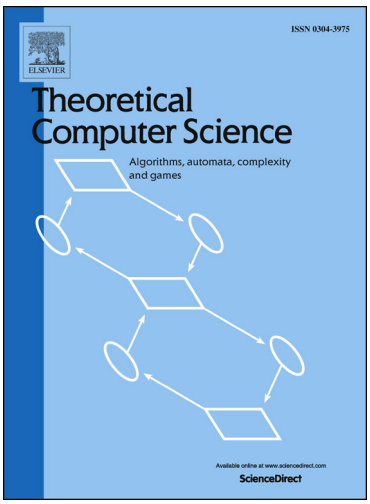
Journal Pre-proof



Computing the depth distribution of a set of boxes

Jérémy Barbay, Pablo Pérez-Lantero and Javiel Rojas-Ledesma

PII: S0304-3975(21)00350-9
DOI: https://doi.org/10.1016/j.tcs.2021.06.007
Reference: TCS 12974

To appear in: *Theoretical Computer Science*

Received date: 2 October 2019
Revised date: 30 December 2020
Accepted date: 1 June 2021

Please cite this article as: J. Barbay, P. Pérez-Lantero and J. Rojas-Ledesma, Computing the depth distribution of a set of boxes, *Theoretical Computer Science*, doi: https://doi.org/10.1016/j.tcs.2021.06.007.

Computing the Depth Distribution of a Set of Boxes

Jérémy Barbay[a,1], Pablo Pérez-Lantero[b,1], Javiel Rojas-Ledesma[a,1]

[a] *Departamento de Ciencias de la Computación, Universidad de Chile, Chile.*
[b] *Departamento de Matemática y Ciencia de la Computación, Universidad de Santiago, Chile.*

Abstract

Motivated by the analysis of range queries in databases, we introduce the computation of the depth distribution of a set $\mathcal{B}$ of n d-dimensional boxes (i.e., axis aligned d-dimensional hyperrectangles), which generalizes the computation of the Klee's measure and maximum depth of $\mathcal{B}$. We present an algorithm to compute the depth distribution running in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$, and refine these upper bound for various measures of difficulty of the input instances. Moreover, we introduce conditional lower bounds for this problem which not only provide insights on how fast the depth distribution can be computed, but also clarify the relation between the Depth Distribution problem and other fundamental problems in computer science.

Key words: Depth Distribution, Klee's Measure, Matrix Multiplication, Maximum Depth, Sets of Boxes, High dimensional data

1. Introduction

Problems studied in computational geometry have found important applications in the processing and querying of massive databases [1], such as the computation of the Maxima of a set of points [2, 4], or compressed data structures for Point Location and Rectangle Stabbing [3]. In particular, we

Email addresses: jeremy@barbay.cl (Jérémy Barbay), pablo.perez.l@usach.cl (Pablo Pérez-Lantero), jrojas@dcc.uchile.cl (Javiel Rojas-Ledesma)

[1] This work was supported by projects CONICYT Fondecyt/Regular nos 1170366 and 1160543, and CONICYT-PCHA/Doctorado Nacional/2013-63130209 (Chile). Pablo Pérez-Lantero was partially supported by projects DICYT 041933PL Vicerrectoría de Investigación, Desarrollo e Innovación USACH (Chile), and Programa Regional STICAMSUD 19-STIC-02.

consider cases where the input or queries are composed of d-dimensional boxes (i.e., axis aligned hyperrectangles in $\mathbb{R}^d$): in the context of databases it corresponds for instance to a search for objects for sale within the intersection of ranges in price, availability and quality ratings.

Consider a set $\mathcal{B}$ of n d-dimensional boxes, for fixed (constant) d. We focus on two measures on such a set of boxes: the Klee's measure and the maximum depth[1]. The Klee's measure of $\mathcal{B}$ is the size of the "shadow" projected by $\mathcal{B}$, more formally, the volume of the union of the boxes in $\mathcal{B}$. Originally suggested on the line by Klee [21], its computation is well studied in higher dimensions [7, 8, 9, 10, 27], and can be done in time within $\mathcal{O}(n^{d/2})$, using an algorithm introduced by Chan [10] based on a new paradigm called "Simplify, Divide and Conquer". The maximum depth of $\mathcal{B}$ is the maximum number of boxes covering a same point, and its computational complexity is similar to that of Klee's measure's, converging to the same complexity within $\mathcal{O}(n^{d/2})$ [10].

Hypothesis. The known algorithms to compute these two measures are all strikingly similar, to the point that Chan [10] states that all known techniques used for computing the Klee's measure can be applied to the computation of the maximum depth. That would suggest a reduction from one to the other, but those two measures are completely distinct: the Klee's measure is a volume whose value might be a real number, while the maximum depth is a cardinality whose value is an integer in the range $[1..n]$. **Is there any way to formalize the close relationship between the computation of these two measures?**

Our Results. We give a first step towards such a formalization, in the form of **a new problem**, which we show to be intermediary in terms of the techniques being used, between the Klee's measure and the maximum depth, and with interesting applications and results of its own. We introduce the notion of *depth distribution* of a set $\mathcal{B}$ of n boxes in $\mathbb{R}^d$, formed by the vector of n values

[1] We use capital letters to highlight names of Problems, and serif font to highlight names of measures of a set of boxes.

$(V_1, \ldots, V_n)$, where V_i corresponds to the volume covered by exactly i boxes from $\mathcal{B}$. The depth distribution of the set $\mathcal{B}$ can be interpreted as a probability distribution function (hence the name): if a point p is selected uniformly at random from the region covered by the boxes in $\mathcal{B}$, the probability that p hits exactly k boxes from $\mathcal{B}$ is $(V_k / \sum_{i=1}^{n} V_i)$, for all $k \in [1..n]$.

The depth distribution **refines both** the Klee's measure and the maximum depth. It is a measure finer than the Klee's measure in the sense that the Klee's measure of a set $\mathcal{B}$ can be obtained in time linear in the size n of $\mathcal{B}$ by summing the components of the depth distribution of $\mathcal{B}$. Similarly, the depth distribution is a measure finer than the maximum depth in the sense that the maximum depth of a set $\mathcal{B}$ can be obtained in linear time by finding the largest $i \in [1..n]$ such that $V_i \neq 0$. For an example of practical application of depth distribution, consider the case of a car dealer's website receiving queries about car within some range of price, safety rates or oil consumption rates per mile. Each query composed of r such ranges corresponds to a box in dimension r. The depth distribution of the set of queries received by a given dealer is a distribution of the interest of the clients in the space of all possible cars, and can server such dealer in guiding its future purchases in order to better satisfy the desires expressed by the clients queries.

We present computational upper bounds for the DEPTH DISTRIBUTION problem. In the classical computational complexity model where one studies the worst case over instances of fixed size n, the trivial approach (of partitioning the space into cells that are completely contained within all the boxes that they intersect), results in a solution with prohibitive running time within $\mathcal{O}(n^{d+1})$. Simple variants of the techniques previously used to compute the Klee's measure [10, 27] result in a solution running in time within $\mathcal{O}(n^{d/2+1})$, using linear space, or a solution running in time within $\mathcal{O}(n^{(d/2+1/2} \log n)$, but using space within $\mathcal{O}(n^{d/2} \log n)$. We combine those two techniques into a single one which computes the depth distribution in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$. We also analyze the computation of the depth distribution of a set of boxes in the refined complexity model where one studies the worst case complexity over

additional parameters which describe the difficulty of the instance beyond its size [4, 20]. We consider distinct measures of difficulty for the instances of these problems, such as the *profile* of the input set [13] and the *treewidth* of the intersection graph of the boxes [22], and describe algorithms in these models to compute the depth distribution, the Klee's measure and the maximum depth.

To complement these results, we study computational lower bounds for the Depth distribution problem. We show that this problem generalizes not only the Klee's Measure and Maximum Depth problems, but many other problems generally considered unrelated to sets of boxes. For instance, we prove that the classical Matrix Multiplication problem is a special case of the Depth distribution problem, a minor step towards answering the open question of the computational complexity of the Klee's Measure posed by Chan [9], and more importantly, yields a lower bound for the computational complexity of the depth distribution for d as small as 2 within $\Omega(n^{\frac{\omega}{2}})$, where ω is the Matrix Multiplication exponent. Such a lower bound suggests that the generalization of Klee's Measure and the Maximum Depth problems comes at a price: one cannot solve the Depth Distribution problem in the same asymptotic running time than the former two.

After a short overview of the known results on the Klee's Measure and the Maximum Depth problems (in Section 2), we present in Section 3 different algorithms for computing the depth distribution of a set of boxes. Then, in Section 4, we introduce conditional lower bounds for the Depth distribution problem, and conclude in Section 5 with a discussion on discrete variants and further refinements of the analysis.

2. Background

The techniques used to compute the Klee's measure have evolved over time, and can all be used to compute the maximum depth. We retrace some of the main results, which inspire the algorithms we present in Section 3.

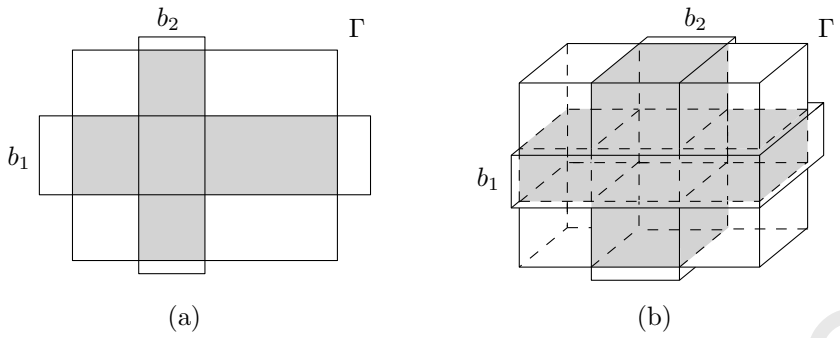


Figure 1: An illustration in dimensions 2 (a) and 3 (b) of two boxes b_1, b_2 equivalent to slabs when restricted to the box Γ. The Klee's measure of $\{b_1, b_2\}$ within Γ is the area (resp. volume) of the shadowed region in (a) (resp. (b)).

