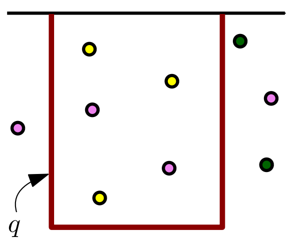
APPROXIMATE RANGE COUNTING REVISITED [*]

Saladi Rahul [†]

Abstract. We study range-searching for colored objects, where one has to count (approximately) the number of colors present in a query range. The problems studied mostly involve orthogonal range-searching in two and three dimensions, and the dual setting of rectangle stabbing by points. We present optimal and near-optimal solutions for these problems. Most of the results are obtained via reductions to the approximate uncolored version, and improved data-structures for them. An additional contribution of this work is the introduction of nested shallow cuttings.

1 Introduction

Let S be a set of n geometric objects in $\mathbb{R}^d$ which are segregated into disjoint groups (i.e., *colors*). Given a query $q \subseteq \mathbb{R}^d$, a color c *intersects (or is present in)* q if any object in S of color c intersects q, and let k be the number of colors of S present in q. In the *approximate colored range-counting problem*, the task is to preprocess S into a data structure, so that for a query q, one can efficiently report the *approximate* number of colors present in q. Specifically, return any value in the range $[(1-\varepsilon)k, (1+\varepsilon)k]$, where $\varepsilon \in (0,1)$ is a pre-specified parameter.



Colored range searching and its related problems have been studied before [31, 39, 40, 37, 32, 30, 23, 13, 8, 24, 27, 29, 33, 34, 38, 43]. They are known as GROUP-BY queries in the database literature. A popular variant is the *colored orthogonal range searching* problem: S is a set of n colored points in $\mathbb{R}^d$, and q is an axes-parallel rectangle. As a motivating example for this problem, consider the following query: "How many countries have employees aged between X_1 and X_2 while earning annually more than Y rupees?". An employee is represented as a colored point $(age, salary)$, where the color encodes the country, and the query is the axes-parallel rectangle $[X_1, X_2] \times [Y, \infty)$.

1.1 Previous work and background

In the *standard* approximate range counting problem there are no colors. One is interested in the approximate number of objects intersecting the query. Specifically, if k is the number of objects of S intersecting q, then return a value in the range $[(1-\varepsilon)k, (1+\varepsilon)k]$.

[*]A preliminary version of this paper appeared at *33rd International Symposium on Computational Geometry (SoCG 2017).*

[†]*Department of Computer Science and Automation, Indian Institute of Science, Bangalore,* saladi@iisc.ac.in

ε-approximations. In the *additive-error ε-approximation*, a set $Z \subseteq S$ is picked such that, given a query q, we *only* inspect Z and return a value which lies in the range $[k - \varepsilon n, k + \varepsilon n]$. Vapnik and Chervonenkis [47] proved that a random sample Z of size $O(\frac{\delta}{\varepsilon^2} \log \frac{\delta}{\varepsilon})$ provides an ε-approximation with good probability, where δ is the VC-dimension (δ is usually a constant for standard geometric settings).

Relative (p, ε)-approximation. Har-Peled and Sharir [25] introduced the notion of *relative (p, ε)-approximation* for geometric settings. The goal is to pick a *small* set $Z \subset S$ which can be used to compute a relative approximation for queries with large value of k. Formally, given a parameter $p \in (0, 1)$, a set $Z \subset S$ is a relative (p, ε)-approximation if:

$$|Z \cap q| \cdot \frac{n}{|Z|} \in \begin{cases} [(1-\varepsilon)k, (1+\varepsilon)k] & \text{if } k \geq pn \\ [k - \varepsilon pn, k + \varepsilon pn] & \text{otherwise.} \end{cases}$$

Har-Peled and Sharir prove that a sample Z from S of size $O\left(\frac{1}{\varepsilon^2 p}\left(\delta \log \frac{1}{p} + \log \frac{1}{q}\right)\right)$ will succeed with probability at least $1 - q$.

Har-Peled and Sharir construct relative (p, ε)-approximations for point sets and halfspaces in $\mathbb{R}^d$, for $d \geq 2$, and use them to answer approximate counting for any query which contains more than pn points. A nice feature of these results is that they are *sensitive* to the value of k. Specifically, the larger the value of k is, the faster the query is answered. The intuition is that the larger the value of k is, the larger is the error the query is allowed to make and hence, a smaller sample suffices. Even though relative (p, ε)-approximations give a relative approximation only for queries with large values of k, Aronov and Sharir [11], and Sharir and Shaul [42] incorporated them into data structures which give an approximate count for all values of k.

General reduction to companion problems. Aronov and Har-Peled [10], and Kaplan, Ramos and Sharir [28] presented general techniques to answer approximate range counting queries. In both instances, the authors reduce the task of answering an approximate counting query, into answering a few queries in data-structures solving an easier *(companion)* problem. Aronov and Har-Peled's companion problem is the emptiness query, where the goal is to report whether $|S \cap q| = 0$. Specifically, assume that there is a data structure of size $S(n)$ which answers the emptiness query in $O(Q(n))$ time. Aronov and Har-Peled show that there is a data structure of size $O(S(n) \log n)$ which answers the approximate counting query in $O(Q(n) \log n)$ time (for simplicity we ignore the dependency on ε). Kaplan *et al.*'s companion problem is the range-minimum query, where each object of S has a weight associated with it and the goal is to report the object in $S \cap q$ with the minimum weight.

Even though the reductions of [10] and [28] seem different, there is an interesting discussion in Section 6 of [10] about the underlying "sameness" of both techniques.

Levels. Informally, for a set S of n objects, a *$(\leq t)$-level* of S is the locus of all the points contained in at most t objects of S. A t-level is then defined as the boundary between the $(\leq t)$-level and the $(\leq t{+}1)$-level of S. Range counting can be reduced in some cases to deciding the level of a query point. Unfortunately, the complexity of a single level is not well understood. For example, for hyperplanes in the plane, the t-level has super-linear complexity $\Omega(n2^{\sqrt{\log t}})$ [45] in the worst-case (the known upper bound is $O(nt^{1/3})$ [19] and closing the gap is a major open problem). In particular, the prohibitive complexity of such levels makes them inapplicable for the approximate range counting problem, where one shoots for linear (or near-linear) space data-structures.

Shallow cuttings. A *t-level shallow cutting* is a set of simple cells which cover the $(\leq t)$-level and are strictly inside the $(\leq O(t))$-level. For many geometric objects in two and three dimensions, such t-level shallow cuttings have $O(n/t)$ cells [6]. Using such cuttings leads to efficient data-structures for approximate range counting. Specifically, one uses binary search on a "ladder" of approximate levels (realized via shallow cuttings) to find the approximation.

For halfspaces in $\mathbb{R}^3$, Afshani and Chan [2] avoid doing the binary search and find the two consecutive levels in optimal $O(\log \frac{n}{k})$ expected time. Later, Afshani, Hamilton and Zeh [4] obtained a worst-case optimal solution for many geometric settings. Interestingly, their results hold in the pointer machine model, the I/O-model and the cache-oblivious model. However, in the word-RAM model their solution is not optimal and the query time is $\Omega(\log\log U + (\log\log n)^2)$.

Specific problems. Approximate counting for orthogonal range searching in $\mathbb{R}^2$ was studied by Nekrich [38], and Chan and Wilkinson [16] in the word-RAM model. In this setting, the input set is points in $\mathbb{R}^2$ and the query is a rectangle in $\mathbb{R}^2$. A hyper-rectangle in $\mathbb{R}^d$ is $(d+k)$*-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. Nekrich [38] presented a data structure for approximate colored 3-sided range searching in $\mathbb{R}^2$, where the input is points and the query is a 3-sided rectangle in $\mathbb{R}^2$. However, it has 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 an approximate colored counting query.

1.2 Motivation

Avoiding expensive counting structures. A search problem is decomposable if given two disjoint sets of objects S_1 and S_2, the answer to $F(S_1 \cup S_2)$ can be computed in constant time, given the answers to $F(S_1)$ and $F(S_2)$ separately. This property is widely used in the literature [7] for counting in standard problems (going back to the work of Bentley and Saxe [12] in the late 1970s). For colored counting problems, however, $F(\cdot)$ is not decomposable. If $F(S_1)$ (resp. $F(S_2)$) has n_1 (resp. n_2) colors, then this information is insufficient to compute $F(S_1 \cup S_2)$, as they might have common colors.

As a result, for *many exact* colored counting queries the known space and query time bounds are expensive. For example, for colored orthogonal range searching problem in $\mathbb{R}^d$, existing structures use $O(n^d)$ space to achieve polylogarithmic query time [29]. Any substantial improvement in the preprocessing time *and* the query time would lead to a substantial improvement in the best exponent of matrix multiplication [29] (which is a major open problem). Similarly, counting structures for colored halfspace counting in $\mathbb{R}^2$

and $\mathbb{R}^3$ [23] are expensive.

Instead of an exact count, if one is willing to settle for an approximate count, then this work presents a data structure with $O(n \text{ polylog } n)$ space and $O(\text{polylog } n)$ query time.

Approximate counting in the speed of emptiness. In an emptiness query, the goal is to decide if $S \cap q$ is empty. The approximate counting query is at least as hard as the emptiness query: When $k = 0$ and $k = 1$, no error is tolerated. Therefore, a natural goal while answering approximate range counting queries is to match the bounds of its corresponding *emptiness query*.

1.3 Our results and techniques

1.3.1 Specific problems

The focus of the paper is building data structures for approximate colored counting queries, which exactly match or *almost* match the bounds of their corresponding emptiness problem.

