Testing Properties of Distributions in the Streaming Model

Sampriti Roy
Department of Computer Science and Engineering, IIT Madras, Chennai, India

Yadu Vasudev
Department of Computer Science and Engineering, IIT Madras, Chennai, India

Abstract

We study distribution testing in the standard access model and the conditional access model when the memory available to the testing algorithm is bounded. In both scenarios, we consider the samples appear in an online fashion. The goal is to test the properties of distribution using an optimal number of samples subject to a memory constraint on how many samples can be stored at a given time. First, we provide a trade-off between the sample complexity and the space complexity for testing identity when the samples are drawn according to the conditional access oracle. We then show that we can learn a succinct representation of a monotone distribution efficiently with a memory constraint on the number of samples that are stored that is almost optimal. We also show that the algorithm for monotone distributions can be extended to a larger class of decomposable distributions.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Property testing, distribution testing, streaming

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.56

Related Version *Full Version*: https://arxiv.org/abs/2309.03245 [22]

1 Introduction

Sublinear algorithms that analyze massive amounts of data are crucial in many applications currently. Understanding the underlying probability distribution that generates the data is important in this regard. In the field of distribution testing, a sub-field of property testing, the goal is to test whether a given unknown distribution has a property $\mathcal{P}$ or is far from having the property $\mathcal{P}$ (where the farness is defined with respect to total variation distance). Starting from the work of Goldreich and Ron ([19]), a vast literature of work has studied the problem of testing probability distributions for important properties like identity, closeness, support size as well as properties relating to the structure of the distribution like monotonicity, k-modality, and histograms among many others; see Canonne's survey ([10]) for an overview of the problems and results.

In the works of Canonne et al ([12]) and Chakraborty et al ([13]), distribution testing with conditional samples was studied. In this model, the algorithm can choose a subset of the support, and the samples of the distribution conditioned on this subset are generated. This allows adaptive sampling from the distribution and can give better sample complexity for a number of problems. In particular, ([12]) and ([13]) give testers for uniformity and other problems that use only a constant number of samples.

The natural complexity measure of interest is the number of samples of the underlying distribution that is necessary to test the property. In many cases, when data is large, it might be infeasible to store all the samples that are generated. A recent line of work has been to study the trade-off between the sample complexity and the space complexity of algorithms

34th International Symposium on Algorithms and Computation (ISAAC 2023).
Editors: Satoru Iwata and Naonori Kakimura; Article No. 56; pp. 56:1–56:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for learning and testing properties of distributions. This model can be equivalently thought of as a data stream of i.i.d samples from an unknown distribution, with the constraint that you are allowed to store only a small subset of these samples at any point in time.

In this work, we study distribution testing problems in the standard model, and when the algorithm is allowed to condition on sets to better understand the trade-off between the sample complexity and size. In particular, we study identity testing and testing whether the unknown distribution is monotone. Our work borrows ideas from the recent work of Diakonikolas et al ([17]) and extends the ideas to these problems.

1.1 Related work

Testing and estimating the properties of discrete distributions is well-studied in property testing; see ([10]) for a nice survey of recent results. In our work, we study property testing of discrete distributions under additional memory constraints wherein the algorithm does not have the resources to store all the samples that it obtains.

This line of work has received a lot of attention in recent times. Chien et al ([14]) propose a sample-space trade-off for testing any (ϵ, δ)-weakly continuous properties, as defined by Valiant ([23]). Another work by Diakonikolas et al ([17]) studies the uniformity, identity, and closeness testing problems and presents trade-offs between the sample complexity and the space complexity of the tester. They use the idea of a *bipartite collision tester* where instead of storing all the samples in the memory, the testing can be done by storing a subset of samples and counting the collisions between the stored set and the samples that come later. Another line of work ([1, 2]) focuses on the task of estimating the entropy of distributions from samples in the streaming model, where space is limited. In particular, ([1]) estimate the entropy of an unknown distribution D up to $\pm\epsilon$ using constant space. Berg et al ([7]) study the uniformity testing problem in a slightly different model where the testing algorithm is modeled as a finite-state machine.

Property testing with memory constraints has also been studied in the setting of streaming algorithms as well. Streaming algorithms were first studied in a unified way starting from the seminal work of Alon et al ([3]) where the authors studied the problem of estimating frequency moments. There is a vast amount of literature available on streaming algorithms (see [21, 20]). Bathie et al ([4]) have studied property testing in the streaming model for testing regular languages. Czumaj et al ([16]) show that every constant-query testable property on bounded-degree graphs can be tested on a random-order stream with constant space. Since this line of work is not directly relevant to our work in this paper, we will not delve deeper into it here.

1.2 Our results

In this work, we study the trade-off between sample complexity and space complexity in both the standard access model and the conditional access model. In the standard access model, a set of samples can be drawn independently from an unknown distribution. In the case of the conditional access model, a subset of the domain is given and samples can be drawn from an unknown distribution conditioned on the given set. This is similar to a streaming algorithm where the samples are presented to the algorithm, and the algorithm has a memory constraint of m bits; i.e., only up to m bits of samples can be stored in memory.

In the standard access model, which we will refer to as SAMP , we have a distribution D over the support $\{1, 2, \ldots, n\}$ and the element i is sampled with probability $D(i)$. In the conditional access model, which we will refer to as COND , the algorithm can choose a set

$S \subseteq \{1, 2, \ldots, n\}$ and will obtain samples from the conditional distribution over the set. I.e. the sample $i \in S$ is returned with probability $D(i)/D(S)$. In this work, we will work with the case when the conditioning is done on sets of size at most two - we will refer to this conditional oracle as PCOND ([12]).

Our results are stated below.

- We propose a memory-efficient identity testing algorithm in the PCOND model when the algorithm is restricted by the memory available to store the samples. We adapt the algorithm of Canonne et al ([12]) and reduce the memory requirement by using the CountMin sketch ([15]) for storing the frequencies of the samples. The identity testing algorithm uses $O(\log^2 n \log\log n/m\epsilon^2)$ samples from standard access model where $\frac{\log n\sqrt{\log\log n}}{\epsilon} \leq m \leq \frac{\log^2 n}{\epsilon}$ and an $\tilde{O}(\log^4 n/\epsilon^4)$ samples from conditional access model and does the following, if $D = D^*$, it returns Accept with probability at least $2/3$, and if $d_{TV}(D, D^*) \geq \epsilon$, it returns Reject with probability at least $2/3$. It uses only $O(\frac{m}{\epsilon})$ bits of memory.

  We also observe that by applying oblivious decomposition [8], performing identity and closeness testing on monotone distributions over $[n]$ can be reduced to performing the corresponding tasks on arbitrary distributions over $[O(\log{(n\epsilon)}/\epsilon)]$. We use the streaming model based identity tester from ([17]) and obtain an $O(\log{(n\epsilon)}\log\log{(n\epsilon)}/m\epsilon^5)$ standard access query identity tester for monotone distributions where $\log\log{(n\epsilon)}/\epsilon^2 \leq m \leq (\log{(n\epsilon)}/\epsilon)^{9/10}$. Their closeness testing algorithm also implies a closeness tester for monotone distributions which uses $O(\log{(n\epsilon)}\sqrt{\log\log{(n\epsilon)}}/\sqrt{m}\epsilon^3)$ samples from standard access model, where $\log\log(n\epsilon) \leq m \leq \tilde{\Theta}(min(\log{(n\epsilon)}/\epsilon, \log^{2/3}{(n\epsilon)}/\epsilon^2))$. Both testers require m bits of memory.
- We adapt the idea of the *bipartite collision tester* ([17]) and give an algorithm that uses $O(\frac{n\log n}{m\epsilon^8})$ samples from SAMP and tests if the distribution is monotone or far from being monotone. This algorithm requires only $O(m)$ bits of memory for $\log^2 n/\epsilon^6 \leq m \leq \sqrt{n}/\epsilon^3$. This upper bound is nearly tight since we observe that the lower bound for uniformity testing proved by Diakonikolas et al ([17]) applies to our setting as well. In particular, we show that the "no" distribution that is used in [17] is actually far from monotone, and hence the lower bound directly applies in our setting as well.
- We extend the idea of the previous algorithm for learning and testing a more general class of distribution called (γ, L)-decomposable distribution, which includes monotone and k-modal distributions. Our algorithm takes $O(\frac{nL\log{(1/\epsilon)}}{m\epsilon^9})$ samples from D and needs $O(m)$ bits of memory where $\log n/\epsilon^4 \leq m \leq O(\sqrt{n\log n}/\epsilon^3)$.

