Approximate Range Counting Revisited[*][†]

Saladi Rahul

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA
sala0198@umn.edu

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.

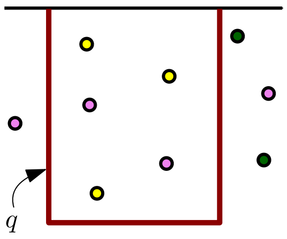
1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases geometric data structures, range searching, rectangle stabbing, approximate counting, colors

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.55

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 [8, 10, 11, 12]. 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

* The full version of this work with the title "Approximate Range Counting Revisited" can be found on https://arxiv.org/abs/1512.01713v3.

† This research was partly supported by a Doctoral Dissertation Fellowship (DDF) from the Graduate School of University of Minnesota.

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 $[(1-\varepsilon)k, (1+\varepsilon)k]$.

General reduction to companion problems. Aronov and Har-Peled [3], and Kaplan, Ramos and Sharir [9] 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 [3] and [9] seem different, there is an interesting discussion in Section 6 of [3] about the underlying "sameness" of both techniques.

Levels. Informally, for a set S of n objects, a *t-level* of S is a surface such that if a point q lies above (resp., on/below) the surface, then the number of objects of S containing q is $> t$ (resp., $\leq t$). 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 $\Omega(n 2^{\sqrt{\log t}})$ in the worst-case (the known upper bound is $O(nt^{1/3})$ 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, that lies strictly below the $2t$-level, and their union covers all the points below (and on) the t-level. For many geometric objects in two and three dimensions, such t-shallow cuttings have $O(n/t)$ cells. 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 [1] 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 [2] 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 [11], and Chan and Wilkinson [5] 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 [11] 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 for counting in standard problems (going back to the work of Bentley and Saxe [4] 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 [10]. Any substantial improvement in the preprocessing time *and* the query time would lead to a substantial improvement in the best exponent of matrix multiplication [10] (which is a major open problem). Similarly, counting structures for colored halfspace counting in $\mathbb{R}^2$ and $\mathbb{R}^3$ [8] 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.

1.3.1.1 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 points, and the query is a 2-sided rectangle) and the *colored* 3-*sided rectangle stabbing* (the input is a set of n 3-sided rectangles, and the query is a point). The results are summarized in Table 1.

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. WR denotes the word-RAM model and PM denotes the pointer machine model.

Dime- -nsion	Input, Query	New Results	Previous Approx. Counting Results	Exact Counting Results	Model
1	intervals, point	S: n, Q: $\log\log U$ Theorem 1	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				
2	points, 3-sided rectangle	S: n, Q: $\log n$ Theorem 15(A)	S: $n\log^2 n$, Q: $\log^2 n$	not studied	PM
2	points, 4-sided rectangle	S: $n\log n$, Q: $\log n$ Theorem 15(B)	S: $n\log^3 n$, Q: $\log^2 n$	S: $n^2\log^6 n$, Q: $\log^7 n$ Kaplan *et al.* [10]	PM
3	points, 3-sided rectangle	S: $n\log^* n$, Q: $\log n\cdot\log\log n$ Theorem 7	S: $n\log^2 n$, Q: $\log^2 n$	not studied	PM

1.3.1.2 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 [6]. The results are summarized in Table 1.

1.3.1.3 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.

1.3.2 General reductions

We present two general reductions for solving approximate colored counting queries by reducing them to “easy" companion queries.

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 [3, 9], 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 [3] 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, and (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 becomes simpler. This reduction is useful when the reporting query is not significantly costlier than the emptiness query. The full version of this work will describe this reduction and its applications.

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 [11] 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.

1.3.3.1 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, the first general reduction is presented. In Section 5, the application of the first reduction to colored orthogonal range search in $\mathbb{R}^2$ problem is shown. Most of the proofs have been omitted and can be found in the full version.

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.

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 $\mathcal{N}(S_c) \cap q = \emptyset$.
- If $S_c \cap q \neq \emptyset$, then exactly one rectangle in $\mathcal{N}(S_c)$ is stabbed by q.

Let $\mathcal{N}(S) = \bigcup_{\forall c} \mathcal{N}(S_c)$, 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 2** (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 lie on a grid $[\![U]\!] \times [\![U]\!]$. 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 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 3.** *For the standard 3-sided rectangle stabbing in 2d problem, consider a setting where 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.*

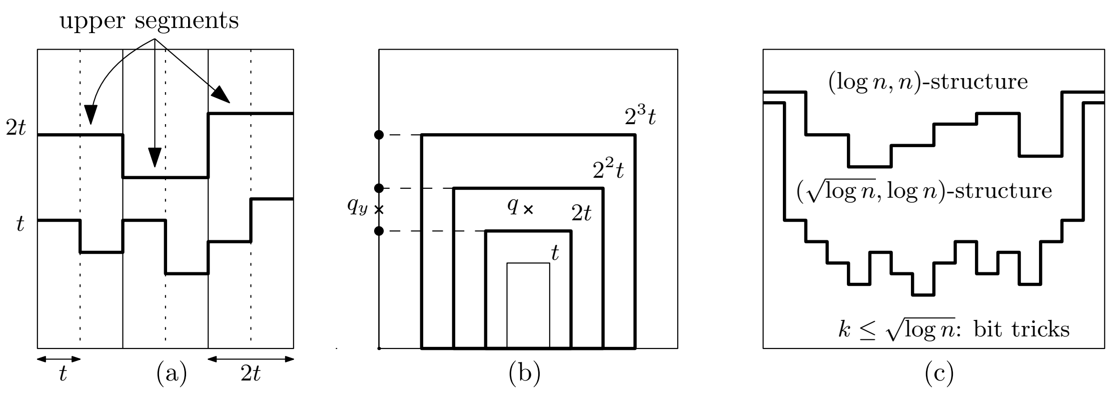


Figure 1 (a) A portion of the t-level and $2t$-level is shown. Notice that by our construction, each cell in the t-level is contained inside a cell in the $2t$-level. (b) A cell in the t-level and the set $\mathcal{C}_r$ associated with it. (c) A high-level summary of our data structure.