3-sided rectangle stabbing in 2d and related problems. In the colored interval stabbing problem, the input is n colored intervals with endpoints in $[\![U]\!] = \{1, \ldots, U\}$, and the query is a point in $[\![U]\!]$. We present a linear-space data structure which answers the approximate counting query in $O(\log \log U)$ time. The new data structure can be used to handle some geometric settings in 2d: the *colored dominance search* (the input is a set of n colored points, and the query is a 2-sided rectangle) and the *colored* 3*-sided rectangle stabbing* (the input is a set of n colored 3-sided rectangles, and the query is a point). The results are summarized in Table 1.

Range searching in $\mathbb{R}^2$. The input is a set of n colored points in the plane. For 3-sided query rectangles, an *optimal* solution (in terms of n) for approximate counting is obtained. For 4-sided query rectangles, an *almost-optimal* solution for approximate counting is obtained. The size of our data structure is off by a factor of $\log \log n$ w.r.t. its corresponding emptiness structure which occupies $O(n\frac{\log n}{\log \log n})$ space and answers the emptiness query in $O(\log n)$ time [17]. The results are summarized in Table 1.

Dominance search in $\mathbb{R}^3$. The input is a set of n colored points in $\mathbb{R}^3$ and the query is a 3-sided rectangle in $\mathbb{R}^3$ (i.e., an octant). An almost-optimal solution is obtained requiring $O(n \log \log n)$ space and $O(\log n)$ time to answer the approximate counting query.

For the sake of completeness, in Section 7 we present results for a couple of other colored problems which have expensive exact counting structures. One shortcoming of our results is the preprocessing time. Except for Theorem 1, the preprocessing time for all the other results is $n^{O(d)}$.

Dime--nsion	Input, Query	New Results	Previous Approx. Counting Results	Exact Counting Results	Model
1	intervals, point	S: n, Q: $\log\log U$	S: n, Q: $\log\log U+ (\log\log n)^2$	S: n, Q: $\log\log U + \log_w n$	WR
2	points, 2-sided rectangle				
2	3-sided rectangles, point	Theorem 1	Remark 1	Remark 2	
2	points, 3-sided rectangle	S: n, Q: $\log n$ Theorem 5(A)	S: $n\log^2 n$, Q: $\log^2 n$ Remark 5	not studied	PM
2	points, 4-sided rectangle	S: $n\log n$, Q: $\log n$ Theorem 5(B)	S: $n\log^3 n$, Q: $\log^2 n$ Remark 5	S: $n^2\log^6 n$, Q: $\log^7 n$ Kaplan *et al.* [29]	PM
3	points, 3-sided rectangle	S: $n\log^* n$, Q: $\log n \cdot \log\log n$ Theorem 2	S: $n\log^2 n$, Q: $\log^2 n$ Remark 3	not studied	PM

Table 1: A summary of the results obtained for several approximate colored counting queries. To avoid clutter, the $O(\cdot)$ symbol and the dependency on ε is not shown in the space and the query time bounds. For the second column in the table, the first entry is the input and the second entry is the query. For each of the results column in the table, the first entry is the space occupied by the data structure and the second entry is the time taken to answer the query. All our results are worst case bounds. Finally, WR denotes the word-RAM model and PM denotes the pointer machine model. The appendix has a description of these two models.

1.3.2 General reductions

We present two general reductions for solving approximate colored counting queries by reducing them to "easy" companion queries (Theorem 3 and Theorem 4).

Reduction-I (Reporting + C-approximation). In the first reduction a colored approximate counting query is answered using two companion structures: (a) *reporting structure* (its objective is to report the k colors), and (b) *C-approximation structure* (its objective is to report any value z s.t. $k \in [z, Cz]$, where C is a constant). Significantly, unlike previous reductions [10, 28], there is *no asymptotic loss* of efficiency in space and query time bounds w.r.t. to the two companion problems.

Reduction-II (Only Reporting). The second reduction is a modification of the Aronov and Har-Peled [10] reduction. We present the reduction for the following reasons:

a) Unlike reduction-I, this reduction is “easier" to use since it uses only the reporting structure and avoids the C-approximation structure.
b) The analysis of Aronov and Har-Peled is slightly complicated because of their insistence on querying emptiness structures. We show that by using reporting structures the analysis of our reduction becomes arguably simpler. Our reduction is useful when the reporting query is not significantly costlier than the emptiness query.

1.3.3 Our techniques

The results are obtained via a non-trivial combination of several techniques. For example, (a) new reductions from colored problems to standard problems, (b) obtaining a linear-space data structure by performing random sampling on a super-linear-size data structure, (c) refinement of path-range trees of Nekrich [38] to obtain an optimal data structure for C-approximation of colored 3-sided range search in $\mathbb{R}^2$, and (d) *random sampling on colors* to obtain the two general reductions.

In addition, we introduce *nested shallow cuttings* for 3-sided rectangles in 2d. The idea of using a hierarchy of cuttings (or samples) is, of course, not new. However, for this specific setting, we get a hierarchy where there is no penalty for the different levels being compatible with each other. Usually, cells in the lower levels have to be clipped to cells in the higher levels of the hierarchy, leading to a degradation in performance. In our case, however, cells of the lower levels are fully contained in the cells of the level above it. The shallow cuttings used in [9] have nested property for cells within the same level, but in our setting the nested property holds across different levels which is crucial in avoid binary search on levels.

Paper organization. In Section 2, we present a solution to the colored 3-sided rectangle stabbing in 2d problem. In Section 3, we present a solution to the colored dominance search in $\mathbb{R}^3$ problem. In Section 4 and 5, the two general reductions are presented. In Section 6, the application of the first reduction to colored orthogonal range search in $\mathbb{R}^2$ problem is shown. In Section 7, applications of the second reduction is shown. Finally, we conclude in Section 8.

2 3-sided Rectangle Stabbing in 2d

The goal of this section is to prove the following theorem.

Theorem 1. *Consider the following three colored geometric settings:*

1. **Colored interval stabbing in 1d**, *where the input is a set S of n colored intervals in one-dimension and the query q is a point. The endpoints of the intervals and the query point lie on a grid $[\![U]\!]$.*

2. **Colored dominance search in 2d**, *where the input is a set S of n colored points in 2d and the query q is a quadrant of the form $[q_x, \infty) \times [q_y, \infty)$. The input points and the point (q_x, q_y) lie on a grid $[\![U]\!] \times [\![U]\!]$.*

3. **Colored 3-sided rectangle stabbing in 2d**, *where the input is a set S of n colored 3-sided rectangles in 2d and the query q is a point. The endpoints of the rectangles and the query point lie on a grid $[\![U]\!] \times [\![U]\!]$.*

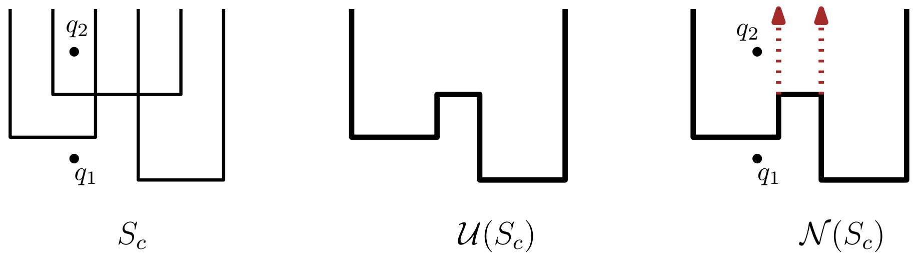
Then there exists an $O_\varepsilon(n)$ size word-RAM data structure which can answer an approximate counting query for these three settings in $O_\varepsilon(\log\log U)$ time. The notation $O_\varepsilon(\cdot)$ hides the dependency on ε. The preprocessing time is $n \cdot \log^{O(1)} n$.

Our strategy for proving this theorem is the following: In Subsection 2.1, we present a transformation of these three colored problems to the *standard* 3-sided rectangle stabbing in 2d problem. Then in Subsection 2.2, we construct nested shallow cuttings and use them to solve the standard 3-sided rectangle stabbing in 2d problem.

2.1 Transformation to a standard problem

From now on the focus will be on colored 3-sided rectangle stabbing in 2d problem, since the geometric setting of (1) and (2) in Theorem 1 are its special cases. We present a transformation of the colored 3-sided rectangle stabbing in 2d problem to the *standard* 3-sided rectangle stabbing in 2d problem.

Let $S_c \subseteq S$ be the set of 3-sided 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 $\mathcal{U}(S_c)$. (2) The vertices of $\mathcal{U}(S_c)$ include original vertices of S_c and some new vertices. Perform a *vertical decomposition* of $\mathcal{U}(S_c)$ by shooting a vertical ray upwards from every *new* vertex of $\mathcal{U}(S_c)$ till it hits $+\infty$. This leads to a decomposition of $\mathcal{U}(S_c)$ into $\Theta(|S_c|)$ pairwise-disjoint 3-sided rectangles. Call these new set of rectangles $\mathcal{N}(S_c)$.



Given a query point q, we can make the following two observations:

- If $S_c \cap q = \emptyset$, then $\mathcal{N}(S_c) \cap q = \emptyset$. See query point q_1 in the above figure.
- If $S_c \cap q \neq \emptyset$, then exactly one rectangle in $\mathcal{N}(S_c)$ is stabbed by q. See query point q_2 in the above figure.

Let $\mathcal{N}(S) = \bigcup_c \mathcal{N}(S_c)$, and clearly, $|\mathcal{N}(S)| = O(n)$. Therefore, the colored 3-sided rectangle stabbing in 2d problem on S has been reduced to the *standard* 3-sided rectangle stabbing in 2d problem on $\mathcal{N}(S)$.

2.2 Standard 3-sided rectangle stabbing in 2d

In this subsection we will prove the following lemma.

