Counting Colors in Boxes *

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

July 5, 2006

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 B, count the number of distinct colors of the points of $P \cap B$. 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. This reduction 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.

*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 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 $B \subset \mathbb{R}^d$, can efficiently count the number of *distinct colors* of points in $B \cap P$. We call this problem *colored orthogonal range counting*.

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 *attributes* of all the items in a query (e.g., how many different cities of birth do the persons in a query 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.

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. [6] for a comprehensive review of the state of the art). Halfspace colored range searching problems were studied in [8], and orthogonal colored range searching problems were studied in [7, 9]. 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*: Partitioning the query box into two sub-boxes and counting the number of colors in each sub-box tells us practically nothing about the number of colors in the full box.

In [7], Gupta et al. describe data structures for orthogonal colored range *reporting* in any dimension, which require storage larger by an $O(n^\epsilon)$ factor compared to those required for the non-colored versions. However orthogonal colored range *counting* turned out to be much harder. In [9], Gupta et al. describe solutions to colored range counting in dimensions $d = 1$ and $d = 2$. For $d = 1$, they do it by a reduction[2] to standard orthogonal range counting in $\mathbb{R}^2$. The resulting solution has query time $O(\log n)$ and storage and preprocessing cost $O(n \log n)$. Using persistence, they extend their one-dimensional data structure into a two-dimensional structure which supports queries that involve *3-sided boxes*; that is, boxes of the form, say, $[a, b] \times (-\infty, c]$. By creating a linear number of copies of this structure, they obtain a complete solution for $d = 2$, with query time $O(\log^2 n)$, and storage and preprocessing cost $O(n^2 \log^2 n)$.

We present a different approach, based on a reduction from colored orthogonal range counting 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 of all (points representing) query boxes containing a point with color c is the union of positive orthants, denoted as $Q(c)^+$.

Next, 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 naive implementation of this technique yields a solution for the colored orthogonal range counting problem in $\mathbb{R}^d$, having query time $O(\log^{2d-1} n)$, space 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 perfor-

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

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

mance parameters of the algorithms of Gupta et al. [6] 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 reduces all three performance parameters by a $\log n$ factor.

In fact, the storage size and preprocessing time of our algorithm depend on the 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). However, in practice the number can be expected to be much smaller. In particular, we show that, for random point sets (according to a natural model that we detail in Section 3), 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 expense of increasing the query time. In $\mathbb{R}^2$ we use a technique of Gupta et al. [6] 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-coordinate. (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 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 priciple 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 [11, 13] 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 $O\left(\left(\frac{n}{X}\right)^2 \log^8 n + n \log^4 n\right)$, for any tradeoff threshold parameter $1 \leq X \leq n$. The construction time of the data structure is (the $O^*(\cdot)$ notation hides polylogarithmic factors):

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

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), and $\alpha > 0.294$ is another parameter related to matrix multiplication (see Section 4) [4, 5]. In particular it follows from these bounds that for $m \leq n$, we can answer m queries in overall time $O^*(nm^{\frac{\omega-1}{\omega+1}}) = O(nm^{0.408})$.

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 in $o(nm^{\frac{\omega-1}{4}}) = o(n^{1.34})$ time, for $m^{\frac{1+\omega}{4}} \leq n$, then we can obtain a better algorithm for sparse rectangular matrix multiplication than the best known todate. 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.

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 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 Boolean matrix A with N nonzero entries in a 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 fo 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* ($ORSMM_d$), and 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 give a reduction from the offline version of colored range counting in *complements* of boxes to this matrix multiplication problem. In an Appendix we describe efficient algorithms for $ORSMM_d$.

2 Reducing Colored Range Counting 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 general colored axis-parallel box counting problem 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 $p = (p_1, \ldots, p_n)$, 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 $A \subset \mathbb{R}^d$ of size n, 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. [2] concerning the complexity of the union of n congruent axis-parallel cubes in $\mathbb{R}^d$ (think of the orthants as sufficiently 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 [2].

We use Theorem 2.1 to solve the semi-unbounded colored range counting problem as follows. For each color $1 \le c \le C$, 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[3] $q \in \mathbb{R}^d$. 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. This structure has query time $O(\log^{d-1} n)$, and space and preprocessing time $O(n^{\lfloor d/2 \rfloor} \log^{d-1} n)$. 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

[3]The problem is called the “stabbing query problem” in [1].

structure, thereby saving a logarithmic factor in both storage and preprocessing time. We thus obtain the following theorem.

