Approximating Klee's Measure Problem and a Lower Bound for Union Volume Estimation

Karl Bringmann
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Kasper Green Larsen
Aarhus University, Denmark

André Nusser
Université Côte d'Azur, CNRS, Inria, I3S, Sophia Antipolis, France

Eva Rotenberg
Technical University of Denmark, Kongens Lyngby, Denmark

Yanheng Wang
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Union volume estimation is a classical algorithmic problem. Given a family of objects $O_1, \dots, O_n \subset \mathbb{R}^d$, we want to approximate the volume of their union. In the special case where all objects are boxes (also called hyperrectangles) this is known as Klee's measure problem. The state-of-the-art $(1+\varepsilon)$-approximation algorithm [Karp, Luby, Madras '89] for union volume estimation as well as Klee's measure problem in constant dimension d uses a total of $O(n/\varepsilon^2)$ queries of three types: (i) determine the volume of O_i; (ii) sample a point uniformly at random from O_i; and (iii) ask whether a given point is contained in O_i.

First, we show that if an algorithm learns about the objects only through these types of queries, then $\Omega(n/\varepsilon^2)$ queries are necessary. In this sense, the complexity of [Karp, Luby, Madras '89] is optimal. Our lower bound holds even if the objects are equiponderous axis-aligned polygons in $\mathbb{R}^2$, if the containment query allows arbitrary (not necessarily sampled) points, and if the algorithm can spend arbitrary time and space examining the query responses.

Second, we provide a more efficient approximation algorithm for Klee's measure problem, which improves the running time from $O(n/\varepsilon^2)$ to $O((n+\frac{1}{\varepsilon^2}) \cdot \log^{O(d)}(n))$. We circumvent our lower bound by exploiting the geometry of boxes in various ways: (1) We sort the boxes into classes of similar shapes after inspecting their corner coordinates. (2) With orthogonal range searching, we show how to sample points from the union of boxes in each class, and how to merge samples from different classes. (3) We bound the amount of wasted work by arguing that most pairs of classes have a small intersection.

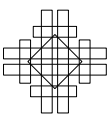
2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases approximation, volume of union, union of objects, query complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.25

Related Version *Full Version*: `https://arxiv.org/abs/2410.00996` [5]

Funding *Karl Bringmann and Yanheng Wang:* This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979).
Kasper Green Larsen: Supported by a DFF Sapere Aude Research Leader Grant No. 9064-00068B.
André Nusser: This work was supported by the French government through the France 2030 investment plan managed by the National Research Agency (ANR), as part of the Initiative of



Excellence of Université Côte d'Azur under reference number ANR-15-IDEX-01. Part of this work was conducted while the author was at BARC, University of Copenhagen, supported by the VILLUM Foundation grant 16582.
Eva Rotenberg: Supported by DFF Grant 2020-2023 (9131-00044B) "Dynamic Network Analysis", the VILLUM Foundation grant VIL37507 "Efficient Recomputations for Changeful Problems" and the Carlsberg Foundation Young Researcher Fellowship CF21-0302 "Graph Algorithms with Geometric Applications".

1 Introduction

We revisit the classical problem of *union volume estimation*: given objects $O_1, \ldots, O_n \subset \mathbb{R}^d$, we want to estimate the volume of $O_1 \cup \ldots \cup O_n$.[1] This problem has several important applications such as DNF counting and network reliability; see the discussion in Section 1.2.

The state-of-the-art solution [19] works in a model where one has access to each input object O_i by three types of queries: (i) determine the volume of the object, (ii) sample a point uniformly at random from the object, and (iii) ask whether a point is contained in the object. Apart from these types of queries, the model allows arbitrary computations. The complexity of algorithms is thus measured by the number of queries.

After Karp and Luby [18] introduced this model, Karp, Luby and Madras [19] gave an algorithm that $(1+\varepsilon)$-approximates the volume of n objects in this model with constant success probability.[2] It uses $O(n/\varepsilon^2)$ queries plus $O(n/\varepsilon^2)$ additional time, which improves earlier algorithms [18, 22], and only asks containment queries of previously sampled points. The last 35 years have seen no algorithmic improvement. Is this classical upper bound best possible? We resolve the question in this work by providing a matching lower bound.

The union volume estimation problem was also studied recently in the streaming setting [25, 23], where a stream of objects $O_1, \ldots, O_n \subset \Omega$ arrive in order. When we are at position i in the stream, we can only query object O_i. This line of work yields algorithms with constant success probability that use $O(\text{polylog}(|\Omega|)/\varepsilon^2)$ queries and additional time per object (and the same bound also holds for space complexity). Summed over all n objects, the bounds match the classical algorithm up to the $\text{polylog}(|\Omega|)$ factor. So, interestingly, even in the streaming setting the same upper bound can be achieved.[3]

The perhaps most famous application of the algorithm by Karp, Luby, and Madras [19] is *Klee's measure problem* [21]. This is a fundamental problem in computational geometry in which we are given n axis-aligned boxes in $\mathbb{R}^d$ and want to compute the volume of their union. An axis-aligned box is a set in the form $[a_1, b_1] \times \ldots \times [a_d, b_d] \subset \mathbb{R}^d$, and the input consists of the corner coordinates $a_1, b_1, \ldots, a_d, b_d$ of each box. A long line of research on this problem and various special cases (e.g., for fixed dimensions or for cubes) [31, 26, 11, 2, 1, 12, 32, 3] lead to an exact algorithm running in time $O(n^{d/2} + n \log n)$ for constant d [13]. A conditional lower bound suggests that any faster algorithm would require fast matrix multiplication techniques [12], but it is unclear how to apply fast matrix multiplication to this problem. On the approximation side, note that the three queries can be implemented in time $O(d)$ for any d-dimensional axis-aligned box. Thus the union volume estimation algorithm can be applied, and it computes a $(1+\varepsilon)$-approximation to Klee's measure problem in time $O(nd/\varepsilon^2)$, as

[1] Technically, the objects need to be measurable. In the most general form, $O_1, \ldots, O_n$ can be any measurable sets in a measure space, and we want to estimate the measure of their union. However, this work only deals with boxes in $\mathbb{R}^d$ (in our algorithm) and polygons in the plane (in our lower bound).
[2] The success probability can be boosted to $1-\delta$, adding a $\log(1/\delta)$ factor in time and query complexity.
[3] See also [30] for earlier work studying Klee's measure problem in the streaming setting.

has been observed in [4]. This direct application of union volume estimation was the state of the art for approximate solutions for Klee's measure problem until our work. See Section 1.2 for interesting applications of Klee's measure problem.

1.1 Our contribution

Lower bound for union volume estimation

Given the state of the art, a natural question is to ask *whether the query complexity of the general union volume estimation algorithm of [19] can be further improved.* Any such improvement would speed up several important applications, cf. Section 1.2. On the other hand, any lower bound showing that the algorithm of [19] is optimal also implies tightness of the known streaming algorithms (up to logarithmic factors), as the streaming algorithms match the static running time bound.