Lemma 1. (Standard 3-sided rectangle stabbing in 2d.) *In this geometric setting, the input is a set S of n uncolored 3-sided rectangles of the form $[x_1, x_2] \times [y, \infty)$, and the query q is a point. The endpoints of the rectangles and q lie on a grid $[\![U]\!] \times [\![U]\!]$. Then, there exists a data structure of size $O_\varepsilon(n)$ which can answer an approximate counting query in $O_\varepsilon(\log\log U)$ time.*

By a standard rank-space reduction, the rectangles of S can be projected to a $[\![2n]\!] \times [\![n]\!]$ grid: Let S_x (resp., S_y) be the list of the $2n$ vertical (resp., n horizontal) sides of S in increasing order of their $x-$ (resp., $y-$) coordinate value. Then each rectangle $r = [x_1, x_2] \times [y, \infty) \in S$ is projected to a rectangle $[rank(x_1), rank(x_2)] \times [rank(y), \infty)$, where $rank(x_i)$ (resp., $rank(y)$) is the index of x_i (resp., y) in the list S_x (resp., S_y). Given a query point $q \in [\![U]\!] \times [\![U]\!]$, we can use the van Emde Boas structure [46] to perform a predecessor search on S_x and S_y in $O(\log\log U)$ time to find the position of q on the $[\![2n]\!] \times [\![n]\!]$ grid. Now we will focus on the new setting and prove the following result.

Lemma 2. *For the standard 3-sided rectangle stabbing in 2d problem, consider a setting where q and the rectangles have endpoints lying on a grid $[\![2n]\!] \times [\![n]\!]$. Then there exists a data structure of size $O_\varepsilon(n)$ which can answer the approximate counting query in $O_\varepsilon(1)$ time.*

2.2.1 Nested shallow cuttings

To prove Lemma 2, we will first construct shallow cuttings for 3-sided rectangles in 2d.

Lemma 3. *Let S be a set of 3-sided rectangles (of the form $[x_1, x_2] \times [y, \infty)$) whose endpoints lie on a $[\![2n]\!] \times [\![n]\!]$ grid. A t-*level shallow cutting* of S produces a set $\mathcal{C}$ of interior-disjoint 3-sided rectangles/cells of the form $[x_1, x_2] \times (-\infty, y]$. There exists a set $\mathcal{C}$ with the following three properties:*

1. *$|\mathcal{C}| = 2n/t$.*
2. *If q does not lie inside any of the cell in $\mathcal{C}$, then $|S \cap q| \geq t$.*
3. *Each cell in $\mathcal{C}$ intersects at most $2t$ rectangles of S.*

Proof. Partition the plane into $\frac{2n}{t}$ vertical slabs, such that t vertical lines of S lie in each slab, i.e., each slab has a width of t. See Figure 1(a). Consider a slab $s = [x_1, x_2] \times (-\infty, +\infty)$. Among all the rectangles of S which completely span the slab s, let y_t be the y-coordinate

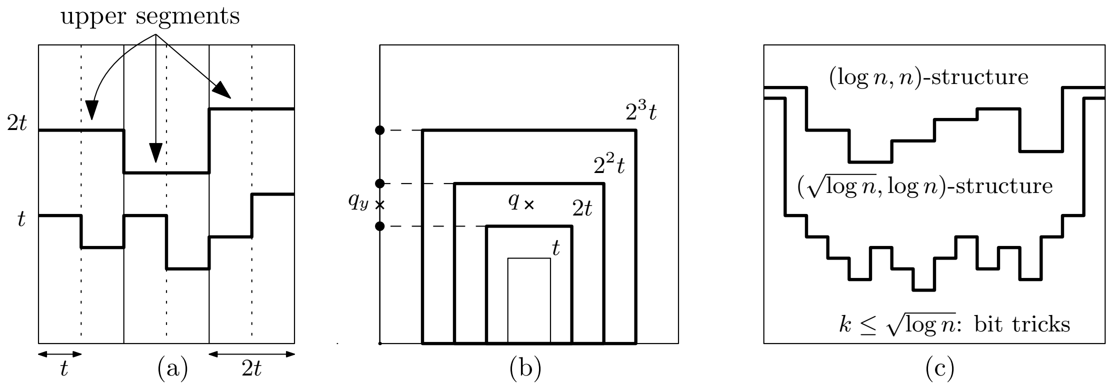


Figure 1: (a) A portion of the t-level and $2t$-level is shown. Notice that by our construction, each cell in the t-level is contained inside a cell in the $2t$-level. (b) A cell in the t-level and the set $\mathcal{C}_r$ associated with it. (c) A high-level summary of our data structure.

of the rectangle with the t-th smallest y-coordinate. If less than t segments of S span slab s, then set $y_t := +\infty$. Let the *upper segment* of the slab s be the horizontal segment $[x_1, x_2] \times [y_t]$. Each slab contributes a cell $[x_1, x_2] \times (-\infty, y_t]$ to set $\mathcal{C}$. See Figure 1(a).

Property 1 is easy to verify, since $\frac{2n}{t}$ slabs are constructed. To prove Property 2, consider a point q which lies in slab s but does not lie in the cell $[x_1, x_2] \times (-\infty, y_t]$. This implies that there are at least t rectangles of S which contain q, and hence, $|S \cap q| \ge t$. To prove Property 3, consider a cell r and its corresponding slab s. The rectangles of S which intersect r either span the slab s or partially span the slab s. By our construction, there can be at most t rectangles of S of each type. □

Observation 1. (Nested Property) *Let t and i be integers. Consider a t-level and a $2^i t$-level shallow cutting. By our construction, each cell in $2^i t$-level contains exactly 2^i cells of the t-level. More importantly, there is only one cell in $2^i t$-level that contains each cell in t-level. (see Figure 1(a)).*

2.2.2 Data structure

Now we will use nested shallow cuttings to find a constant-factor approximation for the 3-sided rectangle stabbing in 2d problem. In [4], the authors show how to convert a constant-factor approximation into a $(1+\varepsilon)$-approximation for this geometric setting. Our solution is based on (t, t')*-level-structure* and $(\le \sqrt{\log n})$*-level lookup table.*

(t, t')-**level structure.** Let i, t and t' be integers s.t. $t' = 2^i t$. If $q(q_x, q_y)$ lies between the t-level and the t'-level cutting of S, then a (t, t')-level-structure will answer the approximate counting query in $O(1)$ time and occupy $O\left(n + \frac{n}{t}\log t'\right)$ space.

Structure. Construct a shallow cutting of S for levels $2^j t, \forall j \in [0, i]$. For each cell, say r, in the t-level we do the following: Let $\mathcal{C}_r$ be the set of cells from the $2^1 t, 2^2 t, 2^3 t, \ldots, 2^i t$-

level, which contain r (Observation 1 guarantees this property). Now project the upper segment of each cell of $\mathcal{C}_r$ onto the y-axis (each segment projects to a point). Based on the y-coordinates of these $|\mathcal{C}_r|$ projected points build a fusion-tree [22]. Since there are $O(n/t)$ cells in the t-level and $|\mathcal{C}_r| = O(\log t')$, the total space occupied is $O(\frac{n}{t}\log t')$. See Figure 1(b).

Query algorithm. Since $q_x \in [\![2n]\!]$, it takes $O(1)$ time to find the cell r of the t-level whose x-range contains q_x. If the predecessor of q_y in $\mathcal{C}_r$ belongs to the $2^j t$-level, then $2^j t$ is a constant-factor approximation of k. The predecessor query also takes $O(1)$ time.

$(\leq \sqrt{\log n})$-level lookup table. Suppose q lies in a cell in the $\sqrt{\log n}$-level shallow cutting of S. Then constructing the $(\leq \sqrt{\log n})$-level lookup table will answer the exact counting query in $O(1)$ time. We will need the following lemma.

Lemma 4. *For a cell c in the $\sqrt{\log n}$-level shallow cutting of S, its* conflict list *S_c is the set of rectangles of S intersecting c. Although the number of cells in the $\sqrt{\log n}$-level is $O\left(\frac{n}{\sqrt{\log n}}\right)$, the number of combinatorially "different" conflict lists is merely $O(\sqrt{n})$.*

Proof. Consider any set S_c from the shallow cutting. By a standard rank-space reduction the endpoints of S_c will lie on a $[\![2|S_c|]\!] \times [\![|S_c|]\!]$ grid. Any set S_c on the $[\![2|S_c|]\!] \times [\![|S_c|]\!]$ grid can be *uniquely* represented using $O(|S_c|\log|S_c|) = O(\sqrt{\log n}\log\log n)$ bits as follows: (a) assign a *label* to each rectangle, and (b) write down the label of each rectangle in increasing order of their y-coordinates. The label for a rectangle $[x_1, x_2] \times [y, \infty)$ will be "x_1x_2" which requires $O(\log\log n)$ bits. The number of combinatorially different conflict lists 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 $\delta < 1$. We set $\delta = 1/2$. □

Lookup table. Construct a $\sqrt{\log n}$-level shallow cutting of S. For each cell c, perform a rank-space reduction of its conflict list S_c. Collect the combinatorially different conflict lists. On each conflict list, the number of combinatorially different queries will be only $O(|S_c|^2) = O(\log n)$. In a lookup table, for each pair of (S_c, q) we store the exact value of $|S_c \cap q|$. The total number of entries in the lookup table is $O(n^{1/2}\log n)$.

Query algorithm. Given a query $q(q_x, q_y)$, the following three $O(1)$ time operations are performed: (a) Find the cell c in the $\sqrt{\log n}$-level which contains q. If no such cell is found, then stop the query and conclude that $k \geq \sqrt{\log n}$. (b) Otherwise, perform a rank-space reduction on q_x and q_y to map it to the $[\![2|S_c|]\!] \times [\![|S_c|]\!]$ grid. Since, $|S_c| = O(\sqrt{\log n})$, we can build fusion trees [22] on S_c to perform the rank-space reduction in $O(1)$ time. (c) Finally, search for (S_c, q) in the lookup table and report the exact count.