Theorem 2.2. *There exists a data structure for colored semi-unbounded (orthant) range counting queries 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 $p = (x_1(p), \ldots, x_i(p), \ldots, x_d(p)) \in P$, obtaining 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 theorem.

Theorem 2.3. *There exists a data structure for colored orthogonal range counting queries 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.*

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, where, for specificity, we assume that these latter projections are all unbounded in the negative respective coordinate directions. 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. [6], 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 [6] 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 an enhanced algorithm, in which a query takes $O(\log^2 n)$ time, and the storage and the preprocessing cost are both $O(n^2 \log^2 n)$. The same enhancement can be applied in any dimension, leading to the following result.

Theorem 2.4. *There exists a data structure for colored orthogonal range counting queries 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 ≥ 4. In particular, for d even ≥ 4 and $k = 0$, we can improve all three performance parameters in Theorem 2.2 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). We also remark that for certain enhancements of our technique, the improvement in Theorem 2.4 is not applicable, and one has to use Theorem 2.3 instead.

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, for any $i = 1, \ldots, d$.[4] In this section 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 $1 \le i \le d$. Thus $O = \prod_{i=1}^{d}(-\infty, x_i(s_i)]$ (see Figure 1(left)). 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 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 $\langle s_1, \ldots, s_d \rangle$.

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 1(right) for an illustration.

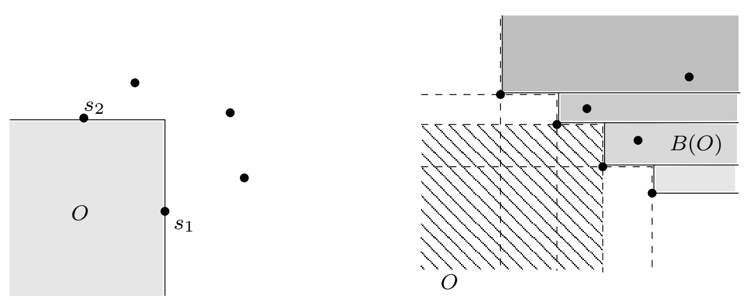

Figure 1: A maximal empty orthant (left), and the box decomposition of $U(A)$ (right).

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 $U(A)$.*

[4] We can remove the general position assumption by imposing a strict order on the x_i-coordinates of all points for $1 \le i \le d$, 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.

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 $A = \{p_1, \ldots, p_n\} \subset \mathbb{R}$. Let us assume that $p_1 < p_2 < \cdots < p_n$. 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 $1 \le j \le n$, 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 $x_d \ge x_d(p_n)$.

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 $q_{j+1} \le x_d(s_d)$ (or $x_d(s_d) = \infty$ for $j = n$), $\max_{j<d}\{x_d(s_j)\} \le q_j$, and $x_1(s_1) < \infty$. See Figure 2(left) 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 $H_j \cap U(A)$.

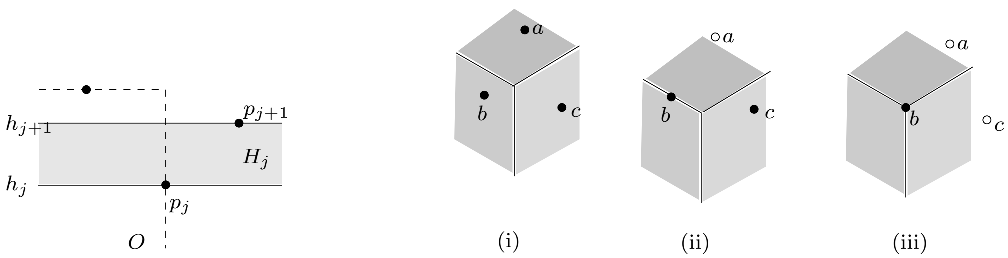

Figure 2: The proof of Lemma 3.1 (left), and three legal shifts that turn a maximal empty orthant into the one that has point b as an apex (right).

Indeed, let $D_j = \{p_i \mid i \le j\}$, 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 2(left) 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 contains $U(A) \cap H_j$. Repeating this argument for each slab H_j completes the proof. $\square$

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. [2]. 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 2(right). Clearly, the process always terminates. At that time no point of A lies in the relative interior of any facet of O'. Hence each boundary point of O' determines at least two of the coordinate values that define facets of O', and, since we assume general position, 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.