Previously, a folklore lower bound of $\Omega(n + 1/\varepsilon^2)$ was known. But between this and the upper bound $O(n/\varepsilon^2)$ there is still a large gap; consider for example the regime $\varepsilon = 1/\sqrt{n}$ where the bounds are $\Omega(n)$ and $O(n^2)$, respectively. We close this gap by strengthening the lower bound to $\Omega(n/\varepsilon^2)$, thereby also showing optimality of [19]. Note that the lower bound is about query complexity in the model; it does not make any computational assumption. In particular, it holds even if the algorithm spends arbitrary time and space in computation.

▶ **Theorem 1.** *Any algorithm that computes a $(1+\varepsilon)$-approximation to the volume of the union of n objects via volume, sampling and containment queries with success probability at least $2/3$ must make $\Omega(n/\varepsilon^2)$ queries.*

We highlight that our lower bound holds even for equiponderous, axis-aligned polygons in the plane.

Upper bound for Klee's measure problem

Our lower bound for union volume estimation implies that any improved algorithm for Klee's measure problem must exploit the geometric structure of boxes. To this end, we exploit our knowledge of the corner coordinates and classify the boxes by shapes. In addition, we apply the geometric tool of orthogonal range searching. They allow us to approximate Klee's measure problem faster than the previous $O(n/\varepsilon^2)$ algorithm. Here and throughout, we assume that the dimension d is a constant; in particular, we suppress factors of $2^{O(d)}$ in the time complexity.

▶ **Theorem 2.** *There is an algorithm that runs in time $O\left((n + \frac{1}{\varepsilon^2}) \cdot \log^{2d+1}(n)\right)$ and with probability at least 0.9 computes a $(1+\varepsilon)$-approximation for Klee's measure problem.*

This is a strict improvement when $\varepsilon \leq 1/\log^{d+1}(n)$ is small. Consider for example the regime $\varepsilon = 1/\sqrt{n}$: Our algorithm runs in near-linear time, whereas the previous algorithm runs in quadratic time. We remark that the success probability can be boosted to any $1 - \delta$ by standard techniques which incurs an extra $\log(1/\delta)$ factor in the running time. Finally, we highlight that the core of our algorithm is an efficient method to sample uniformly and independently with a given density from the union of the input boxes. This can be of independent interest outside the context of volume estimation.

1.2 Related work

A major application of union volume estimation is *DNF counting*, in which we are given a Boolean formula in disjunctive normal form and want to count the number of satisfying assignments. Computing the exact number of satisfying assignments is #P-complete, so it likely requires superpolynomial time. Approximating the number of satisfying assignments can be achieved by a direct application of union volume estimation, as described in [19]. Their algorithm remains the state of the art for approximate DNF counting, see e.g. [24]. This has been extended to more general model counting [27, 9, 24], probabilistic databases [20, 15, 28], and probabilistic queries on databases [7].

We also mention *network reliability* as another application for union volume estimation, which was already discussed in [19]. Additionally, Karger's famous paper on the problem [17] uses the algorithm of [19] as a subroutine. However, the current state-of-the-art algorithms no longer use union volume estimation as a tool [8].

Finally, we want to draw a connection to the following well-known query sampling bound. Canetti, Even, and Goldreich [6] showed that approximating the mean of a random variable whose codomain is the unit interval requires $\Omega(1/\varepsilon^2)$ queries, thus obtaining tight bounds for the sampling complexity of the mean estimation problem. Their bound generalize to $\Omega(1/(\mu\varepsilon^2))$ on the number of queries needed to estimate the mean μ of a random variable in general. Before our work it was thus natural to expect that the $1/\varepsilon^2$ dependence in the number of queries for union volume estimation is optimal. However, whether the factor n is necessary, or the number of queries could be improved to, say, $O(n + 1/\varepsilon^2)$, was open to the best of our knowledge.

Klee's measure problem is an important problem in computational geometry. One reason for its significance is that techniques developed for it can often be adapted to solve various related problems, such as the depth problem (given a set of boxes, what is the largest number of boxes that can be stabbed by a single point?) [13] or Hausdorff distance under translation in L_∞ [14]. Moreover, various other problems can be reduced to Klee's measure problem or to its related problems. For example, deciding whether a set of boxes covers its boundary box can be reduced to Klee's measure problem [13]. The continuous k-center problem on graphs (i.e., finding centers that can lie on the edges of a graph that cover the vertices of a graph) is reducible to Klee's measure problem as well [29]. Finding the smallest hypercube containing at least k points among n given points can be reduced to the depth problem [16, 10, 13]. In light of this, it would be interesting to see if our approximation techniques generalize to any of these related problems.

2 Technical overview

We now sketch our upper and lower bounds at an intuitive level. The formal arguments will be presented in Section 3 and Section 4, respectively. In this paper we denote $[n] := \{1, 2 \dots, n\}$.

2.1 Upper bound for Klee's measure problem

Due to our lower bound, we have to exploit the structure of the input boxes to obtain a running time of the form $O\left((n + \frac{1}{\varepsilon^2}) \cdot \text{polylog}(n)\right)$. We follow the common algorithmic approach of sampling. Specifically, we aim to sample a set S from the union of boxes such that each point is selected with probability density p. For appropriately defined p, the value $|S|/p$ is a good estimate of the volume of the union. In the overview we assume that a proper p is given and focus on the main difficulty: creating a sample set S for the given p.

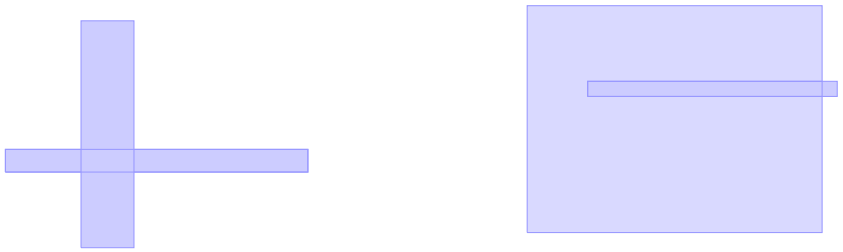


Figure 1 When the side lengths of two boxes differ a lot in at least one of their dimensions (in our examples, the y-axis), their intersection is small compared to their union.

We start by sorting the input boxes into *classes* by shape. Two boxes are in the same class if, in each dimension $k \in [d]$, their side lengths are both in $[2^{L_k}, 2^{L_k+1})$ for some $L_k \in \mathbb{Z}$. We call two classes *similar* if the corresponding side lengths are polynomially related (e.g., within a factor of n^4) in each dimension. We make two crucial observations:

1. Each class is similar to only polylogarithmically many classes (Observation 11).
2. Dissimilar classes have a small intersection compared to their union (Observation 12, Figure 1).

Our algorithm continues as follows. We scan through the classes in an arbitrary order. For each class, we sample with density p from the union of the boxes in this class, but we only keep a point if it is not contained in any class that comes later in the order. To efficiently test containment in a later class, we use an orthogonal range searching data structure (with an additional dimension for the index of the class). When the scan finishes, we obtain a desired sample set S.

Let us elaborate why the algorithm is efficient.

Sampling from a single class. Our approach is simple yet powerful: (i) grid the space into cells of side lengths comparable to the boxes in this class; (ii) sample points from the relevant cells uniformly at random; (iii) discard points outside the union by querying an orthogonal range searching data structure. See Figure 2. All these steps exploit the geometry of boxes. Since the grid cells and boxes have roughly the same shape, a significant fraction of the points sampled in (ii) are contained in the union, i.e., not discarded in (iii). The orthogonal range searching data structure allows us to quickly decide which points to discard. These explain the efficiency of sampling from a class.