2.2.1 Nested shallow cuttings

To prove Lemma 3, we will first construct shallow cuttings for 3-sided rectangles in 2d. Unlike the general class of shallow cuttings, the shallow cuttings we construct for 3-sided rectangles will have a stronger property of cells in the lower level lying completely inside the cells of a higher level.

▶ **Lemma 4.** *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 $|S \cap q| \ge t$.*
3. *Each cell in $\mathcal{C}$ intersects at most $2t$ rectangles of S.*

Proof. Refer to the full version. ◀

▶ **Observation 5** (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, each cell in the t-level is contained inside a single cell of $2^i t$-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 [2], the authors show how to convert a constant-factor approximation into a $(1+\varepsilon)$-approximation for this geometric setting. The solution is based on *(t, t')-level-structure* and *$(\le \sqrt{\log n})$-level shared table.*

2.2.2.1 (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 5 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 [7]. 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.

2.2.2.2 ($\leq \sqrt{\log n}$)-level shared 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 shared table will answer the exact counting query in $O(1)$ time. We will need the following lemma.

▶ **Lemma 6.** *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. Refer to the full version. ◀

Shared 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 [7] 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.

2.2.2.3 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 shared table will efficiently handle the small values of t.

Therefore, the strategy is to construct the following: (a) a ($\leq \sqrt{\log n}$)-level shared 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.

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

▶ **Theorem 7.** *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. 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.

3.2 Initial strcuture

▶ **Lemma 8.** *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.

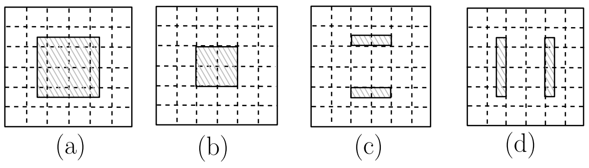


Figure 2

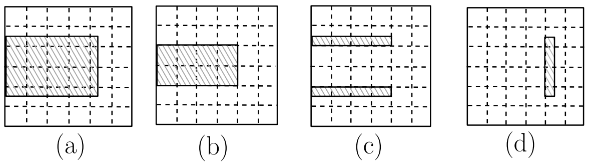


Figure 3

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 bottommost 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 2(b)), two *column rectangles*, which are the portions of r lying in column c_l and c_r (see Figure 2(d)), and two *row rectangles*, which are the remaining portion of the rectangle r lying in row r_b and r_t (see Figure 2(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.

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 3(b), one *column rectangle*, as shown in Figure 3(d), and two *row rectangles*, as shown in Figure 3(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.

▶ **Observation 9.** *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 [2] 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 9, 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 8.

3.3 Final structure

▶ **Lemma 10.** *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.*

Refer to the full version for a proof of this lemma.

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 11.** *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

$$\mathcal{S}_{\varepsilon app}(n) = O(\mathcal{S}_{rep}(n) + \mathcal{S}_{capp}(n)) \tag{1}$$

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

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 12** (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 13.** *Consider a query q which satisfies the two conditions stated in Lemma 12. 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. Refer to the full version. ◀

▶ **Lemma 14** (Finding a suitable R). *Pick a random sample R as defined in Lemma 13. 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. Refer to the full version. ◀

Refinement structure and query algorithm

In the preprocessing stage pick a random sample $R \subseteq \mathcal{C}$ as stated in Lemma 13. 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 13, $|R \cap q| \leq (1+\varepsilon)\cdot kM = O(\varepsilon^{-2}\log n)$. 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 choice of $\sqrt{C}$ will become clear in the analysis. This occupies $O(\mathcal{S}_{capp}(n))$ space.
3. *Refinement structures.* Build the refinement structure of Lemma 12 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 $\sum O(\mathcal{S}_{rep}(nM)) = O(\mathcal{S}_{rep}(n))$, since $\mathcal{S}_{rep}(\cdot)$ is converging and $\sum nM = O(n)$. 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. $z \leq k_a \leq \sqrt{C}z$. It is trivial to verify that $k \in [z, Cz]$. This takes $O(\mathcal{Q}_{rep}(n) + \varepsilon^{-2}\log n)$ time.

5 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 $\mathbb{R}^2$. We only prove Theorem 15(1) here. Refer to the full version for the proof of Theorem 15(2).

▶ **Theorem 15.** *Consider the following two problems:*

1. **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.*
2. **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.*

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

We use the framework of Theorem 11 to prove the result of Theorem 15(1). For this geometric setting, a colored reporting structure with $\mathcal{S}_{rep} = n$ and $\mathcal{Q}_{rep} = \log n$ is already known [12]. The path-range tree of Nekrich [11] 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 16.** *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 16 in the rest of this subsection. Our solution is based on an interval tree and we will need the following fact about it.

▶ **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. Refer to the full version. ◀

5.1.1 Initial structure

▶ **Lemma 18.** *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 17 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 [2]. By using this structure the 5-sided RSP can be solved in $O(\log^3 n)$ time. ◀

5.1.2 Final structure

Now we will present the optimal C-approximation structure of Lemma 16.

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 18. 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. 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 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.

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, 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.

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

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

5 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.

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

7 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.

8 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Computational geometry: Generalized intersection searching. In *Handbook of Data Structures and Applications.* 2004.

9 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.

10 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.

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

12 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.