Constructing the boxes. 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 contructive 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 $U(A)$. Instead, we present in Appendix A an *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 $U(A)$.

Random sets of points. The decomposition-based data structure of Theorem 2.3 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. Consider a point set P 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, and these permutations are indepndent of each other. In Appendix B we prove that the expected number of maximal empty boxes (i.e., bounded or semi-unbounded boxes containing no point of P in their interior which are maximal with respect to this property) for such a point set is only $O(n \log^{d-1} n)$. This leads to the following two consequences.

Theorem 3.2. *Let P be a random colored point-set of cardinality n, such that the points of each color are selected as above. 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.*

Theorem 3.3. *Let P be a random colored point-set of cardinality n, such that the points of each color are selected as above. Then there exists a data structure supporting semi-unbounded colored range counting queries in $O(\log^{3d-1} n)$ time,* expected *storage requirement of $O(n \log^{3d-1} n)$ and which can be constructed in expected $O(n \log^{3d-1} n)$ time.*

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

We present an efficient solution for colored orthogonal range counting in $\mathbb{R}^2$. Our method uses the more space-efficient data structure 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. Since the data structure has three levels, the number of such sets in a query output is $O(\log^3 n)$

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 [6], 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 and $[a, b] \times (-\infty, d]$. 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. Clearly, this data structure requires $O(n \log^3 n)$ storage, and can be constructed in $O(n \log^3 n)$ time.

For each node t in one of the auxiliary structures (i.e., a node 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 $[c, d]$. Either the search runs off T, or it reaches a (highest) node, denoted by v_q, such that $[c, d]$ contains $Y(v_q)$. In the former case, we stop. In the latter case, if v_q is a leaf, we check whether the single point that it stores lies in q, and return its color if this is the case.

If v_q is an internal node, we query the two structures at v 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 [6] for more details). Hence, using the exclusion and inclusion principle, the number of colors that appear in q is equal to

$$\sum_{s \in U_q} |c(s)| + \sum_{t \in D_q} |c(t)| - \sum_{s \in U_q,\, t \in D_q} |c(s) \cap c(t)|. \tag{1}$$

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 be done in $O(\log^6 n)$, using (1).

Instead, we derive a tradeoff between storage and query time, determined by a threshold parameter X in the range $1 \le X \le n$. 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., $[a,b] \times (-\infty,d]$). A node $t \in \mathcal{D} \cup \mathcal{U}$ is called X*-heavy* if $|c(t)| > X$; otherwise it is called X*-light*. We construct and store, as part of the preprocessing stage, a matrix M, whose rows and columns correspond to the X-heavy nodes in $\mathcal{D}$ and $\mathcal{U}$, respectively. For each pair of X-heavy canonical nodes $s \in \mathcal{U}$ and $t \in \mathcal{D}$, we store in $M_{s,t}$ the value of $|c(t) \cap c(s)|$. The overall size of all the canonical subsets associated with the nodes of $\mathcal{U} \cup \mathcal{D}$ is $O(n \log^4 n)$. Hence, the number of X-heavy canonical nodes in $\mathcal{D} \cup \mathcal{U}$ is $O\left(\frac{n}{X} \log^4 n\right)$, so M has size $O\left(\left(\frac{n}{X}\right)^2 \log^8 n\right)$. (The time needed to construct M is discussed later.)

For each canonical node $t \in \mathcal{D} \cup \mathcal{U}$, 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 $\sum_{t \in \mathcal{D} \cup \mathcal{U}} |c(t)| = O(n \log^4 n)$, these additional structures take $O(n \log^4 n)$ extra storage.

The total space complexity of our solution is thus $O\left(\left(\frac{n}{X}\right)^2 \log^8 n + n \log^4 n\right)$.

Remark: Let $\mathcal{D}(v)$ (resp., $\mathcal{U}(v)$) be the set of canonical nodes in the auxiliary structures of node $v \in T$ used for answering queries of type $[a,b] \times [c,\infty)$ (resp., $[a,b] \times (-\infty,d]$). We can reduce the space to $O\left(\left(\frac{n}{X}\right)^2 \log^6 n + n \log^4 n\right)$ by keeping a matrix $M(v)$ for each $v \in T$ 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 $M(v)_{s,t}$ the value of $|c(t) \cap c(s)|$. The calculation of the preprocessing time however becomes more complicated.

