Approximate Range Counting Revisited

Saladi Rahul

1 Dept. of Computer Science & Engineering, University of Minnesota, USA
sala0198@umn.edu

Abstract
This work presents several new results for approximate range counting. For a given query, if the actual count is \(k\), then the data structures in this paper output a value lying in the range \([(1-\varepsilon)k, (1+\varepsilon)k]\). Some of the results include: an optimal solution for 3-sided rectangle stabbing, colored dominance search in \(\mathbb{R}^2\), and colored 3-sided rectangle stabbing in the word-RAM model; an optimal solution for colored 3-sided range search in \(\mathbb{R}^2\), colored dominance search in \(\mathbb{R}^3\) in the pointer machine model; the first known \(k\)-sensitive structure for halfspace range counting in \(\mathbb{R}^d\), \(d \geq 4\); and, finally cheap approximate counting structures for several colored problems which have expensive exact counting structures.

Several new ideas and approaches are used to obtain these results. For example, strengthening the properties of shallow cuttings for 3-sided rectangles; two different types of random sampling techniques to handle large and small values of \(k\) for colored dominance search in \(\mathbb{R}^3\); a general reduction for colored problems which has no deterioration w.r.t. its companion problems; another general reduction which is \(k\)-sensitive; a refinement of Nekrich’s path-range trees; and finally, two different structures to handle rectangle stabbing in \(\mathbb{R}^3\): one of them is space optimal, while the other one is query time optimal.

1 Introduction

Standard geometric intersection query (Standard GIQ). In a standard geometric intersection query (GIQ), a set \(S\) of \(n\) geometric objects in \(\mathbb{R}^d\) is preprocessed into an efficient data structure so that for any geometric query object, \(q\), all the objects in \(S\) intersected by \(q\) can be reported (reporting query) or counted (counting query) quickly. In an approximate counting query, an approximate value of the number of objects in \(S\) intersecting \(q\) has to be reported; specifically, any value \(\tau\) which lies in the range \([(1-\varepsilon)k, (1+\varepsilon)k]\), where \(k = |S \cap q|\) and \(\varepsilon \in (0, 1)\). In an emptiness query, we want to decide if \(|S \cap q| = 0\) or not. Notice that the approximate counting query is at least as hard as the emptiness query: When \(k = 0\), we do not tolerate any error. Therefore, a natural goal while solving an approximate counting query is to match the space and the query time bounds of the corresponding emptiness query.

A brief history of approximate range counting. Most of the focus in the early days of research on approximate range counting was on halfspace range queries. Starting from the work of Aronov and Har-Peled [7], there was a series of results by Kaplan and Sharir [20], Afshani and Chan [1], Aronov, Har-Peled and Sharir [8], and Kaplan, Ramos and Sharir [18]. These papers dealt with either halfspace range queries in low dimensional space \((d \leq 3)\) or high dimensional space \((d \geq 4)\). Later, Afshani, Hamilton and Zeh [3] obtained an optimal solution for a general class of problems which included halfspace range query in \(\mathbb{R}^3\), dominance query in \(\mathbb{R}^3\) and 3-sided orthogonal range query in \(\mathbb{R}^2\). Interestingly, their results hold in the pointer machine model, the I/O-model and the cache-oblivious model as well. Two-dimensional orthogonal range searching query was studied by Nekrich [25], and Chan and Wilkinson [10] in the word RAM model.

Colored Geometric Intersection Queries (Colored-GIQ). Several practical applications have motivated the study of a more general class of GIQ problems, known as colored-GIQ

[arXiv:1512.01713v2 [cs.CG] 23 Dec 2016]
Approximate Range Counting Revisited

problems \[22, 29, 30, 27, 23, 21, 14, 9, 5, 13, 17, 19, 24, 25, 28, 34\]. In this setting, the set \(S\) of \(n\) geometric objects in \(\mathbb{R}^d\) come aggregated in disjoint groups. Each group is assigned a unique color. Given a geometric query object, \(q\), we are interested in reporting (colored reporting query) or counting (colored counting query) the colors which have at least one object intersected by \(q\). Note that a standard GIQ problem is a special case of its corresponding colored-GIQ problem (assign each object in the standard GIQ problem a unique color). The most popular and well studied colored-GIQ problem is the orthogonal colored range searching problem: \(S\) is a set of \(n\) points in \(\mathbb{R}^d\) and \(q\) is an axes-parallel rectangle in \(\mathbb{R}^d\). A motivating example for this problem would be the following database query: “How many countries have employees aged between 30 and 40 while earning more than 80,000 per year?”. Each employee can be represented as a point (age, salary) and the query is represented as an axes-parallel orthogonal rectangle (unbounded in one direction) \([30, 40] \times [80,000, \infty]\). Each employee is assigned a color based on his nationality. Colored-GIQ problems are known as GROUP BY queries in the database literature. Interest in colored-GIQ problems was further enhanced by the work of Muthukrishnan \[27\] who established their connection to document retrieval problems.

Nekrich \[28\] presented an approximate solution for colored 3-sided range searching in \(\mathbb{R}^2\), but with an approximation factor of \((4 + \varepsilon)\), whereas we are interested in obtaining a tighter approximation factor of \((1 + \varepsilon)\). To the best of our knowledge, this is the only work directly addressing colored approximate counting.

**Our results for standard GIQ problems:** In this paper we study the halfspace range searching problem and the rectangle stabbing problem.

**2D Approximate rectangle stabbing counting.** This paper initiates the concrete study of approximate rectangle stabbing counting query. The input is a set \(S\) of \(n\) axes-parallel rectangles whose vertices lie on a \([2n] \times [2n]\) grid and the query \(q\) is a 2D point. This specific problem is studied in the word-RAM model of computation. Consider the setting where \(S\) contains 3-sided rectangles of the form \([x_1, x_2] \times [y_1, +\infty)\). Adapting existing techniques (for e.g., Afshani, Hamilton and Zeh \[3\]) leads to a structure with space \(O_{\varepsilon}(n^{\frac{2}{3}})\) and query time \(O_{\varepsilon}((\log \log n)^2)\) time. In this paper, we present an optimal solution for this problem: a linear-space structure with a query time of \(O_{\varepsilon}(1)\) (see Theorem 1).

We do not study the case where \(S\) contains 4-sided rectangles of the form \([x_1, x_2] \times [y_1, y_2]\); because, this problem does not have a gap between the counting query and the emptiness query. For the emptiness query and the counting query, Patrascu \[32\] and Patrascu \[31\], respectively, gave a lower bound of \(\Omega\left(\frac{\log n}{\log \log n}\right)\) query time for any data structure which uses \(O(n \text{ polylog } n)\) space. JaJa, Mortensen and Shi \[10\] gave a linear-space structure with \(O\left(\frac{\log n}{\log \log n}\right)\) query time for both the problems.

**Approximate Halfspace Range Counting in \(\mathbb{R}^d, d \geq 4\):** We present a structure for halfspace range counting which is sensitive to the value of \(k\). The input is a set \(S\) of \(n\) points in \(\mathbb{R}^d\) and the query is a halfspace. The data structure occupies \(O(n)\) space and solves the query in \(\tilde{O}\left((n/k)^{1-1/(d/2)}\right)\) time. When \(k = \Theta(n)\), then the query time is \(\tilde{O}(1)\), which is an attractive property to have. See Theorem 10 for a formal statement. In \(\mathbb{R}^d, d \geq 4\), existing structures occupy \(\tilde{O}(n)\) space and solve the query in \(\tilde{O}\left(n^{1-1/(d/2)}\right)\) time. Previously, such sensitive data structures were known only in \(d = 2, 3\) \[3\].

---

1 For the sake of clarity in presentation, the notation \(O_{\varepsilon}\) hides the dependency on \(\varepsilon\).
2 The symbol \(\tilde{O}\) hides the dependency on the \(O(\text{polylog } (n/k))\) term and the \(\varepsilon\) term.
**Our results for colored-GIQ problems.** For *most* colored exact counting problems the known space and query time bounds are very expensive. For example, for orthogonal colored range searching problem in \( \mathbb{R}^d \), existing structures use \( O(n^d) \) space to achieve polylogarithmic query time. Any substantial improvement in these bounds would require improving the best exponent of matrix multiplication \[19\]. Instead of an exact count, if one is willing to settle for an approximate count, then this paper presents a result with attractive bounds: roughly, an \( O(n \log n) \) space data structure and an \( O(\varepsilon^{-2} \log^{d+1} n) \) query time algorithm. Please see Theorem 15(b) and Theorem 24 for problems, where the previously expensive counting structures now have an alternate cheap approximate counting structures.

There are, however, a *couple* of colored exact counting problems for which \( O(n \text{polylog } n) \) space and \( O(\text{polylog } n) \) query time solutions exist. While constructing approximate counting structures for such problems, our goal is to *match* the space and the query time bounds of the corresponding emptiness query. For many colored-GIQ problems, we have been able to achieve this goal. Table 1 has a list of those problems.

| Problem                          | Space         | Query Time                          |
|----------------------------------|---------------|-------------------------------------|
| Dominance search in \( \mathbb{R}^2 \), 3-sided rectangle stabbing in \( \mathbb{R}^2 \) | \( O_\varepsilon(n) \) | \( O_\varepsilon(1) \) |
| 3-sided range search in \( \mathbb{R}^2 \) | \( O(n) \) | \( O_\varepsilon(\log n) \) |
| Dominance search in \( \mathbb{R}^3 \) | \( O_\varepsilon(n \log^* n) \) | \( O_\varepsilon(\log n \cdot \log \log n) \) |

Table 1 A list of colored problems for which we have been able to match the bounds of their corresponding emptiness query. The first two entries correspond to problems which are studied in the word-RAM model of computation. The last two entries correspond to problems which are studied in the pointer machine model.

**Our Techniques.** This paper presents new ideas and approaches for solving approximate counting problems.

1. **2D rectangle stabbing.** The result for approximate rectangle stabbing counting in \( \mathbb{R}^2 \) is obtained by a new clever way of using shallow cuttings. The previously best known solution \[3\] was also based on shallow cuttings, but it had to perform \( \Omega(\log \log n) \) point location queries. We *strengthen* the properties of shallow cuttings for 3-sided rectangles, which leads to a query algorithm performing only a constant number of point location queries. The rectangle stabbing problem finds other applications since a couple of colored problems will be reduced to it (see Section 7).

2. **Colored dominance search in \( \mathbb{R}^3 \).** The result for approximate colored 3D dominance search is obtained by a non-trivial combination of two different types of random sampling techniques. One of the random sampling technique is used to handle large values of \( k \), whereas the other random sampling technique is used to handle *small* values of \( k \).