2 Notation and Preliminaries

Throughout this paper, we study distributions D that are supported over the set $\{1, 2, \ldots, n\} = [n]$. The notion of distance between distributions will be *total variation distance* or *statistical distance* which is defined as follows: for two distributions D_1 and D_2, the total variation distance, denoted by $d_{TV}(D_1, D_2) = \frac{1}{2}|D_1 - D_2|_1 = \frac{1}{2}\sum_{i\in[n]}|D_1(x) - D_2(x)| = max_{S\subseteq[n]}((D_1(S) - D_2(S))$. We will use $\mathcal{U}$ to denote the uniform distribution over $[n]$. We use $|.|_1$ for the ℓ_1 norm, $||.||_2$ for the ℓ_2 norm.

Let D_1 and D_2 be two distributions over $[n]$, if $d_{TV}(D_1, D_2) \leq \epsilon$, for some $0 \leq \epsilon \leq 1$, we say that D_1 is ϵ close to D_2. Let $\mathcal{D}$ be the set of all probability distributions supported on $[n]$. A property $\mathcal{P}$ is a subset of $\mathcal{D}$. We say that a distribution D is ϵ far from $\mathcal{P}$, if D is ϵ far from all the distributions having the property $\mathcal{P}$. I.e. $d_{TV}(D, D') > \epsilon$ for every $D' \in \mathcal{P}$.

We define the probability of self-collision of the distribution D by $||D||_2$. For a set S of samples drawn from D, coll(S) defines the pairwise collision count between them. Consider $S_1, S_2 \subset S$, the *bipartite collision* of D with respect to S is defined by coll(S_1, S_2) is the number of collision between S_1 and S_2.

We will be using the count of collisions among sample points to test closeness to uniformity. The following lemma connects the collision probability and the distance to uniformity.

▶ **Lemma 1** ([6]). *Let D be a distribution over $[n]$. If $\max_x D(x) \leq (1+\epsilon). \min_x D(x)$ then $||D||_2^2 \leq (1+\epsilon^2)/n$. If $||D||_2^2 \leq (1+\epsilon^2)/n$ then $d_{TV}(D, \mathcal{U}) \leq \epsilon$.*

One way to test the properties of distributions is to first learn an explicit description of the distribution. We now define the notion of flattened and reduced distributions that will be useful towards this end.

▶ **Definition 2** (Flattened and reduced distributions). *Let D be a distribution over $[n]$, and there exists a set of partitions of the domain into ℓ disjoint intervals, $\mathcal{I} = \{I_j\}_{j=1}^{\ell}$. The flattened distribution $(D^f)^{\mathcal{I}}$ corresponding to D and $\mathcal{I}$ is a distribution over $[n]$ defined as follows : for $j \in [\ell]$ and $i \in I_j$; $(D^f)^{\mathcal{I}}(i) = \frac{\sum_{t \in I_j} D(t)}{|I_j|}$. A reduced distribution D^r is defined over $[\ell]$ such that $\forall i \in \ell, D^r(i) = D(I_i)$.*

If a distribution D is ϵ close to its flattened distribution according to some partition $\{\mathcal{I}_j\}_{j=1}^{\ell}$, we refer D to be (ϵ, ℓ)-flattened. We note that if a distribution is monotonically non-increasing, then its flattened distribution is also monotonically non-increasing but its reduced distribution is not necessarily the same.

The following folklore result shows that the empirical distribution is close to the actual distribution provided sufficient number of samples are taken.

▶ **Lemma 3** (Folklore). *Given a distribution D supported over $[n]$ and an interval partition $\mathcal{I} = \{I_1, ..., I_\ell\}$, using $S = O(\frac{\ell^2}{\epsilon^2} \log \ell)$ points from SAMP, we can obtain an empirical distribution $\tilde{D}$ in the following way: $\forall I_j \in \mathcal{I}; \tilde{D}(I_j) = \frac{occ(S, I_j)}{|S|}$ ($occ(S, I_j)$ is the number of samples from S lies inside I_j) over $[\ell]$ such that for all interval I_j, with probability at least $2/3$, $|D(I_j) - \tilde{D}(I_j)| \leq \frac{\epsilon}{\ell}$. Moreover, let the flattened distribution of D be $(D^f)^{\mathcal{I}}$ and the flattened distribution of $\tilde{D}$ be $(\tilde{D}^f)^{\mathcal{I}}$, we can say that $d_{TV}((D^f)^{\mathcal{I}}, (\tilde{D}^f)^{\mathcal{I}}) < \epsilon$.*

While designing a tester for monotonicity, we use the following theorem due to Birge ([8])

▶ **Lemma 4** (Oblivious partitioning [8]). *Let D be a non-increasing distribution over $[n]$ and $\mathcal{I} = \{I_1, ..., I_\ell\}$ is an interval partitioning of D such that $|I_j| = (1+\epsilon)^j$, for $0 < \epsilon < 1$, then $\mathcal{I}$ has the following properties,*

- $\ell = O(\frac{1}{\epsilon} \log n\epsilon)$
- *The flattened distribution corresponding to $\mathcal{I}$, $(D^f)^{\mathcal{I}}$ is ϵ-close to D, or D is (ϵ, ℓ)-flattened.*

Next, we describe a data structure called the CountMin sketch which is used to estimate the frequencies of elements in a one-pass stream. It was introduced by Cormode et al ([15]). As we are dealing with a one-pass streaming algorithm with a memory constraint, it would be important to store samples in less space. CountMin sketch uses hash functions to store frequencies of the stream elements in sublinear space and returns an estimate of the same.

▶ **Definition 5** (CountMin sketch). *A CountMin (CM) sketch with parameters (ϵ, δ) is represented by a two-dimensional array counts with width w and depth d: $count[1,1], ..., count[d, w]$. Set $w = \frac{e}{\epsilon}$ and $d = \log 1/\delta$. Each entry of the array is initially zero. Additionally, d hash* *functions $h_1, ..., h_d : \{1, ..., n\} \rightarrow \{1, ..., w\}$ chosen uniformly at random from a pairwise-independent family. The space requirement for the count min sketch is wd words. The sketch can be queried for the frequency of an element from the universe $\mathcal{U}$ of elements, and will return an estimate of its frequency.*

The lemma below captures the fact that the frequency of any element x_i can be estimated from a CountMin sketch.

▶ **Lemma 6** ([15]). *Let $\{x_1, ..., x_S\}$ be a stream of length S and f_{x_i} be the actual frequency of an element x_i. Suppose $\tilde{f}_{x_i}$ be the stored frequency in count min sketch, then the following is true with probability at least $(1 - \delta)$, $f_{x_i} \leq \tilde{f}_{x_i} \leq f_{x_i} + \epsilon S$.*

3 Testing identity in the streaming model using PCOND

In this section, we revisit the identity testing problem using PCOND queries: given sample access and PCOND query access to an unknown distribution D we have to test whether D is identical to a fully specified distribution D^* or they are ϵ far from each other. Canonne et al ([12]) address the problem and propose a PCOND query-based identity tester. In their algorithm, the domain of D^* is divided into a set of "buckets" where the points are having almost the same weights. The algorithm samples $\tilde{O}(\log^2 n/poly(\epsilon))$ points from D and estimates the weight of each bucket. They prove if D and D^* are far then there exists at least one bucket where the weight of D^* and weight of $\tilde{D}$ will differ. If not, then the algorithm runs a process called *Compare* to estimate the ratio of the weight of each pair of points (y, z) where y is taken from a set of samples drawn from D^* and z is taken from a set of samples according to D. The following lemma is used to compare the weights of two points.

▶ **Lemma 7** ([12]). *Given as input two disjoint subsets of points X, Y together with parameters $\eta \in (0, 1], K \geq 1$ and $\delta \in (0, \frac{1}{2}]$ as well as COND query access to a distribution D, there exists a procedure* Compare *which estimates the ratio of the weights of two sets and either outputs a value $\rho > 0$ or outputs High or Low and satisfies the following:*

- *If $D(X)/K \leq D(Y) \leq K \cdot D(X)$ then with probability at least $1 - \delta$ the procedure outputs a value $\rho \in [1 - \eta, 1 + \eta]D(Y)/D(X)$;*
- *If $D(Y) > K \cdot D(X)$ then with probability at least $1 - \delta$ the procedure outputs either High or a value $\rho \in [1 - \eta, 1 + \eta]D(Y)/D(X)$;*
- *If $D(Y) < D(X)/K$ then with probability at least $1 - \delta$ the procedure outputs either Low or a value $\rho \in [1 - \eta, 1 + \eta]D(Y)/D(X)$.*

The procedure performs $O(\frac{K \log 1/\delta}{\eta^2})$ conditional queries on the set $X \cup Y$.