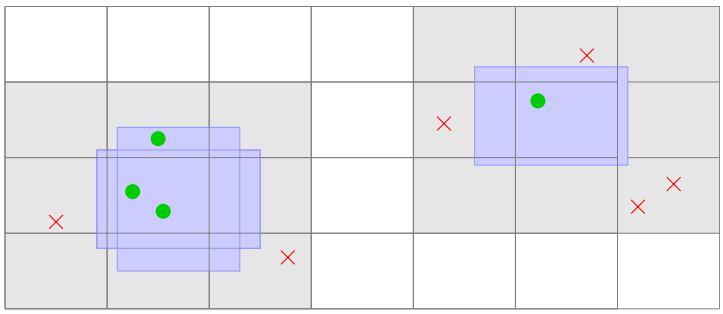


Figure 2 We sample points in the grid cells $\mathcal{G}$ that are intersected by a box $\mathcal{O}_i$ from a fixed class. We then use orthogonal range searching to determine whether a sampled point is in a box from the class and should be kept (•), or is not and should be discarded (×).

Bounding the amount of wasted work. Sampled points that appear in later classes also need to be discarded, and this is a potential (and only) source of inefficiency. So we need to bound the number of them. Such a point shows up later either (i) in a similar class, or (ii) in a dissimilar class. Our first observation stated that a class is similar to at most polylogarithmically many classes. So (i) can happen up to polylogarithmically many times from the perspective of a fixed point in the union. On the other hand, our second observation stated that the intersection of dissimilar classes is small. So on average (ii) rarely happens and does not affect the expected running time significantly.

2.2 Lower bound for union volume estimation

Our lower bound is shown by a reduction from the Query-Gap-Hamming problem: Given two vectors $x, y \in \{-1, +1\}^\ell$, distinguish whether their inner product is greater than $\sqrt{\ell}$ or less than $-\sqrt{\ell}$. It is known that any algorithm distinguishing these two cases with success probability at least $2/3$ must access $\Omega(\ell)$ bits of x and y.

We first sketch why $\Omega(1/\varepsilon^2)$ queries are necessary to $(1+\varepsilon)$-approximate the volume of the union of two objects in the query model. Given a Query-Gap-Hamming instance x, y, we construct two objects $X = \{(j, x_j) : j \in [\ell]\}$ and $Y = \{(j, y_j) : j \in [\ell]\}$. These are discrete point sets, but they can be inflated to polygons such that cardinality corresponds to volume. See Figure 3 for an example. Note that for all $k \in \{0, \dots, \ell\}$, we have

$$|X \cup Y| = \ell + k \iff \langle x, y\rangle = \ell - 2k.$$

If an algorithm $\mathcal{A}$ can $(1+\varepsilon)$-approximate $|X \cup Y|$, then it can approximate k within additive error $2\varepsilon\ell$, thus it can also approximate $\langle x, y\rangle$ within additive error $4\varepsilon\ell$. Setting $\ell = 1/(16\varepsilon^2)$, the additive error is at most $4\varepsilon\ell = 4 \cdot \frac{1}{4\sqrt{\ell}} \cdot \ell = \sqrt{\ell}$, which suffices to distinguish $\langle x, y\rangle = \sqrt{\ell}$ from $\langle x, y\rangle = -\sqrt{\ell}$ and thereby solves the Query-Gap-Hamming instance.

Note that $|X| = |Y| = \ell$, so a volume query does not disclose any information about x and y. Each sample or containment query accesses at most one bit of x or y. Therefore, algorithm $\mathcal{A}$ has to make $\Omega(\ell) = \Omega(1/\varepsilon^2)$ queries to X, Y.

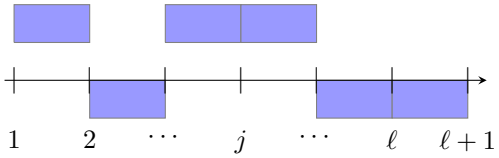


Figure 3 The vector $x = (+1, -1, +1, +1, -1, -1)$ represented as the set $\{(j, x_j) : j \in [6]\}$, where each point is drawn as a rectangle.

In order to generalize this lower bound for the union of two objects to an $\Omega(n/\varepsilon^2)$ lower bound for the union of n objects, we need to ensure that each query gives away only $O(1/n)$ bits of information about x and y. We apply two obfuscations that jointly slow down the exposure of bits; see Figure 4. Firstly, we introduce objects $X_1, \dots, X_n$ whose union is X and objects $Y_1, \dots, Y_n$ whose union is Y. Imagine cutting each X-induced rectangle in Figure 3 into n side-by-side pieces and distributing them randomly among $X_1, \dots, X_n$. Do the same for Y. The idea is that one needs to make $\Omega(n)$ containment queries on the rectangular region in order to hit the correct piece; only then is the corresponding bit revealed. Secondly, we introduce a large band shared by all X_i and Y_i for $i \in [n]$. In Figure 4, this is the long dark-blue rectangle that spans from left to right. Intuitively it enforces $\Omega(n)$ sample queries to obtain a single point that contains any information about x and y.

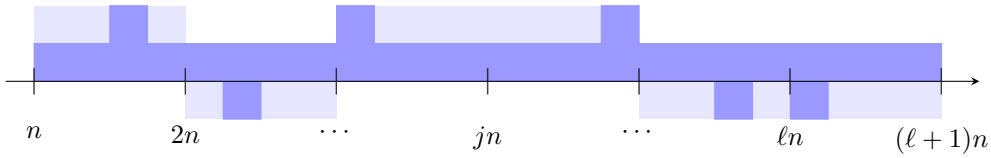


Figure 4 The vector y or $x = (+1, -1, +1, +1, -1, -1)$ gives rise to n polygons; one of these polygons is illustrated in dark blue. The light blue area indicates the union of all these n polygons.

3 Approximation algorithm for Klee's measure problem

In this section, we present our approximation algorithm for Klee's measure problem in constant dimension d:

▶ **Theorem 2.** *There is an algorithm that runs in time* $O\left((n + \frac{1}{\varepsilon^2}) \cdot \log^{2d+1}(n)\right)$ *and with probability at least* 0.9 *computes a* $(1+\varepsilon)$*-approximation for Klee's measure problem.*

Lemmas marked with $\star$ are folklore or straightforward. Their proofs can be found in the full version of this paper [5].

3.1 Preliminaries

In Klee's measure problem we are given *boxes* $O_1, \ldots, O_n \subset \mathbb{R}^d$. Here, a box is an object of the form $O_i = [a_1, b_1] \times \ldots \times [a_d, b_d]$, and as input we are given the coordinates $a_1, b_1, \ldots, a_d, b_d$ of each box. Based on these, it is easy to compute the side lengths and volume of each box.

Throughout we write $V := \text{Volume}(\bigcup_{i=1}^n O_i)$ for the volume of the union of boxes. The goal is to approximate V up to a factor of $1+\varepsilon$. Our approach is based on sampling, so now let us introduce the relevant notions.