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 and we simply retrieve it. Otherwise, we can assume, without loss of generality, that $|c(s)| \le X$. 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 disctionary 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 M.

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 0 otherwise. It follows that $M = M_D {M_U}^T$.

Each of the matrices M_D and M_U has $r = O\left(\frac{n}{x} \log^4 n\right)$ 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. [11] (which extends a technique of Yuster and Zwick [13]). Specifically, for two matrices A, B, each having t rows and $\le N$ non-zero items, these methods construct AB^T in time

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

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., [11]) 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:

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

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:

- For $m \leq n$, m queries can be answered in overall time $O^*(nm^{\frac{\omega-1}{\omega+1}}) = O(nm^{0.408})$.
- For $n \leq m \leq n^{1.616}$, m queries can be answered in overall time $O(n^{0.862}m^{0.546})$.
- For $m \geq n^{1.616}$, m 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 $O(n^{1.408})$, including time spent at the preprocessing stage.

4.2 Tradeoff in higher dimensions

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 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 [1], 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 2.3 for answering the query. The query time is $O(\log^{2d-1} n_i)$, and the space and preprocessing complexity are both

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

(use 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.

Two additional methods are presented in Appendix C. 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.

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 a generalized version of sparse matrix multiplication that is related to colored orthogonal range counting in higher dimesions.

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 is an index of a row of A and j is an index of a column 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.[5]

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 $N = n\log^4 n$ non-zero entries and we ask for $M = m\log^6 n$ entries of the output. The reduction is takes $O(n\log^4 n)$ time.*

The following theorem shows that a reverse reduction also exists.

Theorem 5.2. *The $ORSMM$ problem, where the matrix A has N one-entries, and 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. Let $S_i \subseteq X$ consist of the 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, since $|S_i \cap S_j| = |S_i| + |S_j| - |S_i \cup S_j|$, is equivalent to computing for each pair $(i, j) \in O$ the quantity $|S_i \cup S_j|$.

We construct a colored range counting instance where the points are $p_1 = (1, 1)$, $p_2 = (2, 2)$, $\ldots$, $p_k = (k, k)$ and $p'_1 = (2k + 1, 1)$, $p'_2 = (2k + 2, 2)$, $\ldots$, $p'_k = (3k, k)$. We have a color for each column of A. The point p_i in fact represents $|S_i|$ points. Each of these $|S_i|$ points is colored with a color that corresponds to a different item in S_i. The points are close to each other so we model them by one point with $|S_i|$ colors. Similarly, p'_i also represents $|S_i|$ points, close to each other, each with a color that corresponds to a different item in S_i.

Now, suppose we have a query to calculate $|S_i \cup S_j|$ for $j \le i$. It can be easily seen that the only points contained in the rectangle $[i, j + 2k] \times [j, i]$ are p_i and p'_j. So we can compute the cardinality of the union $|S_i \cup S_j|$ for each pair (i, j) by a colored range counting query. □

We next describe some observations regarding the complexity of *ORSMM* and the implications of Theorem 5.2. For any $t \times t$ matrix A, computing AA^T is a special case of *ORSMM* where

[5] We can instead ask for M entries in the product of two boolean matrices A and B with N nonzeros in both. Our results carry over to this version.

$N = M = t^2$. The best algorithm known today for computing AA^T runs in $O(t^\omega \approx N^{\omega/2})$ time. So any algorithm for $ORSMM$ whose running time is faster than $O(\min(N, M)^{\omega/2})$ would immediately imply a better algorithm for boolean matrix multiplication, thereby solving a long-standing open question. 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 $t \times t$ matrix A.

Similary, consider the problem of computing AA^T where A is a sparse rectangular matrix with t rows and N ones such that $N \geq t^{\frac{\omega+1}{2}}$. The best algorithm to perform this computation today 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 $N \geq M^{\frac{1+\omega}{4}}$, 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 $o(nm^{\frac{\omega-1}{4}})$ time, for $n \geq m^{\frac{1+\omega}{4}}$, implies an algorithm that can compute AA^T for any rectangular matrix A with t rows and N ones such that $N \geq t^{\frac{\omega+1}{2}}$, faster than what is known to date.

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 a 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, whose proof is given in Appendix D, generalizes Theorem 5.2 to dimension $d > 2$.