3. **First general reduction.** We present a general reduction to efficiently answer any colored approximate counting query. For a formal statement, see Theorem 11. There are a couple of interesting features of this reduction. Firstly, this reduction can be viewed as an enhancement of Aronov and Har-Peled’s approximate counting technique \[7\]. We introduce the idea of performing random sampling on *colors* (instead of input objects) to approximately count the colors intersecting the query object. Secondly, there is *no deterioration* in the performance (space and query time) w.r.t. its two companion
Approximate Range Counting Revisited

4. **Halfspace range search in** $\mathbb{R}^d, d \geq 4$. We present another general reduction in which the query time is inversely proportional to the value of $k$. See Theorem 15 for a formal statement. This turns out to be a very desirable property to have for problems such as halfspace range search in $\mathbb{R}^d, d \geq 4$ (see Theorem 19). The query algorithm of Aronov and Har-Peled essentially performs a binary search and hence, insensitive to $k$; our contribution is a new $k$-sensitive query algorithm.

5. **Buffet of range searching techniques.** Throughout the paper we use a wide variety of range searching techniques from the literature and combine them in a non-trivial manner; for example, while refining the path-range tree structure of Nekrich (Theorem 15(a)), and while constructing two different data structures for rectangle stabbing problem in $\mathbb{R}^3$: one of them is query optimal (Theorem 7), while the other one is space optimal (Section 5).

**Notations and Definitions.** Let $S$ be the set of input objects and $q$ be a query object. A GIQ problem is polynomially bounded if there are only $n^{O(1)}$ possible outcomes of $S \cap q$, over all possible values of $q$. For example, in 1d orthogonal range search on $n$ points, there are only $\Theta(n^2)$ possible outcomes of $S \cap q$. A function $f(n)$ is converging if $\sum_{i=0}^{n} n_i = n$, then $\sum_{i=0}^{n} f(n_i) = O(f(n))$. For example, it is easy to verify that $f(n) = n \log n$ is converging. For a given constant $C$, in a $C$-approximation query we return any value $z$ s.t. $k \in [z, Cz]$. A hyper-rectangle in $\mathbb{R}^d$ is called $(d+k)$-sided if it is bounded on both sides in $k$ out of the $d$ dimensions and unbounded on one side in the remaining $d-k$ dimensions.

## 2 2D Approximate Rectangle Stabbing Counting

In this section we present an optimal solution for approximate 3-sided rectangle stabbing problem. Let $S$ be a set of 3-sided rectangles (of the form $[x_1, x_2] \times [y, \infty)$). In a general setting, the rectangles of set $S$ will lie on an $[U] \times [U]$ grid. For the sake of simplicity, assume that the $x-$ and the $y-$coordinates of all the vertices of $S$ are unique. By a standard rank-reduction technique, the rectangles of $S$ can be projected to a $[2n] \times [n]$ grid: Sort the $2n$ vertical (resp. $n$ horizontal) sides in increasing order of their $x-$ (resp. $y-$) coordinate value; call the list $S_x$ (resp. $S_y$). Then each rectangle $r = [x_1, x_2] \times [y, \infty)$ is projected to a rectangle $[\text{rank}(x_1), \text{rank}(x_2)] \times [\text{rank}(y), \infty)$, where $\text{rank}(x_i)$ (resp. $\text{rank}(y)$) is the position of $x_i$ (resp. $y$) in the list $S_x$ (resp. $S_y$). Given a query point $q$, we can perform a predecessor search to find the position of $q$ on the $[2n] \times [n]$ grid, in $O(\log \log U)$ time (using a van Emde Boas structure) or in $O(\log \log n)$ time (using a fusion tree). Therefore, solving the problem on a $[2n] \times [n]$ grid is the interesting version. In Section 7 of the appendix, we show the applications of this result to colored problems.

> **Theorem 1.** Let $S$ be a set of $n$ 3-sided rectangles lying on a $[2n] \times [n]$ grid. There exists a data structure of size $O(n)$ which can solve approximate 3-sided rectangle stabbing counting problem in $O_c(1)$ time. The result is optimal in terms of $n$. This is the only problem studied in the word-RAM model.

We will first construct shallow cuttings tailored for 3-sided rectangles. Unlike the general class of shallow cuttings, the shallow cuttings we construct for 3-sided rectangles will have a stronger property which has not been used before in the literature.

> **Lemma 2.** Let $S$ be a set of 3-sided rectangles (of the form $[x_1, x_2] \times [y, \infty)$). Define a $t$-level shallow cutting to be a set $C$ of interior-disjoint 3-sided rectangles/cells (of the form $[x_1, x_2] \times (-\infty, y]$) which satisfy the following three properties:
1. \(|C| = 2n/t\).
2. If \(q\) does not lie inside any of the cell in \(C\), then \(|S \cap q| \geq t\).
3. Each cell in \(C\) intersects at most \(2t\) rectangles of \(S\).

**Proof.** Now we present a construction of the \(t\)-level shallow cutting. Partition the plane into \(2n/t\) vertical slabs, such that there are \(t\) vertical lines of \(S\) in each slab, i.e., each slab has a width of \(t\). See Figure 1(a). For each slab, say \(s = [x_1, x_2] \times (-\infty, +\infty)\), we construct a horizontal segment (referred to as upper segment) which spans the entire slab \(s\). The \(y\)-coordinate of the segment is set to \(y_t\) which determined as follows: Among all the rectangles of \(S\) which completely span the slab \(s\), let \(y_t\) be the \(y\)-coordinate of the rectangle with the \(t\)-th smallest \(y\)-coordinate. If less than \(t\) segments of \(S\) span slab \(s\), then set \(y_t := +\infty\). The cell \([x_1, x_2] \times (-\infty, y_t]\) is added to the set \(C\). See Figure 1(a).

Property 1 is easy to verify, since \(2n/t\) slabs are constructed. To prove Property 2, consider a point \(q\) which does not lie inside any cell in \(C\). Let \(s\) be the slab containing \(q\). Therefore, the \(y\)-coordinate of \(q\) is larger than \(y_t\) (where \(y_t\) corresponds to slab \(s\)). This implies that there are at least \(t\) rectangles of \(S\) which completely span the slab \(s\) and contain \(q\). Therefore, \(|S \cap q| \geq t\). To prove Property 3, consider a cell \(r \in C\) and its corresponding slab \(s\). At most \(t\) rectangles of \(S\) completely span slab \(s\) and intersect \(r\); and at most \(t\) rectangles of \(S\) partially span slab \(s\) and potentially intersect \(r\).

**Lemma 3.** (Containment Property) Let \(t\) and \(i\) be integers. Consider a \(t\)-level and a \(2^i\)-level shallow cutting. By our construction, each cell in \(2^{i-1}\)-level contains exactly \(2^i\) cells of the \(t\)-level. More importantly, each cell in the \(t\)-level is contained inside a single cell of \(2^{i-1}\)-level (see Figure 1(a)).

The key step in this problem is to find a \(4\)-approximation, which will be the focus for the remaining part of this section. (In [3], they show how to convert this into a \((1 + \varepsilon)\)-approximation.) Construct the \(2^i\)-level shallow cutting, for integers \(i = 0, 1, 2, \ldots, \log n\). Roughly speaking, given a query point \(q(x_q, y_q)\), the goal is to find the first upper segment hit by a downward vertical ray shot from \(q\). If the upper segment hit belongs to the \(t\)-level, then it is easy to verify that \(t\) will be a \(4\)-approximation of \(k\). The crux of our solution is the \((t, t')\)-level-structure.

**Lemma 4.** Suppose we know that \(q\) lies between the \(t\)-level and the \(t'\)-level shallow cutting
Approximate Range Counting Revisited

of $S$, where $t' = 2^it$ for some integer $i$. Then constructing a $(t, t')$-level-structure will answer the approximate counting query in $O(1)$ time. It will occupy $O\left(n + \frac{n}{t} \log t'\right)$ space.

Proof. We describe the construction of a $(t, t')$-level-structure. It consists of two components:

1. We say that $q_x$ lies in the $x$-range of a cell $r$ from $t$-level if the projection of $r$ onto the $x$-axis contains $q_x$. We maintain an array of $2n$ entries, so that for any $q_x \in [2n]$, we can find in $O(1)$ time the cell in the $t$-level whose $x$-range contains $q_x$.

2. For each cell, say $r$, in the $t$-level we do the following: Let $C_r$ be the set of cells from the $2^1t, 2^2t, 2^3t, \ldots, 2^tt$-level, which contain $r$ (Lemma 6 guarantees this). Now project the upper segment of each cell of $C_r$ onto the $y$-axis (each segment projects to a point). Based on the $y$-coordinates of these $|C_r|$ projected points build a fusion-tree [12], which can answer the predecessor query in $O(1)$ time, since $|C_r| = O(\log t')$. For a given query point $q(x, y)$ which lies between the $t$-level and the $t'$-level, if $q_x$ lies in the $x$-range of $r$, then it is easy to observe that finding the predecessor of $q_x$ in $C_r$ will achieve our goal. See Figure 1(b). Since there are $O(n/t)$ cells in the $t$-level, the space occupied by this structure is $O\left(\frac{n}{t} \log t'\right)$.

Structure. At first thought, one might be tempted to construct a $(0, n)$-level-structure; but, it requires $O(n \log n)$ space. Instead, we adopt the following strategy: (a) a $(\sqrt{\log n}, n)$-level-structure, and (b) a $(\log n, n)$-level-structure. These two structures will handle the case where $k \geq \sqrt{\log n}$. Crucially, the space occupied by both the structures will be $O(n)$.

To handle $k < \sqrt{\log n}$, we will perform bit tricks. We will need a shared lookup table of size $O(\sqrt{n})$. For each cell $c$ in the $\sqrt{\log n}$-level, let $S_c$ be the rectangles of $S$ intersecting $c$. We perform a rank-space reduction, so that the rectangles lie on an $[2|S_c|] \times [|S_c|]$ grid. The set $S_c$ can be represented using $O(\sqrt{\log n} \log \log n)$ bits by simply writing down the representation of each rectangle in increasing order of their $y$-coordinates. Each rectangle $[x_1, x_2] \times [y, \infty)$ is represented as “$x_1 \times x_2$” which requires $O(\log \log n)$ bits. The number of different sets of rectangles which can be represented using $O(\sqrt{\log n} \log \log n)$ bits is bounded by $2^{O(\sqrt{\log n} \log \log n)} = O(n^\delta)$, for an arbitrarily small $\delta < 1$. For each pair of (input set, query) we keep an entry in the lookup table of the exact count. The total number of entries in the lookup table is $O(n^\delta) \times O(\log^2 n) = O(n^{1/2})$. Given a query $q(x, y)$, we perform the following three $O(1)$ time operations: (a) find the cell $c$ in the $\sqrt{\log n}$-level whose $x$-range contains $q_x$, (b) then perform a rank-reduction on $q_x$ and $q_y$ to map it onto the $[2|S_c|] \times [|S_c|]$ grid. Since, $|S_c| = O(\sqrt{\log n})$, we can use a fusion tree [12] to perform the rank reduction in $O(1)$ time, and (c) finally search for $(S_c, q)$ in the lookup table.