However, for storing $\tilde{O}(\log^2 n/poly(\epsilon))$ samples for estimating the weights of the buckets, an $\tilde{O}(\log^3 n/poly(\epsilon))$ space is required considering each sampled point takes $\log n$ bits of memory. As we are dealing with a memory constraint of m bits, for $m < O(\log^3 n)$, implementing the algorithm is not memory efficient. We use the main idea of Canonne et al ([12]), but instead of storing all samples, we use the CountMin sketch data structure for storing the frequencies of the elements of the stream. Later, the frequencies are used to estimate the weight of each bucket. By choosing the parameters of the CountMin sketch suitably, the total space required for our algorithm is at most $O(m/\epsilon)$ bits. The main concept of our algorithm lies in the theorem below,

▶ **Theorem 8** (Testing Identity [12]). *There exists an identity tester that uses an $\tilde{O}(\log^4 n/\epsilon^4)$ PCOND queries and does the following: for every pair of distributions D, D^* over $[n]$, where D^* is fully specified, the algorithm outputs Accept with probability at least $2/3$ if $D = D^*$ and outputs Reject with probability at least $2/3$ if $d_{TV}(D, D^*) \geq \epsilon$.*

Before moving into the algorithm, we define the *bucketization* technique according to ([12]). For an explicit distribution D^*, the domain is divided into ℓ buckets $\mathcal{B} = \{B_1, ..., B_\ell\}$, where each bucket contains a set of points which satisfies $B_j = \{i \in [n] : 2^{j-1}\eta/n \le D^*(i) \le 2^j\eta/n\}$ and $B_0 = \{i \in [n] : D^*(i) < \eta/n\}$, where $\eta = \epsilon/c$ for c to be a constant. The number of buckets $\ell = O(\lceil \log n/\eta + 1\rceil + 1)$.

We are now ready to present our PCOND query-based one-pass streaming algorithm for identity testing. Our algorithm and the correctness borrow from ([12]) with the extra use of CountMin sketches to improve the trade-off between the sample complexity and the space used.

Algorithm 1 PCOND Identity Testing Streaming.

```
Input  : SAMP and PCOND access to $D$, an explicit distribution $D^*$, parameters
         $0 < \epsilon \le 1$, $\eta = \epsilon/6$, $\ell$ buckets of $D^*$, space requirement $O(m)$ bits
         $\frac{\log n\sqrt{\log\log n}}{\epsilon} \le m \le \frac{\log^2 n}{\epsilon}$
Output : Accept if $D = D^*$, Reject if $d_{TV}(D, D^*) \ge \epsilon$
1  Sample $S = O(\frac{\log^2 n \log\log n}{m\epsilon^2})$ points $\{x_1, ..., x_S\}$ from SAMP
2  for ($i = 1$ to $S$) do
3      Estimate the frequency of $x_i$ using CountMin sketch $(\frac{\epsilon}{m}, \frac{1}{100})$ such that
       $f_{x_i} \le \tilde{f}_{x_i} \le f_{x_i} + \frac{\epsilon}{m}S$
4  Define the frequency of each bucket $B_j$ to be $f_{B_j} = \sum_{x_i \in B_j} f_{x_i}$, such that
   $f_{B_j} \le \tilde{f}_{B_j} \le f_{B_j} + \frac{\epsilon}{m}S^2$
5  if $\frac{\tilde{f}_{B_j}}{S} < D^*(B_j) - \frac{\sqrt{m\epsilon}}{\log n}$ or $\frac{\tilde{f}_{B_j}}{S} > D^*(B_j) + \frac{\sqrt{m\epsilon}}{\log n} + \frac{\log^2 n \log\log n}{\epsilon m^2}$ then
6      Reject and Exit
7  Select $s = O(\ell/\epsilon)$ points $\{y_1, ..., y_s\}$ from $D^*$
8  for each $y_k \in s$ do
9      Sample $s$ points $\{z_1, ..., z_s\}$ from $D$ as a stream
10     for each pair of points $(y_k, z_l)$ such that $\frac{D^*(y_k)}{D^*(z_l)} \in [1/2, 2]$ do
11         Run Compare $(y_k, z_l, \eta/4\ell, 2, 1/10s^2)$)
12         if Compare returns Low or a value smaller than $(1 - \eta/2\ell)\frac{D^*(y_k)}{D^*(z_l)}$ then
13             Reject and Exit
14 Accept
```

▶ **Theorem 9.** *The algorithm* PCOND IDENTITY TESTING STREAMING *uses an* $O(\log^2 n \log\log n/m\epsilon^2)$ *length stream of standard access query points and an* $\tilde{O}(\log^4 n/\epsilon^4)$ *length of conditional stream and does the following, If* $D = D^*$*, it returns* Accept *with probability at least* $2/3$*, and if* $d_{TV}(D, D^*) \ge \epsilon$*, it returns* Reject *with probability at least* $2/3$*. The memory requirement for the algorithm is* $O(\frac{m}{\epsilon})$ *(due to the parameters set in* CountMin *sketch) where* $\frac{\log n\sqrt{\log\log n}}{\epsilon} \le m \le \frac{\log^2 n}{\epsilon}$.

Proof.

Completeness. Suppose $D = D^*$. We prove that the algorithm does not return Reject in Line 6. Let $\tilde{D}(B_j)$ be the estimated weight of a bucket B_j where $\tilde{D}(B_j) = \frac{f_{B_j}}{S}$ for $S = O(\log^2 n \log\log n/m\epsilon^2)$. An additive Chernoff bound [followed by a union bound over the buckets] shows that with high probability, $\forall B_j, |D(B_j) - \tilde{D}(B_j)| \le \frac{\sqrt{m\epsilon}}{\log n}$. Using Lemma 6, with probability at least 99/100, for every element x_i in the stream, $f_{x_i} \leq \tilde{f}_{x_i} \leq f_{x_i} + \frac{\epsilon S}{m}$. Summing over all the elements in a bucket B_j, we get $\tilde{f}_{B_j} - \frac{\epsilon}{m} S^2 \leq f_{B_j} \leq \tilde{f}_{B_j}$. Substituting $\tilde{D}(B_j) = \frac{f_{B_j}}{S}$, we can see that $\frac{\tilde{f}_{B_j}}{S} - \frac{\epsilon S}{m} \leq \tilde{D}(B_j) \leq \frac{\tilde{f}_{B_j}}{S}$. As $D = D^*$, $\tilde{D}(B_j)$ is a good estimate of $D^*(B_j)$. Using $|D^*(B_j) - \tilde{D}(B_j)| \leq \frac{\sqrt{m\epsilon}}{\log n}$, we get $\frac{\tilde{f}_{B_j}}{S} - \frac{\epsilon S}{m} - \frac{\sqrt{m\epsilon}}{\log n} \leq D^*(B_j) \leq \frac{\tilde{f}_{B_j}}{S} + \frac{\sqrt{m\epsilon}}{\log n}$. This can be written as $D^*(B_j) - \frac{\sqrt{m\epsilon}}{\log n} \leq \frac{\tilde{f}_{B_j}}{S} \leq D^*(B_j) + \frac{\sqrt{m\epsilon}}{\log n} + \frac{\log^2 n \log\log n}{\epsilon m^2}$ by replacing $S = O(\log^2 n \log\log n / m\epsilon^2)$. Hence, the algorithm will not output Reject with high probability. As $D = D^*$, for all pairs (y_k, z_l) such that $\frac{D^*(y_k)}{D^*(z_l)} \in [1/2, 2]$, it follows from Lemma 7 that the estimated ratio of weights of each pair (y_k, z_l) is less than $(1 - \eta/2\ell)\frac{D^*(y_k)}{D^*(z_l)}$ [for $\eta = \epsilon/6$] with probability at most $1/10s^2$. A union bound over all $O(s^2)$ pairs proves that with a probability of at least 9/10 the algorithm outputs Accept .