The computation of the Klee's measure of a set $\mathcal{B}$ of n d-dimensional boxes was first posed by Klee [21] in 1977. After some initial progresses [6, 15, 21], Overmars and Yap [27] described a solution running in time within $\mathcal{O}(n^{d/2} \log n)$. This remained the best solution for more than 20 years until 2013, when Chan [10] presented a simpler and faster algorithm running in time within $\mathcal{O}(n^{d/2})$.

The algorithms described by Overmars and Yap [27] and by Chan [10], respectively, take both advantage of solutions to the special case of the problem where all the boxes are slabs. A box b is said to be a *slab* within another box Γ if $b \cap \Gamma = \{(x_1, \ldots, x_d) \in \Gamma \mid \alpha \leq x_i \leq \beta\}$, for some integer $i \in [1..d]$ and some real values α, β (see Figure 1 for an illustration). Overmars and Yap [27] showed that, if all the boxes in $\mathcal{B}$ are slabs inside the domain box Γ, then the Klee's measure of $\mathcal{B}$ within Γ can be computed in linear time (provided that the boxes have been pre-sorted in each dimension).

Overmars and Yap's algorithm [27] is based on a technique originally described by Bentley [6]: solve the static problem in d dimensions by combining a data structure for the dynamic version of the problem in $d-1$ dimensions with a plane sweep over the d-th dimension. The algorithm starts by partitioning the space into $\mathcal{O}(n^{d/2})$ rectangular cells such that the boxes in $\mathcal{B}$ are equivalent to slabs when restricted to each of those cells. Then, the algorithm builds a *tree-like*

data structure whose leaves are the cells of the partition, supporting insertion and deletion of boxes during the plane sweep while keeping track of the Klee's measure of the boxes.

Chan's algorithm [10] is a simpler *divide-and-conquer* algorithm, where the slabs are *simplified* and removed from the input before the recursive calls (Chan [10] named this technique *Simplify, Divide and Conquer*, *SDC* for short). To obtain the recursive subproblems, the algorithm assigns a constant weight of $2^{\frac{i+j}{2}}$ to each (d-2)-face intersecting the domain and orthogonal to the i-th and j-th dimensions, $i, j \in [1..d]$. Then, the domain is partitioned into two sub-domains by the hyperplane $x_1 = m$, where m is the weighted median of the (d-2)-faces orthogonal to the first dimension. This yields a decrease by a factor of $2^{2/d}$ in the total weight of the (d-2)-faces intersecting each sub-domain. Chan [10] uses this, and the fact that slabs have no (d-2)-face intersecting the domain, to prove that the SDC algorithm runs in time within $\mathcal{O}(n^{d/2})$.

Unfortunately, there are sets of n boxes in $\mathbb{R}^d$ which require partitions of the space into a number of cells within $\Omega(n^{d/2})$ to ensure that every box in the set is equivalent to a slab when restricted to each cell. Hence, without a radically new technique, any algorithm based on this approach will run in time within $\Omega(n^{d/2})$.

The only lower bound known for the computational complexity of the Klee's Measure problem, other than the trivial $\Omega(n)$, was proved by Fredman and Weide [15] in the linear decision tree model. They showed a bound within $\Omega(n \log n)$ by reducing the ε-Closeness problem to the Klee's Measure problem on intervals. In higher dimensions, Chan [9] introduced a conditional lower bound of $\Omega(n^{d/2} - o(1))$ for any *combinatorial* algorithm, via a reduction from the parameterized k-Clique problem in which, given a graph on n nodes, the goal is to decide whether there are k that form a clique. The reduction allows to argue that, with current knowledge [**? ? ? ?**], one cannot hope for a purely *combinatorial* algorithm for the Klee's Measure problem that beats $\mathcal{O}(n^{d/2})$ time by more than polylogarithmic factors. He described an analogous argument for the Maximum Depth problem [10]. As a consequence, recent

work have focused on the study of special cases which can be solved faster than $\Omega(n^{d/2})$, like for instance when all the boxes are *orthants* [10], *α-fat boxes* [7], or cubes [8]. In turn, we show in Section 3.2 that there are measures which gradually separate *easy* instances for these problems from the *hard* ones.

In the next section, we present an algorithm for the computation of the depth distribution inspired by a combination of the approaches described above, outperforming naive applications of those techniques.

3. Algorithms Computing the Depth Distribution

We combine the techniques previously used to compute the Klee's measure [10, 27] into an algorithm to compute the depth distribution of a set of n d-boxes in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using $\mathcal{O}(n \log n)$-space. We present this algorithm in Section 3.1, and then, in Section 3.2, we refine this upper bound for various measures of difficulty of the input instances of the DEPTH DISTRIBUTION problem.

3.1. Worst-case Analysis

We introduce an algorithm to compute the depth distribution inspired by a combination of the techniques introduced by Chan [10], and by Overmars and Yap [27], for the computation of the Klee's measure. As in those approaches, the algorithm partitions the domain Γ into $\mathcal{O}(n^{d/2})$ cells inside which every box in $\mathcal{B}$ is equivalent to a slab. After that, it computes the depth distribution within each cell, and combines those solutions into the final answer. Two main issues must be addressed: how to compute the depth distribution when the boxes are slabs, and how to partition the domain efficiently.

We first address the special case of slabs. We show in Lemma 1 that computing the depth distribution of a set of n d-dimensional slabs within a domain Γ can be done by using an algorithm to multiply polynomials of degree at most n.

Lemma 1. *Let $\mathcal{B}$ be a set of n d-dimensional boxes whose intersection with a domain box Γ are slabs. The computation of the depth distribution $(V_1, \ldots, V_n)$ of $\mathcal{B}$ within Γ can be performed via a multiplication of d polynomials of degree at most n.*

Proof. Assume w.l.o.g. that no box in $\mathcal{B}$ covers completely the domain Γ. For all $i \in [1..d]$, let $\mathcal{B}_i$ be the subset of slabs that are orthogonal to the i-th dimension, and let $\left(V_1^i, \ldots, V_n^i\right)$ be the depth distribution within Γ of the intervals that result from projecting $\mathcal{B}_i$ into the i-th dimension. We associate a polynomial $P_i(x)$ of degree n with each $\mathcal{B}_i$ as follows:

- let Γ_i be the projection of the domain Γ into the i-th dimension, and
- let V_0^i be the length of the region of Γ_i not covered by a box in $\mathcal{B}_i$ (i.e., $V_0^i = (|\Gamma_i| - \sum_{j=1}^{n} V_j^i)$); then
- $P_i(x) = \sum_{j=0}^{n} V_j^i \cdot x^j$.

Since any slab entirely covers the domain in all the dimensions but the one to which it is orthogonal, any point p has depth k in $\mathcal{B}$ if and only if it has depths j_1 in $\mathcal{B}_1$, j_2 in $\mathcal{B}_2$, ..., and j_d in $\mathcal{B}_d$, such that $j_1 + j_2 + \ldots + j_d = k$. Thus, for all $k \in [0..n]$:

$$V_k = \sum_{\substack{0 \leq j_1, \ldots, j_d \leq n \\ j_1 + \ldots + j_d = k}} \left(\prod_{i=1}^{d} V_{j_i}^i \right),$$

which is precisely the $(k+1)$-th coefficient of $P_1(x) \cdot P_2(x) \cdot \ldots \cdot P_d(x)$. Thus, this product yields the depth distribution $(V_1, \ldots, V_n)$ of $\mathcal{B}$ in Γ. □

Using standard *Fast Fourier Transform* techniques, two polynomials can be multiplied in $O(n \log n)$-time (considering elementary arithmetic operations over the coefficients, such as additions and multiplications, as primitive operations that consume constant time) [12]. Moreover, the depth distribution of a set of intervals (i.e., when $d = 1$) can be computed in linear time after sorting, by a simple scan-line algorithm, as can be done for the Klee's measure [10]. Thus, as a consequence of Lemma 1, when the boxes in $\mathcal{B}$ are slabs within a domain box Γ, the depth distribution of $\mathcal{B}$ inside Γ can be computed in time within $\mathcal{O}(n \log n)$.

Corollary 1. *Let $\mathcal{B}$ be a set of n d-dimensional boxes whose intersections with a d-dimensional box Γ are slabs. The depth distribution of $\mathcal{B}$ inside Γ can be computed in time within $\mathcal{O}(n \log n)$.*

A naive application of previous techniques for the Klee's Measure problem [10, 27] to the computation of the depth distribution yields poor results:

- On one hand, combining the result described in Corollary 1 with the partition of the space and data structure described by Overmars and Yap [27] yields an algorithm to compute the depth distribution in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, but using prohibitive space within $\Theta(n^{d/2} \log n)$.
- On the other hand, combining the result in Corollary 1 with Chan's partition of the space [10], yields an algorithm using space linear in the number of boxes, but running in time within $\Theta(n^{\frac{d}{2}+1} \log n)$ (i.e., paying an extra $\mathcal{O}(n^{\frac{1}{2}})$-factor for the reduction in space usage of Overmars and Yap [27]).

We combine these two approaches into a recursive algorithm which achieves the best features of both: it runs in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, and uses $\mathcal{O}(n \log n)$-space. As in Chan's approach [10] we use a recursive simplify, divide and conquer algorithm, but we show that the running time is asymptotically the same as if the partition and data structure described by Overmars and Yap [27] were used (see Algorithm 1 for a detailed description). With each recursive step, the domain becomes smaller. When the domain is small enough such that all the boxes within it are slabs, we compute the depth distribution using Corollary 1 (base case, steps 1-4). We apply a simplification step before each recursive call removing the boxes that cover the entire domain and updating a value which keeps track of the number of such boxes (steps 6-8). To obtain the smaller domains for the recursive calls, we cut the current domain into two by a hyperplane, using for this the weighted median of the (d-2)-faces of the boxes orthogonal to the first dimensions (steps 9-12).