The choice of $\sqrt{\log n}$ was arbitrary; any value $t < o(\log n / \log \log n)$ would have ensured that the lookup table, and the $(t, \log n)$-level-structure occupied $O(n)$ space. See Figure 1(c) for a summary of our data structure.

3 Colored Dominance Search in $\mathbb{R}^3$

In this section we study the approximate colored dominance search in $\mathbb{R}^3$. The input set $S$ is a set of $n$ colored points in $\mathbb{R}^3$ and the query $q$ is an octant in $\mathbb{R}^3$ (of the form $[x, \infty) \times [y, \infty) \times [z, \infty)$).

Theorem 5. For approximate colored dominance search in $\mathbb{R}^3$, there is a data structure of $O(n \log^* n + n \varepsilon^{-2})$ size which can answer the query in $O(\log n \cdot \log \log n + \varepsilon^{-2} \log n)$ time. With constant probability (say 99/100 or 999/1000) $\tau$ lies in the range $[(1 - \varepsilon)k, (1 + \varepsilon)k]$.
3.1 Reduction to 5-sided rectangle stabbing in $\mathbb{R}^3$.

In this subsection we present a reduction of colored dominance search in $\mathbb{R}^3$ to the standard 5-sided rectangle stabbing problem in $\mathbb{R}^3$. Let $S$ be a set of $n$ colored points lying in $\mathbb{R}^3$. Let $S_c \subseteq S$ be the set of points of color $c$. Let $p_1, p_2, \ldots, p_t$ be the sequence of points of $S_c$ in decreasing order of their $z$-coordinate value. With each point $p_i (p_{ix}, p_{iy}, p_{iz})$, we associate a region $\phi_i$ in $\mathbb{R}^3$ which satisfies the following invariant: a point $q (x, y, z)$ belongs to $\phi_i$ if and only if $p_i$ has the largest $z$-coordinate among points of $S_c$ in $[x, +\infty) \times [y, +\infty) \times [z, +\infty)$.

The following assignment of regions ensures the invariant:

$\phi_1 = (-\infty, p_{1x}] \times (-\infty, p_{1y}] \times (-\infty, p_{1z}]$

$\phi_i = (-\infty, p_{ix}] \times (-\infty, p_{iy}] \times (-\infty, p_{iz}] \setminus \bigcup_{j=1}^{i-1} \phi_j, \forall i \in [2, |S_c|].$

By our construction, each region $\phi_i$ is unbounded in the negative $z$-direction. Each region $\phi_i$ is broken into disjoint 5-sided rectangles via vertical decomposition in the $xy$-plane (see Figure 2(a)). The vertical decomposition ensures that the total number of disjoint rectangles generated is bounded by $O(|S_c|)$. Now we can observe that (i) If a color $c$ has at least one point inside $q$, then exactly one of its transformed rectangle will contain $q$, and (ii) If a color $c$ has no point inside $q$, then none of its transformed rectangles will contain $q$. Therefore, the colored dominance search in $\mathbb{R}^3$ has been reduced to the 5-sided rectangle stabbing query.

3.2 When $k \in [C\varepsilon^{-2} \log^4 n, n]$

Using the above reduction, in this subsection we study the approximate 5-sided rectangle stabbing in $\mathbb{R}^3$ (5-sided ARS) and prove the following theorem.

\textbf{Theorem 6.} Suppose $k \geq C\varepsilon^{-2} \log^4 n$, where $C$ is a suitably large constant. Then there exists a data structure of size $O(\varepsilon^{-2} n)$ which can solve 5-sided ARS problem in $O(\log n)$ time.

A brief overview of the proof of Theorem 6. The idea of using random sampling for answering an approximate range counting query has been used in the past in [7, 10, 15] which deal with non-colored objects. Random sampling helps reduce the size of the input set and thus, allows us to use a slightly space-inferior data structure. Theorem 7 presents the slightly space-inferior but crucially query time optimal structure for solving 5-sided ARS
problem. Then the random sampling technique of Lemma 8 is applied along with Theorem 7 to obtain a space optimal and query time optimal solution when $k \geq C\varepsilon^{-2}\log^4 n$.

**Theorem 7.** Let $R$ be a set of $m$ 5-sided rectangles in $\mathbb{R}^3$. Then there exists a data structure of size $O(\varepsilon^{-2}m\log^3 m)$ which can solve 5-sided ARS problem in $O(\log m)$ time.

**Intuition behind proof of Theorem 7.** The formal proof of Theorem 7 can be found in Section 8. We only provide some intuition behind the proof. Consider a simpler setting where the set $R$ is $m$ 3-sided rectangles of the form $[y_1, y_2] \times (-\infty, z]$ in $\mathbb{R}^2$. Project these rectangles onto the $y$-axis. Let $E_y = (e_1, e_2, \ldots, e_{2m})$ be the sorted sequence (in increasing order of $y$-coordinate) of the endpoints of these projected intervals. We divide the $y$-axis into elementary segments $(-\infty, e_1), [e_1, e_2), [e_2, e_3), \ldots, [e_{2m-1}, e_{2m}], (e_{2m}, \infty)$. For any elementary segment, say $es$, let $R^{es}$ be the set of rectangles completely crossing the segment $es$. If the query point $q$ lies in $es$, to answer the approximate counting query it is enough to store a sketch of $R^{es}$. See Figure 2(b). The size of the sketch of $R^{es}$ is only $O(\varepsilon^{-1}\log m)$. The total size of all the sketches will be $O(\varepsilon^{-1}\log m)$.

More machinery is needed to handle the more involved case of 5-sided rectangles; we use ideas such as (a) constructing an “external-memory style” segment tree [6] with fanout $\varepsilon^{-1}\log m$, and (b) fractional cascading to efficiently query the $O\left(\frac{\log m}{\log(\varepsilon^{-1}\log m)}\right)$ sketches in the segment tree.

**Lemma 8.** For a particular GIQ problem which is polynomially bounded, let $S$ be a set of $n$ objects in $\mathbb{R}^d$. We consider query objects $q$ such that $k \geq \delta(C\varepsilon^{-2}\log n)$. Then there exists a set $R \subset S$ of size $O(n/\delta)$ such that $|\{R \cap q \cdot \delta\} \in [(1 - \varepsilon)k, (1 + \varepsilon)k]$.

**Final Structure (Combining Theorem 7 and Lemma 8):** Let $S$ be a set of $n$ 5-sided rectangles in $\mathbb{R}^3$. The number of combinatorially different query points on $S$ is bounded by $O(n^2)$. We set $\delta \leftarrow \log^3 n$ and define a new parameter $\varepsilon' \leftarrow \varepsilon/4$. Now we apply Lemma 8 to obtain a set $R$ of size $O(n/\log^3 n)$. Based on the set $R$ and with error parameter $\varepsilon'$, we build the data structure of Theorem 7. Given a query on $S$, we query the data structure built on $R$. Let $\tau_R$ be the value returned. Then we report $\tau_R\log^2 n$ as the answer to the 5-sided ARS problem on $S$. The analysis can be found in Section 10 of the appendix.

### 3.3 When $k \in [0, C\varepsilon^{-2}\log^4 n]$}

To handle this case, we will still work with the 5-sided ARS problem in $\mathbb{R}^3$. We need a variant of Lemma 8 which is stated next.

**Lemma 9.** For a particular GIQ problem, let $S$ be a set of $n$ objects in $\mathbb{R}^d$. We consider query objects $q$ such that $k \geq C\varepsilon^{-2}\delta$. Then there exists a set $R \subset S$ of size $O(n/\delta)$ such that with a certain constant positive probability $|\{R \cap q\cdot \delta\} \in [(1 - \varepsilon)k, (1 + \varepsilon)k]$.

Let $S$ be the set of $n$ 5-sided rectangles in $\mathbb{R}^3$. Now we are ready to present the solution for all the sub-cases:

1. **When $k \in [0, C\varepsilon^{-2}\log n]$**: Rahul [33] presented a structure of size $O(n\log^* n)$ which can answer a 5-sided rectangle stabbing query in $O(\log n \cdot \log \log n + k)$ time. Build this structure on all the rectangles in set $S$. This can be used to determine the exact value of $k$ or conclude that $k > C\varepsilon^{-2}\log n$.

2. **When $k \in [C\varepsilon^{-2}\log n, C\varepsilon^{-2}\log^2 n]$**: We shall now use Lemma 9. We set $\delta = \log n$. Then as per the requirement of Lemma 9 the query $q$ indeed has $k > C\varepsilon^{-2}\log n$. To apply
Lemma 9 efficiently, we need a structure for computing $|R \cap q|$. For this we build the reporting structure of Rahul \[33\] on all the rectangles of $R$. Given a query point $q$, we query this structure to keep reporting all the rectangles in $R \cap q$ till one of the following event happens: either all the rectangles in $R \cap q$ have been reported or $2\epsilon^{-2} \log n + 1$ rectangles in $R \cap q$ have been reported. If the first event happens, then we have succeeded in obtaining the exact value of $|R \cap q|$ in $O(\log n \cdot \log \log n + \epsilon^{-2} \log n)$ time. We report $|R \cap q| \cdot \delta$ as the answer. On the other hand, if the second event happens, then we conclude that with certain constant positive probability: $(1 + \epsilon)k \geq |R \cap q| \cdot \delta > 2\epsilon^{-2} \log^2 n \implies k > \epsilon^{-2} \log^2 n$

(3) When $k \in [C\epsilon^{-2} \log^2 n, C\epsilon^{-2} \log^3 n]$ and $k \in [C\epsilon^{-2} \log^3 n, C\epsilon^{-2} \log^4 n]$: They are handled similar to case (2), by appropriately invoking Lemma 9.

**Lemma 10.** When $k \leq C\epsilon^{-2} \log^4 n$, there exists a data structure of size $O(n \log^* n)$ which can solve the approximate colored dominance search in $\mathbb{R}^3$ in $O(\log n \cdot \log \log n + \epsilon^{-2} \log n)$ time. With constant probability (such as 99/100 or 999/1000) the value of $\tau$ lies in the range $[(1-\epsilon)k, (1+\epsilon)k]$.