Recall the Poisson distribution $\text{Pois}(\lambda)$ with mean and variance λ: It captures the number of active points in a universe, under the assumption that active points occur uniformly and independently at random across the universe, and that λ points are active on average. The following definition is usually referred to as a homogeneous *Poisson point process* at rate p. Intuitively, we activate each point in some universe $U \subset \mathbb{R}^d$ independently with "probability density" p, thus the number of activated points follows the Poisson distribution with mean $p \cdot \text{Volume}(U)$.

▶ **Definition 3** (p-sample)**.** *Let* $U \subset \mathbb{R}^d$ *be a measurable set, and let* $p \in [0,1]$*. We say that a random subset* $S \subseteq U$ *is a* p-sample of U *if for any measurable* $U' \subseteq U$ *we have that* $|S \cap U'| \sim \text{Pois}(p \cdot \text{Volume}(U'))$.

In particular, if S is a p-sample of U, then $|S| \sim \text{Pois}(p \cdot \text{Volume}(U))$. Two more useful properties follow from the definition:

(i) For any measurable subset $U' \subseteq U$, the restriction $S \cap U'$ is a p-sample of U'.

(ii) The union of p-samples of two disjoint sets U, U' is a p-sample of $U \cup U'$.

Besides sampling, we will apply orthogonal range searching to handle the so-called APPEARS(x, i) query: Given $x \in \mathbb{R}^d$ and $i \in \mathbb{N}$, is $x \in O_i \cup \cdots \cup O_n$?

▶ **Lemma 4** ($\star$)**.** *We can build a data structure in* $O(n \log^{d+1}(n))$ *time that answers* APPEARS(x, i) *queries in* $O(\log^{d+1}(n))$ *time.*

For our algorithm to work, we also need a constant-factor approximation of the volume V. It is known that this can be computed in $O(n)$ time [19]. In order to stay simple and self-contained, we state a weaker result by implementing the Karp-Luby algorithm [18] with the help of APPEARS queries.

▶ **Lemma 5** (⋆, adapted from [18]). *Given the data structure from Lemma 4, there exists an algorithm that computes in $O(n \log^{d+1}(n))$ time a 2-approximation to V with probability at least 0.9.*

3.2 Classifying boxes by shapes

As our first step in the algorithm, we classify boxes by their shapes.

▶ **Definition 6.** *Let $L_1, \ldots, L_d \in \mathbb{Z}$. We say that a box $O \subset \mathbb{R}^d$ is of type $(L_1, \ldots, L_d)$ if its side length in dimension k is contained in $[2^{L_k}, 2^{L_k+1})$, for all $k \in [d]$.*

Using this definition, we partition the input boxes $O_1, \ldots, O_n$ into *classes* $C_1, \ldots, C_m$ such that each class corresponds to one type of boxes. The notation is fixed from now on. For each $t \in [m]$, we denote $U_t := \bigcup_{O \in C_t} O \subset \mathbb{R}^d$, namely the union of boxes in class C_t.

Similar to APPEARS, we may use orthogonal range searching to handle the so-called INCLASS(x, t) query: Is a given point $x \in \mathbb{R}^d$ contained in U_t?

▶ **Lemma 7** (⋆). *We can build a data structure in $O(n \log^{d+1}(n))$ time that answers INCLASS(x, t) queries in $O(\log^{d+1}(n))$ time.*

Sampling from a class

The next lemma shows that we can p-sample from any U_t efficiently by rejection sampling.

▶ **Lemma 8.** *Given $t \in [m]$, $p \in [0, 1]$ and the data structure from Lemma 7, one can generate a p-sample of U_t in expected time $O(|C_t| \log |C_t| + p \cdot \text{Volume}(U_t) \cdot \log^{d+1}(n))$.*

Proof. Let $(L_1, \ldots, L_d)$ be the type of class C_t. We subdivide $\mathbb{R}^d$ into the grid

$$\mathcal{G}_\infty := \{[i_1 \, 2^{L_1}, (i_1 + 1) \, 2^{L_1}) \times \cdots \times [i_d \, 2^{L_d}, (i_d + 1) \, 2^{L_d}) \mid i_1, \ldots, i_d \in \mathbb{Z}\}.$$

We call each element of $\mathcal{G}_\infty$ a *cell*. Let $\mathcal{G} := \{G \in \mathcal{G}_\infty \mid G \cap U_t \neq \emptyset\}$ be the set of cells that intersect with U_t. Let $U := \bigcup_{G \in \mathcal{G}} G$.

First we create a p-sample S of U as follows. Generate $K \sim \text{Pois}(p \cdot \text{Volume}(U))$, the number of points we are going to include. Then sample K points uniformly at random from U by repeating the following step K times: Select a cell $G \in \mathcal{G}$ uniformly at random and then sample a point from G uniformly at random. The sampled points constitute our set S.

Next we compute $S \cap U_t$: For each $x \in S$, we query INCLASS(x, t); if the answer is true then we keep x, otherwise we discard it. The resulting set $S \cap U_t$ is a p-sample of U_t, since restricting to a fixed subset preserves the p-sample property.

Before we analyze the running time, we show that U_t makes up a decent proportion of U. Recall that every box in class C_t is of type $(L_1, \ldots, L_d)$. Consider the projection to any dimension $k \in [d]$. Each projected box from C_t can intersect at most three projected cells from $\mathcal{G}$. So each box from C_t intersects at most 3^d cells from $\mathcal{G}$, implying that $|\mathcal{G}| \leq 3^d \, |C_t|$. Moreover, since the volume of any cell is at most the volume of a box in C_t, we have $\text{Volume}(U) \leq 3^d \, \text{Volume}(U_t)$.

Recall that we assume d to be constant and hence $3^d = O(1)$. The computation of $\mathcal{G}$ takes $O(|\mathcal{G}| \log |\mathcal{G}|) \subseteq O(|C_t| \log |C_t|)$ time. The remaining time is dominated by the inClass queries. The expected size of S is $p \cdot \text{Volume}(U) \leq 3^d\, p\, \text{Volume}(U_t)$. As we query inClass once for each point of S, the total expected time is $O(p \cdot \text{Volume}(U_t) \cdot \log^{d+1}(n))$ by Lemma 7. ◀

Classes do not overlap much

We show the following interesting property of classes, that the sum of their volumes is within a polylogarithmic factor of the total volume V.

▶ **Lemma 9.** *We have* $\sum_{t=1}^{m} \text{Volume}(U_t) \leq 2^{3d+1} \log^d(n) \cdot V$.

We later exploit it to draw p-samples from $\bigcup_{i=1}^{n} O_i = \bigcup_{t=1}^{m} U_t$ efficiently. Towards proving the lemma, let us collect some simple observations.

▶ **Definition 10.** *We say that a class of type* $(L_1, \ldots, L_d)$ *is* similar *to a class of type* $(L'_1, \ldots, L'_d)$ *if* $2^{|L_k - L'_k|} < n^4$ *for all* $k \in [d]$. *Otherwise they are said to be* dissimilar.

▶ **Observation 11.** *Every class is similar to at most* $8^d \log^d(n)$ *classes.*

Proof. Fix a type $(L_1, \ldots, L_d)$. For each $k \in [d]$, there are at most $8 \log n$ many integers L'_k such that $2^{|L_k - L'_k|} < n^4$. ◀

▶ **Observation 12.** *If two boxes* O, O' *are in dissimilar classes, then* $\text{Volume}(O \cap O') \leq 2V/n^4$.