Final structure. At first thought, one might be tempted to construct a $(0, n)$-level-structure. However, that would occupy $O(n\log n)$ space. The issue is that the (t, t')-level structure requires super-linear space for small values of t. Luckily, the $(\leq \sqrt{\log n})$-level lookup table will efficiently handle the small values of t.

Therefore, the strategy is to construct the following: (a) a ($\leq \sqrt{\log n}$)-level lookup table, (b) a ($\sqrt{\log n}, \log n$)-level-structure, and (c) a ($\log n, n$)-level-structure. Now, the space occupied by all the three structures will be $O(n)$. See Figure 1(c) for a summary of our data structure.

Remark 1. For the standard 3-sided rectangle stabbing in 2d problem, a simple binary search on the levels leads to a linear-space data structure with a query time of $O_\varepsilon(\log\log U + (\log\log n)^2)$. The technique of Afshani *et al.* [4] can be used to answer this approximate counting query. However, their analysis works well for structures with query time of the form $\log n$ or $\log_B n$, but breaks down for structures with query time of the form $\log\log n$.

Remark 2. If we want an exact count for the standard 3-sided rectangle stabbing in 2d problem, then the problem can be reduced to exact counting for standard dominance search in 2d [20]. Jaja *et al.* [26] present a linear-space structure which can answer the exact counting for dominance search in 2d in $O_\varepsilon(\log\log U + \log_w n)$ time.

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

Theorem 2. *In the colored dominance search in $\mathbb{R}^3$ problem, the input set S is n colored points in $\mathbb{R}^3$ and the query q is a point. Then there is a pointer machine data structure of size $O_\varepsilon(n\log^* n)$ which can answer an approximate colored counting query in $O_\varepsilon(\log n \cdot \log\log n)$ time. The notation $O_\varepsilon(\cdot)$ hides the dependency on ε.*

The strategy to prove this theorem is the following. First, we reduce the colored dominance search in $\mathbb{R}^3$ problem to a *standard* problem of 5-sided rectangle stabbing in $\mathbb{R}^3$. Then in the remaining section we solve the standard 5-sided rectangle stabbing in $\mathbb{R}^3$ problem.

3.1 Reduction to 5-sided rectangle stabbing in $\mathbb{R}^3$

In this subsection we present a reduction of colored dominance search in $\mathbb{R}^3$ problem to the standard 5-sided rectangle stabbing in $\mathbb{R}^3$ problem. 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, and $p_1, p_2, \ldots, p_t$ be the 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 ϕ_i in $\mathbb{R}^3$ which satisfies the following invariant: a point (x, y, z) belongs to ϕ_i if and only if in the region $[x, +\infty) \times [y, +\infty) \times [z, +\infty)$ the point of S_c with the largest z-coordinate is p_i. 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 ϕ_i is unbounded in the negative z-direction. Each region ϕ_i is broken into disjoint 5-sided rectangles via *vertical decomposition* in the xy-plane (see Figure 2). 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 transformed to the standard 5-sided rectangle stabbing query.

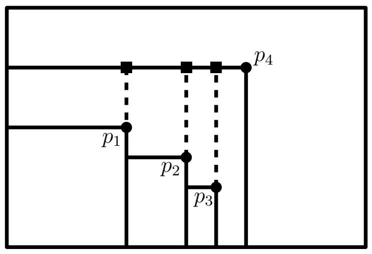


Figure 2: A dataset containing four points. The projection of ϕ_1, ϕ_2, ϕ_3 and ϕ_4 onto the xy-plane is shown. The dashed lines are created during the vertical decomposition. Each rectangle created during the vertical decomposition is lifted back to a 5-sided rectangle in $\mathbb{R}^3$.

3.2 Initial strcuture

Lemma 5. *In the standard* 5*-sided rectangle stabbing in* $\mathbb{R}^3$ *problem, the input is a set* S *of* n 5*-sided rectangles in* $\mathbb{R}^3$ *and the query* q *is a point. Then there exists a pointer machine data structure of size* $O_\varepsilon(n \log\log n)$ *which can answer an approximate counting query in* $O_\varepsilon(\log n \cdot \log\log n)$ *time.*

The rest of the subsection is devoted to proving this lemma.

Recursion tree. Define a parameter $t = \log_{1+\varepsilon} n$. We will assume that the 5-sided rectangles are unbounded along the z-axis. Consider the projection of the rectangles of S on to the xy-plane and impose an orthogonal $[\![2\sqrt{\frac{n}{t}}]\!] \times [\![2\sqrt{\frac{n}{t}}]\!]$ grid such that each horizontal and vertical slab contains the projections of $\sqrt{nt}$ sides of S. Call this the root of the recursion tree. Next, for each vertical and horizontal slab, we recurse on the rectangles of S which are *sent* to that slab. At each node of the recursion tree, if we have m rectangles in the subproblem, then t is changed to $\log_{1+\varepsilon} m$ and the grid size changes to $[\![2\sqrt{\frac{m}{t}}]\!] \times [\![2\sqrt{\frac{m}{t}}]\!]$. We stop the recursion when a node has less than c rectangles, for a suitably large constant c.

Assignment of rectangles. For a node in the tree, the intersection of every pair of horizontal and vertical grid line defines a *grid point.* Each rectangle of S is assigned to $O_\varepsilon(\log\log n)$ nodes in the tree. The assignment of a rectangle to a node is decided by the following three cases:

Case-I. The xy-projection of a rectangle intersects none of the grid points, i.e., it lies completely inside one of the row slab or/and the column slab. Then the rectangle is not assigned to this node, but sent to the child node corresponding to the row or column the rectangle lies in.

Case-II. The xy-projection of a rectangle r intersects at least one of the grid points. Let c_l and c_r be the leftmost and the rightmost column of the grid intersected by r. Similarly, let r_b and r_t be the bottom most and the topmost row of the grid intersected by r.

Then the rectangle is broken into at most five disjoint pieces: a *grid rectangle*, which is the bounding box of all the grid points lying inside r (see Figure 3(b)), two *column rectangles*, which are the portions of r lying in column c_l and c_r (see Figure 3(d)), and two *row rectangles*, which are the remaining portion of the rectangle r lying in row r_b and r_t (see Figure 3(c)). The grid rectangle is *assigned* to the node. Note that each column rectangle (resp., row rectangle) is now a 4-sided rectangle in $\mathbb{R}^3$ w.r.t. the column (resp., row) it lies in, and is sent to its corresponding child node.

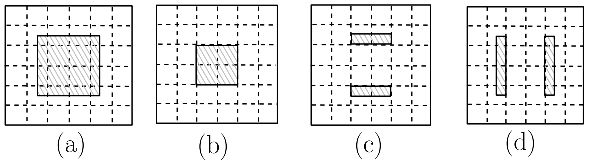


Figure 3:

Case-III. The xy-projection of a 4-*sided rectangle* r intersects at least one of the grid points. Without loss of generality, assume that the 4-sided rectangle r is unbounded along the negative x-axis. Then the rectangle is broken into at most four disjoint pieces: a *grid rectangle,* as shown in Figure 4(b), one *column rectangle*, as shown in Figure 4(d), and two *row rectangles*, as shown in Figure 4(c). The grid rectangle and the two row rectangles are *assigned* to the node. Note that the two row rectangles are now 3-sided rectangles in $\mathbb{R}^3$ w.r.t. their corresponding rows (unbounded in one direction along $x-$, $y-$ and $z-$axis). The column rectangle is sent to its corresponding child node. Analogous partition is performed for 4-sided rectangles which are unbounded along positive x-axis, positive y-axis and negative y-axis.

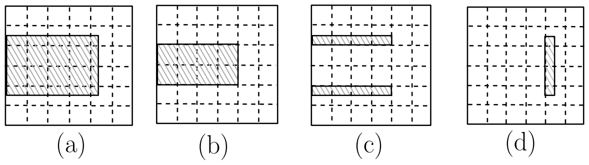


Figure 4:

Observation 2. *A rectangle of S gets assigned to at most four nodes at each level in the recursion tree.*

Proof. Consider a rectangle $r \in S$. If r falls under Case-II, then its grid rectangle is assigned to the node. Note that r can fall under Case-II only once, since each of its four row and column rectangles are now effectively 4-sided rectangles. Let r' be one of these row or column rectangles. If r' falls under Case-III at a node, then it gets assigned there. However, this time exactly *one* of the broken portion of r' will be sent to the child node. Therefore, there can be at most four nodes at each level where rectangle r (and broken portions of r) can get assigned. □

Data structures at each node. We build two types of structures at each node in the tree.

Structure-I. A rectangle r' is *higher* than rectangle r'' if r' has a larger span than r'' along z-direction. For each cell c of the grid, based on the rectangles which completely cover c, we construct a *sketch* as follows: select the rectangle with the $(1+\varepsilon)^0, (1+\varepsilon)^1, (1+\varepsilon)^2, \ldots$-th largest span. For a given cell, the size of the sketch will be $O(\log_{1+\varepsilon} m)$.

Structure-II. For a given row or column in the grid, let $\hat{S}$ be the 3-sided rectangles in $\mathbb{R}^3$ assigned to it. We build the linear-size structure of [4] on $\hat{S}$, which will return a $(1+\varepsilon)$-approximation of $|\hat{S} \cap q|$ in $O_\varepsilon(\log n)$ time. This structure is built for each row and column slab.

Space analysis. Consider a node in the recursion tree with m rectangles. There will be $\left(2\sqrt{\frac{m}{t}}\right) \times \left(2\sqrt{\frac{m}{t}}\right) = 4\frac{m}{t}$ cells at this node. The space occupied by structure-I will be $O\left(\frac{m}{t} \cdot \log_{1+\varepsilon} m\right) = O(m)$. The space occupied by structure-II will be $O(m)$. Using Observation 2, the total space occupied by all the nodes at a particular level will be $O(n)$. Since the height of the recursion tree is $O_\varepsilon(\log\log n)$, the total space occupied is $O_\varepsilon(n \log\log n)$.

