Counting Colors in Boxes *

Haim Kaplan† Natan Rubin‡ Micha Sharir§ Elad Verbin¶

Abstract

Let P be a set of n points in $\mathbb{R}^d$, so that each point is colored by one of C given colors. We present algorithms for preprocessing P into a data structure that efficiently supports queries of the form: Given an axis-parallel box Q, count the number of distinct colors of the points of $P \cap Q$. We present a general and relatively simple solution that has polylogarithmic query time and worst-case storage about $O(n^d)$. It is based on several interesting structural properties of the problem that we derive. We also show that for random inputs, the data structure requires almost linear expected storage.

We then present several techniques for achieving space-time tradeoff. In $\mathbb{R}^2$, the most efficient solution uses fast matrix multiplication in the preprocessing stage. In higher dimensions we use simpler tradeoff mechanisms, which behave just as well. We give a reduction from matrix multiplication to the offline version of problem, which shows that in $\mathbb{R}^2$ our time-space tradeoffs are close to optimal in the sense that improving them substantially would improve the best exponent of matrix multiplication. Finally, we present a generalized matrix multiplication problem and show its intimate relation to counting colors in boxes in any dimension.

1 Introduction

We consider the following range counting problem. Let P be an input set of n points in $\mathbb{R}^d$, each colored in one of C different colors. Our goal is to preprocess P into a data structure that, for a given query axis-parallel box $Q \subset \mathbb{R}^d$, can efficiently count the number of *distinct colors* of points in $Q \cap P$. We call this problem *colored orthogonal range counting*. In this work we only deal with the static setting of the problem, not allowing insertions or deletions to the data structure.

The problem arises in many applications. For example, in database applications, the items in the database have some attribute (e.g., the city where they have been born, the college that they have attended, or even numerical attributes such as their age), and the query asks for the number of different *attribute values* of all the items in a query box (e.g., how many different cities of birth do the persons in a query box have?). In geometric contexts, the problem arises, e.g., in the following scenario: We are given C rectilinear polygons in the plane with a total of n edges, and wish to count the number of distinct polygons that intersect a given query box. Similar problems arise in higher dimensions. (We note that the attribute (color) is not related to the d coordinates of the data points. We could have added it as an additional coordinate, but this would not have changed the nature of the problem.)

*Work by Haim Kaplan has been supported by Grant 975/06 from the Israel Science Fund. Work by Micha Sharir was partially supported by NSF Grant CCF-05-14079, by a grant from the U.S.-Israeli Binational Science Foundation, by grant 155/05 from the Israel Science Fund, Israeli Academy of Sciences, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. This work is part of the second author's M.Sc. Dissertation, prepared under the supervision of the first and third authors.

†School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: haimk@post.tau.ac.il

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: rubinnat@post.tau.ac.il

§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. E-mail: michas@post.tau.ac.il

¶School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: eladv@post.tau.ac.il

1.1 Related work

Colored intersection range searching. In colored intersection range searching we are given n colored objects of constant description complexity. We wish to construct a data structure, that can report or count the colors of objects in a query range. Colored range searching problems[1] have been studied for many years by Gupta, Janardan, Smid and others (see a recent survey by Gupta *et al.* [12] for a comprehensive review). As has been observed, colored variants of range searching are in general much harder to solve than the standard variants. The reason for this discrepancy is that colored problems are not *decomposable*. For example, partitioning the query box into two (disjoint) sub-ranges and counting the number of colors in each sub-range tells us practically nothing about the number of colors in the full range. Halfspace colored range searching (with halfspaces as queries and points as colored objects) was studied in [15]. Orthogonal colored range searching problems (where queries are axis-parallel boxes) were studied in [4, 6, 13, 16, 19]. Additional colored range searching problems were studied in [4, 6, 14, 15]. Batched colored range intersection problems, where we report all pairs of colors (c_1, c_2) such that an object of color c_1 intersects an object of color c_2, were studied in [5, 20].

Orthogonal colored range reporting. In [16], Gupta *et al.* describe solutions for colored range reporting in $\mathbb{R}^1$, $\mathbb{R}^2$, and $\mathbb{R}^3$, with poly-logarithmic query time and near-linear storage. For $\mathbb{R}^1$, they perform orthogonal colored range searching (counting and reporting) in both static and dynamic settings, by a reduction to standard orthogonal range searching in $\mathbb{R}^2$. Specifically, they obtain a dynamic data structure of size $O(n)$, such that the i distinct colors of the points in a query interval can be reported in $O(\log n + i)$ time (or counted in $O(\log n)$ time), while supporting updates (insertions and deletions of points) in $O(\log n)$ time. Specifically, if the points of some color are $p_1 < p_2 < \ldots < p_n$, then they are mapped to the points $(-\infty, p_1), (p_1, p_2), \ldots, (p_{n-1}, p_n)$ in $\mathbb{R}^2$. A query interval $Q = [a, b]$ is mapped to the semi-unbounded rectangle $\tilde{Q} = [a, b] \times (-\infty, a]$. It is an easy observation that $[a, b]$ contains at least one (resp., no) point of color c if and only if $[a, b] \times (-\infty, a]$ contains exactly one (resp. no) transformed point of color c. Hence, counting or reporting colors in intervals in $\mathbb{R}^1$ is equivalent to counting or reporting points in the above kind of semi-unbounded rectangles in the transformed set in $\mathbb{R}^2$.

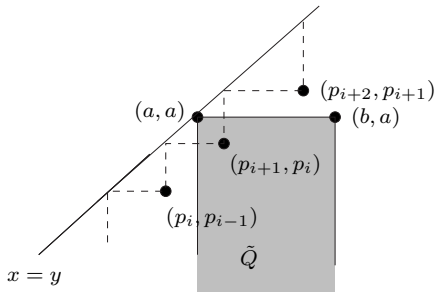

Figure 1: The transformed point set of color c (points below the line $x = y$). The highlighted semi-unbounded rectangle $\tilde{Q} = [a, b] \times (-\infty, a]$ contains exactly one transformed point of color c; the interval $[a, b]$ contains two original points of color c.

In $\mathbb{R}^2$, the static data structure of Janardan *et al.* [19] uses $O(n \log^2 n)$ space and answers queries in time $O(\log n + i)$, where as above, i is the output size. Gupta *et al.* [16] obtain a semi-dynamic solution for the reporting problem by first dealing with the special case, when the queries

[1] These problems are also referred to in the literature as *generalized range searching* problems; see [12].

are quadrants of the form $[a, \infty) \times [b, \infty)$. They reduce this special case to standard ray-segment intersection searching in $\mathbb{R}^2$. In order to process arbitrary rectangular queries, they decompose a rectangular query into four quadrant queries, each answered over an appropriate subset of points. The resulting data structure requires $O(n \log^2 n)$ space and allows reporting the i distinct colors of points contained in a query rectangle in $O(\log^2 n + i)$ time. We can insert a point into this data structure in $O(\log^3 n)$ amortized time. Finally, Gupta *et al.* [16] gave a fully dynamic solution to the reporting problem in $\mathbb{R}^2$. Their solution is based on decomposing the two-dimensional query into $O(\log n)$ one-dimensional queries, each answered over a proper subset of points. This data structure uses $O(n \log n)$ space, supports queries in $O(\log^2 n + i \log n)$ time, where i is the number of reported colors, and allows insertions and deletions in $O(\log^2 n)$ time.

In $\mathbb{R}^3$, Gupta *et al.* [16] extend their two-dimensional semi-dynamic solution and describe a static data structure of size $O(n \log^4 n)$ with $O(\log^2 n + i)$ query time, where i is the number of reported colors. In [13], Gupta *et al.* describe a static data structure for orthogonal colored range reporting in any dimension, which requires storage $O(n^{1+\epsilon})$[2], such that for any query box in $\mathbb{R}^d$, the i distinct colors of points contained in it are reported in $O(\log n + i)$ time.

These results use the fact that if we decompose the problem into a small (constant or logarithmic) number of subproblems, and solve each subproblem separately then each color is discovered only a small number of times. This still keeps the query time close to $O(\text{polylog}n + i)$.

Orthogonal colored range counting. Orthogonal colored range *counting* turns out to be harder than orthogonal colored range reporting. In [16], Gupta *et al.* describe solutions for colored range counting in $\mathbb{R}^1$ and $\mathbb{R}^2$. Similarly to the case of reporting, they reduce[3] the one-dimensional problem to standard orthogonal range searching in $\mathbb{R}^2$. The resulting solution has query time $O(\log n)$ and storage and preprocessing cost $O(n \log n)$. Its *dynamic* version also requires $O(n \log n)$ space and supports queries and updates in $O(\log^2 n)$ time. (The storage of the static and the dynamic data structures can be further reduced by a logarithmic factor, as noted in [4].) Using persistence, Gupta *et al.* [16] extend their one-dimensional data structure into a static two-dimensional structure which supports queries that involve *3-sided boxes*; that is, boxes of the form, say, $[a, b] \times (-\infty, c]$. Using a linear number of copies of this structure, they obtain a complete solution in $\mathbb{R}^2$, with query time $O(\log^2 n)$, and storage and preprocessing cost $O(n^2 \log^2 n)$.

Halfspace colored range searching. Efficient static data structures for halfspace colored range searching were described by Gupta *et al.* [15]. Their solution in $\mathbb{R}^2$ and $\mathbb{R}^3$ for counting and reporting is based, through duality, on a straightforward reduction to an instance of the *ray-envelope intersection problem*. In this problem, we are given a set of upper envelopes of linear functions (the lines/planes dual to the input points) such that each envelope is computed for the lines/planes of some fixed color. The overall complexity of the envelopes is $O(n)$. The objective is to preprocess the envelopes into a data structure such that given an upward vertical ray we can report (resp., count) the envelopes that it intersects. See Figure 2. The solution in $\mathbb{R}^2$ uses $O(n \log n)$ space and answers reporting (resp., counting) queries in $O(\log^2 n + i)$ (resp. $O(n^{1/2})$) time. The solution in $\mathbb{R}^3$ is based on partition trees (see [22]) and uses $O(n \log^2 n)$ space and answers reporting (resp., counting) queries in $O(n^{1/2+\epsilon} + i)$ (resp., $O(n^{2/3+\epsilon})$) time. Using cutting trees instead [24], reporting in $\mathbb{R}^3$ can be solved with $O(n^{2+\epsilon})$ storage, so that a query takes $O(\log^2 n + i)$ time. Note that the time and space bounds for the colored range counting in $\mathbb{R}^2$ and $\mathbb{R}^3$ are close to those achieved for standard range counting problem in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively [22, 24].

[2]Bounds of this form hold for any $\epsilon > 0$; the constant of proportionality depends on ϵ, and generally tends to ∞ as ϵ descends to 0.

[3]In a sense, their reduction is the forefather of the decomposition scheme that we develop for the higher-dimensional cases; see Section 3.

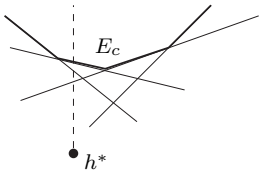

Figure 2: E_c is the upper envelope of the dual hyperplanes of the input points having color c. h^* is the point dual to the hyperplane h bounding the query half-space below h. There is an input point below h having color c if and only if the the upward vertical ray emanating from h^* intersects E_c (that is, h^* lies below E_c).

This approach does not efficiently generalize to $\mathbb{R}^d$, for $d > 3$. Gupta *et al.* [15] solve the reporting problem in $\mathbb{R}^d$, for $d > 3$, using a balanced binary tree CT built over the colors. Each node v in CT points to a halfspace range-emptiness data structure (described in [21]), built over the points, whose colors belong to the subtree of v. The query algorithm starts at the root of CT and continues to the children of a node v only if the query range contains at least one point, whose color belongs to the subtree of v. The resulting data structure requires $O(n^{\lfloor d/2 \rfloor}/\log^{\lfloor d/2 \rfloor - 1-\epsilon} n)$ storage, and can report the i distinct colors of points contained in a query halfspace in time $O(\log n + i \log^2 n)$. Other applications of this approach can be found in [14].

Batched colored intersection searching. In a *batched colored intersection searching problem* we are given a set of colored geometric objects, and wish to compute all the pairs of colors (c_1, c_2) such that there are two intersecting input objects, one of color c_1 and one of color c_2. Often a *bipartite* version of the problem is considered, where we are given two sets of objects of two different classes, and wish to report all pairs (c_1, c_2) such that an object of the first set having color c_1 intersects an object of the second set having color c_2. Efficient solutions of the colored batched intersection searching problem for line segments in $\mathbb{R}^1$, line segments in $\mathbb{R}^2$, axis-parallel boxes in $\mathbb{R}^d$, points and triangles in $\mathbb{R}^2$, and points and halfspaces in $\mathbb{R}^d$, are given in [5, 20]. For example, in the case of segments in $\mathbb{R}^1$ the problem can be solved in time $O(nc^{0.69})$ if $n \geq c^{1.69}$, and $O(n^{0.7}c^{1.21} + c^2)$ if $n \leq c^{1.69}$, using matrix multiplication techniques [20]. Variants of these techniques will also be used in this paper. See Section 4.

