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

Karl Bringmann* Kasper Green Larsen† André Nusser‡ Eva Rotenberg§
Yanheng Wang¶

Abstract

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

First, we show that if one can only interact with the objects via the aforementioned three queries, the query complexity of [Karp, Luby, Madras '89] is indeed optimal, i.e., $\Omega(n/\varepsilon^2)$ queries are necessary. Our lower bound already holds for estimating the union of equiponderous axis-aligned polygons in $\mathbb{R}^2$, and even if the algorithm is allowed to inspect the coordinates of the points sampled from the polygons, and still holds when a containment query can ask containment of an arbitrary (not necessarily sampled) point.

Second, guided by the insights of the lower bound, we provide a more efficient approximation algorithm for Klee's measure problem improving the $O(n/\varepsilon^2)$ time to $O((n+\frac{1}{\varepsilon^2}) \cdot \log^{O(d)} n)$. We achieve this improvement by exploiting the geometry of Klee's measure problem in various ways: (1) Since we have access to the boxes' coordinates, we can split the boxes into classes of boxes of similar shape. (2) Within each class, we show how to sample from the union of all boxes, by using orthogonal range searching. And (3) we exploit that boxes of different classes have small intersection, for most pairs of classes.

*bringmann@cs.uni-saarland.de. Saarland University and Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany. 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).

†larsen@cs.au.dk. Aarhus University. Supported by a DFF Sapere Aude Research Leader Grant No. 9064-00068B.

‡andre.nusser@cnrs.fr. CNRS, Inria, I3S, Université Côte d'Azur, France. 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.

§erot@dtu.dk. Technical University of Denmark. 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".

¶yanhwang@cs.uni-saarland.de. Saarland University and Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany. 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).

1 Introduction

We revisit the classical problem of *union volume estimation*: given objects $O_1, \ldots, O_n \subseteq \mathbb{R}^d$, 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 to the input objects.

After Karp and Luby [19] introduced this model, Karp, Luby and Madras [20] showed that one can $(1+\varepsilon)$-approximate the volume of n objects in this model using $O(n/\varepsilon^2)$ queries with constant success probability[2], by an algorithm that uses $O(n/\varepsilon^2)$ additional time (and their solution only asks containment queries of previously sampled points). This improved earlier related algorithms by Karp and Luby [19] and Luby [23]. In the last 35 years this problem has seen no improvement of the upper bound. Hence, it is natural to ask whether this classical upper bound is best possible and whether one can give a matching lower bound. We resolve this question in this work by providing a matching lower bound.

The union volume estimation problem was also studied very recently in the streaming setting [26, 24]. Here, the objects come in a stream $O_1, \ldots, O_n$, and when we are at position i in the stream, we can only query object O_i. Assuming the objects are subsets of a universe Ω, this line of work gives a streaming algorithm using $O(\text{polylog}(|\Omega|)\log(1/\delta)/\varepsilon^2)$ queries per object (the same bound holds for the space usage and update time additional to the queries). Summed over n boxes this yields the same total running time as the general tool, apart from the $\text{polylog}(|\Omega|)$ factor. So, interestingly, even in the streaming setting the same running time can be achieved.[3]

The perhaps most famous application of the algorithm by Karp, Luby, and Madras [20] is *Klee's measure problem* [22]: 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. Here an axis-aligned box is any set of the form $[a_1, b_1] \times \ldots \times [a_d, b_d] \subset \mathbb{R}^d$, and the input consists of the 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) [32, 27, 11, 2, 1, 12, 33, 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 for a d-dimensional axis-aligned box, the three queries can be implemented in time $O(d)$. Thus, the union volume estimation algorithm can be applied, and it computes a $(1+\varepsilon)$-approximation of Klee's measure problem in time $O(nd/\varepsilon^2)$, as 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] Technically, the objects need to be measurable. In fact, a generalization of this problem allows $O_1, \ldots, O_n$ to be any measurable subsets of a measure space, and we want to estimate the measure of their union. However, throughout this paper the objects will always be boxes in $\mathbb{R}^d$ (in our algorithm) or polygons in the plane (in our lower bound construction), and thus these technicalities are irrelevant in our context.

[2] The success probability can be boosted to $1-\delta$ at the cost of a factor $\log(1/\delta)$ in the number of queries and running time.

[3] See also [31] for earlier work studying Klee's measure problem in the streaming setting.

1.1 Our Contribution

Our contribution is twofold.

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 [20] 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 [20] is optimal also implies tightness of the known streaming algorithms (up to logarithmic factors), as the streaming algorithms match the static running time bound.

We answer this question negatively in the aforementioned query model. Note that the model allows unbounded computational power, examining the numerical coordinates of sampled points, and asking containment queries on arbitrary points. In contrast, these powers are not exploited by [20]. So our lower bound encompasses a much wider paradigm of algorithms. We show a query complexity lower bound of $\Omega(n/\varepsilon^2)$ for this model, which matches the upper bound of [20]:

Theorem 1. *Any algorithm for computing a $(1+\varepsilon)$-approximation to the cardinality of the union of n objects via volume, sampling and containment queries with success probability at least $4/5$ must make $\Omega(\varepsilon^{-2}n)$ queries.*

We want to particularly highlight that our lower bound even holds for subsets of $\mathbb{Z}^2$, and for equiponderous, axis-aligned polygons in the plane.

Upper bound for Klee's measure problem

Our lower bound for union volume estimation implies that we can only achieve an improvement of the current upper bound of Klee's measure problem if we exploit the geometric structure of boxes. Specifically, we exploit that we can split the input boxes into classes of similar boxes, since we have access to the boxes' coordinates, and we make use of orthogonal range searching. This allows us to break the barrier that is possible within the query model and provide an algorithm that improves Klee's measure problem from time $O(n/\varepsilon^2)$ to $O((n+\frac{1}{\varepsilon^2})\cdot \text{polylog}(n))$ in constant dimension.

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

The success probability can be boosted to any $1-\delta$ using standard techniques and incurring an additional $\log(1/\delta)$ factor in the running time. We also want to 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 objects. While this allows us to $(1+\varepsilon)$-approximate the volume of the union, we believe that our efficient sampling method is also of independent interest.

Throughout this work, for simplicity and readability we assume the dimension d to be constant. We remark that our running time bounds hide factors of the form $2^{O(d)}$.