Soundness. Let $d_{TV}(D, D^*) \geq \epsilon$. In this case, if one of the estimates of $\tilde{f}_{B_j}$ passes Line 5, the algorithm outputs Reject . Let's assume that the estimates are correct with high probability. The rest of the analysis follows from ([12]), we give a brief outline of the proof for making it self-contained. Define high-weight and low-weight buckets in the following way, for $\eta = \epsilon/6$, as follows: $H_j = \{x \in B_j : D(x) > D^*(x) + \eta/\ell|B_j|\}$, and $L_j = \{x \in B_j : D(x) \leq D^*(x) - \eta/\ell|B_j|\}$. It can be shown that at least one point will occur from the low-weight bucket while sampling s points in Line 7 and at least one point will come from the high-weight bucket while obtaining s points in Line 9. Using the definition of high-weight and low-weight buckets, there exists a pair (y_k, z_l) such that $D(y_k) \leq (1 - \eta/2\ell)D^*(y_k)$ and $D(z_k) > (1 + \eta/2\ell)D^*(z_k)$. By Lemma 7, with probability at least $1 - 1/10s^2$, *Compare* will return low or a value at most $(1 - \eta/2\ell)\frac{D^*(y_k)}{D^*(z_l)}$ in Line 12. Hence, the algorithm outputs Reject with high probability. ◀

We use CountMin sketch with parameters $(\frac{\epsilon}{m}, \frac{1}{100})$ in our algorithm. Comparing it with (ϵ, δ) CountMin sketch defined in ([15]), we set the width of the array to be $w = em/\epsilon$ and depth $d = \log 100$. So the space required for the algorithm is $w.d$ words which imply $O(\frac{m}{\epsilon})$ bits. For running the *Compare* procedure, we are not using any extra space for storing samples. This is because for every element in $\{y_1, ..., y_s\}$ we are sampling s length stream $\{z_1, ..., z_s\}$ and running *Compare* for each pair of points taken from each stream respectively. This leads to running compare process s^2 times. A single run of compare works in the following way in the streaming settings, for a pair (y_k, z_l), sample $O(\log^2 n/\epsilon^2)$ points from D conditioned on (y_k, z_l) and keep two counters for checking the number of times each of them appeared in the stream. Each round of *Compare* process requires $O(\log^2 n/\epsilon^2)$ length of the stream. Hence, the total stream length is $\tilde{O}(\log^4 n/\epsilon^4)$.

4 Testing Monotonicity in the streaming model using SAMP

In this section, we give an algorithm for testing monotonicity in the SAMP model when the samples are obtained via a one-pass stream. The algorithm of Batu et al ([5]) provides a sample-efficient algorithm for testing monotonicity, by dividing the support $[n]$ into intervals which are either low-weight or close to uniform. In our case, we start with the oblivious decomposition of Birge ([8]) and check if the total weight of the intervals that are far from uniform is small. To check if an interval is far from uniform, we count the number of collisions in the sample obtained from the interval. To improve the space complexity of the algorithm, we modify the part of counting collisions to counting bipartite collisions, like in ([17]). We now describe the algorithm for testing monotonicity using bipartite collisions. The sample complexity for this algorithm is worse than the algorithm of Batu et al ([5]), but we will then show that this can be converted to an algorithm in the one-pass streaming model with better space complexity.

4.1 Testing Monotonicity using Bipartite Collisions

In this section, we perform the monotonicity testing in a slightly different fashion which functions as the building block of a streaming-based monotonicity tester. Here, unlike counting pairwise collisions between the samples, we divide the samples into two sets and count the bipartite collisions between them. The idea of the bipartite collision tester is adapted from ([17]). A key Lemma 11 proves how the bipartite collision is used to estimate the collision probability. Given sample access to an unknown distribution D over $[n]$, first, we divide the domain according to the oblivious decomposition. We count the bipartite collisions inside the intervals where enough samples lie. If D is monotone, the total weight of high collision intervals can not be too high. Prior to describing the algorithm, the lemma below clarifies the fact "enough samples" and the intervals holding them.

▶ **Lemma 10.** *Let D be a distribution over $[n]$, and $\mathcal{I} = \{I_1, ..., I_\ell\}$ be an interval partitions of $[n]$. Let $\mathcal{J} \subset \mathcal{I}$ be the set of intervals and for all $I_j \in \mathcal{J}$, $D(I_j) \geq \epsilon_1/\log n$, where $\epsilon_1 = \epsilon^2$. If $S = O(\frac{n \log n}{\epsilon^8})$ samples are drawn according to D, then all $I_j \in \mathcal{J}$ contain $|S_{I_j}| \geq O(|I_j|/\epsilon^4)$ samples.*

Proof. Fix an I_j and define a random variable, $X_i = 1$ if i^{th} sample is in I_j else 0. Let $X = \sum_{i=1}^{S} X_i = S_{I_j}$. Then the expectation $\mathbb{E}[X] = |S| \cdot D(I_j) \geq \frac{|S|\epsilon_1}{\log n}$.

By Chernoff bound, we can see that $Pr\Big[X < (1-\epsilon)\frac{|S|\epsilon_1}{\log n}\Big] = Pr\Big[X < (1-\epsilon)\mathbb{E}[X]\Big] \leq e^{-\epsilon^2 \mathbb{E}[X]} \leq e^{-\epsilon^2 \frac{|S|\epsilon^2}{\log n}} < \frac{\epsilon^2}{10 \log n}$.

The last inequality is obtained from the fact that $|S| = O(\frac{n \log n}{\epsilon^8})$ and using $\frac{n}{\epsilon^4} > \log(10 \log n/\epsilon^2)$. Applying union bound over all $\ell = O(\frac{\log n}{\epsilon_1})$ partitions, we can conclude that, $[\epsilon_1 = \epsilon^2]$ $\forall I_j$; such that $D(I_j) \geq \frac{\epsilon_1}{\log n}$ with probability at least 9/10, the following happens, $S_{I_j} \geq (1-\epsilon)\frac{|S|\epsilon_1}{\log n} \geq (1-\epsilon)\frac{n}{\epsilon^6} \geq O(|I_j|/\epsilon^4)$ ◀

The main intuition behind our algorithm is counting the bipartite collision between a set of samples. The next lemma, defines the necessary conditions for estimating collision probability using bipartite collision count.

▶ **Lemma 11.** *Let D be an unknown distribution over $[n]$ and S be the set of samples drawn according to SAMP . Let $I \subset [n]$ be an interval and S_I be the set of points lying in the interval I. Let S_I be divided into two disjoint sets S_1 and S_2; $\{S_1\} \cup \{S_2\} = \{S_I\}$ such that $|S_1| \cdot |S_2| \geq O(|S_I|/\epsilon^4)$, then with probability at least 2/3,*

$$||D_I||_2^2 - \frac{\epsilon^2}{64|I|} \leq \frac{coll(S_1, S_2)}{|S_1||S_2|} \leq ||D_I||_2^2 + \frac{\epsilon^2}{64|I|}.$$

Proof. Define the random variable $X_{ij} = 1$ if i^{th} sample in S_1 is same as j^{th} sample in S_2, 0 otherwise.

$$X = \sum_{(i,j) \in S_1 \times S_2} X_{ij} = coll(S_1, S_2)$$

$$\mathbb{E}[X] = |S_1| \cdot |S_2| \cdot ||D_I||_2^2$$

Where $||D_I||_2$ is collision probability. Let $Y_{ij} = X_{ij} - \mathbb{E}[X_{ij}] = X_{ij} - ||D_I||_2^2$.

$$\begin{aligned} Var[\sum_{(i,j)\in S_1\times S_2} X_{ij}] &= \mathbb{E}\Big[(\sum_{(i,j)\in S_1\times S_2} Y_{ij})^2\Big] \\ &= \mathbb{E}\Big[\sum_{(i,j)\in S_1\times S_2} Y_{ij}^2 + \sum_{(i,j)\neq(k,l);|\{i,j,k,l\}|=3} Y_{ij}Y_{kl}\Big] \end{aligned}$$

We calculate the following,

$$\begin{aligned} \mathbb{E}[Y_{ij}^2] &= \mathbb{E}[X_{ij}^2] - 2(\mathbb{E}[X_{ij}])^2 + (\mathbb{E}[X_{ij}])^2 \\ &= ||D_I||_2^2 - ||D_I||_2^4 \\ \mathbb{E}[Y_{ij}Y_{kl}] &= \mathbb{E}\Big[(X_{ij} - ||D_I||_2^2)(X_{kl} - ||D_I||_2^2)\Big] \\ &= \mathbb{E}\Big[X_{ij}X_{kl}\Big] - ||D_I||_2^2(\mathbb{E}[X_{ij}] + \mathbb{E}[X_{kl}]) + ||D_I||_2^4 \\ &= \mathbb{E}\Big[X_{ij}X_{kl}\Big] - ||D_I||_2^4 \end{aligned}$$

Now,

$$\begin{aligned} Var[\sum_{(i,j)\in S_1\times S_2} X_{ij}] &= \sum_{(i,j)\in S_1\times S_2} (||D_I||_2^2 - ||D_I||_2^4) + \\ &\quad \sum_{(i,j)\neq(k,l);|\{i,j,k,l\}|=3} (\mathbb{E}\Big[X_{ij}X_{kl}\Big] - ||D_I||_2^4) \\ &= |S_1|.|S_2|(||D_I||_2^2 - ||D_I||_2^4) + \sum_{(i,j);(k,j)\in S_1\times S_2;i\neq k} \mathbb{E}\Big[X_{ij}X_{kj}\Big] \\ &\quad + \sum_{(i,j);(i,l)\in S_1\times S_2;j\neq l} \mathbb{E}\Big[X_{ij}X_{il}\Big] - \sum_{(i,j)\neq(k,l);|\{i,j,k,l\}|=3} ||D_I||_2^4 \\ &= |S_1|.|S_2|(||D_I||_2^2 - ||D_I||_2^4) + |S_2|\binom{|S_1|}{2}||D_I||_3^3 \\ &\quad + |S_1|\binom{|S_2|}{2}||D_I||_3^3 - \Big(|S_2|\binom{|S_1|}{2} + |S_1|\binom{|S_2|}{2}\Big)||D_I||_2^4 \\ &\leq |S_1||S_2|\Big[(||D_I||_2^2 - ||D_I||_2^4) + (|S_1| + |S_2|)(||D_I||_3^3 - ||D_I||_2^4)\Big] \end{aligned}$$