Theorem 5.3. *The d-dimensional colored orthogonal range counting problem on $O(n)$ points and m query rectangles is not easier than a d-dimensional output restricted sparse matrix multiplication problem with $N = O(n)$ nonzero entries in the matrix A and a set O of $M = m$ tuples.*

In Appendix D we also present an efficient solution of the $ORSMM_d$ problem.

References

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

[2] 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.

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

[4] D. Coppersmith. Rectangular matrix multiplication revisited. *J. Complexity*, 13(1):42–49, 1997.

[5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[6] 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, 2005.

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

[8] D. Gupta, R. Janardan and M. Smid, Algorithms for generalized halfspace range searching and other intersection searching problems, *Computational Geometry: Theory and Applications*, 5 (1996), 321–340.

[9] D. Gupta, R. Janardan and M. Smid, Further results on generalized intersection problems: Counting, Reporting, and Dynamization, *Journal of algorithms*, 19 (1995), 282–317.

[10] X. Huang and V.Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998.

[11] H. Kaplan, M. Sharir and E. Verbin, Colored intersection searching via sparse rectangular matrix multiplication, *Proc. 22th ACM Symp. on Computational Geometry* (2006), 52–60.

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

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

Appendix

A Output sensitive construction of the decomposition of $U(A)$

In this Section we show an *output sensitive* algorithm for constructing all maximal empty orthants with respect to a set of points 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. Let $D_j = \{p_i \mid i \leq j\}$, and let D'_j be the projection of D_j onto π.

After π passes through p_j, we maintain 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 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 interior). See Figure 3.

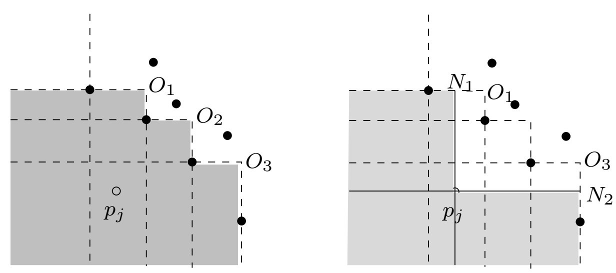

Figure 3: (a) The set Q of $(d-1)$-orthants that are eliminated by p_j. (b) The set N of $(d-1)$-orthants that are generated by p_j. In the example, $|Q| = 3$ and $|N| = 2$. N_1 is obtained by shrinking O_1 to the left, and N_2 is obtained by shrinking O_3 downwards.

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). Let p'_j be the projection of p_j on π. We 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 [12] 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 extending 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 trees is bounded by the number B of maximal empty orthants with respect to A. Indeed, this is clear for orthants in Q: each such orthant O corresponds to a unique d-orthant that is "completed" when O gets removed from Q. Each orthant O in N will either be completed into a 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 $O(B \log^{d-1} n)$, where B is the number of decomposition boxes. (In the worst case, $B = O(n^{\lfloor d/2 \rfloor})$.) This implies the second part of Theorem 2.1, and thus completes its proof. $\square$

B Random sets of points

The decomposition-based data structure of Theorem 2.3 is especially efficient, when the number of maximal empty orthants with respect to each of the sets $P_c \subset \mathbb{R}^d$, for $1 \le c \le C$, are all small. In this subsection we prove that the expected number of maximal empty boxes for a random point-set with n points is only $O(n \log^{d-1} n)$.

We assume that the point 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, and these permutations are indepndent of each other. Let $N_{a,k}(P)$ denote the number of axis-parallel boxes B that are *defined* by a points of P (these points lie on ∂B, and B is the smallest box that contains all these points[6]) and contain k other points in their interior. We refer to boxes defined by a points as *a-boxes*, and note that $2 \le a \le 2d$. We also denote by $N_{a,\le k}(P)$ the number of boxes as above, with at most k points of P in their interior. Finally, we let $N^{(d)}_{a,k}(n)$ (resp., $N^{(d)}_{a,\le k}(n)$) denote the expected value of $N_{a,k}(P)$ (resp., $N_{a,\le k}(P)$), over the random choice of the set P.