Query algorithm. Given a query point q, we start at the root node. At each visited node, the following three steps are performed:

1. *Query structure-I.* Locate the cell c on the grid containing q. Scan the sketch of cell c to return a $(1+\varepsilon)$-approximation of the number of rectangles which cover c and contain q. This takes $O_\varepsilon(\log m)$ time.

2. *Query structure-II.* Next, query structure-II of the horizontal and the vertical slab containing q, to find a $(1+\varepsilon)$-approximation of the 3-sided rectangles containing q. This takes $O_\varepsilon(\log m)$ time.

3. *Recurse.* Finally, we recurse on the horizontal and the vertical slab containing q.

The final output is the *sum* of the count returned by all the nodes queried.

Query time analysis. Let $Q(n)$ denote the overall query time. Then

$$Q(n) = 2Q(\sqrt{nt}) + O_\varepsilon(\log n), t = \log_{1+\varepsilon} n.$$

This solves to $Q(n) = O_\varepsilon(\log n \cdot \log\log n)$. This finishes the proof of Lemma 5.

3.3 Final structure

In this subsection we improve upon the data structure built in the previous subsection by reducing the size to $O_\varepsilon(n \log^* n)$.

Lemma 6. *In the standard* 5*-sided rectangle stabbing in* $\mathbb{R}^3$ *problem, the input is a set* S *of* n 5*-sided rectangles in* $\mathbb{R}^3$ *and the query* q *is a point. Then there exists a pointer machine data structure of size* $O_\varepsilon(n \log^* n)$ *which can solve an approximate counting problem in* $O_\varepsilon(\log n \cdot \log\log n)$ *time.*

Let $\varepsilon' \leftarrow \varepsilon/4$ and $C > 3$. The reason for choosing these parameters will become clear later. We divide the solution into two cases.

3.3.1 Case-I: $k \in [0, C\varepsilon'^{-2} \log n \cdot \log\log n]$

For the reporting version of 5-sided rectangle stabbing in $\mathbb{R}^3$ problem, Rahul [41] presented a structure of size $O(n \log^* n)$ which can answer a query in $O(\log n \cdot \log\log n + k)$ time. Build this structure on all the rectangles in set S. Given a query point q, query the structure till all the rectangles in $S \cap q$ have been reported or $C\varepsilon'^{-2} \log n \cdot \log\log n + 1$ rectangles in $S \cap q$ have been reported. If the first event happens, then the exact value of k is reported. Otherwise, we conclude that $k > C\varepsilon'^{-2} \log n \cdot \log\log n$.

3.3.2 Case-II: $k \in [C\varepsilon^{-2} \log n \cdot \log\log n, n]$

We will need the following random sampling based lemma.

Lemma 7. *Let* S *be a set of* n 5*-sided rectangles in* $\mathbb{R}^3$*. Consider a query point* q *such that* $k \geq C\varepsilon'^{-2} \log n \cdot \log\log n$*. Then there exists a set* $R \subset S$ *of size* $O\left(\frac{n}{\log\log n}\right)$ *such that* $(|R \cap q| \cdot \log\log n) \in [(1-\varepsilon')k, (1+\varepsilon')k]$.

Proof. Fix a parameter $\delta = \log\log n$. Choose a random sample R where each object of S is picked independently with probability $1/\delta$. Therefore, the expected size of R is n/δ (if the size of R exceeds $O(n/\delta)$, then 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 [36] we observe that

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

Set C to be greater than 3. There are $O(n^3)$ combinatorially different query points on the set S. Therefore, by union bound it follows that there exists a subset $R \subset S$ of size $O(n/\delta)$ such that $|k - |R \cap q| \cdot \delta| \leq \varepsilon' k$, for any q such that $k \geq C\varepsilon'^{-2} \log n \cdot \log\log n$. □

Preprocessing steps. We perform the following steps:

- Apply Lemma 7 on set S to obtain a set R of size $O(n/\log\log n)$.
- Build the data structure of Lemma 5 based on set R with error parameter ε'.

Query algorithm. For a given a query q, let τ_R be the value returned by the data structure built on R. Then we report $\tau_R \cdot \log\log n$ as the answer.

Analysis. Since $|R| = O(n/\log\log n)$, by Lemma 5 the space occupied by this data structure will be $O_\varepsilon(n)$. The query time follows from Lemma 5. Next, we will prove that $(1-\varepsilon)k \le \tau_R \cdot \log\log n \le (1+\varepsilon)k$.

If we knew the exact value of $|R \cap q|$, then from Lemma 7 we can infer that:

$$(1-\varepsilon')k \le |R \cap q| \log\log n \le (1+\varepsilon')k \tag{1}$$

However, by using Lemma 5 we only get an approximate value of $|R \cap q|$:

$$(1-\varepsilon')|R \cap q| \le \tau_R \le (1+\varepsilon')|R \cap q| \tag{2}$$

Combining the above two equations, it is easy to verify that $(1-\varepsilon)k \le \tau_R \log\log n \le (1+\varepsilon)k$, where $\varepsilon = 4\varepsilon'$. This finishes the proof of Lemma 6.

Remark 3. The general technique of Aronov and Har-Peled [10] can be adapted to answer the approximate counting query for the colored dominance search in $\mathbb{R}^3$ problem. Assume that we have a data structure of size $S(n)$ which can answer the emptiness query in $Q(n)$ time. Ignoring the dependence on ε, the technique of [10] guarantees a data structure of size $O(S(n)\log^2 n)$ which can answer a colored approximate counting query in $O(Q(n)\log n)$ time ($O(\log^2 n)$ emptiness structures are built with each of them storing $\Theta(n)$ objects in the worst-case). To answer an emptiness query, we can use a standard reporting structure instead of a colored reporting structure, since a non-empty (resp., an empty) intersection of the input objects with the query trivially implies that the number of colors intersecting the query is greater than zero (resp., zero). For colored dominance search in $\mathbb{R}^3$, plugging in $S(n) = O(n)$ and $Q(n) = O(\log n)$ [1] (a standard reporting structure), we get a data structure which requires $O_\varepsilon(n\log^2 n)$ space and $O_\varepsilon(\log^2 n)$ query time.

4 Reduction-I: Reporting + C-approximation

Our first reduction states that given a colored reporting structure and a colored C-approximation structure, one can obtain a colored $(1+\varepsilon)$-approximation structure with no additional loss of efficiency. We need a few definitions before stating the theorem. A geometric setting 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 $\sum_{i=0}^{t} n_i = n$, then $\sum_{i=0}^{t} f(n_i) = O(f(n))$. For example, it is easy to verify that $f(n) = n\log n$ is converging.

Theorem 3. *For a colored geometric setting, assume that we are given the following two structures:*

- *a colored reporting structure of $\mathcal{S}_{rep}(n)$ size which can solve a query in $O(\mathcal{Q}_{rep}(n)+\kappa)$ time, where κ is the output-size, and*
- *a colored C-approximation structure of $\mathcal{S}_{capp}(n)$ size which can solve a query in $O(\mathcal{Q}_{capp}(n))$ time.*

We also assume that: (a) $\mathcal{S}_{rep}(n)$ and $\mathcal{S}_{capp}(n)$ are converging, and (b) the geometric setting is polynomially bounded. Then we can obtain a $(1+\varepsilon)$-approximation using a structure that requires $\mathcal{S}_{\varepsilon app}(n)$ space and $\mathcal{Q}_{\varepsilon app}(n)$ query time, such that

$$\mathcal{S}_{\varepsilon app}(n) = O(\mathcal{S}_{rep}(n) + \mathcal{S}_{capp}(n)) \tag{3}$$
$$\mathcal{Q}_{\varepsilon app}(n) = O\left(\mathcal{Q}_{rep}(n) + \mathcal{Q}_{capp}(n) + \varepsilon^{-2} \cdot \log n\right). \tag{4}$$

4.1 Refinement Structure

The goal of a refinement structure is to convert a constant-factor approximation of k into a $(1+\varepsilon)$-approximation of k.

Lemma 8. (Refinement structure) *Let $\mathcal{C}$ be the set of colors in set S, and $\mathcal{C} \cap q$ be the set of colors in $\mathcal{C}$ present in q. For a query q, assume we know that:*

- *$k = |\mathcal{C} \cap q| = \Omega(\varepsilon^{-2} \log n)$, and*
- *$k \in [z, Cz]$, where z is an integer.*

Then there is a refinement structure of size $O\left(\mathcal{S}_{rep}\left(\frac{\varepsilon^{-2} n \log n}{z}\right)\right)$ which can report a value $\tau \in [(1-\varepsilon)k, (1+\varepsilon)k]$ in $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \log n)$ time.

The following lemma states that sampling colors (instead of input objects) is a useful approach to build the refinement structure.

Lemma 9. *Consider a query q which satisfies the two conditions stated in Lemma 8. Let c_1 be a sufficiently large constant and c be another constant s.t. $c = \Theta(c_1 \log e)$. Choose a random sample R where each color in $\mathcal{C}$ is picked independently with probability $M = \frac{c_1 \varepsilon^{-2} \log n}{z}$. Then with probability $1 - n^{-c}$ we have $\left|k - \frac{|R \cap q|}{M}\right| \leq \varepsilon k$.*

Proof. For each of the k colors which are present in 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,

$$\mathbf{Pr}\left[\Big||R \cap q| - E[|R \cap q|]\Big| > \varepsilon \cdot E[|R \cap q|]\right] < \exp\Big(-\varepsilon^2 E[|R \cap q|]\Big)$$
$$< \exp\left(-\varepsilon^2 \cdot kM\right) < \exp\left(-\varepsilon^2 zM\right) < \exp\left(-c_1 \log n\right) \leq \frac{1}{n^c}$$

Therefore, with high probability $\Big||R \cap q| - kM\Big| \leq \varepsilon \cdot kM$. □

Lemma 10. (Finding a suitable R) *Pick a random sample R as defined in Lemma 9. Let n_R be the number of objects of S whose color belongs to R. We say R is* suitable *if it satisfies the following two conditions:*

- $\left|k - \frac{|R \cap q|}{M}\right| \leq \varepsilon k$ *for all queries which have* $k = \Omega(\varepsilon^{-2} \log n)$.
- $n_R \leq 10nM$. *This condition is needed to bound the size of the data structure.*

A suitable R always exists.