1.2 Our contribution

We present a different approach to orthogonal colored range counting, based on a reduction from this problem to standard orthogonal range counting, which works in any dimension. We transform P into a higher-dimensional space, where each query box corresponds to a point, and for each color $c \in C$, the space $Q(c)^+$ of all (points representing) query boxes containing a point with color c is the union of positive orthants.

For each color c, we show how to decompose $Q(c)^+$ into pairwise disjoint boxes. As a consequence, a query box contains a point of color c iff the point corresponding to the query in the transformed space is contained in exactly one of the boxes in the decomposition of $Q(c)^+$. Our algorithm thus collects the decomposition boxes, over all colors c, and stores them in a data structure that supports efficient containment counting queries of the form: Given a query point x, count the number of boxes that contain x.

A straightforward implementation of this technique yields a solution for the colored orthogonal range counting problem in $\mathbb{R}^d$, having query time $O(\log^{2d-1} n)$, storage complexity $O(n^d \log^{2d-1} n)$ and worst case (deterministic) preprocessing time $O(n^d \log^{2d-1} n)$. A simple enhancement of the

technique reduces all three performance parameters by a factor of $O(\log n)$, thus matching the performance parameters of the algorithms of Gupta et al. [12] for $d = 2$. If our queries are orthants; that is, boxes that are semi-unbounded in each (say, negative) coordinate direction, we can do better, obtaining a data structure with a query time of $O(\log^{d-1} n)$ and storage and preprocessing cost $O(n^{\lfloor d/2 \rfloor} \log^{d-1} n)$. For $d \geq 4$ and even, the same enhancement trick mentioned above reduces all three performance parameters by a $\log n$ factor.

In fact, the storage size and preprocessing time of our algorithm depend on the overall number of boxes in the decomposition of $Q(c)^+$, over all colors c. This can be as large as $O(n^d)$ (for general box queries), but in practice we expect it to be much smaller. To support this statement, we show that, for random point sets (drawn independently and uniformly at random from $[0,1]^d$), the expected number of boxes in our decomposition is only $O(n \log^{d-1} n)$, which leads to algorithms with polylogarithmic query time and near-linear expected storage and preprocessing cost.

Time-space tradeoff. We also consider techniques for reducing the storage (and preprocessing) at the cost of increasing the query time. In $\mathbb{R}^2$, we use a technique of Gupta *et al.* [12] to decompose a query $[a,b] \times [c,d]$ into two 3-sided queries $[a,b] \times (-\infty, d]$ and $[a,b] \times [c, \infty)$ on secondary structures stored at the nodes of a binary search tree over the points sorted by their y-coordinates. (This leads to significant gains, because, as we show, the structure for answering 3-sided queries requires only near-linear storage and preprocessing, whereas the structure for 4-sided queries may require near-quadratic storage.) We obtain the solution for each 3-sided query as a collection of pairwise disjoint canonical sets of colors. The difficulty is that a single color may appear in the solutions of the two 3-sided queries but we need to count it only once. To efficiently compute the union of the answers of the two queries we use the principle of inclusion-exclusion to reduce the problem to computing the sizes of all pairwise intersections of the output canonical subsets. We use sparse matrix multiplication techniques [7, 20, 26] to precompute efficiently some of these intersection sizes, and handle others on the fly when processing a query.

We obtain a solution that has query time $O(X \log^7 n)$, and storage $O\left(\left(\frac{n}{X}\right)^2 \log^6 n + n \log^4 n\right)$, for any tradeoff parameter $1 \leq X \leq n$. The construction time of the data structure is (the $O^*(\cdot)$ notation hides polylogarithmic factors):

$$\begin{cases} O^*\left(\frac{n^{(\omega+1)/2}}{X^{(\omega-1)/2}}\right) = O\left(\frac{n^{1.688}}{X^{0.688}}\right) & \text{when } X \geq n^{\frac{\omega-1}{\omega+1}} \approx n^{0.408}, \\ O^*\left(\frac{n^{\frac{2-\alpha\beta+2\beta}{\beta+1}}}{X^{\frac{2-\alpha\beta}{\beta+1}}}\right) = O\left(\frac{n^{1.898}}{X^{1.203}}\right) & \text{when } n^{\frac{\alpha/2}{\alpha/2+1}} \approx n^{0.128} \leq X \leq n^{\frac{\omega-1}{\omega+1}} \approx n^{0.408}, \\ O^*(\frac{n^2}{X^2}) & \text{when } X \leq n^{\frac{\alpha/2}{\alpha/2+1}} \approx n^{0.128}. \end{cases}$$

Here ω is the smallest number such that two $t \times t$ matrices can be multiplied in time $O(t^\omega)$ (the best known upper bound on ω is 2.376), $\alpha > 0.294$ is another parameter related to matrix multiplication and $\beta = \frac{\omega-2}{1-\alpha}$; see [10, 11] and Section 4. In particular, it follows from these bounds that, for $m \leq n$, we can answer m queries in overall time $O^*(nm^{\frac{\omega-1}{\omega+1}}) = O(nm^{0.408})$ (including preprocessing).

Interestingly, we also show a reduction of a version of sparse matrix multiplication to an offline version of colored orthogonal range counting in $\mathbb{R}^2$. This reduction implies that if we can answer m queries on a set of n points, for $m^{\frac{1+\omega}{4}} \leq n$, in $o(nm^{\frac{\omega-1}{4}}) = o(n^{1.34})$ time, then we can obtain a better algorithm for sparse rectangular matrix multiplication than the best known to date. Furthermore, if we can answer n such queries in $o(n^{\frac{2.376}{2}}) = o(n^{1.188})$ time, we improve the best known bound on ω. This suggest that our bounds, while not claimed to be near optimal, are significant, and that substantial improvements are likely to be quite different.

A simple bucketing technique allows us to trade time for space also in dimension $d > 2$. Specifically, for any threshold parameter $1 \leq X \leq n$, we obtain a data structure, having query time

$O(X \log^d n + \log^{2d-1} n)$ and preprocessing and storage cost $O(\frac{n^d}{X^{d-1}} \log^{2d-1} n)$. We suggest two additional techniques to improve this tradeoff for $X = \Omega^*(n^{\frac{d-2}{d-1}})$.

Finally, we show that colored orthogonal range counting in dimension $d > 2$ is related to the following generalization of sparse matrix multiplication, which we believe to be of independent interest. One is given a 0-1 matrix A with N nonzero entries in sparse representation, and a list O of M d-tuples of indices of rows of A. Let t be the number of columns in A. The goal is to compute for each tuple $(i_1, \ldots, i_d) \in O$ the sum $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$. We call this problem *$d$-dimensional output restricted sparse matrix multiplication* ($ORSMM_d$). (Note that the case $d = 2$ asks for computing t specified entries in AA^T.) We give a reduction from this problem to colored range counting in boxes in dimension d, showing that the offline version of the latter problem is at least as hard as this generalized matrix multiplication problem. Finally, we describe reasonably efficient algorithms for $ORSMM_d$.

The paper is organized as follows.

In Section 2 we describe the solution to colored orthogonal range counting in $\mathbb{R}^d$, having polylogarithmic query time and about $O(n^d)$ space. The solution is based on decomposition of the c-positive region $Q^+(c)$ (the union of the positive orthants dual to points with color c —see below), for each of the colors c, into pairwise disjoint boxes.

In Section 3 we describe the decomposition of the c-positive region into disjoint boxes and analyze its complexity and construction time. Next, we analyze, for random point sets (according to a natural model that we detail in Section 3.2), the expected number of boxes in our decomposition. This leads to algorithms with polylogarithmic query time and near-linear expected storage and preprocessing cost.

In Section 4 we describe efficient methods for achieving time-space tradeoff. We also consider the time required to answer m queries over an input set of n colored points.

In Section 5 we consider the relation between colored orthogonal range counting in $\mathbb{R}^d$ and sparse matrix multiplication, or its generalized version $ORSMM_d$, as defined above. In addition, we provide efficient algorithms for solving the generalized $ORSMM_d$.

2 Reducing to Standard Orthogonal Range Counting

We first solve the *semi-unbounded colored range counting* problem, in which the query boxes are orthants of the form $\prod_{i=1}^{d}(-\infty, a_i]$. Then we show how to reduce the colored counting problem in general boxes to the semi-unbounded case.

Let us represent each query orthant $\prod_{i=1}^{d}(-\infty, a_i]$ by its apex $(a_1, \ldots, a_d) \in \mathbb{R}^d$. Fix a color c, $1 \le c \le C$, and let P_c denote the subset of points of P with color c. For a point $p \in P_c$, denote by $Q_p^+ \subseteq \mathbb{R}^d$ the locus of all points that represent (closed) query orthants containing p. Clearly, if $p = (p_1, \ldots, p_d)$, then Q_p^+ is the positive orthant $\prod_{i=1}^{d}[p_i, \infty) \subseteq \mathbb{R}^d$. We refer to Q_p^+ as the *dual orthant* of p.

The *c-positive region* of color c, denoted $Q(c)^+$, is the region in $\mathbb{R}^d$ of all points representing queries that contain at least one point of P_c. Clearly, $Q(c)^+ = \bigcup_{p \in P_c} Q_p^+$. For any point set $A \subset \mathbb{R}^d$, define $U(A) := \bigcup_{p \in A} Q_p^+$. According to this definition, $Q(c)^+ = U(P_c)$. In Section 3 we establish the following theorem.

Theorem 2.1. *For any set $A \subset \mathbb{R}^d$ of n points, we can decompose $U(A)$ into $O(n^{\lfloor d/2 \rfloor})$ pairwise disjoint boxes. Furthermore we can generate these boxes in $O(B \log^{d-1} n)$ time, where B is the number of boxes.*

We remark that Theorem 2.1 can be regarded as a refinement of the result of Boissonnat et al. [3] concerning the complexity of the union of n congruent axis-parallel cubes in $\mathbb{R}^d$ (think of the

orthants as very large congruent cubes). As a matter of fact, our constructive proof of the theorem follows similar footsteps to those in the proof of the bound in [3]; see also [8]. In the rest of the paper, unless stated otherwise, decomposition of $U(A)$ means the disjoint decomposition asserted in Theorem 2.1.

We use Theorem 2.1 to solve the semi-unbounded colored range counting problem as follows. For each color $1 \leq c \leq C$, we generate the boxes in the decomposition of $U(P_c) = Q(c)^+$. We build a standard orthogonal range searching data structure for counting the number of boxes containing a query point[4] $q \in \mathbb{R}^d$. By construction (using the fact that the boxes corresponding to any fixed color are pairwise disjoint), this number is equal to the number of distinct colors that appear in the original query orthant.

To implement the data structure, we use a d-dimensional segment tree with fractional cascading at its deepest level. Let B_c be the number of boxes in the decomposition of $Q(c)^+$. Then this structure has query time $O(\log^{d-1} n)$, and space and preprocessing time $O(\sum_{1 \leq c \leq C} B_c \log^{d-1} n)$. These improved bounds on the storage and preprocessing time (by a logarithmic factor, as compared with standard d-dimensional segment trees) are based on the fact, established below in Section 3, that all the decomposition boxes are unbounded in the positive x_1-direction. This allows us to use a simple search tree instead of a segment tree at the bottom level of the data structure, thereby saving a logarithmic factor in both storage and preprocessing time. Fractional cascading takes care care of the corresponding saving in the query time. We thus obtain the following theorem.

Theorem 2.2. *Let P be a set of n colored points in $\mathbb{R}^d$, let B_c be the number of boxes in the decomposition of $Q(c)^+$, for each color $1 \leq c \leq C$, and let $B = \sum_{1 \leq c \leq C} B_c$. Then there exists a data structure supporting semi-unbounded (orthant) colored range counting queries in $O(\log^{d-1} n)$ time, whose storage is $O(B \log^{d-1} n)$ and which can be constructed in $O(B \log^{d-1} n)$ time.*

Using the bounds of Theorem 2.1, we immediately obtain the following corollary.