We clearly have $N^{(1)}_{2,0}(n) = n-1$. For $d = 2$, sample a set $P \subset \mathbb{R}^2$ as above, and project it onto the x_1-axis. The resulting set P^* is a random set of points on the line, according to the model that we assume. An empty 2-box B of P projects to an interval $B^* \subset \mathbb{R}$ whose endpoints are the projections of the two defining points p, q. However, B^* may contain any number k of other projected points (that lie above or below B in the plane). Let P_B denote the subset of P consisting of these points, including p and q. Since the x_2-coordinates are chosen independently of the choice of the x_1-coordinates, the x_2-order of the points of P_B is a random permutation of this set. The (conditional) probability that B be empty is the same as the conditional probability that p and q are consecutive in this permutation, which is $\dfrac{2!(k+1)!}{(k+2)!} = \dfrac{2}{k+1}$. See Figure 4. This implies the following recurrence:

$$N^{(2)}_{2,0}(n) = \sum_{k=0}^{n-2} \frac{2}{k+1} N^{(1)}_{2,k}(n).$$

Rearranging this sum, we obtain

$$N^{(2)}_{2,0}(n) = \sum_{k=0}^{n-3} \frac{2}{(k+1)(k+2)} N^{(1)}_{2,\le k}(n) + \frac{2}{n-1} N^{(1)}_{2,\le n-2}(n).$$

Since we have, trivially, $N^{(1)}_{2,\le k}(n) = O(nk)$, we obtain $N^{(2)}_{2,0}(n) = O(n \log n)$.

This analysis can be extended to any dimension, and to any number $2 \le a \le 2d$ of defining points. To estimate $N^{(d)}_{a,\le k}(n)$, we apply the Clarkson-Shor probabilistic technique [3] to conclude that $N^{(d)}_{a,\le k}(n) = O(k^a N^{(d)}_{a,0}(n/k))$. This in turn leads to a recurrence that yields $N^{(d)}_{a,0}(n) =$

[6] This definition allows only bounded boxes, but we can extend it to include semi-unbounded maximal empty boxes too.

$O(n \log^{d-1} n)$. (The Clarkson-Shor bound uses a random sample of n/k elements of P. By our drawing model, such a random sample is a random set of n/k points chosen by the same random drawing rule.) We thus have shown:

Theorem B.1. *Let P be a random point-set of cardinality n in $\mathbb{R}^d$, selected as above. 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.*

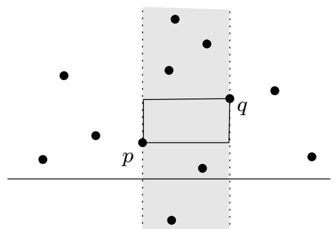

Figure 4: Bounding the expected number of empty boxes in the plane.

Specializing to the case of maximal empty orthants,[7] and using our output sensitive construction of the decomposition of the sets $U(P_c)$, we obtain the following theorem.

Theorem B.2. *Let P be a random colored point-set of cardinality n, such that the points of each color are selected as above. 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.*

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. This reduction transforms the points by doubling each coordinate. After doing so we can no longer assert 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 transformed. It can be easily verified 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 B.1 in order to upper-bound the number of maximal empty orthants with respect to $\hat{A}$ and obtain the following theorem.

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

Remark: The enhancement that shaves off a logarithmic factor, as provided in Theorem 2.4, 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.

[7]We have used empty boxes instead of empty orthants because the transformation from box queries to orthant queries does not preserve randomness—see below.

C 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 (1). 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 $\mathcal{U}$, $\mathcal{D}$ is now $O(n^{d-1}/X^{d-2} \log^{2d-1} n)$, the size of the matrix M is $O((n^{d-1}/X^{d-2}/X)^2) \log^{4d-2} n) = O((n/X)^{2d-2} \log^{4d-2} n)$, and the total storage is thus

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

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

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

Grid. Using this method, we achieve additional improvement of 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 $1 \le i \le d$. We fix a parameter t, and partition $\mathbb{R}^d$, for each $1 \le i \le d$, into t slabs, each containing n/t points of P and bounded by two hyperplanes orthogonal to the x_i-axis. These partitions 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 $O(t^{2d-2} n \log^{d+1-1/d} n + t^{2d} \log^{2d-1} n)$.

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 $n' \le 2dn/t$ points of P that lie in the "fringe" $Q \setminus Q_0$ of Q, using a standard orthogonal range *reporting* data structure [1]. See Figure 5. 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 $c = 1, \dots, C$, for d-dimensional

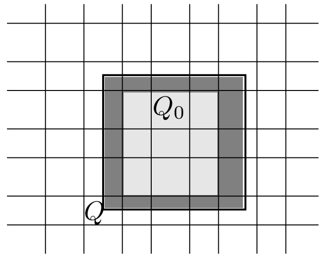

Figure 5: The grid construction in the plane, a query rectangle Q, its "core" Q_0, and its fringe.