Proof. Let n^α be the number of combinatorially different queries q on the set S. From Lemma 9, by setting $c = \alpha + 1$, we can conclude that $\tau \longleftarrow \frac{|R \cap q|}{M}$ 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 $1/n^{\alpha+1} \times n^\alpha = 1/n$.

Next, it is easy to observe that the *expected* value of n_R is nM: Let n_c be the number of objects of S having color c. Then $\mathbf{E}[n_R] = \sum_c n_c \cdot M = nM$. 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, we are done. 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. □

Refinement structure and query algorithm. In the preprocessing stage pick a random sample $R \subseteq \mathcal{C}$ as stated in Lemma 9. If the sample R is *not suitable*, then discard R and re-sample, till we get a suitable sample. Based on all the objects of S whose color belongs to R, build a colored reporting structure. Given a query q, the colored reporting structure is queried to compute $|R \cap q|$. We report $\tau \longleftarrow (|R \cap q|/M)$ as the final answer. The query time is bounded by $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \log n)$, since by Lemma 9, $|R \cap q| \leq (1+\varepsilon) \cdot kM = O(\varepsilon^{-2} \log n)$. This finishes the description of the refinement structure.

4.2 Overall solution

Data structure. The data structure consists of the following three components:

1. *Reporting structure.* Based on the set S we build a colored reporting structure. This occupies $O(\mathcal{S}_{rep}(n))$ space.
2. *$\sqrt{C}$-approximation structure.* Based on the set S we build a $\sqrt{C}$-approximation structure. The reason for choosing $\sqrt{C}$ will become clear in the analysis. This occupies $O(\mathcal{S}_{capp}(n))$ space.
3. *Refinement structures.* Build the refinement structure of Lemma 8 for the values $z = (\sqrt{C})^i \cdot \varepsilon^{-2} \log n, \forall i \in \left[0, \log_{\sqrt{C}}\left(\lceil \varepsilon^2 n \rceil\right)\right]$. The total size of all the refinement

structures will be $\sum_z O\left(\mathcal{S}_{rep}(nM)\right) = O(\mathcal{S}_{rep}(n))$, since $\mathcal{S}_{rep}(\cdot)$ is converging and $\sum_z nM = O(n)$. Note that our choice of z ensures that the size of the data structure is independent of ε.

Query algorithm. The query algorithm performs the following steps:

1. Given a query object q, the colored reporting structure reports the colors present in $S \cap q$ till all the colors have been reported or $\varepsilon^{-2}\log n + 1$ colors have been reported. If the first event happens, then the exact value of k is reported. Otherwise, we conclude that $k = \Omega(\varepsilon^{-2}\log n)$. This takes $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2}\log n)$ time.
2. If $k > \varepsilon^{-2}\log n$, then
   (a) First, query the $\sqrt{C}$-approximation structure. Let k_a be the $\sqrt{C}$-approximate value returned s.t. $k \in [k_a, \sqrt{C}k_a]$. This takes $O(\mathcal{Q}_{capp}(n))$ time.
   (b) Then query the refinement structure with the largest value of z s.t. $z \le k_a \le \sqrt{C}z$. It is trivial to verify that $k \in [z, Cz]$. This takes $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2}\log n)$ time.

5 Reduction-II: Using Only Reporting Structure

In this section we will present our second general reduction. The reader is assumed to be familiar with Section 4.

Theorem 4. *For a given colored geometric setting, assume that we are given a colored reporting structure of $\mathcal{S}_{rep}(n)$ size which can answer the query in $O(\mathcal{Q}_{rep}(n) + \kappa)$ time. We also assume that: (a) $\mathcal{S}_{rep}(n)$ is converging, and (b) the geometric setting is* polynomially bounded. *Then we can obtain a $(1+\varepsilon)$ approximation using a structure which requires $\mathcal{S}_{\varepsilon app}(n) = O(\mathcal{S}_{rep}(n))$ space and*

$$\mathcal{Q}_{\varepsilon app}(n) = O\bigg(\left(\mathcal{Q}_{rep}(n) + \varepsilon^{-2}\cdot\log n\right)\cdot\log(\log_{1+\varepsilon}|\mathcal{C}|)\bigg) \tag{5}$$

query time, where $\mathcal{C}$ is the number of colors in S.

Similar to Section 4, a colored reporting structure will be built on S to either report the exact value of $|\mathcal{C} \cap q|$ or report that $|\mathcal{C} \cap q|$ is greater than $\varepsilon^{-2}\log n$. From now on we will assume that $k = |\mathcal{C} \cap q| = \Omega(\varepsilon^{-2}\log n)$.

5.1 Decision structure

Lemma 11. (Decision structure) *Let $z = \Omega(\varepsilon^{-2}\log n)$ be a pre-specified parameter. Given a query q, the decision structure reports whether $|\mathcal{C} \cap q| \ge z$ or $|\mathcal{C} \cap q| < z$. The data structure is allowed to make a mistake when $|\mathcal{C} \cap q| \in [(1-\varepsilon)z, (1+\varepsilon)z]$. There is a decision structure of size $O\left(\mathcal{S}_{rep}\left(\frac{\varepsilon^{-2}n\log n}{z}\right)\right)$ which can answer the query in $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2}\log n)$ time.*

In this subsection we will prove the above lemma. A few words on the intuition behind the solution. Suppose each color in $\mathcal{C}$ 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 $\mathcal{C} \cap q$ sampled will be less than $\log n$ (resp., greater than $\log n$). We will start by proving the following lemma.

Lemma 12. *Let c_1 be a sufficiently large constant and c be another constant s.t. $c = \Theta(c_1 \log e)$. Consider a random sample R where each color in $\mathcal{C}$ is picked independently with probability $M = \frac{c_1\varepsilon^{-2}\log n}{z}$, where $\varepsilon \in (0, 1/2]$. Then*

$$\boldsymbol{Pr}\left[|R \cap q| > zM \;\middle|\; k \le (1-\varepsilon)z\right] \le \frac{1}{n^c}.$$

Similarly,

$$\boldsymbol{Pr}\left[|R \cap q| \le zM \;\middle|\; k \ge (1+\varepsilon)z\right] \le \frac{1}{n^c}$$

Proof. For each of the k colors present in 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, let $Y = |R \cap q|$. We only prove the first fact here. The proof for the second fact is similar. Let

$$\alpha = \mathbf{Pr}\left[Y > zM \;\middle|\; k \le (1-\varepsilon)z\right]$$

The value α is maximized when $k = (1-\varepsilon)z$. Therefore,

$$\alpha \le \mathbf{Pr}\left[Y > zM \;\middle|\; k = (1-\varepsilon)z\right]$$

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

$$\begin{aligned}
\alpha &\le \mathbf{Pr}[Y > zM] = \mathbf{Pr}\left[Y > \frac{1}{1-\varepsilon}\mathbf{E}[Y]\right] \le \mathbf{Pr}\left[Y > (1+\varepsilon)\mathbf{E}\left[Y\right]\right] \\
&\le \exp\left(-\frac{\varepsilon^2\mathbf{E}[Y]}{4}\right) \qquad \text{By Chernoff bound} \\
&= \exp\left(-\varepsilon^2(1-\varepsilon)z\left(\frac{c_1\varepsilon^{-2}\log n}{4z}\right)\right) = \exp\left(-c_1(1-\varepsilon)\frac{\log n}{4}\right) \le \exp\left(-\frac{c_1}{8}\log n\right) \qquad \text{since } \varepsilon \le 1/2 \\
&\le \frac{1}{n^c}
\end{aligned}$$

□

Lemma 13. *Let $z = \Omega(\varepsilon^{-2}\log n)$ be a pre-specified parameter. Using notation from Section 4, a sample $R \subseteq \mathcal{C}$ 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 $k \ge (1+\varepsilon)z$ then $|R \cap q| \ge c_1\varepsilon^{-2}\log n$.*

- $n_R \leq 10nM$.

  Such an R always exists.

Proof. The proof is exactly the same as the proof in Lemma 10. The only difference is that we replace Lemma 9 with Lemma 12. □

Decision structure and query algorithm. In the preprocessing phase pick a random sample $R \subseteq \mathcal{C}$ as stated in Lemma 12. If the sample R is *not suitable*, then discard R and re-sample, till we get a suitable sample. Based on all the points of S whose color belongs to R, build a colored reporting structure. Given a query object q, the colored reporting structure reports $R \cap q$, till all the colors have been reported or $c_1\varepsilon^{-2}\log n$ colors have been reported. If the first event happens, then we report $k < z$. Otherwise, we report $k \geq z$. The query time is bounded by $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2}\log n)$,

In Lemma 12, we assumed $\varepsilon \in (0, 1/2]$. Handling $\varepsilon \in (1/2, 1]$ is easy: Set a new variable $\varepsilon_{new} \longleftarrow 1/2$. The decision structure will be built with the error parameter ε_{new} (and not ε). Since $\varepsilon_{new} < \varepsilon$, the error made by the decision structure is tolerable. Since $\frac{1}{\varepsilon_{new}} \leq \frac{2}{\varepsilon}$, the space and the query time bounds are also not affected.