## 4 First General Reduction

Our first reduction states that given a colored reporting structure and a colored $C$-approximation structure, one can obtain a colored $(1 + \epsilon)$-approximation structure with no additional loss of efficiency.

**Theorem 11.** For a given colored-GIQ problem, assume that we are given the following two structures: (a) a colored reporting structure of $S_{rep}(n)$ size which can solve the query in $O(Q_{rep}(n) + \kappa)$ time, where $\kappa$ is the output-size; and (b) a colored $C$-approximation structure of $S_{app}(n)$ size which can solve the query in $O(Q_{app}(n))$ time.

We also assume the following two facts: (a) $S_{rep}(n)$ and $S_{app}(n)$ are converging, and (b) the GIQ problem is polynomially bounded.

Then, one can obtain a $(1 + \epsilon)$ approximation using a structure of $S_{app}(n)$ size and query time $Q_{app}(n)$ with

\[
S_{app}(n) = O(S_{rep}(n) + S_{app}(n)) \tag{1}
\]

\[
Q_{app}(n) = O(Q_{rep}(n) + Q_{app}(n) + \epsilon^{-2} \cdot \log n) \tag{2}
\]

At a high-level, the query algorithm consists of the following steps:

1. If $k \leq \epsilon^{-2} \log n$, then we will report the exact value of $k$. Based on the set $S$ we build a colored reporting structure. Given a query object $q$, we query the colored reporting structure to keep reporting the colors in $S \cap q$ till one of the following event happens: either all the colors in $S \cap q$ have been reported or $\epsilon^{-2} \log n + 1$ colors in $S \cap q$ have been reported. If the first event happens, then we have succeeded in obtaining the exact value of $k$. On the other hand, if the second event happens, then we can conclude that $k = \Omega(\epsilon^{-2} \log n)$ (i.e., $k$ is large). This query algorithm takes $O(Q_{rep}(n) + \epsilon^{-2} \log n)$ time.

2. If $k > \epsilon^{-2} \log n$, then
   a. First, we find a $C$-approximation of $k$.
   b. Then, we refine the $C$-approximation of $k$ to a $(1 + \epsilon)$-approximation of $k$. 

In the remaining section, we will describe an efficient implementation of step 2(b). The implementation of Step 2(a) is specific to each colored GIQ problem and is handled later in the paper.

**Sampling colors.** Let $C$ be the set of colors in the set $S$. From now on we can safely assume that $k = |C \cap q| = \Omega(\varepsilon^{-2}\log n)$. From Step 2(a) we can conclude that $k \in [z, Cz]$. Now the task of the refinement query is to find a value $\tau \in [(1 - \varepsilon)k, (1 + \varepsilon)k]$. The following lemma shows that sampling colors (instead of input objects) is an effective tool to attack the problem.

**Lemma 12.** Let $c_1$ be a sufficiently large constant and $c$ be another constant s.t. $c = \Theta(c_1 \log e)$. We are given a query $q$ s.t. $k \in [z, Cz]$.

Consider a random sample $R$ where each color in $C$ is picked independently with probability $M = \frac{c_1 \varepsilon^{-2} \log n}{n}$. Then with probability $1 - n^{-c}$ we have $|k - |R \cap q|| \leq \varepsilon k$. Here $R \cap q$ denotes the set of colors in $R$ which have at least one object inside $q$.

**Proof.** For each of the $k$ colors of $C$ which intersect $q$ define an indicator variable $X_i$. Set $X_i = 1$ if the corresponding color is in the random sample $R$, otherwise set $X_i = 0$. Then $|R \cap q| = \sum_{i=1}^{k} X_i$ and $E[|R \cap q|] = k \cdot M$. By Chernoff bound,

$$
\Pr \left[ \left| |R \cap q| - E[|R \cap q|] \right| > \varepsilon \cdot E[|R \cap q|] \right] < \exp \left( -\varepsilon^2 E[|R \cap q|] \right)
$$

$$
< \exp \left( -\varepsilon^2 \cdot kM \right) < \exp \left( -\varepsilon^2 \cdot 2M \right) < \exp \left( -c_1 \log n \right) \leq \frac{1}{n^c}
$$

Therefore, with high probability $\left| |R \cap q| - kM \right| < \varepsilon \cdot kM$.

**Lemma 13. (Finding a suitable $R$)** Pick a random sample $R$ as defined in Lemma 12. Let $n_R$ be the number of points of $S$ whose color belongs to $R$. We say $R$ is suitable if it satisfies the following two conditions:

- $|k - \frac{|R \cap q|}{M}| \leq \varepsilon k$ for all the queries which have $k \in [z, Cz]$.
- $n_R \leq 10nM$. This condition is needed to bound the size of the data structure.

A suitable $R$ always exists.

**Proof.** Let $n^\alpha$ be the number of combinatorially different queries $q$ on the set $S$. From Lemma 12, by setting $c = \alpha + 1$, we can conclude that $\tau \leftarrow \frac{|R \cap q|}{M}$ will lie in the range $[(1 - \varepsilon)k, (1 + \varepsilon)k]$ with probability at least $1 - 1/n^{\alpha + 1}$. By the standard union bound, it implies that the probability of the random sample $R$ failing for any query is at most $1/n^{\alpha + 1} \times n^\alpha = 1/n$.

It is easy to observe that the expected value of $n_R$ is $nM$: Let $n_c$ be the number of points of $S$ having color $c$. Then $E[n_R] = \sum_{c} n_c \cdot M = nM$. By Markov’s inequality, the probability of $n_R$ being larger than $10nM$ is less than or equal to $1/10$. By union bound, $R$ will be not be suitable with probability $\leq 1/n + 1/10$. Therefore, with probability $\geq 9/10 - 1/n$, $R$ will be suitable and hence, a suitable $R$ exists. We do not discuss the preprocessing time here, since it is not known how to efficiently verify if a sample $R$ is suitable. We leave this as an interesting open problem.

**Lemma 14.** The refinement query can be solved in query time $O(Q_{rep}(n) + \varepsilon^{-2}\log n)$. The answer $\tau$ lies in the range $[(1 - \varepsilon)k, (1 + \varepsilon)k]$ with probability 1.
Proof. In the preprocessing stage we pick a random sample $R \subseteq C$ as stated in Lemma 12. If the sample $R$ is not suitable, then we discard $R$ and re-sample, till we get a suitable sample. Based on all the objects of $S$ whose color belongs to $R$, we build a colored reporting structure. Given a query object $q$, we query the colored reporting structure to keep reporting the colors in $R \cap q$. We report $\tau \leftarrow (\lfloor R \cap q \rfloor / M)$ as the answer to the query. The query time is bounded by $O(Q_{rep}(n) + \varepsilon^{-2} \log n)$, since by Lemma 12, $|R \cap q| \leq (1 + \varepsilon) \cdot kM = O(\varepsilon^{-2} \log n)$. ▶

Overall Data Structure. Build the refinement structure of Lemma 13 for the values $\tau = (\sqrt{C})^i \cdot \varepsilon^{-2} \log n$ for $i \in \left[0, \log \sqrt{C} \left(\lfloor \varepsilon^2 n \rfloor \right)\right]$. The total size of all the refinement structures will be $\sum O(S_{rep}(nM)) = O(S_{rep}(n))$, since $S_{rep}(\cdot)$ is converging and $\sum nM = O(n)$. Note that our choice of $\tau$ ensures that the size of the data structure is independent of $\varepsilon$.

Query algorithm: Let $k_a$ be the $\sqrt{C}$-approximate value returned in Step 2(a), s.t., $k \in [k_a, \sqrt{C}k_a]$. Then we query the refinement structure with the largest value of $\tau$ s.t. $\tau \leq k_a \leq \sqrt{C}$. It is trivial to verify that $k \in [z, Cz]$.

5 Colored Orthogonal Range Search

In this section we study the approximate colored 3-sided/4-sided range search in $\mathbb{R}^2$. The input set $S$ is a set of colored points in $\mathbb{R}^2$, and the query $q$ is a 3-sided/4-sided rectangle in $\mathbb{R}^2$, respectively.

- Theorem 15. (a) For approximate colored 3-sided range search in $\mathbb{R}^2$, there is a data structure of $O(n)$ size which can answer the query in $O(\varepsilon^{-2} \log n)$ time.

(b) For approximate colored 4-sided range search in $\mathbb{R}^2$, there is a data structure of $O(n \log n)$ size which can answer the query in $O(\varepsilon^{-2} \log n)$ time.

Proof of Theorem 15(a). We use the framework of Theorem 11 to prove the result. For this problem, a reporting structure with $S_{rep} = n$ and $Q_{rep} = \log n$ is already known in the literature [34]. In this section we present a $C$-approximation structure. The path-range tree of Nekrich [28] gives a $(4 + \varepsilon)$-approximation, but the space occupied by path-range tree is super-linear. Our $C$-approximation structure can be viewed as a refinement of path-range tree for pointer machine.

- Lemma 16. There exists a $C$-approximation structure of $O(n)$ size for colored 3-sided range search in $\mathbb{R}^2$ which can answer the query in $O(\log n)$ time.

By a simple exercise, the colored 3-sided range search in $\mathbb{R}^2$ can be reduced to the colored dominance search in $\mathbb{R}^3$. Therefore, using the reduction of Subsection 3.1 the 3-sided colored range search in $\mathbb{R}^2$ also reduces to 5-sided rectangle stabbing problem.

Initial structure: First, we present a linear-sized data structure to answer the approximate 5-sided rectangle stabbing (5-sided ARS) in $O(\log^2 n)$ query time. The approximation factor will be $C = 2$. Call this structure $D$.

- Lemma 17. Using interval trees, a query on $(3 + t)$-sided rectangles in $\mathbb{R}^3$ can be broken down into $O(\log n)$ queries on $(2 + t)$-sided rectangles in $\mathbb{R}^3$. Here $t \in [1, 3]$.

Proof. Let $R$ be a set of $n$ $(3 + t)$-sided rectangles. We build an interval tree $I\mathcal{T}$ as follows. W.l.o.g. assume that the rectangles are bounded along the $x$-axis. Then project these rectangles onto the $x$-axis and let $E_x$ be the set of endpoints of the projected segments. Then find the median of $E_x$ and vertically divide the space into two halves. The splitting halfplane
is stored at the root of $TT$ and then the two subtrees are built recursively. Define $split(v)$ to be the splitting halfplane stored at a node $v \in TT$. Consider a rectangle $r \in R$. $r$ is stored at the highest node $v$ s.t. $r$ intersects $split(v)$. Let $R_v$ be the set of rectangles assigned to a node $v$. Each rectangle in $R_v$ is split by $split(v)$ into two rectangles $R^-_v$ and $R^+_v$.