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. 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: A query takes $O(X \log^{d-1} n)$ time, the storage is $O((\frac{n}{X})^{2d} + n \log^{d-1} n)$, and the preprocessing cost is $O\left(\left(\frac{n}{X}\right)^{2d-2} n \log^{d+1-1/d} n + \left(\frac{n}{X}\right)^{2d} \log^{2d-1} n\right)$.

D Output-restricted sparse matrix multiplication

We recall the definition of the d-dimensional output restricted sparse matrix multiplication problem ($OSRMM_d$). You are given a matrix A with N nonzero entries in a sparse representation. Also part of the input is 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 fo each tuple $(i_1, \ldots, i_d) \in O$ the sum $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$. We assume that A is a zero-one matrix. We prove the following theorem from Section 5.

Theorem D.1. *The d-dimensional colored orthogonal range counting problem on n points and m query rectangles is not easier than a d-dimensional output restricted sparse matrix multiplication problem with n nonzero entries in the matrix A and m entries of A^d to compute.*

Proof. Let X be the set of columns of A. The matrix multiplication problem is equivalent to the following problem. You are given a family of sets $\mathcal{F} = \{S_1, \ldots, S_k\}$, such that $S_i \subseteq X$, and $\sum_{i=1}^{k} |S_i| = N$. You also get M vectors $\{(i_1, \ldots, i_d)\}_{i=1}^{M}$. You wish to compute, for each i, the 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 i the quantity $|\bigcup_{j \in J} S_{i_j}|$ for every $J \subset \{1, \ldots, d\}$. Without any loss of generality, we show how to do it for the case $J = \{1, \ldots, d\}$.

Let $\vec{1} \in \mathbb{R}^d$ be the vector which has 1 in every coordinate, and let $\vec{1}_{<j} \in \mathbb{R}^d$ be the vector which has 1 in locations 1 through $j-1$, and 0 after this. Our colored range counting instance has dk points, each of $t = |X|$ possible colors (again, we allow to give a point multiple colors, just for convenience, to get a regular instance we can split each such point to many points one for each color).

These points are $p_i^j = i\vec{1} + 2k\vec{1}_{<j}$ for $1 \le i \le k$, and $1 \le j \le d$. For any j, the point p_i^j has $|S_i|$ colors corresponding to the elements of the set S_i. Now, suppose we need to compute $|\bigcup_{j=1}^{d} S_{i_j}|$ where $1 \le i_d < i_{d-1} < \ldots < i_1 \le k$. We claim that there exists a rectangle that contains just the points $\{p_{i_j}^j : 1 \le j \le d\}$. This will prove the theorem. This rectangle is $R = [i_1, i_2 + 2k] \times [i_2, i_3 + 2k] \times \ldots \times [i_{d-2}, i_{d-1} + 2k] \times [i_{d-1}, i_d + 2k] \times [i_d, i_1]$.

Consider $p^j_{i_j}$. The first $j-1$ coordinates of this point are equal to i_j+2k, and the rest are equal to i_j. So it is easy to verify 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 wish to show 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 $[i_j, i_{j+1}+2k]$, but the j^{th} coordinate of p^j_a is a, and since $a < i_j$ we get $p^j_a \notin R$.

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^d_a is a, and since $a < i_d$ we get $p^d_a \notin R$.

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

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^1_a is a, and since $a > i_1$ we get $p^1_a \notin R$.

□

D.1 Algorithms for $ORSMM_d$

We start by considering the non-sparse version of the problem where A is a $t \times t$ matrix and we would like to compute $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$ for every d rows of A. We call this problem the d-dimensional matrix multiplication, and we call the result the d-dimensional matrix power of A.