[2] As in Chan's approach [10], we consider the weight of a (d-2)-face orthogonal to dimensions x_i, x_j to be $2^{\frac{i+j}{d}}$, for $i, j \in [1..d]$. In particular, the weight of a (d-2)-face orthogonal to x_1 and x_j, for $j \in [1..d]$ is $2^{\frac{j+1}{d}}$. (see Section 2 for more details)

Algorithm 1 `SDC-DDistribution`$(\mathcal{B}, \Gamma, c, (V_1, \ldots, V_n))$

Input: A set $\mathcal{B}$ of n boxes in $\mathbb{R}^d$; a d-dimensional domain box Γ; the number c of boxes not in $\mathcal{B}$ but in the original set that completely contain Γ; and a vector $(V_1, \ldots, V_n)$ representing the depth distribution computed so far.

1: **if** no box in $\mathcal{B}$ has a (d-2)-face intersecting Γ (i.e., all the boxes are slabs) **then**
2: Compute the depth distribution $\left(V'_1, \ldots, V'_{|\mathcal{B}|}\right)$ of $\mathcal{B}$ within Γ using Corollary 1
3: **for** $i \in [1..|\mathcal{B}|]$ **do**
4: $V_{i+c} \leftarrow V_{i+c} + V'_i$
5: **else**
6: Let $\mathcal{B}^0 \subseteq \mathcal{B}$ be the subset of boxes completely containing Γ
7: $c \leftarrow c + |\mathcal{B}^0|$
8: Let $\mathcal{B}' = \mathcal{B} \setminus \mathcal{B}^0$
9: Let m be the weighted median [2]of the (d-2)-faces orthogonal to x_1
10: Split Γ into Γ_L, Γ_R by the hyperplane $x_1 = m$
11: Rename the dimensions so that $x_1, \ldots, x_d$ becomes $x_2, \ldots, x_d, x_1$
12: Let $\mathcal{B}_L$ and $\mathcal{B}_R$ be the subsets of $\mathcal{B}'$ intersecting Γ_L and Γ_R respectively
13: Call `SDC-DDistribution`$(\mathcal{B}_\mathcal{L}, \Gamma_L, c, (V_1, \ldots, V_n))$
14: Call `SDC-DDistribution`$(\mathcal{B}_\mathcal{R}, \Gamma_R, c, (V_1, \ldots, V_n))$

To analyze this algorithm, we first bound in Lemma 2 the height of its recursion tree with respect to the size $\mathcal{B}$, and then bound in Theorem 1 its running time and space usage.

Lemma 2. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$, and Γ be a domain box. When Algorithm 1 is executed with $\mathcal{B}$ and Γ as input, the height h of its recursion tree is at most $\frac{d}{2}\log n + \mathcal{O}(1)$.*

Proof. As described in Section 2, we consider the weight of a (d-2)-face orthogonal to dimensions x_i, x_j to be the value $2^{\frac{i+j}{d}}$ (which is always between 1 and 4). Let

W be the total weight of the (d-2)-faces of $\mathcal{B}$ that intersect Γ. We will show that $h \leq \lceil d/2 \log W \rceil$, thus proving the desired bound since $W \in \mathcal{O}(n)$ [3].

First, consider a (d-2)-face orthogonal to the i-th and j-th axes, with $i, j > 1$. After the renumbering in step 11, its weight changes from $2^{\frac{i+j}{d}}$ to $2^{\frac{i-1+j-1}{d}} = 2^{\frac{i+j}{d}-\frac{2}{d}}$, i.e., decreasing by a factor of $2^{2/d}$. Therefore, after each recursive invocation the total weight of such (d-2)-faces decrease by a factor of $2^{2/d}$.

Next, consider a (d-2)-face orthogonal to dimensions x_1 and x_j, with $j > 1$. After renumbering the axes, its weight changes from $2^{\frac{1+j}{d}}$ to $2^{\frac{d+j-1}{d}} = 2^{\frac{1+j}{d}+\frac{d-2}{d}}$, thus increasing by a factor of $2^{d-2}d$. However, after the partition of Γ into Γ_L and Γ_R in step 10, the total weight of the (d-2)-faces orthogonal to the first dimension and intersecting either subcell decreases by a factor of 2. This is due to the weighted median partition, and the fact that each such (d-2)-face cannot intersect the interior of Γ_L and Γ_R at the same time. Therefore, this implies a a net decrease by a factor of $2^{2/d}$ in the weights of such (d-2)-faces as well.

Since the total weight of all (d-2)-faces intersecting the domain drops by a factor of $2^{2/d}$ after each recursive invocation, any path in the recursion tree can have length at must $\lceil \log_{2^{2/d}} W \rceil = \lceil d/2 \log W \rceil$. Thus the height of the recursion tree can be at most this value. □

Theorem 1. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$. The depth distribution of $\mathcal{B}$ can be computed in time within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$, using space within $\mathcal{O}(n \log n)$.*

Proof. First, we show that the running time $T(n)$ of Algorithm 1 is within $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$. We can charge the number of boxes in the set to the number of (d-1)-faces intersecting the domain: if a box in $\mathcal{B}$ does not have a (d-1)-face intersecting the domain, then it covers the entire domain, and it would have been simplified (steps 6-8). Note that the (d-1)-faces orthogonal to dimension x_1 cannot intersect both the sub-domains Γ_L and Γ_R of the recursive calls at the same time (because the algorithm uses a hyperplane orthogonal to x_1 to split

[3] The arguments to prove the lemma are very similar to the those presented by Chan [10] for the Klee's Measure problem, we include them here 'aiming for completeness.

the domain into Γ_L and Γ_R). Hence, although at the d-th level of the recursion there are 2^d recursive calls, any (d-1)-face can appear in at most 2^{d-1} of those. In general, for any i, there are at most 2^i recursive calls at the i-th level of recursion, but any (d-1)-face of the original set can intersect at most $2^{\lceil i/d \rceil (d-1)}$ of the cells corresponding to the domain of those calls. Hence, the total number of (d-1)-faces which "survive" until the i-th level of the recursion tree is within $\mathcal{O}(n2^{i(d-1)/d})$ (a similar argument was used by Overmars and Yap [27] to bound the running time of the data structure they introduced): the ceiling can be ignored within the asymptotic notation, as it is hiding constants depending only on d.

Let h be the height of the recursion tree of Algorithm 1. By Lemma 2 we know that $h \in \frac{d}{2}\log n + \mathcal{O}(1)$. To bound $T(n)$, we analyze separately the total cost $T_I(n)$ of the *interior nodes* of the recursion tree (i.e., the nodes corresponding to recursive calls which fail the base case condition in step 1) from the total cost $T_L(n)$ of the *leaves* of the recursion tree.

Since, the cost of each interior node is linear in the number of (d-1)-faces intersecting the corresponding domain, $T_I(n)$ is bounded by:

$$\begin{aligned} T_I(n) &\in \sum_{i=1}^{h} \mathcal{O}\left(n \cdot 2^{i(d-1)/d}\right) \\ &\subseteq \mathcal{O}\left(n \sum_{i=1}^{h} 2^{i(d-1)/d}\right) \\ &\subseteq \mathcal{O}\left(n \cdot 2^{\frac{d-1}{d}h}\right) \\ &\subseteq \mathcal{O}\left(n \cdot 2^{\frac{d-1}{d}\frac{d}{2}\log n}\right) \qquad \text{(as } h \in \frac{d}{2}\log n + \mathcal{O}(1)\text{)} \\ &= \mathcal{O}\left(n \cdot n^{\frac{d-1}{2}}\right) = \mathcal{O}\left(n^{\frac{d+1}{2}}\right) \end{aligned}$$

To analyze the total cost of the leaves of the recursion tree, first note that the total number l of such recursive calls is within $\mathcal{O}(n^{d/2})$. Let $n_1, \ldots, n_l$ denote the number of (d-1)-faces in each of those recursive calls, respectively. Note that $T_L(n)$ is within $\mathcal{O}(\sum_{i=1}^{l} n_i \log n_i)$ because the result of Lemma 1 is used in step 1 of the algorithm. Besides, since the number of (d-1)-faces which survive until the h-th

level of the recursion tree is within $\mathcal{O}(n^{\frac{d-1}{2}})$, $\sum_{i=1}^{l} n_i \in \mathcal{O}(n^{\frac{d+1}{2}})$. That bound, and the fact that $\log n_i \leq \log n$, for all $i \in [1..l]$, yields $T_L(n) \in \mathcal{O}(n^{\frac{d+1}{2}} \log n)$. As $T(n) = T_I(n) + T_L(n)$, the bound for the running time follows.

With respect to the space used by the algorithm, note that only one path in the recursion tree is active at any moment, and that at most $\mathcal{O}(n)$ extra space is needed within each recursive call. Since the height of the recursion tree is within $\mathcal{O}(\log n)$, the total space used by the algorithm is clearly within $\mathcal{O}(n \log n)$. □

Note that in the algorithm `SDC-DDistribution`, the depth distribution is accumulated into a parameter. This is only to simplify the description and analysis of the algorithm, and does not impact its computational or space complexity. The initialization of the parameters of the algorithm `SDC-DDistribution` should be done as shown in Algorithm 2.

Algorithm 2 `DDistribution`$(\mathcal{B}, \Gamma)$

Input: A set $\mathcal{B}$ of n boxes in $\mathbb{R}^d$, a d-dimensional domain box Γ

Output: The depth distribution of $\mathcal{B}$ within Γ

1: $(V_1, V_2, \ldots, V_n) \leftarrow (0, 0, \ldots, 0)$

2: `SDC-DDistribution`$(\mathcal{B}, \Gamma, 0, (V_1, V_2, \ldots, V_n))$

3: return $(V_1, V_2, \ldots, V_n)$

The bound for the running time in Theorem 1 is worse than that of computing the Klee's measure (and maximum depth) by a factor within $\mathcal{O}(\sqrt{n} \log n)$, which raises the question of the optimality of the bound: we consider this matter in Section 4.