1.2 Related Work

A major application of union volume estimation is *DNF Counting*, in which we are given a formula in disjunctive normal form and want to count its number of satisfying assignments. Computing the exact number of satisfying assignments is #P-complete, therefore it likely requires exponential time. Approximating the number of satisfying assignments can be achieved by an easy application

of union volume estimation, as described in [20]. Their algorithm remains the state of the art for this problem to this day, see, e.g., [25]. In particular, a direct application of the union volume estimation algorithm of [20] gives the best known complexity for approximate DNF Counting. This has been extended to more general model counting [28, 9, 25], probabilistic databases [21, 15, 29], and probabilistic queries on databases [6].

We also want to mention *Network Reliability* as another application for union volume estimation, which was already discussed in [20]. Additionally, Karger's famous paper on the problem [18] uses the algorithm of [20] as a subroutine. However, the current state-of-the-art algorithms no longer use union volume estimation as a tool [7].

Finally, we want to draw a connection to the following well-known query sampling bound. Canetti, Even, and Goldreich [5] showed that approximating the mean of a random variable whose codomain is the unit interval requires $\Omega(\log(1/\delta)/\varepsilon^2)$ queries, thus obtaining tight bounds for the sampling complexity of the mean estimation problem. Their bound generalises to $\Omega(\log(1/\delta)/(\mu\varepsilon^2))$ 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 importance is that techniques that have been developed for Klee's measure problem 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, e.g., 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) can also be reduced to Klee's measure problem [30], and finding the smallest hypercube containing at least k points among n given points can be reduced to the depth problem [17, 10, 13]. In light of this, it would be interesting to see whether our approximation techniques generalize to any of these related problems.

1.3 Technical Overview

We now give an overview of our results, starting with our upper bound result for Klee's measure problem. We keep the statements on an intuitive level and hide many technical details. For the formal statements and proofs, see Section 2 for the upper bound and Section 3 for the lower bound.

Upper bound for Klee's measure problem

We first remark that due to our lower bound result, we know that we have to exploit the structure of the input to obtain a running time of the form $O((n + \frac{1}{\varepsilon^2}) \cdot \text{polylog}(n))$. Following a common algorithmic approach, we use sampling to approximate the volume of the union. Specifically, we want to draw a sample S from the union of boxes with density p, such that in the end $|S|/p$ is a good estimate of the volume of the union of input boxes. We defer how to set p to the end of this overview and first focus on the main difficulty, i.e., how to create a sample for a given p.

We start with a simple classification of the input boxes into classes of similar shape. Two boxes are in the same *class* if the side lengths of both boxes in each dimension $i \in [d]$ lie in the same interval $[2^j, 2^{j+1})$ for some $j \in \mathbb{Z}$. We call two classes *similar* if their side lengths are polynomially related (e.g., within a factor of n^4) in each dimension.

We use the following three crucial insights to obtain an efficient algorithm:

1. We can efficiently sample from the union of boxes of a single class, see Lemma 4 (and Figure 1).
2. Each class has only few (i.e., a polylogarithmic number of) classes that are similar to it, see Observation 1.
3. Classes that are not similar have a small intersection compared to their union, see Observation 2 (and Figure 2).

In the remainder we give some more details on these insights and how they lead us to an efficient algorithm. The rough idea of our algorithm is as follows. We go through the classes in arbitrary order. For each class we sample with density p from the union of the boxes of this class, but we only keep a point if it is not contained in any class that comes later in the order. To efficiently check for containment in a later class, we use an orthogonal range searching data structure (with an additional dimension for the index of the class).

To understand why our algorithm is efficient, we have to look at two different parts:

Sampling from a class: One of our main technical ingredients is to sample from the union of boxes of similar shape. Note that efficient sampling implies efficient volume estimation, so to break our lower bound we must exploit additional input structure than those offered in the query model. Our main approach here is simple but powerful: We can sample points from the union of similar shaped boxes uniformly by (1) gridding the space into cells of side lengths comparable to these boxes, (2) sampling points from the relevant cells, and (3) discarding points not in the union by querying an orthogonal range searching data structure. As the grid size is similar to the shape of the boxes in the class, we ensure that a significant fraction of the points sampled in (2) are contained in the union, i.e., not discarded. The orthogonal range searching data structure allows us to quickly check for containment.

Bound the number of drawn samples: As we discard samples that appear in later classes, this is a potential source of inefficiency. Therefore, we need to bound the number of samples that we discard using the second and third insight from above. The second insight states that there are only few, i.e., polylogarithmically many, similar classes. Hence, a point might be discarded because it is contained in one of these similar classes, but as there are only few, this will only happen a polylogarithmic number of times. On the other hand, the third insight states that the intersection of dissimilar classes is small. Thus, the probability that we discard a sampled point because of a dissimilar class is small, and such events will not have a significant impact on the running time.

Finally, to set the sampling probability p, we need a crude estimate of the volume. To obtain a constant factor approximation, one can use the classical algorithms (by Karp and Luby [19] or Karp, Luby, Madras [20]) with a constant error parameter (say, $\frac{1}{2}$), to obtain a constant approximation factor in near-linear time. To keep our work self-contained, we provide a brief description and a simplified correctness proof of this case for union volume estimation, based on Karp and Luby [19], in Section 2.1.

Lower bound

We now give an overview of our lower bound result. The lower bound is proven by a reduction from a variant of the Gap-Hamming problem, defined as follows: Given two vectors $x, y \in \{-1, +1\}^T$, distinguish whether their inner product is greater than $\sqrt{T}$ or less than $-\sqrt{T}$. It is known that any

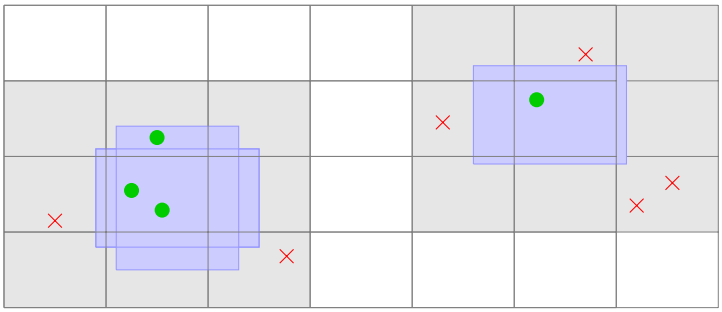

Figure 1: We sample points in the grid cells $\mathcal{G}$ that are intersected by a box $\mathcal{O}_i$ from a fixed class. We then use orthogonal range searching to determine whether a sampled point is in a box from the class and should be kept (●), or is not and should be discarded (×).