Proof. Let $(L_1, \ldots, L_d)$ be the type of O, and $(L'_1, \ldots, L'_d)$ be the type of O'. Since the boxes belong to dissimilar classes, there is a dimension $k \in [d]$ such that $2^{|L_k - L'_k|} \geq n^4$. Without loss of generality, assume $2^{L_k - L'_k} \geq n^4$; the other case is symmetric. Let $[a_k, b_k]$ and $[a'_k, b'_k]$ be the projections of O and O' onto dimension k, respectively. Note that $b_k - a_k \in [2^{L_k}, 2^{L_k+1})$ and $b'_k - a'_k \in [2^{L'_k}, 2^{L'_k+1})$. So we have $\frac{b_k - a_k}{b'_k - a'_k} \geq 2^{L_k - (L'_k+1)} \geq n^4/2$. In other words, at most a $2/n^4$ fraction of the interval $[a_k, b_k]$ intersects the interval $[a'_k, b'_k]$. Hence, $\text{Volume}(O \cap O') \leq \text{Volume}(O) \cdot 2/n^4 \leq 2V/n^4$. ◀

Proof of Lemma 9. Without loss of generality assume $\text{Volume}(U_1) \geq \cdots \geq \text{Volume}(U_m)$. We construct a set of indices $T \subseteq [m]$ by the following procedure:

- Initialize $T = \emptyset$.
- For $t = 1, \ldots, m$, if C_t and C_s are dissimilar for all $s \in T$, then add t to T.

For $t \in [m]$, we have $t \notin T$ only if there exists an $s \in T$ such that C_s, C_t are similar and $\text{Volume}(U_s) \geq \text{Volume}(U_t)$; we call s a *witness* of t. If multiple witnesses exist, then we pick an arbitrary one. Conversely, every $s \in T$ can be a witness at most $8^d \log^d(n)$ times by Observation 11. Hence,

$$\sum_{t=1}^{m} \text{Volume}(U_t) \leq 8^d \log^d(n) \cdot \sum_{t \in T} \text{Volume}(U_t). \tag{1}$$

It remains to bound $\sum_{t \in T} \text{Volume}(U_t)$. Consider any distinct $s, t \in T$. By construction, C_s and C_t are dissimilar; and each class contains at most n boxes. So $\text{Volume}(U_s \cap U_t) \leq n^2 \cdot (2V/n^4) = 2V/n^2$ by Observation 12. Using this and inclusion-exclusion, we bound

$$\begin{aligned} \sum_{t \in T} \text{Volume}(U_t) \;&\leq\; \text{Volume}\left(\bigcup_{t \in T} U_t\right) + \sum_{\{s,t\} \subseteq T} \text{Volume}(U_s \cap U_t) \\ &\leq\; V + \binom{m}{2} \frac{2V}{n^2} \;\leq\; 2V. \end{aligned}$$

Plugging this into the right-hand side of Expression (1), we obtain the lemma statement. ◀

3.3 Joining the classes

Recall that $C_1, \ldots, C_m$ are the classes of the input boxes and $U_1, \ldots, U_m$ their respective unions. Assume without loss of generality that the boxes are ordered in accordance with the class ordering, that is, $C_1 = \{O_1, \cdots, O_{i_1}\}$ form the first class, $C_2 = \{O_{i_1+1}, \cdots, O_{i_2}\}$ form the second class, and so on. In general, we assume that $C_t = \{O_{i_{t-1}+1}, \ldots, O_{i_t}\}$ for all $t \in [m]$, where $0 = i_0 < i_1 < \ldots < i_m = n$.

Let $D_t := U_t \setminus (\bigcup_{s=t+1}^{m} U_s)$ be the points in U_t that are not contained in later classes. Note that $D_1, \ldots, D_m$ is a partition of $\bigcup_{t=1}^{m} U_t = \bigcup_{i=1}^{n} O_i$. Hence, to generate a p-sample of $\bigcup_{i=1}^{n} O_i$, it suffices to draw p-samples from each D_t and then take their union.[4] To this end, we draw a p-sample S_t from U_t via Lemma 8. Then we remove all $x \in S_t$ for which $\textsc{appears}(x, i_t + 1) = \text{true}$; these are exactly the points that appear in a later class. What remains is a p-sample of D_t. The union of these sets thus is a p-sample of $\bigcup_{i=1}^{n} O_i$, and we can use the size of this p-sample to estimate the volume V of $\bigcup_{i=1}^{n} O_i$. The complete algorithm is summarized in Algorithm 1.

Algorithm 1 Approximation of Klee's measure.

1. Partition the boxes into classes $C_1, \ldots, C_m$. Relabel the boxes so that their indices are in accordance with the class ordering, i.e., $C_t = \{O_{i_{t-1}+1}, \ldots, O_{i_t}\}$ for all $t \in [m]$.
2. Build the data structures from Lemmas 4 and 7.
3. Obtain a crude estimate $\widetilde{V}$ by Lemma 5. Set $p := 8/(\varepsilon^2 \widetilde{V})$.
4. For $t = 1, \ldots, m$ do:
   - Draw a p-sample S_t from the union $U_t = \bigcup_{O \in C_t} O$ via Lemma 8.
   - Compute $|S'_t|$ where $S'_t := \{x \in S_t : \textsc{appears}(x, i_t + 1) = \text{false}\}$.
5. Output $\sum_{t=1}^{m} |S'_t|/p$.

▶ **Lemma 13.** *Conditioned on $\widetilde{V} \le 2V$, Algorithm 1 outputs a $(1+\varepsilon)$-approximation to V with probability at least $3/4$.*

Proof. Note that for all $t \in [m]$, the set S'_t is a p-sample of D_t. Since $D_1, \ldots, D_m$ partition $\bigcup_{t=1}^{m} U_t = \bigcup_{i=1}^{n} O_i$, their union $\bigcup_{t=1}^{m} S'_t$ is a p-sample of $\bigcup_{i=1}^{n} O_i$. It follows that $N := \sum_{t=1}^{m} |S'_t| \sim \text{Pois}(pV)$. The expectation and variance of N are $pV = 8V/(\varepsilon^2 \widetilde{V}) \ge 4/\varepsilon^2$. So by Chebyshev,

$$\Pr(|N - pV| > \varepsilon pV) \le \frac{\text{Var}[N]}{(\varepsilon pV)^2} \le \frac{1}{4}.$$

Hence, with probability at least $3/4$, the output N/p is a $(1+\varepsilon)$-approximation to V. ◀

▶ **Lemma 14.** *Conditioned on $\widetilde{V} \ge \frac{V}{2}$, Algorithm 1 runs in time $O\left((n + \frac{1}{\varepsilon^2}) \cdot \log^{2d+1}(n)\right)$ in expectation.*

Proof. Step 1 takes $O(n \log n)$ time: we can compute the side lengths of each box, determine its class, then sort the boxes by class index. Step 2 takes $O(n \log^{d+1}(n))$ time by Lemmas 4 and 7. Step 3 takes $O(n \log^{d+1}(n))$ time by Lemma 5.

[4] This idea has previously been used on objects, by considering the difference $D'_i := O_i \setminus (\bigcup_{j=i+1}^{n} O_j)$ [18, 25], while we use this idea on classes.

In step 4, iteration t, sampling S_t costs expected time $O((i_t - i_{t-1})\log(i_t - i_{t-1}) + p\,\text{Volume}(U_t)\cdot\log^{d+1}(n))$ by Lemma 8, and computing S'_t costs expected time $O((1 + p\,\text{Volume}(U_t))\cdot\log^{d+1}(n))$ by Lemma 4. Over all iterations, the expected running time is

$$O\left(\log^{d+1}(n)\cdot\left(n + p\sum_{t=1}^{m}\text{Volume}(U_t)\right)\right).$$

Substituting $p = 8/(\varepsilon^2\tilde{V}) \le 16/(\varepsilon^2 V)$ and applying Lemma 9, we can bound

$$p\sum_{t=1}^{m}\text{Volume}(U_t) \le \frac{16}{\varepsilon^2 V}\sum_{t=1}^{m}\text{Volume}(U_t) \le \frac{2^{3d+5}\log^d(n)}{\varepsilon^2}.$$

Hence the expected running time of step 4 is $O\left(\log^{2d+1}(n)\cdot(n + \frac{1}{\varepsilon^2})\right)$.

Finally, step 5 takes $O(n)$ time. ◀

Proof of Theorem 2. Run Algorithm 1 with a time budget tenfold the bound in Lemma 14; abort the algorithm as soon as it spends more time than the budget allows. So the stated time bound is clearly satisfied. Now consider three bad events:

- $\widetilde{V} \notin [\frac{V}{2}, 2V]$.
- $\widetilde{V} \in [\frac{V}{2}, 2V]$, but the algorithm is aborted.
- $\widetilde{V} \in [\frac{V}{2}, 2V]$ and the algorithm is not aborted, but it does not output a $(1+\varepsilon)$-approximation to V.

By Lemma 5, the first event happens with probability at most 0.1. By Lemma 14 and Markov's inequality, the second event happens with probability at most 0.1. Lastly, by Lemma 13, the third event happens with probability at most 1/4. So the total error probability is at most $0.1 + 0.1 + \frac{1}{4} = \frac{9}{20}$. If none of the bad events happen, then the algorithm correctly outputs a $(1+\varepsilon)$-approximation to V. The success probability of $\frac{11}{20}$ can be boosted to, say, 0.9 by returning the median of a sufficiently large constant number of repetitions. ◀

4 Lower bound for union volume estimation

In this section, any randomized algorithm or protocol is assumed to have success probability at least 2/3. We consider the discrete version of union volume estimation in two dimensions, where each object O_i is a finite subset of the integer lattice $\mathbb{Z}^2$. The goal is to $(1+\varepsilon)$-approximate the cardinality $|O_1 \cup \ldots \cup O_n|$ of their union. The algorithm does not have direct access to the objects, but it can ask three forms of queries:

- Volume(O_i): Return the cardinality $|O_i|$.
- Sample(O_i): Draw a uniform random point from O_i.
- Contains($(a,b), O_i$): Given a point $(a,b) \in \mathbb{Z}^2$, return whether $(a,b) \in O_i$ or not.

We aim to prove that $\Omega(n/\varepsilon^2)$ queries are necessary. Later in Section 5 we translate this to a lower bound in the continuous space $\mathbb{R}^2$, thereby proving Theorem 1.

Our starting point is the Query-Gap-Hamming problem: The inputs are two (hidden) vectors $x, y \in \{-1,1\}^\ell$ and we can access one bit of x or y of our choice at a time. The goal is to distinguish the cases $\langle x, y\rangle \ge \sqrt{\ell}$ and $\langle x, y\rangle \le -\sqrt{\ell}$ using as few accesses as possible. The following lemma is folklore; its proof can be found in the full version of this paper [5].

▶ **Lemma 15 (⋆).** *There exists a constant $\delta \in (0,1)$ with the following property. For every $\ell \in \mathbb{N}$, any randomized algorithm for Query-Gap-Hamming on vectors of length ℓ needs at least $\delta\ell$ accesses in expectation.*

From now on, let us fix the constant $\delta \in (0,1)$ guaranteed by Lemma 15. Let $n \in \mathbb{N}$, $\varepsilon \in (0,1)$ be arbitrary and define $\ell := \frac{1}{36\varepsilon^2}$. We reduce Query-Gap-Hamming on vectors of length ℓ to estimating the cardinality of the union of $2n$ objects in $\mathbb{Z}^2$.

The reduction

From the hidden vectors $x, y \in \{-1,1\}^\ell$ we construct $2n$ objects $X_1, \ldots, X_n, Y_1, \ldots, Y_n \subset \mathbb{Z}^2$. Let $R := \{(n+1, 0), \ldots, (n\ell + n, 0)\}$. Take independent random permutations $\pi_1, \ldots, \pi_\ell$ of $[n]$ and define

$$X_i := R \cup \{(jn + \pi_j(i), x_j) : j \in [\ell]\}$$

for $i \in [n]$. Then take another set of independent random permutations $\tau_1, \ldots, \tau_\ell$ and define

$$Y_i := R \cup \{(jn + \tau_j(i), y_j) : j \in [\ell]\}$$

for $i \in [n]$. Note that R is a subset of all X_i and Y_i.

How is $\langle x, y\rangle$ related to the cardinality of the union $|X_1 \cup \cdots \cup X_n \cup Y_1 \cup \cdots \cup Y_n|$? Consider an arbitrary index $j \in [\ell]$. If $x_j = y_j$ then the point sets $\{(jn + \pi_j(i), x_j) : i \in [n]\}$ and $\{(jn + \tau_j(i), y_j) : i \in [n]\}$ are equal (regardless of the concrete permutations), so they together contribute n to the cardinality of the union. On the other hand, if $x_j \neq y_j$ then they are disjoint and thus contribute $2n$. Furthermore, the point set R is contained in all objects and contributes $n\ell$. Hence, the cardinality of the union is equal to

$$n\ell + \sum_{j:x_j=y_j} n + \sum_{j:x_j\neq y_j} 2n = \frac{5}{2}n\ell - \frac{1}{2}n \cdot \left(\sum_{j:x_j=y_j} 1 + \sum_{j:x_j\neq y_j} (-1) \right) = \frac{5}{2}n\ell - \frac{1}{2}n\langle x, y\rangle.$$

Suppose we get a $(1+\varepsilon)$-approximation ρ to the cardinality of the union, then

$$\rho \in \left[(1-\varepsilon)(\tfrac{5}{2}n\ell - \tfrac{1}{2}n\langle x, y\rangle),\ (1+\varepsilon)(\tfrac{5}{2}n\ell - \tfrac{1}{2}n\langle x, y\rangle) \right].$$

Since $|\langle x, y\rangle| \leq \ell$, by computing $(\frac{5}{2}n\ell - \rho) \cdot \frac{2}{n}$ we obtain a value in $\langle x, y\rangle \pm 6\varepsilon\ell$, that is, an additive $6\varepsilon\ell$ approximation to $\langle x, y\rangle$. Our choice of $\varepsilon = \frac{1}{6\sqrt{\ell}}$ allows to distinguish $\langle x, y\rangle \geq \sqrt{\ell}$ from $\langle x, y\rangle \leq -\sqrt{\ell}$.