Theorem 2.3. *There exists a data structure for colored semi-unbounded (orthant) range counting queries on n colored points in dimension $d \geq 2$, which answers a query in $O(\log^{d-1} n)$ time, requires $O(n^{\lfloor d/2 \rfloor} \log^{d-1} n)$ space, and can be constructed in $O(n^{\lfloor d/2 \rfloor} \log^{d-1} n)$ time.*

General orthogonal range counting. The general colored orthogonal range counting problem, in which queries are arbitrary bounded axis-parallel boxes in $\mathbb{R}^d$, can be reduced to the semi-unbounded case in $\mathbb{R}^{2d}$, as follows. We denote the x_i-coordinate of a point p by $x_i(p)$. Double all the coordinates of each point $p = (x_1(p), \ldots, x_i(p), \ldots, x_d(p)) \in P$, to obtain the point $(x_1(p), x_1(p), \ldots, x_i(p), x_i(p), \ldots, x_d(p), x_d(p))$ in $\mathbb{R}^{2d}$, which is given the same color as the original point p. Now, answering a colored range counting query $\prod_{i=1}^{d} [a_i, b_i]$ on the original point set is equivalent to answering the query $\prod_{i=1}^{d} [a_i, \infty) \times (-\infty, b_i]$ on the transformed point set. Thus we obtain the following theorems.

Theorem 2.4. *Let P be a set of n colored points in $\mathbb{R}^d$, let $\tilde{P} \subset \mathbb{R}^{2d}$ be the transformed point set of P as defined above, let B_c be the number of boxes in the decomposition of $U(\tilde{P}_c)$, for color $1 \leq c \leq C$, and let $B = \sum_{1 \leq c \leq C} B_c$. Then there exists a data structure supporting colored range counting queries in $O(\log^{2d-1} n)$ time, whose storage is $O(B \log^{2d-1} n)$ and which can be constructed in $O(B \log^{2d-1} n)$ time.*

Theorem 2.5. *There exists a data structure for colored orthogonal range counting queries on n colored points in dimension $d \geq 2$, which answers a query in $O(\log^{2d-1} n)$ time, requires $O(n^d \log^{2d-1} n)$ space, and can be constructed in $O(n^d \log^{2d-1} n)$ time.*

[4]The problem is called the "stabbing query problem" in [2].

We can generalize this approach to colored orthogonal range counting queries for boxes with bounded projections on k specific coordinates, and semi-unbounded projections on the remaining $d-k$ coordinates. In this case we can reduce the problem to a semi-unbounded problem in $\mathbb{R}^{d+k}$, by duplicating the k "bounded" coordinates of each point in our input set. A query then takes $O(\log^{d+k-1} n)$ time, and the storage and preprocessing cost are both $O(n^{\lfloor (d+k)/2 \rfloor} \log^{k+d-1} n)$.

As a special case, consider the colored orthogonal range counting problem on n points in the plane, where the queries are 3-sided boxes of the form $[a,b] \times (-\infty, c]$. Here $d=2$, $k=1$, and we obtain an algorithm with $O(\log^2 n)$ query time and space and preprocessing cost $O(n \log^2 n)$. These performance parameters are the same as those obtained by Gupta et al. [12], using a different approach based on persistence (which, as already noted, does not seem to extend to higher dimensions).

We can then use the same paradigm as in [12] to extend this solution to the general case of bounded box queries. That is, let $(p_1, \ldots, p_n)$ be the sorted sequence of the points of P in their increasing y-order. For each $i = 1, \ldots, n$, construct the above data structure (for 3-sided queries) for the set $P_i^+ = \{p_j \mid j \geq i\}$. Now, given a query box $[a,b] \times [c,d]$, we find, by binary search, the index i satisfying $y(p_i) \geq c > y(p_{i-1})$, and search with the 3-sided box $[a,b] \times (-\infty, d]$ in the structure of P_i^+. This yields a slightly improved algorithm, in which a query takes $O(\log^2 n)$ time, and the storage and the preprocessing cost are both $O(n^2 \log^2 n)$ (saving a logarithmic factor over the bounds in Theorem 2.5). The same enhancement can be applied in any dimension, leading to the following result.

Theorem 2.6. *There exists a data structure for colored orthogonal range counting queries on n colored points in dimension $d \geq 2$, which answers a query in $O(\log^{2d-2} n)$ time, requires $O(n^d \log^{2d-2} n)$ space, and can be constructed in $O(n^d \log^{2d-2} n)$ time.*

The same enhancement can save a logarithmic factor for any d and k for which $d+k$ is even and ≥ 4. In particular, for $d \geq 4$ even and $k=0$, we can improve all three performance parameters in Theorem 2.3 by a logarithmic factor.

In Section 4 we present a more sophisticated approach that reduces colored box range counting to colored range counting with boxes unbounded in one direction, so as to obtain a much more significant reduction in storage and preprocessing (at the cost of increasing query time).

3 Decomposing the Union of Orthants into Disjoint Boxes

Let A be a set of n points in $\mathbb{R}^d$ in general position, meaning that no two points have the same x_i-coordinate[5], for any $i = 1, \ldots, d$. In this subsection we prove Theorem 2.1 and show how to decompose $U(A)$ into pairwise disjoint boxes.

An open *maximal empty orthant* O (with respect to A) is a region of the form $\prod_{i=1}^{d} (-\infty, a_i)$, where $a_i \in \mathbb{R} \cup \{+\infty\}$ for each i, which does not contain any point of A, and is maximal with this property under inclusion. That is, any open orthant O' that strictly contains O must also contain a point of A. It follows that each facet of O must contain a distinct point of A in its *relative interior*. Let s_i be the point in the relative interior of the facet of O orthogonal to the x_i-axis, for $1 \leq i \leq d$. Thus $O = \prod_{i=1}^{d} (-\infty, x_i(s_i))$ (see Figure 3). Note that we also include under this definition orthants O that are unbounded in the positive direction of some coordinate axes. As an extreme example, the halfspace $x_1 < \min_{s \in A} x_1(s)$ is such a degenerate maximal empty orthant. In such cases, not all the points s_i are defined. (Alternatively, for each coordinate x_i in which O

[5]We can remove the general position assumption by imposing a strict order on the x_i-coordinates of all points for $1 \leq i \leq d$, breaking ties arbitrarily. We then construct the decomposition using the new coordinates. In terms of the old coordinates some of the boxes that we obtain may be empty.

is unbounded in both directions, we can define s_i to be the point at infinity whose x_i-coordinate is $+\infty$ and all other coordinates are $-\infty$.) We say that O is defined by the tuple of points $\langle s_1, \ldots, s_d \rangle$.

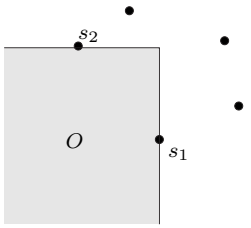

Figure 3: A maximal empty orthant.

Our decomposition of $U(A)$ is constructed so that there is a bijection between its boxes and the maximal empty orthants defined by tuples $\langle s_1, \ldots, s_d \rangle$ such that $x_1(s_1) < \infty$. Specifically, let O be such a maximal empty orthant defined by $\langle s_1, \ldots, s_d \rangle$. The box in our decomposition corresponding to O is

$$B(O) = [x_1(s_1), \infty) \times \prod_{i=2}^{d} \left[\max_{j<i} \{x_i(s_j)\}, x_i(s_i) \right) .$$

Note that each interval in the product is nonempty, which follows from obvious properties of orthants. See Figure 4 for an illustration.

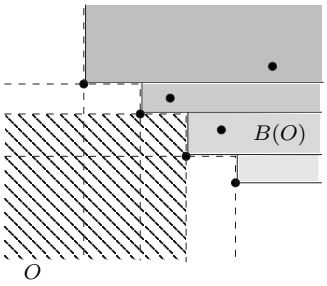

Figure 4: A maximal empty orthant O and its corresponding box $B(O)$ in the decomposition of $U(A)$.

Lemma 3.1. *Let A be a finite point set in $\mathbb{R}^d$, and let $\mathcal{B}$ be the collection of boxes $B(O)$, where O ranges over all maximal empty orthants with respect to A, which are defined by tuples $\langle s_1, \ldots, s_d \rangle$ satisfying $x_1(s_1) < \infty$. Then the boxes of $\mathcal{B}$ are pairwise disjoint, and their union is $U(A)$.*

Proof. We establish the lemma by induction on the dimension of the space containing A.

For the induction basis, consider a one-dimensional set of points $A = \{p_1, \ldots, p_n\} \subset \mathbb{R}$, where $p_1 < p_2 < \cdots < p_n$. The orthant $(-\infty, p_1)$, defined by $\langle p_1 \rangle$, is the only maximal empty orthant with respect to A (which clearly satisfies $x_1(p_1) < \infty$). The corresponding box $B = [x_1(s_1), \infty) = [p_1, \infty)$ is indeed equal to $U(A)$, as required.

Assume now that the lemma is true for any set of points in $\mathbb{R}^{d-1}$. Sweep $U(A)$ with a hyperplane h orthogonal to the x_d-axis. Let $(p_1, \ldots, p_n)$ be the sequence of the points of A sorted in increasing order of their x_d-coordinates. For $1 \leq j \leq n$, put $q_j := x_d(p_j)$, let h_j be the hyperplane $x_d = q_j$, and let H_j be the half-closed slab bounded between h_j and h_{j+1} (containing h_j but not h_{j+1}); let H_n be the infinite slab "beyond" h_n, i.e., the slab $x_d \geq x_d(p_n)$.

By definition, the boxes in our decomposition that intersect H_j correspond to maximal empty orthants defined by tuples $\langle s_1, \ldots, s_d \rangle$, such that $q_{j+1} \leq x_d(s_d)$ (or $x_d(s_d) = \infty$ for $j = n$), $\max_{j<d}\{x_d(s_j)\} \leq q_j$, and $x_1(s_1) < \infty$. See Figure 5 for an illustration. Denote by $\mathcal{O}_j$ the set of these orthants, and denote by $\mathcal{B}(\mathcal{O}_j)$ the corresponding set of boxes. We claim that the intersections of the boxes in $\mathcal{B}(\mathcal{O}_j)$ with H_j form a pairwise disjoint decomposition of $H_j \cap U(A)$.

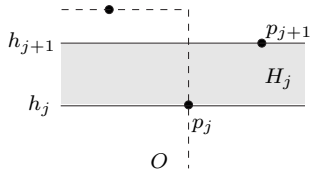

Figure 5: The proof of Lemma 3.1.

Indeed, let $D_j = \{p_i \mid i \leq j\}$, and let D'_j be the orthogonal projection of D_j on the hyperplane h_j. Let $\mathcal{O}'_j$ be the set of maximal empty orthants with respect to D'_j in the $(d-1)$-dimensional space h_j, defined by tuples $\langle s_1, \ldots, s_{d-1} \rangle$ satisfying $x_1(s_1) < \infty$. Let $\mathcal{B}(\mathcal{O}'_j)$ be the corresponding set of boxes within h_j. The following facts are easy to verify.

1. The intersection of $U(A)$ with h_j is equal to the intersection of $U(D_j)$ with h_j, which in turn is equal to $U(D'_j)$.
2. Each maximal empty orthant $O' \in \mathcal{O}'_j$ is the intersection of an orthant $O \in \mathcal{O}_j$ with h_j. Conversely, for each orthant $O \in \mathcal{O}_j$, the $(d-1)$-orthant $O' = O \cap h_j$ is a maximal empty orthant with respect to D'_j.
3. Let $O \in \mathcal{O}_j$ and let $B(O)$ be the corresponding box. The intersection of $B(O)$ with h_j is equal to $B(O')$, where O' is the $(d-1)$-orthant $O \cap h_j$.

See Figure 5 for an illustration. By the induction hypothesis, the boxes of $\mathcal{B}(\mathcal{O}'_j)$ form a pairwise disjoint decomposition of $U(D'_j)$. Furthermore, by (2) and (3), each box $B(O)$ in $\mathcal{B}(\mathcal{O}_j)$ corresponds to a box $B(O')$ in $\mathcal{B}(\mathcal{O}'_j)$, where $O' = O \cap h_j$, so that $B(O') \times [q_j, q_{j+1}) \subseteq B(O)$, and vice versa. Thus the boxes in $\mathcal{B}(\mathcal{O}_j)$ are pairwise disjoint within H_j, and by (1) their union is equal to $U(A) \cap H_j$. Repeating this argument for each slab H_j completes the proof. □

The number of boxes. Let $A \subset \mathbb{R}^d$ be an arbitrary finite set of points. As we have seen, the number of cells in the decomposition of $U(A)$ is bounded by the number of maximal empty orthants with respect to A. Hence, it suffices to bound the latter quantity.

Here is a straightforward constructive derivation of the bound $O(n^{\lfloor d/2 \rfloor})$ for this latter quantity, which resembles the analysis of Boissonnat *et al.* [3]. Given any empty orthant O, with some points of A on its boundary (possibly on lower-dimensional boundary faces), a *shift* of O is the operation of shrinking O by translating a facet of O in the negative direction (of the orthogonal coordinate axis). By the general position assumption, when a shift starts, exactly one point leaves the boundary of O. The shift is *legal* if such a point lies in the relative interior of a facet. A legal shift terminates as soon as one of the points of A on ∂O reaches the relative boundary of the face it is currently on, so that it now lies on a lower-dimensional face. Note that the orthant reached at the end of a legal shift is contained in the original orthant and is thus also empty.

