.2594545`.

17 Zengfeng Huang and Ke Yi. The communication complexity of distributed epsilon-approximations. *SIAM Journal of Computing*, 46(4):1370–1394, 2017.

18 Allan Grønlund Jørgensen and Kasper Green Larsen. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proc. 22ndProceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–813, 2011.

19 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.

20 Danny Krizanc, Pat Morin, and Michiel Smid. Range mode and range median queries on lists and trees. *Nordic J. Comput.*, 12(1):1–17, 2005.

21 Jiří Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Algorithms and Combinatorics. Springer Berlin Heidelberg, 2009.

22 Jeff M. Phillips. Coresets and sketches. *CoRR*, abs/1601.00617, 2016. `arXiv:1601.00617`.

23 Zhewei Wei and Ke Yi. Beyond simple aggregates: Indexing for summary queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 117–128. ACM, 2011.

24 Ke Yi, Lu Wang, and Zhewei Wei. Indexing for summary queries: Theory and practice. *ACM Transactions on Database Systems (TODS)*, 39(1), January 2014.