Applying Chebyshev's inequality, we get,

$$\begin{aligned} &Pr[|X - \mathbb{E}[X]| > \frac{\epsilon^2}{64|I|}|S_1||S_2|] \leq \frac{64^2 Var[X]|I|^2}{\epsilon^4|S_1|^2|S_2|^2} \\ &\leq \frac{|S_1||S_2|\Big[(||D_I||_2^2 - ||D_I||_2^4) + (|S_1| + |S_2|)(||D_I||_3^3 - ||D_I||_2^4)\Big]64^2|I|^2}{\epsilon^4|S_1|^2|S_2|^2} \\ &\leq \frac{\Big[||D_I||_2^2 - ||D_I||_2^4 + (|S_1| + |S_2|)(||D_I||_2^3 - ||D_I||_2^4)\Big]64^2|I|^2}{\epsilon^4|S_1|.|S_2|} \\ &\leq \frac{\Big[||D_I||_2^2 - ||D_I||_2^4 + (|S_1| + |S_2|)(||D_I||_2^2 - ||D_I||_2^4)\Big]64^2|I|^2}{\epsilon^4|S_1|.|S_2|} \\ &\leq \frac{||D_I||_2^2\Big[1 - ||D_I||_2^2 + (|S_1| + |S_2|)(1 - ||D_I||_2^2)\Big]64^2|I|^2}{\epsilon^4|S_1|.|S_2|} \\ &\leq \frac{||D_I||_2^2\Big(1 - ||D_I||_2^2\Big)\Big(1 + |S_1| + |S_2|\Big)64^2|I|^2}{\epsilon^4|S_1|.|S_2|} \end{aligned}$$

Where the third inequality uses the fact that $||D_I||_3 \leq ||D_I||_2$ and the fourth inequality uses the fact that $||D_I||_2^3 \leq ||D_I||_2^2$ as $||D_I||_2 \in (0,1]$. To make the probability $< 1/3$, we have,

$$\begin{aligned}
|S_1|.|S_2| &\geq 3 \times 64^2 |I|^2 \frac{1}{\epsilon^4} ||D_I||_2^2 \Big(1 - ||D_I||_2^2\Big) \Big(1 + |S_1| + |S_2|\Big) \\
&\geq 3 \times 64^2 \frac{|I|^2}{\epsilon^4} ||D_I||_2^2 \frac{||D_I||_2^2}{100} \Big(|S_1| + |S_2|\Big) \\
&\geq 3 \times 64^2 \frac{1}{100\epsilon^4} \Big(|S_1| + |S_2|\Big) \\
&\geq O(\frac{S_I}{\epsilon^4})
\end{aligned}$$

In the second inequality we have used the fact that $(1 - ||D_I||_2^2) \geq \frac{1}{100}||D_I||_2^2$ as $||D_I||_2^2 \leq \frac{100}{101} < 1$. The third inequality is obtained from the fact that $||D||_2^2 \geq \frac{1}{|I|}$. The final inequality is obtained from the fact that $|S_I| = |S_1| + |S_2|$. Therefore, provided $|S_1|.|S_2| \geq O(\frac{|S_I|}{\epsilon^4})$, with probability at least $2/3$, $||D_I||_2^2 - \frac{\epsilon^2}{64|I|} \leq \frac{coll(S_1,S_2)}{|S_1||S_2|} \leq ||D_I||_2^2 + \frac{\epsilon^2}{64|I|}$. ◀

The bipartite collision-based tester works by verifying the total weight of the intervals where the conditional distributions are far from uniformity. Let S_I be the set of samples inside an interval I and let it satisfy the condition of Lemma 11. The following lemma shows that bipartite collision count is used to detect such intervals.

▶ **Lemma 12.** *Let D be an unknown distribution over $[n]$ and $I \subset [n]$ is an interval. Let S_I be the set of points lying in the interval I and S_I can be divided into two sets S_1 and S_2 such that $|S_1||S_2| \geq O(|S_I|/\epsilon^4)$, then the following happens with probability at least $2/3$*

- *If $d_{TV}(D_I,\mathcal{U}_I) > \frac{\epsilon}{4}$, then $\frac{coll(S_1,S_2)}{|S_1||S_2|} > \frac{1}{|I|} + \frac{\epsilon^2}{64|I|}$*
- *If $d_{TV}(D_I,\mathcal{U}_I) \leq \frac{\epsilon}{4}$, then, $\frac{coll(S_1,S_2)}{|S_1||S_2|} \leq \frac{1+\epsilon^2/64}{|I|} + \frac{\epsilon^2}{16}$*

Proof. Let, $d_{TV}(D_I,\mathcal{U}_I) > \frac{\epsilon}{4}$, squaring both sides, we get $(d_{TV}(D_I,\mathcal{U}_I))^2 > \frac{\epsilon^2}{16} > \frac{\epsilon^2}{32}$. Using the fact that $d_{TV}(D_I,\mathcal{U}_I) \leq \sqrt{|I|} \cdot ||D_I - \mathcal{U}_I||_2$, we deduce $|I| \cdot ||D_I - \mathcal{U}_I||_2^2 > \frac{\epsilon^2}{32}$. Simplifying the inequality, we get $||D_I - \mathcal{U}_I||_2^2 > \frac{\epsilon^2}{32|I|}$. Now, we obtain the following inequality by using $||D_I - \mathcal{U}_I||_2^2 = ||D_I||_2^2 - \frac{1}{|I|}$.

$$\begin{aligned}
||D_I||_2^2 - \frac{1}{|I|} &> \frac{\epsilon^2}{32|I|} \\
||D_I||_2^2 &> \frac{\epsilon^2}{32|I|} + \frac{1}{|I|}
\end{aligned}$$

Consider S_I is divided into two sets so that $|S_1| \cdot |S_2| \geq O(|S_I|/\epsilon^4)$, by Lemma 11 we obtain,

$$\begin{aligned}
\frac{coll(S_1,S_2)}{|S_1||S_2|} + \frac{\epsilon^2}{64|I|} &> \frac{\epsilon^2}{32|I|} + \frac{1}{|I|} \\
\frac{coll(S_1,S_2)}{|S_1||S_2|} &> \frac{1}{|I|} + \frac{\epsilon^2}{64|I|}
\end{aligned}$$

Similarly, when $d_{TV}(D_I,\mathcal{U}_I) \leq \frac{\epsilon}{4}$, we get $||D_I||_2^2 \leq \frac{\epsilon^2}{16} + \frac{1}{|I|}$. Given S_I can be divided into two sets such that $|S_1| \cdot |S_2| \geq O(|S_I|/\epsilon^4)$, by Lemma 11, $\frac{coll(S_1,S_2)}{|S_1| \cdot |S_2|} \leq \frac{1+\epsilon^2/64}{|I|} + \frac{\epsilon^2}{16}$. ◀

Now, we present the bipartite collision-based monotonicity tester.

Algorithm 2 Bipartite Collision Monotonicity.

```
  Input   : SAMP access to $D$, $\ell = O(\frac{1}{\epsilon_1}\log(n\epsilon_1 + 1))$ oblivious partitions
            $\mathcal{I} = \{I_1, .., I_\ell\}$ and error parameter $\epsilon, \epsilon_1 \in (0,1]$, where $\epsilon_1 = \epsilon^2$
  Output  : Accept if $D$ is monotone, Reject if $D$ is not $7\epsilon$ close to monotone
1 Sample $T = O(\frac{1}{\epsilon^6}\log^2 n \log\log n)$ points from SAMP
2 Get the empirical distribution $\tilde{D}$ over $\ell$
3 Obtain an additional sample $S = O(\frac{n\log n}{\epsilon^8})$ from SAMP
4 Let $J$ be the set of intervals where the number of samples (in each interval $I_j$) is
    $|S_{I_j}| \geq O(|I_j|/\epsilon^4)$ and $S_{I_j}$ can be partitioned into two disjoint sets $S_1$ and $S_2$ such
    that $|S_1||S_2| \geq O(|I_j|/\epsilon^8)$ and $\frac{coll(S_1,S_2)}{|S_1||S_2|} \geq (\frac{1+\epsilon^2/64}{|I_j|} + \frac{\epsilon^2}{16})$
5 if $\sum_{I_j \in J} \tilde{D}(I_j) > 5\epsilon$ then
6     Reject and Exit
7 Define a flat distribution $(\tilde{D}^f)^{\mathcal{I}}$ over $[n]$
8 Output Accept if $(\tilde{D}^f)^{\mathcal{I}}$ is $2\epsilon$-close to a monotone distribution. Otherwise output
    Reject
```

▶ **Theorem 13.** *The algorithm* BIPARTITE COLLISION MONOTONICITY *uses* $O(\frac{n\log n}{\epsilon^8})$ *SAMP queries and outputs Accept with probability at least* $2/3$ *if* D *is a monotone distribution and outputs Reject with probability at least* $2/3$ *when* D *is not* 7ϵ*-close to monotone.*

Proof. While sampling $O(n\log n/\epsilon^8)$ points according to D, an application of Chernoff bound shows that the intervals with $D(I_j) \geq \epsilon^2/\log n$ will contain at least $S_{I_j} = O(|I_j|/\epsilon^4)$ points. There will be at least one such interval with $D(I_j) \geq \epsilon^2/\log n$ as there are $O(\log n/\epsilon^2)$ partitions.

Completeness. Let D be monotone. By oblivious partitioning with parameter $\epsilon_1 = \epsilon^2$, we have $\sum_{j=1}^{\ell}\sum_{x\in I_j}|D(x) - \frac{D(I_j)}{|I_j|}| \leq \epsilon_1$ which implies $\sum_{j=1}^{\ell} D(I_j) d_{TV}(D_{I_j}, \mathcal{U}_{I_j}) \leq \epsilon^2$. Let J' be the set of intervals where for all I_j, $d_{TV}(D_{I_j}, \mathcal{U}_{I_j}) > \frac{\epsilon}{4}$, then $\sum_{I_j\in J'} D(I_j) \leq 4\epsilon$.

Let $\hat{J}$ is the set of intervals where $|S_1||S_2| \geq O(|S_{I_j}|/\epsilon^4)$ and $d_{TV}(D_{I_j}, \mathcal{U}_{I_j}) > \frac{\epsilon}{4}$. So, $\hat{J} \subseteq J'$. From Lemma 12, we know $\hat{J}$ is the set of intervals where $\frac{coll(S_1,S_2)}{|S_1||S_2|} > \frac{1}{|I_j|} + \frac{\epsilon^2}{64|I_j|}$. Let J be the set of intervals where $|S_1||S_2| \geq O(|S_{I_j}|/\epsilon^4)$ and $\frac{coll(S_1,S_2)}{|S_1||S_2|} > \frac{1+\epsilon^2/64}{|I_j|} + \frac{\epsilon^2}{16}$, then $J \subseteq \hat{J} \subseteq J'$. We know $\sum_{I_j\in J'} D(I_j) \leq 4\epsilon$. So, we can conclude that $\sum_{I_j\in J} D(I_j) \leq 4\epsilon$.

When $d_{TV}(D_{I_j}, \mathcal{U}_{I_j}) \leq \frac{\epsilon}{4}$, the algorithm does not sum over such $D(I_j)$ even if $|S_1||S_2| \geq O(|S_{I_j}|/\epsilon^4)$. This is because by Lemma 12 we know $\frac{coll(S_1,S_2)}{|S_1||S_2|} \leq \frac{1+\epsilon^2/64}{|I_j|} + \frac{\epsilon^2}{16}$. As a result, we can say that when D is monotone $\sum_{I_j\in J} D(I_j) \leq 4\epsilon$.

We use the empirical distribution $\tilde{D}$ and deduce that $\sum_{I_j\in J}\tilde{D}(I_j) \leq 5\epsilon$. Hence, the algorithm will NOT output Reject in Step 6. We also conclude as D is monotone, the flattened distribution $(\tilde{D}^f)^{\mathcal{I}}$ is 2ϵ close to monotone and the algorithm will output Accept in Step 8.

Soundness. We prove the contrapositive of the statement. Let the algorithm outputs Accept , then we need to prove that D is 7ϵ close to monotone.

As the algorithm accepts, $\sum_{I_j\in J}\tilde{D}(I_j) \leq 5\epsilon$, for the set of intervals J where $|S_1||S_2| \geq O(|S_{I_j}|/\epsilon^4)$ and $\frac{coll(S_1,S_2)}{|S_1||S_2|} \geq (\frac{1+\epsilon^2/64}{|I_j|} + \frac{\epsilon^2}{16})$. For all such intervals $I_j \in J$ by Lemma 11, we obtain $d_{TV}(D_{I_j}, \mathcal{U}_{I_j}) \geq \frac{\epsilon}{4}$.

Now, we calculate the distance between D and the flattened distribution and we get $d_{TV}(D, (D^f)^{\mathcal{I}}) < 4\epsilon$

We also know from Lemma 3, $d_{TV}((D^f)^{\mathcal{I}}, (\tilde{D}^f)^{\mathcal{I}}) < \epsilon$. By triangle inequality, $d_{TV}(D, (\tilde{D}^f)^{\mathcal{I}}) < 5\epsilon$. As the algorithm outputs accept, there exists a monotone distribution M, such that $d_{TV}(\tilde{D}^f)^{\mathcal{I}}, M) \leq 2\epsilon$. By triangle inequality, we have $d_{TV}(D, M) < 7\epsilon$. ◀

4.2 Testing Monotonicity in Streaming model

In this section, we present the monotonicity tester in the streaming settings. A set of samples is drawn according to the standard access model that is revealed online one at a time. The task is to test whether an unknown distribution is a monotone or ϵ far from monotonicity. Also, there is a memory bound of m bits. We use the notion of bipartite collision monotonicity tester 2 discussed in the previous section. For satisfying the memory bound, we store an optimal number of samples for such intervals and count bipartite collision between the stored samples and the remaining ones. We present the algorithm below,

Algorithm 3 Streaming Monotonicity.

```
Input  : SAMP access to $D$, $\ell = O(\frac{1}{\epsilon_1}\log(n\epsilon_1 + 1))$ oblivious partitions
         $\mathcal{I} = \{I_1, .., I_\ell\}$ and error parameter $\epsilon, \epsilon_1 \in (0,1]$, where $\epsilon_1 = \epsilon^2$, memory
         requirement $\log^2 n/\epsilon^6 \leq m \leq \sqrt{n}/\epsilon^3$
Sample $T = \tilde{O}(\frac{1}{\epsilon^6}\log^2 n)$ points from SAMP
Get the empirical distribution $\tilde{D}$ over $\ell$
Obtain an additional sample $S = O(\frac{n\log n}{m\epsilon^8})$ from SAMP
For each interval store the first set of $S_1 = O(\frac{m\epsilon^2}{\log^2 n})$ samples in memory
Let $J$ be the set of intervals, where for the next set of $S_2 = O(\frac{n}{m\epsilon^4})$ points, the
  following condition is satisfied, $\frac{coll(S_1,S_2)}{|S_1||S_2|} \geq (\frac{1+\epsilon^2/64}{|I_j|} + \frac{\epsilon^2}{16})$
Check if $\sum_{I_j \in J} \tilde{D}(I_j) > 5\epsilon$ then
    Reject and Exit
Define a flat distribution $(\tilde{D}^f)^{\mathcal{I}}$ over $[n]$
Output Accept if $(\tilde{D}^f)^{\mathcal{I}}$ is $2\epsilon$-close to a monotone distribution. Otherwise output
  Reject
```

▶ **Theorem 14.** *The algorithm* STREAMING MONOTONICITY *uses* $O(\frac{n\log n}{m\epsilon^8})$ *SAMP queries and outputs Accept with probability at least* $2/3$ *if* D *is a monotone distribution and outputs Reject with probability at least* $2/3$ *when* D *is not* 7ϵ *close to monotone. It uses* $O(m)$ *bits of memory for* $\log^2 n/\epsilon^6 \leq m \leq \sqrt{n}/\epsilon^3$.

Proof. As there are $O(\frac{\log n}{\epsilon^2})$ partitions, there will be at least one interval with $D(I_j) \geq \frac{\epsilon^2}{\log n}$. An application of Chernoff bound shows that with high probability all such intervals contain $|S_{I_j}| = O(n/m\epsilon^4)$ points. In the algorithm, we divide S_{I_j} into two sets S_1 and S_2 such that for $\log^2 n/\epsilon^6 \leq m \leq \sqrt{n}/\epsilon^3$, $|S_1| + |S_2| = O(m\epsilon^2/\log^2 n) + O(n/m\epsilon^4) = O(n/m\epsilon^4)$ and $|S_1|.|S_2| = O(n/\epsilon^2\log^2 n) \geq O(n/m\epsilon^8) = (1/\epsilon^4)|S_{I_j}|$. (The inequality is obtained by the fact that $m \geq \log^2 n/\epsilon^6$). This implies that the condition of Lemma 11 is satisfied by these intervals and they are eligible for estimating the collision probability using bipartite collision count. The rest of the analysis follows from Theorem 13.