We start with a maximal empty orthant O, and apply to it any sequence of legal shifts, until we reach an orthant O' to which no further legal shifts can be applied. See Figure 6. Upon termination, no point of A lies in the relative interior of any facet of O'. Hence each point of A on $\partial O'$ determines at least two of the coordinate values that define facets of O', and, since we assume general position,

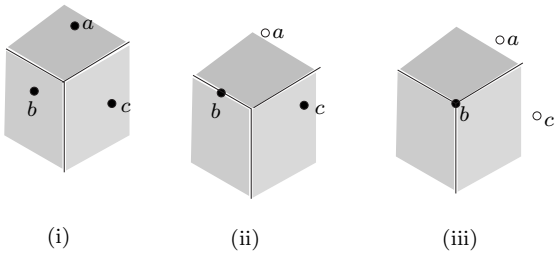

Figure 6: Three legal shifts that turn a maximal empty orthant into the one that has point b as an apex.

no such coordinate value is determined by more than one point. Hence, the number of points on $\partial O'$ is at most $\lfloor d/2 \rfloor$, which implies that the number of such empty orthants O', which can be obtained from some maximal empty orthant in the manner described above, is $O(n^{\lfloor d/2 \rfloor})$. Moreover, knowing the sequence of directions of the legal shifts that have transformed an original maximal empty orthant O into O', the orthant O can be uniquely reconstructed from O', using a sequence of reverse legal shifts, each stopping when the facet that moves outwards hits a new point. Clearly, the total number of such sequences of shifts is at most $d!$. The bound on the number of maximal empty orthants with respect to A follows. This completes the proof of the first part of Theorem 2.1.

3.1 Output sensitive construction

In order to construct the desired decomposition of $U(A)$, it suffices to enumerate all the maximal empty orthants with respect to A, each with the d-tuple defining it. We have seen that each such maximal empty orthant can be defined (up to a constant number of possibilities) by a t-tuple of points of A, such that $t \leq \lfloor d/2 \rfloor$. The constructive proof of the bound on the number of boxes (or, rather, of maximal open orthants) can be trivially converted into an algorithm that generates $O(n^{\lfloor d/2 \rfloor})$ candidate empty orthants, prunes away those that are not empty (using orthogonal emptiness range queries on the set A), and extends each of them, by some sequence (out of $O(1)$ possible ones) of reverse legal shifts, into a maximal empty orthant.

The running time of this straightforward algorithm is close to $O(n^{\lfloor d/2 \rfloor})$, and is always $\Omega(n^{\lfloor d/2 \rfloor})$, which might be much larger than the actual complexity of the decomposition of $U(A)$. In this section, we present an alternative *output sensitive* algorithm for constructing all maximal empty orthants with respect to A. This immediately leads to an *output sensitive* construction of our decomposition of $U(A)$.

We can enumerate all maximal empty orthants with respect to A by implementing the sweep that was used to establish Lemma 3.1. Let $p_1, \ldots, p_n$ be the points in the increasing order of their x_d-coordinates. We sweep $\mathbb{R}^d$ with a hyperplane π orthogonal to the x_d-direction. As above, let $D_j = \{p_i \mid i \leq j\}$, let p'_j be the projection of p_j on π, and let D'_j be the projection of D_j onto π, i.e., $D'_j = \{p'_i \mid i \leq j\}$.

After π passes through p_j, we update the maximal empty ($(d-1)$-dimensional) orthants with respect to D'_j on π. Specifically, we find the set Q of all maximal empty orthants (on π) with respect to D'_{j-1} which are not maximal empty orthants with respect to D'_j (because the projection p'_j of p_j, or rather, p_j itself at this moment, lies in their relative interior). Each such orthant, together with p_j, defines a maximal empty orthant with respect to A, which we output. We delete all orthants in Q, and generate a new set N of maximal empty orthants with respect to D'_j which were not maximal empty $(d-1)$-orthants with respect to D'_{j-1} (those have a $(d-2)$-facet containing p_j in

its relative interior). See Figure 7.

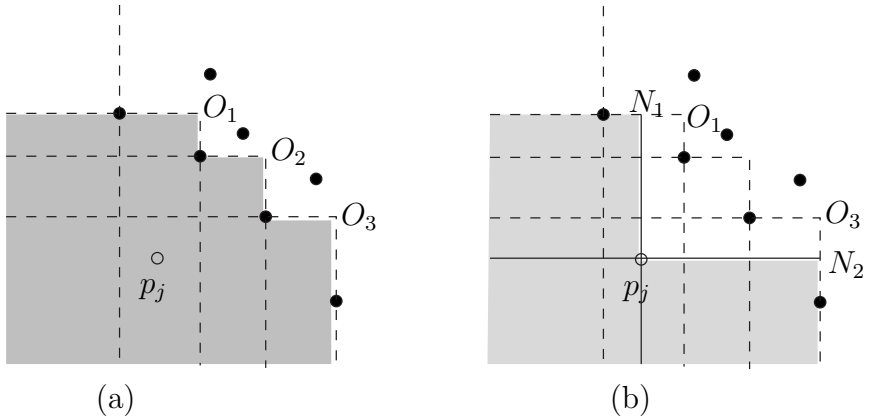

Figure 7: (a) The set Q of $(d-1)$-orthants that are eliminated by p_j. (b) The set N of $(d-1)$-orthants that are generated by p_j. In the example, $|Q| = 3$ and $|N| = 2$. The new orthant N_1 is obtained by shrinking O_1 to the left, and N_2 is obtained by shrinking O_3 downwards.

To efficiently identify the set Q when processing p_j, we maintain the maximal empty $(d-1)$-orthants on π in a dynamic $(d-1)$-dimensional range tree, where each orthant is represented by its apex (a point on π). Using this structure, we can find all k orthants containing p'_j in $O(\log^{d-1} n + k)$ time. We can also delete from, or insert into the dynamic data structure an orthant, in $O(\log^{d-1} n)$ time (see [25] for more details).

To efficiently identify the set N, we make the following observation. Let O be an orthant in N, containing p_j on its facet which is orthogonal to the x_i-axis, for some i. Ignoring p_j, and shifting the facet containing p_j away from O, we either hit a point of D'_{j-1}, or have this facet of O reach infinity. In both cases, we end up with an orthant O' of Q. Hence, each orthant in N can be obtained by taking an orthant O' of Q, and by replacing a facet of O' by a parallel facet through p_j. (Not all orthants obtained in this manner are valid: one also needs to ensure that each facet of the shrunk orthant, other than the one through p_j, still contains a point of D'_j in its relative interior.)

It is easily verified that the total number of updates to the dynamic range tree is bounded by the number H of maximal empty orthants with respect to A. Indeed, this is clear for orthants in Q: each such orthant O corresponds to a unique maximal empty d-orthant that is "completed" when the sweep hyperplane reaches its present position. Each orthant O in N will either be completed into a maximal empty d-orthant when it hits a new point at some future position of the sweep hyperplane, or else survive until the sweep is completed, and then become a degenerate d-orthant, unbounded in both directions of the x_d-axis. Hence, the decomposition of $U(A)$ can be constructed in time $O(H \log^{d-1} n)$.

It remains to prove that the number B of boxes in the decomposition of $U(A)$, is $O(H)$. Note that not every maximal empty orthant induces a decomposition cell $B(O)$, because O may be unbounded in the positive direction of the x_1-axis, having $x_1(s_1) = \infty$. Instead, we claim that each maximal empty orthant corresponds to a distinct face on the boundary of $U(A)$. This is sufficient, since the total complexity of the boundary of $U(A)$ is clearly $\Theta(B)$. To establish the claim, let O be a maximal empty orthant, defined by the d-tuple $< s_1, \ldots, s_d >$. Let F be the set of all points o, such that for $1 \leq i \leq d$, $x_i(o) = x_i(s_i)$ if $x_i(s_i) < \infty$, and $x_i(o) \geq \max_{1 \leq j \neq i \leq d}\{x_i(s_j)\}$ otherwise. First, observe that o belongs to the boundary of $U(A)$. Moreover, let o' be a point obtained by shifting o in the positive x_i-direction, for some $1 \leq i \leq d$ such that $x_i(s_i) < \infty$. Clearly, the

(negatively-oriented) orthant O' whose apex is o' contains s_i in its interior. Hence, o' does not belong to $U(A)$. Let k be the number of indices $1 \leq i \leq d$ such that $x_i(s_i) = \infty$. Then F is a k-dimensional face, on the boundary of $U(A)$, having nonzero (and unbounded) extent in exactly the k coordinates just defined. Moreover, as easy to check, O is uniquely defined by F.

This implies the second part of Theorem 2.1, and thus completes its proof. □

3.2 Random sets of points

The decomposition-based data structure of Theorem 2.5 is especially efficient, when the number of maximal empty orthants with respect to each of the sets $P_c \subset \mathbb{R}^d$, for $1 \leq c \leq C$, are all small. A *maximal empty box* with respect to a point set P is an orthogonal axis-parallel box, and which does not contain any point of P in its interior, and is maximal with this property under inclusion (so, as in the case of orthants, each facet of the box contains a point of P in its relative interior). We allow a maximal empty box to be unbounded in certain directions, so every maximal empty orthant with respect to P is also a maximal empty box. In this subsection we prove that the expected number of maximal empty boxes for a random point-set with n points in $\mathbb{R}^d$ is only $O(n \log^{d-1} n)$. It would be interesting to explore the connection between this result and the known bound of $O(\log^{d-1} n)$ on the number of *maximal* points in a set of n random points in $\mathbb{R}^d$; see [1, 17].

We assume that the set P is constructed so that each point is chosen independently and uniformly at random from the uniform distribution on $[0,1]^d$. Thus, with probability 1, the points of P are in general position, and, for each i, the x_i-coordinates of the sampled points form a random permutation, which are independent of each other. We define a *t-box* to be an axis-parallel box B, that may be unbounded in certain directions, containing t points on its boundary, such that each one of the finite facets of B contains a point of P (possibly on its relative boundary). Let $B_{t,k}(P)$ (resp., $B_{t,\leq k}(P)$), for $1 \leq t \leq 2d$, denote the set of t-boxes containing exactly (resp., at most) k points of P in their interior, and put $N_{t,\leq k}(P) = |B_{t,\leq k}(P)|$. Note that we may have degenerate boxes with identical opposite facets. However, the number of such boxes is at most $O(n)$, due to the general position assumption. Finally, we let $N^{(d)}_{t,k}(n)$ (resp., $N^{(d)}_{t,\leq k}(n)$) denote the expected value of $|B_{t,k}(P)|$ (resp., $|B_{t,\leq k}(P)|$), over the random choice of a set P of n points in $\mathbb{R}^d$. In order to obtain the asserted bound on the expected number of maximal empty boxes with respect to P, it suffices to show that $N^{(d)}_{t,0}(n) = O(n \log^{d-1} n)$, for all $1 \leq t \leq 2d$.

We prove the bound by induction on d. The case $d = 1$ is trivial: We clearly have $N^{(1)}_{1,0}(n)$ is $O(n)$, since any empty non-degenerate 1-box in $\mathbb{R}^1$ is an empty ray emanating from a point of P. Similarly, $N^{(1)}_{2,0}(n) = n - 1$, since any empty 2-box is a segment bounded by a pair of points of P.

Assume now that $d > 1$, and that the bound holds for $d - 1$ (and for all $t \leq 2(d-1)$). Assume also (without really changing the model) that we draw each point $p \in P$ by first drawing a point $p' \in [0,1]^{d-1}$, thereby fixing the projection of p on the hyperplane $x_d = 0$ to be p', and then drawing x_d uniformly from $[0,1]$ to be the x_d-coordinate of p.

Let $P' = \{p' \mid p \in P\}$ be the set of projections of all points in P on the hyperplane $x_d = 0$. Let $\tilde{B}_{t,0}(P)$ denote the set of empty t-boxes of P whose both facets orthogonal to the x_d-axis contain a point of P on their relative boundary. Set $\tilde{N}_{t,0}(P) = |\tilde{B}_{t,0}(P)|$ and $\tilde{N}_{t,0}(n) = \max_{|P|=n} \tilde{N}_{t,0}(P)$. Every t-box B in $\tilde{B}_{t,0}(P)$ corresponds to the t-box B' of P', obtained by projecting B onto $x_d = 0$; note, however, that B' is not necessarily empty.