3.2. Adaptive Analysis

Even though the asymptotic complexity of $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$ is the best we know so far for the Depth distribution problem in the worst case, the constructions for which that bound is met are rather artificial. From a practical perspective there are many instances which can be solved faster. While some of those "easy" instances can be mere particular cases, others could be hints of some hidden

measures of difficulty of the DEPTH DISTRIBUTION problem. We show that there are at least two such difficulty measures, gradually separating instances of the same size n into various classes of difficulty. Informally, the first one measures how separable the boxes are by means of axis-aligned hyperplanes, while the second one measures how "complex" are the interactions between the boxes in the set.

3.2.1. A Profile-Sensitive Algorithm for the DEPTH DISTRIBUTION *problem*

A special type of set of boxes commonly arising in practice is that where the set has bounded profile [13, 26]. The *i-th profile* p_i of a set $\mathcal{B}$ of d-dimensional boxes is the maximum number of boxes intersected by any hyperplane orthogonal to the i-th dimension; and the *profile* p of $\mathcal{B}$ is $p = \min_{i \in [1..d]}\{p_i\}$. D'Amore et al. [13] showed how to compute its value in linear time (after sorting the coordinates of the boxes in each dimension).

We show in the following lemma that the depth distribution can be computed in time sensitive to the profile of the input set.

Theorem 2. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$ with profile p, and Γ be a d-dimensional domain box. The depth distribution of $\mathcal{B}$ within Γ can be computed in time within $\mathcal{O}(n \log n + np^{\frac{d-1}{2}} \log p) \subseteq \mathcal{O}(n^{\frac{d+1}{2}} \log n)$.*

Proof. We describe an algorithm which partitions the domain Γ into disjoint cells, computes the depth distribution within each cell, and combines the results into the final answer. For this, it sweeps the boxes with a hyperplane orthogonal to the dimension with smallest profile, and after every $2p$ endpoints of the boxes, it creates a new cell cutting the space with a hyperplane orthogonal to this dimension. This yields a partition of Γ into $\mathcal{O}(n/p)$ cells, each intersecting at most $\mathcal{O}(p)$ boxes. Finally, the algorithm computes the depth distribution of $\mathcal{B}$ within each cell in time within $\mathcal{O}(p^{\frac{d+1}{2}} \log p)$, and obtains the depth distribution of $\mathcal{B}$ within Γ by summing the respective components of the depth distribution within each slab. In total, this takes time within $\mathcal{O}(n \log n + np^{\frac{d-1}{2}} \log p)$. □

The theorem above automatically yields refined results for the computation of the Klee's measure and maximum depth of a set of boxes $\mathcal{B}$. However, applying the technique in an *ad-hoc* way to these problems yields a slightly better bound:

Corollary 2. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$ with profile p, and Γ be a d-dimensional domain box. The Klee's measure and maximum depth of $\mathcal{B}$ within Γ can be computed in time within $\mathcal{O}(n \log n + np^{\frac{d-2}{2}}) \subseteq \mathcal{O}(n^{d/2})$.*

The algorithms from Theorem 2 and Corollary 2 asymptotically outperform previous ones in the sense that their running time is never worse than previous algorithms by more than a constant factor when the profile p is within $\mathcal{O}(n^{1-\varepsilon})$, for some constant $\varepsilon > 0$.

An orthogonal approach is to consider how complex are the interactions between the boxes in the input set $\mathcal{B}$, analyzing, for instance, the intersection graph of $\mathcal{B}$. We study such a technique in the next section.

3.2.2. Adaptivity to the Treewidth and Degeneracy of the Intersections Graph

The *treewidth* of a graph is a concept which captures how "close" to a tree the graph is. The treewidth was introduced independently several times under different names, and many graph problems that are NP-hard for general graphs can be solved in polynomial time for graphs with small treewidth (see Sections 10.4 and 10.5 of Kleinberg and Tardos's book [22] for a nice overview). We describe below a technique to improve the running time of the depth distribution of sets of boxes whose intersection graphs have small treewidth.

A *k-degenerate* graph is an undirected graph in which every subgraph has a vertex of degree at most k [24]. The *degeneracy* of a graph G is the smallest value k such that G is k-degenerate. Every k-degenerate graph accepts an ordering of the vertices (called *degenerate ordering*) in which every vertex is connected with at most k of the vertices that precede it.

In the following lemma we show that this ordering can be used to compute the depth distribution of a set $\mathcal{B}$ of n boxes in running time sensitive to the degeneracy of the intersection graph of $\mathcal{B}$.

Lemma 3. *Let $\mathcal{B}$ be a set of n d-boxes, let Γ be a domain d-box, and let k be the degeneracy of the intersection graph G of $\mathcal{B}$. The depth distribution of $\mathcal{B}$ in Γ can be computed in time within $\mathcal{O}(n \log^d n + e + nk^{\frac{d+1}{2}})$, where $e \in \mathcal{O}(n^2)$ is the number of edges of G.*

Proof. We describe an algorithm that runs in time within the bound in the lemma. The algorithm first computes the intersection graph G of $\mathcal{B}$ in time within $\mathcal{O}(n \log^d n + e)$ [14], as well as the k-degeneracy of this graph and a degenerate ordering O of the vertices in time within $\mathcal{O}(n + e)$ [25]. For $i \in [1..n]$ let $O[1..i]$ denote the first i vertices of O, and $O[i]$ denote the i-th vertex of O. The algorithm then iterates over O maintaining the invariant that, after the i-th step, the depth distribution within Γ of the boxes corresponding to vertices in $O[1..i]$ has been correctly computed.

For any subset U of vertices of G, let $\mathtt{DD}^{\Gamma}_{\mathcal{B}}(U)$ denote the depth distribution within Γ of the boxes in $\mathcal{B}$ corresponding to the vertices in U. From $\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i\text{-}1])$ (which the algorithm "knows" after the (i-1)-th iteration), $\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i])$ can be obtained as follows: (*i.*) let P be the subset of $O[1..i\text{-}1]$ adjacent to $O[i]$; (*ii.*) compute $\mathtt{DD}^{O[i]}_{\mathcal{B}}(P \cup \{O[i]\})$ in time within $\mathcal{O}(k^{\frac{d+1}{2}} \log k)$ using Algorithm 1 (note that the domain this time is $O[i]$ itself, instead of Γ); (*iii.*) add to $(\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i]))_1$ the value of $(\mathtt{DD}^{O[i]}_{\mathcal{B}}(P \cup O[i]))_1$; and (*iv.*) for all $j = [2..k{+}1]$, subtract from $(\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i]))_{j-1}$ the value of $(\mathtt{DD}^{O[i]}_{\mathcal{B}}(P \cup O[i]))_j$ and add it to $(\mathtt{DD}^{\Gamma}_{\mathcal{B}}(O[1..i\text{-}1]))_j$.

Since the updates to the depth distribution in each step take time within $\mathcal{O}(k^{\frac{d+1}{2}} \log k)$, and there are n such steps, the lemma follows. □

The degeneracy k of a graph G is always at most the treewidth t of G [22] (i.e. $k \leq t$). Thus, the algorithm described above is sensitive to the treewidth of the intersection graph.

Theorem 3. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$, let Γ be a d-dimensional domain box, and let t be the treewidth of the intersection graph G of the boxes in $\mathcal{B}$. The depth distribution of $\mathcal{B}$ within Γ can be computed in time within $\mathcal{O}(n \log^d n + e + nt^{\frac{d+1}{2}})$, where $e \in \mathcal{O}(n^2)$ is the number of edges of G.*

Unlike the algorithm sensitive to the profile described in Section 3.2.1, this algorithm can run in time within $\mathcal{O}(n^{1+\frac{d+1}{2}})$ (e.g., when $k \in \Theta(n)$) in the worst-case, which is better than the $\mathcal{O}(n^{\frac{d+1}{2}})$ complexity of algorithm `SDC-DDistribution` only for values of the degeneracy k within $\mathcal{O}(n^{1-\frac{2}{d}})$. A simple dove-tailing combination yields a solution whose asymptotic running time is the best of both.

As in the case of the algorithm sensitive to the profile of the input instance, applying the same technique in an *ad-hoc* fashion to the Klee's Measure and Maximum Depth problems yields improved solutions:

Corollary 3. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$, let Γ be a d-dimensional box, and let t be the treewidth of the intersection graph G of $\mathcal{B}$. The Klee's measure and maximum depth of $\mathcal{B}$ within Γ can be computed in time within $\mathcal{O}(n \log^d n + e + nt^{\frac{d}{2}})$, where $e \in \mathcal{O}(n^2)$ is the number of edges of G.*

The algorithms for the Depth distribution problem described here, even those taking advantage of distinct measures of difficulty of the input, are computationally more expensive than their analogy for the Klee's Measure and Maximum Depth problems. In the next section, we argue on why such a gap in the computational complexity of these problems might require major breakthrough results.

4. Lower Bounds for the Depth distribution problem

The only non-trivial lower bound known for the computational complexity of the Depth distribution problem is $\Omega(n \log n)$ [15] (in the decision tree model with linear tests), which follows from the fact that this problem has the Klee's Measure problem as a special case (see Section 2 for details). Note, however, that this bound is known to be tight only when the input is a set of intervals (i.e., $d = 1$). For higher dimensions, the lower bound of $\Omega(n^{d/2})$ conjectured by Chan in 2008 [9] for the computational complexity of the Klee's Measure problem can be extended analogously to the computational complexity of the Depth distribution problem. Again, note that there is a gap between this

conjectured lower bound and the upper bound of $\mathcal{O}(n^{\frac{d+1}{2}} \log n)$ described in Section 3.