Given a query point $q$, we trace a path of length $O(\log n)$ from the root to a leaf node corresponding to $q$. The crucial observation is that if $q$ lies to the left (resp. right) of $split(v)$, (a) then answering a query on $R_v \cap q$ is equivalent to answering it on $R^-_v \cap q$ (resp. $R^+_v \cap q$); and (b) now we can treat $R^-_v$ (resp. $R^+_v$) as $(2 + t)$-sided rectangles in $R^3$, since $split(v)$ is effectively $+\infty$ (resp. $-\infty$).

By inductively applying the above lemma twice, we can decompose the 5-sided rectangle stabbing to $O(\log^2 n)$ 3-sided rectangle stabbing problem in $R^3$. The structure of $[3]$ has linear size and answers the approximate 3-sided rectangle stabbing problem (a.k.a. approximate dominance counting in $R^3$) in $O(\log n)$ time. We pick an approximation factor of $C = 2$. Therefore, the time taken to answer the 5-sided ARS problem is $O(\log^3 n)$. Since the induction is applied only twice, the space remains linear.

**Final structure:** Sort the points of $S$ based on their $x$-coordinate value and divide them into buckets of $\log^2 n$ consecutive points each. For the points in each bucket, build an instance of structure $D$. Next, build a height-balanced binary search tree $T$ on top of these buckets based on their left to right ordering along the $x$-axis. Let $v$ be a proper ancestor of a leaf node $u$ and let $\Pi(u,v)$ be the path from $u$ to $v$ (excluding $u$ and $v$). Let $S_l(u,v)$ be the set of points in the subtrees rooted at nodes that are left children of nodes on the path $\Pi(u,v)$ but not themselves on the path. Similarly, let $S_r(u,v)$ be the set of points in the subtrees rooted at nodes that are right children of nodes on the path $\Pi(u,v)$ but not themselves on the path. See Figure [3] which illustrates these sets for two leaves $u = u_l$ and $u = u_r$ that are identified in the query algorithm discussed below. For each pair $(u,v)$, let $S'_l(u,v)$ (resp. $S'_r(u,v)$) be the set of points that each have the highest $y$-coordinate value among the points of the same color in $S_l(u,v)$ (resp. $S_r(u,v)$).

Finally, for each pair $(u,v)$, construct a sketch, $S''_l(u,v)$, of the set $S'_l(u,v)$: by selecting the $2^0, 2^1, 2^2, \ldots$-th highest $y$-coordinate point in $S'_l(u,v)$. A symmetric construction is performed to obtain $S''_r(u,v)$. The number of $(u,v)$ pairs is bounded by $O((n/\log^2 n) \times (\log n)) = O(n/\log n)$ and hence, the space occupied by all the $S''_l(u,v)$ and $S''_r(u,v)$ sets is $O(n)$.

To answer a query $q = [x_1, x_2] \times [y, \infty)$, we first determine the leaf nodes $u_l$ and $u_r$ of $T$ containing $x_1$ and $x_2$, respectively. If $u_l = u_r$, then we query the $D$ structure corresponding to the leaf node and we are done. If $u_l \neq u_r$, then we find the node $v$ which is the least common ancestor of $u_l$ and $u_r$. The query is now broken into four sub-queries: First, report the approximate count in the leaves $u_l$ and $u_r$. These colors are obtained by querying the $D$ structure of $u_l$ with $[x_1, \infty) \times [y, \infty)$ and the $D$ structure of $u_r$ with $(-\infty, x_2] \times [y, \infty)$.\[\Box\]
Next, we scan the list $S''_i(u, v)$ (resp. $S''_i(u_r, v)$) to find a 2-approximation of the number of colors of $S_i(u, v)$ (resp. $S_i(u_r, v)$) lying inside $q$.

We report the sum of the count returned by the four sub-queries as the final answer. The time taken to find $u$, $u_r$, and $v$ is bounded by $O(\log n)$. Querying the leaf structures takes $O((\log(\log^2 n))^3) = O(\log n)$ time. The time taken for scanning the lists $S''_i(u, v)$ and $S''_i(u_r, v)$ is bounded by $O(\log n)$. Therefore, the overall query time is bounded by $O(\log n)$.

Since each of the four queries give a 2-approximation, overall we get an 8-approximation. This finishes the proof of Theorem 15(a). The proof of Theorem 15(b) can be found in the appendix (Section 12).

6 Second General Reduction

Finally, we will informally state a portion of our second general reduction, then give a brief overview of the solution, and present an application of this reduction to approximate halfspace range reporting. The appendix presents the precise and complete statement of our general reduction, and a proof of it (Section 13). Section 14 of the appendix presents an overview of the solution, and presents an application of this reduction to approximate halfspace range search counting. The appendix presents the precise and complete statement of Theorem 18.

**Theorem 18.** For a given colored-GIQ problem, assume that we are given a colored reporting structure of $S_{\text{rep}}(n)$ size which can solve the query in $O(Q_{\text{rep}}(n) + \kappa)$ time. Then, one can obtain a 1.5-approximation using a structure of $S_{\text{app}}(n) = O(S_{\text{rep}}(n))$ size and query time

$$Q_{\text{app}}(n) = \tilde{O} \left( \frac{Q_{\text{rep}}(n)}{\kappa} \right)$$

Observe that the query time is sensitive to the value of $\kappa$.

**Overview of the data structure.** Given a number $z = \Omega(\log n)$, a decision data structure will determine if $|C \cap q| \geq z$ or $|C \cap q| < z$? The data structure is allowed to make a mistake when $|C \cap q| \in [z/2, 3z/2]$. For the values $z_i = (3/2)^i \log n$, for $i = 1, 2, 3, \ldots, W = O(\log |C|)$, decision data structures $D_i$ are built. Each $D_i$ is built on roughly $O(n/z_i)$ objects and $\tilde{O}(Q_{\text{rep}}(n/z_i))$ is the time taken to query it.

**Query algorithm:** For a moment, assume that we query all the data structures $D_1, \ldots, D_W$. Then we will see a sequence of structures $D_j$ for $j \in [1, i]$ claiming $|C \cap q| > z_j$, followed by a sequence of structures $D_{i+1}, \ldots, D_W$ claiming $|C \cap q| \leq z_j$. Then we shall set $\tau \leftarrow z_i$ as the answer to the approximate colored counting query. A simple calculation reveals that $\tau$ will lie in the range $[k/2, 3k/2]$. We propose a query algorithm which will prove equation 3 of Theorem 18.

The key idea is to query the data structures in *increasing order* of their sizes. Query the data structures in order $D_W, D_{W-1}, D_{W-2}, \ldots D_i$ to find the index $i$. Since $D_i$ is the largest-size data structure queried, the overall query time will be bounded by

$$\frac{\text{# structures queried}}{(W - i)} \cdot \tilde{O} \left( Q_{\text{rep}} \left( \frac{n}{z_i} \right) \right) \approx \tilde{O} \left( Q_{\text{rep}} \left( \frac{n}{k} \right) \right)$$

**Approximate halfspace counting query.** Afshani and Chan [11] presented an $O(n)$ space structure for halfspace range reporting in $\mathbb{R}^d, d \geq 4$ which can solve the query in
Approximate Range Counting Revisited

\( \tilde{O}(n^{1-1/|d/2|} + k) \) time. Applying Theorem 18 (recall that a standard GIQ is a special case of colored GIQ), we obtain the following result.

**Theorem 19.** There is a data structure of size \( O(n) \) which can solve the approximate halfspace range counting in \( \mathbb{R}^d, d \geq 4 \) in \( \tilde{O}\left((n/k)^{1-1/|d/2|}\right) \) time.

References

1. Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. *Discrete \\& Computational Geometry*, 42(1):3–21, 2009.
2. 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.
3. Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry: Theory and Applications*, 43(8):700–712, 2010.
4. Pankaj K. Agarwal, Siu-Wing Cheng, Yufei Tao, and Ke Yi. Indexing uncertain data. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 137–146, 2009.
5. Pankaj K. Agarwal, Sathish Govindarajan, and S. Muthukrishnan. Range searching in categorical data: Colored range searching on grid. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 17–28, 2002.
6. Lars Arge and Jeffrey Scott Vitter. Optimal dynamic interval management in external memory. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 560–569, 1996.
7. Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008.
8. Boris Aronov, Sariel Har-Peled, and Micha Sharir. On approximate halfspace range counting and relative epsilon-approximations. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 327–336, 2007.
9. Panayiotis Bozanis, Nectarios Kitsios, Christos Makris, and Athanasios K. Tsakalidis. New upper bounds for generalized intersection searching problems. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 464–474, 1995.
10. Timothy M. Chan and Bryan T. Wilkinson. Adaptive and approximate orthogonal range counting. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 241–251, 2013.
11. Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986.
12. Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences (JCSS)*, 47(3):424–436, 1993.
13. Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *Journal of Algorithms*, 19(2):282–317, 1995.
14. Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Computational geometry: Generalized intersection searching. In *Handbook of Data Structures and Applications*. 2004.
15. Sariel Har-Peled and Micha Sharir. Relative \((p, \varepsilon)\)-approximations in geometry. *Discrete \\& Computational Geometry*, 45(3):462–496, 2011.
16. Joseph JáJá, Christian Worm Mortensen, and Qingmin Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings*, pages 558–568, 2004.
Ravi Janardan and Mario A. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications*, 3(1):39–69, 1993.

Haim Kaplan, Edgar Ramos, and Micha Sharir. Range minima queries with respect to a random permutation, and approximate range counting. *Discrete & Computational Geometry*, 45(1):3–33, 2011.

Haim Kaplan, Natan Rubin, Micha Sharir, and Elad Verbin. Counting colors in boxes. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 785–794, 2007.

Haim Kaplan and Micha Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar Voronoi diagrams, and approximate range counting. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 484–493, 2006.

Haim Kaplan, Micha Sharir, and Elad Verbin. Colored intersection searching via sparse rectangular matrix multiplication. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 52–60, 2006.

Marek Karpinski and Yakov Nekrich. Top-k color queries for document retrieval. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 401–411, 2011.

Ying Kit Lai, Chung Keung Poon, and Benyun Shi. Approximate colored range and point enclosure queries. *Journal of Discrete Algorithms*, 6(3):420–432, 2008.

Kasper Green Larsen and Rasmus Pagh. I/O-efficient data structures for colored range and prefix reporting. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 583–592, 2012.

Kasper Green Larsen and Freek van Walderveen. Near-optimal range reporting structures for categorical data. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 265–276, 2013.

Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002.

Yakov Nekrich. Efficient range searching for categorical and plain data. *ACM Transactions on Database Systems (TODS)*, 39(1):9, 2014.