Fix P', and consider the random drawings of the x_d-coordinates of the points of P. What is the probability that a t-box B' in $B_{t,k}(P')$ corresponds to an empty t-box B in $\tilde{B}^d_{t,0}(P)$? For this to happen, it is necessary and sufficient that the t boundary points of B' be consecutive in the permutation defined by the $t + k$ boundary and interior points of B' along the x_d-axis (see Figure 8). This happens with probability $\dfrac{t!(k+1)!}{(k+t)!} = \dfrac{t!}{(k+2)(k+3)\cdots(k+t)}$. So we obtain the

following inequality:

$$\mathbb{E}(\tilde{N}_{t,0}(P)) \leq \sum_{k=0}^{n-t} \frac{t!}{\prod_{i=2}^{t}(k+i)} |B_{t,k}(P')|.$$

Rearranging this sum, we get:

$$\mathbb{E}(|\tilde{B}_{t,0}(P)|) \leq \sum_{k=0}^{n-t} \frac{t!(t-1)}{\prod_{i=2}^{t+1}(k+i)} |B_{t,\leq k}(P')| + \frac{t!}{\prod_{i=2}^{t}(n-t+i)} |B_{t,\leq n-t}(P')|.$$

This holds for every choice of P', so if we average over the choices of P', we obtain the following recurrence:

$$\tilde{N}_{t,0}^{(d)}(n) \leq \sum_{k=0}^{n-t} \frac{t!(t-1)}{\prod_{i=2}^{t+1}(k+i)} N_{t,\leq k}^{(d-1)}(n) + \frac{t!}{\prod_{i=2}^{t}(n-t+i)} N_{t,\leq n-t}^{(d-1)}(n). \tag{1}$$

We next consider empty t-boxes, for which neither of their facets orthogonal to the x_d-axis contain points of P in their relative boundaries. We only consider boxes that are not strips bounded by one or two hyperplanes orthogonal to the x_d-direction (there are $O(n)$ such strips). Any such box B can be charged to a box in $\tilde{B}_{t',0}(P)$, for some $t-2 \leq t' \leq t$, by applying two legal shifts to B (as in Section 3), the first shifting the top facet down, and the second shifting the bottom facet up. With some care, this also applies to boxes that are unbounded in the x_d-direction. The resulting box $\tilde{B}$ is uniquely charged in this manner. Similarly, an empty t-box B, such that one of its facets orthogonal to the x_d-axis does not contain a point in the relative boundary, can be charged to a box $\tilde{B}$ in $\tilde{B}_{t',0}(P)$, for some $t-1 \leq t' \leq t$. This can be done by applying just one legal shift that shifts that facet. Such a $\tilde{B}$ is charged at most twice. Hence we have

$$N_{t,0}^{(d)}(n) = O(\tilde{N}_{t,0}^{(d)}(n) + \tilde{N}_{t-1,0}^{(d)}(n) + \tilde{N}_{t-2,0}^{(d)}(n) + n). \tag{2}$$

To estimate $N_{t,\leq k}^{(d)}(n)$, for $k \geq 1$, we note that if we sample a random subset of P of size n/k, we obtain a random set of n/k points drawn independently from the same uniform distribution as P. Hence, we can apply the Clarkson-Shor probabilistic technique [9] to conclude that

$$N_{t,\leq k}^{(d)}(n) = O(k^t N_{t,0}^{(d)}(n/k)). \tag{3}$$

By the induction hypothesis we have $N_{t,0}^{(d-1)}(n) = O(n \log^{d-2} n)$, for any $1 \leq t \leq 2d$, and for any n. Hence, by (3), $N_{t,\leq k}^{(d)}(n) = O(nk^{t-1} \log^{d-2} n)$, for any $1 \leq t \leq 2d$. Plugging this into inequality (1) and using (2), we obtain the bound $N_{t,0}^{(d)}(n) = O(n \log^{d-1} n)$, for $1 \leq t \leq 2d$.

We thus have proven the following theorem.

Theorem 3.2. *Let P be a set of n points in $\mathbb{R}^d$, drawn independently from the uniform distribution on $[0,1]^d$. Then the expected number of maximal empty boxes with respect to P is $O(n \log^{d-1} n)$, where the constant of proportionality depends on d.*

In particular, the bound of Theorem 3.2 also applies to the expected number of maximal empty orthants. Using Theorem 2.2, we obtain the following result.

Theorem 3.3. *Let P be a set of n colored points in $\mathbb{R}^d$, such that each point is drawn independently from the uniform distribution on $[0,1]^d$. Then there exists a data structure supporting semi-unbounded colored range counting queries in $O(\log^{d-1} n)$ time, whose expected storage is $O(n \log^{2d-2} n)$ and which can be constructed in expected $O(n \log^{2d-2} n)$ time.*

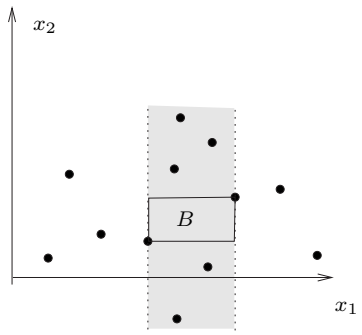

Figure 8: Bounding the expected number of empty boxes in the plane. The highlighted box B belongs to $\tilde{B}_{2,0}(P)$, and its x-projection belongs to $B_{2,5}(P')$. The two points defining B must be consecutive in the y-order of the points in the shaded strip.

General colored range counting for random sets. Recall that we solve the problem when queries are general bounded boxes by a reduction to the semi-bounded problem. However, since this reduction transforms the points by doubling each coordinate, we can no longer assume that the transformed set satisfies our randomness assumption.

We overcome this difficulty using the following observation. Let A be a random point set in $\mathbb{R}^d$ and $\hat{A}$ the point set in $\mathbb{R}^{2d}$ into which A is mapped. It follows by definition that every maximal empty orthant in $\mathbb{R}^{2d}$ with respect to $\hat{A}$ corresponds to a maximal empty box in $\mathbb{R}^d$ with respect to A. Hence, we can still apply Theorem 3.2 to the original set A in order to upper-bound the number of maximal empty orthants with respect to $\hat{A}$. Using the data structure of Theorem 2.4 we obtain the following result.

Theorem 3.4. *Let P be a random colored point-set of cardinality n in $\mathbb{R}^d$, such that the points of each color are selected as above. Then there exists a data structure supporting colored range counting queries in $O(\log^{2d-1} n)$ time, whose expected storage is $O(n \log^{3d-2} n)$ and which can be constructed in expected $O(n \log^{3d-2} n)$ time.*

Using the same technique, we can build a data structure to answer colored range counting queries for boxes with bounded projections on k specific coordinates, for a random point set, by reducing the problem to a semi-unbounded problem in $\mathbb{R}^{d+k}$ and using Theorem 3.2.
Remark: The enhancement that shaves off a logarithmic factor, as provided in Theorem 2.6, does not apply in this case, because it is based on n copies of the structure, and the expected number of maximal empty orthants may be linear in each subproblem, giving rise to an overall data structure of super-quadratic size.

4 Achieving Time-Space Tradeoff

In this section we present several techniques for reducing storage at the expense of increasing query time.

4.1 The planar case: Splitting boxes

Our tradeoff technique for the planar case uses the more space efficient data structure of Theorem 2.3 that supports colored range counting queries with 3-sided boxes of the form $[a, b] \times (-\infty, d]$ or $[a, b] \times [c, \infty)$. Recall that this problem can be transformed into a standard orthogonal containment searching in $\mathbb{R}^3$, and that the resulting data structure requires only $O(n \log^2 n)$ storage and

preprocessing cost. For the present solution, instead of just counting colors, we want to output the set of all colors in the query 3-sided box as a union of canonical subsets of colors. Since the data structure has three levels, the number of such sets in a query output is $O(\log^3 n)$, and each color appears in at most one of these sets (because of the disjointness of the boxes in the decomposition of any single $U(P_c)$).

The full data structure. In order to handle general queries of the form $[a,b] \times [c,d]$, we use the following technique, which has already been used in [12], for solving orthogonal colored range *reporting* problems. We store the points of P, in the increasing order of their y-coordinates, at the leaves of a balanced binary tree T. At each internal node v, we store a pair of auxiliary data structures, one for answering queries of the form $[a,b] \times [c,\infty)$, and one for queries of the form $[a,b] \times (-\infty,d]$. The first (resp., second) structure is built on the points stored at the left (resp., right) subtree of v. We also store at v a y-coordinate $Y(v)$ that separates the y-coordinates of the points stored at the left subtree from those stored at the right subtree of v. If v is a leaf, we let $Y(v)$ be the y-coordinate of the singleton point stored at v.

For each node t of some tree at the deepest (third) level of the structure, denote by $c(t)$ the set of colors represented in the canonical subset of t.

Let $q = [a,b] \times [c,d]$ be a query box. We search down the tree T with $[c,d]$ and denote by v_q the highest node for which $[c,d]$ contains $Y(v_q)$. If v_q is a leaf, we check whether the single point that it stores lies in q, and return 1 if it does and 0 otherwise. If v_q is an internal node, we query the two structures at v_q with $[a,b] \times [c,\infty)$ and with $[a,b] \times (-\infty,d]$, respectively. Let D_q (resp., U_q) denote the set of $O(\log^3 n)$ canonical nodes that are returned by the former (resp., latter) sub-query. Observe that the set of colors in q is exactly the union of the canonical sets of colors associated with the nodes in $U_q \cup D_q$. Another crucial property (which follows from the fact that, for each color c, the boxes that represent $U(P_c)$ are pairwise disjoint) is that each color in the output appears in the canonical set of at most one node of U_q and at most one node of D_q (see [12] for more details). Hence, using the exclusion-inclusion principle, the number of colors that appear in q is equal to

$$\sum_{s \in U_q} |c(s)| + \sum_{t \in D_q} |c(t)| - \sum_{s \in U_q,\, t \in D_q} |c(s) \cap c(t)|. \tag{4}$$

Ideally, we would like to have for every $s \in U_q$ and $t \in D_q$ the value of $|c(s) \cap c(t)|$ pre-stored. However, doing this for every possible pair of canonical sets would be too expensive, and may result in super-quadratic space complexity. Nevertheless, if we did have these values available, answering a query could then be done in $O(\log^6 n)$ time, using (4).

Instead, we derive a tradeoff between storage and query time, determined by a threshold parameter X in the range $1 \le X \le n$. Let $\mathcal{D}(v)$ (resp., $\mathcal{U}(v)$) be the set of canonical nodes in the auxiliary structure of node $v \in T$ used for answering queries of type $[a,b] \times [c,\infty)$ (resp., $[a,b] \times (-\infty,d]$). A node $t \in \mathcal{D}(v) \cup \mathcal{U}(v)$ is called *X-heavy* if $|c(t)| > X$; otherwise it is called *X-light*. For every node $v \in T$ we construct and store, as part of the preprocessing stage, a matrix $M(v)$, whose rows and columns correspond to the X-heavy nodes in $\mathcal{D}(v)$ and $\mathcal{U}(v)$, respectively. For each pair of X-heavy canonical nodes $s \in \mathcal{U}(v)$ and $t \in \mathcal{D}(v)$, we store in $M_{s,t}(v)$ the value of $|c(t) \cap c(s)|$. Let n_v be the number of points stored at the subtree of T rooted at v. The overall size of all the canonical subsets associated with the nodes of $\mathcal{U}(v) \cup \mathcal{D}(v)$ is $O(n_v \log^3 n)$. Hence, the number of X-heavy canonical nodes in $\mathcal{D}(v) \cup \mathcal{U}(v)$ is $O\left(\frac{n_v}{X} \log^3 n\right)$, so $M(v)$ has size $O\left(\left(\frac{n_v}{X}\right)^2 \log^6 n\right)$. (The time needed to construct $M(v)$, for all nodes $v \in T$, is discussed later.) Summing this bound over all nodes v of T, we get an overall bound of $O(\left(\frac{n}{X}\right)^2 \log^2 n)$.

For each canonical node $t \in \mathcal{D}(v) \cup \mathcal{U}(v)$, in addition to storing the size $|c(t)|$ of its canonical subset, we also store a dictionary data structure on $c(t)$ (implemented as a binary search tree),

supporting logarithmic-time searches. Clearly, since $\sum_{t\in\mathcal{D}(v)\cup\mathcal{U}(v)} |c(t)| = O(n_v \log^3 n)$, these additional structures take a total of $O(n\log^4 n)$ extra storage.

The total space complexity of our solution is thus $O\left(\left(\frac{n}{X}\right)^2 \log^6 n + n\log^4 n\right)$.

Handling queries. It suffices to describe how to compute $|c(s)\cap c(t)|$ efficiently, for each $s \in U_q$ and $t \in D_q$. If both s and t are X-heavy, then this value is stored in $M(v_q)$ and we simply retrieve it. Otherwise, we can assume, without loss of generality, that $|c(s)| \leq X$. In this case we check, for each $c \in c(s)$, whether $c \in c(t)$, and count the number of such colors. This takes a total of $O(X\log n)$ time. We repeat this for each pair $(s,t) \in U_q \times D_q$, for a total of $O(X\log^7 n)$ query time.