In this section we study whether the gap between the computational complexities of the Depth distribution problem and the Klee's Measure and Maximum Depth problems can be eluded. Although we do not know the answer to this question yet, we argue that it is in fact a very hard (and relevant) question about not only boxes, but other central problems in Computer Science. We show that proving that such a gap can be eluded implies breakthrough results for two fundamental problems in computer science, the Integer Multiplication and Matrix Multiplication problems:

- First, we consider in Section 4.1 the case when all the boxes are slabs, and the input is given pre-sorted in each dimension. Both the Klee's measure and the maximum depth of such instances can be computed in linear time. We show that if there is an algorithm computing the depth distribution of such instances in linear time then there is also a linear-time algorithm for multiplying two n-bit numbers, which would contradict a conjecture of Schönhage and Strassen [30] for the Integer Multiplication problem, widely accepted by the community [16, 18].

- Then, in Section 4.2 we consider the planar case of the Depth distribution problem. The Klee's measure and maximum depth of a set of n rectangles can be computed in time within $\mathcal{O}(n \log n)$ [10]. We show that any algorithm computing the depth distribution in time within $\mathcal{O}(n \log n)$ can be transformed into an algorithm which multiplies two $n \times n$ matrices in time within $\mathcal{O}(n^2 \log n)$. Thus, any such algorithm for the Depth distribution problem would solve a fundamental and long-standing question for the Matrix Multiplication problem [11, 17, 29, 31].

4.1. Conditional lower bound for the case of sorted slabs

Let Γ be a domain d-box, and let $\mathcal{B}$ be a set of n d-boxes which are all slabs within Γ. Suppose that, together with Γ and $\mathcal{B}$, we are given as part of the input

the relative order of the endpoints of the boxes in $\mathcal{B}$ in each dimension. It is well known that both the Klee's measure and maximum depth of $\mathcal{B}$ can be computed in linear time under these settings [10, 27]. We argue that the same is not true for the depth distribution: we prove that computing the depth distribution of $\mathcal{B}$ requires time within $\Omega(n \log n)$, unless two n-bit integers can be multiplied in time within $o(n \log n)$.

Computing the product of two n-bit integers, known as the Integer Multiplication problem, is an important fundamental problem in algorithmic number theory, algebra and theoretical computer science [23]. A naive approach leads to an algorithm that uses $\mathcal{O}(n^2)$ bit operations, but in 1963 Karatsuba and Ofman [19] showed how to reduce the number of operations to within $\mathcal{O}(n^{\log_2 3})$. In a major breakthrough, in 1971, Schönhage and Strassen [30] described an efficient algorithm for multiplying integers using fast polynomial multiplication, and running in time within $\mathcal{O}(n \cdot \log n \cdot \log \log n)$. Since then, the prevailing (and widely accepted) conjecture has always been that the computational complexity of this problem is within $\Theta(n \log n)$ [16, 18, 30].

We show that any algorithm $\mathcal{A}$ for the Depth distribution problem over a set of pre-sorted slabs can be used to multiply two n-bit integers via a reduction using $\mathcal{O}(n)$ bit operations. We do this in two steps: first, in Lemma 4, we show that any two polynomials with non-negative integer coefficients can be multiplied using $\mathcal{A}$, and then in Theorem 4, using the fact that the product of two n-bits integers can be obtained by means of an algorithm for polynomial multiplication, we prove the conditional lower bound. Since the Integer Multiplication problem is usually studied in the multi-tape Turing machine model (also referred to as the *bit-complexity* model) [28], we show that our reduction can be executed in linear time on a 3-tape Turing machine. Thus, the conditional lower bound described here also applies to any computational model with at least as much power as a multi-tape Turing machine (which is already a highly restrictive computational model).

Lemma 4. *Let $P(x), Q(x)$ be two m-degree polynomials with integer coefficients in $[0..(2^m-1)]$, and let $n = 2m^2$ (i.e., n is a bound for the size in bits of the representations of P, Q). There is a set $\mathcal{B}$ with $2m$ rectangles, and a rectangle Γ such that:*

1. *Both the domain Γ, and the set $\mathcal{B}$, can be computed in time within $\mathcal{O}(n)$;*
2. *All the rectangles, including Γ have their endpoints in $[0..m2^m]$;*
3. *All the rectangles of $\mathcal{B}$ are slabs in Γ, and the relative order of their endpoints in each dimension is known;*
4. *If the depth distribution of $\mathcal{B}$ within Γ can be computed in time $T(n)$, then the polynomial $P(x) \cdot Q(x)$ can be computed in time within $\mathcal{O}(T(n) + n)$.*

Proof. Given $P(x)$ and $Q(x)$, we create a set $\mathcal{B}$ with $2m$ rectangles, and a domain rectangle Γ, such that from the depth distribution $(V_1, V_2, \ldots, V_{2m})$ of $\mathcal{B}$ within Γ, the coefficients of $P(x) \cdot Q(x)$ can be obtained via a simple linear time procedure. Let p_i and q_i denote the i-th coefficient of $P(x)$ and $Q(x)$, respectively, for all $i = [1..m]$. Choose $\mathcal{B}$ and Γ as follows (see Figure 2 for an illustration):

1. Take Γ as the rectangle $\{(x, y) \mid 0 \leq x \leq \sum_{j=0}^{m} p_i, 0 \leq y \leq \sum_{j=0}^{m} q_i\}$;
2. Take $\mathcal{B} = \mathcal{B}_p \cup \mathcal{B}_q$, where $\mathcal{B}_p$ and $\mathcal{B}_q$ are the sets of slabs (orthogonal to the x and y axes, respectively) defined as:

$$\mathcal{B}_p = \bigcup_{i=1}^{m} \left\{ b_i^p = \left\{ (x, y) \in \Gamma \mid 0 \leq x \leq \sum_{k=0}^{i} p_{m-k} \right\} \right\}$$
$$\mathcal{B}_q = \bigcup_{i=1}^{m} \left\{ b_i^q = \left\{ (x, y) \in \Gamma \mid 0 \leq y \leq \sum_{k=0}^{i} q_{m-k} \right\} \right\}.$$

Let Γ_x, Γ_y denote the intervals resulting from projecting Γ to the x and y axes, respectively, and let $|I|$ denote the length of any interval I. $\mathcal{B}_p$ is defined so that when its m rectangles are projected to the x-axis, the length of the region in Γ_x covered by exactly k rectangles is precisely p_k, for all $k = [0..m]$. Thus, if we let $(V_1^x, \ldots, V_m^x)$ denote the depth distribution of the projection of $\mathcal{B}_p$ to the

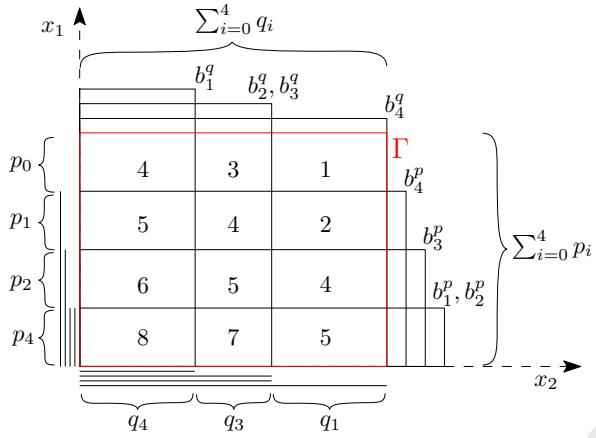


Figure 2: A product $P(n) \cdot Q(n)$ as an instance of Depth Distribution, for $P(n) = p_4n^4 + p_2n^2 + p_1n + p_0$, and $Q(n) = q_4n^4 + q_3n^3 + q_1n$. The set of rectangles generated is $\mathcal{B} = \mathcal{B}_p \cup \mathcal{B}_q$, where $\mathcal{B}_p = \{b_1^p, b_2^p, b_3^p, b_4^p\}$ and $\mathcal{B}_q = \{b_1^q, b_2^q, b_3^q, b_4^q\}$. The number within each cell indicates the depth of the corresponding region. The lines to the left of (resp. below) the x_1 axis (resp. the x_2 axis) illustrate the intervals that would result from projecting $\mathcal{B}_p$ (resp. $\mathcal{B}_q$) to x_1 (resp. to x_2). The numbers over curly brackets indicate the length of the region delimited by the brackets.

x-axis within Γ_x, and let $V_0^x = |\Gamma_x| - \sum_{i=1}^{m} V_i^x$, then $p_k = V_k^x$, for all $k = [0..m]$. Analogous observations can be made for $Q(x), \mathcal{B}_q, \Gamma_y$ and $(V_0^y, V_1^y, \ldots, V_m^y)$.

Let c_k denote the k-th coefficient of $P(x) \cdot Q(x)$. Since any point p has depth k in $\mathcal{B}$ if and only if it has depth i in $\mathcal{B}_p$ and j in $\mathcal{B}_q$ such that $i + j = k$, for all $k \in [1..2m]$ we have:

$$V_k = \sum_{\substack{0 \le i,j \le m \\ i+j=k}} V_i^x \cdot V_j^y \quad = \sum_{\substack{0 \le i,j \le m \\ i+j=k}} p_i \cdot q_j \quad = \quad a_k$$

Thus, the $2m$ components of the depth distribution of $\mathcal{B}$ within Γ are precisely the first $2m$ coefficients (in descending order of the exponents) of $P(x) \cdot Q(x)$. Besides, the constant coefficient of the product is given by $|\Gamma| - \sum_{i=1}^{m} V_i$, where $|\Gamma|$ is the volume of Γ. Therefore, the product $P(x) \cdot Q(x)$ can be obtained almost directly from the Depth Distribution of $\mathcal{B}$ and Γ.