5.2 Final structure

Data structure. Recall that we only have to handle $k = \Omega(\varepsilon^{-2}\log n)$. For the values $z_i = c_1(\varepsilon^{-2}\log n)(1+\varepsilon)^i$, for $i = 1, 2, 3, \ldots, W = O(\log_{1+\varepsilon}|\mathcal{C}|)$, we build a decision structure $\mathcal{D}_i$ using Lemma 11. By performing similar analysis as in Section 4, the overall size will be $O(\mathcal{S}_{rep}(n))$.

Query algorithm. For a moment, assume that we query all the data structures $\mathcal{D}_1, \ldots, \mathcal{D}_W$. Then we will see a sequence of structures $\mathcal{D}_j$ for $j \in [1, i]$ claiming $|\mathcal{C} \cap q| > z_j$, followed by a sequence of structures $\mathcal{D}_{i+1}, \ldots, \mathcal{D}_W$ claiming $|\mathcal{C} \cap q| \leq z_j$. Then we report $\tau \leftarrow z_i$ as the answer to the approximate colored counting query. A simple calculation reveals that τ will lie in the range $[(1-\varepsilon)k, (1+\varepsilon)k]$. We perform a binary search on $\mathcal{D}_1, \ldots, \mathcal{D}_W$ to efficiently find the index i. The query time will be $O\Big(\big(\mathcal{Q}_{rep}(n) + \varepsilon^{-2} \cdot \log n\big) \cdot \log(\log_{1+\varepsilon}|\mathcal{C}|)\Big)$.

Remark 4. *Our result is a generalization of the reduction of Aronov and Har-Peled [10] to colored problems. Handling "small" values of k efficiently is usually challenging, since the error tolerated is small. Using the reporting structure makes it easy to handle the "small" values of k (unlike an emptiness structure which was used by [10]). Random sampling and Chernoff bound are easy to apply for "large" values of k. As a result, the analysis of our reduction is easier than [10].*

6 Colored Orthogonal Range Search in $\mathbb{R}^2$

To illustrate an application of Reduction-I, we study the approximate colored counting query for orthogonal range search in $\mathbb{R}^2$.

Theorem 5. *Consider the following two problems:*

A) **Colored 3-sided range search in $\mathbb{R}^2$.** *In this setting, the input set S is n colored points in $\mathbb{R}^2$ and the query q is a 3-sided rectangle in $\mathbb{R}^2$. There is a data structure of $O(n)$ size which can answer the approximate colored counting query in $O(\varepsilon^{-2}\log n)$ time. This pointer machine structure is optimal in terms of n.*

B) **Colored 4-sided range search in $\mathbb{R}^2$.** *In this setting, the input set S is n colored points in $\mathbb{R}^2$ and the query q is a 4-sided rectangle in $\mathbb{R}^2$. There is a data structure of $O(n\log n)$ size which can answer the approximate colored counting query in $O(\varepsilon^{-2}\log n)$ time.*

6.1 Colored 3-sided range search in $\mathbb{R}^2$

We use the framework of Theorem 3 to prove the result of Theorem 5(A). For this geometric setting, a colored reporting structure with $\mathcal{S}_{rep} = n$ and $\mathcal{Q}_{rep} = \log n$ is already known [43]. The path-range tree of Nekrich [38] gives a $(4+\varepsilon)$-approximation, but it requires super-linear space. The C-approximation structure presented in this subsection is a refinement of the path-range tree for the pointer machine model.

Lemma 14. *For the colored 3-sided range search in $\mathbb{R}^2$ problem, there is a C-approximation structure which requires $O(n)$ space and answers a query in $O(\log n)$ time.*

We prove Lemma 14 in the rest of this subsection.

6.1.1 Interval tree

Our solution is based on an interval tree and we will need the following fact about it.

Lemma 15. *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 $\mathcal{IT}$ as follows: W.l.o.g., assume that the rectangles are bounded along the x-axis. Let h be a plane perpendicular to the x-axis such that there are equal number of endpoints of R on each side of the plane. The splitting halfplane h is stored at the root of $\mathcal{IT}$ and the two subtrees are built recursively. In general, $h(v)$ is the splitting halfplane stored at a node $v \in \mathcal{IT}$. A rectangle $r \in R$ is stored at the highest node v s.t. r intersects $h(v)$. Let R_v be the set of rectangles stored at a node v. Each rectangle in $r \in R_v$ is split by $h(v)$ into two rectangles r^- and r^+. Define $R_v^- := \bigcup_{r\in R_v} r^-$ and $R_v^+ := \bigcup_{r\in R_v} r^+$.

Given a query point q, trace a path Π of length $O(\log n)$ from the root to a leaf node corresponding to q. For a node $v \in \Pi$, if q lies to the left (resp., right) of $h(v)$, 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 we can treat R_v^- (resp., R_v^+) as $(2+t)$-sided rectangles in $\mathbb{R}^3$, since $h(v)$ is effectively $+\infty$ (resp., $-\infty$). □

6.1.2 Initial structure

Lemma 16. *For the colored* 3*-sided range search in* $\mathbb{R}^2$ *problem, there is a* 2*-approximation structure which requires* $O(n)$ *space and answers a query in* $O(\log^3 n)$ *time.*

Proof. 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 colored 3-sided range search in $\mathbb{R}^2$ also reduces to standard 5-sided rectangle stabbing problem (for brevity, call it 5-sided RSP).

There is a simple linear-size data structure which reports in $O(\log^3 n)$ time a 2-approximation for the 5-sided RSP: By inductively applying Lemma 15 twice, we can decompose 5-sided RSP to $O(\log^2 n)$ 3-sided RSPs. For 3-sided RSP, there is a linear-size structure of which reports a 2-approximation in $O(\log n)$ time [4]. By using this structure the 5-sided RSP can be solved in $O(\log^3 n)$ time. □

6.1.3 Final structure

Now we will present the optimal C-approximation structure of Lemma 14.

Structure. Sort the points of S based on their x-coordinate value and divide them into buckets containing $\log^2 n$ consecutive points. Based on the points in each bucket, build a D-structure which is an instance of Lemma 16. Next, build a height-balanced binary search tree $\mathcal{T}$, where the buckets are placed at the leaves from left to right based on their 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 5, which illustrates these sets for two leaves $u = u_l$ and $u = u_r$. 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*, $S''_l(u, v)$, 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 $O(n)$.

Query algorithm. To answer a query $q = [x_1, x_2] \times [y, \infty)$, we first determine the leaf nodes u_l and u_r of $\mathcal{T}$ 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 by querying the D-structure of u_l with $[x_1, \infty) \times [y, \infty)$ and the D-structure of u_r with $(-\infty, x_2] \times [y, \infty)$. Next, scan the list $S''_r(u_l, v)$ (resp., $S''_l(u_r, v)$) to find a 2-approximation of the number of colors of $S_r(u_l, v)$ (resp., $S_l(u_r, v)$) present in q.

The final answer is the sum of the count returned by the four sub-queries. The time

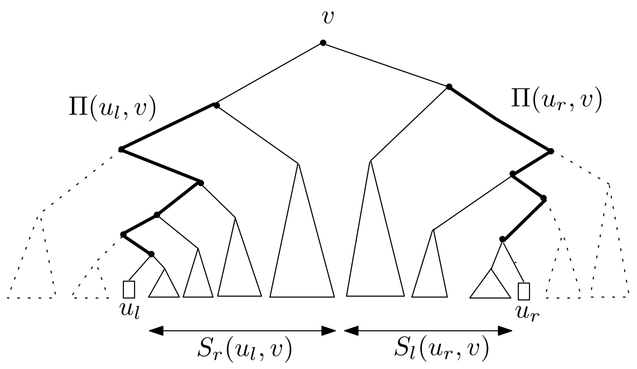


Figure 5: u_l and u_r are the leaf nodes containing x_1 and x_2.

taken to find u_l, u_r and v is $O(\log n)$. Querying the leaf structures takes $O((\log(\log^2 n))^3) = O(\log n)$ time. The time taken for scanning the lists $S''_r(u_l, v)$ and $S''_l(u_r, v)$ is $O(\log n)$. Therefore, the overall query time is bounded by $O(\log n)$. Since each of the four sub-queries give a 2-approximation, overall we get a 8-approximation.

6.2 C-approximation for 4-sided range search

Now we will prove Theorem 5(B). Again we will use the framework of Theorem 3. It is straightforward to obtain a data structure with $\mathcal{S}_{capp} = O(n \log n)$, $\mathcal{Q}_{capp} = O(\log n)$ and $C = 16$. Simply build a binary range tree on the y-coordinates of S and at each node build an instance of Lemma 14 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 [43] presented a reporting structure with $\mathcal{S}_{rep} = O(n \log n)$ and $\mathcal{Q}_{rep} = O(\log n)$. Plugging in these values into Theorem 3 proves Theorem 5(B).

Remark 5. As discussed in Remark 3, the technique of [10] can be adapted to answer a colored approximate counting query. For colored 3-sided range search in $\mathbb{R}^2$, plugging in $S(n) = O(n)$ and $Q(n) = O(\log n)$ [35] leads to a data structure of size $O(n \log^2 n)$ and query time $O(\log^2 n)$. For colored 4-sided range search in $\mathbb{R}^2$, plugging in $S(n) = O(n \log n)$ and $Q(n) = O(\log n)$ [18] leads to a data structure of size $O(n \log^3 n)$ and query time $O(\log^2 n)$ (the standard reporting structure of Chazelle [17] can be used to obtain slightly better space).

7 Applications of Reduction-II

In this section we present a few applications of reduction-II. For the colored problems discussed in this section, their exact counting structures are expensive [23, 29].

Theorem 6. *Consider the following three colored geometric settings:*

1. **Colored halfplane range search**, *where the input is a set of n colored points in $\mathbb{R}^2$ and the query is a halfplane. There is a data structure of $O(n)$ size which can answer the approximate counting query in $O\left(\frac{1}{\varepsilon^2} \cdot \log n \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|)\right)$ time.*

2. **Colored halfspace range search in** $\mathbb{R}^3$, *where the input is a set of* n *points in* $\mathbb{R}^3$ *and the query is a halfspace. There is a data structure of* $O(n \log n)$ *size which can answer the approximate counting query in* $O\left(\frac{1}{\varepsilon^2} \log^3 n \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|)\right)$ *time.*

3. **Colored orthogonal range search in** $\mathbb{R}^d$, *where the input is a set of* n *points in* $\mathbb{R}^d$ *and the query is an axis-parallel rectangle. There is a data structure of* $O(n \log^d n)$ *size which can answer the approximate counting query in* $O\left(\frac{1}{\varepsilon^2} \cdot \log^{d+1} n \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|)\right)$ *time.*

Colored orthogonal range search 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 $\mathcal{Q}_{rep}(n) = O(\log^{d-1} n)$ query time. Using this structure, the colored orthogonal range reporting problem in $\mathbb{R}^d$ can be answered in $O(\mathcal{Q}_{rep}(n) + \kappa \mathcal{Q}_{rep}(n) \log n)$ query time using a structure of size $O(M(n) \log n)$. Here κ is the number of colors reported (see Section 1.3.4 of [23] for the details of this transformation).

By applying Theorem 4, the space occupied by the approximate counting structure will be $O(n \log^d n)$. In Theorem 4, we assumed that the query time of the colored reporting structure can be expressed as $O(\mathcal{Q}_{rep} + \kappa)$, whereas for this problem the query time is being expressed as $O(\mathcal{Q}_{rep} + \kappa \mathcal{Q}_{rep} \log n)$. Therefore, equation 5 of the query time $\mathcal{Q}_{\varepsilon app}(n)$ in Theorem 4 can be rewritten as

$$O\Bigg(\big(\mathcal{Q}_{rep}(n) + (\varepsilon^{-2} \log n) \mathcal{Q}_{rep}(n) \log n\big) \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|) \Bigg)$$

Plugging in the value of $\mathcal{Q}_{rep}(n)$ into the above expression, we get

$$\mathcal{Q}_{\varepsilon app}(n) = O\Bigg(\big(\log^{d-1} n + \varepsilon^{-2} \log^{d+1} n\big) \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|) \Bigg) = O\Bigg(\varepsilon^{-2} \cdot \log^{d+1} n \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|) \Bigg)$$

Colored halfspace range search 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 $O(n^{1/2+\delta} + \kappa)$ time [23]. 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 $\mathcal{Q}_{rep}(n) = O(\log n)$ query time [3]. Using this structure, the colored halfspace range reporting problem in $\mathbb{R}^3$ can be answered in $O(\mathcal{Q}_{rep}(n) + \kappa \mathcal{Q}_{rep}(n) \log n) = O(\varepsilon^{-2} \log^3 n)$ query time using a structure of size $O(M(n) \log n) = O(n \log n)$.

Applying Theorem 4, the space occupied by the approximate counting structure will be $O(n \log n)$. The query time will be

$$O\Bigg(\big(\mathcal{Q}_{rep}(n) + (\varepsilon^{-2} \log n) \mathcal{Q}_{rep}(n) \log n\big) \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|) \Bigg) = O\Bigg((\varepsilon^{-2} \log^3 n) \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|) \Bigg)$$

Colored halfplane range search. In [23] 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 [5] 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 halfplane range reporting with the same bounds. Plugging in this result into Theorem 4, we obtain an $O(n)$ size data structure which can answer the approximate counting query in $O(\varepsilon^{-2} \log n \cdot \log(\log_{1+\varepsilon} |\mathcal{C}|))$ time.

8 Conclusion and Future Directions

In this work, we built optimal and near-optimal approximate counting data structures for several colored and uncolored (i.e., standard) geometric settings. We finish by presenting a couple of interesting future directions.

Preprocessing time. We do not discuss the preprocessing time in this paper since most of our solutions (except Theorem 1) are based on verifying if a sample is “suitable", and we do not know how to verify if a sample is suitable in $n \log^{O(1)} n$ time. This is challenging but an important problem to resolve.

Colored orthogonal range search in 2d and 3d. Colored orthogonal range search in 1d is the simplest possible geometric setting, where the input is a set of n colored points and the query is an interval. Recently, for this setting El-Zein *et al.* [21] obtained a succinct data structure to answer approximate colored counting query in constant time. Approximate colored counting query for orthogonal range search in 2d and 3d remains open. Note that the reductions presented in this paper lead to a query time of $\Omega(\log n)$, whereas colored reporting structures with sub-logarithmic query time and near-linear space are known in the word-RAM model. For example, the recent result of Chan and Nekrich with $O(n \log^{3/4+\varepsilon} n)$ space and $O(\log \log U + k)$ query time in 2d [15], and another recent result of Chan, He and Nekrich with $O(n \log^{2+\varepsilon} n)$ space and $O(\log \log U + k \log \log n)$ query time in 3d [14]. Obtaining corresponding matching bounds in 2d and 3d (with $k = 0$) for approximate colored counting is open.

Instance-specific bounds for exact colored counting. As mentioned in the Introduction, many exact colored counting problems have expensive space and query time bounds. However, not all instances of colored objects are “hard”. In extreme cases, if all the objects have the same color, then exact colored counting reduces to an emptiness query, and if all the objects have distinct colors, then exact colored counting reduces to “standard” counting. In general, is it possible to capture the hardness of a colored instance, and then construct data structures whose space and query time bounds are sensitive to the hardness?

Acknowledgements. I am thankful to Prof. Sariel Har-Peled, Prof. Ravi Janardan, Yuan Li, Sivaramakrishnan Ramamoorthy, Stavros Sintos, and Jie Xue for fruitful discussions on these problems and comments on the previous drafts. Thanks to the anonymous referees whose comments helped in immensely improving the content and the presentation of the paper.

References

[1] Peyman Afshani. On dominance reporting in 3D. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 41–51, 2008.

[2] Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. *Discrete & Computational Geometry*, 42(1):3–21, 2009.

[3] 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.

[4] 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.

[5] 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.

[6] Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal of Computing*, 29(3):912–953, 1999.

[7] Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Advances in Discrete and Computational Geometry*, pages 1–56, 1999.

[8] 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.

[9] Lars Arge, Gerth Stølting Brodal, Rolf Fagerberg, and Morten Laustsen. Cache-oblivious planar orthogonal range searching and counting. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 160–169, 2005.

[10] Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008.

[11] Boris Aronov and Micha Sharir. Approximate halfspace range counting. *SIAM Journal of Computing*, 39(7):2704–2725, 2010.

[12] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[13] 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.

[14] Timothy M. Chan, Qizheng He, and Yakov Nekrich. Further results on colored range searching. In *International Symposium on Computational Geometry (SoCG)*, pages 28:1–28:15, 2020.

[15] Timothy M. Chan and Yakov Nekrich. Better data structures for colored orthogonal range reporting. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–636, 2020.

[16] 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.

[17] Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM Journal of Computing*, 15(3):703–724, 1986.

[18] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 3rd edition, 2008.

[19] Tamal K. Dey. Improved bounds for planar k -sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.

[20] Herbert Edelsbrunner and Mark H. Overmars. On the equivalence of some rectangle problems. *Information Processing Letters (IPL)*, 14(3):124–127, 1982.

[21] Hicham El-Zein, J. Ian Munro, and Yakov Nekrich. Succinct color searching in one dimension. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 30:1–30:11, 2017.

[22] 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.

[23] Prosenjit Gupta, Ravi Janardan, Saladi Rahul, and Michiel H. M. Smid. Computational geometry: Generalized (or colored) intersection searching. In *Handbook of Data Structures and Applications*. 2018.

[24] 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.

[25] Sariel Har-Peled and Micha Sharir. Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.

[26] Joseph JáJá, Christian Worm Mortensen, and Qingmin Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 558–568, 2004.

[27] Ravi Janardan and Mario A. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications*, 3(1):39–69, 1993.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] Edward M. McCreight. Priority search trees. *SIAM Journal of Computing*, 14(2):257–276, 1985.

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

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

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

[39] 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.

[40] 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.

[41] 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.

[42] Micha Sharir and Hayim Shaul. Semialgebraic range reporting and emptiness searching with applications. *SIAM Journal of Computing*, 40(4):1045–1074, 2011.

[43] 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.

[44] Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences (JCSS)*, 18(2):110–127, 1979.

[45] Géza Tóth. Point sets with many k-sets. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 37–42, 2000.

[46] Peter van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters (IPL)*, 6(3):80–82, 1977.

[47] V.N. Vapnik and A.Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

Appendix: Models of computation

Word-RAM model. In this model [22], we have a collection of cells, each of which is a w-bit word. Each cell can, therefore, store integer values in the range $\{0, \ldots, 2^w - 1\}$. Random access to any cell can be performed in constant time. Basic operations on words (which are performed in modern programming languages such as C, C++, or Java) take constant time. This includes arithmetic operations (such as $+, -, *, /, \%$), comparisons ($<, >, =$), and bitwise boolean operations (bitwise-AND, OR, and exclusive-OR). We assume that $w \geq \log U$ and $w \geq \log n$, so that the coordinate of any object fits in a single word and the *memory location* of any of the n objects also fits in a single word, respectively. The space of the data structure is measured in terms of the number of words/cells occupied.

Pointer machine model. This model has been used extensively for proving several interesting lower bounds and upper bounds for range searching and related problems. Loosely speaking, in this model the data structure is modeled as a graph and one is not allowed to do a random access. Formally, as defined by Tarjan [44], in this model a data structure can be regarded as a directed graph, where each node stores $O(1)$ real values and $O(1)$ pointers to other nodes. Random access to a node is not allowed and only pointers can be used to access a node. We begin answering a query using a pointer to a *root node* of the data structure. The *query time* of an algorithm is the total number of nodes visited, whereas the *size* of a structure is the number of its nodes and edges. Further details can be found in Agarwal and Erickson [7].