Preprocessing. Clearly, T with all its auxiliary data structures can be constructed in time $O(n\log^3 n)$ and the dictionary structures at the nodes of $|\mathcal{U}\cup\mathcal{D}|$ can be constructed in overall time $O(n\log^4 n)$. It remains to describe the construction of matrices M_v.

Denote by $\mathcal{D}$ (resp., $\mathcal{U}$) the set of canonical nodes in all the auxiliary structures used for answering queries of type $[a,b]\times[c,\infty)$ (resp., $[a,b]\times(-\infty,d]$). Let M_D (resp., M_U) denote the matrix whose rows correspond to the X-heavy nodes of $\mathcal{D}$ (resp., of $\mathcal{U}$), and whose columns correspond to the colors $1,\ldots,C$, such that, for $t\in\mathcal{D}$ (resp., $t\in\mathcal{U}$) and color c, the (t,c)-entry of the matrix is 1 if $c\in c(t)$ and is 0 otherwise. It follows that $M = M_D {M_U}^T$ contains all the matrices M_v as submatrices[6].

Using the fact that $\sum_v n_v = O(n\log n)$, we get that each of the matrices M_D and M_U has $t = O\left(\frac{n}{X}\log^4 n\right)$ rows and $N = O(n\log^4 n)$ non-zero entries. Hence, we can construct M using the sparse rectangular matrix multiplication technique of Kaplan *et al.* [20] (which extends a technique of Yuster and Zwick [26] and of Chan [7]). Specifically, for two matrices A, B, each having t rows and at most N non-zero items, these methods construct AB^T in time

$$\begin{cases} O(Nt^{\frac{\omega-1}{2}}) & \text{if } N \geq t^{\frac{\omega+1}{2}}, \\ O(N^{\frac{2\beta}{\beta+1}} t^{\frac{2-\alpha\beta}{\beta+1}}) & \text{if } t^{1+\frac{\alpha}{2}} \leq N \leq t^{\frac{\omega+1}{2}}, \\ O(t^2) & \text{if } N \leq t^{1+\frac{\alpha}{2}}. \end{cases}$$

Here (i) ω is the exponent of matrix multiplication, i.e., ω is the smallest number such that two $t\times t$ matrices can be multiplied in time $O(t^\omega)$; (ii) α is the largest value of r for which a $t\times t^r$ matrix and a $t^r\times t$ matrix can be multiplied in time $O(t^2)$; and (iii) $\beta = \frac{\omega-2}{1-\alpha}$. It is known (see, e.g., [18]) that $\omega < 2.376$ and $\alpha > 0.294$; thus we can use $\beta \approx 0.533$.

Hence, the construction time of the data structure is:

$$\begin{cases} O^*\left(\frac{n^{(\omega+1)/2}}{X^{(\omega-1)/2}}\right) = O\left(\frac{n^{1.688}}{X^{0.688}}\right) & \text{when } X \geq n^{\frac{\omega-1}{\omega+1}} \approx n^{0.408}, \\ O^*\left(\frac{n^{\frac{2-\alpha\beta+2\beta}{\beta+1}}}{X^{\frac{2-\alpha\beta}{\beta+1}}}\right) = O\left(\frac{n^{1.898}}{X^{1.203}}\right) & \text{when } n^{\frac{\alpha/2}{\alpha/2+1}} \approx n^{0.128} \leq X \leq n^{\frac{\omega-1}{\omega+1}} \approx n^{0.408}, \\ O^*(\frac{n^2}{X^2}) & \text{when } X \leq n^{\frac{\alpha/2}{\alpha/2+1}} \approx n^{0.128}. \end{cases} \tag{5}$$

We thus have the following result.

Theorem 4.1. *Let P be a set of n colored points in the plane, and let $1\leq x\leq n$ be a given tradeoff parameter. Then we can preprocess P into a data structure of size $O\left(\left(\frac{n}{X}\right)^2\log^6 n + n\log^4 n\right)$ so that a colored range counting query can be answered in $O(X\log^7 n)$ time. The preprocessing cost is as given in (5).*

[6] We combine all the matrices M_v into one matrix M to simplify the presentation. In doing so, we lose a polylogarithmic factor in the time bound.

Optimizing for a fixed number of queries. Suppose next that we know (or guess) in advance the number m of queries. We can then optimize the choice of X, so as to minimize the overall time for answering m colored range counting queries, including the time spent at the preprocessing stage. A simple calculation yields:

Corollary 4.2. *Let P be a set of n colored points in the plane.*
(a) $m \leq n$ colored range counting queries can be answered in overall time $O^*(nm^{\frac{\omega-1}{\omega+1}}) = O(nm^{0.408})$.
(b) $n \leq m \leq n^{\frac{4+4\beta-\alpha\beta-\alpha}{(\alpha+2)(\beta+1)}} \approx n^{1.616}$ colored range counting queries can be answered in overall time $O^*(n^{\frac{2-\alpha\beta+2\beta}{2-\alpha\beta+\beta+1}} m^{\frac{2-\alpha\beta}{2-\alpha\beta+\beta+1}}) = O(n^{0.862}m^{0.546})$.
(c) $m \geq n^{\frac{4+4\beta-\alpha\beta-\alpha}{(\alpha+2)(\beta+1)}} \approx n^{1.616}$ colored range counting queries can be answered in overall time $O^*(m^{2/3}n^{2/3})$.

In particular, n colored range counting queries can be answered in overall time $O(n^{\frac{2\omega}{\omega+1}}) = O(n^{1.408})$, including time spent at the preprocessing stage.

4.2 Tradeoff in higher dimensions

In higher dimensions, there are other approaches to achieving a tradeoff between query time and storage.

Bucketing. Partition the set of colors into $O(\log n)$ "buckets" $\mathcal{C}_i$, such that $c \in \mathcal{C}_i$ if and only if $2^{i-1} \leq |P_c| < 2^i$. Put $C_i := |\mathcal{C}_i|$, and let n_i be the number of points having colors in $\mathcal{C}_i$. Clearly, $n_i = \Theta(2^i C_i)$. We solve the colored range counting problem separately within each $\mathcal{C}_i$, and output the sum of the counts obtained in each of these $O(\log n)$ subproblems.

Again, we fix some threshold parameter $1 \leq X \leq n$. Answering a colored range counting query with a box q within $\mathcal{C}_i$ depends on the relationship between X and C_i. If $X \geq C_i$, we simply test, for each color c in $\mathcal{C}_i$, whether $q \cap P_c \neq \emptyset$, and count up the number of colors with this property. Using the standard orthogonal range searching machinery [2], this takes $O(\log^{d-1} |P_c|) = O(i^{d-1})$ time per color c, for a total of $O(i^{d-1}C_i) = O(i^{d-1}X)$ time. Summing these bounds over all buckets with $X \geq C_i$, we obtain a total of $O(X \log^d n)$ time.

If $X < C_i$, we use the technique of Theorem 2.5 for answering the query. The query time is $O(\log^{2d-1} n_i)$, and the space and preprocessing cost are both

$$O\left(C_i \cdot 2^{id} \log^{2d-1} n_i\right) = O\left(\frac{n_i^d}{X^{d-1}} \log^{2d-1} n\right)$$

(we use here the facts that $n_i/C_i = \Theta(2^i)$ and that $X < C_i$). Hence, summing over the buckets, and adding the costs for buckets with $X \geq C_i$, the overall query time is $O(X \log^d n + \log^{2d-1} n)$. (For this, we use a single colored range counting data structure for all the buckets with $C_i > X$.) The overall preprocessing and space complexity is $O\left(\frac{n^d}{X^{d-1}} \log^{2d-1} n\right)$. Optimizing the choice of X, we can answer m colored range counting queries in time $O(nm^{1-1/d} \log^{d+1-1/d} n + m \log^{2d-1} n)$, including time spent at the preprocessing stage. We thus have the following result.

Theorem 4.3. *Let P be a set of n colored points in $\mathbb{R}^d$, $d > 2$, and let $1 \leq X \leq n$ be a tradeoff parameter. We can preprocess P, in time $O\left(\frac{n^d}{X^{d-1}} \log^{2d-1} n\right)$, into a data structure of size $O\left(\frac{n^d}{X^{d-1}} \log^{2d-1} n\right)$, which supports colored range counting queries in time $O(X \log^d n)$. In particular, m such queries can be answered in total time $O(nm^{1-1/d} \log^{d+1-1/d} n + m \log^{2d-1} n)$, including the cost of preprocessing.*

Two additional methods are presented in Subsection 4.3. They partially improve the upper bounds in the time-space tradeoff for $X = \Omega^*(n^{\frac{d-2}{d-1}})$, but they are more expensive at the preprocessing stage. Ignoring storage, the bucketing method described above provides the best upper bound so far on the time required to answer any fixed number of queries, if time spent at the preprocessing stage is also included.

4.3 Additional time-space tradeoffs in dimension $d > 2$

Bucketing with box-splitting. We further improve the tradeoff achieved by bucketing for values of $X = \Omega^*(n^{\frac{d-2}{d-1}})$, by using the box-splitting technique within each bucket, as in the planar case. Specifically, apply bucketing with some threshold value $1 \le X \le n$. Again, if the number of colors within a fixed bucket $\mathcal{C}_i$ is smaller than X, test each of the C_i colors in the bucket for intersection with the query range. This costs $O(X \log^{d-1} n)$ time per bucket, for a total of $O(X \log^d n)$.

Consider then buckets with $C_i > X$. Apply the box splitting technique for the colors in $\mathcal{C}_i$ (using the same parameter X). That is, construct a binary tree on the points sorted by their x_1-coordinates. For each node of the tree we maintain two auxiliary structures for querying with boxes that are semi-unbounded in the x_1-direction. As above, the data structures can be implemented so that processing a query takes $O(\log^{2d-1} n)$ time, and returns the output as the disjoint union of $O(\log^{2d-1} n)$ canonical sets, each stored at some node of the structure. The preprocessing and space complexity of the structure are both $O(C_i \cdot (n_i/C_i)^{d-1} \log^{2d-1} n_i) = O(n_i^{d-1}/X^{d-2} \log^{2d-1} n)$.

To process a bounded-box query, we proceed as above, finding the highest node in the tree that splits the x_1-span of the query box, performing appropriate queries in the auxiliary data structures with the two corresponding semi-unbounded "half-boxes", and combining the solutions, using an appropriate variant of (4). The time of the query is then $O(X \log^{4d-1} n)$ (extending the analysis from the planar case).

Since the total size of all canonical subsets of nodes of the respective collections $\mathcal{U}(v)$, $\mathcal{D}(v)$ is now $O(n_v^{d-1}/X^{d-2} \log^{2d-1} n)$, for a given vertex $v \in T$, the size of the matrix $M(v)$ is

$$O((n_v^{d-1}/X^{d-2}/X)^2) \log^{4d-2} n) = O((n_v/X)^{2d-2} \log^{4d-2} n), \tag{6}$$

and the total storage, over all nodes v and buckets i, is thus

$$O\left(n^{d-1}/X^{d-2} \log^{2d-1} n + (n/X)^{2d-2} \log^{4d-2} n\right). \tag{7}$$

Comparing this with the bound for the bucketing technique alone, we get an improved tradeoff for $X = \Omega^*(n^{\frac{d-2}{d-1}})$, as asserted.

It remains to bound the preprocessing time, which is dominated by the cost of computing the matrix M in each bucket. The analysis is similar to the one in Section 4.1, and it yields the bounds

$$\begin{cases} O^*\left(\frac{n^{(d-1)(\omega+1)/2}}{X^{(d-1)(\omega+1)/2-1}}\right) & \text{when } n^{\frac{(d-1)(\omega-1)}{(d-1)(\omega-1)+2}} \le X, \\ O^*\left(\frac{n^{(d-1)\frac{2\beta+2-\alpha\beta}{\beta+1}}}{X^{(d-2)\frac{2\beta}{\beta+1}+(d-1)\frac{2-\alpha\beta}{\beta+1}}}\right) & \text{when } n^{\frac{(d-1)\alpha}{(d-1)\alpha+2}} \le X \le n^{\frac{(d-1)(\omega-1)}{(d-1)(\omega-1)+2}}, \\ O^*\left(\frac{n^{2d-2}}{X^{2d-2}}\right) & \text{when } X \le n^{\frac{(d-1)\alpha}{(d-1)\alpha+2}}. \end{cases} \tag{8}$$

That is, we have:

Theorem 4.4. *Let P be a set of n colored points in $\mathbb{R}^d$, $d > 2$, and let $1 \le X \le n$ be a tradeoff parameter. We can preprocess P, in time as in (8), into a data structure of size*

$$O\left(n^{d-1}/X^{d-2} \log^{2d-1} n + (n/X)^{2d-2} \log^{4d-2} n\right),$$

which supports colored range counting queries in time $O(X \log^{4d-1} n)$.