The reduction is not yet complete. The catch is that an algorithm for union volume estimation can only learn about the objects via queries. It is left to *us* to answer those queries. In Algorithm 2 we describe how to answer queries on object X_i. Queries on object Y_i can be handled similarly by replacing π_j with τ_j and x_j with y_j. The correctness is easy to verify, and this completes the reduction. So far, the argument works for arbitrary permutations; randomness becomes relevant when we bound the number of bit accesses.

Algorithm 2 Answering queries.

- For query Volume(X_i): Return $(n+1)\ell$.
- For query Sample(X_i): Draw a random block $j \in [\ell]$ and a random shift $s \in [n]$. With probability $1 - \frac{1}{n+1}$ return $(jn + s, 0)$. Otherwise return $(jn + \pi_j(i), x_j)$.
- For query Contains($(a,b), X_i$): If $(a,b) \in R$ then return true. Else, compute $j := \lfloor a/n \rfloor$ and $s := a - jn$. If $j \notin [\ell]$ or $\pi_j(i) \neq s$ then return false. Otherwise, if $x_i = b$ then return true; else return false.

Bounding the number of accesses

Now for the sake of contradiction, suppose there exists a randomized algorithm $\mathcal{A}$ that $(1+\varepsilon)$-approximates the cardinality of the union of any $2n$ objects in $\mathbb{Z}^2$, using less than $\frac{\delta}{360\cdot 2^{9/\delta}} \cdot \frac{n}{\varepsilon^2}$ queries. We run $\mathcal{A}$ on the objects defined earlier (using Algorithm 2 to handle its queries). From the output, we can distinguish $\langle x, y\rangle \geq \sqrt{\ell}$ and $\langle x, y\rangle \leq -\sqrt{\ell}$. We claim that less than $\delta\ell$ bits of x, y are accessed from the side of Algorithm 2.

To this end, note that accesses to x, y only happen in the "otherwise" clauses of Sample() and Contains() in Algorithm 2. For analysis we isolate the following mini game:

▶ **Lemma 16.** *Consider a game of $m := \lfloor 2^{-9/\delta} n \rfloor$ rounds. At the start, Merlin generates a secret random permutation π of $[n]$. In each round, Arthur adaptively sends one of the following two requests to Merlin:*

- Random()*: Answer "yes" with probability $\frac{1}{n+1}$ and "no" with the remaining probability.*
- Probe(i, s)*: Answer "yes" if $\pi(i) = s$ and "no" otherwise.*

No matter what strategy Arthur employs, the probability that Merlin ever answers "yes" in the game is at most $2\delta/5$.

Proof. Let E be the event that Merlin ever answers "yes" *and* the first one was due to a Random() request. Let F be the event that Merlin ever answers "yes" *and* the first one was due to a Probe() request. We will bound $\Pr(E)$ and $\Pr(F)$ separately.

To bound $\Pr(E)$, define events $E_1, \ldots, E_m$ where E_t indicates that Arthur's t-th request is Random() and the answer is "yes". Clearly $\Pr(E_t) \leq \frac{1}{n+1}$, thus $\Pr(E) \leq \sum_{t=1}^{m} \Pr(E_t) \leq \frac{m}{n+1} < 2^{-9/\delta} < \delta/15$.

To bound $\Pr(F)$ we use an entropy argument. In particular, we will describe and analyze an encoding scheme for a random permutation π of $[n]$.

The encoder receives π and acts as Merlin in the game, using π as the secret permutation. It interacts with Arthur till the end of the game. If F does not happen then the encoding is simply a bit `0` followed by π. On the other hand, if F happens, then consider the first round $T \in [m]$ when "yes" appeared; denote that request by Probe(i, s). The encoding starts with a bit `1`, then the number T, and finishes with the ordering π' of $[n] \setminus \{i\}$ induced by π.

The decoder receives the encoding and tries to reconstruct π. First it reads the leading bit of the encoding. If the bit is `0` then π immediately follows. If the bit is `1` then the decoder recovers the number T. It pretends to be Merlin and starts a game with Arthur, using the same random tape as the encoder but without knowledge of π. For the first $T-1$ requests it just answers "no" to Arthur. Now comes the T-th request from Arthur, which must be a Probe(i, s) request with answer "yes". This allows the decoder to deduce $\pi(i) = s$. The remaining entries in π can be fully restored from π'.

Writing $p := \Pr(F)$, the encoding length is at most

$$\begin{aligned}
&(1-p)\Big(1 + \lceil \log(n!) \rceil\Big) + p\Big(1 + \lceil \log(m) \rceil + \lceil \log((n-1)!) \rceil\Big) \\
<&(1-p)\big(\log(n!) + 3\big) + p\big(\log(n) - 9/\delta + \log((n-1)!) + 3\big) \\
=&\log(n!) + 3 - 9p/\delta.
\end{aligned}$$

where we used $m := \lfloor 2^{-9/\delta} n \rfloor$ in the second line. Since the encoding length must be at least the Shannon entropy $\log(n!)$ of the encoded permutation π, we conclude that $\Pr(F) = p \leq \delta/3$.

Putting the two bounds together, we have $\Pr(E \cup F) \leq \Pr(E) + \Pr(F) \leq 2\delta/5$. ◀

Where does the game come into play? The reader might find it useful to recall Figure 4. The space $\mathbb{Z}^2$ is split into ℓ blocks horizontally, where each block $j \in [\ell]$ holds information about the j-th bit of the input vector. Let us focus on a particular block j. As $\mathcal{A}$ runs, some of

its queries "hit" this block while others don't. Precisely, we say that a Sample(X_i) query *hits* j if the random block chosen in Algorithm 2 is j. Likewise, we say that a Contains($(a,b), X_i$) query *hits* j if $\lfloor a/n \rfloor = j$. Now we restrict ourselves to the queries hitting j and ignore all the others. Then, over time, $\mathcal{A}$ (as Arthur) is exactly playing the game with us (as Merlin) on the secret permutation π_j: Sample() queries correspond to Random() requests; Contains() queries correspond to Probe() requests; Algorithm 2 branches to the "otherwise" clauses if and only if the answer is "yes". Whatever information $\mathcal{A}$ collects in the queries hitting other blocks j', it has no effect on the decision in this game, as all other permutations are independent of π_j, and as x, y do not play a role in this game.

With this in mind, let random variable N_j count the number of queries hitting j. Define

$$J := \{j \in [\ell] : x_j \text{ is accessed and } N_j \le m\} \quad \text{and} \quad J' := \{j \in [\ell] : N_j > m\}.$$

where $m := \lfloor 2^{-9/\delta} n \rfloor$. By Lemma 16, $\Pr(j \in J) \le 2\delta/5$ for every $j \in [\ell]$, so $\mathbb{E}[|J|] \le 2\delta\ell/5$. On the other hand, by definition of J' and our assumption that the number of queries is at most $\frac{\delta}{360 \cdot 2^{9/\delta}} \cdot \frac{n}{\varepsilon^2}$, we can bound

$$|J'| < \frac{1}{m+1} \cdot \frac{\delta}{360 \cdot 2^{9/\delta}} \cdot \frac{n}{\varepsilon^2} \le \frac{\delta\ell}{10}.$$

So Algorithm 2 accesses less than $\frac{2\delta\ell}{5} + \frac{\delta\ell}{10} = \frac{\delta\ell}{2}$ bits of x in expectation. Symmetrically, the same bound holds for y. Altogether it accesses less than $\delta\ell$ bits of x, y. But as we argued, the output can be used to distinguish $\langle x, y\rangle \ge \sqrt{\ell}$ and $\langle x, y\rangle \le -\sqrt{\ell}$. This contradicts Lemma 15.

5 Reduction from discrete to continuous space

Union volume estimation and Klee's measure problem were formulated in the continuous space $\mathbb{R}^d$. Their analogs in discrete space $\mathbb{Z}^d$ are only simpler due to the following reduction. Consider the natural embedding φ that blows up each point $x = (x_1, \ldots, x_d) \in \mathbb{Z}^d$ into a continuous cube $\varphi(x) = [x_1, x_1 + 1] \times \cdots \times [x_d, x_d + 1] \subset \mathbb{R}^d$. For any finite subset $X \subset \mathbb{Z}^d$, denote its image by $\varphi(X) \subset \mathbb{R}^d$. We have:

- Volume($\varphi(X)$) $= |X|$.
- To sample from $\varphi(X)$ uniformly, we can first sample $x \in X$ uniformly and then sample from $\varphi(x)$ uniformly.
- If X is a discrete box then $\varphi(X)$ is a continuous box, and vice versa.

Given this correspondence, our algorithm for Klee's measure problem can be made to handle discrete boxes in $\mathbb{Z}^d$. On the other hand, the discrete objects in our lower bound construction can be realized as axis-aligned polygons in $\mathbb{R}^2$.

References

1 Pankaj K. Agarwal. An improved algorithm for computing the volume of the union of cubes. In David G. Kirkpatrick and Joseph S. B. Mitchell, editors, *Proceedings of the 26th ACM Symposium on Computational Geometry, Snowbird, Utah, USA, June 13-16, 2010*, pages 230–239. ACM, 2010. doi:10.1145/1810959.1811000.

2 Pankaj K. Agarwal, Haim Kaplan, and Micha Sharir. Computing the volume of the union of cubes. In Jeff Erickson, editor, *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 294–301. ACM, 2007. doi:10.1145/1247069.1247121.

3 Karl Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Comput. Geom.*, 45(5-6):225–233, 2012. doi:10.1016/j.comgeo.2011.12.001.

4 Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.*, 43(6-7):601–610, 2010. doi:10.1016/j.comgeo.2010.03.004.

5 Karl Bringmann, Kasper Green Larsen, André Nusser, Eva Rotenberg, and Yanheng Wang. Approximating klee's measure problem and a lower bound for union volume estimation. *CoRR*, abs/2410.00996, 2024. doi:10.48550/arXiv.2410.00996.

6 Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995. doi:10.1016/0020-0190(94)00171-T.

7 Nofar Carmeli, Shai Zeevi, Christoph Berkholz, Alessio Conte, Benny Kimelfeld, and Nicole Schweikardt. Answering (unions of) conjunctive queries using random access and random-order enumeration. *ACM Trans. Database Syst.*, 47(3):9:1–9:49, 2022. doi:10.1145/3531055.

8 Ruoxu Cen, William He, Jason Li, and Debmalya Panigrahi. Beyond the quadratic time barrier for network unreliability. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1542–1567. SIAM, 2024. doi:10.1137/1.9781611977912.62.

9 Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 200–216. Springer, 2013. doi:10.1007/978-3-642-40627-0_18.

10 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discret. Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.

11 Timothy M. Chan. Semi-online maintenance of geometric optima and measures. *SIAM J. Comput.*, 32(3):700–716, 2003. doi:10.1137/S0097539702404389.

12 Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010. doi:10.1016/j.comgeo.2009.01.007.

13 Timothy M. Chan. Klee's measure problem made easy. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 410–419. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.51.

14 Timothy M. Chan. Minimum L_∞ Hausdorff distance of point sets under translation: Generalizing klee's measure problem. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.24.

15 Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007. doi:10.1007/s00778-006-0004-3.

16 David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discret. Comput. Geom.*, 11:321–350, 1994. doi:10.1007/BF02574012.

17 David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM J. Comput.*, 29(2):492–514, 1999. doi:10.1137/S0097539796298340.

18 Richard M. Karp and Michael Luby. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *J. Complex.*, 1(1):45–64, 1985. doi:10.1016/0885-064X(85)90021-4.

19 Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989. doi:10.1016/0196-6774(89)90038-2.

20 Benny Kimelfeld, Yuri Kosharovsky, and Yehoshua Sagiv. Query efficiency in probabilistic XML models. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 701–714. ACM, 2008. doi:10.1145/1376616.1376687.

21 Victor Klee. Can the measure of $\bigcup_1^n [a_i, b_i]$ be computed in less than $O(n \log n)$ steps? *The American Mathematical Monthly*, 84(4):284–285, 1977. doi:10.1080/00029890.1977.11994336.

22 Michael G. Luby. Monte-Carlo methods for estimating system reliability. Technical report, Report UCB/CSD 84/168, Computer Science Division, University of California, Berkeley, 1983.

23 Kuldeep S. Meel, Sourav Chakraborty, and N. V. Vinodchandran. Estimation of the size of union of delphic sets: Achieving independence from stream size. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 41–52. ACM, 2022. doi:10.1145/3517804.3526222.

24 Kuldeep S. Meel, Aditya A. Shrotri, and Moshe Y. Vardi. Not all FPRASs are equal: demystifying FPRASs for DNF-counting. *Constraints An Int. J.*, 24(3-4):211–233, 2019. doi:10.1007/s10601-018-9301-x.

25 Kuldeep S. Meel, N. V. Vinodchandran, and Sourav Chakraborty. Estimating the size of union of sets in streaming models. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 126–137. ACM, 2021. doi:10.1145/3452021.3458333.

26 Mark H. Overmars and Chee-Keng Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991. doi:10.1137/0220065.

27 Aduri Pavan, N. V. Vinodchandran, Arnab Bhattacharyya, and Kuldeep S. Meel. Model counting meets F_0 estimation. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 299–311. ACM, 2021. doi:10.1145/3452021.3458311.

28 Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 886–895. IEEE Computer Society, 2007. doi:10.1109/ICDE.2007.367934.

29 Qiaosheng Shi and Binay K. Bhattacharya. Application of computational geometry to network p-center location problems. In *Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008*, 2008.

30 Srikanta Tirthapura and David P. Woodruff. Rectangle-efficient aggregation in spatial data streams. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 283–294. ACM, 2012. doi:10.1145/2213556.2213595.

31 Jan van Leeuwen and Derick Wood. The measure problem for rectangular ranges in d-space. *J. Algorithms*, 2(3):282–300, 1981. doi:10.1016/0196-6774(81)90027-4.

32 Hakan Yildiz and Subhash Suri. On Klee's measure problem for grounded boxes. In Tamal K. Dey and Sue Whitesides, editors, *Proceedings of the 28th ACM Symposium on Computational Geometry, Chapel Hill, NC, USA, June 17-20, 2012*, pages 111–120. ACM, 2012. doi:10.1145/2261250.2261267.