The algorithm uses $O(m)$ bits of memory for implementation in a single-pass streaming model. For obtaining the empirical distribution $\tilde{D}$, we will use one counter for each of the ℓ intervals. When a sample x comes, if $x \in I_j$, the corresponding counter for I_j will be incremented by 1. In the end, the counters will give the number of samples that fall in each of the intervals, and using those values we can explicitly obtain the distribution $\tilde{D}$. Each counter takes $O(\log n)$ bits of memory. There are total $\ell = (\log n/\epsilon^2)$ counters. So, the memory requirement for this step is $O(\log^2 n/\epsilon^2) < m$ bits. Also, using the distribution $\tilde{D}$ we can obtain the flattened distribution $(\tilde{D}^f)^{\mathcal{I}}$ without storing it explicitly. Hence, the Line 9 does not require any extra space for checking whether $(\tilde{D}^f)^{\mathcal{I}}$ is 2ϵ close to monotone or not. For storing the first set of $S_1 = O(m\epsilon^2/\log^2 n)$ samples for an interval will take $O(m\epsilon^2/\log n)$ bits of memory. As we are storing S_1 samples for all $\ell = O(\log n/\epsilon^2)$ intervals, it will take total $O(m)$ bits of memory. ◀

▶ Remark 15. If the input to the algorithm is a monotone distribution, then the streaming algorithm computes a distribution over the intervals $\mathcal{I}$ such that the flattening is close to a monotone distribution. Since the number of intervals in the partition is $O(\log n/\epsilon)$, the explicit description of the distribution can be succinctly stored.

We would also like to point out that the final step in the algorithm requires testing if the learnt distribution is close to some monotone distribution, and we have not explicitly bounded the space required for that.

4.2.1 Lower bound for testing monotonicity

In this section, we prove the lower bound for monotonicity testing problem in the streaming settings. We start with the discussion of the uniformity testing lower bound by ([17]) in the streaming model and later we show how the same lower bound is applicable in our case.

▶ **Theorem 16** (Uniformity testing lower bound in streaming framework [17]). *Let $\mathcal{A}$ be an algorithm which tests if a distribution D is uniform versus ϵ-far from uniform with error probability $1/3$, can access the samples in a single-pass streaming fashion using m bits of memory and S samples, then $S.m = \Omega(n/\epsilon^4)$. Furthermore, if $S < n^{0.9}$ and $m > S^2/n^{0.9}$ then $S \cdot m = \Omega(n \log n/\epsilon^4)$.*

The proof of the above lemma proceeds by choosing a random bit $X \in \{0, 1\}$, where $X = 0$ defines a *Yes* instance (uniform distribution) and $X = 1$ defines a *No* instance (ϵ-far from uniform) and calculating the mutual information between X and the bits stored in the memory after seeing S samples. In their formulation, the *Yes* instance is a uniform distribution over $2n$ and the *No* instance is obtained by pairing $(2i - 1, 2i)$ indices together and assigning values by tossing an ϵ-biased coin. In particular, the *No* distribution is obtained as follows, pair the indices as $\{1, 2\}, \{3, 4\}, ..., \{2n - 1, 2n\}$. Pick a bin $\{2i - 1, 2i\}$ and for each bin a random bit $Y_i \in \{\pm 1\}$ to assign the probabilities as,

$$(D(2i-1), D(2i)) = \begin{cases} \frac{1+\epsilon}{2n}, \frac{1-\epsilon}{2n} & \text{if } Y_i = 1 \\ \frac{1-\epsilon}{2n}, \frac{1+\epsilon}{2n} & \text{if } Y_i = -1 \end{cases}$$

It is straightforward that the *Yes* distribution is a monotone distribution as well. We show that any distribution D from the *No* instance set is $O(\epsilon)$-far from monotonicity. We start by choosing an $\alpha \in (0, \epsilon/4)$ and defining a set of partitions $\mathcal{I} = \{I_1, ..., I_\ell\}$ such that $|I_j| = \lfloor (1 + \alpha)^j \rfloor$ for $1 \le j \le \ell$. Let $(D^f)^{\mathcal{I}}$ be the flattened distribution corresponding to $\mathcal{I}$. We use the following lemma from ([9]) which reflects the fact if D is far from $(D^f)^{\mathcal{I}}$, then D is also far from being monotone. In particular, we define the lemma as follows,

▶ **Lemma 17** ([9]). *Let D be a distribution over domain $[n]$ and $\mathcal{I} = \{I_1, ..., I_\ell\}$ are the set of partitions defined obliviously with respect to a parameter $\alpha \in (0, 1)$ where $\ell = O(\frac{1}{\alpha} \log n\alpha)$ and $|I_j| = \lfloor (1 + \alpha)^j \rfloor$. If D is ϵ-close to monotone non-increasing, then $d_{TV}(D, (D^f)^{\mathcal{I}}) \le 2\epsilon + \alpha$ where $(D^f)^{\mathcal{I}}$ is the flattened distribution of D with respect to $\mathcal{I}$.*

Let, D be a distribution chosen randomly from the *No* instance set. We have the following observation,

▶ **Lemma 18.** *Let $\mathcal{I} = \{I_1, ..., I_\ell\}$ be the oblivious partitions of D with parameter α such that $|I_j| = \lfloor (1+\alpha)^j \rfloor$.*

- *If $|I_j|$ is odd, then $\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| = \frac{\epsilon}{2n}(|I_j| - \frac{1}{|I_j|})$.*
- *If $|I_j|$ is even, then $\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| \geq \frac{\epsilon}{2n}(|I_j| - \frac{4}{|I_j|})$.*

Proof. If $|I_j|$ is odd, it will contain k (any positive integer) number of bin where each bin is of form $(2x-1, 2x)$ and an extra index i' which can have the probability weight either $\frac{1+\epsilon}{2n}$ or $\frac{1-\epsilon}{2n}$. Let $D(i') = \frac{1+\epsilon}{2n}$. In this case, $D(I_j) = \frac{|I_j|}{2n} + \frac{\epsilon}{2n}$.

$$\begin{aligned}\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| &= \sum_{i\in I_j} |D(i) - \frac{1}{2n} - \frac{\epsilon}{2n|I_j|}| \\ &= \frac{\epsilon}{2n}(1 - \frac{1}{|I_j|})\frac{|I_j|-1}{2} + \frac{\epsilon}{2n}(1 + \frac{1}{|I_j|})\frac{|I_j|-1}{2} + \frac{\epsilon}{2n}(1 - \frac{1}{|I_j|}) \\ &= \frac{\epsilon}{2n}(|I_j| - \frac{1}{|I_j|})\end{aligned}$$

When $D(i') = \frac{1-\epsilon}{2n}$, similar calculation will follow.

If $|I_j|$ is even, there are two possibilities, (i) I_j consists of k (positive integer) bins. So, there will be equal number of $\frac{1+\epsilon}{2n}$ and $\frac{1-\epsilon}{2n}$ in I_j and $D(I_j) = \frac{|I_j|}{2n}$. In this case, it is straightforward to observe that $\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| = \frac{\epsilon|I_j|}{2n}$. Another case is, (ii) I_j contains $b_p, ..., b_{p+k-1}$ bins completely and $i' \in b_{p-1}$, and $i'' \in b_{p+k}$ where $D(i') = D(i'')$; the case when $D(i') \neq D(i'')$ will be similar to (i) that we saw earlier. Let $D(i') = D(i'') = \frac{1+\epsilon}{2n}$. In this case, $D(I_j) = \frac{|I_j|}{2n} + \frac{\epsilon}{n}$.

$$\begin{aligned}\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| &= \sum_{i\in I_j} |D(i) - \frac{1}{2n} - \frac{\epsilon}{n|I_j|}| \\ &= \frac{\epsilon}{2n}(1 - \frac{1}{|I_j|})\frac{|I_j|-2}{2} + \frac{\epsilon}{2n}(1 + \frac{1}{|I_j|})\frac{|I_j|-2}{2} + \frac{\epsilon}{n}(1 - \frac{2}{|I_j|}) \\ &= \frac{\epsilon}{2n}(|I_j| - \frac{4}{|I_j|})\end{aligned}$$

Combining (i) and (ii), we say $\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| \geq \frac{\epsilon}{2n}(|I_j| - \frac{4}{|I_j|})$. Similar calculation will follow when $D(i') = D(i'') = \frac{1-\epsilon}{2n}$. ◀

In our case, we apply oblivious partitions on D (chosen randomly from the *No* set) with respect to the parameter α and use the above lemma, to conclude the following,

▶ **Lemma 19.** *Let D be a randomly chosen distribution from the No instance set, then D is $\epsilon/4$-far from any monotone non-increasing distribution.*

Proof. We calculate $d_{TV}(D, (D^f)^{\mathcal{I}}) = \sum_{j=1}^{\ell} \sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| = \sum_{|I_j| \text{is even}} \sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| + \sum_{|I_j| \text{is odd}} \sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}|$. Each odd length interval contributes $\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| = \frac{\epsilon}{2n}(|I_j| - \frac{1}{|I_j|})$ and each even length interval contributes $\sum_{i\in I_j} |D(i) - \frac{D(I_j)}{|I_j|}| \geq \frac{\epsilon}{2n}(|I_j| - \frac{4}{|I_j|})$.

Hence, simplifying the distance, we get, $d_{TV}(D,(D^f)^{\mathcal{I}}) \geq \sum_{|I_j|\text{is even}} \frac{\epsilon}{2n}(|I_j| - \frac{4}{|I_j|}) + \sum_{|I_j|\text{is odd}} \frac{\epsilon}{2n}(|I_j| - \frac{1}{|I_j|}) \geq \frac{\epsilon}{2n}\sum_{I_j \in \ell}|I_j| - \frac{\epsilon}{2n}\big(\sum_{|I_j|\text{is even}}\frac{4}{|I_j|} + \sum_{|I_j|\text{is odd}}\frac{1}{|I_j|}\big) \geq \epsilon - \frac{\epsilon}{2n}.5\ell \geq \frac{3\epsilon}{4} > 2\frac{\epsilon}{4} + \alpha$, for $\alpha = \epsilon/4$. The third inequality is obtained by using the fact that $|I_j| \geq 1$ and the fourth inequality considers $\ell < n/10$. Now, by using the contra-positive of the Lemma 17, D is $\epsilon/4$-far from any monotone non-increasing distribution. ◀

Therefore, the uniformity testing lower bound from [17] is applicable in our case for distinguishing monotone from $\epsilon/4$-far monotone. We formalize this in the theorem below.

▶ **Theorem 20.** *Let $\mathcal{A}$ be an algorithm that tests if a distribution D is monotone versus $\epsilon/4$-far from monotonicity with error probability $1/3$, can access the samples in a single-pass streaming fashion using m bits of memory and S samples, then $S.m = \Omega(n/\epsilon^4)$. Furthermore, if $n^{0.34}/\epsilon^{8/3} + n^{0.1}/\epsilon^4 \leq m \leq \sqrt{n}/\epsilon^3$, then $S.m = \Omega(n \log n/\epsilon^4)$.*

We obtain the above theorem as analogous to the Theorem 16 by showing that lower bound for uniformity implies lower bound for monotonicity in the streaming framework. In particular, the uniform distribution is monotone non-increasing by default and we show that a randomly chosen distribution from *No* instance set is $\epsilon/4$-far from monotone no-increasing. Hence, the correctness of the above theorem follows directly from the Theorem 16.

4.3 Learning decomposable distributions in the streaming model

The algorithm and analysis from the previous section of monotone distributions extend to a more general class of structured distributions known as (γ, L)-decomposable distributions ([11, 18]). Formally, the class of (γ, L)-decomposable distributions is defined as follows.

▶ **Definition 21** ((γ, L)-decomposable distribution)**.** *A class $\mathcal{C}$ of distributions is said to be (γ, L)-decomposable, if for every $D \in \mathcal{C}$, there exists an $\ell \leq L$ and a partition $\mathcal{I} = \{I_1, .., I_\ell\}$ of $[n]$ into intervals such that for every interval $I_j \in \mathcal{I}$ one of the following conditions hold.*

- $D(I_j) \leq \frac{\gamma}{L}$
- $max_{i \in I_j} D(i) \leq (1+\gamma) min_{i \in I_j} D(i)$

In particular, monotone distributions, k-modal distributions, k-histograms are (γ, L)-decomposable for suitable values of γ and L. We refer to the appendix for a discussion regarding the same. We can use the ideas from the previous section and modify the algorithm of Fischer et al ([18]) to obtain trade-offs between the sample complexity and space complexity for learning the class of (γ, L)-decomposable distributions. In particular, we have the following theorem,

▶ **Theorem 22.** *If D is an $(\epsilon/2000, L)$-decomposable distribution, then the algorithm* LEARNING L-DECOMPOSABLE DISTRIBUTION STREAMING *outputs a distribution $(\tilde{D}^f)^{\mathcal{I}}$ such that $d_{TV}(D, (\tilde{D}^f)^{\mathcal{I}}) \leq \epsilon$ with probability at least $1 - \delta$. The algorithm requires $O(\frac{nL \log(1/\epsilon)}{m\epsilon^9})$ samples from D and needs $O(m)$ bits of memory where $\log n/\epsilon^4 \leq m \leq O(\sqrt{n \log n}/\epsilon^3)$.*

5 Conclusion

We give efficient algorithms for testing identity, monotonicity and (γ, L)-decomposability in the streaming model. For a memory constraint m, the number of samples required is a function of the support size n and the constraint m. For monotonicity testing, our bounds are nearly optimal. We note that the trade-off that we achieve, and lower bounds work for certain parameters of the value m. Furthermore, we have not tried to tighten the dependence of the bound on the parameter ϵ. One natural question to ask is if the dependence of sample complexity on m can be improved, and whether it can work for a larger range of values.

References

1 Jayadev Acharya, Sourbh Bhadane, Piotr Indyk, and Ziteng Sun. Estimating entropy of distributions in constant space. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

2 Maryam Aliakbarpour, Andrew McGregor, Jelani Nelson, and Erik Waingarten. Estimation of entropy in constant space with improved sample complexity. *arXiv preprint arXiv:2205.09804*, 2022.

3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

4 Gabriel Bathie and Tatiana Starikovskaya. Property testing of regular languages with applications to streaming property testing of visibly pushdown languages. In *ICALP 2021*, GLASGOW (virtual conference), United Kingdom, 2021.

5 T. Batu, L. Fortnow, E. Fischer, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 442–451, Washington, DC, USA, 2001. IEEE Computer Society.

6 Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 381–390, New York, NY, USA, 2004. ACM.

7 Tomer Berg, Or Ordentlich, and Ofer Shayevitz. On the memory complexity of uniformity testing. In Po-Ling Loh and Maxim Raginsky, editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 3506–3523. PMLR, 02–05 July 2022.

8 Lucien Birge. On the Risk of Histograms for Estimating Decreasing Densities. *The Annals of Statistics*, 15(3):1013–1022, 1987.

9 Clément L. Canonne. Big Data on the rise: Testing monotonicity of distributions. In *42nd International Conference on Automata, Languages and Programming (ICALP)*, 2015.

10 Clément L. Canonne. Topics and techniques in distribution testing: A biased but representative sample. *Found. Trends Commun. Inf. Theory*, 19(6):1032–1198, 2022. doi:10.1561/0100000114.

11 Clément L. Canonne, Ilias Diakonikolas, Themis Gouleakis, and Ronitt Rubinfeld. Testing shape restrictions of discrete distributions. *Theory of Computing Systems*, 62(1):4–62, January 2018. Publisher Copyright: © 2017, Springer Science+Business Media New York.

12 Clément L. Canonne, Dana Ron, and Rocco A. Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015.

13 Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM Journal on Computing*, 45(4):1261–1296, 2016.

14 Steve Chien, Katrina Ligett, and Andrew McGregor. Space-efficient estimation of robust statistics and distribution testing. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science – ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 251–265. Tsinghua University Press, 2010.

15 Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. In *J. Algorithms*, 2004.

16 Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler. Testable properties in general graphs and random order streaming. In *24th International Conference on Randomization and Computation (RANDOM)*, 2020.

17 Ilias Diakonikolas, Themis Gouleakis, Daniel M. Kane, and Sankeerth Rao. Communication and memory efficient testing of discrete distributions. In *Annual Conference Computational Learning Theory*, 2019.

18 Eldar Fischer, Oded Lachish, and Yadu Vasudev. Improving and extending the testing of distributions for shape-restricted properties. *Algorithmica, Springer*, 81,3765–3802, 2019. arXiv:1609.06736.

19 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electron Colloq Comput Complexity*, 7, January 2000.

20 Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014.

21 Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005.

22 Sampriti Roy and Yadu Vasudev. Testing properties of distributions in the streaming model, 2023. arXiv:2309.03245.

23 Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.