Let ω be a constant such that (regular) matrix multiplication takes $O(t^{\omega})$ (Recall that the best known bound on ω is 2.376.) Let ω_d be a constant such that d-dimensional matrix muliplication takes $O(t^{\omega_d})$ time (in particualr $\omega_2 = \omega$). 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 sum $\sum_{j=1}^{t} \prod_{k=1}^{d} A_{i_k,j}$ 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 D.2. *$\omega_d \leq \omega + d - 2$. 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. The matrix B is a $t^{\lceil d/2 \rceil} \times t$ matrix such that its rows contain the element-wise products of all sets of $\lceil d/2 \rceil$ rows of A. The matrix C is a $t \times t^{\lfloor d/2 \rfloor}$ matrix such that its columns contain the element-wise products of all sets of $\lfloor d/2 \rfloor$ rows of A. Both B and C can be computed in time $O(t^d)$. It is easy to see that by re-ordering the (usual, two-dimensional) product BC we get a solution to the d-dimensional matrix power of A. How long does it take to compute BC? By partitioning it to blocks of size $t \times t$ we can see that it takes at most $t^{\omega+d-2}$. In fact, if $\omega > 2$ then using the results of Huang and Pan [10] about rectangular matrix multiplication allows us to improve this, thus getting the second part of the lemma. □

This result is interesting because it indicates that we can probably obtain a bound better than the one in this lemma for $d > 2$ assuming that $\omega > 2$. If A is rectangular $s \times t$ matrix where $s \leq t$ we can obtain the d-dimensional matrix multiplication of A by breaking A into t/s square matrices. This would take $s^{(\omega_d - 1)}t$ time.

Our algorithm for $ORSMM_d$ uses as a subroutine an algorithm for a d-dimensional sparse matrix multiplication problem in which we bound the number of ones in A by N (as in $ORSMM_d$), and we bound the number of rows of A by s. 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 using the same approach as in [11] (which is built upon [13]). Let x be a parameter to be determined. We look separately at columns with less 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 to z^d products. So the total time to handle the light columns is at most $\sum x_i^d$ where x_i is the number of nonzeros in the ith light column. This expression is maximized, and equals to $\frac{n}{x}x^d$, if we split the $\le N$ nonzeros evenly between N/x light columns. We have at most N/x heavy columns and we compute their contribution to the result by d-dimensional rectangular multiplication of the matrix with s rows and N/x columns. Assuming $\frac{N}{x} \ge s$ this takes $O(s^{\omega_d - 1}N/x)$ time. The total cost to compute the d-dimensional power is $O(\frac{N}{x}x^d + s^{\omega_d-1}N/x)$. To minimize this expression we choose $x = s^{(\omega_d-1)/d}$. So we obtain that if $N \ge s^{1+(\omega_d-1)/d}$ the d-dimensional product can be computed in $O(Ns^{\frac{d-1}{d}(\omega_d-1)})$ time.

Next we give a solution to $ORSMM_d$. We look separately at 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. For all such products this takes $O(My)$ 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 matrix sparse multiplication problem with $s = \frac{N}{y}$ rows and N nonzeros. By our previous argument this takes $O(N(N/y)^{\frac{d-1}{d}(\omega_d-1)})$ if $N \ge (N/y)^{1+(\omega_d-1)/d}$. So the total time for all products is $O(My + N(N/y)^{\frac{d-1}{d}(\omega_d-1)})$ which is minimized for $y = NM^{\frac{-d}{d+(d-1)(\omega_d-1)}}$. We get the following theorem

Theorem D.3. *If $N \ge M^{\frac{d+\omega_d-1}{d+(d-1)(\omega_d-1)}}$ then we can solve $ORSMM_d$ in $O(NM^{\frac{(d-1)(\omega_d-1)}{d+(d-1)(\omega_d-1)}})$ time.*

In particular we get for $d = 2$ a solution for $ORSMM_2$ that takes $O(NM^{\frac{\omega-1}{\omega+1}})$ time. It may be interesting to look at the following instance of $ORSMM_2$. Take a $t \times t$ matrix A where each row and each column have exactly $t^{(\omega-1)/2}$ nonzeros so $N = t^{(\omega+1)/2}$. Suppose that we want to compute $M = t^{(\omega+1)/2}$ entries from $A \cdot A^T$, $t^{(\omega-1)/2}$ entries from each row and $t^{(\omega-1)/2}$ entries from each column. Theorem D.3 shows that we can solve this instance in $O(t^\omega)$ time. This time bound is the same as the time bound we obtain by solving it directly by each of the following three simple methods.

1. Perform the multiplication in time $O(t^\omega)$.
2. For each element of the output that you need, compute it brute-force. This takes $t^{(\omega-1)/2}$ for each element of the output we need, for a total of $O(t^\omega)$.
3. For each column $a \in A$, compute aa^T in a naive way, and add up all these matrices. This takes $t^{\omega-1}$ for each column, so a total of $O(t^\omega)$