Grid. Using this method, we achieve additional improvement of the time-space tradeoff for $X = \Omega^*(n^{\frac{d+1}{d+2}})$. Again, we assume general position of the points of P, in the sense that no two points have the same x_i-coordinate for any $1 \leq i \leq d$. We fix a parameter t, and partition $\mathbb{R}^d$, for each $1 \leq i \leq d$, into t slabs, each containing n/t points of P and bounded by two hyperplanes orthogonal to the x_i-axis. These partitions, when superimposed on each other, induce a non-uniform grid G with t^d cells. There are $O(t^{2d})$ "canonical" boxes, each bounded by $2d$ grid hyperplanes. We preprocess each of these boxes, and compute the number of colors that it contains. This can be done, using the bucketing method, in time $O(t^{2d-2} n \log^{d+1-1/d} n + t^{2d} \log^{2d-1} n)$.

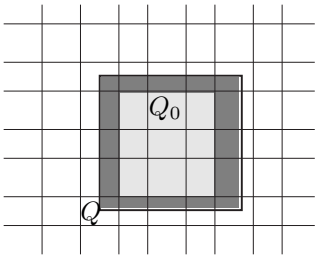

Figure 9: The grid construction in the plane, a query rectangle Q, its (lightly-shaded) "core" Q_0, and its (darkly-shaded) fringe.

To perform an actual query with some box Q, we find the maximal canonical box Q_0 that it contains, and retrieve the number C_0 of colors in Q_0. We then retrieve the $n' \leq 2dn/t$ points of P that lie in the "fringe" $Q \setminus Q_0$ of Q, using a standard orthogonal range *reporting* data structure [2]. See Figure 9. For each such point p, we check whether its color $c(p)$ is a color that appears in Q_0. For this, we preprocess each of the monochromatic subsets P_c of P, for $c = 1, \ldots, C$, for d-dimensional orthogonal emptiness range queries. For each $p \in Q \setminus Q_0$, with color $c = c(p)$, such that this is the first fringe point of that color, we test whether $Q_0 \cap P_{c(p)} = \emptyset$ using the corresponding structure, in $O(\log^{d-1} n)$ time. If this is the case, we add 1 to the color count. In all other cases p is ignored. The overall query time is $O\left(\frac{n}{t} \log^{d-1} n\right)$. The data structure uses $O(n \log^{d-1} n + t^{2d})$ storage, and can be constructed in time $O(t^{2d-2} n \log^{d+1-1/d} n + t^{2d} \log^{2d-1} n)$. Alternatively, substituting $X := \frac{n}{t}$, we obtain tradeoff bounds that are similar to the previous ones (we leave it to the reader to verify that we get an improvement for $X = \Omega^*(n^{\frac{d+1}{d+2}})$).

Theorem 4.5. *Let P be a set of n colored points in $\mathbb{R}^d$, $d > 2$, and let $1 \leq X \leq n$ be a tradeoff parameter. We can preprocess P, in time $O\left(\left(\frac{n}{X}\right)^{2d-2} n \log^{d+1-1/d} n + \left(\frac{n}{X}\right)^{2d} \log^{2d-1} n\right)$, into a data structure of size $O((\frac{n}{X})^{2d} + n \log^{d-1} n)$, which supports colored range counting queries in time $O(X \log^{d-1} n)$.*

5 Colored Range Counting and Sparse Matrix Multiplication

In this section we further elaborate on the relation between sparse matrix multiplication and colored orthogonal range counting. We use this relation to derive lower bounds for the offline version of the problem in $\mathbb{R}^2$. We also define and analyze a generalized version of sparse matrix multiplication that is related to colored orthogonal range counting in higher dimensions.

5.1 Hardness of orthogonal colored range counting

Consider the following *Output Restricted Sparse Matrix Multiplication* problem ($ORSMM$). The input is a sparse matrix A with N non-zero entries (and therefore at most N rows and columns), and a set O of M pairs, (i, j), where i and j are indices of two rows of A. The goal is to compute, for each pair $(i, j) \in O$, the (i, j)-th entry of the product AA^T. We further assume here that A is Boolean, although AA^T is computed over the integrals (or reals).[7]

The offline version of the colored orthogonal range counting problem with n points and m queries is closely related to the $ORSMM$ problem, as follows from our solution to colored range counting in $\mathbb{R}^2$ in Section 4:

Theorem 5.1. *The 2-dimensional orthogonal colored range counting problem on n points and m query rectangles can be reduced to an $ORSMM$ problem, where the matrix A has $N = O(n \log^4 n)$ non-zero entries and we ask for $M = O(m \log^6 n)$ entries of the matrix AA^T. The reduction takes $O(n \log^4 n)$ time.*

The following theorem shows that a reverse reduction also exists.

Theorem 5.2. *The $ORSMM$ problem, for a Boolean matrix A with N non-zero entries, where we need to compute M output pairs, can be reduced in linear time to a 2-dimensional colored orthogonal range counting problem on $O(N)$ points and M query rectangles.*

Proof. We can restate the $ORSMM$ problem as follows. Let X be the set of columns in the matrix A. For each row i, let $S_i \subseteq X$ denote the the set of columns where row i has ones. Let O be the set of M output pairs to be computed. For each pair $(i, j) \in O$ we have to compute $|S_i \cap S_j|$, which is the (i, j)-th entry of AA^T. Since $|S_i \cap S_j| = |S_i| + |S_j| - |S_i \cup S_j|$, this is equivalent to computing, for each pair $(i, j) \in O$, the quantity $|S_i \cup S_j|$.

Let k denote the number of nonempty rows of A. We construct a colored range counting instance where the points are $p_1 = (1, 1)$, $p_2 = (2, 2)$, ..., $p_k = (k, k)$ and $p'_1 = (k + 1, 1)$, $p'_2 = (k + 2, 2)$, ..., $p'_k = (2k, k)$. We assign a distinct color to each column of A. We next replace each point p_i by $|S_i|$ points, which are close to each other within distance $\epsilon \ll 1$ of p_i. We color these points by the colors of the columns in S_i. We do the same for each of the points p'_i. Let P denote the resulting point set; we have $|P| = 2N$. Clearly, with an appropriate representation of the input, this construction takes $O(N)$ time.

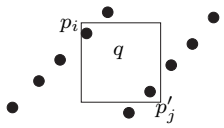

Figure 10: The rectangle q contains only p_i and p'_j.

Now, in order to calculate $|S_i \cup S_j|$, for $j \leq i$, we query P with the rectangle $q = [i - \epsilon, k + j + \epsilon] \times [j - \epsilon, i + \epsilon]$. Since q contains only the points p_i and p'_j, it is clear that the number of distinct colors in q is $|S_i \cup S_j|$. See Figure 10. □

We next describe some observations regarding the complexity of $ORSMM$ and the implications of Theorem 5.2. For any $t \times t$ Boolean matrix A, computing AA^T (over the reals) is a special case of $ORSMM$, with $N = M = t^2$ [11]. The best known algorithm for computing AA^T runs in $O(t^\omega) = O(N^{\omega/2})$ time, for $\omega \simeq 2.376$. So any algorithm for $ORSMM$ whose running time is

[7] We can instead ask for M entries in the product of two arbitrary Boolean matrices A and B with N nonzero items in both. Our results carry over to this generalized version.

faster than $O(\min(N,M)^{\omega/2})$ would immediately imply a better algorithm for Boolean matrix multiplication (over the reals), thereby solving a long-standing open problem. Using Theorem 5.2, we obtain that an algorithm for planar colored orthogonal range counting that can answer m box queries with respect to a set of $O(n)$ colored points in $o(\min(n,m)^{\omega/2})$ time would imply an algorithm that can compute AA^T in $o(t^\omega)$ time, for any $t \times t$ Boolean matrix A.

Similarly, consider the problem of computing AA^T, where A is a sparse rectangular Boolean matrix with t rows and N ones such that $N \geq t^{\frac{\omega+1}{2}}$. The best known algorithm for performing this computation runs in $O(Nt^{\frac{\omega-1}{2}})$ time (see Section 4). For any such matrix A, computing AA^T is also an instance of $ORSMM$, with N ones and $M = t^2$ pairs. So an algorithm for $ORSMM$ that runs in $o(NM^{\frac{\omega-1}{4}})$ time, for $N \geq M^{\frac{\omega+1}{4}}$, would imply a faster algorithm for sparse rectangular matrix multiplication than the best known to date. Using Theorem 5.2, we obtain that the existence of an algorithm for planar colored orthogonal range counting that can answer m box queries with respect to a set of n colored points in $o(nm^{\frac{\omega-1}{4}}) \approx o(nm^{0.344})$ time, for $n \geq m^{\frac{1+\omega}{4}} \approx m^{0.844}$, would also imply an algorithm that can compute AA^T for any rectangular Boolean matrix A with t rows and N ones, such that $N \geq t^{\frac{\omega+1}{2}}$, faster than what is known to date.

Generalized matrix multiplication and higher-dimensional colored range counting. For $d > 2$, we can show a similar relation between colored orthogonal range counting and a generalization of $ORSMM$. In this generalization, one is given a Boolean matrix A with N nonzero entries in sparse representation, and a list O of M d-tuples of indices of rows of A. Let t be the number of columns in A. The goal is to compute, for each tuple $(i_1, \ldots, i_d) \in O$, the sum $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$. We call this problem the *d-dimensional Output Restricted Sparse Matrix Multiplication* problem, and denote it by $ORSMM_d$. The following theorem generalizes Theorem 5.2 to dimension $d > 2$.

Theorem 5.3. *Any instance of the $ORSMM_d$ problem, of a Boolean matrix A with N nonzero entries and M output tuples, can be reduced, in linear time, to $O(1)$ instances of d'-dimensional colored orthogonal range counting, for $d' \leq d$, each on $O(N)$ points and M query boxes.*

Proof. Let X be the set of columns of A. The matrix multiplication problem is equivalent to the following problem. We are given a family of sets $\mathcal{F} = \{S_1, \ldots, S_k\}$, such that $S_i \subseteq X$, for each i, and $\sum_{i=1}^{k} |S_i| = N$, and M d-tuples $\{(i_1, \ldots, i_d)\}_{i=1}^{M}$; the goal is to compute, for each of the tuples, the corresponding quantity $|\bigcap_{j=1}^{d} S_{i_j}|$.

By using the inclusion-exclusion principle, one can easily verify that the above problem is equivalent to computing, for each output tuple $(i_1, \ldots, i_d)$, the quantities $|\bigcup_{j \in J} S_{i_j}|$, for every $J \subseteq \{1, \ldots, d\}$. Without loss of generality, we show how to do it for the case $J = \{1, \ldots, d\}$.

Let k denote the number of nonempty rows of A. Each column of A is assigned a distinct color. Let $\vec{1} \in \mathbb{R}^d$ be the vector with all components equal to 1. Let $\vec{1}_{<j} \in \mathbb{R}^d$ be the vector whose ℓ-th component is 1, for $\ell < j$, and 0 for $\ell \geq j$. Our colored range counting instance has dk "point clusters" $p_i^j = i\vec{1} + k\vec{1}_{<j}$, for $1 \leq i \leq k$ and $1 \leq j \leq d$. For each i and j, we replace p_i^j by a set P_i^j of $|S_i|$ distinct points placed at distance $< \epsilon \ll 1$ from p_i^j and given the colors of the columns in S_i. Let P be the resulting overall set of points.

Now, in order to compute $|\bigcup_{j=1}^{d} S_{i_j}|$, where $1 \leq i_d < i_{d-1} < \ldots < i_1 \leq k$, we query P with the box $R = [i_1 - \epsilon, i_2 + k + \epsilon] \times [i_2 - \epsilon, i_3 + k + \epsilon] \times \cdots \times [i_{d-2} - \epsilon, i_{d-1} + k + \epsilon] \times [i_{d-1} - \epsilon, i_d + k + \epsilon] \times [i_d - \epsilon, i_1 + \epsilon]$. We claim that $R \cap P = \bigcup_{j=1}^{d} P_{i_j}^j$. The number of colors in R is then $|\bigcup_{i=1}^{d} S_{i_j}|$. We have $|P| = dN$, and P can be constructed in $O(N)$ time, assuming an appropriate representation of A. We repeat this reduction for each subset $J \subseteq \{1, \ldots, d\}$, and obtain an instance of $|J|$-dimensional colored orthogonal range counting, with M query boxes. Hence, the overall resulting collection of $O(M)$ queries solve the given instance of $ORSMM_d$.

To prove the claim, consider a fixed point $p^j_{i_j}$. The first $j-1$ coordinates of this point are equal to $i_j + k$, and the rest are equal to i_j. It easily follows that this point is contained in R. On the other hand, let p^j_a be a point such that $a \neq i_j$. We argue that $p^j_a \notin R$.

Case 1a: If $a < i_j$ and $j \neq d$ then the projection of R on the j-th coordinate is the interval $[i_j, i_{j+1} + k]$, but the j-th coordinate of p^j_a is a, and since $a < i_j$, we get that $p^j_a \notin R$.

Case 1b: If $a < i_j$ and $j = d$ then the projection of R on the d^{th} coordinate is the interval $[i_d, i_1]$, but the d-th coordinate of p^j_a is a, and since $a < i_d$, we get that $p^j_a \notin R$.

Case 2a: If $a > i_j$ and $j \neq 1$ then the projection of R on the $(j-1)$-th coordinate is the interval $[i_{j-1}, i_j + k]$, but the $(j-1)$-th coordinate of p^j_a is $a + k$, and since $a > i_j$, we get that $p^j_a \notin R$.

Case 2b: If $a > i_j$ and $j = 1$ then the projection of R on the d-th coordinate is the interval $[i_d, i_1]$, but the d-th coordinate of p^j_a is a, and since $a > i_1$, we get that $p^j_a \notin R$.

This establishes the claim and thus completes the proof of the theorem. □

5.2 Efficient algorithms for $ORSMM_d$

We obtain an efficient algorithm for $ORSMM_d$ in three stages, where each stage uses the previous algorithm as a subroutine.

The case of non-sparse matrices with no output restriction. We start by considering the non-sparse version of the problem where A is any $t \times t$ matrix and we want to compute $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$ for any d-tuple of rows of A. With a slight possible abuse of notation, we refer to this problem as *d-dimensional matrix multiplication*.

Let $\omega < 2.376$ be the smallest constant such that (regular) matrix multiplication (of $t \times t$ matrices) takes $O(t^\omega)$ time. Let ω_d be a constant such that d-dimensional matrix multiplication takes $O(t^{\omega_d})$ time (in particular, $\omega_2 = \omega$). The output size of the d-dimensional matrix multiplication problem is $\Theta(t^d)$, so $\omega_d \geq d$. On the other hand we can compute each of the $O(t^d)$ sums $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$ in $O(t)$ time, so $\omega_d \leq d+1$. In fact it is not hard to obtain a better upper bound on ω_d:

Lemma 5.4. *$\omega_d \leq \omega + d - 2$. Furthermore, if $\omega > 2$ and $d > 2$ then $\omega_d < \omega + d - 2$.*

Proof. For $d = 2$ the lemma obviously holds, so we assume $d \geq 3$. Let A be the input matrix. First compute two matrices B and C, so that B (resp., C) is a $t^{\lceil d/2 \rceil} \times t$ matrix (resp., $t \times t^{\lfloor d/2 \rfloor}$ matrix) whose rows (resp., columns) contain the element-wise products of all sets of $\lceil d/2 \rceil$ (resp., $t^{\lfloor d/2 \rfloor}$) rows of A. Both B and C can be computed in time $O(\binom{t}{\lceil d/2 \rceil} \cdot t) = O(t^d)$. By construction, the (usual, two-dimensional) product BC provides a solution to the d-dimensional matrix multiplication for A. By partitioning BC into blocks of size $t \times t$, computing BC can be done in $O(t^{\omega+d-2})$ time. In fact, if $\omega > 2$ then the results of Huang and Pan [18] about rectangular matrix multiplication allows us to improve this, thus establishing the second part of the lemma. □

Lemma 5.5. *Let A be an $s \times t$ rectangular matrix. Then one can compute $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$, for all d-tuples of rows of A, in time*

$$\begin{cases} O(s^{\omega+d-3}t) & \text{if } s \leq t, \\ O(s^{d-\alpha\beta}t^\beta) & \text{if } s^\alpha < t < s, \\ O(s^d) & \text{if } t \leq s^\alpha. \end{cases}$$

Proof. As in the proof of of Lemma 5.4, we reduce the problem, in $O(\binom{s}{\lceil d/2 \rceil} \cdot t)$ time, to the multiplication of two rectangular matrices B and C of size $s^{\lceil d/2 \rceil} \times t$ and $t \times s^{\lfloor d/2 \rfloor}$, respectively. The multiplication can be done by partitioning B and C in blocks of size $s \times t$ and $t \times s$, respectively. The lemma follows using the upper bound

$$\begin{cases} O(s^{\omega-1}t) & \text{if } s \le t, \\ O(s^{2-\alpha\beta}t^{\beta}) & \text{if } s^{\alpha} < t < s, \\ O(s^2) & \text{if } t \le s^{\alpha} \end{cases}$$

on the complexity of multiplication of two rectangular matrices of size $s \times t$ and $t \times s$, respectively (Coppersmith [10], Huang and Pan [18]; see also [7, 20, 26]). □

The case of sparse matrices with no output restriction. We next consider the d-dimensional sparse matrix multiplication problem, where the Boolean input matrix has N ones and s rows, and the goal is to compute $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$, for all d-tuples of the rows. For $d = 2$ we already considered this problem in Section 4.

We solve this problem, for $d > 2$, using the same approach as in [20] (which is built upon the earlier technique of [26]; see also [7]). Let x be a parameter to be determined shortly. We consider separately columns with fewer than x non-zero entries (*light columns*), and columns with at least x non-zero entries (*heavy columns*). A light column with $z \le x$ nonzero entries contributes a nonzero term to z^d products. So the total time to handle the light columns is at most $O(\sum_i x_i^d)$, where x_i is the number of nonzero items in the i-th light column. This expression is maximized, and equals to $\frac{N}{x}x^d$, if we split the $\le N$ nonzero entries among $\le N/x$ light columns, placing x nonzero entries in each column. We have at most N/x heavy columns and we compute their contribution to the output by the previous algorithm for d-dimensional multiplication of the corresponding rectangular (non-sparse) matrix with s rows and N/x columns, as provided by Lemma 5.5. Optimizing over x, we obtain the following overall bound on the running time:

$$\begin{cases} O\left(Ns^{(\omega+d-3)\frac{d-1}{d}}\right) & \text{if } N \ge s^{1+(\omega+d-3)/d}, \\ O\left(N^{\frac{\beta-1}{d+\beta-1}(d-1)+1}s^{\frac{d-\alpha\beta}{d+\beta-1}(d-1)}\right) & \text{if } s^{1+\alpha-\frac{\alpha}{d}} \le N \le s^{1+(\omega+d-3)/d}, \\ O(s^d) & \text{if } N \le s^{1+\alpha-\frac{\alpha}{d}}. \end{cases}$$

The general case: An efficient algorithm for $ORSMM_d$. We fix another threshold parameter y, and consider separately the rows with at most y non-zero entries (the *light rows*), and the rows with at least y non-zero entries (the *heavy rows*). We can compute each product in which a light row takes part in $O^*(y)$ time, so all such products can be computed in $O^*(My)$ time. We are left with products that include only heavy rows. Since there are at most N/y heavy rows, we can compute all remaining products by solving a d-dimensional sparse sparse multiplication problem (with no output restriction) with $s = \frac{N}{y}$ rows and N nonzero entries, using the algorithm just described. Optimizing over y in order to minimize the total time complexity of the solution, we obtain the following theorem.

Theorem 5.6. *$ORSMM_d$ with N nonzero entries and M output tuples can be solved in time:*

$$\begin{cases} O^*\left(NM^{\frac{(d-1)(\omega+d-3)}{d+(d-1)(\omega+d-3)}}\right) & \text{if } N \ge M^{\frac{\omega+2d-3}{d+(d-1)(\omega+d-3)}}, \\ O^*\left(N^{\frac{(d-\alpha\beta+\beta-1)(d-1)+d+\beta-1}{(d-\alpha\beta)(d-1)+d+\beta-1}}M^{1-\frac{d+\beta-1}{(d-1)(d-\alpha\beta)+d+\beta-1}}\right) & \text{if } M^{\frac{1+\alpha-\alpha/d}{d-\alpha+\alpha/d}} \le N \le M^{\frac{d+\omega+d-3}{d+(d-1)(\omega+d-3)}}, \\ O^*\left(N^{\frac{d}{d+1}}M^{\frac{d}{d+1}}\right) & \text{if } N \le M^{\frac{1+\alpha-\alpha/d}{d-\alpha+\alpha/d}}. \end{cases}$$

In particular we get for $d = 2$ a solution for $ORSMM_2$ that takes $O^*\left(NM^{\frac{\omega-1}{\omega+1}}\right)$ time, for $M \leq N$. Note that this solution is inferior to the one given in Subsection 4.1 for the case $M = \binom{t}{2}$ (no output restriction): the previous algorithm takes $O\left(Nt^{\frac{\omega-1}{2}}\right)$ time, whereas the new one takes $O\left(Nt^{\frac{2(\omega-1)}{\omega+1}}\right)$ time, which is indeed much larger. It would be interesting to improve the new algorithm for all possible values of M.

References

[1] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, On the average number of maxima in a set of vectors and applications, *J. ACM* 25 (1978), 536–543.

[2] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer Verlag, Heidelberg, 2000.

[3] J.-D. Boissonnat, M. Sharir, B. Tagansky and M. Yvinec, Voronoi diagrams in higher dimensions under certain polyhedral distance functions, *Discrete Comput. Geom.* 19 (1998), 485–519.

[4] P. Bozanis, N. Kitsios, C. Makris, and A. Tsakalidis, New upper bounds for generalized intersection searching problems, *Proc. 22nd International Colloquium on Automata, Languages and Programming*, volume 944 of Lecture Notes in Computer Science, pages 464–475, Berlin,1995, Springer-Verlag.

[5] P. Bozanis, N. Kitsios, C. Makris, and A. Tsakalidis, New results on intersection query problems, *The Computer Journal* 40 (1997), 22–29.

[6] P. Bozanis, N. Kitsios, C. Makris, and A. Tsakalidis, Red-blue intersection reporting for objects of non-constant size, *The Computer Journal* 39 (1996), 541–546.

[7] T. Chan, Dynamic subgraph connectivity with geometric applications, *Proc. 34th ACM Sympos. Theory Comput.* (2002), 7–13.

[8] L. P. Chew, D. Dor, A. Efrat, and K. Kedem, Geometric pattern matching in d-dimensional space, *Discrete Comput. Geom.* 21 (1999), 257–274.

[9] K. L. Clarkson and P. W. Shor, Applications of random sampling in computational geometry II, *Discrete Comput. Geom.* 4 (1989), 387–421.

[10] D. Coppersmith, Rectangular matrix multiplication revisited, *J. Complexity* 13 (1997), 42–49.

[11] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *J. Symb. Comput.* 9 (1990), 251–280.

[12] D. Gupta, R. Janardan and M. Smid, Computational geometry: Generalized intersection searching, Chapter 64 in *Handbook of Data Structures and Applications*, D. Mehta and S. Sahni (editors), Chapman & Hall/CRC, Boca Raton, FL, 64-1–64-17, 2005.

[13] D. Gupta, R. Janardan and M. Smid, A technique for adding range restrictions to generalized searching problems, *Inform. Process. Lett.* 69 (1999), 7–13.

[14] D. Gupta, R. Janardan and M. Smid, Algorithms for some intersection searching problems involving circular objects, *Internat. J. Math. Algorithms* 1 (1999), 35–52.

[15] D. Gupta, R. Janardan and M. Smid, Algorithms for generalized halfspace range searching and other intersection searching problems, *Comput. Geom. Theory Appl.* 5 (1996), 321–340.

[16] D. Gupta, R. Janardan and M. Smid, Further results on generalized intersection problems: Counting, reporting, and dynamization, *J. of Algorithms* 19 (1995), 282–317.

[17] S. Har-Peled, On the expected complexity of random convex hulls, Technical Report 330/98, School Math. Sci., Tel-Aviv Univ., Tel-Aviv, Israel, 1998.

[18] X. Huang and V.Y. Pan, Fast rectangular matrix multiplication and applications, *J. Complexity* 14 (1998), 257–299.

[19] R. Janardan and M. Lopez, Generalized intersection searching problems, *Internat. J. Comput. Geom. Appl.* 3 (1993), 39–69.

[20] H. Kaplan, M. Sharir and E. Verbin, Colored intersection searching via sparse rectangular matrix multiplication, *Proc. 22th ACM Sympos. Comput. Geom.* (2006), 52–60.

[21] J. Matoušek and O. Schwarzkopf, On ray shooting in convex polytopes, *Discrete Comput. Geom.* 10 (1993), 215–232.

[22] J. Matoušek, Efficient partition trees, *Discrete Comput. Geom.* 8 (1992), 315–334.

[23] J. Matoušek, Cutting hyperplane arrangements, *Discrete Comput. Geom.* 6 (1991), 385–406.

[24] J. Matoušek, Range searching with efficient hierarchical cuttings, *Discrete Comput. Geom.* 10 (1993), 157–182.

[25] D. E. Willard and G. S. Lueker, Adding range restriction capability to dynamic data structures, *J. ACM* 32 (1985), 597–617.

[26] R. Yuster and U. Zwick, Fast sparse matrix multiplication, *ACM Trans. Algorithms* 1 (2005), 2–13.