Now, let us analyze the running time of this reduction in a 3-Tapes Turing Machine (the input and output tapes, and one additional tape to keep the

partial sums). We assume the input tape contains first all the coefficients of $P(x)$ in descending order of the exponents, and then all the coefficients of $Q(x)$. To produce the set $\mathcal{B}_p$ the coefficients of $P(x)$ are read one by one, keeping track of the partial sum of the ones read so far in the additional tape. To process the i-th coefficient of $P(x)$ we add p_i to the partial sum, and write to the output tape the box b_i^p. Note that, to represent each endpoint of a box in $\mathcal{B}$, $\lceil \log(m \cdot 2^m) \rceil$-bits suffice, and thus processing each coefficient of $P(x)$ costs within $\mathcal{O}(\log(m \cdot 2^m))$ operations. Therefore, the total operations required to produce $\mathcal{B}_p$ is within $\mathcal{O}(m \log(m \cdot 2^m)) \subseteq \mathcal{O}(m \log m + m^2) \subseteq \mathcal{O}(n)$. The same is true for $\mathcal{B}_q$, and therefore also for $\mathcal{B}$. Besides, Γ can be computed also trivially with $\mathcal{O}(m)$ operations. Finally, let $\mathcal{A}$ be an algorithm which computes the depth distribution of $\mathcal{B}$ within Γ in time $T(n)$. We execute $\mathcal{A}$ over $\mathcal{B}, \Gamma$, and produce the polynomial $P(x) \cdot Q(x)$ from the output of $\mathcal{A}$ as already described, which can be done with a total number of operations linear in size of the representation of $P(x) \cdot Q(x)$. To complete the proof, let us bound the bits required to represent $P(x) \cdot Q(x)$. Note that $P(x) \cdot Q(x)$ has at most $2m$ coefficients, and since each of those coefficients is the sum of at most 2^m coefficients of $P(x)$ and $Q(x)$, the number of bits required to represent it is within $\mathcal{O}(\log(2^m \cdot 2^m)) \subseteq \mathcal{O}(m)$. Therefore, the total number of bits in $P(x) \cdot Q(x)$ is within $\mathcal{O}(n)$. □

Since the product of two n-bit integers can be obtained from the product of two polynomials with non-negative integer coefficients, the result of Lemma 4 yields the following conditional lower bound for the Depth distribution problem on pre-sorted slabs:

Theorem 4. *Let Γ be a domain d-box, and let $\mathcal{B}$ be a set of d-boxes which are all slabs within Γ, and let n be a the number of bits required to represent $\mathcal{B}$ and Γ. The depth distribution of $\mathcal{B}$ within Γ cannot be computed in time within $o(n \log n)$ unless two n-bit integers can be multiplied using within $o(n \log n)$ bit operations.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ which computes the depth distribution of $\mathcal{B}$ within Γ using $T(n) \in o(n \log n)$ bit operations. Let a, b be two n-bit integers. Assume that n is a power of 2 (if not, we can add padding zeros until

this condition is met), and let $m = \sqrt{n}$. The product $c = a \cdot b$ can be computed using $\mathcal{A}$ as follows:

1. Split a and b into m blocks $\{a_1, \ldots, a_m\}$ and $\{b_1, \ldots, b_m\}$, respectively, such that each block has m bits, and

   $$a = \sum_{i=0}^{m-1} a_i 2^{im}, \quad b = \sum_{i=0}^{m-1} b_i 2^{im} \qquad \text{(equivalently, express } a, b \text{ in base } 2^m\text{)};$$

2. Let $P_a(x), P_b(x)$ be the m-degree polynomials defined as

   $$P_a(x) = \sum_{i=0}^{m-1} a_i x^i, \quad P_b(x) = \sum_{i=0}^{m-1} b_i x^i \quad \text{(note that } a = P_a(2^m), b = P_b(2^m)\text{)};$$

3. Compute $P_c(x) = P_a(x) \cdot P_b(x)$ using algorithm $\mathcal{A}$ as in Lemma 4;
4. Finally, perform carrying $\mod (2^m + 1)$ over the coefficients of $P_c(x)$ in ascending order of their exponents to obtain c.

This procedure, inspired by Schönhage and Strassen's algorithm [30], performs within $\mathcal{O}(T(n) + n)$ operations on a 3-tapes Turing machine. This is clear for the first two steps, and for the third one it directly derives from Lemma 4. For the fourth step, note that $P_c(x)$ has at most $2m$ coefficients, each of those being the sum of at most 2^m coefficients of $P_a(x)$ and $P_b(x)$. Thus, the number of bits required to represent each coefficient of $P_c(x)$ is within $\mathcal{O}(\log(2^m \cdot 2^m)) \subseteq \mathcal{O}(m)$, and the total number of bits in $P_c(x)$ is within $\mathcal{O}(n)$. □

Therefore, proving that the depth distribution of a set of sorted slabs can be computed in the same running time than its Klee's measure or its maximum depth, would contradict Schönhage and Strassen's conjecture [30] of the optimality of $\Theta(n \log n)$ for the complexity of the Integer Multiplication problem.

4.2. Conditional lower bound for sets of planar boxes

For general sets of boxes in the Euclidean plane, one can also argue that the Depth distribution problem is computationally harder than the Klee's Measure and the Maximum Depth problems. Proving the opposite would

imply breakthrough results in the long-standing problem of Matrix Multiplication. We prove that any instance of Matrix Multiplication can be solved by using an algorithm which computes the depth distribution of a set of planar boxes. For this, we make use of the following simple observation:

Observation. *Let A, B be two $n \times n$ matrices of real numbers, and let C_i denote the $n \times n$ matrix that results from multiplying the $n \times 1$ vector corresponding to the i-th column of A with the $1 \times n$ vector corresponding to the i-th row of B. Then, $AB = \sum_{i=1}^{n} C_i$.*

We show in Theorem 5 that multiplying two $n \times n$ matrices can be done by transforming the input into a set of $\mathcal{O}(n^2)$ axis aligned rectangles, and computing the depth distribution of the resulting set. Moreover, this transformation can be done in linear time, thus, the theorem yields a conditional lower bound for the computation of the depth distribution.

Theorem 5. *Let A, B be two $n \times n$ matrices of non-negative real numbers. There is a set $\mathcal{B}$ of planar boxes of size within $O(n^2)$, and a domain box Γ, such that the depth distribution of $\mathcal{B}$ within Γ can be projected to obtain the value of the product AB. Moreover, the set $\mathcal{B}$ can be computed in $\mathcal{O}(n^2)$ time.*

Proof. We create a *gadget* to represent each C_i. Within the i-th gadget, there will be a rectangular region for each coefficient of C_i with the value of that coefficient as area (see Figure 3 for a general outlook of such instance). We arrange the boxes so that two of such rectangular regions have the same depth if and only if they represent the same respective coefficients of two different matrices C_i and $C_{i'}$ (formally, they represent some coefficients $(C_i)_{j,k}$ and $(C_{i'})_{j',k'}$, respectively, such that $i \neq i', j = j'$, and $k = k'$).

We describe a set $\mathcal{B}$ of $5n^2$ planar boxes (one box for each of the n^2 coefficients of A, two boxes for each of the n^2 coefficients of B, and $2n^2$ additional boxes) such that, for each $i, j \in [1..n]$, the $(2ni + 2j + 1)$-th component of the depth distribution of $\mathcal{B}$ is equal to the component $AB_{i,j}$ of the product AB. Such a set can be constructed in linear time as follows (see Figure 4 on page 30 for a graphical representation of such an instance):

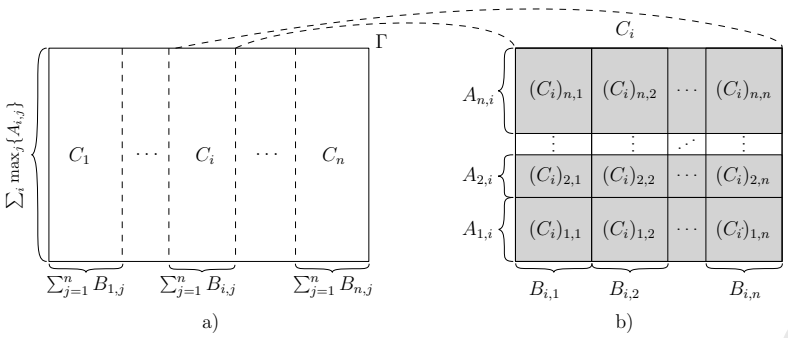


Figure 3: An outlook of the instance generated for the product AB: we add a gadget for each $C_1, \ldots, C_n$, within a domain Γ as in a). In b), a representation of C_i with $2n$ boxes, the volumes of the rectangular regions correspond to the coefficients of C_i (the regions in a gadget must have different depths to avoid that their areas are added into a same component of the depth distribution).

- Let the domain $\Gamma = \{(x, y) \mid 0 \leq x \leq \sum_i \sum_j B_{i,j}, 0 \leq y \leq \sum_i \max_j\{A_{i,j}\}\}$.
- For all $i \in [1..n]$ we create a gadget for C_i that covers the entire domain in the y-direction, and that spans from $C_i^{start} = \sum_{j=1}^{i-1} \sum_{k=0}^{n} B_{j,k}$ to $C_i^{end} = C_i^{start} + \sum_{k=0}^{n} B_{i,k}$ in the x-direction. This step can be done in $\mathcal{O}(n^2)$ time by pre-computing the partial sums needed for the constructions.
- Within the gadget for C_i we place one box for each $A_{j,i}$ and two boxes for each $B_{i,j}$, for $i, j \in [1..n]$, as follows: the one corresponding to $A_{j,i}$ will span C_i entirely in the x-direction, and is bounded by $(\sum_{k=1}^{j} \max_{l=1}^{n}\{A_{k,l}\}) \leq y \leq (A_{j,i} + \sum_{k=1}^{j} \max_{l=1}^{n}\{A_{k,l}\})$ in the y-direction. For $B_{i,j}$ we place two identical boxes entirely spanning C_i in the y-direction, and in the x-direction bounded by $(C_i^{start} + \sum_{k=1}^{j-1} B_{i,k}) \leq x \leq C_i^{end}$. Again, by pre-computing and storing the partial sums needed for the construction we can perform this step in time within $\mathcal{O}(n^2)$.
- Finally, we add $2n^2$ boxes to ensure that rectangular regions corresponding to two coefficients $C_{i,j}$ and $C_{i,k}$ in different rows j, k of a same C_i do not share the same depth, for all $i, j, k \in [1..n]$. For this, for all $j \in [1..n]$ we

add $2n$ identical boxes entirely spanning the domain in the x-direction, and spanning from $(\sum_{k=1}^{j} \max_{l=1}^{n}\{A_{k,l}\})$ to $(\sum_i \max_j\{A_{i,j}\}))$ in the y-direction.