Yakov Nekrich and Jeffrey Scott Vitter. Optimal color range reporting in one dimension. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 743–754, 2013.

Manish Patil, Sharma V. Thankachan, Rahul Shah, Yakov Nekrich, and Jeffrey Scott Vitter. Categorical range maxima queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 266–277, 2014.

Mihai Patrascu. Lower bounds for 2-dimensional range counting. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA*, June 11-13, 2007, pages 40–46, 2007.

Mihai Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM Journal of Computing*, 40(3):827–847, 2011.

Saladi Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 200–211, 2015.

Qingmin Shi and Joseph JáJá. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Information Processing Letters (IPL)*, 95(3):382–388, 2005.
Appendix

7 Applications of rectangle stabbing to colored problems

We show applications of the approximate rectangle stabbing result to the following three colored problems: colored dominance search in $\mathbb{R}^2$, colored interval stabbing in $\mathbb{R}^1$ and colored 3-sided rectangle stabbing in $\mathbb{R}^2$. In the colored dominance search in $\mathbb{R}^2$ the input is a set of $n$ colored points in $\mathbb{R}^2$ and the query is a quadrant. In the colored interval stabbing in $\mathbb{R}^1$ the input is a set of $n$ colored intervals in $\mathbb{R}^1$ and the query is a point. Now the colored 3-sided rectangle stabbing in $\mathbb{R}^2$ subsumes the above two problems and hence, we will only focus on this problem. Here the input is a set $S$ of $n$ colored 3-sided rectangles in $\mathbb{R}^2$ and the query $q$ is a point in $\mathbb{R}^2$.

Let $S_c$ be the set of rectangles of a color $c$. In the preprocessing phase, we perform the following steps: (1) Construct a union of the rectangles of $S_c$. Call it $U(S_c)$. (2) Now perform a vertical decomposition of $U(S_c)$ by shooting a vertical ray upwards from every vertex of $U(S_c)$ till it hits $\infty$. This leads to a decomposition of $U(S_c)$ into $|S_c|$ pairwise-disjoint 3-sided rectangles. Call these new set of rectangles $S'_c$. For every color $c$ we construct the set $S'_c$. Now we can observe that (i) If $S_c \cap q = \emptyset$, then $S'_c \cap q = \emptyset$, and (ii) $S_c \cap q \neq \emptyset$, then exactly one rectangle in $S'_c$ is stabbed by $q$. Therefore, the colored 3-sided rectangle stabbing in $\mathbb{R}^2$ has been reduced to the standard 3-sided rectangle stabbing query in $\mathbb{R}^2$.

Let $S' = \bigcup_{c} S'_c$. Clearly $|S'| = O(n)$. On $S'$ we build an approximate 3-sided rectangle stabbing in $\mathbb{R}^2$ data structure of Theorem 1. This leads to a data structure of size $O_2(n)$ which can answer the query in $O_2(1)$ time.

8 Proof of Theorem 7: Optimal query time

Primary Structure: Inspired by external memory tree structures with large fanout, our primary structure will be a segment tree $\overline{ST}$ with fanout $f = \varepsilon^{-1}\log m$. Project the rectangles of $R$ onto the $x$-axis. Let $E_x$ be the set of endpoints of these projected intervals (each rectangle gets projected into an interval). Build an $f$-ary tree $ST$ with the set $E_x$ stored at the leaves. For each internal node $v \in ST$ we define its range on the $x$-axis, say $xrange(v)$, as the union of the $xrange(\cdot)$ of its children $v_1, v_2, \ldots, v_f$, i.e., $xrange(v) = \bigcup_{i=1}^{f} xrange(v_i) = [x_1, x_f]$. For each internal node $v \in ST$ we also define $f + 1$ boundary slabs $b_1(v), b_2(v), \ldots, b_{f+1}(v)$: $b_1(v) = x_1, b_{f+1} = x_f$ and $\forall 1 < i < f + 1, b_i$ is the boundary separating $xrange(v_{i-1})$ and $xrange(v_i)$.

Consider a rectangle $r = [x_1, x_2] \times [y_1, y_2] \times (-\infty, z) \in R$. To store the rectangle $r$ in $ST$, we start a tour from the root node of $ST$. Let $v$ be the current node visited. There are two possibilities: (a) If $[x_1, x_2]$ does not intersect any of the boundary slabs, then we visit the child of $v$ whose $xrange(\cdot)$ completely contains $[x_1, x_2]$. (b) Otherwise, the rectangle $r$ is assigned to node $v$ and broken into three disjoint rectangles as follows: Let $x_1$ lie in $xrange(v_i)$ and $x_2$ lie in $xrange(v_j)$. Then $r$ is broken into a left rectangle $r_l = [x_1, b_{i+1}(v)] \times [y_1, y_2] \times (-\infty, z)$, middle rectangle $r_m = [b_i(v), b_j(v)] \times [y_1, y_2] \times (-\infty, z]$ and a right rectangle $r_r = (b_j(v), x_2] \times [y_1, y_2] \times (-\infty, z]$. The left (resp. right) rectangle is recursively stored in the subtree of $v_i$ (resp. $v_j$). See Figure 3 for an illustration of a node $v \in ST$. The net effect is that each rectangle in $\overline{R}$ is assigned to at most two nodes per level in $ST$.

We augment each node $v \in ST$ with a couple of secondary data structures. Define $R_v$ to
be the set of rectangles assigned to node $v$. Next, define $R_{v_i} \subseteq R_v$ to be the set of rectangles completely crossing the $xrange(v_i)$ (in Figure 4 rectangle $r$ completely crosses $xrange(v_2)$ and $xrange(v_3)$). For a given query point $q(q_x, q_y, q_z)$, if $q_z \in xrange(v_i)$, then the problem of approximately estimating the size of $R_v \cap q$ is reduced to the problem of approximately estimating the size of $R_{v_i} \cap q$. More importantly, the problem is now reduced to 3-sided rectangles in the $yz$-plane. Next, we present a secondary structure to efficiently estimate the size of $R_{v_i} \cap q$, under the assumption that $q_z \in xrange(v_i)$.

Secondary Structure: Project the rectangles $R_{v_i}$ onto the $y$-axis. Let $E_y = (e_1, e_2, \ldots, e_{2|R_{v_i}|})$ be the sorted sequence (in increasing order of $y$-coordinate) of the endpoints of these projected intervals. We divide the $y$-axis into elementary segments $(-\infty, e_1], (e_1, e_2], (e_2, e_3], \ldots, (e_{2|R_{v_i}| - 1}, e_{2|R_{v_i}|], (e_{2|R_{v_i}|}, e_{2|R_{v_i}|}], (e_{2|R_{v_i}|}, \infty)$. For each elementary segment, say $es$, let $R_{v_i}^{es} \subseteq R_{v_i}$ be the set of rectangles completely crossing the segment $es$. The rectangles in $R_{v_i}^{es}$ are sorted in decreasing order of their span along the $z$-axis (rectangle $r_1 = [s, s] \times [s, s] \times (-\infty, z_1]$ has a larger span than rectangle $r_2 = [s, s] \times [s, s] \times (-\infty, z_2]$ if $z_1 > z_2$). For simplicity of notation, assume $R_{v_i}^{es}$ itself represents that sorted sequence. Storing the lists $R_{v_i}^{es}$ is not feasible as that would lead to a space consumption of $\Omega(m^2)$ for the entire data structure. Instead, we store an approximation $A(R_{v_i}^{es})$, of the list $R_{v_i}^{es}$:

- The first $[1/\varepsilon]$ entries in $A(R_{v_i}^{es})$ will be the 1-st, 2-nd, $\ldots$, $[1/\varepsilon]$-th largest span rectangle in $R_{v_i}^{es}$.
- The following entries in $A(R_{v_i}^{es})$ will be the $[\varepsilon^{-1}(1 + \varepsilon)]$-th, $[\varepsilon^{-1}(1 + \varepsilon)^2]$-th, $[\varepsilon^{-1}(1 + \varepsilon)^3]$-th, $\ldots$ largest span rectangle in $R_{v_i}^{es}$.

Query Algorithm: For a node $v \in ST$, if $q_z \in xrange(v_i)$ and $q_y \in es$, then we just need to find the predecessor of $q_z$ in $A(R_{v_i}^{es})$ to compute the approximate size of $R_v \cap q$. If the $j$-th element in $A(R_{v_i}^{es})$ is the predecessor of $q_z$, then the rank of the $j$-th element in $R_{v_i}^{es}$ will be a valid approximation of $|R_{v_i} \cap q|$. If no predecessor is found, then 0 is a valid approximation of $|R_{v_i} \cap q|$. Finding the predecessor in $A(R_{v_i}^{es})$ can be done in $O(\log(\varepsilon^{-1} \log m))$ time. For a given point $q(q_x, q_y, q_z)$, let $\Pi$ be the path in $ST$ from the root to the leaf node containing $q_x$. Note that $\Pi = O(\log m / \log(\varepsilon^{-1} \log m))$. Roughly speaking, we have $O(\log(\varepsilon^{-1} \log m))$ time at each node in $\Pi$ to search for the segment $es$ containing $q_y$. Fortunately, this can be done by using the framework of fractional cascading. The final value $t$ reported will be the sum of the approximate value returned at each node in $\Pi$.

Analysis: First we prove that $\tau \in [(1 - \varepsilon)k, k]$. Consider a node $v \in \Pi$ and let $\tau_v$ be the
In this case we report the exact value of $|R_v \cap q|$.

(i) When $i < \lceil 1/\epsilon \rceil$: In this case we report the exact value of $|R_v \cap q|$. Let the predecessor found in $A(R_v^\epsilon)$ be the $i$-th entry. We look at various ranges of $i$ and handle each of them separately:

(ii) When $i \geq \lceil 1/\epsilon \rceil$: In this case we report $\tau_v = \lceil \epsilon^{-1}(1 + \epsilon)^{i-\lceil 1/\epsilon \rceil} \rceil$. By our construction of $A(R_v^\epsilon)$, it should be clear that $\tau_v \leq |R_v \cap q|$. Now we show that $\tau_v \geq (1 - \epsilon)|R_v \cap q|.$

$$|R_v \cap q| \leq \lceil \epsilon^{-1}(1 + \epsilon)^{i-\lceil 1/\epsilon \rceil}+1 \rceil - 1$$ since the $i+1$-th element in $A(R_v^\epsilon)$ is the successor

$$\leq \left(\epsilon^{-1}(1 + \epsilon)^{i-\lceil 1/\epsilon \rceil}+1 \right) - 1$$

$$= \epsilon^{-1}(1 + \epsilon)^{i-\lceil 1/\epsilon \rceil}+1$$

$$\leq (1 + \epsilon) \lceil \epsilon^{-1}(1 + \epsilon)^{i-\lceil 1/\epsilon \rceil} \rceil$$

which implies that $\tau_v \geq (1 - \epsilon)|R_v \cap q|$.