Figure 2: When boxes differ a lot in side length for at least one of their dimensions (in this case, the y-axis), their intersection is small compared to their union.

algorithm distinguishing these two cases with success probability at least $2/3$ must perform $\Omega(T)$ queries into x and y.

We first give the intuition why $\Omega(1/\varepsilon^2)$ samples are necessary to $(1+\varepsilon)$-approximate the union of two sets with constant probability in the query model. Given a Gap-Hamming instance x, y, we construct two sets $X = \{(i, x_i) : i \in [T]\}$ and $Y = \{(i, y_i) : i \in [T]\}$, see Figure 3 for an example. Note that for all $k \in \{0, \ldots, T\}$, we have

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

Hence, if we have an algorithm $\mathcal{A}$ that computes a $(1+\varepsilon)$-approximation of $|X \cup Y|$ with probability $2/3$, then we can distinguish between $\langle x, y\rangle \geq \varepsilon T$ and $\langle x, y\rangle \leq -\varepsilon T$. Setting $T = 1/\varepsilon^2$, we therefore distinguish $\langle x, y\rangle \geq 1/\varepsilon = \sqrt{T}$ and $\langle x, y\rangle \leq -1/\varepsilon = -\sqrt{T}$. Hence, our algorithm $\mathcal{A}$ solves the Gap-Hamming instance.

Note that the volumes of X and Y are fixed (depending only on the length of the vectors x and y but not their entries), and thus a volume query does not disclose any information about x and y. Each sample or containment query concerns at most one entry of x or y. Consequently, any union volume estimation algorithm has to use $\Omega(T) = \Omega(\varepsilon^{-2})$ queries to X or Y.

In order to generalize this lower bound for estimating the union of two sets to an $\Omega(n/\varepsilon^2)$ lower bound for estimating the union of n sets, we need to ensure that the sampled points do not give away too much information about the entries of x and y. We apply two obfuscations that jointly ensure a lower bound on the number of queries; see Figure 4. Firstly, we introduce sets $X_1, \ldots, X_n$ whose union is X and sets $Y_1, \ldots, Y_n$ whose union is Y. Imagine cutting each rectangle in Figure 3

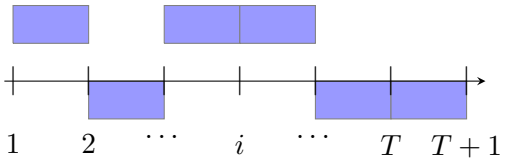

Figure 3: The vector $x = (+1, -1, +1, +1, -1, -1)$ represented as the set $\{(i, x_i) : i \in [6]\}$, where each point is drawn as a rectangle.

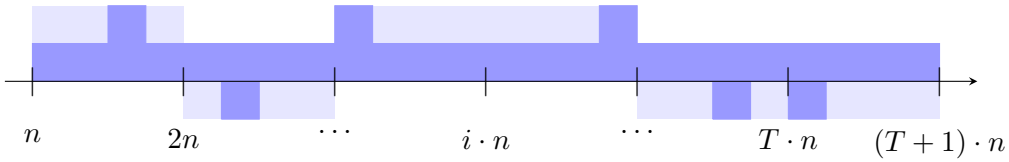

Figure 4: The vector y or $x = (+1, -1, +1, +1, -1, -1)$ gives rise to n polygons; one of these polygons is illustrated in dark blue. The light blue area indicates the union of all these n polygons.

into n side-by-side pieces and distributing them randomly among $X_1, \ldots, X_n$; similarly for Y. The idea is that one needs to make $\Omega(n)$ containment queries on a set in order to hit the correct piece. Hence, the effort for revealing one bit in x or y is $\Omega(n)$. Secondly, we introduce a large set 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. This large set intuitively enforces $\Omega(n)$ samples to even obtain a single point that contains any information about x and y.

2 Approximation Algorithm for Klee's Measure Problem

In this section we give our upper bound for Klee's measure problem.

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

2.1 Preliminaries

In Klee's measure problem we are given *boxes* $O_1, \ldots, O_n$ in $\mathbb{R}^d$. 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 input box. Throughout this section we assume d to be constant. Note that given the coordinates of a box, it is easy to compute its side lengths and volume. Throughout, we write $V \coloneqq \text{Volume}(\bigcup_{i=1}^{n} O_i)$ for the volume of the union of boxes. We want 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 that $\text{Pois}(\lambda)$ is the Poisson distribution with mean and variance λ. It captures the number of active points in a space, under the assumption that active points occur uniformly and independently at random across the space, 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 space $U \subset \mathbb{R}^d$ independently with "probability density" p, thus the number of activated points follows the Poisson distribution with mean $p \cdot \text{Volume}(U)$.

Definition 1 (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 $U \cup U'$.

We will make use of orthogonal range searching. Specifically, we need the query APPEARS(x, i), which upon receiving $x \in \mathbb{R}^d$ and $i \in \mathbb{N}$ returns true if $x \in O_i \cup \cdots \cup O_n$ and false otherwise.

Lemma 1. *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.*

Proof. For each $j \in [n]$, map the box $O_j \subset \mathbb{R}^d$ to a higher-dimensional box

$$O_j^+ := O_j \times (-\infty, j] \subset \mathbb{R}^{d+1}.$$

We then apply orthogonal range searching, specifically we build a multi-level segment tree over $\{O_1^+, \ldots, O_n^+\}$, which takes $O(n \log^{d+1} n)$ time; see [16, Section 10.4]. To answer the request APPEARS(x, i) where $x \in \mathbb{R}^d$ and $i \in \mathbb{N}$, we query the segment tree whether there exists a box O_j^+ that contains the point (x, i); or phrased differently, whether $x \in O_j$ for some $j \geq i$. The query takes only $O(\log^{d+1} n)$ time. □

For our main algorithm to work, we need a constant-factor approximation of the volume V. It is known that this can be computed in $O(n)$ time [20]. In order stay simple and self-contained, we prove a weaker result by implementing an algorithm of Karp and Luby [19] with the use of APPEARS queries.