Note that in the instance generated, for $i, j \in [1..n]$:

- a region has odd depth if and only if its area is equal to some coefficient of any C_i;
- the regions corresponding to coefficients of the i-th rows have depth between $(2in + 3)$ and $(4in + 1)$;
- within the gadget for each C_i, the rectangular region with volume corresponding to the coefficient $C_{i,j}$ has depth $(2ni + 2j + 1)$, and no other rectangular region within the gadget has that depth;
- two regions have the same depth if and only if they represent the same respective coefficients of two matrices C_i and $C_{i'}$.

The arguments above and the fact that, by definition of the depth distribution, the volumes of regions with the same depth are accumulated together, yield the result of the theorem. □

The optimal time to compute the product of two $n \times n$ matrices in an arithmetic circuit [17] is still an intriguing open question. It can naturally be computed using within $O(n^3)$ arithmetic operations. However, Strassen showed in 1969 that within $O(n^{2.81})$ arithmetic operations are enough [31]. This gave rise to a new area of research, where the central question is to determine the value of the exponent of the computational complexity of square matrix multiplication, denoted ω, and defined as the minimum value such that two $n \times n$ matrices can be multiplied using within $O(n^{\omega+\varepsilon})$ arithmetic operations for any $\varepsilon > 0$.

The result of Theorem 5 directly yields a conditional lower bound on the complexity of the Depth distribution problem: in particular, if the depth distribution of n boxes in the plane can be computed in time within $\mathcal{O}(n \log n)$, then Matrix Multiplication can be computed in time within $\mathcal{O}(n^2 \log n)$,

implying that $\omega = 2$. However, this would be a great breakthrough in the area, the best known upper bound to date is approximately $\omega \leq 2.37$, when improvements in the last 30 years [11, 17, 29] have been in the range $[2.3728, 2.3754]$.

Note that although the reduction described in the proof of Theorem 5 works only for matrices with non-negative coefficients, this fact does not weaken the result in the theorem. The multiplication of two matrices with some negative coefficients can be obtained from the multiplication of two matrices with only positive coefficients, and such a transformation requires only time linear in the size of the matrices. Moreover, even for the multiplication of boolean matrices (i.e., matrices with all its coefficients either zero or one) the best bound known so far for ω is the same as for the general problem [34]. Since the set $\mathcal{B}$ of Theorem 5 can be obtained using within $\mathcal{O}(n)$ arithmetic operations in an arithmetic circuit, we obtain the following conditional lower bound:

Corollary 4 (Conditional lower bound)**.** *Computing the depth distribution of a set $\mathcal{B}$ of n d-dimensional boxes requires within $\Omega(n^{1+c})$ arithmetic operations, for some constant $c > 0$, unless two $n \times n$ matrices can be multiplied using a number of arithmetic operations within $\mathcal{O}(n^{2+\varepsilon})$, for any constant $\varepsilon > 0$*

In the next section we summarize the results described in this article and present some future directions of research on the Depth Distribution problem.

5. Discussion

The computation of the depth distribution generalizes not only that of the Klee's measure and the maximum depth, but also generalizes many other central problems with no apparent relation to orthogonal boxes (such as Matrix Multiplication and its generalizations). As a measure, the depth distribution captures many of the features in common between the Klee's measure and the maximum depth, so that any new upper bound on the computation of the depth distribution will yield corresponding results for the computation of those two measures. Nevertheless, we know of no direct reduction from the Depth distribution problem to Klee's Measure or the Maximum Depth problems,

and in fact we argued that such reduction is unlikely. We discuss below some further issues to ponder about these measures.

Discrete variants. In practice, multidimensional range queries are applied to a database of multidimensional points. This yields discrete variants of each of the problems previously discussed, such as the DISCRETE KLEE'S MEASURE problem defined by Yildiz et al.. In this discrete variant, the input is composed not only of a set $\mathcal{B}$ of n boxes, but also of a set S of m points. The goal is now to compute not the volume of the union of the boxes, but the number (and/or the list) of points which are covered by those boxes. Inspired by this definition, we define a discrete version of the MAXIMUM DEPTH problem (which points are covered by the maximum number of boxes) and of the DEPTH DISTRIBUTION problem (how many and which points are covered by exactly i boxes, for $i \in [1..n]$). Interestingly enough, the computational complexity of these discrete variants is much less than that of their continuous versions when there are reasonably few points [33]: the discrete variant becomes hard only when there are many more points than boxes [1]. Nevertheless, "easy" configurations of the boxes also yield "easy" instances in the discrete case: it will be interesting to analyze the discrete variants of these problems according to the *profile* of the input set, and according to the *degeneracy* and *treewidth* of its intersection graph, which we introduced for the continuous versions in Section 3.2.

Tighter Bounds. Chan [9] conjectured that any *combinatorial* algorithm which computes the Klee's measure or the maximum depth of a set of n boxes in $\mathbb{R}^d$ requires running time within $\Omega(n^{d/2})$. This conjectured lower bound on the computation of the Klee's measure applies of course to the more general problem of computing the depth distribution. However, the running time of the algorithm described in Section 3.1 can be worse than that lower bound by a factor within $\mathcal{O}(\sqrt{n}\log n)$. The gap between the asymptotic upper and lower bounds on the computational complexity of the DEPTH DISTRIBUTION problem could be closed either by finding an improved algorithm, or by proving more restrictive lower bounds: it is not clear which is the best candidate. The depth distribution of

a set of boxes yields much more information than its Klee's measure (actually, a large part of this information can be ignored during the computation of the Klee's measure). This may hint that computing the depth distribution is actually asymptotically harder than the special cases, and that proving more restrictive lower bounds is plausible. We were able to argue in Section 3.2 that at least in two dimensions this is the case, but it is unclear whether (or how) that argument can be extended to higher dimensions.

Special cases of the input. As a consequence of the lower bound conjectured by Chan [9], recent work on the computation of the Klee's measure [7, 8, 10] have focused on the study of special cases which can be solved in time within $o(n^{d/2})$. For instance, Chan showed [10] that when all the boxes are *orthants* the Klee's measure can be computed in time within $O(n^{d/3} \operatorname{polylog} n)$; various authors showed [5, 7, 8, 10] that the Klee's measure of a set of hypercubes can be computed in time within $O(n^{(d+1)/3} \operatorname{polylog} n)$; and Yildiz and Suri showed [32] that the case of 2-*grounded* boxes can be solved in time within $O(n^{(d-1)/2} \log^2 n)$, for any dimension $d \geq 3$. Similar improvements for the computation of the depth distribution of such special cases are likely. An interesting question is whether in some of these special cases, one can show that the difficulty of computing the depth distribution is asymptotically equivalent to that of computing the Klee's measure.

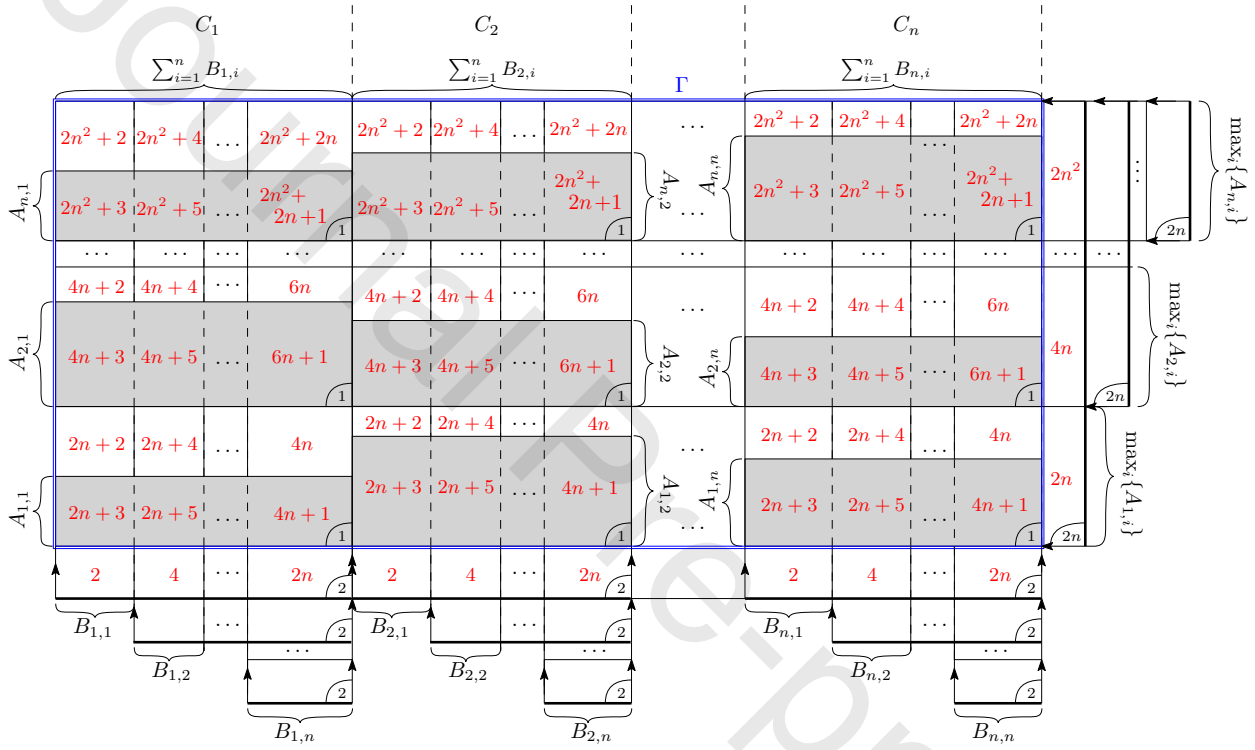


Figure 4: Illustration of an instance of Depth Distribution generated for the product AB. The (red) values inside each rectangular area denote the depth of the respective region they are in. The small arrows indicate that the boxes they delimit span the entire domain in the direction they point to. The small numbers in the corner of each box indicate the number of exact copies of the box added to the instance (intuitively, the weight of the box). The double-lined (blue) box is the domain Γ. Finally, the numbers over curly brackets indicate the length of the region delimited by the brackets.