Therefore, we have shown that $\tau_v \in [(1 - \epsilon)|R_v \cap q|, |R_v \cap q|]$. Since, $\tau = \sum_{v \in \Pi} \tau_v$, we conclude that $\tau = [(1 - \epsilon)k, k]$.

Next, we analyze the size of our data structure. For a given $R_v$, the total size of all the approximate lists will be

$$\sum_{v \in s} |A(R_v^\epsilon)| = O(\epsilon^{-1}|R_v| \log m) = O(\epsilon^{-1}|R_v| \log m)$$

For any given node $v \in ST$, since it has $f$ children, the total size of all the approximate lists stored at node $v$ will be:

$$O(f \epsilon^{-1}|R_v| \log m) = O(\epsilon^{-2}|R_v| \log^2 m)$$

The total size of all the secondary structures in $ST$ will be:

$$\sum_{v \in ST} O(\epsilon^{-2}|R_v| \log^2 m) \leq \epsilon^{-2} \log^2 m \sum_{v \in ST} O(|R_v|)$$

$$\leq (\epsilon^{-2} \log^2 m) \cdot O(m \log m)$$

since height of tree is bounded by $O(\log m)$

$$= O(\epsilon^{-2} m \log^3 m)$$

Finally we analyze the query time. Since the height of the tree is $O\left(\frac{\log m}{\log(\epsilon^{-1} \log m)}\right)$, finding the elementary segment $es$ at all nodes in $\Pi$ can be done in $O\left(\frac{\log m}{\log(\epsilon^{-1} \log m)} \times O(\log(\epsilon^{-1} \log m))^2\right) = O(\log m)$ time. Then $O(\log(\epsilon^{-1} \log m))$ time is spent to find the predecessor in $A(R_v^\epsilon)$ at each node in $\Pi$. Therefore, the overall query time is bounded by $O(\log m)$.

### 9 Proof of Lemma 8

Construct a random sample $R$ where each object of $S$ is picked with probability $1/\delta$. Therefore, the expected size of $R$ is $n/\delta$ (if the size of $R$ exceeds $O(n/\delta)$ then we re-sample till we get
the desired size). For a given query \( q \), \( E[|R \cap q|] = |S \cap q|/\delta = k/\delta \). Therefore, by Chernoff bound \([26]\) we observe that

\[
\Pr\left[\left| |R \cap q| - \frac{k}{\delta} \right| > \frac{\varepsilon k}{\delta}\right] \leq e^{-\Omega((\varepsilon^2 k)/\delta)} \leq e^{-\Omega(C\varepsilon^2 \log n)} \leq e^{-\Omega(C\log n)} = n^{-\Omega(C)} \leq o(1/n^C)
\]

We will pick a \( C \) such that \( C > c_1 \). Then observe that, as there are only \( O(n^{c_1}) \) number of combinatorially different query objects on the set \( S \), by union bound it follows that there exists a subset \( R \subseteq S \) of size \( O(n/\delta) \) such that \( |k - |R \cap q| \cdot \delta| \leq \varepsilon k \).

10 Analysis of Theorem 6

Since \( |R| = O(n/\log^3 n) \), by Theorem \([7]\) the space occupied by our data structure will be \( O(\varepsilon^{-3} n) \). By Theorem \([7]\) the query time will be \( O(\log n) \). Next, we will prove that \( (1 - \varepsilon)k \leq \tau_R \log^3 n \leq (1 + \varepsilon)k \).

If we knew the exact count of \( |R \cap q| \), then from Lemma \([8]\), we can infer that:

\[
(1 - \varepsilon')k \leq |R \cap q| \log^3 n \leq (1 + \varepsilon')k
\]

However, by using Theorem \([7]\) we only get the following approximation of \( |R \cap q| \):

\[
(1 - \varepsilon')|R \cap q| \leq \tau_R \leq (1 + \varepsilon')|R \cap q|
\]

Combining the above two equations, we get the following:

\[
(1 - \varepsilon')^2 k \leq (1 - \varepsilon')|R \cap q| \log^3 n \leq \tau_R \log^3 n \leq (1 + \varepsilon')|R \cap q| \log^3 n \leq (1 + \varepsilon')^2 k
\]

\[
\implies (1 + \varepsilon'^2 - 2\varepsilon')k \leq \tau_R \log^3 n \leq (1 + \varepsilon'^2 + 2\varepsilon')k
\]

\[
\implies (1 - \varepsilon)k \leq \tau_R \log^3 n \leq (1 + \varepsilon)k \quad \text{where } \varepsilon = 4\varepsilon'
\]

This finishes the proof of Theorem \([9]\).

11 Proof of Lemma 9

Construct a random sample \( R \) where each object in \( S \) is picked independently with probability \( 1/\delta \). For a given query \( q \), \( E[|R \cap q|] = |S \cap q|/\delta = k/\delta \). Therefore, by Chernoff bound \([26]\) we observe that

\[
\Pr\left[\left| |R \cap q| - \frac{k}{\delta} \right| > \frac{\varepsilon k}{\delta}\right] \leq e^{-\Omega((\varepsilon^2 k)/\delta)} \leq e^{-\Omega(C\varepsilon^2 \log n)} < e^{-\Omega(C)} < 1
\]

by assuming sufficiently large \( C \).

12 Proof of Theorem 15(b)

It is straightforward to obtain a data structure with \( S_{\text{app}} = O(n \log n) \), \( Q_{\text{app}} = O(\log n) \) and \( C = 16 \). Simply build the standard binary range tree and at each node build an instance of Lemma \([16]\) based on the points in its subtree. Given a 4-sided query rectangle \( q \), it can be broken down into two 3-sided query rectangles. Shi and Jaja \([34]\) presented a reporting structure with \( S_{\text{rep}} = O(n \log n) \) and \( Q_{\text{rep}} = O(\log n) \). Plugging in these values into Theorem \([11]\) proves Theorem 15(b).
13 Second General Reduction

In this section we will present our second general reduction. For the sake of completeness, we will repeat some of the discussion from Section 4.

Theorem 20. For a given colored-GIQ problem, assume that we are given a colored reporting structure of $S_{\text{rep}}(n)$ size which can solve the query in $O(Q_{\text{rep}}(n) + \kappa)$ time. We also assume the following two facts: (a) $S_{\text{rep}}(n)$ is converging, and (b) the GIQ problem is polynomially bounded.

Then, one can obtain a $(1 + \varepsilon)$ approximation using a structure of $S_{\varepsilon\text{app}}(n) = O(S_{\text{rep}}(n))$ size and query time $Q_{\varepsilon\text{app}}(n)$ with either

$$Q_{\varepsilon\text{app}}(n) = O\left((Q_{\text{rep}}(n) + \varepsilon^{-2} \cdot \log n) \cdot \log(\log_{1+\varepsilon}|C|)\right)$$

(6)

$$Q_{\varepsilon\text{app}}(n) = O\left((Q_{\text{rep}}(n) + \varepsilon^{-2} \cdot \log n) \cdot \log(\log_{1+\varepsilon}|C|)\right)$$

(7)

Observe that in the first case, the query time is sensitive the value of $k$.

13.1 Handling small values of $k$

Based on the set $S$ we build a colored reporting structure. Given a query object $q$, we query the colored reporting structure to keep reporting the colors in $S \setminus q$ till one of the following event happens: either all the colors in $S \setminus q$ have been reported or $\varepsilon^{-2} \log n + 1$ colors in $S \setminus q$ have been reported. If the first event happens, then we have succeeded in obtaining the exact value of $k$. On the other hand, if the second event happens, then we can conclude that $k = \Omega(\varepsilon^{-2} \log n)$ (i.e., $k$ is large). This query algorithm takes $O(Q_{\text{rep}}(n) + \varepsilon^{-2} \log n)$ time.

13.2 Decision query

From now on we can safely assume that $k = |C \cap q| = \Omega(\varepsilon^{-2} \log n)$. We start off by solving a decision problem: Given a number $z = \Omega(\varepsilon^{-2} \log n)$, is $|C \cap q| \geq z$ or $|C \cap q| < z$? The data structure is allowed to make a mistake when $|C \cap q| \in [(1 - \varepsilon)z, (1 + \varepsilon)z]$.

A few words on the intuition behind our solution. Suppose each color in $C$ is sampled with probability $\approx (\log n)/z$. For a given query $q$, if $k < z$ (resp. $k > z$), then the expected number of colors from $C \cap q$ sampled will be less than $\log n$ (resp. greater than $\log n$). This intuition is converted into an algorithm. We will start by proving the following lemma.

Lemma 21. Let $c_1$ be a sufficiently large constant and $c$ be another constant s.t. $c = \Theta(c_1 \log \varepsilon)$. Consider a random sample $R$ where each color in $C$ is picked independently with probability $M = \frac{\Omega(\varepsilon^{-2} \log n)}{z}$. Then

$$\Pr\left(|R \cap q| > zM \mid k \leq (1 - \varepsilon)z\right) \leq \frac{1}{n^c}.$$

Similarly,

$$\Pr\left(|R \cap q| \leq zM \mid k \geq (1 + \varepsilon)z\right) \leq \frac{1}{n^c}.$$

Here $R \cap q$ denotes the set of colors in $R$ which have at least one point inside $q$. 
Proof. For each of the $k$ colors of $C$ which intersect $q$ define an indicator variable $X_i$. Set $X_i = 1$ if the corresponding color is in the random sample $R$; otherwise set $X_i = 0$. Then $|R \cap q| = \sum_{i=1}^{k} X_i$ and $E[|R \cap q|] = k \cdot M$. For the sake of brevity, we define $Y = |R \cap q|$. We only prove the first fact here. The proof for the second fact is similar. We will divide the proof into two cases based on the value of $\varepsilon$.

Case 1, $\varepsilon \in (0, 1/2)$: Let

$$\alpha = \Pr[Y > zM \mid k \leq (1 - \varepsilon)z]$$

The value $\alpha$ is maximized when $k = |C \cap q| = (1 - \varepsilon)z$. Therefore,

$$\alpha \leq \Pr[Y > zM \mid k = (1 - \varepsilon)z]$$

In this case, $E[Y] = kM = (1 - \varepsilon)zM$. Therefore,

$$\alpha \leq \Pr[Y > zM] = \Pr[Y > \frac{1}{1 - \varepsilon}E[Y]] \leq \Pr[Y > (1 + \varepsilon)E[Y]]$$

$$\leq \exp\left(-\frac{\varepsilon^2 E[Y]}{4}\right)$$

By Chernoff bound

$$= \exp\left(-\varepsilon^2 (1 - \varepsilon)z \left(\frac{c_1 \varepsilon^{-2} \log n}{4z}\right)\right) = \exp\left(-c_1 (1 - \varepsilon) \frac{\log n}{4}\right) \leq \exp\left(-\frac{c_1}{8} \log n\right)$$

since $\varepsilon \leq 1/2$

Case 2, $\varepsilon \in (1/2, 1)$: In this case we ignore the value of $\varepsilon$ and set a new variable $\varepsilon_{\text{new}} \leftarrow 1/2$. The entire solution is built assuming that the error parameter is $\varepsilon_{\text{new}}$. Since $\varepsilon_{\text{new}} < \varepsilon$, clearly the error produced by this data structure will be within the tolerable limits. Also, observe that $\frac{1}{\varepsilon_{\text{new}}} \leq \frac{2}{\varepsilon}$. Therefore, the space and the query time bounds are also not affected.

\[\text{Lemma 22.}\] Using notation from Section 4, a sample $R \subseteq C$ is called suitable if

- For all queries, (a) if $k < (1 - \varepsilon)z$ then $|R \cap q| < c_1 \varepsilon^{-2} \log n$, and (b) if $k \geq (1 + \varepsilon)z$ then $|R \cap q| \geq c_1 \varepsilon^{-2} \log n$.
- $n_R \leq 10nM$.

Such an $R$ always exists.

Proof. The proof is exactly the same as the proof in Lemma 13. The only difference is that we replace Lemma 12 with Lemma 21.

\[\text{Lemma 23.}\] The decision query can be solved in query time $O(Q_{\text{rep}}(n) + \varepsilon^{-2} \log n)$ in the worst-case. The answer $\tau$ lies in the range $[(1 - \varepsilon)k, (1 + \varepsilon)k]$ with probability 1.

Proof. In the preprocessing stage we pick a random sample $R \subseteq C$ as stated in Lemma 21. If the sample $R$ is not suitable, then we discard $R$ and re-sample, till we get a suitable sample. Based on all the points of $S$ whose color belongs to $R$, we build a colored reporting structure. Given a query object $q$, we query the colored reporting structure to keep reporting the colors in $R \cap q$, till one of the following event happens: either all the colors in $R \cap q$ have been reported or $c_1 \varepsilon^{-2} \log n$ colors have been reported. If the first event happens, then we report $k < z$; else, we report $k \geq z$. The query time is bounded by $O(Q_{\text{rep}}(n) + \varepsilon^{-2} \log n)$.
Approximate Range Counting Revisited

Overall Data Structure. Recall that we only have to handle \( k = \Omega(\varepsilon^{-2} \log n) \). For the values \( z_i = c_i(\varepsilon^{-2} \log n)(1 + \varepsilon)^i \), for \( i = 1, 2, 3, \ldots, W = O(\log_{1+\varepsilon} |C|) \), we build a data structure \( D_i \) using Lemma \[23\].

Query algorithm: For a moment, assume that we query all the data structures \( D_1, \ldots, D_W \). Then we will see a sequence of structures \( D_j \) for \( j \in [1, i] \) claiming \(|C \cap q| > z_j\), followed by a sequence of structures \( D_{i+1}, \ldots, D_W \) claiming \(|C \cap q| \leq z_j\). Then we shall set \( \tau \leftarrow z_i \) as the answer to the approximate colored counting query. A simple calculation reveals that \( \tau \) will lie in the range \([ (1 - \varepsilon)k, (1 + \varepsilon)k] \). We propose two different query algorithms:

1. Query the data structures in the order \( D_W, D_{W-1}, D_{W-2}, \ldots, D_i \) to find the index \( i \). Since \( D_i \) is the largest-size data structure queried, the overall query time will be bounded by

\[
\begin{align*}
\text{# structures queried} & \leq (W - i) \cdot O\left( Q_{\text{rep}}(\frac{n \varepsilon^{-2} \log n}{z_i}) + \varepsilon^{-2} \log n \right) \\
& \leq (\varepsilon^{-1} \log n) \cdot O\left( Q_{\text{rep}}\left( \frac{n}{k} (\varepsilon^{-2} \log n) \right) + (\varepsilon^{-2} \log n) \right) \quad \text{since } z_i \geq (1 - \varepsilon)k \geq k/2
\end{align*}
\]

This proves the first bullet of Theorem \[20\].

2. Perform binary search on \( D_1, \ldots, D_W \) to efficiently find the index \( i \). The overall query time will be \( O((Q_{\text{rep}}(n) + \varepsilon^{-2} \log n) \cdot \log(\varepsilon^{-1} \log n)) \). This proves the second bullet of Theorem \[20\].

Space analysis: By performing similar analysis as in Section \[4\] the overall size of the decision query structures can be bounded by \( O(S_{\text{rep}}(n)) \).

Applications of Theorem \[20\] (Second General Reduction)

In this section present a few applications of Theorem \[20\]. For the colored problems discussed in this section, their exact counting structures are very expensive. If we are willing for an approximate count, then it is possible to obtain the following attractive bounds.

- **Theorem 24. 1.** For approximate colored halfspace range search in \( \mathbb{R}^2 \), there is a data structure of \( O(n) \) size which can answer the query in \( O\left( \frac{1}{\varepsilon^2} \cdot \log n \cdot \log(\log_{1+\varepsilon} |C|) \right) \) time.

- **Theorem 24. 2.** For approximate colored halfspace range search in \( \mathbb{R}^3 \), there is a data structure of \( O(n \log n) \) size which can answer the query in \( O\left( \frac{1}{\varepsilon^2} \cdot \log^3 n \cdot \log(\log_{1+\varepsilon} |C|) \right) \) time.

- **Theorem 24. 3.** For approximate orthogonal colored range search in \( \mathbb{R}^d \), there is a data structure of \( O(n \log^d n) \) size which can answer the query in \( O\left( \frac{1}{\varepsilon^2} \cdot \log^{d+1} n \cdot \log(\log_{1+\varepsilon} |C|) \right) \) time.

Colored orthogonal range search in \( \mathbb{R}^d \). In this problem the input is a set of \( n \) colored points in \( \mathbb{R}^d \) and the query is an axis-parallel rectangle in \( \mathbb{R}^d \). First consider the standard orthogonal range emptiness query in \( \mathbb{R}^d \) (\( d \geq 2 \)). Using range trees, this problem can be solved using \( M(n) = O(n \log^{d-1} n) \) space and \( Q_{\text{rep}}(n) = O(\log^{d-1} n) \) query time. Using this structure, the colored orthogonal range reporting problem in \( \mathbb{R}^d \) can be answered in \( O(Q_{\text{rep}}(n) + \kappa Q_{\text{rep}}(n) \log n) \) query time using a structure of size \( O(M(n) \log n) \). Here \( \kappa \) is the number of colors reported (see Section 1.3.4 of \[13\] for the details of this transformation).

By applying Theorem \[20\], the space occupied by the approximate counting structure will be \( O(n \log^d n) \). In Theorem \[20\], we assumed that the query time of the colored reporting structure can be expressed as \( O(Q_{\text{rep}} + \kappa) \), whereas for this problem the query time is being...
expressed as $O(Q_{\text{rep}} + \kappa Q_{\text{rep}} \log n)$. Therefore, equation 7 of the query time $Q_{\text{app}}(n)$ in Theorem 20 can be rewritten as

$$O\left( (Q_{\text{rep}}(n) + (\varepsilon^{-2} \log n) Q_{\text{rep}}(n) \log n) \cdot \log(\log 1 + \varepsilon |C|) \right)$$

Plugging in the value of $Q_{\text{rep}}(n)$ into the above expression, we get

$$Q_{\text{app}}(n) = O\left( \log^{d-1} n + \varepsilon^{-2} \log^{d+1} n \cdot \log(\log 1 + \varepsilon |C|) \right) = O\left( \varepsilon^{-2} \log^{d+1} n \cdot \log(\log 1 + \varepsilon |C|) \right)$$

**Colored halfspace range search in $\mathbb{R}^3$.** In this problem the input is a set of $n$ points in $\mathbb{R}^3$ and the query is a halfspace in $\mathbb{R}^3$. There exists an $O(n \log^2 n)$ space reporting data structure for this problem which can answer the query in $O(n^{1/2 + \varepsilon} + \kappa)$ time [11]. But we will not use this structure, since for our purpose $\kappa = O(\varepsilon^{-2} \log n)$ and the transformation technique used for the colored orthogonal range search problem will give us a reporting data structure with better bounds. Again, first consider the standard halfspace range emptiness query in $\mathbb{R}^3$. This problem can be solved using $M(n) = O(n)$ space and $Q_{\text{rep}}(n) = O(\log n)$ query time [2]. Using this structure, the colored halfspace range reporting problem in $\mathbb{R}^3$ can be answered in $O(Q_{\text{rep}}(n) + \kappa Q_{\text{rep}}(n) \log n) = O(\varepsilon^{-2} \log^3 n)$ query time using a structure of size $O(M(n) \log n) = O(n \log n)$.

Applying Theorem 20, the space occupied by the approximate counting structure will be $O(n \log n)$. The query time will be

$$O\left( (Q_{\text{rep}}(n) + (\varepsilon^{-2} \log n) Q_{\text{rep}}(n) \log n) \cdot \log(\log 1 + \varepsilon |C|) \right) = O\left( \varepsilon^{-2} \log^3 n \cdot \log(\log 1 + \varepsilon |C|) \right)$$

**Colored halfspace range search in $\mathbb{R}^2$.** In this problem the input is a set of $n$ colored points in $\mathbb{R}^2$ and the query is a halfspace in $\mathbb{R}^2$. A reduction is presented from this problem to the segments-below-point problem: Given a set of $n$ segments in the plane, report all the segments hit by a vertical query ray. Recently, this segments-below-point has been solved by Agarwal, Cheng, Tao and Yi [11] in the context of designing data structures for uncertain data. They present an $O(n)$ space data structure to solve the query in $O(\log n + \kappa)$ time. This implies a solution for colored halfspace range reporting in $\mathbb{R}^2$ with the same bounds. Plugging in this result into Theorem 20, we obtain an $O(n)$ size data structure which can answer the approximate query in $O(\varepsilon^{-2} \log n \cdot \log(\log 1 + \varepsilon |C|))$ time.

**Acknowledgements:** I am thankful to Prof. Ravi Janardan, Yuan Li, Sivaramakrishnan Ramamoorthy, Stavros Sintos, Jie Xue for fruitful discussions on these problems and for comments on the previous drafts. I would also like to thank the anonymous referees whose comments helped in immensely improving the content and the presentation of the paper.