Lemma 2 (Adapted from Karp and Luby [19])**.** *Given the data structure from Lemma 1, there exists an algorithm that computes in time $O(n \log^{d+1} n)$ a 2-approximation to V with probability at least* 0.9*.*

Algorithm 1 Crude volume estimator

1. Compute prefix sums $S_j := \sum_{i=1}^{j} \text{Volume}(O_i)$ for all $j \in \{0, \ldots, n\}$.
2. Initialise counter $N := 0$.
3. Repeat $40n$ times:
   - Sample $u \in [0, 1]$ uniformly at random. Binary search for the smallest i such that $u \leq \frac{S_i}{S_n}$.
   - Sample $x \in O_i$ uniformly at random.
   - Increment N if not APPEARS$(x,\ i + 1)$.
4. Output $\tilde{V} := \frac{N}{40n} \cdot S_n$.

Proof. We claim that Algorithm 1 has the desired properties. The time bound is easy to see: The computation of the prefix sums takes $O(n)$ time. In each iteration, binary searching for i costs $O(\log n)$ time, sampling of x costs $O(1)$ time, and calling APPEARS takes $O(\log^{d+1} n)$ time. So in total we spend $O(n \log^{d+1} n)$ time.

For the correctness argument, we define two sets

$$P := \{(i,x) \ : \ i \in [n],\, x \in O_i\},$$
$$Q := \{(i,x) \ : \ i \in [n],\, x \in O_i \setminus (O_{i+1} \cup \ldots \cup O_n)\}.$$

Consider an iteration in step 3. For any fixed value $j \in [n]$, we have

$$\Pr(i = j) = \Pr\left(\frac{S_{j-1}}{S_n} < u \leq \frac{S_j}{S_n}\right) = \frac{S_j - S_{j-1}}{S_n} = \frac{\text{Volume}(O_j)}{S_n}.$$

With this we can calculate the probability that the counter N increments in this iteration:

$$\begin{aligned}\Pr((i,x) \in Q) &= \sum_{j=1}^{n} \Pr((i,x) \in Q \mid i = j) \cdot \Pr(i = j) \\ &= \sum_{j=1}^{n} \frac{\text{Volume}(O_j \setminus (O_{j+1} \cup \ldots \cup O_n))}{\text{Volume}(O_j)} \cdot \frac{\text{Volume}(O_j)}{S_n} = \frac{V}{S_n}.\end{aligned}$$

Since all iterations are independent, at the end of the algorithm we have $N \sim \text{Bin}(40n, V/S_n)$. Hence $\widetilde{V}$ is an unbiased estimator for V.

To analyse deviation, we observe that $V \geq \max_{i=1}^{n} \text{Volume}(O_i) \geq S_n/n$. Therefore, $\mathbb{E}[N] = 40nV/S_n \geq 40$. By Chebyshev and as $\mathbb{E}[N] \geq \text{Var}[N]$, we have

$$\Pr\left(|N - \mathbb{E}[N]| \geq \frac{\mathbb{E}[N]}{2}\right) \leq \frac{4\text{Var}[N]}{(\mathbb{E}[N])^2} \leq \frac{4}{\mathbb{E}[N]} \leq 0.1.$$

That is, with probability at least 0.9 the output $\widetilde{V}$ is a 2-approximation to V. □

2.2 Classifying Boxes by Shapes

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

Definition 2. *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 i is contained in $[2^{L_i}, 2^{L_i+1})$, for each $i \in [d]$.*

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. We will fix this notation throughout. For each $t \in [m]$, let us also define $U_t := \bigcup_{O \in C_t} O \subseteq \mathbb{R}^d$, namely the union of boxes in class C_t.

Similar to APPEARS, we can answer queries of the form: Is a given point $x \in \mathbb{R}^d$ contained in U_t? We call this an INCLASS(x,t) query.

Lemma 3. *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.*

Proof. Similar to the proof of Lemma 1, we transform each $O_i \in C_t$ to a higher-dimensional box

$$O_i \times \{t\} \subset \mathbb{R}^{d+1}$$

and build a multi-level segment tree on top. The query INCLASS(x,t) is thus implemented by querying the point $x \times \{t\} \in \mathbb{R}^{d+1}$ in the segment tree. □

Sampling from a class

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

Lemma 4. *Given $t \in [m]$, $p \in [0,1]$ and the data structure from Lemma 3, 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. Write $(L_1, \dots, L_d)$ for the type corresponding to class C_t. We subdivide $\mathbb{R}^d$ into the grid

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

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 have a non-empty intersection with U_t. Write $U := \bigcup_{G \in \mathcal{G}} G$.

First we create a p-sample S of U as follows. Generate $K \sim \text{Pois}(p \cdot \text{Volume}(U))$, which determines the number of points we are going to sample. 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 $\text{inClass}(x, t)$; 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 $(L_1, \dots, L_d)$. In any dimension $k \in [d]$, one 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 $\text{Volume}(U) \leq 3^d \, \text{Volume}(U_t)$.

Regarding the running time, recall that we assume d to be constant and hence drop factors only depending on d. 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 the data structure from Lemma 1 once for each point of S, the expected time of the inClass queries is $O(p \cdot \text{Volume}(U_t) \cdot \log^{d+1} n)$. □

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

We later use this property to draw p-samples from $\bigcup_{i=1}^{n} O_i = \bigcup_{t=1}^{m} U_t$ efficiently. To show this property, we first need some simple definitions and observations.

Definition 3. *We call classes of type $(L_1, \dots, L_d)$ and $(L'_1, \dots, L'_d)$* similar *if for all $k \in [d]$ we have $2^{|L_k - L'_k|} < n^4$. Otherwise we call them* dissimilar*.*

Observation 1. *Every class is similar to at most $8^d \log^d n$ classes.*

Proof. Fix a type $(L_1, \dots, L_d)$. 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 2. *Let O and O' be boxes 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 intervals resulting from projecting the boxes 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,

$$\text{Volume}(O \cap O') \leq \text{Volume}(O) \cdot 2/n^4 \leq 2V/n^4. \qquad \square$$

We are now ready to prove Lemma 5.

Proof of Lemma 5. 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:

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

We have $t \notin T$ for some $t \in [m]$ 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 thus 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 1. 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 2. 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. $\square$

2.3 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. More formally, we ensure that $C_t = \{O_{i_{t-1}+1}, \ldots, O_{i_t}\}$ for $0 = i_0 < i_1 < \ldots < i_m = n$.

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 4. 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

Algorithm 2 Volume estimator

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 1 and 3.
3. Call Algorithm 1 to obtain a crude estimate $\widetilde{V}$. 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 4.
   - Compute $S'_t := \{x \in S_t : \textsc{appears}(x, i_t + 1) = \text{false}\}$.
5. Output $\sum_{t=1}^{m} |S'_t|/p$.

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 2.

Lemma 6. *Conditioned on $\widetilde{V} \le 2V$, Algorithm 2 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,

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

In other words, with probability at least $3/4$, the output N/p is a $1+\varepsilon$ approximation to V. □

Lemma 7. *Conditioned on $\widetilde{V} \ge \frac{V}{2}$, Algorithm 2 runs in expected time $O\left(\log^{2d+1}(n) \cdot (n + \frac{1}{\varepsilon^2})\right)$.*

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

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 4, and computing S'_t takes expected time $O((1 + p \,\text{Volume}(U_t)) \cdot \log^{d+1} n)$ by Lemma 1. Therefore, the expected running time over all iterations 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 5, 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 5 is $O\left(\log^{2d+1}(n) \cdot (n + \frac{1}{\varepsilon^2})\right)$. □

[4] This idea has previously been used on objects, by considering the difference $D'_i := O_i \setminus (\bigcup_{j=i+1}^{n} O_j)$ [19, 26], while we use this idea on classes.

Proof of Theorem 2. We run Algorithm 2 with a time budget tenfold the bound in Lemma 7; if step 5 spends excessive time then we immediately abort the algorithm. 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 2, the first event happens with probability at most 0.1. By Markov's inequality, the second event happens with probability at most 0.1. Lastly, by Lemma 6, 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 $1 - \frac{9}{20}$ can be boosted to, say, 0.9 by returning the median of a sufficiently large constant number of repetitions of the algorithm. □

2.4 Handling Discrete Boxes

We now argue that our algorithm for boxes in $\mathbb{R}^d$ also solves the following discrete variant of Klee's measure problem: Given boxes $O_1, \ldots, O_n$ in $\mathbb{Z}^d$, count the number of points in the union $\bigcup_{i=1}^n O_i$. To solve this problem, we employ the following embedding of $\mathbb{Z}^d$ into $\mathbb{R}^d$:

$$\varphi : (x_1, \ldots, x_d) \in \mathbb{Z}^d \mapsto [x_1, x_1 + 1] \times \cdots \times [x_d, x_d + 1] \subset \mathbb{R}^d.$$

Note that φ transforms discrete boxes into continuous boxes, and that the cardinality of any $U \subset \mathbb{Z}^d$ is equal to the volume of its image $\varphi(U) \subset \mathbb{R}^d$. Hence the discrete variant of Klee's measure problem reduces to the continuous counterpart.

3 Lower Bound for Union Volume Estimation

We consider estimating the volume of the union of n (measurable) objects $O_1, \ldots, O_n \subset \mathbb{R}^2$. These objects are only accessible through the following three queries:

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

It is known that $O(n\varepsilon^{-2})$ queries suffice to return with constant probability a $(1+\varepsilon)$-approximation to the volume of the union $O_1 \cup \ldots \cup O_n$. Here we prove a matching lower bound.

For convenience, we also consider a discrete version of the problem in which each object O_i is instead a finite subset of the integer lattice $\mathbb{Z}^2$. The queries are then

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

The goal is to give a $(1 + \varepsilon)$-approximation to the cardinality $|O_1 \cup \ldots \cup O_n|$ of the union.

In Section 3.1 we show a lower bound for the discrete version, and then in Section 3.2 we show that a lower bound for the discrete version implies a similar lower bound for the continuous version.

3.1 Lower Bound for Discrete Union

In the remainder, we write $[n] \coloneqq \{1, 2, \ldots, n\}$. The starting point is what we call the Query-Gap-Hamming problem: The input is two (hidden) vectors $x, y \in \{-1, 1\}^T$ and we can access an arbitrary bit of x or y at a time. The goal is to distinguish the cases $\langle x, y\rangle > \sqrt{T}$ and $\langle x, y\rangle < -\sqrt{T}$ using as few accesses as possible. Query-Gap-Hamming has linear query complexity:

Lemma 8. *Any randomized algorithm solving Query-Gap-Hamming with probability at least* $2/3$ *requires* $\Omega(T)$ *accesses to* x *and* y*, regardless of the computational resources it uses.*

Proof. This follows by a folklore argument from the fact that the Gap-Hamming problem has linear randomized communication complexity [8]. We next describe the details.

We reduce from the communication complexity of the Gap-Hamming problem, where Alice holds a vector $x \in \{-1, 1\}^T$, Bob holds a vector $y \in \{-1, 1\}^T$, and their goal is to distinguish $\langle x, y\rangle > \sqrt{T}$ from $\langle x, y\rangle < -\sqrt{T}$ while communicating as few bits as possible. It is known that the two-way, public-coin randomized communication complexity of Gap-Hamming is $\Omega(n)$ [8]. Now suppose that a randomized algorithm can solve Query-Gap-Hamming with probability at least $2/3$, while making only $o(n)$ accesses to x and y. We construct a protocol between Alice and Bob: They simulate the algorithm synchronously, using a shared random tape. Whenever the algorithm tries to access x_j, Alice sends the bit x_j to Bob. Whenever it tries to access y_j, Bob sends the bit y_j to Alice. Clearly both parties can simulate the algorithm till the end, and output the answer of the algorithm. The communication cost is $o(n)$ bits, which contradicts the aforementioned communication complexity. □

Next we give a reduction from Query-Gap-Hamming to estimating the cardinality of a union of objects. In more detail, from the hidden input vectors $x, y \in \{-1, 1\}^T$ we (implicitly) define $2n$ objects $X_1, \ldots, X_n, Y_1, \ldots, Y_n \subset \mathbb{Z}^2$. Write $R \coloneqq \{(n+1, 0), \ldots, (nT+n, 0)\}$. Given permutations $\pi_1, \ldots, \pi_T$ of $[n]$, we define

$$X_i = X_i(x, \pi_1, \ldots, \pi_T) \coloneqq R \cup \{(jn + \pi_j(i), x_j) : j \in [T]\}$$

for every $i \in [n]$. Analogously, given a different set of permutations $\tau_1, \ldots, \tau_T$, we define

$$Y_i = Y_i(y, \tau_1, \ldots, \tau_T) \coloneqq R \cup \{(jn + \tau_j(i), y_j) : j \in [T]\}$$

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

Consider an arbitrary index $j \in [T]$. 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, 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 nT. Hence, the cardinality of the union equals

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

Let ρ be a $(1+\varepsilon)$-approximation to the cardinality of the union, i.e., $\frac{5}{2}nT - \frac{1}{2}n\langle x, y\rangle$. Since

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

and $|\langle x, y\rangle| \leq T$, by computing $(\frac{5}{2}nT - \rho) \cdot \frac{2}{n}$ we obtain a value in $[\langle x, y\rangle - 6\varepsilon T, \langle x, y\rangle + 6\varepsilon T]$, namely an additive $6\varepsilon T$ approximation to $\langle x, y\rangle$ with probability at least $4/5$. For $\varepsilon \leq 1/(6\sqrt{T})$ this allows to decide $\langle x, y\rangle > \sqrt{T}$ or $\langle x, y\rangle < -\sqrt{T}$.

Let $\mathcal{A}$ be a (possibly randomized) algorithm that $(1+\varepsilon)$-approximates the volume of union of any $2n$ objects $O_1, \ldots, O_{2n} \subset \mathbb{Z}^2$ with probability at least $4/5$, using q queries. We assume that $q \geq 10n$; otherwise we modify $\mathcal{A}$ to ask $10n - q$ dummy queries.

We now simulate $\mathcal{A}$ as if the input were the $2n$ objects $X_1, \ldots, X_n$, $Y_1, \ldots, Y_n$. It remains to argue that we can answer all queries by $\mathcal{A}$ while accessing few bits in x and y. Specifically, the number of accesses would be only $O(q/n)$. The details of the simulation algorithm are as follows:

Algorithm $\mathcal{S}$:

1. Sample random permutations $\pi_1, \ldots, \pi_T$ and $\tau_1, \ldots, \tau_T$ of $[n]$ uniformly and independently.
2. Simulate algorithm $\mathcal{A}$ and answer its queries as follows.
   - Volume(i): Answer $nT + T$.
   - Sample(i): In the case $i \leq n$:
     (S1) With probability $nT/(nT+T) = 1-1/(n+1)$, answer with a uniform random point $p \in R$.
     (S2) With the remaining probability, pick a uniform random $j \in [T]$. If we have not accessed x_j yet, access it and keep it in memory. Then answer with the point $(jn + \pi_j(i), x_j)$.

     In the case $i > n$, do the same with x_j replaced by y_j and $\pi_j(i)$ replaced by $\tau_j(i-n)$.
   - Contains($(a,b), i$): Let $j = \lfloor (a-1)/n \rfloor$. In the case $i \leq n$:
     (C1) If $(a,b) \in R$ then answer **true**.
     (C2) Else, if $j \notin [n]$ or $jn + \pi_j(i) \neq a$ then answer **false**.
     (C3) Else, we have $jn + \pi_j(i) = a$. If we have not accessed x_j yet, access it and keep it in memory. If $b = x_j$ then answer **true**, otherwise answer **false**.

     In the case $i > n$, do the same with x_j replaced by y_j and $\pi_j(i)$ replaced by $\tau_j(i-n)$.
3. Let ρ be output of $\mathcal{A}$ and return $(\frac{5}{2}nT - \rho) \cdot \frac{2}{n}$.

This finishes the description of algorithm $\mathcal{S}$. It is immediate from the algorithm that the execution of $\mathcal{A}$ is the same as if actually running it on the objects $X_i(x, \pi_1, \ldots, \pi_T)$ and $Y_i(y, \tau_1, \ldots, \tau_T)$ for $i \in [n]$. What remains is to bound the number of accesses to x and y by $\mathcal{S}$ during the simulation.

To this end, observe that an access to x (respectively y) occurs only when the query enters (S2) or (C3). In both (S2) and (C3), a permutation entry $\pi_j(i)$ (respectively $\tau_j(i-n)$) is involved, and we say that the entry is *hit* by the query.

By definition, the number of accesses to x and y is exactly the number of entries $\pi_j(i)$ and $\tau_j(i-n)$ hit by some query. In light of this, we can move on to upper bound the latter.

We consider two bad events. Let E_1 be the event that more than $20q/n$ entries are hit by (S2). Let E_2 be the event that at most $20q/n$ entries are hit by (S2), but more than $k := 40q/n$ entries are *freshly* hit by (C3). Here "freshly" means that the entry was not hit by any query before it is hit by (C3).

Entries hit by (S2). We first consider the number of entries hit by (S2). For $t \in [q]$, define an indicator random variable Z_t taking the value 1 iff the t-th query of $\mathcal{S}$ enters case (S2). Since every query may hit at most one entry, the total number of entries hit by (S2) is at most $\sum_{t=1}^{q} Z_t$. Note that $\Pr[Z_t = 1] \leq 1/(n+1)$ for all t and hence $\mathbb{E}[\sum_{t=1}^{q} Z_t] < q/n$. So by Markov's inequality, $\Pr[E_1] \leq \Pr[\sum_{t=1}^{q} Z_t > 20q/n] < 1/20$.

Entries freshly hit by (C3). The tricky query to analyze is the Contains$((a,b),i)$ query. We will show that $\Pr[E_2] < 1/20$. Roughly, we need to argue that if $\pi_j(i)$ was not hit previously then $\mathcal{A}$ is unlikely to ask a query with $a = jn + \pi_j(i)$. The intuition is that $\mathcal{A}$ is unaware of the permutations $\pi_1, \ldots, \pi_T$ and $\tau_1, \ldots, \tau_T$, and thus to get a fresh hit it has to "guess" an entry of a permutation.

For the proof, assume for the sake of contradiction that $\Pr[E_2] \geq 1/20$. Under this assumption, we give an algorithm for encoding the random permutations $\pi_1, \ldots, \pi_T, \tau_1, \ldots, \tau_T$ in less than $2T \lg(n!)$ bits in expectation. This is an information theoretic contradiction. More formally, our proof considers a game between an encoder and a decoder. The encoder receives $\pi_1, \ldots, \pi_T, \tau_1, \ldots, \tau_T, x, y$ as well as the random tape r used by $\mathcal{S}$ and $\mathcal{A}$ in simulation step 2. The decoder receives x, y, r. The encoder must send a message to the decoder which allows the decoder to reconstruct $\pi_1, \ldots, \pi_T, \tau_1, \ldots, \tau_T$. Since the Shannon entropy is $H(\pi_1, \ldots, \pi_T, \tau_1, \ldots, \tau_T \mid x, y, r) = 2T \lg(n!)$, it follows by Shannon's source coding theorem that the expected length of the message must be at least $2T \lg(n!)$ bits.

The way we use the assumption $\Pr[E_2] \geq 1/20$, is that the encoder will send the indices of the queries among $1, \ldots, q$ which freshly hit an entry in (C3). The encoder will further send information that allows the decoder to simulate $\mathcal{S}$ for the remaining queries. Whenever the decoder reaches one of the specified queries, she knows that the point (a,b) given by the Contains$((a,b),i)$ query satisfies $jn + \pi_j(i) = a$. This allows her to recover $\pi_j(i)$, i.e., roughly $\lg n$ bits of information. But sending k such indices costs $\lg \binom{q}{k} \approx k \lg(q/k)$ bits, or $\lg(q/k)$ bits per index. Since $q/k \ll n$, we use less bits than the information theoretic lower bound, which is a contradiction. We now proceed to give the formal details.

Encoding procedure. The encoder receives random permutations $\pi_1, \ldots, \pi_T, \tau_1, \ldots, \tau_T$ and also x, y, r, and proceeds as follows:

1. Initialize algorithm $\mathcal{S}$ with the given permutations. Run it from step 2 onward, using the given tape r to make random choices for $\mathcal{S}$ and $\mathcal{A}$.

2. If the event E_2 does not happen, send a `0`-bit followed by a naive encoding of all permutations.

3. Otherwise E_2 happens. Signal this by sending a `1`-bit. Then send the indices $I \subseteq [q]$ of the first k queries that freshly hit some entry in (C3). Next, denote $\ell := \max I$. For $t = 1, \ldots, \ell$ in that order, if the t-th query hits an entry in (S2) then send the value of that entry. Finally, for each permutation π_j and τ_j, send the induced permutation on its entries not hit by queries $1, \ldots, \ell$.

Decoding procedure. We next argue that we can recover the permutations $\pi_1, \ldots, \pi_T$ and $\tau_1, \ldots, \tau_T$ after receiving x, y, r and the above encoding.

1. If the leading bit of the encoding is a `0`, then we immediately recover all permutations from the rest of the encoding.

2. If the leading bit is a `1`, we start by recovering I and $\ell := \max I$. Then we simulate algorithm $\mathcal{S}$ up to the ℓ-th query, as if we knew the permutations. In the meantime we gradually recover all entries $\pi_j(i)$ and $\tau_j(i-n)$ that are hit. More precisely, for $t = 1, \ldots, \ell$ we answer the t-th query by $\mathcal{A}$ as follows.

   - Volume(i): Answer $nT + T$.

- Sample(i): In the case $i \le n$:
  - If the tape r decides to give a point $p \in R$, then answer with this point p.
  - Else, the tape decides to give a $j \in [T]$. Since $\pi_j(i)$ is hit by this query in (S2), its value is readily available in the encoding. We answer $(jn + \pi_j(i), x_j)$.

  In the case $i > n$, do the same with x_j replaced by y_j and $\pi_j(i)$ replaced by $\tau_j(i-n)$.
- Contains($(a,b), i$): Let $j = \lfloor (a-1)/n \rfloor$. In the case $i \le n$:
  - If $(a,b) \in R$ then answer **true**.
  - Else, if $j \notin [n]$ then answer **false**.
  - Else, if $t \in I$ then the current query freshly hits $\pi_j(i)$, so it must be the case that $a = jn + \pi_j(i)$. We have thus recovered $\pi_j(i) = a - jn$. Then we answer **true** if $b = x_j$; otherwise we answer **false**.
  - Finally, if $t \notin I$ then $\pi_j(i)$ was hit before, or it is not hit by the current query. In the former case we know its value, so we answer **true** if $(a,b) = (jn + \pi_j(i), x_j)$, and **false** otherwise. In the latter case we know that $a \neq jn + \pi_j(n)$, so we simply answer **false**.

  In the case $i > n$, do the same with x_j replaced by y_j and $\pi_j(i)$ replaced by $\tau_j(i-n)$.

3. Having recovered all entries of $\pi_1, \ldots, \pi_T$ and $\tau_1, \ldots, \tau_T$ that are hit by queries $1, \ldots, \ell$, we finally recover the remaining entries from the rest of the encoding.

Encoding length. We finally analyze the expected encoding length to derive a contradiction to the assumption that $\Pr[E_2] \ge 1/20$.

If E_2 does not happen then the encoding length is $1 + \lceil 2T \lg(n!) \rceil \le 2 + 2T \lg(n!)$ bits. If E_2 happens then we can save a significant number of bits. To this end, let us focus on the queries $1, \ldots, \ell$. Let m be the number of entries hit by (S2); note that $m \le 20q/n$ under the event E_2. For $j = 1, \ldots, T$ let n_j be the number of entries in π_j *not* hit by any query. Similarly, for $j = T+1, \ldots, 2T$ let n_j be the number of entries in τ_j *not* hit by any query. Then the encoding length is

$$\begin{aligned}
& 1 + \left\lceil \lg \binom{q}{k} \right\rceil + m\lceil \lg n \rceil + \left\lceil \sum_{j=1}^{2T} \lg(n_j!) \right\rceil \\
\le\ & 3 + m + \lg \binom{q}{k} + m \lg n + \sum_{j=1}^{2T} \lg(n_j!) \\
\le\ & 3 + m + k \lg(eq/k) + m \lg n + \sum_{j=1}^{2T} \lg(n!) - \sum_{j=1}^{2T} \lg(n!/n_j!) \\
=\ & 3 + m + k \lg(eq/k) + m \lg n + 2T \lg(n!) - \sum_{j=1}^{2T} \lg(n!/n_j!).
\end{aligned}$$

By Stirling's approximation, we have $n! \ge (n/e)^n$. Hence, the product of the $n - n_j$ largest terms in the factorial (namely $n!/n_j!$) is at least $(n/e)^{n-n_j}$. Thus

$$\sum_{j=1}^{2T} \lg(n!/n_j!) \ \ge\ \sum_{j=1}^{2T} (n - n_j) \lg(n/e) \ \ge\ (\lg(n) - 2) \cdot \sum_{j=1}^{2T} (n - n_j).$$

Since $\sum_{j=1}^{2T}(n-n_j) = m+k$ is exactly the number of entries hit by queries $1,\ldots,\ell$, the encoding length is at most

$$\begin{aligned} & 3 + m + k\lg(eq/k) + m\lg n + 2T\lg(n!) - (\lg(n)-2)\cdot(k+m) \\ = \;& 3 + 3m + k\lg(4eq/(kn)) + 2T\lg(n!) \\ \leq \;& 3 + 60q/n + k\lg(4eq/(kn)) + 2T\lg(n!) \end{aligned}$$

where we used $m \leq 20q/n$ by event E_2.

Recalling our choice of $k = 40q/n$ and the assumption that $q \geq 10n$, the above is at most

$$\begin{aligned} & 3 + 60q/n + (40q/n)\lg(e/10) + 2T\lg(n!) \\ < \;& 3 + 60q/n - 75q/n + 2T\lg(n!) \\ \leq \;& 2T\lg(n!) - 147. \end{aligned}$$

Therefore, the expected encoding length is no more than

$$\begin{aligned} & (1-\Pr[E_2])\cdot(2+2T\lg(n!)) + \Pr[E_2]\cdot(2T\lg(n!)-147) \\ \leq \;& 2T\lg(n!) + 2 - 147\Pr[E_2] \\ < \;& 2T\lg(n!) - 5. \end{aligned}$$

where the last line follows from the assumption that $\Pr[E_2] \geq 1/20$. This contradicts with the information theoretic lower bound.

Conclusion. We have now shown that $\Pr[E_1] \leq 1/20$ and $\Pr[E_2] \leq 1/20$. By a union bound, we have that none of the events happen, so $\mathcal{S}$ computes a $6\varepsilon T$ additive approximation to $\langle x, y\rangle$, with probability at least $4/5 - 1/10 \geq 2/3$. In this case, the number of hit entries is at most $20q/n + 40q/n = 60q/n$, so is the number of accesses to x, y. If $\mathcal{S}$ performs more than $60q/n$ queries, we may simply abort and return an arbitrary answer; this does not affect the probability bound.

Recall that we made the simplifying assumption $q \geq 10n$. If the algorithm $\mathcal{A}$ that we began with asks less than $10n$ queries, then we added dummy queries to ensure $q = 10n$, and the number of accesses to x, y becomes $60q/n = 600$. In any case, the number of accesses is $O(q/n)$. We thus have an algorithm $\mathcal{S}$ that makes only $O(q/n)$ accesses and returns a $6\varepsilon T$ additive approximation with probability at least $2/3$. We may set $T = \varepsilon^{-2}/144$ to obtain an additive $6\varepsilon T = \varepsilon^{-1}/24 = \sqrt{T}/2$ approximation. This is enough to solve the Query-Gap-Hamming problem and hence the number of accesses must be $\Omega(T) = \Omega(\varepsilon^{-2})$ by Lemma 8. We thus have $q/n = \Omega(\varepsilon^{-2})$, or $q = \Omega(\varepsilon^{-2}n)$. This proves Theorem 1, in the discrete setting with objects in $\mathbb{Z}^2$.

3.2 Continuous to Discrete

To prove a lower bound for estimating the volume of the union of n objects in $\mathbb{R}^2$, we give a simple reduction from estimating the cardinality of the union of n objects in $\mathbb{Z}^2$. Let $\mathcal{A}$ be an algorithm for estimating the volume of the union of n objects in $\mathbb{R}^2$ using Volume, Sample and Contains queries.

We use $\mathcal{A}$ to estimate the cardinality of the union of n sets in $\mathbb{Z}^2$ as follows. Let $O_1,\ldots,O_n \subset \mathbb{Z}^2$ be the objects. We think of them as objects in $\mathbb{R}^2$ by replacing each point (x, y) in an object O_i by the unit square that has (x, y) in its lower left corner, i.e. $[x, x+1)\times[y, y+1)$. Denote the resulting objects in $\mathbb{R}^2$ by O_i'. (Note that applying this transformation to our objects from the previous reduction gives connected axis-aligned polygons.)

The volume of the union $O'_1 \cup \ldots \cup O'_n$ is the same as the cardinality of the union $O_1 \cup \ldots \cup O_n$. We thus merely need to simulate $\mathcal{A}$ as if the input was $O'_1, \ldots, O'_n$. For this, on every Volume(i) query made by $\mathcal{A}$ to the object O'_i, we ask the same query to O_i. For a Sample(i) query made by $\mathcal{A}$, we run Sample(i) on O_i, receive an integer point $(a, b) \in \mathbb{Z}^2$. We then draw $r_x \in [0, 1)$ and $r_y \in [0, 1)$ independently and uniformly at random and feed $\mathcal{A}$ the point $(x + r_x, y + r_y)$ as the result of the Sample(i) query. Finally, when $\mathcal{A}$ asks a Contains$((a, b), i)$ query, we simply round the coordinates down to the nearest integers to obtain a point $(a', b') = (\lfloor a \rfloor, \lfloor b \rfloor) \in \mathbb{Z}^2$. When then query Contains$((a', b'), i)$ on O_i. Correctness follows immediately and we conclude:

Theorem 3. *Any algorithm for computing a $(1 + \varepsilon)$-approximation to the volume of the union of n objects in $\mathbb{R}^2$ with probability at least $4/5$ via* Volume, Sample *and* Contains *queries, must use $\Omega(\varepsilon^{-2} n)$ queries.*

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.

[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.

[3] Karl Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Comput. Geom.*, 45(5-6):225–233, 2012.

[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.

[5] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995.

[6] 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.

[7] Ruoxu Cen, William He, Jason Li, and Debmalya Panigrahi. Beyond the quadratic time barrier for network unreliability. *CoRR*, abs/2304.06552, 2023.

[8] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 51–60, 2011.

[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.

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

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

[12] Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010.

[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.

[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.

[15] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[16] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[17] David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discret. Comput. Geom.*, 11:321–350, 1994.

[18] 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.

[19] Richard M. Karp and Michael Luby. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *J. Complex.*, 1(1):45–64, 1985.

[20] Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Jan van Leeuwen and Derick Wood. The measure problem for rectangular ranges in d-space. *J. Algorithms*, 2(3):282–300, 1981.

[33] 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.