References

[1] M. Abo Khamis, H. Q. Ngo, C. Ré, and A. Rudra. Joins via geometric resolutions: Worst-case and beyond. In T. Milo and D. Calvanese, editors, *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS), Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 213–228. ACM, 2015. ISBN 978-1-4503-2757-2. doi: 10.1145/2745754.2745776. URL `http://doi.acm.org/10.1145/2745754.2745776`.

[2] P. Afshani. Fast computation of output-sensitive maxima in a word RAM. In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Portland, Oregon, USA, January 5-7, 2014*, pages 1414–1423. SIAM, 2014. ISBN 978-1-61197-338-9, 978-1-61197-340-2. doi: 10.1137/1.9781611973402.104. URL `http://dx.doi.org/10.1137/1.9781611973402.104`.

[3] P. Afshani, L. Arge, and K. G. Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In T. K. Dey and S. Whitesides, editors, *Symposuim on Computational Geometry (SoCG), Chapel Hill, NC, USA, June 17-20, 2012*, pages 323–332. ACM, 2012. ISBN 978-1-4503-1299-8. doi: 10.1145/2261250.2261299. URL `http://doi.acm.org/10.1145/2261250.2261299`.

[4] P. Afshani, J. Barbay, and T. M. Chan. Instance-optimal geometric algorithms. *Journal of the ACM (JACM)*, 64(1):3:1–3:38, Mar. 2017. ISSN 0004-5411. doi: 10.1145/3046673. URL `http://doi.acm.org/10.1145/3046673`.

[5] P. K. Agarwal. An improved algorithm for computing the volume of the union of cubes. In D. G. Kirkpatrick and J. S. B. Mitchell, editors, *Proceedings of the 26th ACM Symposium on Computational Geometry (SoCG), Snowbird, Utah, USA*, pages 230–239. ACM, 2010. ISBN 978-1-4503-0016-2. doi: 10.1145/1810959.1811000. URL `http://doi.acm.org/10.1145/1810959.1811000`.

[6] J. L. Bentley. Algorithms for Klee's rectangle problems. Technical report, Computer Science Department, Carnegie Mellon University, 1977.

[7] K. Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Computational Geometry, Theory and Applications (CGTA)*, 45(5-6):225–233, 2012. doi: 10.1016/j.comgeo.2011.12.001. URL `http://dx.doi.org/10.1016/j.comgeo.2011.12.001`.

[8] K. Bringmann. Bringing order to special cases of Klee's Measure Problem. In K. Chatterjee and J. Sgall, editors, *Mathematical Foundations of Computer Science 2013 - 38th International Symposium (MFCS), Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science (LNCS)*, pages 207–218. Springer, 2013. ISBN 978-3-642-40312-5. doi: 10.1007/978-3-642-40313-2_20. URL `http://dx.doi.org/10.1007/978-3-642-40313-2_20`.

[9] T. M. Chan. A (slightly) faster algorithm for Klee's Measure Problem. In M. Teillaud, editor, *Proceedings of the 24th ACM Symposium on Computational Geometry (SoCG), College Park, MD, USA, June 9-11, 2008*, pages 94–100. ACM, 2008. doi: 10.1145/1377676.1377693. URL `http://doi.acm.org/10.1145/1377676.1377693`.

[10] T. M. Chan. Klee's Measure Problem made easy. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Berkeley, CA, USA, 26-29 October, 2013*, pages 410–419. IEEE Computer Society, 2013. ISBN 978-0-7695-5135-7. doi: 10.1109/FOCS.2013.51. URL `http://dx.doi.org/10.1109/FOCS.2013.51`.

[11] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In A. V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, New York, New York, USA*, pages 1–6. ACM, 1987. ISBN 0-89791-221-7. doi: 10.1145/28395.28396. URL `http://doi.acm.org/10.1145/28395.28396`.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009. ISBN 978-0-262-03384-8. URL http://mitpress.mit.edu/books/introduction-algorithms.

[13] F. d'Amore, V. H. Nguyen, T. Roos, and P. Widmayer. On optimal cuts of hyperrectangles. *Computing*, 55(3):191–206, 1995. doi: 10.1007/BF02238431. URL http://dx.doi.org/10.1007/BF02238431.

[14] H. Edelsbrunner. A new approach to rectangle intersections part i. 13(3-4): 209–219, 1983.

[15] M. L. Fredman and B. W. Weide. On the complexity of computing the measure of $\cup_1^n [a_i, b_i]$. *Communications of the ACM (CACM)*, 21(7):540–544, 1978. doi: 10.1145/359545.359553. URL http://doi.acm.org/10.1145/359545.359553.

[16] M. Fürer. Faster integer multiplication. *SIAM Journal on Computing (SICOMP)*, 39(3):979–1005, 2009. doi: 10.1137/070711761. URL https://doi.org/10.1137/070711761.

[17] F. L. Gall. Powers of tensors and fast matrix multiplication. In K. Nabeshima, K. Nagasaka, F. Winkler, and Á. Szántó, editors, *International Symposium on Symbolic and Algebraic Computation (ISSAC), Kobe, Japan, July 23-25, 2014*, pages 296–303. ACM, 2014. ISBN 978-1-4503-2501-1. doi: 10.1145/2608628.2608664. URL http://doi.acm.org/10.1145/2608628.2608664.

[18] D. Harvey, J. van der Hoeven, and G. Lecerf. Even faster integer multiplication. *Journal of Complexity (JOC)*, 36:1–30, 2016. doi: 10.1016/j.jco.2016.03.001. URL https://doi.org/10.1016/j.jco.2016.03.001.

[19] A. Karatsuba and Y. Ofman. Multiplication of Multidigit Numbers on Automata. *Soviet Physics-Doklady*, 7:595–596, 1963.

[20] D. G. Kirkpatrick and R. Seidel. Output-size sensitive algorithms for finding maximal vectors. In J. O'Rourke, editor, *Proceedings of the First*

Annual Symposium on Computational Geometry (SoCG), Baltimore, Maryland, USA, June 5-7, 1985, pages 89–96. ACM, 1985. ISBN 0-89791-163-6. doi: 10.1145/323233.323246. URL http://doi.acm.org/10.1145/323233.323246.

[21] V. Klee. Can the measure of $\cup_1^n [a_i, b_i]$ be computed in less than $o(n \log n)$ steps? *The American Mathematical Monthly*, 84(4):284–285, 1977.

[22] J. M. Kleinberg and É. Tardos. *Algorithm design*. Addison-Wesley, 2006. ISBN 978-0-321-37291-8.

[23] D. E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 3rd Edition*. Addison-Wesley, 1998. ISBN 0201896842. URL http://www.worldcat.org/oclc/312898417.

[24] D. R. Lick and A. T. White. k-degenerate graphs. *Canadian Journal of Mathematics (CJM)*, 22:1082–1096, 1970.

[25] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, July 1983. ISSN 0004-5411. doi: 10.1145/2402.322385. URL http://doi.acm.org/10.1145/2402.322385.

[26] V. H. Nguyen and P. Widmayer. Binary space partitions for sets of hyperrectangles. In *Algorithms, Concurrency and Knowledge: 1995 Asian Computing Science Conference (ACSC), Pathumthani, Tailand, December 11-13, 1995, Proceedings*, pages 59–72, 1995. doi: 10.1007/3-540-60688-2_35. URL https://doi.org/10.1007/3-540-60688-2_35.

[27] M. H. Overmars and C. Yap. New upper bounds in Klee's measure problem. *SIAM Journal on Computing (SICOMP)*, 20(6):1034–1045, 1991. doi: 10.1137/0220065. URL http://dx.doi.org/10.1137/0220065.

[28] C. H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007. ISBN 978-1-4288-1409-7.

[29] L. Roditty and V. V. Williams. Subquadratic time approximation algorithms for the girth. In Y. Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Kyoto, Japan, January 17-19, 2012*, pages 833–845. SIAM, 2012. ISBN 978-1-61197-210-8. doi: 10.1137/1.9781611973099. URL http://portal.acm.org/citation.cfm?id=2095183&CFID=63838676&CFTOKEN=79617016.

[30] A. Schönhage and V. Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7(3-4):281–292, 1971. doi: 10.1007/BF02242355. URL https://doi.org/10.1007/BF02242355.

[31] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, Aug. 1969. ISSN 0029-599X.

[32] H. Yildiz and S. Suri. On Klee's measure problem for grounded boxes. In T. K. Dey and S. Whitesides, editors, *Symposuim on Computational Geometry (SoCG), Chapel Hill, NC, USA, June 17-20, 2012*, pages 111–120. ACM, 2012. ISBN 978-1-4503-1299-8. doi: 10.1145/2261250.2261267. URL http://doi.acm.org/10.1145/2261250.2261267.

[33] H. Yildiz, J. Hershberger, and S. Suri. A discrete and dynamic version of klee's measure problem. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG), Toronto, Ontario, Canada, August 10-12, 2011*, 2011. URL http://www.cccg.ca/proceedings/2011/papers/paper28.pdf.

[34] H. Yu. An improved combinatorial algorithm for boolean matrix multiplication. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *42nd International Colloquium on Automata, Languages, and Programming (ICALP), Proceedings, Part I, Kyoto, Japan*, volume 9134 of *Lecture Notes in Computer Science*, pages 1094–1105. Springer, 2015. ISBN 978-3-662-47671-0. doi: 10.1007/978-3-662-47672-7_89. URL http://dx.doi.org/10.1007/978-3-662-47672-7_89.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: