Testing equivalence between distributions using conditional samples*

(Extended Abstract)

Clément Canonne† Dana Ron‡ Rocco A. Servedio§

July 5, 2013

Abstract

We study a recently introduced framework [7, 8] for property testing of probability distributions, by considering distribution testing algorithms that have access to a *conditional sampling oracle.* This is an oracle that takes as input a subset $S \subseteq [N]$ of the domain $[N]$ of the unknown probability distribution D and returns a draw from the conditional probability distribution D restricted to S. This model allows considerable flexibility in the design of distribution testing algorithms; in particular, testing algorithms in this model can be adaptive.

In this paper we focus on algorithms for two fundamental distribution testing problems: testing whether $D = D^*$ for an explicitly provided D^*, and testing whether two unknown distributions D_1 and D_2 are equivalent. We give two efficient algorithms for each of these two problems. At a high level our main finding is that the new "conditional sampling" framework we consider is a powerful one: while both these problems above have $\Omega(\sqrt{N})$ sample complexity in the standard model we give $\mathrm{poly}(\log N, 1/\epsilon)$-query algorithms (and in some cases $\mathrm{poly}(1/\epsilon)$-query algorithms independent of N) for both of them in our conditional sampling setting.

*This paper contains selected results from [7].

†ccanonne@cs.columbia.edu, Columbia University.

‡danaron@tau.ac.il, Tel Aviv University. Supported by ISF grant number 246/08.

§rocco@cs.columbia.edu, Columbia University. Supported by NSF grants CCF-0915929 and CCF-1115703.

1 Introduction

1.1 Background: Distribution testing in the standard model

One of the most fundamental problem paradigms in statistics is that of inferring some information about an unknown probability distribution D given access to independent samples drawn from it. More than a decade ago, Batu et al. [3][1] initiated the study of problems of this type from within the framework of *property testing* [23, 12]. In a property testing problem there is an unknown "massive object" that an algorithm can access only by making a small number of "local inspections" of the object, and the goal is to determine whether the object has a particular property. The algorithm must output ACCEPT if the object has the desired property and output REJECT if the object is far from every object with the property. (See [10, 20, 21, 11] for detailed surveys and overviews of the broad field of property testing.)

In distribution property testing the "massive object" is an unknown probability distribution D over an N-element set, and the algorithm accesses the distribution by drawing independent samples from it. A wide range of different properties of probability distributions have been investigated in this setting, and upper and lower bounds on the number of samples required have by now been obtained for many problems. These include testing whether D is uniform [13, 4, 18], testing whether D is identical to a given known distribution D^* [2], testing whether two distributions D_1, D_2 (both available via sample access) are identical [3, 27], and testing whether D has a monotonically increasing probability mass function [6], as well as related problems such as estimating the entropy of D [1, 26], and estimating its support size [19, 27, 26]. Similar problems have also been studied by researchers in other communities, see e.g., [15, 17, 18].

One broad insight that has emerged from this past decade of work is that while sublinear-sample algorithms do exist for many distribution testing problems, the number of samples required is in general quite large. Even the basic problem of testing whether D is the uniform distribution $\mathcal{U}$ over $[N] = \{1, \ldots, N\}$ versus ϵ-far from uniform requires $\Omega(\sqrt{N})$ samples[2] for constant ϵ, and the other problems mentioned above have sample complexities at least this high, and in some cases *almost linear in* N [19, 27, 26]. Since such sample complexities could be prohibitively high in real-world settings where N can be extremely large, it is natural to explore problem variants where it may be possible for algorithms to succeed using fewer samples. Indeed, researchers have studied distribution testing in settings where the unknown distribution is guaranteed to have some special structure, such as being monotone, k-modal or a "k-histogram" over $[N]$ [5, 9, 14], or being monotone over $\{0,1\}^n$ [22] or over other posets [6], and have obtained significantly more sample-efficient algorithms using these additional assumptions.

1.2 Our model: Conditional sampling

In this work we pursue a different line of investigation: rather than restricting the class of probability distributions under consideration, we consider testing algorithms that may use a more powerful form of access to the unknown distribution D. This is a *conditional sampling oracle*, which allows the algorithm to obtain a draw from D_S, the conditional distribution of D restricted to a subset S of the domain (where S is specified by the algorithm). More precisely, we have:

Definition 1 *Fix a distribution D over $[N]$. A* COND *oracle for D, denoted* COND_D*, is defined as follows: The oracle is given as input a* query set $S \subseteq [N]$ *that has $D(S) > 0$. The oracle returns an element $i \in S$,*

[1] There is a more recent full version of this work [4] and we henceforth reference this recent version.

[2] This follows from the difficulty to distinguish between the uniform distribution and a distribution that is uniform over a random half of the domain using less than $\sqrt{N}/c$ samples (for a sufficiently large constant $c > 1$).

where the probability that element i is returned is $D_S(i) = D(i)/D(S)$, independently of all previous calls to the oracle.[3]

For compatibility with our COND_D notation we will write SAMP_D to denote an oracle that takes no input and, each time it is invoked, returns an element from $[N]$ drawn according to D independently from all previous draws. This is the sample access to D that is used in the standard model of testing distributions, and this is of course the same as a call to $\mathsf{COND}_D([N])$.

Motivation and Discussion. One purely theoretical motivation for the study of the COND model is that it may further our understanding regarding what forms of information (beyond standard sampling) can be helpful for testing properties of distributions. In both learning and property testing it is generally interesting to understand how much power algorithms can gain by making queries, and COND queries are a natural type of query to investigate in the context of distributions. As we discuss in more detail below, in several of our results we actually consider restricted versions of COND queries that do not require the full power of obtaining conditional samples from arbitrary sets.

A second attractive feature of the COND model is that it enables a new level of "richness" for algorithms that deal with probability distributions. In the standard model where only access to SAMP_D is provided, all algorithms must necessarily be non-adaptive, with the same initial step of simply drawing a sample of points from SAMP_D, and the difference between two algorithms comes only from how they process their samples. In contrast, the essence of the COND model is to allow algorithms to *adaptively* determine later query sets S based on the outcomes of earlier queries.

A natural question about the COND model is its plausibility: are there settings in which an investigator could actually make conditional samples from a distribution of interest? We feel that the COND framework provides a reasonable "first approximation" for scenarios that arise in application areas (e.g., in biology or chemistry) where the parameters of an experiment can be adjusted so as to restrict the range of possible outcomes. For example, a scientist growing bacteria or yeast cells in a controlled environment may be able to deliberately introduce environmental factors that allow only cells with certain desired characteristics to survive, thus restricting the distribution of all experimental outcomes to a pre-specified subset. We further note that techniques which are broadly reminiscent of COND sampling have long been employed in statistics and polling design under the name of "stratified sampling" (see e.g. [28, 16]). We thus feel that the study of distribution testing in the COND model is well motivated both by theoretical and practical considerations.

Given the above motivations, the central question is whether the COND model enables significantly more efficient algorithms than are possible in the weaker SAMP model. Our results (see Section 1.3) show that this is indeed the case.

Before detailing our results, we note that several of them will in fact deal with a weaker variant of the COND model, which we now describe. In designing COND-model algorithms it is obviously desirable to have algorithms that only invoke the COND oracle on query sets S which are "simple" in some sense. Of course there are many possible notions of "simplicity"; in this work we consider the size of a set as a measure of its simplicity, and consider algorithms which only query small sets. More precisely, we consider the following restriction of the general COND model: a PCOND (short for "pair-cond") *oracle for D* is a restricted version of COND_D that only accepts input sets S which are either $S = [N]$ (thus providing the

[3]Note that as described above the behavior of $\mathsf{COND}_D(S)$ is undefined if $D(S) = 0$, i.e., the set S has zero probability under D. While various definitional choices could be made to deal with this, we shall assume that in such a case, the oracle (and hence the algorithm) outputs "failure" and terminates. This will not be a problem for us throughout this paper, as (a) our lower bounds deal only with distributions that have $D(i) > 0$ for all $i \in [N]$, and (b) in our algorithms $\mathsf{COND}_D(S)$ will only ever be called on sets S which are "guaranteed" to have $D(S) > 0$. (More precisely, each time an algorithm calls $\mathsf{COND}_D(S)$ it will either be on the set $S = [N]$, or will be on a set S which contains an element i which has been returned as the output of an earlier call to COND_D.)

Problem	Our results		Standard model
Is $D = D^*$ for a known D^*?	COND_D	$\tilde{O}\left(\frac{1}{\epsilon^4}\right)$	$\tilde{\Theta}\left(\frac{\sqrt{N}}{\epsilon^2}\right)$ [2, 18]
	$\mathsf{PCOND}_\mathsf{D}$	$\tilde{O}\left(\frac{\log^4 N}{\epsilon^4}\right)$ $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$	
Are D_1, D_2 (both unknown) equivalent?	COND_{D_1,D_2}	$\tilde{O}\left(\frac{\log^5 N}{\epsilon^4}\right)$	$\tilde{O}\left(\frac{N^{2/3}}{\epsilon^{8/3}}\right)$ [4] $\Omega\left(N^{2/3}\right)$ [4, 27]
	$\mathsf{PCOND}_{\mathsf{D}_1,\mathsf{D}_2}$	$\tilde{O}\left(\frac{\log^6 N}{\epsilon^{21}}\right)$	

Table 1: Comparison between the COND model and the standard model for the problems studied in this paper. The upper bounds are for testing whether the property holds (i.e. $d_{\mathrm{TV}} = 0$) versus $d_{\mathrm{TV}} \geq \epsilon$, and the lower bound is for testing with $\epsilon = \Theta(1)$.

power of a SAMP_D oracle) or $S = \{i, j\}$ for some $i, j \in [N]$, i.e. sets of size two. The PCOND oracle may be viewed as a "minimalist" variant of COND that essentially permits an algorithm to compare the relative weights of two items under D (and to draw random samples from D, by setting $S = [N]$).

1.3 Our results

In this early work on the COND model we focus on the simplest (and, we think, most fundamental) concrete problems in distribution testing: specifically, testing whether $D = D^*$ for an explicitly provided D^*, and testing whether $D_1 = D_2$ given COND_{D_1} and COND_{D_2} oracles. We give a detailed study of these two problems in both the COND model and its PCOND variant described above. Our results show that the ability to do conditional sampling provides a significant amount of power to property testers, enabling $\mathrm{polylog}(N)$-query, or even constant-query, algorithms for these problems, both of which have sample complexities $N^{\Omega(1)}$ in the standard model; see Table 1.[4] In what follows d_{TV} denotes the variation distance, that is, $d_{\mathrm{TV}}(D_1, D_2) \stackrel{\mathrm{def}}{=} \frac{1}{2}\|D_1 - D_2\|_1 = \frac{1}{2}\sum_{i\in[N]} |D_1(i) - D_2(i)|$.

Testing equivalence to a known distribution. We consider the question of testing whether D (accessible via a PCOND or COND oracle) is equivalent to D^*, where D^* is an arbitrary "known" distribution over $[N]$ that is explicitly provided to the testing algorithm (say as a vector $(D^*(1), \ldots, D^*(N))$ of probabilities). For this "known D^*" problem, we give a COND_D algorithm testing whether $D = D^*$ versus $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$ using $\tilde{O}(1/\epsilon^4)$ queries (independent of the size of the domain N). We also consider the power of $\mathsf{PCOND}_\mathsf{D}$ oracles for this problem, and give a $\mathsf{PCOND}_\mathsf{D}$ algorithm that uses $\tilde{O}((\log N)^4/\epsilon^4)$ queries. We further show that the $(\log N)^{\Omega(1)}$ query complexity of our $\mathsf{PCOND}_\mathsf{D}$ algorithm is inherent in the problem, by proving that any $\mathsf{PCOND}_\mathsf{D}$ algorithm for this problem must use $\Omega(\sqrt{\log(N)/\log\log(N)})$ queries for constant ϵ.

Testing equivalence between two unknown distributions. We next consider the more challenging problem of testing whether two unknown distributions D_1, D_2 over $[N]$ (available via COND_{D_1} and COND_{D_2} oracles) are identical versus ϵ-far. We give a $\mathrm{poly}(\log N, 1/\epsilon)$ algorithm for this problem in the restricted PCOND model, breaking the $\Omega\left(N^{2/3}\right)$ sample lower bound in the standard model. We also give a completely different algorithm, using general COND queries, that achieves an improved $\mathrm{poly}(\log N, 1/\epsilon)$ query complexity.

Along the way to establishing these testing results, we develop several powerful tools for analyzing distributions in the COND and PCOND models, which we believe may be of independent interest and utility in subsequent work on the COND and PCOND models. These include a procedure for approximately

[4] [7] is an extended version of this work that gives a broad range of additional results, including both upper and lower bounds, for several other problems and variants of the COND model. See Table 1 of [7] for a concise overview of its results.

simulating an "evaluation oracle"[5] and a procedure for estimating the weight of the "neighborhood" of a given point in the domain of the distribution. (See further discussion of these tools below.)

1.3.1 A high-level discussion of our algorithms

Our COND- and PCOND- model algorithms are adaptive, and hence necessarily have quite a different algorithmic flavor from distribution testing algorithms in the standard sampling model (which are of course nonadaptive). As can be seen in the following discussion, our various algorithms share some common themes with each other, though each has its own unique idea/technique, which we emphasize below.

For intuition, consider first a special case of **testing equality to a known distribution** D^* where D^* is the uniform distribution over $[N]$. It is not hard to verify that if a distribution D is ϵ-far from uniform, then the following holds: if we select $\Theta(1/\epsilon)$ points according to D and select $\Theta(1/\epsilon)$ points uniformly from $[N]$, then with high constant probability we shall obtain a point x in the first sample, and a point y in the second sample such that $D(x)/D(y)$ is lower bounded by $(1+\Omega(\epsilon))$. This can be detected with high constant probability by performing $\Theta(1/\epsilon^2)$ $\mathsf{PCOND}_\mathsf{D}$ queries on each such pair of points. Since when D^* is the uniform distribution, $D(x)/D(y)=1$ for every pair of points x, y, this provides evidence that $D \neq D^*$, and we can get an algorithm for testing equality to the uniform distribution in the $\mathsf{PCOND}_\mathsf{D}$ model whose complexity is $\mathrm{poly}(1/\epsilon)$. While this simple approach using $\mathsf{PCOND}_\mathsf{D}$ queries succeeds with only $\mathrm{poly}(1/\epsilon)$ queries when D^* is the uniform distribution, we show that for general distributions D^* the query complexity of any $\mathsf{PCOND}_\mathsf{D}$ algorithm for testing equality with D^* must be $(\log N)^{\Omega(1)}$.

In order to obtain an algorithm whose complexity is $\mathrm{poly}(1/\epsilon)$ in the COND_D model we extend the basic idea from the uniform case as follows. Rather than comparing the relative weight of pairs of points, we compare the relative weight of pairs in which one element is a point and the other is a subset of points. Roughly speaking, we show how points can be paired with subsets of points of comparable weight (according to D^*) such that the following holds. If D is far from D^*, then by taking $\tilde{O}(1/\epsilon)$ samples from D and selecting subsets of points in an appropriate manner (depending on D^*), we can obtain (with high probability) a point x and a subset Y such that $D(x)/D(Y)$ differs significantly from $D^*(x)/D^*(Y)$ *and* $D^*(x)/D^*(Y)$ is a constant (the latter is essential for getting $\mathrm{poly}(1/\epsilon)$ query complexity overall).

Returning to the $\mathsf{PCOND}_\mathsf{D}$ model, we show that by sampling from both D and D^* and allowing the number of samples to grow with $\log N$, with high probability we either obtain a pair of points (x, y) such that $D(x)/D(y)$ differs by at least $(1 \pm \Omega(\epsilon))$ from $D^*(x)/D^*(y)$ where $D^*(x)/D^*(y)$ is a constant, or we detect that for some set of points B we have that $|D(B) - D^*(B)|$ is relatively large.[6]

We next turn to the more challenging problem of **testing equality between two unknown distributions** D_1 **and** D_2. In this problem we need to cope with the fact that we no longer "have a hold" on a known distribution. Our PCOND algorithm can be viewed as creating such a hold in the following sense. By sampling from D_1 we obtain (with high probability) a (relatively small) set of points R that *cover* the distribution D_1. By "covering" we mean that except for a subset having small weight according to D_1, all points y in $[N]$ have a *representative* $r \in R$, i.e. a point r such that $D_1(y)$ is close to $D_1(r)$. We then show that if D_2 is far from D_1, then one of the following must hold: (1) There is relatively large weight, either according to D_1 or according to D_2, on points y such that for some $r \in R$ we have that $D_1(y)$ is close to $D_1(r)$ but $D_2(y)$ is not sufficiently close to $D_2(r)$; (2) There exists a point $r \in R$ such that the set of points y for which $D_1(y)$ is close to $D_1(r)$ has significantly different weight according to D_2 as compared to D_1.

[5] An EVAL_D oracle (evaluation oracle for D) takes as input a point $i \in [N]$ and outputs the probability $D(i)$ that D puts on i.

[6] Here we use B for "Bucket", as we consider a bucketing of the points in $[N]$ based on their weight according to D^*. We note that bucketing has been used extensively in the context of testing properties of distributions, see e.g. [4, 2].

A key subroutine employed by our PCOND algorithm is ESTIMATE-NEIGHBORHOOD, which, given a point x and PCOND access to D returns an estimate of the weight of a subset of points whose probability (according to D) is similar to that of x. The difficulty with performing this task is due to points whose probability is close to the "similarity threshold" that determines the neighborhood set; our ESTIMATE-NEIGHBORHOOD procedure surmounts this difficulty by making a random choice of the similarity threshold. We believe that the ESTIMATE-NEIGHBORHOOD subroutine may be useful in further work as well; indeed [7] uses it in a COND algorithm for estimating the distance between two probability distributions.

Our general COND algorithm for testing the equality of two (unknown) distributions is based on a subroutine that estimates $D(x)$ (to within $(1 \pm O(\epsilon))$) for a given point x given access to COND_D. Obtaining such an estimate for *every* $x \in [N]$ cannot be done efficiently for some distributions.[7] However, we show that if we allow the algorithm to output UNKNOWN on some subset of points with total weight $O(\epsilon)$, then the relaxed task can be performed using $\mathrm{poly}(\log N, 1/\epsilon)$ queries, by performing a kind of randomized binary search "with exceptions". This relaxed version, which we refer to as an *approximate* EVAL *oracle*, suffices for our needs in distinguishing between the case that D_1 and D_2 are the same distribution and the case in which they are far from each other. It is possible that this procedure will be useful for other tasks as well.

1.4 The work of Chakraborty et al. [8]

Chakraborty et al. [8] propose essentially the same COND model that we study, differing only in what happens on query sets S such that $D(S) = 0$. In our model such a query causes the COND oracle and algorithm to return FAIL, while in their model such a query returns a uniform random $i \in S$.

Related to testing equality of distributions, [8] provides an (adaptive) algorithm for testing whether D is equivalent to a specified distribution D^* using $\mathrm{poly}(\log^* N, 1/\epsilon)$ COND queries. Recall that we give an algorithm for this problem that performs $\tilde{O}(1/\epsilon^4)$ COND queries. [8] also gives a *non-adaptive* algorithm for this problem that performs $\mathrm{poly}(\log N, 1/\epsilon)$ COND queries. [8] Testing equivalence between two unknown distributions is not considered in [8], and the same is true for testing in the PCOND model.

Both [8] and [7] also present additional results for a range of other problems (problems which are largely disjoint between the two papers) but we do not discuss those results here.

1.5 Organization

Due to space constraints this extended abstract includes sketches for only two of our results, which we hope give a flavor of our work: The general COND algorithm for testing equivalence to a known distribution and the PCOND algorithm for testing equivalence between two unknown distributions. Full details of all our results can be found in the accompanying full version of this paper.

2 Some useful procedures

Our algorithms use two procedures as subroutines: COMPARE and ESTIMATE-NEIGHBORHOOD. The COMPARE procedure is quite straightforward and may be viewed as a low-level tool, while the ESTIMATE-NEIGHBORHOOD procedure is more sophisticated. We give complete descriptions of these routines and proofs of their performance guarantees in the full version.

[7] As an extreme case consider a distribution D for which $D(1) = 1 - \phi$ and $D(2) = \cdots = D(N) = \phi/(N-1)$ for some very small ϕ (which in particular may depend on N), and for which we are interested in estimating $D(2)$. This requires $\Omega(1/\phi)$ queries.

[8] We note that it is only possible for them to give a non-adaptive algorithm because their model is more permissive than ours (if a query set S is proposed for which $D(S) = 0$, their model returns a uniform random element of S while our model returns FAIL). In our stricter model, any non-adaptive algorithm which queries a proper subset $S \subsetneq N$ would output FAIL on some distribution D.

The procedure COMPARE estimates the ratio between the weights of two disjoint sets of points by performing COND queries on the union of the sets. In the special case when each set is of size one, the queries performed are PCOND queries.

Lemma 1 *Given as input two disjoint subsets of points X, Y together with parameters $\eta \in (0,1]$, $K \geq 1$, and $\delta \in (0, 1/2]$, as well as* COND *query access to a distribution D, there exists a procedure* COMPARE *that performs $O\left(\frac{K \log(1/\delta)}{\eta^2}\right)$* COND *queries on the set $X \cup Y$ and either outputs a value $\rho > 0$ or outputs* High *or* Low*, and satisfies the following:*

1. *If $D(X)/K \leq D(Y) \leq K \cdot D(X)$ then with probability at least $1-\delta$ the procedure outputs a value $\rho \in [1-\eta, 1+\eta] D(Y)/D(X)$;*
2. *If $D(Y) > K \cdot D(X)$ then with probability at least $1-\delta$ the procedure outputs either* High *or a value $\rho \in [1-\eta, 1+\eta] D(Y)/D(X)$;*
3. *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 ESTIMATE-NEIGHBORHOOD is given as input a point x and it provides an estimate of the weight of a set of points y such that $D(y)$ is similar to $D(x)$. In order to specify the behavior of the procedure more precisely, we introduce the following notation. For a distribution D over $[N]$, a point $x \in [N]$ and a parameter $\gamma \in [0,1]$, let $U_\gamma^D(x) \stackrel{\text{def}}{=} \left\{ y \in [N] : \frac{1}{1+\gamma} D(x) \leq D(y) \leq (1+\gamma) D(x) \right\}$ denote the set of points whose weight is "γ-close" to the weight of x. If we take a sample of points distributed according to D, then the expected fraction of these points that belong to $U_\gamma^D(x)$ is $D(U_\gamma^D(x))$. If this value is not too small, then the actual fraction in the sample is close to the expected value. Hence, if we could efficiently determine for any given point y whether or not it belongs to $U_\gamma^D(x)$, then we could obtain a good estimate of $D(U_\gamma^D(x))$. The difficulty is that it is not possible to perform this task efficiently for "boundary" points y such that $D(y)$ is very close to $(1+\gamma)D(x)$ or to $\frac{1}{1+\gamma} D(x)$. However, for our purposes, it is not important that we obtain the weight and size of $U_\gamma^D(x)$ for a specific γ, but rather it suffices to do so for γ in a given range, as stated in the next lemma.

Lemma 2 *Given as input a point x together with parameters $\kappa, \beta, \eta, \delta \in (0, 1/2]$ as well as* PCOND *query access to a distribution D, there exists a procedure* ESTIMATE-NEIGHBORHOOD *that performs $O\left(\frac{\log(1/\delta) \cdot \log(\log(1/\delta)/(\beta\eta^2))}{\kappa^2 \eta^4 \beta^3 \delta^2}\right)$* PCOND *queries and outputs a pair $(\hat{w}, \alpha) \in [0,1] \times (\kappa, 2\kappa)$ such that α is uniformly distributed in $\{\kappa + i\theta\}_{i=0}^{\kappa/\theta - 1}$ for $\theta = \frac{\kappa\eta\beta\delta}{64}$, and such that the following holds:*

1. *If $D(U_\alpha^D(x)) \geq \beta$, then with probability at least $1-\delta$ we have $\hat{w} \in [1-\eta, 1+\eta] \cdot D(U_\alpha^D(x))$, and $D(U_{\alpha+\theta}^D(x) \setminus U_\alpha^D(x)) \leq \eta\beta/16$;*
2. *If $D(U_\alpha^D(x)) < \beta$, then with probability at least $1-\delta$ we have $\hat{w} \leq (1+\eta) \cdot \beta$, and $D(U_{\alpha+\theta}^D(x) \setminus U_\alpha^D(x)) \leq \eta\beta/16$.*

The main idea is that, by picking α at random as described in the lemma, the total weight of points "at the boundary" of the α-neighborhood will with high probability be very small, so a high-accuracy estimate of $D(U_\alpha^D(x))$ can be obtained.

3 Testing equivalence to a known distribution D^*

In this section we describe the idea behind an algorithm COND-TEST-KNOWN, whose properties are stated in the following theorem.

Theorem 1 COND-TEST-KNOWN *is a $\tilde{O}(1/\epsilon^4)$-query COND_D testing algorithm for testing equivalence to a known distribution D^*. That is, for every pair of distributions D, D^* over $[N]$ (such that D^* is fully specified and there is* COND *query access to D), the algorithm outputs* ACCEPT *with probability at least $2/3$ if $D = D^*$ and outputs* REJECT *with probability at least $2/3$ if $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$.*

High-level overview of the algorithm and its analysis: First, we note that by reordering elements of $[N]$ we may assume without loss of generality that $D^*(1) \leq \cdots \leq D^*(N)$; this will be convenient for us.

As we show in the full version, our $(\log N)^{\Omega(1)}$ query lower bound for $\mathsf{PCOND}_\mathsf{D}$ algorithms exploits the intuition that comparing two points using the $\mathsf{PCOND}_\mathsf{D}$ oracle might not provide much information (e.g. if one of the two points was a priori "known" to be much heavier than the other). In contrast, with a general COND_D oracle at our disposal, we can compare a given point $j \in [N]$ with *any subset* of $[N] \setminus \{j\}$. Thus the following definition will be useful:

Definition 2 (Comparable points) *Fix $0 < \lambda \leq 1$. A point $j \in \mathrm{supp}(D^*)$ is said to be* λ-comparable *if there exists a set $S \subseteq ([N] \setminus \{j\})$ such that $D^*(j) \in [\lambda D^*(S), D^*(S)/\lambda]$. Such a set S is then said to be a* λ-comparable-witness for *j (according to D^*), which is denoted $S \cong^* j$. We say that a set $T \subseteq [N]$ is λ-comparable if every $i \in T$ is λ-comparable.*

We stress that the notion of being λ-comparable deals only with the known distribution D^*.

Fix $\epsilon_1 = \Theta(\epsilon)$ (see full version for its exact value). Our analysis and algorithm consider two possible cases for the distribution D^* (where it is not hard to verify, and we provide an explanation in the full version, that one of the two cases must hold):

- The first case is that for some $i^* \in [N]$ we have $D^*(\{1, \ldots, i^*\}) > 2\epsilon_1$ but $D^*(\{1, \ldots, i^*-1\}) \leq \epsilon_1$. In this case $1 - \epsilon_1$ of the total probability mass of D^* must lie on a set of at most $1/\epsilon_1$ elements, and in such a situation it is easy to efficiently test whether $D = D^*$ using $\mathrm{poly}(1/\epsilon)$ queries (see Algorithm COND_D-TEST-KNOWN-HEAVY and its analysis in the full version).
- The second case is that there exists an element $k^* \in [N]$ such that $\epsilon_1 < D^*(\{1, \ldots, k^*\}) \leq 2\epsilon_1 < D^*(\{1, \ldots, k^* + 1\})$. This is the more challenging (and typical) case. In this case, it can be shown that every element $j > k^*$ has at least one ϵ_1-comparable-witness within $\{1, \ldots, j\}$. In fact, one can show that either (a) $\{1, \ldots, j-1\}$ is an ϵ_1-comparable witness for j, or (b) the set $\{1, \ldots, j-1\}$ can be partitioned into disjoint sets $S_1, \ldots, S_t$ such that *each* S_i, $1 \leq i \leq t$, is a $\frac{1}{2}$-comparable-witness for j. Case (a) is relatively easy to handle so we focus on (b) in our informal description below.

The partition $S_1, \ldots, S_t$ is useful to us for the following reason: Suppose that $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$. It is not difficult to show that unless $D(\{1, \ldots, k^*\}) > 3\epsilon_1$ (which can be easily detected and provides evidence that the tester should reject), a random sample of $\Theta(1/\epsilon)$ draws from D will with high probability contain a "heavy" point $j > k^*$, that is, a point $j > k^*$ such that $D(j) \geq (1 + \epsilon_2)D^*(j)$ (where $\epsilon_2 = \Theta(\epsilon)$). Given such a point j, there are two possibilities:

- The first possibility is that a significant fraction of the sets $S_1, \ldots, S_t$ have $D(j)/D(S_i)$ "noticeably different" from $D^*(j)/D^*(S_i)$. (Observe that since each set S_i is a $\frac{1}{2}$-comparable witness for j, it is possible to efficiently check whether this is the case.) If this is the case then our tester should reject since this is evidence that $D \neq D^*$.
- The second possibility is that almost every S_i has $D(j)/D(S_i)$ very close to $D^*(j)/D^*(S_i)$. If this is the case, though, then since $D(j) \geq (1 + \epsilon_2)D^*(j)$ and the union of $S_1, \ldots, S_t$ is $\{1, \ldots, j-1\}$, it must be the case that $D(\{1, \ldots, j\})$ is "significantly larger" than $D^*(\{1, \ldots, j\})$. This will be revealed by random sampling from D and thus our testing algorithm can reject in this case as well.

The detailed proof of Theorem 1, including a complete description and analysis of the algorithm, is given in the full version.

4 Testing equality between two unknown distributions

In this section we consider the problem of testing whether two unknown distributions D_1, D_2 are identical versus ϵ-far, given PCOND access to these distributions. Although this is known to require $\Omega\big(N^{2/3}\big)$ many samples in the standard model [4, 27], we are able to give a $\mathrm{poly}(\log N, 1/\epsilon)$-query algorithm using PCOND queries, by taking advantage of comparisons to perform some sort of *clustering* of the domain.

On a high level the algorithm works as follows. First it obtains (with high probability) a small set of points R such that almost every element in $[N]$, except possibly for some negligible subset according to D_1, has probability weight (under D_1) close to some "representative" in R. Next, for each representative r in R it obtains an estimate of the weight, according to D_1, of a set of points U such that $D_1(u)$ is close to $D_1(r)$ for each u in U (i.e, r's "neighborhood under D_1"). This is done using the procedure ESTIMATE-NEIGHBORHOOD. Note that these neighborhoods can be interpreted roughly as a succinct *cover* of the support of D_1 into (not necessarily disjoint) sets of points, where within each set the points have similar weight (according to D_1). Our algorithm is based on the observation that, if D_1 and D_2 are far from each other, it must be the case that one of these sets, denoted U^*, reflects it in one of the following ways: (1) $D_2(U^*)$ differs significantly from $D_1(U^*)$; (2) U^* contains a subset of points V^* such that $D_2(v)$ differs significantly from $D_2(r)$ for each v in V^*, and either $D_1(V^*)$ is relatively large or $D_2(V^*)$ is relatively large. (This structural result is made precise in Lemma 4). We thus take additional samples, both from D_1 and from D_2, and compare the weight (according to both distributions) of each point in these samples to the representatives in R (using the procedure COMPARE). In this manner we detect (with high probability) that either (1) or (2) holds.

We begin by formalizing the notion of a cover discussed above:

Definition 3 (Weight-Cover) *Given a distribution D on $[N]$ and a parameter $\epsilon_1 > 0$, we say that a point $i \in [N]$ is* ϵ_1-covered *by a set $R = \{r_1, \ldots, r_t\} \subseteq [N]$ if there exists a point $r_j \in R$ such that $D(i) \in [1/(1+\epsilon_1), 1+\epsilon_1]D(r_j)$. Let the set of points in $[N]$ that are ϵ_1-covered by R be denoted by $C^D_{\epsilon_1}(R)$. We say that R is an* (ϵ_1, ϵ_2)-cover for D *if $D([N] \setminus C^D_{\epsilon_1}(R)) \leq \epsilon_2$.*

The following lemma (proved in the full version) says that a small sample of points drawn from D gives a cover with high probability:

Lemma 3 *Let D be any distribution over $[N]$. Given any fixed $c > 0$, there exists a constant $c' > 0$ such that with probability at least $99/100$, a sample R of size $m = c' \frac{\log(N/\epsilon)}{\epsilon^2} \cdot \log\left(\frac{\log(N/\epsilon)}{\epsilon}\right)$ drawn according to distribution D is an $(\epsilon/c, \epsilon/c)$-cover for D.*

The next lemma (proved in the full version) formalizes the sense in which some "neighborhood" of a point in a cover must "witness" the fact that D_1 and D_2 are far from each other:

Lemma 4 *Suppose $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$, and let $R = \{r_1, \ldots, r_t\}$ be an $(\tilde{\epsilon}, \tilde{\epsilon})$-cover for D_1 where $\tilde{\epsilon} \leq \epsilon/100$. Then, there exists $j \in [t]$ such that at least one of the following conditions holds for every $\alpha \in [\tilde{\epsilon}, 2\tilde{\epsilon}]$:*

1. *$D_1(U^{D_1}_\alpha(r_j)) \geq \frac{\tilde{\epsilon}}{t}$ and $D_2(U^{D_1}_\alpha(r_j)) \notin [1-\tilde{\epsilon}, 1+\tilde{\epsilon}]D_1(U^{D_1}_\alpha(r_j))$, or $D_1(U^{D_1}_\alpha(r_j)) < \frac{\tilde{\epsilon}}{t}$ and $D_2(U^{D_1}_\alpha(r_j)) > \frac{2\tilde{\epsilon}}{t}$;*

2. $D_1(U_\alpha^{D_1}(r_j)) \geq \frac{\tilde{\epsilon}}{t}$, *and at least a* $\tilde{\epsilon}$*-fraction of the points* i *in* $U_\alpha^{D_1}(r_j)$ *satisfy* $\frac{D_2(i)}{D_2(r_j)} \notin [1/(1+\alpha+\tilde{\epsilon}), 1+\alpha+\tilde{\epsilon}]$*;*
3. $D_1(U_\alpha^{D_1}(r_j)) \geq \frac{\tilde{\epsilon}}{t}$*, and the total weight according to* D_2 *of the points* i *in* $U_\alpha^{D_1}(r_j)$ *for which* $\frac{D_2(i)}{D_2(r_j)} \notin [1/(1+\alpha+\tilde{\epsilon}), 1+\alpha+\tilde{\epsilon}]$ *is at least* $\frac{\tilde{\epsilon}^2}{t}$.

Algorithm 1: Algorithm $\mathsf{PCOND}_{\mathsf{D_1},\mathsf{D_2}}$-TEST-EQUALITY-UNKNOWN

Input: PCOND query access to distributions D_1 and D_2 and a parameter ϵ.

1. Set $\tilde{\epsilon} = \epsilon/100$. Draw a sample R of size $t = \tilde{\Theta}\left(\frac{\log N}{\epsilon^2}\right)$ from D_1.
2. For each $r_j \in R$:
   (a) Call ESTIMATE-NEIGHBORHOOD$_{D_1}$ on r_j with $\kappa = \tilde{\epsilon}$, $\eta = \frac{\tilde{\epsilon}}{8}$, $\beta = \frac{\tilde{\epsilon}}{2t}$, $\delta = \frac{1}{100t}$ and let the output be denoted by $(\hat{w}_j^{(1)}, \alpha_j)$.
   (b) Set $\theta = \kappa\eta\beta\delta/64 = \tilde{\Theta}(\epsilon^7/\log^2 N)$ and draw a sample S_1 from D_1, of size $s_1 = \Theta\left(\frac{t}{\epsilon^2}\right) = \tilde{\Theta}\left(\frac{\log N}{\epsilon^4}\right)$. and a sample S_2 from D_2, of size $s_2 = \Theta\left(\frac{t \log t}{\epsilon^3}\right) = \tilde{\Theta}\left(\frac{\log N}{\epsilon^5}\right)$.
   (c) For each point $i \in S_1 \cup S_2$ call COMPARE$_{D_1}(\{r_j\}, \{i\}, \theta/4, 4, 1/(200t(s_1+s_2)))$ and COMPARE$_{D_2}(\{r_j\}, \{i\}, \theta/4, 4, 1/(200t(s_1+s_2)))$, and let the outputs be denoted $\rho_{r_j}^{(1)}(i)$ and $\rho_{r_j}^{(2)}(i)$, respectively (where in particular these outputs may be High or Low).
   (d) Let $\hat{w}_j^{(2)}$ be the fraction of occurrences of $i \in S_2$ such that $\rho_{r_j}^{(1)}(i) \in [1/(1+\alpha_j+\theta/2), 1+\alpha_j+\theta/2]$.
   (e) If ($\hat{w}_j^{(1)} \leq \frac{3}{4}\frac{\tilde{\epsilon}}{t}$ and $\hat{w}_j^{(2)} > \frac{3}{2}\frac{\tilde{\epsilon}}{t}$) or ($\hat{w}_j^{(1)} > \frac{3}{4}\frac{\tilde{\epsilon}}{t}$ and $\hat{w}_j^{(2)}/\hat{w}_j^{(1)} \notin [1-\tilde{\epsilon}/2, 1+\tilde{\epsilon}/2]$), then output REJECT.
   (f) If there exists $i \in S_1 \cup S_2$ such that $\rho_{r_j}^{(1)}(i) \in [1/(\alpha_j+\tilde{\epsilon}/2), 1+\alpha_j+\tilde{\epsilon}/2]$ and $\rho_{r_j}^{(2)}(i) \notin [1/(\alpha_j+3\tilde{\epsilon}/2), 1+\alpha_j+3\tilde{\epsilon}/2]$, then output REJECT.
3. Output ACCEPT.

Theorem 2 *If* $D_1 = D_2$ *then with probability at least* $2/3$ *Algorithm* PCOND-TEST-EQUALITY-UNKNOWN *returns* ACCEPT*, and if* $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$*, then with probability at least* $2/3$ *Algorithm* PCOND-TEST-EQUALITY-UNKNOWN *returns* REJECT. *The number of* PCOND *queries performed is* $\tilde{O}\left(\frac{\log^6 N}{\epsilon^{21}}\right)$.

Proof: The (straightforward) query complexity analysis is given in the full version.

We now turn to establishing the correctness of the algorithm. We shall use the shorthand U_j for $U_{\alpha_j}^{D_1}(r_j)$, and U_j' for $U_{\alpha_j+\theta}^{D_1}(r_j)$. In the full version we show that with high proability all of the following "desirable" events hold:

1. The event E_1 is that the sample R is a $(\tilde{\epsilon}, \tilde{\epsilon})$-weight-cover for D_1 (for $\tilde{\epsilon} = \epsilon/100$).
2. The event E_2 is that all calls to ESTIMATE-NEIGHBORHOOD are as specified by Lemma 2.
3. The event E_3 is that all calls to the procedure COMPARE are as specified by Lemma 1.
4. The event E_4 is that $D_2(U_j' \setminus U_j) \leq \eta\beta/16 = \tilde{\epsilon}^2/(256t)$ for each j.
5. The event E_5 is defined as follows. For each j, if $D_2(U_j) \geq \tilde{\epsilon}/(4t)$, then $|S_2 \cap U_j|/|S_2| \in [1-\tilde{\epsilon}/10, 1+\tilde{\epsilon}/10]D_2(U_j)$, and if $D_2(U_j) < \tilde{\epsilon}/(4t)$ then $|S_2 \cap U_j|/|S_2| < (1+\tilde{\epsilon}/10)\tilde{\epsilon}/(4t)$.

6. The event E_6 is that for each j we have $|S_2 \cap (U'_j \setminus U_j)|/|S_2| \leq \tilde{\epsilon}^2/(128t)$.

From this point on we assume that events $E_1 - E_6$ all hold. Note that in particular this implies the following:

1. By E_2, for every j: If $D_1(U_j) \geq \beta = \tilde{\epsilon}/(2t)$, then $\hat{w}_j^{(1)} \in [1-\eta, 1+\eta]D_1(U_j) = [1-\tilde{\epsilon}/8, 1+\tilde{\epsilon}/8]D_1(U_j)$. If $D_1(U_j) < \tilde{\epsilon}/(2t)$, then $\hat{w}_j^{(1)} \leq (1+\tilde{\epsilon}/8)(\tilde{\epsilon}/(2t))$.
2. By E_3, for every j and for each point $i \in S_1 \cup S_2$: If $i \in U_j$, then $\rho_{r_j}^{(1)}(i) \in [1/(1+\alpha_j+\frac{\theta}{2}), 1+\alpha_j+\frac{\theta}{2}]$. If $i \notin U'_j$, then $\rho_{r_j}^{(1)}(i) \notin [1/(1+\alpha_j+\frac{\theta}{2}), 1+\alpha_j+\frac{\theta}{2}]$.
3. By the previous item and E_4–E_6: If $D_2(U_j) \geq \tilde{\epsilon}/(4t)$, then $\hat{w}_j^{(2)} \geq (1-\tilde{\epsilon}/10)D_2(U_j)$ and $\hat{w}_j^{(2)} \leq (1+\tilde{\epsilon}/10)D_2(U_j)+\tilde{\epsilon}^2/(128t) \leq (1+\tilde{\epsilon}/8)D_2(U_j)$. If $D_2(U_j) < \tilde{\epsilon}/(4t)$ then $\hat{w}_j^{(2)} \leq (1+\tilde{\epsilon}/10)\tilde{\epsilon}/(4t)+\tilde{\epsilon}^2/(128t) \leq (1+\tilde{\epsilon}/4)(\tilde{\epsilon}/(4t))$.

Completeness. Assume D_1 and D_2 are the same distribution D. For each j, if $D(U_j) \geq \tilde{\epsilon}/t$, then by the foregoing discussion, $\hat{w}_j^{(1)} \geq (1-\tilde{\epsilon}/8)D(U_j) > 3\tilde{\epsilon}/(4t)$ and $\hat{w}_j^{(2)}/\hat{w}_j^{(1)} \in [(1-\tilde{\epsilon}/8)^2, (1+\tilde{\epsilon}/8)^2] \subset [1-\tilde{\epsilon}/2, 1+\tilde{\epsilon}/2]$, so that the algorithm does not reject in Line 2-e. Otherwise (i.e., $D(U_j) < \tilde{\epsilon}/t$), we consider two subcases. Either $D(U_j) \leq \tilde{\epsilon}/(2t)$, in which case $\hat{w}_j^{(1)} \leq 3\tilde{\epsilon}/(4t)$, or $\tilde{\epsilon}/(2t) < D(U_j) < \tilde{\epsilon}/t$, and then $\hat{w}_j^{(1)} \in [1-\tilde{\epsilon}/8, 1+\tilde{\epsilon}/8]D_1(U_j)$. Since in both cases $\hat{w}_j^{(2)} \leq (1+\tilde{\epsilon}/8)D(U_j) \leq 3\tilde{\epsilon}/(2t)$, the algorithm does not reject in Line 2-e. By E_3, the algorithm does not reject in Line 2-f either.

Soundness. Assume $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$. By applying Lemma 4 on R (and using E_1), there exists an index j for which one of the items in the lemma holds. We denote this index by j^*, and consider the three items in the lemma.

1. If Item 1 holds, then we consider its two cases:

   • In the first case, $D_1(U_{j^*}) \geq \tilde{\epsilon}/t$ and $D_2(U_{j\_*}) \notin [1-\tilde{\epsilon}, 1+\tilde{\epsilon}]D_1(U_{j^*})$. Due to the lower bound on $D_1(U_{j^*})$ we have that $\hat{w}_{j^*}^{(1)} \in [1-\tilde{\epsilon}/8, 1+\tilde{\epsilon}/8]D_1(U_{j^*})$, so that in particular $\hat{w}_{j^*}^{(1)} > 3\tilde{\epsilon}/(4t)$. As for $\hat{w}_{j^*}^{(2)}$, either $\hat{w}_{j^*}^{(2)} < (1-\tilde{\epsilon})(1+\tilde{\epsilon}/8)D_1(U_{j^*})$ (this holds both when $D_2(U_{j^*}) \geq \tilde{\epsilon}/(4t)$ and when $D_2(U_{j^*}) < \tilde{\epsilon}/(4t)$) or $\hat{w}_{j^*}^{(2)} > (1+\tilde{\epsilon})(1-\tilde{\epsilon}/10)D_1(U_{j^*})$. In either (sub)case $\hat{w}_{j^*}^{(2)}/\hat{w}_{j^*}^{(1)} \notin [1-\tilde{\epsilon}/2, 1+\tilde{\epsilon}/2]$, causing the algorithm to reject in (the second part of) Line 2-e.

   • In the second case, $D_1(U_{j^*}) < \tilde{\epsilon}/t$ and $D_2(U_{j^*}) > 2\tilde{\epsilon}/t$. Due to the lower bound on $D_2(U_{j^*})$ we have that $\hat{w}_{j^*}^{(2)} \geq (1-\tilde{\epsilon}/10)D_2(U_{j^*}) > (1-\tilde{\epsilon}/10)(2\tilde{\epsilon}/t)$, so that in particular $\hat{w}_{j^*}^{(2)} > (3\tilde{\epsilon}/(2t))$. As for $\hat{w}_{j^*}^{(1)}$, if $D_1(U_{j^*}) \leq \tilde{\epsilon}/(2t)$, then $\hat{w}_{j^*}^{(1)} \leq 3\tilde{\epsilon}/(4t)$, causing the algorithm to reject in (the first part of) Line 2-e. If $\tilde{\epsilon}/(2t) < D_1(U_{j^*}) \leq \tilde{\epsilon}/t$, then $\hat{w}_{j^*}^{(1)} \in [1-\tilde{\epsilon}/8, 1+\tilde{\epsilon}/8]D_1(U_{j^*}) \leq (1+\tilde{\epsilon}/8)(\tilde{\epsilon}/t)$, so that $\hat{w}_{j^*}^{(2)}/\hat{w}_{j^*}^{(1)} \geq \frac{(1-\tilde{\epsilon}/10)(2\tilde{\epsilon}/t)}{(1+\tilde{\epsilon}/8)\tilde{\epsilon}/t} > (1+\tilde{\epsilon}/2)$, causing the algorithm to reject in (either the first or second part of) Line 2-e.
2. If Item 2 holds, then by the choice of the size of S_1, which is $\Theta(t/\tilde{\epsilon}^2)$, with probability at least $99/100$, the sample S_1 will contain a point i for which $\frac{D_2(i)}{D_2(r_{j^*})} \notin [1/(1+\alpha_{j^*}+\tilde{\epsilon}), 1+\alpha_{j^*}+\tilde{\epsilon}]$, and by E_3 this will be detected in Line 2-f.
3. Similarly, if Item 3 holds, then by the choice of the size of S_2, with probability at least $99/100$, the sample S_2 will contain a point i for which $\frac{D_2(i)}{D_2(r_{j^*})} \notin [1/(1+\alpha_{j^*}+\tilde{\epsilon}), 1+\alpha_{j^*}+\tilde{\epsilon}]$, and by E_3 this will be detected in Line 2-f.

The theorem is thus established. ■

References

[1] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating the entropy. *SICOMP*, 35(1):132–150, 2005.

[2] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proceedings of FOCS*, pages 442–451, 2001.

[3] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *Proceedings of FOCS*, pages 189–197, 2000.

[4] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing closeness of discrete distributions. Technical Report abs/1009.5397, 2010. This is a long version of [3].

[5] T. Batu, R. Kumar, and R. Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of STOC*, pages 381–390, 2004.

[6] A. Bhattacharyya, E. Fischer, R. Rubinfeld, and P. Valiant. Testing monotonicity of distributions over general partial orders. In *Proceedings of ITCS*, pages 239–252, 2011.

[7] C. Canonne, D. Ron, and R. Servedio. Testing probability distributions using conditional samples. Technical Report http://arxiv.org/abs/1211.2664, 12 Nov 2012.

[8] S. Chakraborty, E. Fischer, Y. Goldhirsh, and A. Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of ITCS*, 2013. Arxiv posting http://arxiv.org/abs/1210.8338 31 Oct 2012.

[9] C. Daskalakis, I. Diakonikolas, R. Servedio, G. Valiant, and P. Valiant. Testing k-modal distributions: Optimal algorithms via reductions. In *Proceedings of SODA*, 2013.

[10] E. Fischer. The art of uninformed decisions: A primer to property testing. *BEATCS*, 75:97–126, 2001.

[11] O. Goldreich, editor. *Property Testing: Current Research and Surveys*. Springer, 2010. LNCS 6390.

[12] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998.

[13] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. Technical Report TR00-020, ECCC, 2000.

[14] P. Indyk, R. Levi, and R. Rubinfeld. Approximating and Testing k-Histogram Distributions in Sub-linear Time. In *Proceedings of PODS*, pages 15–22, 2012.

[15] S. K. Ma. Calculation of entropy from data of motion. *J. Stat. Phys.*, 26(2), 1981.

[16] Jerzy Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[17] L. Paninski. Estimating entropy on m bins given fewer than m samples. *IEEE-IT*, 50(9):2200–2203, 2004.

[18] L. Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE-IT*, 54(10):4750–4755, 2008.

[19] S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bonds for approximating distributions support size and the distinct elements problem. *SICOMP*, 39(3):813–842, 2009.

[20] D. Ron. Property Testing: A Learning Theory Perspective. *FnTML*, 1(3):307–402, 2008.

[21] D. Ron. Algorithmic and analysis techniques in property testing. *FnTCS*, 5:73–205, 2010.

[22] R. Rubinfeld and R. A. Servedio. Testing monotone high-dimensional distributions. *RSA*, 34(1):24–44, January 2009.

[23] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SICOMP*, 25(2):252–271, 1996.

[24] G. Valiant and P. Valiant. A CLT and tight lower bounds for estimating entropy. Technical Report TR10-179, ECCC, 2010.

[25] G. Valiant and P. Valiant. Estimating the unseen: A sublinear-sample canonical estimator of distributions. Technical Report TR10-180, ECCC, 2010.

[26] G. Valiant and P. Valiant. Estimating the unseen: an $n/\log(n)$-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of STOC*, pages 685–694, 2011. See also [24] and [25].

[27] P. Valiant. Testing symmetric properties of distributions. *SICOMP*, 40(6):1927–1968, 2011.

[28] Wikipedia contributors. Stratified Sampling. http://en.wikipedia.org/wiki/Stratified_sampling, accessed July 1, 2013.

Testing equivalence between distributions using conditional samples*

(Full Version)

Clément Canonne† Dana Ron‡ Rocco A. Servedio§

July 5, 2013

Abstract

We study a recently introduced framework [7, 8] for property testing of probability distributions, by considering distribution testing algorithms that have access to a *conditional sampling oracle.* This is an oracle that takes as input a subset $S \subseteq [N]$ of the domain $[N]$ of the unknown probability distribution D and returns a draw from the conditional probability distribution D restricted to S. This model allows considerable flexibility in the design of distribution testing algorithms; in particular, testing algorithms in this model can be adaptive.

In this paper we focus on algorithms for two fundamental distribution testing problems: testing whether $D = D^*$ for an explicitly provided D^*, and testing whether two unknown distributions D_1 and D_2 are equivalent. We give two efficient algorithms for each of these two problems. At a high level our main finding is that the new "conditional sampling" framework we consider is a powerful one: while both these problems above have $\Omega(\sqrt{N})$ sample complexity in the standard model we give $\mathrm{poly}(\log N, 1/\epsilon)$-query algorithms (and in some cases $\mathrm{poly}(1/\epsilon)$-query algorithms independent of N) for both of them in our conditional sampling setting.

*This paper contains selected results from [7].

†ccanonne@cs.columbia.edu, Columbia University.

‡danaron@tau.ac.il, Tel Aviv University. Supported by ISF grant number 246/08.

§rocco@cs.columbia.edu, Columbia University. Supported by NSF grants CCF-0915929 and CCF-1115703.

1 Introduction

1.1 Background: Distribution testing in the standard model

One of the most fundamental problem paradigms in statistics is that of inferring some information about an unknown probability distribution D given access to independent samples drawn from it. More than a decade ago, Batu et al. [3][1] initiated the study of problems of this type from within the framework of *property testing* [26, 13]. In a property testing problem there is an unknown "massive object" that an algorithm can access only by making a small number of "local inspections" of the object, and the goal is to determine whether the object has a particular property. The algorithm must output ACCEPT if the object has the desired property and output REJECT if the object is far from every object with the property. (See [11, 23, 24, 12] for detailed surveys and overviews of the broad field of property testing; we give precise definitions tailored to our setting in Section 2.)

In distribution property testing the "massive object" is an unknown probability distribution D over an N-element set, and the algorithm accesses the distribution by drawing independent samples from it. A wide range of different properties of probability distributions have been investigated in this setting, and upper and lower bounds on the number of samples required have by now been obtained for many problems. These include testing whether D is uniform [14, 4, 20], testing whether D is identical to a given known distribution D^* [2], testing whether two distributions D_1, D_2 (both available via sample access) are identical [3, 30], and testing whether D has a monotonically increasing probability mass function [6], as well as related problems such as estimating the entropy of D [1, 29], and estimating its support size [21, 30, 29]. Similar problems have also been studied by researchers in other communities, see e.g., [16, 19, 20].

One broad insight that has emerged from this past decade of work is that while sublinear-sample algorithms do exist for many distribution testing problems, the number of samples required is in general quite large. Even the basic problem of testing whether D is the uniform distribution $\mathcal{U}$ over $[N] = \{1, \ldots, N\}$ versus ϵ-far from uniform requires $\Omega(\sqrt{N})$ samples[2] for constant ϵ, and the other problems mentioned above have sample complexities at least this high, and in some cases *almost linear in* N [21, 30, 29]. Since such sample complexities could be prohibitively high in real-world settings where N can be extremely large, it is natural to explore problem variants where it may be possible for algorithms to succeed using fewer samples. Indeed, researchers have studied distribution testing in settings where the unknown distribution is guaranteed to have some special structure, such as being monotone, k-modal or a "k-histogram" over $[N]$ [5, 9, 15], or being monotone over $\{0,1\}^n$ [25] or over other posets [6], and have obtained significantly more sample-efficient algorithms using these additional assumptions.

1.2 Our model: Conditional sampling

In this work we pursue a different line of investigation: rather than restricting the class of probability distributions under consideration, we consider testing algorithms that may use a more powerful form of access to the unknown distribution D. This is a *conditional sampling oracle*, which allows the algorithm to obtain a draw from D_S, the conditional distribution of D restricted to a subset S of the domain (where S is specified by the algorithm). More precisely, we have:

[1] There is a more recent full version of this work [4] and we henceforth reference this recent version.

[2] To verify this, consider the family of all distributions that are uniform over half of the domain, and 0 elsewhere. Each distribution in this family is $\Theta(1)$-far from the uniform distribution. However, it is not possible to distinguish with sufficiently high probability between the uniform distribution and a distribution selected randomly from this family, given a sample of size $\sqrt{N}/c$ (for a sufficiently large constant $c > 1$). This is the case because for the uniform distribution as well as each distribution in this family, the probability of observing the same element more than once is very small. Conditioned on such a collision event not occurring, the samples are distributed identically.

Definition 1 *Fix a distribution D over $[N]$. A* COND *oracle for D, denoted COND_D, is defined as follows: The oracle is given as input a* query set $S \subseteq [N]$ *that has $D(S) > 0$. The oracle returns an element $i \in S$, where the probability that element i is returned is $D_S(i) = D(i)/D(S)$, independently of all previous calls to the oracle.*[3]

For compatibility with our COND_D notation we will write SAMP_D to denote an oracle that takes no input and, each time it is invoked, returns an element from $[N]$ drawn according to D independently from all previous draws. This is the sample access to D that is used in the standard model of testing distributions, and this is of course the same as a call to $\mathsf{COND}_D([N])$.

Motivation and Discussion. One purely theoretical motivation for the study of the COND model is that it may further our understanding regarding what forms of information (beyond standard sampling) can be helpful for testing properties of distributions. In both learning and property testing it is generally interesting to understand how much power algorithms can gain by making queries, and COND queries are a natural type of query to investigate in the context of distributions. As we discuss in more detail below, in several of our results we actually consider restricted versions of COND queries that do not require the full power of obtaining conditional samples from arbitrary sets.

A second attractive feature of the COND model is that it enables a new level of "richness" for algorithms that deal with probability distributions. In the standard model where only access to SAMP_D is provided, all algorithms must necessarily be non-adaptive, with the same initial step of simply drawing a sample of points from SAMP_D, and the difference between two algorithms comes only from how they process their samples. In contrast, the essence of the COND model is to allow algorithms to *adaptively* determine later query sets S based on the outcomes of earlier queries.

A natural question about the COND model is its plausibility: are there settings in which an investigator could actually make conditional samples from a distribution of interest? We feel that the COND framework provides a reasonable "first approximation" for scenarios that arise in application areas (e.g., in biology or chemistry) where the parameters of an experiment can be adjusted so as to restrict the range of possible outcomes. For example, a scientist growing bacteria or yeast cells in a controlled environment may be able to deliberately introduce environmental factors that allow only cells with certain desired characteristics to survive, thus restricting the distribution of all experimental outcomes to a pre-specified subset. We further note that techniques which are broadly reminiscent of COND sampling have long been employed in statistics and polling design under the name of "stratified sampling" (see e.g. [31, 18]). We thus feel that the study of distribution testing in the COND model is well motivated both by theoretical and practical considerations.

Given the above motivations, the central question is whether the COND model enables significantly more efficient algorithms than are possible in the weaker SAMP model. Our results (see Section 1.3) show that this is indeed the case.

Before detailing our results, we note that several of them will in fact deal with a weaker variant of the COND model, which we now describe. In designing COND-model algorithms it is obviously desirable to have algorithms that only invoke the COND oracle on query sets S which are "simple" in some sense. Of course there are many possible notions of "simplicity"; in this work we consider the size of a set as a measure of its simplicity, and consider algorithms which only query small sets. More precisely, we consider

[3]Note that as described above the behavior of $\mathsf{COND}_D(S)$ is undefined if $D(S) = 0$, i.e., the set S has zero probability under D. While various definitional choices could be made to deal with this, we shall assume that in such a case, the oracle (and hence the algorithm) outputs "failure" and terminates. This will not be a problem for us throughout this paper, as (a) our lower bounds deal only with distributions that have $D(i) > 0$ for all $i \in [N]$, and (b) in our algorithms $\mathsf{COND}_D(S)$ will only ever be called on sets S which are "guaranteed" to have $D(S) > 0$. (More precisely, each time an algorithm calls $\mathsf{COND}_D(S)$ it will either be on the set $S = [N]$, or will be on a set S which contains an element i which has been returned as the output of an earlier call to COND_D.)

Problem	Our results		Standard model
Is $D = D^*$ for a known D^*?	COND_D	$\tilde{O}\left(\frac{1}{\epsilon^4}\right)$	$\tilde{\Theta}\left(\frac{\sqrt{N}}{\epsilon^2}\right)$ [2, 20]
	$\mathsf{PCOND}_\mathsf{D}$	$\tilde{O}\left(\frac{\log^4 N}{\epsilon^4}\right)$ $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$	
Are D_1, D_2 (both unknown) equivalent?	COND_{D_1,D_2}	$\tilde{O}\left(\frac{\log^5 N}{\epsilon^4}\right)$	$\tilde{O}\left(\frac{N^{2/3}}{\epsilon^{8/3}}\right)$ [4] $\Omega\left(N^{2/3}\right)$ [4, 30]
	$\mathsf{PCOND}_{\mathsf{D}_1,\mathsf{D}_2}$	$\tilde{O}\left(\frac{\log^6 N}{\epsilon^{21}}\right)$	

Table 1: Comparison between the COND model and the standard model for the problems studied in this paper. The upper bounds are for testing whether the property holds (i.e. $d_{\mathrm{TV}} = 0$) versus $d_{\mathrm{TV}} \geq \epsilon$, and the lower bound is for testing with $\epsilon = \Theta(1)$.

the following restriction of the general COND model: a PCOND (short for "pair-cond") *oracle for* D is a restricted version of COND_D that only accepts input sets S which are either $S = [N]$ (thus providing the power of a SAMP_D oracle) or $S = \{i, j\}$ for some $i, j \in [N]$, i.e. sets of size two. The PCOND oracle may be viewed as a "minimalist" variant of COND that essentially permits an algorithm to compare the relative weights of two items under D (and to draw random samples from D, by setting $S = [N]$).

1.3 Our results

In this early work on the COND model we focus on the simplest (and, we think, most fundamental) concrete problems in distribution testing: specifically, testing whether $D = D^*$ for an explicitly provided D^*, and testing whether $D_1 = D_2$ given COND_{D_1} and COND_{D_2} oracles. We give a detailed study of these two problems in both the COND model and its PCOND variant described above. Our results show that the ability to do conditional sampling provides a significant amount of power to property testers, enabling $\mathrm{polylog}(N)$-query, or even constant-query, algorithms for these problems, both of which have sample complexities $N^{\Omega(1)}$ in the standard model; see Table 1.[4] In what follows d_{TV} denotes the variation distance, that is, $d_{\mathrm{TV}}(D_1, D_2) \stackrel{\mathrm{def}}{=} \frac{1}{2}\|D_1 - D_2\|_1 = \frac{1}{2}\sum_{i\in[N]} |D_1(i) - D_2(i)|$.

Testing equivalence to a known distribution. We consider the question of testing whether D (accessible via a PCOND or COND oracle) is equivalent to D^*, where D^* is an arbitrary "known" distribution over $[N]$ that is explicitly provided to the testing algorithm (say as a vector $(D^*(1), \ldots, D^*(N))$ of probabilities). For this "known D^*" problem, we give a COND_D algorithm testing whether $D = D^*$ versus $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$ using $\tilde{O}(1/\epsilon^4)$ queries (independent of the size of the domain N). We also consider the power of $\mathsf{PCOND}_\mathsf{D}$ oracles for this problem, and give a $\mathsf{PCOND}_\mathsf{D}$ algorithm that uses $\tilde{O}((\log N)^4/\epsilon^4)$ queries. We further show that the $(\log N)^{\Omega(1)}$ query complexity of our $\mathsf{PCOND}_\mathsf{D}$ algorithm is inherent in the problem, by proving that any $\mathsf{PCOND}_\mathsf{D}$ algorithm for this problem must use $\Omega(\sqrt{\log(N)/\log\log(N)})$ queries for constant ϵ.

Testing equivalence between two unknown distributions. We next consider the more challenging problem of testing whether two unknown distributions D_1, D_2 over $[N]$ (available via COND_{D_1} and COND_{D_2} oracles) are identical versus ϵ-far. We give a $\mathrm{poly}(\log N, 1/\epsilon)$ algorithm for this problem in the restricted PCOND model, breaking the $\Omega\left(N^{2/3}\right)$ sample lower bound in the standard model. We also give a completely different algorithm, using general COND queries, that achieves an improved $\mathrm{poly}(\log N, 1/\epsilon)$ query complexity.

[4] [7] is an extended version of this work that gives a broad range of additional results, including both upper and lower bounds, for several other problems and variants of the COND model. See Table 1 of [7] for a concise overview of its results.

Along the way to establishing these testing results, we develop several powerful tools for analyzing distributions in the COND and PCOND models, which we believe may be of independent interest and utility in subsequent work on the COND and PCOND models. These include a procedure for approximately simulating an "evaluation oracle"[5] and a procedure for estimating the weight of the "neighborhood" of a given point in the domain of the distribution. (See further discussion of these tools below.)

1.3.1 A high-level discussion of our algorithms

Our COND- and PCOND- model algorithms are adaptive, and hence necessarily have quite a different algorithmic flavor from distribution testing algorithms in the standard sampling model (which are of course nonadaptive). As can be seen in the following discussion, our various algorithms share some common themes with each other, though each has its own unique idea/technique, which we emphasize below.

For intuition, consider first a special case of **testing equality to a known distribution** D^* where D^* is the uniform distribution over $[N]$. It is not hard to verify that if a distribution D is ϵ-far from uniform, then the following holds: if we select $\Theta(1/\epsilon)$ points according to D and select $\Theta(1/\epsilon)$ points uniformly from $[N]$, then with high constant probability we shall obtain a point x in the first sample, and a point y in the second sample such that $D(x)/D(y)$ is lower bounded by $(1+\Omega(\epsilon))$. This can be detected with high constant probability by performing $\Theta(1/\epsilon^2)$ $\mathsf{PCOND}_\mathsf{D}$ queries on each such pair of points. Since when D^* is the uniform distribution, $D(x)/D(y)=1$ for every pair of points x, y, this provides evidence that $D \neq D^*$, and we can get an algorithm for testing equality to the uniform distribution in the $\mathsf{PCOND}_\mathsf{D}$ model whose complexity is $\mathrm{poly}(1/\epsilon)$. While this simple approach using $\mathsf{PCOND}_\mathsf{D}$ queries succeeds with only $\mathrm{poly}(1/\epsilon)$ queries when D^* is the uniform distribution, we show that for general distributions D^* the query complexity of any $\mathsf{PCOND}_\mathsf{D}$ algorithm for testing equality with D^* must be $(\log N)^{\Omega(1)}$.

In order to obtain an algorithm whose complexity is $\mathrm{poly}(1/\epsilon)$ in the COND_D model we extend the basic idea from the uniform case as follows. Rather than comparing the relative weight of pairs of points, we compare the relative weight of pairs in which one element is a point and the other is a subset of points. Roughly speaking, we show how points can be paired with subsets of points of comparable weight (according to D^*) such that the following holds. If D is far from D^*, then by taking $\tilde{O}(1/\epsilon)$ samples from D and selecting subsets of points in an appropriate manner (depending on D^*), we can obtain (with high probability) a point x and a subset Y such that $D(x)/D(Y)$ differs significantly from $D^*(x)/D^*(Y)$ *and* $D^*(x)/D^*(Y)$ is a constant (the latter is essential for getting $\mathrm{poly}(1/\epsilon)$ query complexity overall).

Returning to the $\mathsf{PCOND}_\mathsf{D}$ model, we show that by sampling from both D and D^* and allowing the number of samples to grow with $\log N$, with high probability we either obtain a pair of points (x, y) such that $D(x)/D(y)$ differs by at least $(1 \pm \Omega(\epsilon))$ from $D^*(x)/D^*(y)$ where $D^*(x)/D^*(y)$ is a constant, or we detect that for some set of points B we have that $|D(B) - D^*(B)|$ is relatively large.[6]

We next turn to the more challenging problem of **testing equality between two unknown distributions** D_1 **and** D_2. In this problem we need to cope with the fact that we no longer "have a hold" on a known distribution. Our PCOND algorithm can be viewed as creating such a hold in the following sense. By sampling from D_1 we obtain (with high probability) a (relatively small) set of points R that *cover* the distribution D_1. By "covering" we mean that except for a subset having small weight according to D_1, all points y in $[N]$ have a *representative* $r \in R$, i.e. a point r such that $D_1(y)$ is close to $D_1(r)$. We then show that if D_2 is far from D_1, then one of the following must hold: (1) There is relatively large weight, either according to D_1 or according to D_2, on points y such that for some $r \in R$ we have that $D_1(y)$ is close to

[5] An EVAL_D oracle (evaluation oracle for D) takes as input a point $i \in [N]$ and outputs the probability $D(i)$ that D puts on i.

[6] Here we use B for "Bucket", as we consider a bucketing of the points in $[N]$ based on their weight according to D^*. We note that bucketing has been used extensively in the context of testing properties of distributions, see e.g. [4, 2].

$D_1(r)$ but $D_2(y)$ is not sufficiently close to $D_2(r)$; (2) There exists a point $r \in R$ such that the set of points y for which $D_1(y)$ is close to $D_1(r)$ has significantly different weight according to D_2 as compared to D_1.

A key subroutine employed by our PCOND algorithm is ESTIMATE-NEIGHBORHOOD, which, given a point x and PCOND access to D returns an estimate of the weight of a subset of points whose probability (according to D) is similar to that of x. The difficulty with performing this task is due to points whose probability is close to the "similarity threshold" that determines the neighborhood set; our ESTIMATE-NEIGHBORHOOD procedure surmounts this difficulty by making a random choice of the similarity threshold. We believe that the ESTIMATE-NEIGHBORHOOD subroutine may be useful in further work as well; indeed [7] uses it in a COND algorithm for estimating the distance between two probability distributions.

Our general COND algorithm for testing the equality of two (unknown) distributions is based on a subroutine that estimates $D(x)$ (to within $(1 \pm O(\epsilon))$) for a given point x given access to COND_D. Obtaining such an estimate for *every* $x \in [N]$ cannot be done efficiently for some distributions.[7] However, we show that if we allow the algorithm to output UNKNOWN on some subset of points with total weight $O(\epsilon)$, then the relaxed task can be performed using $\mathrm{poly}(\log N, 1/\epsilon)$ queries, by performing a kind of randomized binary search "with exceptions". This relaxed version, which we refer to as an *approximate* EVAL *oracle*, suffices for our needs in distinguishing between the case that D_1 and D_2 are the same distribution and the case in which they are far from each other. It is possible that this procedure will be useful for other tasks as well.

1.4 The work of Chakraborty et al. [8]

Chakraborty et al. [8] propose essentially the same COND model that we study, differing only in what happens on query sets S such that $D(S) = 0$. In our model such a query causes the COND oracle and algorithm to return FAIL, while in their model such a query returns a uniform random $i \in S$.

Related to testing equality of distributions, [8] provides an (adaptive) algorithm for testing whether D is equivalent to a specified distribution D^* using $\mathrm{poly}(\log^* N, 1/\epsilon)$ COND queries. Recall that we give an algorithm for this problem that performs $\tilde{O}(1/\epsilon^4)$ COND queries. [8] also gives a *non-adaptive* algorithm for this problem that performs $\mathrm{poly}(\log N, 1/\epsilon)$ COND queries. [8] Testing equivalence between two unknown distributions is not considered in [8], and the same is true for testing in the PCOND model.

Both [8] and [7] also present additional results for a range of other problems (problems which are largely disjoint between the two papers) but we do not discuss those results here.

1.5 Organization

Following some preliminaries in Section 2, in Section 3 we describe and analyze several procedures that are used by our testing algorithms, and may be useful for other algorithms as well. In Section 4 we present our results for testing equivalence to a known distribution, and in Section 5 we present our results for testing equivalence between two unknown distributions. For each of our algorithms, we first give a high-level discussion of the ideas behind it.

[7] As an extreme case consider a distribution D for which $D(1) = 1 - \phi$ and $D(2) = \cdots = D(N) = \phi/(N-1)$ for some very small ϕ (which in particular may depend on N), and for which we are interested in estimating $D(2)$. This requires $\Omega(1/\phi)$ queries.

[8] We note that it is only possible for them to give a non-adaptive algorithm because their model is more permissive than ours (if a query set S is proposed for which $D(S) = 0$, their model returns a uniform random element of S while our model returns FAIL). In our stricter model, any non-adaptive algorithm which queries a proper subset $S \subsetneq N$ would output FAIL on some distribution D.

2 Preliminaries

Throughout the paper we shall work with discrete distributions over an N-element set whose elements are denoted $\{1, \ldots, N\}$; we write $[N]$ to denote $\{1, \ldots, N\}$. For a distribution D over $[N]$ we write $D(i)$ to denote the probability of i under D, and for $S \subseteq [N]$ we write $D(S)$ to denote $\sum_{i \in S} D(i)$. For $S \subseteq [N]$ such that $D(S) > 0$ we write D_S to denote the conditional distribution of D restricted to S, so $D_S(i) = \frac{D(i)}{D(S)}$ for $i \in S$ and $D_S(i) = 0$ for $i \notin S$.

As is standard in property testing of distributions, throughout this work we measure the distance between two distributions D_1 and D_2 using the *total variation distance*:

$$d_{\mathrm{TV}}(D_1, D_2) \stackrel{\mathrm{def}}{=} \frac{1}{2}\|D_1 - D_2\|_1 = \frac{1}{2}\sum_{i \in [N]} |D_1(i) - D_2(i)| = \max_{S \subseteq [N]} |D_1(S) - D_2(S)|.$$

We may view a *property* $\mathcal{P}$ of distributions over $[N]$ as a subset of all distributions over $[N]$ (consisting of all distributions that have the property). The distance from D to a property $\mathcal{P}$, denoted $d_{\mathrm{TV}}(D, \mathcal{P})$, is defined as $\inf_{D' \in \mathcal{P}}\{d_{\mathrm{TV}}(D, D')\}$.

We define testing algorithms for properties of distributions over $[N]$ as follows:

Definition 2 *Let $\mathcal{P}$ be a property of distributions over $[N]$. Let $\mathsf{ORACLE}_{\mathsf{D}}$ be some type of oracle which provides access to D. A $q(\epsilon, N)$-query ORACLE testing algorithm for $\mathcal{P}$ is an algorithm T which is given ϵ, N as input parameters and oracle access to an $\mathsf{ORACLE}_{\mathsf{D}}$ oracle. For any distribution D over $[N]$ algorithm T makes at most $q(\epsilon, N)$ calls to $\mathsf{ORACLE}_{\mathsf{D}}$, and:*

- *if $D \in \mathcal{P}$ then with probability at least $2/3$ algorithm T outputs ACCEPT;*
- *if $d_{\mathrm{TV}}(D, \mathcal{P}) \geq \epsilon$ then with probability at least $2/3$ algorithm T outputs REJECT.*

This definition can easily be extended to cover situations in which there are two "unknown" distributions D_1, D_2 that are accessible via $\mathsf{ORACLE}_{\mathsf{D}_1}$ and $\mathsf{ORACLE}_{\mathsf{D}_2}$ oracles. In particular we shall consider algorithms for testing whether $D_1 = D_2$ versus $d_{\mathrm{TV}}(D_1, D_2)$ in such a setting. We sometimes write $T^{\mathsf{ORACLE}_{\mathsf{D}}}$ to indicate that T has access to $\mathsf{ORACLE}_{\mathsf{D}}$.

In Appendix A we give a range of useful but standard tools from probability (the data processing inequality for total variation distance and several variants of Chernoff bounds.).

3 Some useful procedures

In this section we describe three procedures that will be used by our testing algorithms: COMPARE, ESTIMATE-NEIGHBORHOOD and APPROX-EVAL-SIMULATOR. We present these tools in increasing order of sophistication: COMPARE is quite straightforward, while the algorithm and analysis of APPROX-EVAL-SIMULATOR are both fairly complex. On a first pass the reader may wish to focus on the explanatory prose and performance guaranteees of these procedures (i.e. the statements of Lemma 1, Lemma 2 and Theorem 1); the internal details of the proofs in this section are not necessary for the subsequent sections which use these procedures.

Algorithm 1: COMPARE

Input: COND query access to a distribution D over $[N]$, disjoint subsets $X, Y \subset [N]$, parameters $\eta \in (0,1]$, $K \geq 1$, and $\delta \in (0, 1/2]$.

1. Perform $\Theta\left(\frac{K \log(1/\delta)}{\eta^2}\right)$ COND_D queries on the set $S = X \cup Y$, and let $\hat{\mu}$ be the fraction of times that a point $y \in Y$ is returned.
2. If $\hat{\mu} < \frac{2}{3} \cdot \frac{1}{K+1}$, then return Low.
3. Else, if $1 - \hat{\mu} < \frac{2}{3} \cdot \frac{1}{K+1}$, then return High.
4. Else return $\rho = \frac{\hat{\mu}}{1-\hat{\mu}}$.

3.1 The procedure COMPARE

We start by describing a procedure that estimates the ratio between the weights of two disjoint sets of points by performing COND queries on the union of the sets. In the special case when each set is of size one, the queries performed are PCOND queries.

Lemma 1 *Given as input two disjoint subsets of points X, Y together with parameters $\eta \in (0,1]$, $K \geq 1$, and $\delta \in (0, 1/2]$, as well as* COND *query access to a distribution D, the procedure* COMPARE *(Algorithm 1) either outputs a value $\rho > 0$ or outputs* High *or* Low*, and satisfies the following:*

1. *If $D(X)/K \leq D(Y) \leq K \cdot D(X)$ then with probability at least $1 - \delta$ the procedure outputs a value $\rho \in [1-\eta, 1+\eta] D(Y)/D(X)$;*
2. *If $D(Y) > K \cdot D(X)$ then with probability at least $1 - \delta$ the procedure outputs either* High *or a value $\rho \in [1-\eta, 1+\eta] D(Y)/D(X)$;*
3. *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 $O\left(\frac{K \log(1/\delta)}{\eta^2}\right)$ COND queries on the set $X \cup Y$.

Proof: The bound on the number of queries performed by the algorithm follows directly from the description of the algorithm, and hence we turn to establish its correctness.

Let $w(X) = \frac{D(X)}{D(X)+D(Y)}$ and let $w(Y) = \frac{D(Y)}{D(X)+D(Y)}$. Observe that $\frac{w(Y)}{w(X)} = \frac{D(Y)}{D(X)}$ and that for $\hat{\mu}$ as defined in Line 1 of the algorithm, $\mathrm{E}[\hat{\mu}] = w(Y)$ and $\mathrm{E}[1 - \hat{\mu}] = w(X)$. Also observe that for any $B \geq 1$, if $D(Y) \geq D(X)/B$, then $w(Y) \geq \frac{1}{B+1}$ and if $D(Y) \leq B \cdot D(X)$, then $w(X) \geq \frac{1}{B+1}$.

Let E_1 be the event that $\hat{\mu} \in [1-\eta/3, 1+\eta/3] w(Y)$ and let E_2 be the event that $(1-\hat{\mu}) \in [1-\eta/3, 1+\eta/3] w(X)$. Given the number of COND queries performed on the set $X \cup Y$, by applying a multiplicative Chernoff bound (see Theorem 9), if $w(Y) \geq \frac{1}{4K}$ then with probability at least $1 - \delta/2$ the event E_1 holds, and if $w(X) \geq \frac{1}{4K}$, then with probability at least $1 - \delta/2$ the event E_2 holds. We next consider the three cases in the lemma statement.

1. If $D(X)/K \leq D(Y) \leq KD(X)$, then by the discussion above, $w(Y) \geq \frac{1}{K+1}$, $w(X) \geq \frac{1}{K+1}$, and with probability at least $1-\delta$ we have that $\hat{\mu} \in [1-\eta/3, 1+\eta/3]w(Y)$ and $(1-\hat{\mu}) \in [1-\eta/3, 1+\eta/3]w(X)$. Conditioned on these bounds holding,

$$\hat{\mu} \geq \frac{1-\eta/3}{K+1} \geq \frac{2}{3} \cdot \frac{1}{K+1} \text{ and } 1-\hat{\mu} \geq \frac{2}{3} \cdot \frac{1}{K+1} .$$

   It follows that the procedure outputs a value $\rho = \frac{\hat{\mu}}{1-\hat{\mu}} \in [1-\eta, 1+\eta]\frac{w(Y)}{w(X)}$ as required by Item 1.

2. If $D(Y) > K \cdot D(X)$, then we consider two subcases.

   (a) If $D(Y) > 3K \cdot D(X)$, then $w(X) < \frac{1}{3K+1}$, so that by a multiplicative Chernoff bound (stated in Corollary 10), with probability at least $1-\delta$ we have that

$$1-\hat{\mu} < \frac{1+\eta/3}{3K+1} \leq \frac{4}{3} \cdot \frac{1}{3K+1} \leq \frac{2}{3} \cdot \frac{1}{K+1},$$

   causing the algorithm to output High. Thus Item 2 is established for this subcase.

   (b) If $K \cdot D(X) < D(Y) \leq 3K \cdot D(X)$, then $w(X) \geq \frac{1}{3K+1}$ and $w(Y) \geq \frac{1}{2}$, so that the events E_1 and E_2 both hold with probability at least $1-\delta$. Assume that these events in fact hold. This implies that $\hat{\mu} \geq \frac{1-\eta/3}{2} \geq \frac{2}{3} \cdot \frac{1}{K+1}$, and the algorithm either outputs High or outputs $\rho = \frac{\hat{\mu}}{1-\hat{\mu}} \in [1-\eta, 1+\eta]\frac{w(Y)}{w(X)}$, so Item 2 is established for this subcase as well.

3. If $D(Y) < D(X)/K$, so that $D(X) > K \cdot D(Y)$, then the exact same arguments are applied as in the previous case, just switching the roles of Y and X and the roles of $\hat{\mu}$ and $1-\hat{\mu}$ so as to establish Item 3.

We have thus established all items in the lemma. ■

3.2 The procedure ESTIMATE-NEIGHBORHOOD

In this subsection we describe a procedure that, given a point x, provides an estimate of the weight of a set of points y such that $D(y)$ is similar to $D(x)$. In order to specify the behavior of the procedure more precisely, we introduce the following notation. For a distribution D over $[N]$, a point $x \in [N]$ and a parameter $\gamma \in [0,1]$, let

$$U_\gamma^D(x) \stackrel{\text{def}}{=} \left\{ y \in [N] : \frac{1}{1+\gamma} D(x) \leq D(y) \leq (1+\gamma) D(x) \right\} \tag{1}$$

denote the set of points whose weight is "γ-close" to the weight of x. If we take a sample of points distributed according to D, then the expected fraction of these points that belong to $U_\gamma^D(x)$ is $D(U_\gamma^D(x))$. If this value is not too small, then the actual fraction in the sample is close to the expected value. Hence, if we could efficiently determine for any given point y whether or not it belongs to $U_\gamma^D(x)$, then we could obtain a good estimate of $D(U_\gamma^D(x))$. The difficulty is that it is not possible to perform this task efficiently for "boundary" points y such that $D(y)$ is very close to $(1+\gamma)D(x)$ or to $\frac{1}{1+\gamma}D(x)$. However, for our purposes, it is not important that we obtain the weight and size of $U_\gamma^D(x)$ for a specific γ, but rather it suffices to do so for γ in a given range, as stated in the next lemma.

Lemma 2 *Given as input a point x together with parameters $\kappa, \beta, \eta, \delta \in (0, 1/2]$ as well as* PCOND *query access to a distribution D, the procedure* ESTIMATE-NEIGHBORHOOD *(Algorithm 2) outputs a pair $(\hat{w}, \alpha) \in [0,1] \times (\kappa, 2\kappa)$ such that α is uniformly distributed in $\{\kappa + i\theta\}_{i=0}^{\kappa/\theta - 1}$ for $\theta = \frac{\kappa\eta\beta\delta}{64}$, and such that the following holds:*

1. *If $D(U_\alpha^D(x)) \geq \beta$, then with probability at least $1 - \delta$ we have $\hat{w} \in [1-\eta, 1+\eta] \cdot D(U_\alpha^D(x))$, and $D(U_{\alpha+\theta}^D(x) \setminus U_\alpha^D(x)) \leq \eta\beta/16$;*
2. *If $D(U_\alpha^D(x)) < \beta$, then with probability at least $1 - \delta$ we have $\hat{w} \leq (1+\eta) \cdot \beta$, and $D(U_{\alpha+\theta}^D(x) \setminus U_\alpha^D(x)) \leq \eta\beta/16$.*

The number of PCOND *queries performed by the procedure is $O\Big(\frac{\log(1/\delta) \cdot \log(\log(1/\delta)/(\beta\eta^2))}{\kappa^2\eta^4\beta^3\delta^2}\Big)$.*

Algorithm 2: ESTIMATE-NEIGHBORHOOD

Input: PCOND query access to a distribution D over $[N]$, a point $x \in [N]$ and parameters $\kappa, \beta, \eta, \delta \in (0, 1/2]$

1: Set $\theta = \frac{\kappa\eta\beta\delta}{64}$ and $r = \frac{\kappa}{\theta} = \frac{64}{\eta\beta\delta}$.
2: Select a value $\alpha \in \{\kappa + i\theta\}_{i=0}^{r-1}$ uniformly at random.
3: Call the $\mathsf{SAMP_D}$ oracle $\Theta(\log(1/\delta)/(\beta\eta^2))$ times and let S be the set of points obtained.
4: For each point y in S call COMPARE$_D(\{x\}, \{y\}, \theta/4, 4, \delta/(4|S|))$ (if a point y appears more than once in S, then COMPARE is called only once on y).
5: Let $\hat{w}$ be the fraction of occurrences of points y in S for which COMPARE returned a value $\rho(y) \in [1/(1+\alpha+\theta/2), (1+\alpha+\theta/2)]$. (That is, S is viewed as a multiset.)
6: Return $(\hat{w}, \alpha)$.

Proof of Lemma 2: The number of PCOND queries performed by ESTIMATE-NEIGHBORHOOD is the size of S times the number of PCOND queries performed in each call to COMPARE. By the setting of the parameters in the calls to COMPARE, the total number of PCOND queries is $O\Big(\frac{(|S|)\cdot\log|S|/\delta)}{\theta^2}\Big) = O\Big(\frac{\log(1/\delta)\cdot\log(\log(1/\delta)//(\beta\eta^2))}{\kappa^2\eta^4\beta^3\delta^2}\Big)$. We now turn to establishing the correctness of the procedure.

Since D and x are fixed, in what follows we shall use the shorthand U_γ for $U_\gamma^D(x)$. For $\alpha \in \{\kappa + i\theta\}_{i=0}^{r-1}$, let $\Delta_\alpha \stackrel{\text{def}}{=} U_{\alpha+\theta} \setminus U_\alpha$. We next define several "desirable" events. In all that follows we view S as a multiset.

1. Let E_1 be the event that $D(\Delta_\alpha) \leq 4/(\delta r))$. Since there are r disjoint sets Δ_α for $\alpha \in \{\kappa + i\theta\}_{i=0}^{r-1}$, the probability that E_1 occurs (taken over the uniform choice of α) is at least $1 - \delta/4$. From this point on we fix α and assume E_1 holds.

2. The event E_2 is that $|S \cap \Delta_\alpha|/|S| \leq 8/(\delta r)$ (that is, at most twice the upper bound on the expected value). By applying the multiplicative Chernoff bound using the fact that $|S| = \Theta(\log(1/\delta)/(\beta\eta^2)) = \Omega(\log(1/\delta) \cdot (\delta r))$, we have that $\Pr_S[E_2] \geq 1 - \delta/4$.

3. The event E_3 is defined as follows: If $D(U_\alpha) \geq \beta$, then $|S \cap U_\alpha|/|S| \in [1 - \eta/2, 1 + \eta/2] \cdot D(U_\alpha)$, and if $D(U_\alpha) < \beta$, then $|S \cap U_\alpha|/|S| < (1 + \eta/2) \cdot \beta$. Once again applying the multiplicative Chernoff bound (for both cases) and using that fact that $|S| = \Theta(\log(1/\delta)/(\beta\eta^2))$, we have that $\Pr_S[E_3] \geq 1 - \delta/4$.

4. Let E_4 be the event that all calls to COMPARE return an output as specified in Lemma 1. Given the setting of the confidence parameter in the calls to COMPARE we have that $\Pr[E_4] \geq 1 - \delta/4$ as well.

Assume from this point on that events E_1 through E_4 all hold where this occurs with probability at least $1 - \delta$. By the definition of Δ_α and E_1 we have that $D(U_{\alpha+\theta} \setminus U_\alpha) \leq 4/(\delta r) = \eta\beta/16$, as required (in both items of the lemma). Let T be the (multi-)subset of points y in S for which COMPARE returned a value $\rho(y) \in [1/(1+\alpha+\theta/2), (1+\alpha+\theta/2)]$ (so that $\hat{w}$, as defined in the algorithm, equals $|T|/|S|$). Note first that conditioned on E_4 we have that for every $y \in U_{2\kappa}$ it holds that the output of COMPARE when called on $\{x\}$ and $\{y\}$, denoted $\rho(y)$, satisfies $\rho(y) \in [1-\theta/4, 1+\theta/4](D(y)/D(x))$, while for $y \notin U_{2\kappa}$ either COMPARE outputs High or Low or it outputs a value $\rho(y) \in [1 - \theta/4, 1 + \theta/4](D(y)/D(x))$. This implies that if $y \in U_\alpha$, then $\rho(y) \leq (1+\alpha)\cdot(1+\theta/4) \leq 1+\alpha+\theta/2$ and $\rho(y) \geq (1+\alpha)^{-1}\cdot(1-\theta/4) \geq (1+\alpha+\theta/2)^{-1}$, so that $S \cap U_{\alpha)} \subseteq T$. On the other hand, if $y \notin U_{\alpha+\theta}$ then either $\rho(y) > (1+\alpha+\theta)\cdot(1-\theta/4) \geq 1+\alpha+\theta/2$ or $\rho(y) < (1+\alpha+\theta)^{-1}\cdot(1+\theta/4) \leq (1+\alpha+\theta/2)^{-1}$ so that $T \subseteq S \cap U_{\alpha+\theta}$. Combining the two we have:

$$S \cap U_\alpha \;\subseteq\; T \;\subseteq\; S \cap U_{\alpha+\theta}\,. \tag{2}$$

Recalling that $\hat{w} = \frac{|T|}{|S|}$, the left-hand side of Equation (2) implies that

$$\hat{w} \geq \frac{|S \cap U_\alpha|}{|S|}\,, \tag{3}$$

and by E_1 and E_2, the right-hand-side of Equation (2) implies that

$$\hat{w} \;\leq\; \frac{|S \cap U_{\alpha)}|}{|S|} + \frac{8}{\delta r} \;\leq\; \frac{|S \cap U_\alpha|}{|S|} + \frac{\beta\eta}{8}\,. \tag{4}$$

We consider the two cases stated in the lemma:

1. If $D(U_\alpha) \geq \beta$, then by Equation (3), Equation (4) and (the first part of) E_3, we have that $\hat{w} \in [1-\eta, 1+\eta]\cdot D(U_\alpha)$.
2. If $D(U_\alpha) < \beta$, then by Equation (4) and (the second part of) E_3, we have that $\hat{w} \leq (1+\eta)\beta$.

The lemma is thus established. ■

3.3 Approximate EVAL oracles

We begin by defining the notion of an "approximate EVAL oracle" that we will use. Intuitively this is an oracle which gives a multiplicatively $(1 \pm \epsilon)$-accurate estimate of the value of $D(i)$ for all i in a fixed set of probability weight at least $1 - \epsilon$ under D. More precisely, we have the following definition:

Definition 3 *Let D be a distribution over $[N]$. An (ϵ, δ)-*approximate EVAL_D simulator *is a randomized procedure* ORACLE *with the following property: For each $0 < \epsilon < 1$, there is a fixed set $S^{(\epsilon,D)} \subsetneq [N]$ with $D(S^{(\epsilon,D)}) < \epsilon$ for which the following holds. Given as input an element $i^* \in [N]$, the procedure* ORACLE *either outputs a value $\alpha \in [0,1]$ or outputs* UNKNOWN. *The following holds for all $i^* \in [N]$:*

(i) If $i^* \notin S^{(\epsilon,D)}$ then with probability at least $1-\delta$ the output of ORACLE *on input i^* is a value $\alpha \in [0,1]$ such that $\alpha \in [1-\epsilon, 1+\epsilon]D(i^*)$;*

(i) If $i^* \in S^{(\epsilon,D)}$ then with probability at least $1-\delta$ the procedure either outputs UNKNOWN *or outputs a value $\alpha \in [0,1]$ such that $\alpha \in [1-\epsilon, 1+\epsilon]D(i^*)$.*

We note that according to the above definition, it may be the case that different calls to ORACLE on the same input element $i^* \in [N]$ may return different values. However, the "low-weight" set $S^{(\epsilon,D)}$ is an *a priori* fixed set that does not depend in any way on the input point i^* given to the algorithm. The key property of an (ϵ,δ)-approximate EVAL_D oracle is that it reliably gives a multiplicatively $(1 \pm \epsilon)$-accurate estimate of the value of $D(i)$ for all i in some fixed set of probability weight at least $1-\epsilon$ under D.

3.3.1 Constructing an approximate EVAL_D simulator using COND_D

In this subsection we show that a COND_D oracle can be used to obtain an approximate EVAL simulator:

Theorem 1 *Let D be any distribution over $[N]$ and let $0 < \epsilon, \delta < 1$. The algorithm* APPROX-EVAL-SIMULATOR *has the following properties: It uses*

$$\tilde{O}\left(\frac{(\log N)^5 \cdot (\log(1/\delta))^2}{\epsilon^3}\right)$$

calls to COND_D and it is an (ϵ,δ)-approximate EVAL_D simulator.

A few notes: First, in the proof we give below of Theorem 1 we assume throughout that $0 < \epsilon \leq c$, where c is a small absolute constant. This incurs no loss of generality because if the desired ϵ parameter is in $(c,1)$ then the parameter can simply be set to $c/2$. We further note that in keeping with our requirement on a COND_D algorithm, the algorithm APPROX-EVAL-SIMULATOR only ever calls the COND_D oracle on sets S which are either $S = [N]$ or else contain at least one element i that has been returned as the output of an earlier call to COND_D. (To see this, note that Line 6 is the only line when COND_D queries are performed. In the first execution of the outer "For" loop clearly all COND queries are on set $S_0 = [N]$. In subsequent stages the only way a set S_j is formed is if either (i) S_j is set to $\{i^*\}$ in Line 10, in which case clearly i^* was previously received as the response of a $\mathsf{COND}_D(S_{j-1})$ query, or else (ii) a nonzero fraction of elements $i_1, \ldots, i_m$ received as responses to $\mathsf{COND}_D(S_{j-1})$ queries belong to S_j (see Line 19).)

A preliminary simplification. Fix a distribution D over $[N]$. Let Z denote $\mathrm{supp}(D)$, i.e. $Z = \{i \in [N] : D(i) > 0\}$. We first claim that in proving Theorem 1 we may assume without loss of generality that no two distinct elements $i, j \in Z$ have $D(i) = D(j)$ – in other words, we shall prove the theorem under this assumption on D, and we claim that this implies the general result. To see this, observe that if Z contains elements $i \neq j$ with $D(i) = D(j)$, then for any arbitrarily small $\xi > 0$ and any arbitrarily large M we can perturb the weights of elements in Z to obtain a distribution D' supported on Z such that (i) no two elements of Z have the same probability under D', and (ii) for every $S \subseteq [N]$, $S \cap Z \neq \emptyset$ we have $d_{\mathrm{TV}}(D_S, D'_S) \leq \xi/M$. Since the variation distance between D'_S and D_S is at most ξ/M for an arbitrarily small ξ, the variation distance between (the execution of any M-query COND algorithm run on D) and (the execution of any M-query COND algorithm run on D') will be at most ξ. Since ξ can be made arbitrarily small this means that indeed without loss of generality we may work with D' in what follows.

Thus, we henceforth assume that the distribution D has no two elements in $\mathrm{supp}(D)$ with the same weight. For such a distribution we can explicitly describe the set $S^{(\epsilon,D)}$ from Definition 3 that our analysis will deal with. Let $\pi : \{1, \ldots, |Z|\} \to Z$ be the bijection such that $D(\pi(1)) > \cdots > D(\pi(|Z|))$ (note that the bijection π is uniquely defined by the assumption that $D(i) \neq D(j)$ for all distinct $i, j \in Z$). Given a value $0 < \tau < 1$ we define the set $L_{\tau,D}$ to be $([N] \setminus Z) \cup \{\pi(s), \ldots, \pi(|Z|)\}$ where s is the smallest index in

$\{1, \ldots, |Z|\}$ such that $\sum_{j=s}^{|Z|} D(\pi(j)) < \tau$ (if $D(\pi(|Z|))$ itself is at least τ then we define $L_{\tau,D} = [N] \setminus Z$). Thus intuitively $L_{\tau,D}$ contains the τ fraction (w.r.t. D) of $[N]$ consisting of the lightest elements. The desired set $S^{(\epsilon,D)}$ is precisely $L_{\epsilon,D}$.

Intuition for the algorithm. The high-level idea of the EVAL_D simulation is the following: Let $i^* \in [N]$ be the input element given to the EVAL_D simulator. The algorithm works in a sequence of stages. Before performing the j-th stage it maintains a set S_{j-1} that contains i^*, and it has a high-accuracy estimate $\hat{D}(S_{j-1})$ of the value of $D(S_{j-1})$. (The initial set S_0 is simply $[N]$ and the initial estimate $\hat{D}(S_0)$ is of course 1.) In the j-th stage the algorithm attempts to construct a subset S_j of S_{j-1} in such a way that (i) $i^* \in S_j$, and (ii) it is possible to obtain a high-accuracy estimate of $D(S_j)/D(S_{j-1})$ (and thus a high-accuracy estimate of $D(S_j)$). If the algorithm cannot construct such a set S_j then it outputs $\mathsf{UNKNOWN}$; otherwise, after at most (essentially) $O(\log N)$ stages, it reaches a situation where $S_j = \{i^*\}$ and so the high-accuracy estimate of $D(S_j) = D(i^*)$ is the desired value.

A natural first idea towards implementing this high-level plan is simply to split S_{j-1} randomly into two pieces and use one of them as S_j. However this simple approach may not work; for example, if S_{j-1} has one or more elements which are very heavy compared to i^*, then with a random split it may not be possible to efficiently estimate $D(S_j)/D(S_{j-1})$ as required in (ii) above. Thus we follow a more careful approach which first identifies and removes "heavy" elements from S_{j-1} in each stage.

In more detail, during the j-th stage, the algorithm first performs COND_D queries on the set S_{j-1} to identify a set $H_j \subseteq S_{j-1}$ of "heavy" elements; this set essentially consists of all elements which individually each contribute at least a κ fraction of the total mass $D(S_{j-1})$. (Here κ is a "not-too-small" quantity but it is significantly less than ϵ.) Next, the algorithm performs additional COND_D queries to estimate $D(i^*)/D(S_{j-1})$. If this fraction exceeds $\kappa/20$ then it is straightforward to estimate $D(i^*)/D(S_{j-1})$ to high accuracy, so using $\hat{D}(S_{j-1})$ it is possible to obtain a high-quality estimate of $D(i^*)$ and the algorithm can conclude. However, the typical case is that $D(i^*)/D(S_{j-1}) < \kappa/20$. In this case, the algorithm next estimates $D(H_j)/D(S_{j-1})$. If this is larger than $1 - \epsilon/10$ then the algorithm outputs $\mathsf{UNKNOWN}$ (see below for more discussion of this). If $D(H_j)/D(S_{j-1})$ is less than $1 - \epsilon/10$ then $D(S_{j-1} \setminus H_j)/D(S_{j-1}) \geq \epsilon/10$ (and so $D(S_{j-1} \setminus H_j)/D(S_{j-1})$ can be efficiently estimated to high accuracy), but each element k of $S_{j-1} \setminus H_j$ has $D(k)/D(S_{j-1}) \leq \kappa \ll \epsilon/10 \leq D(S_{j-1} \setminus H_j)/D(S_{j-1})$. Thus it must be the case that the weight under D of $S_{j-1} \setminus H_j$ is "spread out" over many "light" elements.

Given that this is the situation, the algorithm next chooses S'_j to be a random subset of $S_{j-1} \setminus (H_j \cup \{i^*\})$, and sets S_j to be $S'_j \cup \{i^*\}$. It can be shown that with high probability (over the random choice of S_j) it will be the case that $D(S_j) \geq \frac{1}{3} D(S_{j-1} \setminus H_j)$ (this relies crucially on the fact that the weight under D of $S_{j-1} \setminus H_j$ is "spread out" over many "light" elements). This makes it possible to efficiently estimate $D(S_j)/D(S_{j-1} \setminus H_j)$; together with the high-accuracy estimate of $D(S_{j-1} \setminus H_j)/D(S_{j-1})$ noted above, and the high-accuracy estimate $\hat{D}(S_{j-1})$ of $D(S_{j-1})$, this means it is possible to efficiently estimate $D(S_j)$ to high accuracy as required for the next stage. (We note that after defining S_j but before proceeding to the next stage, the algorithm actually checks to be sure that S_j contains at least one point that was returned from the $\mathsf{COND}_D(S_{j-1})$ calls made in the past stage. This check ensures that whenever the algorithm calls $\mathsf{COND}_D(S)$ on a set S, it is guaranteed that $D(S) > 0$ as required by our COND_D model. Our analysis shows that doing this check does not affect correctness of the algorithm since with high probability the check always passes.)

Intuition for the analysis. We require some definitions to give the intuition for the analysis establishing correctness. Fix a nonempty subset $S \subseteq [N]$. Let π_S be the bijection mapping $\{1, \ldots, |S|\}$ to S in such a way that $D_S(\pi_S(1)) > \cdots > D_S(\pi_S(|S|))$, i.e. $\pi_S(1), \ldots, \pi_S(|S|)$ is a listing of the elements of S in order from heaviest under D_S to lightest under D_S. Given $j \in S$, we define the *S-rank of j*, denoted $\mathrm{rank}_S(j)$,

to be the value $\sum_{i:D_S(\pi(i))\leq D_S(j)} D_S(\pi(i))$, i.e. $\mathrm{rank}_S(j)$ is the sum of the weights (under D_S) of all the elements in S that are no heavier than j under D_S. Note that having $i^* \notin L_{\epsilon,N}$ implies that $\mathrm{rank}_{[N]}(i^*) \geq \epsilon$.

We first sketch the argument for correctness. (It is easy to show that the algorithm only outputs FAIL with very small probability so we ignore this possibility below.) Suppose first that $i^* \notin L_{\epsilon,D}$. A key lemma shows that if $i^* \notin L_{\epsilon,D}$ (and hence $\mathrm{rank}_{[N]}(i^*) \geq \epsilon$), then with high probability every set S_{j-1} constructed by the algorithm is such that $\mathrm{rank}_{S_{j-1}}(i^*) \geq \epsilon/2$. (In other words, if i^* is not initially among the ϵ-fraction (under D) of lightest elements, then it never "falls too far" to become part of the $\epsilon/2$-fraction (under $D_{S_{j-1}}$) of lightest elements for S_{j-1}, for any j.). Given that (whp) i^* always has $\mathrm{rank}_{S_{j-1}}(i^*) \geq \epsilon/2$, though, then it must be the case that (whp) the procedure does not output UNKNOWN (and hence it must whp output a numerical value). This is because there are only two places where the procedure can output UNKNOWN, in Lines 14 and 19; we consider both cases below.

1. In order for the procedure to output UNKNOWN in Line 14, it must be the case that the elements of H_j – each of which individually has weight at least $\kappa/2$ under $D_{S_{j-1}}$ – collectively have weight at least $1 - 3\epsilon/20$ under $D_{S_{j-1}}$ by Line 13. But i^* has weight at most $3\kappa/40$ under $D_{S_{j-1}}$ (because the procedure did not go to Line 2 in Line 10), and thus i^* would need to be in the bottom $3\epsilon/20$ of the lightest elements, i.e. it would need to have $\mathrm{rank}_{S_{j-1}}(i^*) \leq 3\epsilon/20$; but this contradicts $\mathrm{rank}_{S_{j-1}}(i^*) \geq \epsilon/2$.

2. In order for the procedure to output UNKNOWN in Line 19, it must be the case that all elements $i_1, \ldots, i_m$ drawn in Line 6 are not chosen for inclusion in S_j. In order for the algorithm to reach Line 19, though, it must be the case that at least $(\epsilon/10 - \kappa/20)m$ of these draws do not belong to $H_j \cup \{i^*\}$; since these draws do not belong to H_j each one occurs only a small number of times among the m draws, so there must be many distinct values, and hence the probability that none of these distinct values is chosen for inclusion in S'_j is very low.

Thus we have seen that if $i^* \notin L_{\epsilon,D}$, then whp the procedure outputs a numerical value; it remains to show that whp this value is a high-accuracy estimate of $D(i^*)$. However, this follows easily from the fact that we inductively maintain a high-quality estimate of $D(S_{j-1})$ and the fact that the algorithm ultimately constructs its estimate of $\hat{D}(i^*)$ only when it additionally has a high-quality estimate of $D(i^*)/D(S_{j-1})$. This fact also handles the case in which $i^* \in L_{\epsilon,D}$ – in such a case it is allowable for the algorithm to output UNKNOWN, so since the algorithm w.h.p. outputs a high-accuracy estimate when it outputs a numerical value, this means the algorithm performs as required in Case (ii) of Definition 3.

We now sketch the argument for query complexity. We will show that the heavy elements can be identified in each stage using $\mathrm{poly}(\log N, 1/\epsilon)$ queries. Since the algorithm constructs S_j by taking a random subset of S_{j-1} (together with i^*) at each stage, the number of stages is easily bounded by (essentially) $O(\log N)$. Since the final probability estimate for $D(i^*)$ is a product of $O(\log N)$ conditional probabilities, it suffices to estimate each of these conditional probabilities to within a multiplicative factor of $(1 \pm O\left(\frac{\epsilon}{\log N}\right))$. We show that each conditional probability estimate can be carried out to this required precision using only $\mathrm{poly}(\log N, 1/\epsilon)$ calls to COND_D; given this, the overall $\mathrm{poly}(\log N, 1/\epsilon)$ query bound follows straightforwardly.

Now we enter into the actual proof. We begin our analysis with a simple but useful lemma about the "heavy" elements identified in Line 7.

Lemma 3 *With probability at least $1 - \delta/K$, every set H_j that is ever constructed in Line 7 satisfies the following for all $\ell \in S_{j-1}$:*

Algorithm 3: APPROX-EVAL-SIMULATOR

Input: access to COND_D; parameters $0 < \epsilon, \delta < 1$; input element $i^* \in [N]$

1: Set $S_0 = [N]$ and $\hat{D}(S_0) = 1$. Set $K = 9$. Set $M = \log N + \log(K/\delta) + 1$. Set $\kappa = \Theta(\epsilon/(M^2 \log(M/\delta)))$.
2: **for** $j = 1$ to M **do**
3: **if** $|S_{j-1}| = 1$ **then**
4: return $\hat{D}(S_{j-1})$ (and exit)
5: **end if**
6: Perform $m = \Theta(\max\{M^2 \log(M/\delta)/(\epsilon^2\kappa), \log(M/(\delta\kappa))/\kappa^2\})$ COND_D queries on S_{j-1} to obtain points $i_1, \ldots, i_m \in S_{j-1}$.
7: Let $H_j = \{k \in [N] : k$ appears at least $\frac{3}{4}\kappa m$ times in the list $i_1, \ldots, i_m\}$
8: Let $\hat{D}_{S_{j-1}}(i^*)$ denote the fraction of times that i^* appears in $i_1, \ldots, i_m$
9: **if** $\hat{D}_{S_{j-1}}(i^*) \geq \frac{\kappa}{20}$ **then**
10: Set $S_j = \{i^*\}$, set $\hat{D}(S_j) = \hat{D}_{S_{j-1}}(i^*) \cdot \hat{D}(S_{j-1})$, increment j, and go to Line 2.
11: **end if**
12: Let $\hat{D}_{S_{j-1}}(H_j)$ denote the fraction of elements among $i_1, \ldots, i_m$ that belong to H_j.
13: **if** $\hat{D}_{S_{j-1}}(H_j) > 1 - \epsilon/10$ **then**
14: return UNKNOWN (and exit)
15: **end if**
16: Set S'_j to be a uniform random subset of $S_{j-1} \setminus (H_j \cup \{i^*\})$ and set S_j to be $S'_j \cup \{i^*\}$.
17: Let $\hat{D}_{S_{j-1}}(S_j)$ denote the fraction of elements among $i_1, \ldots, i_m$ that belong to S_j
18: **if** $\hat{D}_{S_{j-1}}(S_j) = 0$ **then**
19: return UNKNOWN (and exit)
20: **end if**
21: Set $\hat{D}(S_j) = \hat{D}_{S_{j-1}}(S_j) \cdot \hat{D}(S_{j-1})$
22: **end for**
23: Output FAIL.

(i) If $D(\ell)/D(S_{j-1}) > \kappa$, then $\ell \in H_j$;

(ii) If $D(\ell)/D(S_{j-1}) < \kappa/2$ then $\ell \notin H_j$.

Proof: Fix an iteration j. By Line 7 in the algorithm, a point ℓ is included in H_j if it appears at least $\frac{3}{4}\kappa m$ times among $i_1, \ldots, i_m$ (which are the output of COND_D queries on S_{j-1}). For the first item, fix an element ℓ such that $D(\ell)/D(S_{j-1}) > \kappa$. Recall that $m = \Omega(M^2 \log(M/\delta)/(\epsilon^2\kappa)) = \Omega(\log(MN/\delta)/\kappa)$ (since $M = \Omega(\log(N))$). By a multiplicative Chernoff bound, the probability (over the choice of $i_1, \ldots, i_m$ in S_{j-1}) that ℓ appears less than $\frac{3}{4}\kappa m$ times among $i_1, \ldots, i_m$ (that is, less than $3/4$ times the lower bound on the expected value) is at most $\delta/(KMN)$ (for an appropriate constant in the setting of m). On the other hand, for each fixed ℓ such that $D(\ell)/D(S_{j-1}) < \kappa/2$, the probability that ℓ appears at least $\frac{3}{4}\kappa m$ times (that is, at least $3/2$ times the upper bound on the expected value) is at most $\delta/(KMN)$ as well. The lemma follows by taking a union bound over all (at most N) points considered above and over all M settings of j, ■

Next we show that with high probability Algorithm APPROX-EVAL-SIMULATOR returns either UNKNOWN or a numerical value (as opposed to outputting FAIL in Line 23):

Lemma 4 *For any D, ϵ, δ and i^*, Algorithm* APPROX-EVAL-SIMULATOR *outputs* FAIL *with probability at most δ/K.*

Proof: Fix any element $i \neq i^*$. The probability (taken only over the choice of the random subset in each execution of Line 16) that i is placed in S'_j in each of the first $\log N + \log(K/\delta)$ executions of Line 16 is at most $\frac{\delta}{KN}$. Taking a union bound over all $N-1$ points $i \neq i^*$, the probability that any point other than i^* remains in S_{j-1} through all of the first $\log N + \log(K/\delta)$ executions of the outer "for" loop is at most $\frac{\delta}{K}$. Assuming that this holds, then in the execution of the outer "for" loop when $j = \log N + \log(K/\delta) + 1$, the algorithm will return $\hat{D}(S_{j-1}) = \hat{D}(i^*)$ in Line 4. ■

For the rest of the analysis it will be helpful for us to define several "desirable" events and show that they all hold with high probability:

1. Let E_1 denote the event that every set H_j that is ever constructed in Line 7 satisfies both properties (i) and (ii) stated in Lemma 3. By Lemma 3 the event E_1 holds with probability at least $1 - \delta/K$.

2. Let E_2 denote the event that in every execution of Line 9, the estimate $\hat{D}_{S_{j-1}}(i^*)$ is within an additive $\pm\frac{\kappa}{40}$ of the true value of $D(i^*)/D(S_{j-1})$. By the choice of m in Line 6 (i.e., using $m = \Omega(\log(M/\delta)/\kappa^2)$), an additive Chernoff bound, and a union bound over all iterations, the event E_2 holds with probability at least $1 - \delta/K$.

3. Let E_3 denote the event that if Line 10 is executed, the resulting value $\hat{D}_{S_{j-1}}(i^*)$ lies in $[1 - \frac{\epsilon}{2M}, 1 + \frac{\epsilon}{2M}]D(i^*)/D(S_{j-1})$. Assuming that event E_2 holds, if Line 10 is reached then the true value of $D(i^*)/D(S_{j-1})$ must be at least $\kappa/40$, and consequently a multiplicative Chernoff bound and the choice of m (i.e. using $m = \Omega(M^2 \log(M/\delta)/(\epsilon^2\kappa))$) together imply that $\hat{D}_{S_{j-1}}(i^*)$ lies in $[1 - \frac{\epsilon}{2M}, 1 + \frac{\epsilon}{2M}]D(i^*)/D(S_{j-1})$ except with failure probability at most δ/K.

4. Let E_4 denote the event that in every execution of Line 12, the estimate $\hat{D}_{S_{j-1}}(H_j)$ is within an additive error of $\pm\frac{\epsilon}{20}$ from the true value of $D(H_j)/D(S_{j-1})$. By the choice of m in Line 6 (i.e., using $m = \Omega(\log(M/\delta)/\epsilon^2)$) and an additive Chernoff bound, the event E_4 holds with probability at least $1 - \delta/K$.

The above arguments show that E_1, E_2, E_3 and E_4 all hold with probability at least $1 - 4\delta/K$.

Let E_5 denote the event that in every execution of Line 16, the set S'_j which is drawn satisfies $D(S'_j)/D(S_{j-1} \setminus (H_j \cup \{i^*\})) \geq 1/3$. The following lemma says that conditioned on E_1 through E_4 all holding, event E_5 holds with high probability:

Lemma 5 *Conditioned on E_1 through E_4 the probability that E_5 holds is at least $1 - \delta/K$.*

Proof: Fix a value of j and consider the j-th iteration of Line 16. Since events E_2 and E_4 hold, it must be the case that $D(S_{j-1} \setminus (H_j \cup \{i^*\}))/D(S_{j-1}) \geq \epsilon/40$. Since event E_1 holds, it must be the case that every $i \in (S_{j-1} \setminus (H_j \cup \{i^*\}))$ has $D(i)/D(S_{j-1}) \leq \kappa$. Now since S'_j is chosen by independently including each element of $S_{j-1} \setminus (H_j \cup \{i^*\})$ with probability $1/2$, we can apply the first part of Corollary 11 and get

$$\Pr\left[D(S'_j) < \frac{1}{3}D(S_{j-1} \setminus (H_j \cup \{i^*\}))\right] \leq \epsilon^{-4\epsilon/(40 \cdot 9 \cdot 4\kappa)} < \frac{\delta}{KM} ,$$

where the last inequality follows by the setting of $\kappa = \Theta(\epsilon/(M^2 \log(1/\delta)))$. ■

Thus we have established that E_1 through E_5 all hold with probability at least $1 - 5\delta/K$.

Next, let E_6 denote the event that the algorithm never returns UNKNOWN and exits in Line 19. Our next lemma shows that conditioned on events E_1 through E_5, the probability of E_6 is at least $1 - \delta/K$:

Lemma 6 *Conditioned on E_1 through E_5 the probability that E_6 holds is at least $1 - \delta/K$.*

Proof: Fix any iteration j of the outer "For" loop. In order for the algorithm to reach Line 18 in this iteration, it must be the case (by Lines 9 and 13) that at least $(\epsilon/10 - \kappa/20)m > (\epsilon/20)m$ points in $i_1, \ldots, i_m$ do not belong to $H_j \cup \{i^*\}$. Since each point not in H_j appears at most $\frac{3}{4}\kappa m$ times in the list $i_1, \ldots, i_m$, there must be at least $\frac{\epsilon}{15\kappa}$ distinct such values. Hence the probability that none of these values is selected to belong to S'_j is at most $1/2^{\epsilon/(15\kappa)} < \delta/(KM)$. A union bound over all (at most M) values of j gives that the probability the algorithm ever returns UNKNOWN and exits in Line 19 is at most δ/M, so the lemma is proved. ■

Now let E_7 denote the event that in every execution of Line 17, the estimate $\hat{D}_{S_{j-1}}(S_j)$ lies in $[1 - \frac{\epsilon}{2M}, 1 + \frac{\epsilon}{2M}]D(S_j)/D(S_{j-1})$. The following lemma says that conditioned on E_1 through E_5, event E_7 holds with probability at least $1 - \delta/K$:

Lemma 7 *Conditioned on E_1 through E_5, the probability that E_7 holds is at least $1 - \delta/K$.*

Proof: Fix a value of j and consider the j-th iteration of Line 16. The expected value of $\hat{D}_{S_{j-1}}(S_j)$ is precisely

$$\frac{D(S_j)}{D(S_{j-1})} = \frac{D(S_j)}{D(S_{j-1} \setminus (H_j \cup \{i^*\}))} \cdot \frac{D(S_{j-1} \setminus (H_j \cup \{i^*\}))}{D(S_{j-1})}. \tag{5}$$

Since events E_2 and E_4 hold we have that $\frac{D(S_{j-1} \setminus (H_j \cup \{i^*\}))}{D(S_{j-1})} \geq \epsilon/40$, and since event E_5 holds we have that $\frac{D(S_j)}{D(S_{j-1} \setminus (H_j \cup \{i^*\}))} \geq 1/3$ (note that $D(S_j) \geq D(S'_j)$). Thus we have that (5) is at least $\epsilon/120$. Recalling the value of m (i.e., using $m = \Omega(M^2 \log(M/\delta)/\epsilon^2\kappa) = \Omega(M^2 \log(KM/\delta)/\epsilon^3)$) a multiplicative Chernoff bound gives that indeed $\hat{D}_{S_{j-1}}(S_j) \in [1 - \frac{\epsilon}{2M}, 1 + \frac{\epsilon}{2M}]D(S_j)/D(S_{j-1})$ with failure probability at most $\delta/(KM)$. A union bound over all M possible values of j finishes the proof. ■

At this point we have established that events E_1 through E_7 all hold with probability at least $1 - 7\delta/K$.

We can now argue that each estimate $\hat{D}(S_j)$ is indeed a high-accuracy estimate of the true value $D(S_j)$:

Lemma 8 *With probability at least* $1 - 7\delta/K$ *each estimate* $\hat{D}(S_j)$ *constructed by* APPROX-EVAL-SIMULATOR *lies in* $[(1 - \frac{\epsilon}{2M})^j, (1 + \frac{\epsilon}{2M})^j]D(S_j)$.

Proof: We prove the lemma by showing that if all events E_1 through E_7 hold, then the following claim (denoted (*)) holds: each estimate $\hat{D}(S_j)$ constructed by APPROX-EVAL-SIMULATOR lies in $[(1-\frac{\epsilon}{2M})^j, (1+\frac{\epsilon}{2M})^j]D(S_j)$. Thus for the rest of the proof we assume that indeed all events E_1 through E_7 hold.

The claim (*) is clearly true for $j = 0$. We prove (*) by induction on j assuming it holds for $j - 1$. The only places in the algorithm where $\hat{D}(S_j)$ may be set are Lines 10 and 21. If $\hat{D}(S_j)$ is set in Line 21 then (*) follows from the inductive claim for $j - 1$ and Lemma 7. If $\hat{D}(S_j)$ is set in Line 10, then (*) follows from the inductive claim for $j - 1$ and the fact that event E_3 holds. This concludes the proof of the lemma. ■

Finally, we require the following crucial lemma which establishes that if $i^* \notin L_{\epsilon,N}$ (and hence the initial rank $\mathrm{rank}_{[N]}$ of i^* is at least ϵ), then with very high probability the rank of i^* never becomes too low during the execution of the algorithm:

Lemma 9 *Suppose* $i^* \notin L_{\epsilon,N}$. *Then with probability at least* $1 - \delta/K$, *every set* S_{j-1} *constructed by the algorithm has* $\mathrm{rank}_{S_{j-1}}(i^*) \geq \epsilon/2$.

We prove Lemma 9 in Section 3.3.2 below.

With these pieces in place we are ready to prove Theorem 1.

Proof of Theorem 1: It is straightforward to verify that algorithm APPROX-EVAL-SIMULATOR has the claimed query complexity. We now argue that APPROX-EVAL-SIMULATOR meets the two requirements (i) and (ii) of Definition 3. Throughout the discussion below we assume that all the "favorable events" in the above analysis (i.e. events E_1 through E_7, Lemma 4, and Lemma 9) indeed hold as desired (incurring an overall failure probability of at most δ).

Suppose first that $i^* \notin L_{\epsilon,D}$. By Lemma 9 it must be the case that the algorithm does not return UNKNOWN in Line 14. (This is because in order to reach Line 14 it would need to be the case that $D(i^*)/D(S_{j-1}) \leq 3\kappa/40$ (so the algorithm does not instead go to Line 22 in Line 10), but since by Lemma 3 every element k in H_j has $D(k)/D(S_{j-1}) \geq \kappa/2$, this means that i^* does not belong to H_j. In order to reach Line 14, by event E_4 we must have $D(H_j)/D(S_{j-1}) \geq 1 - 3\epsilon/20$. Since every element of H_j has more mass under D (at least $\kappa/2$) than i^* (which has at most $3\kappa/40$), this implies that $\mathrm{rank}_{S_{j-1}}(i^*) \leq 3\epsilon/20$. This contradicts Lemma 9.) And by Lemma 6 it must be the case that the algorithm does not return UNKNOWN in Line 19. Thus the algorithm terminates by returning an estimate $\hat{D}(S_j) = \hat{D}(i^*)$ which, by Lemma 8, lies in $[(1 - \frac{\epsilon}{2M})^j, (1 + \frac{\epsilon}{2M})^j]D(i^*)$. Since $j \leq M$ this estimate lies in $[1 - \epsilon, 1 + \epsilon]D(i^*)$ as required.

Now suppose that $i^* \in L_{\epsilon,D}$. By Lemma 4 we may assume that the algorithm either outputs UNKNOWN or a numerical value. As above, Lemma 8 implies that if the algorithm outputs a numerical value then the value lies in $[1 - \epsilon, 1 + \epsilon]D(i^*)$ as desired. This concludes the proof of Theorem 1. ■

3.3.2 Proof of Lemma 9.

The key to proving Lemma 9 will be proving the next lemma. (In the following, for S a set of real numbers we write $\mathrm{sum}(S)$ to denote $\sum_{\alpha \in S} \alpha$.)

Lemma 10 *Fix* $0 < \epsilon \leq c$. *Set* $\kappa = \Theta(\epsilon/(M^2 \log(1/\delta)))$. *Let* $T = \{\alpha_1, \ldots, \alpha_n\}$ *be a set of values* $\alpha_1 < \cdots < \alpha_n$ *such that* $\mathrm{sum}(T) = 1$. *Fix* $\ell \in [n]$ *and let* $T_L = \{\alpha_1, \ldots, \alpha_\ell\}$ *and let* $T_R = \{\alpha_{\ell+1}, \ldots, \alpha_n\}$, *so* $T_L \cup T_R = T$. *Assume that* $\mathrm{sum}(T_L) \geq \epsilon/2$ *and that* $\alpha_\ell \leq \kappa/10$.

Fix H *to be any subset of* T *satisfying the following two properties: (i)* H *includes every* α_j *such that* $\alpha_j \geq \kappa$; *and (ii)* H *includes no* α_j *such that* $\alpha_j < \kappa/2$. *(Note that consequently* H *does not intersect* T_L.*)*

Let T' be a subset of $(T \setminus (H \cup \{\alpha_\ell\})$ selected uniformly at random. Let $T'_L = T' \cap T_L$ and let $T'_R = T' \cap T_R$.

Then we have the following:

1. *If $\mathrm{sum}(T_L) > 20\epsilon$, then with probability at least $1 - \delta/M$ (over the random choice of T') it holds that*

$$\frac{\mathrm{sum}(T'_L \cup \{\alpha_\ell\})}{\mathrm{sum}(T' \cup \{\alpha_\ell\})} \geq 9\epsilon;$$

2. *If $\epsilon/2 \leq \mathrm{sum}(T_L) < 20\epsilon$, then with probability at least $1 - \delta/M$ (over the random choice of T') it holds that*

$$\frac{\mathrm{sum}(T'_L \cup \{\alpha_\ell\})}{\mathrm{sum}(T' \cup \{\alpha_\ell\})} \geq \mathrm{sum}(T_L)\,(1-\rho)\,,$$

where $\rho = \frac{\ln 2}{M}$.

Proof of Lemma 9 using Lemma 10: We apply Lemma 10 repeatedly at each iteration j of the outer "For" loop. The set H of Lemma 10 corresponds to the set H_j of "heavy" elements that are removed at a given iteration, the set of values T corresponds to the values $D(i)/D(S_{j-1})$ for $i \in S_{j-1}$, and the element α_ℓ of Lemma 10 corresponds to $D(i^*)/D(S_{j-1})$. The value $\mathrm{sum}(T_L)$ corresponds to $\mathrm{rank}_{S_{j-1}}(i^*)$ and the value

$$\frac{\mathrm{sum}(T'_L \cup \{\alpha_\ell\})}{\mathrm{sum}(T' \cup \{\alpha_\ell\})}$$

corresponds to $\mathrm{rank}_{S_j}(i^*)$. Observe that since $i^* \notin L_{\epsilon,N}$ we know that initially $\mathrm{rank}_{[N]}(i^*) \geq \epsilon$, which means that the first time we apply Lemma 10 (with $T = \{D(i) : i \in [N]\}$) we have $\mathrm{sum}(T_L) \geq \epsilon$.

By Lemma 10 the probability of failure in any of the (at most M) iterations is at most δ/K, so we assume that there is never a failure. Consequently for all j we have that if $\mathrm{rank}_{S_{j-1}}(i^*) \geq 20\epsilon$ then $\mathrm{rank}_{S_j}(i^*) \geq 9\epsilon$, and if $\epsilon/2 \leq \mathrm{rank}_{S_{j-1}}(i^*) < 20\epsilon$ then $\mathrm{rank}_{S_j}(i^*) \geq \mathrm{rank}_{S_j}(i^*) \cdot (1-\rho)$. Since $\mathrm{rank}_{S_0}(i^*) \geq \epsilon$, it follows that for all $j \leq M$ we have $\mathrm{rank}_{S_j}(i^*) \geq \epsilon \cdot (1-\rho)^M > \epsilon/2$. ■

Proof of Lemma 10. We begin with the following claim:

Claim 11 *With probability at least $1 - \delta/(2M)$ (over the random choice of T') it holds that $\mathrm{sum}(T'_L) \geq \frac{1}{2} \cdot \mathrm{sum}(T_L) \cdot (1 - \rho/2)$.*

Proof: Recall from the setup that every element $\alpha_i \in T_L$ satisfies $\alpha_i \leq \kappa/10$, and $\mathrm{sum}(T_L) \geq \epsilon/2$. Also recall that $\kappa = \Theta(\epsilon/(M^2 \log(1/\delta)))$ and that $\rho = \frac{\ln 2}{M}$, so that $\rho^2\epsilon/(6\kappa) \geq \ln(2M/\delta)$. The claim follows by applying the first part of Corollary 11 (with $\gamma = \rho/2$). ■

Part (1) of Lemma 10 is an immediate consequence of Claim 11, since in part (1) we have

$$\frac{\mathrm{sum}(T'_L \cup \{\alpha_\ell\})}{\mathrm{sum}(T' \cup \{\alpha_\ell\})} \geq \mathrm{sum}(T'_L) \geq \frac{1}{2} \cdot \mathrm{sum}(T_L) \cdot \left(1 - \frac{\rho}{2}\right) \geq \frac{1}{2} \cdot 20\epsilon \cdot \left(1 - \frac{\rho}{2}\right) \geq 9\epsilon.$$

It remains to prove Part (2) of the lemma. We will do this using the following claim:

Claim 12 *Suppose $\epsilon/2 \leq \mathrm{sum}(T_L) \leq 20\epsilon$. Then with probability at least $1 - \delta/(2M)$ (over the random choice of T') it holds that $\mathrm{sum}(T'_R) \leq \frac{1}{2}\mathrm{sum}(T_R) \cdot (1 + \rho/2)$.*

Proof: Observe first that that $\alpha_i < \kappa$ for each $\alpha_i \in T_R \setminus H$. We consider two cases.

If $\mathrm{sum}(T_R \setminus H) \geq 4\epsilon$, then we apply the first part of Corollary 11 to the α_i's in $T_R \setminus H$ and get that

$$\begin{aligned}
\Pr\left[\mathrm{sum}(T'_R) > \frac{1}{2}\mathrm{sum}(T_R)\cdot(1+\rho/2)\right] &\leq \Pr\left[\mathrm{sum}(T'_R) > \frac{1}{2}\mathrm{sum}(T_R \setminus H))\cdot(1+\rho/2)\right] \\
&< \exp(-\rho^2\mathrm{sum}(T_R \setminus H)/24\kappa) \qquad (6)\\
&\leq \exp(-\rho^2\epsilon/(6\kappa)) \leq \frac{\delta}{(2M)} \qquad (7)
\end{aligned}$$

(recall from the proof of Claim 11 that $\rho^2\epsilon/(6\kappa) \geq \ln(2M/\delta)$).

If $\mathrm{sum}(T_R \setminus H) < 4\epsilon$, (so that the expected value of $\mathrm{sum}(T'_R)$ is less than 2ϵ) then we can apply the second part of Corollary 11 as we explain next. Observe that by the premise of the lemma, $\mathrm{sum}(T_R) \geq 1 - 20\epsilon$ which is at least $1/2$ (recalling that ϵ is at most a small absolute constant c). Consequently, the event "$\mathrm{sum}(T'_R) \geq \frac{1}{2}\cdot\mathrm{sum}(T_R)\cdot(1+\rho/2)$" implies the event "$\mathrm{sum}(T'_R) \geq \frac{1}{4}$", and by applying the second part of Corollary 11 we get

$$\Pr\left[\mathrm{sum}(T'_R) > \frac{1}{2}\mathrm{sum}(T_R)\cdot(1+\rho/2)\right] \leq \Pr\left[\mathrm{sum}(T'_R) > \frac{1}{4}\right] < 2^{-1/4\kappa} < \frac{\delta}{2}\,, \qquad (8)$$

as required. ■

Now we can prove Lemma 10. Using Claims 11 and 12 we have that with probability at least $1 - \delta/M$,

$$\mathrm{sum}(T'_L) \geq \frac{1}{2}\cdot\mathrm{sum}(T_L)\cdot(1-\rho/2) \qquad \text{and} \qquad \mathrm{sum}(T'_R) \leq \frac{1}{2}\mathrm{sum}(T_R)\cdot(1+\rho/2);$$

we assume that both these inequalities hold going forth. Since

$$\frac{\mathrm{sum}(T'_L \cup \{\alpha_\ell\})}{\mathrm{sum}(T' \cup \{\alpha_\ell\})} = \frac{\mathrm{sum}(T'_L) + \alpha_\ell}{\mathrm{sum}(T') + \alpha_\ell} > \frac{\mathrm{sum}(T'_L)}{\mathrm{sum}(T')},$$

it is sufficient to show that $\frac{\mathrm{sum}(T'_L)}{\mathrm{sum}(T')} \geq \mathrm{sum}(T_L)(1-\rho)$; we now show this. As $\mathrm{sum}(T') = \mathrm{sum}(T'_L) + \mathrm{sum}(T'_R)$,

$$\begin{aligned}
\frac{\mathrm{sum}(T'_L)}{\mathrm{sum}(T')} &= \frac{\mathrm{sum}(T'_L)}{\mathrm{sum}(T'_L) + \mathrm{sum}(T'_R)} = \frac{1}{1 + \frac{\mathrm{sum}(T'_R)}{\mathrm{sum}(T'_L)}} \\
&\geq \frac{1}{1 + \frac{(1/2)\cdot\mathrm{sum}(T_R)\cdot(1+\rho/2)}{(1/2)\cdot\mathrm{sum}(T_L)\cdot(1-\rho/2)}} \\
&= \frac{\mathrm{sum}(T_L)\cdot(1-\rho/2)}{\mathrm{sum}(T_L)\cdot(1-\rho/2) + \mathrm{sum}(T_R)\cdot(1+\rho/2)} \\
&\geq \frac{\mathrm{sum}(T_L)\cdot(1-\rho/2)}{\mathrm{sum}(T_L)\cdot(1+\rho/2) + \mathrm{sum}(T_R)\cdot(1+\rho/2)} \\
&= \mathrm{sum}(T_L)\cdot\frac{1-\rho/2}{1+\rho/2} > \mathrm{sum}(T_L)\cdot(1-\rho).
\end{aligned}$$

This concludes the proof of Lemma 10. ■

4 Testing equivalence to a known distribution D^*

4.1 A $\mathrm{poly}(1/\epsilon)$-query COND_D algorithm

In this subsection we present an algorithm COND-TEST-KNOWN and prove the following theorem:

Theorem 2 COND-TEST-KNOWN *is a $\tilde{O}(1/\epsilon^4)$-query COND_D testing algorithm for testing equivalence to a known distribution D^*. That is, for every pair of distributions D, D^* over $[N]$ (such that D^* is fully specified and there is* COND *query access to D), the algorithm outputs* ACCEPT *with probability at least $2/3$ if $D = D^*$ and outputs* REJECT *with probability at least $2/3$ if $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$.*

High-level overview of the algorithm and its analysis: First, we note that by reordering elements of $[N]$ we may assume without loss of generality that $D^*(1) \leq \cdots \leq D^*(N)$; this will be convenient for us.

As we shall see in later subsections, our $(\log N)^{\Omega(1)}$ query lower bound for $\mathsf{PCOND_D}$ algorithms exploits the intuition that comparing two points using the $\mathsf{PCOND_D}$ oracle might not provide much information (e.g. if one of the two points was a priori "known" to be much heavier than the other). In contrast, with a general COND_D oracle at our disposal, we can compare a given point $j \in [N]$ with *any subset* of $[N] \setminus \{j\}$. Thus the following definition will be useful:

Definition 4 (Comparable points) *Fix $0 < \lambda \leq 1$. A point $j \in \mathrm{supp}(D^*)$ is said to be* λ-comparable *if there exists a set $S \subseteq ([N] \setminus \{j\})$ such that*

$$D^*(j) \in [\lambda D^*(S), D^*(S)/\lambda].$$

Such a set S is then said to be a λ-comparable-witness *for j (according to D^*), which is denoted $S \cong^* j$. We say that a set $T \subseteq [N]$ is λ-comparable if every $i \in T$ is λ-comparable.*

We stress that the notion of being λ-comparable deals only with the known distribution D^*; this will be important later.

Fix $\epsilon_1 = \Theta(\epsilon)$ (we specify ϵ_1 precisely in Equation 11 below). Our analysis and algorithm consider two possible cases for the distribution D^* (where it is not hard to verify, and we provide an explanation subsequently, that one of the two cases must hold):

1. The first case is that for some $i^* \in [N]$ we have

$$D^*(\{1, \ldots, i^*\}) > 2\epsilon_1 \quad \text{but} \quad D^*(\{1, \ldots, i^* - 1\}) \leq \epsilon_1. \tag{9}$$

   In this case $1 - \epsilon_1$ of the total probability mass of D^* must lie on a set of at most $1/\epsilon_1$ elements, and in such a situation it is easy to efficiently test whether $D = D^*$ using $\mathrm{poly}(1/\epsilon)$ queries (see Algorithm COND_D-TEST-KNOWN-HEAVY and Lemma 15).

2. The second case is that there exists an element $k^* \in [N]$ such that

$$\epsilon_1 < D^*(\{1, \ldots, k^*\}) \leq 2\epsilon_1 < D^*(\{1, \ldots, k^* + 1\}). \tag{10}$$

   This is the more challenging (and typical) case. In this case, it can be shown that every element $j > k^*$ has at least one ϵ_1-comparable-witness within $\{1, \ldots, j\}$. In fact, we show (see Claim 13) that either (a) $\{1, \ldots, j-1\}$ is an ϵ_1-comparable witness for j, or (b) the set $\{1, \ldots, j-1\}$ can be partitioned into disjoint sets[9] $S_1, \ldots, S_t$ such that *each* S_i, $1 \leq i \leq t$, is a $\frac{1}{2}$-comparable-witness for j. Case (a) is relatively easy to handle so we focus on (b) in our informal description below.

[9] In fact the sets are intervals (under the assumption $D^*(1) \leq \cdots \leq D^*(n)$), but that is not really important for our arguments.

The partition $S_1, \ldots, S_t$ is useful to us for the following reason: Suppose that $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$. It is not difficult to show (see Claim 14) that unless $D(\{1, \ldots, k^*\}) > 3\epsilon_1$ (which can be easily detected and provides evidence that the tester should reject), a random sample of $\Theta(1/\epsilon)$ draws from D will with high probability contain a "heavy" point $j > k^*$, that is, a point $j > k^*$ such that $D(j) \geq (1+\epsilon_2)D^*(j)$ (where $\epsilon_2 = \Theta(\epsilon)$). Given such a point j, there are two possibilities:

1. The first possibility is that a significant fraction of the sets $S_1, \ldots, S_t$ have $D(j)/D(S_i)$ "noticeably different" from $D^*(j)/D^*(S_i)$. (Observe that since each set S_i is a $\frac{1}{2}$-comparable witness for j, it is possible to efficiently check whether this is the case.) If this is the case then our tester should reject since this is evidence that $D \neq D^*$.

2. The second possibility is that almost every S_i has $D(j)/D(S_i)$ very close to $D^*(j)/D^*(S_i)$. If this is the case, though, then since $D(j) \geq (1+\epsilon_2)D^*(j)$ and the union of $S_1, \ldots, S_t$ is $\{1, \ldots, j-1\}$, it must be the case that $D(\{1, \ldots, j\})$ is "significantly larger" than $D^*(\{1, \ldots, j\})$. This will be revealed by random sampling from D and thus our testing algorithm can reject in this case as well.

Key quantities and useful claims. We define some quantities that are used in the algorithm and its analysis. Let

$$\epsilon_1 \stackrel{\text{def}}{=} \frac{\epsilon}{10}; \quad \epsilon_2 \stackrel{\text{def}}{=} \frac{\epsilon}{2}; \quad \epsilon_3 \stackrel{\text{def}}{=} \frac{\epsilon}{48}; \quad \epsilon_4 \stackrel{\text{def}}{=} \frac{\epsilon}{6}. \tag{11}$$

Claim 13 *Suppose there exists an element $k^* \in [N]$ that satisfies Equation (10). Fix any $j > k^*$. Then*

1. *If $D^*(j) \geq \epsilon_1$, then $S_1 \stackrel{\text{def}}{=} \{1, \ldots, j-1\}$ is an ϵ_1-comparable witness for j;*

2. *If $D^*(j) < \epsilon_1$ then the set $\{1, \ldots, j-1\}$ can be partitioned into disjoint sets $S_1, \ldots, S_t$ such that each S_i, $1 \leq i \leq t$, is a $\frac{1}{2}$-comparable-witness for j.*

Proof: First consider the case that $D^*(j) \geq \epsilon_1$. In this case $S_1 = \{1, \ldots, j-1\}$ is an ϵ_1-comparable witness for j because $D^*(j) \geq \epsilon_1 \geq \epsilon_1 D^*(\{1, \ldots, j-1\})$ and $D^*(j) \leq 1 \leq \frac{1}{\epsilon_1} D^*(\{1, \ldots, k^*\}) \leq \frac{1}{\epsilon_1} D^*(\{1, \ldots, j-1\})$, where the last inequality holds since $k^* \leq j-1$.

Next, consider the case that $D^*(j) < \epsilon_1$. In this case we build our intervals iteratively from right to left, as follows. Let $j_1 = j-1$ and let j_2 be the minimum index in $\{0, \ldots, j_1 - 1\}$ such that

$$D^*(\{j_2+1, \ldots, j_1\}) \leq D^*(j).$$

(Observe that we must have $j_2 \geq 1$, because $D^*(\{1, \ldots, k^*\}) > \epsilon_1 > D^*(j)$.) Since $D^*(\{j_2, \ldots, j_1\}) > D^*(j)$ and the function $D^*(\cdot)$ is monotonically increasing, it must be the case that

$$\frac{1}{2} D^*(j) \leq D^*(\{j_2+1, \ldots, j_1\}) \leq D^*(j).$$

Thus the interval $S_1 \stackrel{\text{def}}{=} \{j_2+1, \ldots, j_1\}$ is a $\frac{1}{2}$-comparable witness for j as desired.

We continue in this fashion from right to left; i.e. if we have defined $j_2, \ldots, j_t$ as above and there is an index $j' \in \{0, \ldots, j_t - 1\}$ such that $D^*(\{j'+1, \ldots, j_t\}) > D^*(j)$, then we define j_{t+1} to be the minimum index in $\{0, \ldots, j_t - 1\}$ such that

$$D^*(\{j_{t+1}+1, \ldots, j_t\}) \leq D^*(j),$$

and we define S_t to be the interval $\{j_{t+1}+1,\ldots,j_t\}$. The argument of the previous paragraph tells us that

$$\frac{1}{2}D^*(j) \le D^*(\{j_{t+1}+1,\ldots,j_t\}) \le D^*(j) \tag{12}$$

and hence S_t is an $\frac{1}{2}$-comparable witness for j.

At some point, after intervals $S_1 = \{j_2+1,\ldots,j_1\},\ldots,S_t = \{j_{t+1}+1,\ldots,j_t\}$ have been defined in this way, it will be the case that there is no index $j' \in \{0,\ldots,j_t-1\}$ such that $D^*(\{j'+1,\ldots,j_t\}) > D^*(j)$. At this point there are two possibilities: first, if $j_{t+1}+1 = 1$, then $S_1,\ldots,S_t$ give the desired partition of $\{1,\ldots,j-1\}$. If $j_{t+1}+1 > 1$ then it must be the case that $D^*(\{1,\ldots,j_{t+1}\}) \le D^*(j)$. In this case we simply add the elements $\{1,\ldots,j_{t+1}\}$ to S_t, i.e. we redefine S_t to be $\{1,\ldots,j_t\}$. By Equation (12) we have that

$$\frac{1}{2}D^*(j) \le D^*(S_t) \le 2D^*(j)$$

and thus S_t is an $\frac{1}{2}$-comparable witness for j as desired. This concludes the proof. ■

Definition 5 (Heavy points) *A point $j \in \mathrm{supp}(D^*)$ is said to be* η-heavy *if $D(j) \ge (1+\eta)D^*(j)$.*

Claim 14 *Suppose that $d_{\mathrm{TV}}(D,D^*) \ge \epsilon$ and Equation (10) holds. Suppose moreover that $D(\{1,\ldots,k^*\}) \le 4\epsilon_1$. Let $i_1,\ldots,i_\ell$ be i.i.d. points drawn from D. Then for $\ell = \Theta(1/\epsilon)$, with probability at least $99/100$ (over the i.i.d. draws of $i_1,\ldots,i_\ell \sim D$) there is some point $i_j \in \{i_1,\ldots,i_\ell\}$ such that $i_j > k^*$ and i_j is ϵ_2-heavy.*

Proof: Define H_1 to be the set of all ϵ_2-heavy points and H_2 to be the set of all "slightly lighter" points as follows:

$$H_1 = \{\, i \in [N] \mid D(i) \ge (1+\epsilon_2)D^*(i) \,\}$$
$$H_2 = \{\, i \in [N] \mid (1+\epsilon_2)D^*(i) > D(i) \ge D^*(i) \,\}$$

By definition of the total variation distance, we have

$$\begin{aligned}
\epsilon \le d_{\mathrm{TV}}(D,D^*) &= \sum_{i:D(i)\ge D^*(i)} (D(i)-D^*(i)) = (D(H_1)-D^*(H_1)) + (D(H_2)-D^*(H_2)) \\
&\le D(H_1) + ((1+\epsilon_2)D^*(H_2) - D^*(H_2)) \\
&= D(H_1) + \epsilon_2 D^*(H_2) < D(H_1) + \epsilon_2 = D(H_1) + \frac{\epsilon}{2}.
\end{aligned}$$

So it must be the case that $D(H_1) \ge \epsilon/2 = 5\epsilon_1$. Since by assumption we have $D(\{1,\ldots,k^*\}) \le 4\epsilon_1$, it must be the case that $D(H_1 \setminus \{1,\ldots,k^*\}) \ge \epsilon_1$. The claim follows from the definition of H_1 and the size, ℓ, of the sample. ■

4.1.1 Proof of Theorem 2

It is straightforward to verify that the query complexity of COND_D-Test-Known-Heavy is $\tilde{O}(1/\epsilon^4)$ and the query complexity of COND_D-Test-Known-Main is also $\tilde{O}(1/\epsilon^4)$, so the overall query complexity of COND-TEST-KNOWN is as claimed.

By the definition of i^* (in the first line of the algorithm), either Equation (9) holds for this setting of i^*, or Equation (10) holds for $k^* = i^* - 1$. To prove correctness of the algorithm, we first deal with the simpler case, which is that Equation (9) holds:

Algorithm 4: COND_D-TEST-KNOWN

Input: error parameter $\epsilon > 0$; query access to COND_D oracle; explicit description $(D^*(1), \ldots, D^*(N))$ of distribution D^* satisfying $D^*(1) \leq \cdots \leq D^*(N)$

Let i^* be the minimum index $i \in [N]$ such that $D^*(\{1, \ldots, i\}) > 2\epsilon_1$.
if $D^*(\{1, \ldots, i^* - 1\}) \leq \epsilon_1$ **then**
    Call algorithm COND_D-Test-Known-Heavy$(\epsilon, \mathsf{COND}_D, D^*, i^*)$ (and exit)
else
    Call algorithm COND_D-Test-Known-Main$(\epsilon, \mathsf{COND}_D, D^*, i^* - 1)$ (and exit).
end if

Algorithm 5: COND_D-TEST-KNOWN-HEAVY

Input: error parameter $\epsilon > 0$; query access to COND_D oracle; explicit description $(D^*(1), \ldots, D^*(N))$ of distribution D^* satisfying $D^*(1) \leq \cdots \leq D^*(N)$; value $i^* \in [N]$ satisfying $D^*(\{1, \ldots, i^* - 1\}) \leq \epsilon_1$, $D^*(\{1, \ldots, i^*\}) > 2\epsilon_1$

1: Call the SAMP_D oracle $m = \Theta((\log(1/\epsilon))/\epsilon^4)$ times. For each $i \in \{i^*, \ldots, N\}$ let $\widehat{D}(j)$ be the fraction of the m calls to SAMP_D that returned i. Let $\widehat{D}' = 1 - \sum_{i \in \{i^*, \ldots, N\}} \widehat{D}(i)$ be the fraction of the m calls that returned values in $\{1, \ldots, i^* - 1\}$.
2: **if** either (any $i \in \{i^*, \ldots, N\}$ has $|\widehat{D}(i) - D^*(i)| > {\epsilon_1}^2$) or ($\widehat{D}' - D^*(\{1, \ldots, i^* - 1\}) > \epsilon_1$) **then**
3:    output REJECT (and exit)
4: **end if**
5: Output ACCEPT

Lemma 15 *Suppose that D^* is such that $D^*(\{1, \ldots, i^*\}) > 2\epsilon_1$ but $D^*(\{1, \ldots, i^* - 1\}) \leq \epsilon_1$. Then* COND_D-TEST-KNOWN-HEAVY$(\epsilon, \mathsf{COND}_D, D^*, i^*)$ *returns* ACCEPT *with probability at least $2/3$ if $D = D^*$ and returns* REJECT *with probability at least $2/3$ if $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$.*

Proof: The conditions of Lemma 15, together with the fact that $D^*(\cdot)$ is monotone non-decreasing, imply that each $i \geq i^*$ has $D^*(i) \geq \epsilon_1$. Thus there can be at most $1/\epsilon_1$ many values $i \in \{i^*, \ldots, N\}$, i.e. it must be the case that $i^* \geq N - 1/\epsilon_1 + 1$. Since the expected value of $\widehat{D}(i)$ (defined in Line 1 of COND_D-TEST-KNOWN-HEAVY) is precisely $D(i)$, for any fixed value of $i \in \{i^*, \ldots, n\}$ an additive Chernoff bound implies that $|D(i) - \widehat{D}(i)| \leq (\epsilon_1)^2$ with failure probability at most $\frac{1}{10\left(1+\frac{1}{\epsilon_1}\right)}$. Similarly $|\widehat{D}' - D(\{1, \ldots, i^* - 1\})| \leq \epsilon_1$ with failure probability at most $\frac{1}{10\left(1+\frac{1}{\epsilon_1}\right)}$. A union bound over all failure events gives that with probability at least $9/10$ each value $i \in \{i^*, \ldots, N\}$ has $|D(i) - \widehat{D}(i)| \leq {\epsilon_1}^2$ and additionally $|\widehat{D}' - D(\{1, \ldots, i^* - 1\})| \leq \epsilon_1$; we refer to this compound event as (*).

If $D^* = D$, by (*) the algorithm outputs ACCEPT with probability at least $9/10$.

Algorithm 6: COND_D-TEST-KNOWN-MAIN

Input: error parameter $\epsilon > 0$; query access to COND_D oracle; explicit description $(D^*(1), \ldots, D^*(N))$ of distribution D^* satisfying $D^*(1) \leq \cdots \leq D^*(N)$; value $k^* \in [N]$ satisfying $\epsilon_1 < D^*(\{1, \ldots, k^*\}) \leq 2\epsilon_1 < D^*(\{1, \ldots, k^* + 1\})$

Call the SAMP_D oracle $\Theta(1/\epsilon^2)$ times and let $\widehat{D}(\{1, \ldots, k^*\})$ denote the fraction of responses that lie in $\{1, \ldots, k^*\}$. If $\widehat{D}(\{1, \ldots, k^*\}) \notin [\frac{\epsilon_1}{2}, \frac{5\epsilon_1}{2}]$ then output REJECT (and exit).

Call the SAMP_D oracle $\ell = \Theta(1/\epsilon)$ times to obtain points $i_1, \ldots, i_\ell$.

for all $j \in \{1, \ldots, \ell\}$ such that $i_j > k^*$ **do**

  Call the SAMP_D oracle $m = \Theta(\log(1/\epsilon)/\epsilon)$ times and let $\widehat{D}(\{1, \ldots, i_j\})$ be the fraction of responses that lie in $\{1, \ldots, i_j\}$. If $\widehat{D}(\{1, \ldots, i_j\}) \notin [1 - \epsilon_3, 1 + \epsilon_3] D^*(\{1, \ldots, i_j\})$ then output REJECT (and exit).

  if $D^*(i_j) \geq \epsilon_1$ **then**

    Run $\textsc{Compare}(\{i_j\}, \{1, \ldots, i_j - 1\}, \frac{\epsilon_2}{16}, \frac{2}{\epsilon_1}, \frac{1}{10\ell})$ and let v denote its output. If $v \notin [1 - \frac{\epsilon_2}{8}, 1 + \frac{\epsilon_2}{8}] \frac{D^*(\{1, \ldots, i_j - 1\})}{D^*(\{i_j\})}$ then output REJECT (and exit).

  else

    Let $S_1, \ldots, S_t$ be the partition of $\{1, \ldots, i_j - 1\}$ such that each S_i is an ϵ_1-comparable witness for i_j, which is provided by Claim 13.

    Select a list of $h = \Theta(1/\epsilon)$ elements $S_{a_1}, \ldots, S_{a_h}$ independently and uniformly from $\{S_1, \ldots, S_j\}$.

    For each S_{a_r}, $1 \leq r \leq h$, run $\textsc{Compare}(\{i_j\}, S_{a_r}, \frac{\epsilon_4}{8}, 4, \frac{1}{10\ell h})$ and let v denote its output. If $v \notin [1 - \frac{\epsilon_4}{4}, 1 + \frac{\epsilon_4}{4}] \frac{D^*(S_{a_r})}{D^*(\{i_j\})}$ then output REJECT (and exit).

  end if

end for

Output ACCEPT.

Now suppose that $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$. With probability at least 9/10 we have (*) so we suppose that indeed (*) holds. In this case we have

$$\begin{aligned}
\epsilon \leq d_{\mathrm{TV}}(D, D^*) &= \sum_{i<i^*} |D(i) - D^*(i)| + \sum_{i \geq i^*} |D(i) - D^*(i)| \\
&\leq \sum_{i<i^*} (D(i) + D^*(i)) + \sum_{i \geq i^*} |D(i) - D^*(i)| \\
&\leq D(\{1, \ldots, i^* - 1\}) + \epsilon_1 + \sum_{i \geq i^*} \left(|\widehat{D}(i) - D^*(i)| + {\epsilon_1}^2 \right) \\
&\leq \widehat{D}' + \epsilon_1 + 2\epsilon_1 + \sum_{i \geq i^*} \left(|\widehat{D}(i) - D^*(i)| \right)
\end{aligned}$$

where the first inequality is by the triangle inequality, the second is by (*) and the fact that $D^*(\{1, \ldots, i^* - 1\}) \leq \epsilon_1$, and the third inequality is by (*) and the fact that there are at most $1/\epsilon_1$ elements in $\{i^*, \ldots, N\}$. Since $\epsilon_1 = \epsilon/10$, the above inequality implies that

$$\frac{7}{10}\epsilon \leq \widehat{D}' + \sum_{i \geq i^*} \left(|\widehat{D}(i) - D^*(i)| \right).$$

If any $i \in \{i^*, \ldots, N\}$ has $|\widehat{D}(i) - D^*(i)| > (\epsilon_1)^2$ then the algorithm outputs REJECT so we may assume that $|\widehat{D}(i) - D^*(i)| \leq {\epsilon_1}^2$ for all i. This implies that

$$6\epsilon_1 = \frac{6}{10}\epsilon \leq \widehat{D}'$$

but since $D^*(\{1, \ldots, i^* - 1\}) \leq \epsilon_1$ the algorithm must REJECT. ■

Now we turn to the more difficult (and typical) case, that Equation (10) holds (for $k^* = i^* - 1$), i.e.

$$\epsilon_1 < D^*(\{1, \ldots, k^*\}) \leq 2\epsilon_1 < D^*(\{1, \ldots, k^* + 1\}).$$

With the claims we have already established it is straightforward to argue completeness:

Lemma 16 *Suppose that $D = D^*$ and Equation (10) holds. Then with probability at least $2/3$ algorithm* COND_D-TEST-KNOWN-MAIN *outputs* ACCEPT.

Proof: We first observe that the expected value of the quantity $\widehat{D}(\{1, \ldots, k^*\})$ defined in Line 1 is precisely $D(\{1, \ldots, k^*\}) = D^*(\{1, \ldots, k^*\})$ and hence lies in $[\epsilon_1, 2\epsilon_1]$ by Equation (10). The additive Chernoff bound implies that the probability the algorithm outputs REJECT in Line 1 is at most $1/10$. Thus we may assume the algorithm continues to Line 2.

In any given execution of Line 4, since the expected value of $\widehat{D}(\{1, \ldots, i_j\})$ is precisely $D(\{1, \ldots, i_j\}) = D^*(\{1, \ldots, i_j\}) > \epsilon_1$, a multiplicative Chernoff bound gives that the algorithm outputs REJECT with probability at most $1/(10\ell)$. Thus the probability that the algorithm outputs REJECT in any execution of Line 4 is at most $1/10$. We henceforth assume that the algorithm never outputs REJECT in this step.

Fix a setting of $j \in \{1, \ldots, \ell\}$ such that $i_j > k^*$. Consider first the case that $D^*(i_j) \geq \epsilon_1$ so the algorithm enters Line 6. By item (1) of Claim 13 and item (1) of Lemma 1, we have that with probability at least

$1 - \frac{1}{10\ell}$ COMPARE outputs a value v in the range $[1 - \frac{\epsilon_2}{16}, 1 + \frac{\epsilon_2}{16}]\frac{D^*(\{1,\ldots,i_j-1\})}{D^*(\{i_j\})}$ (recall that $D = D^*$), so the algorithm does not output REJECT in Line 6. Now suppose that $D^*(i_j) < \epsilon_1$ so the algorithm enters Line 8. Fix a value $1 \le r \le h$ in Line 10. By Claim 13 we have that S_{a_r} is a $\frac{1}{2}$-comparable witness for i_j. By item (1) of Lemma 1, we have that with probability at least $1 - \frac{1}{10\ell h}$ COMPARE outputs a value v in the range $[1 - \frac{\epsilon_4}{4}, 1 + \frac{\epsilon_4}{4}]\frac{D^*(S_{a_r})}{D^*(\{i_j\})}$ (recall that $D = D^*$). A union bound over all h values of r gives that the algorithm outputs REJECT in Line 10 with probability at most $1/(10\ell)$. So in either case, for this setting of j, the algorithm outputs REJECT on that iteration of the outer loop with probability at most $1/(10\ell)$. A union bound over all ℓ iterations of the outer loop gives that the algorithm outputs REJECT at any execution of Line 6 or Line 10 is at most $1/10$.

Thus the overall probability that the algorithm outputs REJECT is at most $3/10$, and the lemma is proved. ■

Next we argue soundness:

Lemma 17 *Suppose that $d_{\mathrm{TV}}(D, D^*) \ge \epsilon$ and Equation (10) holds. Then with probability at least $2/3$ algorithm* COND_D-TEST-KNOWN-MAIN *outputs* REJECT.

Proof: If $D(\{1,\ldots,k^*\}) \notin [\epsilon_1, 3\epsilon_1]$ then a standard additive Chernoff bound implies that the algorithm outputs REJECT in Line 1 with probability at least $9/10$. Thus we may assume going forward in the argument that $D(\{1,\ldots,k^*\}) \in [\epsilon_1, 3\epsilon_1]$. As a result we may apply Claim 14, and we have that with probability at least $99/100$ there is an element $i_j \in \{i_1,\ldots,i_\ell\}$ such that $i_j > k^*$ and i_j is ϵ_2-heavy, i.e. $D(i_j) \ge (1+\epsilon_2)D^*(i_j)$. We condition on this event going forward (the rest of our analysis will deal with this specific element i_j).

We now consider two cases:

Case 1: Distribution D has $D(\{1,\ldots,i_j\}) \notin [1-3\epsilon_3, 1+3\epsilon_3]D^*(\{1,\ldots,i_j\})$. Since the quantity $\widehat{D}(\{1,\ldots,i_j\})$ obtained in Line 4 has expected value $D(\{1,\ldots,i_j\}) \ge D(\{1,\ldots,k^*\}) \ge \epsilon_1$, applying the multiplicative Chernoff bound implies that $\widehat{D}(\{1,\ldots,i_j\}) \in [1-\epsilon_3, 1+\epsilon_3]D(\{1,\ldots,i_j\})$ except with failure probability at most $\epsilon/10 \le 1/10$. If this failure event does not occur then since $D(\{1,\ldots,i_j\}) \notin [1-3\epsilon_3, 1+3\epsilon_3]D^*(\{1,\ldots,i_j\})$ it must hold that $\widehat{D}(\{1,\ldots,i_j\}) \notin [1-\epsilon_3, 1+\epsilon_3]D^*(\{1,\ldots,i_j\})$ and consequently the algorithm outputs REJECT. Thus in Case 1 the algorithm outputs REJECT with overall failure probability at least $89/100$.

Case 2: Distribution D has $D(\{1,\ldots,i_j\}) \in [1-3\epsilon_3, 1+3\epsilon_3]D^*(\{1,\ldots,i_j\})$. This case is divided into two sub-cases depending on the value of $D^*(i_j)$.

Case 2(a): $D^*(i_j) \ge \epsilon_1$. In this case the algorithm reaches Line 6. We use the following claim:

Claim 18 *In Case 2(a), suppose that $i_j > k^*$ is such that $D(i_j) \ge (1+\epsilon_2)D^*(i_j)$, and $D(\{1,\ldots,i_j\}) \in [1-3\epsilon_3, 1+3\epsilon_3]D^*(\{1,\ldots,i_j\})$. Then*

$$\frac{D(\{1,\ldots,i_j-1\})}{D(i_j)} \le \Big(1 - \frac{\epsilon_2}{4}\Big) \cdot \frac{D^*(\{1,\ldots,i_j-1\})}{D^*(i_j)}.$$

Proof: To simplify notation we write

$$a \stackrel{\text{def}}{=} D(i_j); \quad b \stackrel{\text{def}}{=} D^*(i_j); \quad c \stackrel{\text{def}}{=} D(\{1,\ldots,i_j-1\}); \quad d \stackrel{\text{def}}{=} D^*(\{1,\ldots,i_j-1\}).$$

We have that

$$a \ge (1+\epsilon_2)b \quad \text{and} \quad a + c \le (1+3\epsilon_3)(b+d). \tag{13}$$

This gives

$$c \leq (1+3\epsilon_3)(b+d) - (1+\epsilon_2)b = (1+3\epsilon_3)d + (3\epsilon_3 - \epsilon_2)b \ < (1+3\epsilon_3)d \ , \tag{14}$$

where in the last inequality we used $\epsilon_2 > 3\epsilon_3$. Recalling that $a \geq (1+\epsilon_2)b$ and using $\epsilon_3 = \epsilon_2/24$ we get

$$\frac{c}{a} < \frac{(1+3\epsilon_3)d}{(1+\epsilon_2)b} = \frac{d}{b} \cdot \frac{1+\epsilon_2/8}{1+\epsilon_2} < \frac{d}{b} \cdot \left(1 - \frac{\epsilon_2}{4}\right) \ . \tag{15}$$

This proves the claim. ■

Applying Claim 18, we get that in Line 6 we have

$$\frac{D(\{1,\ldots,i_j-1\})}{D(i_j)} \leq \left(1 - \frac{\epsilon_2}{4}\right) \cdot \frac{D^*(\{1,\ldots,i_j-1\})}{D^*(i_j)} \ . \tag{16}$$

Recalling that by the premise of this case $D^*(i_j) \geq \epsilon_1$, by applying Claim 13 we have that $\{1,\ldots,i_j-1\}$ is an ϵ_1-comparable witness for i_j. Therefore, by Lemma 1, with probability at least $1 - \frac{1}{10\ell}$ the call to COMPARE$(\{i_j\}, \{1,\ldots,i_j-1\}, \frac{\epsilon_2}{16}, \frac{2}{\epsilon_1}, \frac{1}{10\ell})$ in Line 6 either outputs an element of $\{\mathsf{High}, \mathsf{Low}\}$ or outputs a value $v \leq (1-\frac{\epsilon_2}{4})(1+\frac{\epsilon_2}{16})\frac{D^*(\{1,\ldots,i_j-1\})}{D^*(i_j)} < (1-\frac{\epsilon_2}{8})\frac{D^*(\{1,\ldots,i_j-1\})}{D^*(i_j)}$. In either case the algorithm outputs REJECT in Line 6, so we are done with Case 2(a).

Case 2(b): $D^*(i_j) < \epsilon_1$. In this case the algorithm reaches Line 10, and by item 2 of Claim 13, we have that $S_1,\ldots,S_t$ is a partition of $\{1,\ldots,i_j-1\}$ and each set $S_1,\ldots,S_t$ is a $\frac{1}{2}$-comparable witness for i_j, i.e.,

$$\text{for all } i \in \{1,\ldots,t\}, \quad \frac{1}{2}D^*(j) \leq D^*(S_i) \leq 2D^*(j). \tag{17}$$

We use the following lemma:

Claim 19 *In Case 2(b) suppose $i_j > k^*$ is such that $D(i_j) \geq (1+\epsilon_2)D^*(i_j)$ and $D(\{1,\ldots,i_j\}) \in [1-3\epsilon_3, 1+3\epsilon_3]D^*(\{1,\ldots,i_j\})$. Then at least $(\epsilon_4/8)$-fraction of the sets $S_1,\ldots,S_t$ are such that*

$$D(S_i) \leq (1+\epsilon_4)D^*(S_i).$$

Proof: The proof is by contradiction. Let $\rho = 1 - \epsilon_4/8$ and suppose that there are w sets (without loss of generality we call them $S_1,\ldots,S_w$) that satisfy $D(S_i) > (1+\epsilon_4)D^*(S_i)$, where $\rho' = \frac{w}{t} > \rho$. We first observe that the weight of the w subsets $S_1,\ldots,S_w$ under D^*, as a fraction of $D^*(\{1,\ldots,i_j-1\})$, is at least

$$\frac{D^*(S_1 \cup \cdots \cup S_w)}{D^*(S_1 \cup \cdots \cup S_w) + (t-w)\cdot 2D^*(j)} \geq \frac{w\frac{D^*(i_j)}{2}}{w\frac{D^*(i_j)}{2} + (t-w)\cdot 2D^*(j)} = \frac{w}{4t-3w} = \frac{\rho'}{4-3\rho'},$$

where we used the right inequality in Equation (17) on $S_{w+1},\ldots,S_t$ to obtain the leftmost expression above, and the left inequality in Equation (17) (together with the fact that $\frac{x}{x+c}$ is an increasing function of x for all $c > 0$) to obtain the inequality above. This implies that

$$\begin{aligned} D(\{1,\ldots,i_j-1\}) &= \sum_{i=1}^{w} D(S_i) + \sum_{i=w+1}^{t} D(S_i) \geq (1+\epsilon_4)\sum_{i=1}^{w} D^*(S_i) + \sum_{i=w+1}^{t} D(S_i) \\ &\geq (1+\epsilon_4)\frac{\rho'}{4-3\rho'}D^*(\{1,\ldots,i_j-1\}) \\ &\geq (1+\epsilon_4)\frac{\rho}{4-3\rho}D^*(\{1,\ldots,i_j-1\}) \ . \end{aligned} \tag{18}$$

From Equation (18) we have

$$\begin{aligned} D(\{1,\ldots,i_j\}) &\geq (1+\epsilon_4)\frac{\rho}{4-3\rho}D^*(\{1,\ldots,i_j-1\}) + (1+\epsilon_2)D^*(i_j) \\ &\geq \left(1+\frac{3\epsilon_4}{8}\right)D^*(\{1,\ldots,i_j-1\}) + (1+\epsilon_2)D^*(i_j) \end{aligned}$$

where for the first inequality above we used $D(i_j) \geq (1+\epsilon_2)D^*(i_j)$ and for the second inequality we used $(1+\epsilon_4)\frac{\rho}{4-3\rho} \geq 1+\frac{3\epsilon_4}{8}$. This implies that

$$D(\{1,\ldots,i_j\}) \geq \left(1+\frac{3\epsilon_4}{8}\right)D^*(\{1,\ldots,i_j-1\}) + \left(1+\frac{3\epsilon_4}{8}\right)D^*(i_j) = \left(1+\frac{3\epsilon_4}{8}\right)D^*(\{1,\ldots,i_j\})$$

where the inequality follows from $\epsilon_2 \geq \frac{3\epsilon_4}{8}$. Since $\frac{3\epsilon_4}{8} > 3\epsilon_3$, though, this is a contradiction and the claim is proved. ■

Applying Claim 19, and recalling that $h = \Theta(1/\epsilon) = \Theta(1/\epsilon_4)$ sets are chosen randomly in Line 9, we have that with probability at least $9/10$ there is some $r \in \{1,\ldots,h\}$ such that $D(S_{a_r}) \leq (1+\epsilon_4)D^*(S_{a_r})$. Combining this with $D(i_j) \geq (1+\epsilon_2)D^*(i_j)$, we get that

$$\frac{D(S_{a_r})}{D(i_j)} \leq \frac{1+\epsilon_4}{1+\epsilon_2}\cdot\frac{D^*(S_{a_r})}{D^*(i_j)} \leq \left(1-\frac{\epsilon_4}{2}\right)\cdot\frac{D^*(S_{a_r})}{D^*(i_j)}.$$

By Lemma 1, with probability at least $1-\frac{1}{10\ell h}$ the call to COMPARE$(\{i_j\}, S_{a_r}, \frac{\epsilon_4}{8}, 4, \frac{1}{10\ell n})$ in Line 10 either outputs an element of {High,Low } or outputs a value $v \leq (1-\frac{\epsilon_4}{2})(1+\frac{\epsilon_4}{8})\frac{D^*(S_{a_r})}{D^*(i_j)} < (1-\frac{\epsilon_4}{4})\frac{D^*(S_{a_r})}{D^*(i_j)}$. In either case the algorithm outputs REJECT in Line 10, so we are done in Case 2(b). This concludes the proof of soundness and the proof of Theorem 3. ■

4.2 A $\mathrm{poly}(\log N, 1/\epsilon)$-query $\mathsf{PCOND}_\mathsf{D}$ algorithm

It is natural to try and design an algorithm for testing equivalence to a known D^* that uses the simpler PCOND queries rather than general COND queries. In this subsection we present an algorithm PCOND-TEST-KNOWN and prove the following theorem:

Theorem 3 PCOND-TEST-KNOWN *is a $\tilde{O}((\log N)^4/\epsilon^4)$-query $\mathsf{PCOND}_\mathsf{D}$ testing algorithm for testing equivalence to a known distribution D^*. That is, for every pair of distributions D, D^* over $[N]$ (such that D^* is fully specified and there is* PCOND *query access to D) the algorithm outputs* ACCEPT *with probability at least $2/3$ if $D = D^*$ and outputs* REJECT *with probability at least $2/3$ if $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$.*

In the following subsection we will show that a $(\log N)^{\Omega(1)}$ query complexity is inherent in PCOND algorithms for this problem.

Intuition. Let D^* be a fully specified distribution, and let D be a distribution that may be accessed via a $\mathsf{PCOND}_\mathsf{D}$ oracle. The high-level idea of the PCOND-TEST-KNOWN algorithm is the following: we shall try to "catch" a pair of points x, y such that $\frac{D(x)}{D(y)}$ differs significantly from $\frac{D^*(x)}{D^*(y)}$ (so that calling COMPARE$_D$ on $\{x\}, \{y\}$ will reveal this difference).

Let us start by considering the simplest possible case of testing equivalence to the uniform distribution, where $D^*(z) = 1/N$ for every z and hence $\frac{D^*(x)}{D^*(x)+D^*(y)} = 1/2$ for all x, y. In this case, to get a $\mathrm{poly}(1/\epsilon)$-query algorithm it would be sufficient to show that sampling $\Theta(1/\epsilon)$ points uniformly (i.e., according to

D^*) with high probability yields a point x for which $D(x) < D^*(x) - \Omega(\epsilon/N)$, and that sampling $\Theta(1/\epsilon)$ points from $\mathsf{SAMP}_{\mathsf{D}}$ with high probability yields a point y for which $D(x) > D^*(y) + \Omega(\epsilon/N)$, and indeed this can be done without too much difficulty (this is the basis of the uniformity test given in [7]). However, for general D^* it is not sufficient to get such a pair because it is possible that $D^*(y)$ could be much larger than $D^*(x)$. If this were the case then it might happen that both $\frac{D^*(x)}{D^*(y)}$ and $\frac{D(x)}{D(y)}$ are very small, so calling $\textsc{Compare}_D$ on $\{x\}, \{y\}$ cannot efficiently demonstrate that $\frac{D^*(x)}{D^*(y)}$ differs from $\frac{D(x)}{D(y)}$.

To address this issue we partition the points into $O(\log N/\epsilon)$ "buckets" so that within each bucket all points have similar probability according to D^*. We show that if D is ϵ-far from D^*, then either the probability weight of one of these buckets according to D differs significantly from what it is according to D^* (which can be observed by sampling from D), or we can get a pair $\{x, y\}$ *that belong to the same bucket* and for which $D(x)$ is sufficiently smaller than $D^*(x)$ and $D(y)$ is sufficiently larger than $D^*(y)$. For such a pair $\textsc{Compare}$ will efficiently give evidence that D differs from D^*.

The algorithm and its analysis. We define some quantities that are used in the algorithm and its analysis. Let $\eta \stackrel{\text{def}}{=} \epsilon/c$ for some sufficiently large constant c that will be determined later. As described above we partition the domain elements $[N]$ into "buckets" according to their probability weight in D^*. Specifically, for $j = 1, \ldots, \lceil \log(N/\eta) + 1 \rceil$, we let

$$B_j \stackrel{\text{def}}{=} \{x \in [N] : 2^{j-1} \cdot \eta/N \leq D^*(x) < 2^j \cdot \eta/N\}$$

and we let $B_0 \stackrel{\text{def}}{=} \{x \in [N] : D^*(x) < \eta/N\}$. Let $b \stackrel{\text{def}}{=} \lceil \log(N/\eta) + 1 \rceil + 1$ denote the number of buckets.

We further define $J^h \stackrel{\text{def}}{=} \{j : D^*(B_j) \geq \eta/b\}$ to denote the set of indices of "heavy" buckets, and let $J^\ell \stackrel{\text{def}}{=} \{j : D^*(B_j) < \eta/b\}$ denote the set of indices of "light" buckets. Note that we have

$$\sum_{j \in J^\ell \cup \{0\}} D^*(B_j) < 2\eta. \tag{19}$$

The query complexity of the algorithm is dominated by the number of $\mathsf{PCOND}_{\mathsf{D}}$ queries performed in the executions of $\textsc{Compare}$, which by Lemma 1 is upper bounded by

$$O(s^2 \cdot b^2 \cdot (\log s)/\eta^2) = O\left(\frac{(\log \frac{N}{\epsilon})^4 \cdot \log\left((\log \frac{N}{\epsilon})/\epsilon\right)}{\epsilon^4}\right).$$

We argue completeness and soundness below.

Completeness: Suppose that $D = D^*$. Since the expected value of $\widehat{D}(B_j)$ (defined in Line 3) is precisely $D^*(B_j)$, for any fixed value of $j \in \{0, \ldots, \lceil \log(N/\eta) + 1 \rceil\}$ an additive Chernoff bound implies that $\left|D^*(B_j) - \widehat{D}(B_j)\right| > \eta/b$ with failure probability at most $1/(10b)$. By a union bound over all b values of j, the algorithm outputs REJECT in Line 5 with probability at most $1/10$. Later in the algorithm, since $D = D^*$, no matter what points x_i, y_j are sampled from D^* and D respectively, the following holds for each pair (x_i, y_j) such that $D^*(x)/D^*(y) \in [1/2, 2]$. By Lemma 1 (and the setting of the parameters in the calls to $\textsc{Compare}$), the probability that $\textsc{Compare}$ returns Low or a value smaller than $(1 - \delta/(2b)) \cdot (D^*(x)/D^*(y))$, is at most $1/(10s^2)$. A union bound over all (at most s^2) pairs (x_i, y_j) for which $D^*(x)/D^*(y) \in [1/2, 2]$, gives that the probability of outputting REJECT in Line 13 is at most $1/10$. Thus with overall probability at least $8/10$ the algorithm outputs ACCEPT.

Algorithm 7: $\mathsf{PCOND_D}$-TEST-KNOWN

Input: error parameter $\epsilon > 0$; query access to $\mathsf{PCOND_D}$ oracle; explicit description $(D^*(1), \ldots, D^*(N))$ of distribution D^*

1: Call the $\mathsf{SAMP_D}$ oracle $m = \Theta(b^2(\log b)/\eta^2)$ times to obtain points $h_1, \ldots, h_m$ distributed according to D.
2: **for** $j = 0$ to b **do**
3: Let $\widehat{D}(B_j)$ be the fraction of points $h_1, \ldots, h_m$ that lie in B_j.
4: **if** some j has $|D^*(B_j) - \widehat{D}(B_j)| > \eta/b$ **then**
5: output REJECT and exit
6: **end if**
7: **end for**
8: Select $s = \Theta(b/\epsilon)$ points $x_1, \ldots, x_s$ independently from D^*.
9: Call the $\mathsf{SAMP_D}$ oracle $s = \Theta(b/\epsilon)$ times to obtain points $y_1, \ldots, y_s$ distributed according to D.
10: **for all** pairs (x_i, y_j) (where $1 \le i, j \le s$) such that $\frac{D^*(x)}{D^*(y)} \in [1/2, 2]$ **do**
11: Call COMPARE$(\{x\}, \{y\}, \eta/(4b), 2, 1/(10s^2))$
12: **if** COMPARE returns Low or a value smaller than $(1 - \eta/(2b)) \cdot \frac{D^*(x)}{D^*(y)}$ **then**
13: output REJECT (and exit)
14: **end if**
15: **end for**
16: output ACCEPT

Soundness: Now suppose that $d_{\mathrm{TV}}(D, D^*) \ge \epsilon$; our goal is to show that the algorithm rejects with probability at least $2/3$. Since the algorithm rejects if any estimate $\widehat{D}(B_j)$ obtained in Line 3 deviates from $D^*(B_j)$ by more than $\pm\eta/b$, we may assume that all these estimates are indeed $\pm\eta/b$-close to the values $D^*(B_j)$ as required. Moreover, by an additive Chernoff bound (as in the completeness analysis), we have that with overall failure probability at most $1/10$, each j has $|\widehat{D}(B_j) - D(B_j)| \le \eta/b$; we condition on this event going forth. Thus, for every $0 \le j \le b$,

$$D^*(B_j) - 2\eta/b \le D(B_j) \le D^*(B_j) + 2\eta/b \,. \tag{20}$$

Recalling the definition of J^ℓ and Equation (19), we see that

$$\sum_{j \in J^\ell \cup \{0\}} D(B_j) < 4\eta \,. \tag{21}$$

Let

$$d_j \stackrel{\mathrm{def}}{=} \sum_{x \in B_j} |D^*(x) - D(x)| \,, \tag{22}$$

so that $\|D^* - D\|_1 = \sum_j d_j$. By Equations (19) and (21), we have

$$\sum_{j \in J^\ell \cup \{0\}} d_j \le \sum_{j \in J^\ell \cup \{0\}} (D^*(B_j) + D(B_j)) \le 6\eta \,. \tag{23}$$

Since we have (by assumption) that $\|D^* - D\|_1 = 2d_{\mathrm{TV}}(D^*, D) \ge 2\epsilon$, we get that

$$\sum_{j \in J^h \setminus \{0\}} d_j > 2\epsilon - 6\eta \,. \tag{24}$$

Let $N_j \stackrel{\text{def}}{=} |B_j|$ and observe that $N_j \leq D^*(B_j)/p_j \leq 1/p_j$, where $p_j \stackrel{\text{def}}{=} 2^{j-1} \cdot \eta/N$ is the lower bound on the probability (under D^*) of all elements in B_j. For each B_j such that $j \in J^h \setminus \{0\}$, let $H_j \stackrel{\text{def}}{=} \{x \in B_j : D(x) > D^*(x)\}$ and $L_j \stackrel{\text{def}}{=} \{x \in B_j : D(x) < D^*(x)\}$. We have that

$$\sum_{x \in L_j} (D^*(x) - D(x)) + \sum_{x \in H_j} (D(x) - D^*(x)) = d_j \ . \tag{25}$$

Equation (20) may be rewritten as

$$\left| \sum_{x \in L_j} (D^*(x) - D(x)) - \sum_{x \in H_j} (D(x) - D^*(x)) \right| \leq 2\eta/b \ , \tag{26}$$

and so we have both

$$\sum_{x \in L_j} (D^*(x) - D(x)) \geq d_j/2 - \eta/b \quad \text{and} \quad \sum_{x \in H_j} (D(x) - D^*(x)) \geq d_j/2 - \eta/b \ . \tag{27}$$

Also similarly to what we had before, let $H'_j \stackrel{\text{def}}{=} \{x \in B_j : D(x) > D^*(x) + \eta/(bN_j)\}$, and $L'_j \stackrel{\text{def}}{=} \{x \in B_j : D(x) < D^*(x) - \eta/(bN_j)\}$ (recall that $N_j = |B_j|$); these are the element of B_j that are "significantly heavier" (lighter, respectively) under D than under D^*. We have

$$\sum_{x \in L_j \setminus L'_j} (D^*(x) - D(x)) \leq \eta/b \quad \text{and} \quad \sum_{x \in H_j \setminus H'_j} (D(x) - D^*(x)) \leq \eta/b \ . \tag{28}$$

By Equation (24), there exists $j^* \in J^h \setminus \{0\}$ for which $d_{j^*} \geq (2\epsilon - 6\eta)/b$. For this index, applying Equations (27) and (28), we get that

$$\sum_{x \in L'_{j^*}} D^*(x) \geq \sum_{x \in L'_{j^*}} (D^*(x) - D(x)) \geq (\epsilon - 5\eta)/b \ , \tag{29}$$

and similarly,

$$\sum_{x \in H'_{j^*}} D(x) \geq \sum_{x \in H'_{j^*}} (D(x) - D^*(x)) \geq (\epsilon - 5\eta)/b \ . \tag{30}$$

Recalling that $\eta = \epsilon/6$, we have that $(\epsilon - 5\eta)/b = \epsilon/6b$. Now since $s = \Theta(b/\epsilon)$, with probability at least 9/10 it is the case both that some x_i drawn in Line 8 belongs to L'_{j^*} and that some $y_{i'}$ drawn in Line 9 belongs to H'_{j^*}. By the definitions of L'_{j^*} and H'_{j^*} and the fact for each $j > 0$ it holds that $N_j \leq 1/p_j$ and $p_j \leq D^*(x) < 2p_j$ for each $x \in B_j$, we have that

$$D(x_i) < D^*(x_i) - \eta/(bN_{j^*}) \leq D^*(x_i) - (\eta/b)p_{j^*} \leq (1 - \eta/(2b))D^*(x_i) \tag{31}$$

and

$$D(y_{i'}) > D^*(y_{i'}) + \eta/(bN_{j^*}) \geq D^*(y_{i'}) + (\eta/b)p_j \geq (1 + \eta/(2b))D^*(y_{i'}) \ . \tag{32}$$

Therefore,

$$\frac{D(x_i)}{D(y_{i'})} < \frac{1 - \eta/(2b)}{1 + \eta/(2b)} \cdot \frac{D^*(x_i)}{D^*(y_{i'})} < \left(1 - \frac{3\eta}{4b}\right) \cdot \frac{D^*(x_i)}{D^*(y_{i'})} \ . \tag{33}$$

By Lemma 1, with probability at least $1 - 1/(10s^2)$, the output of COMPARE is either Low or is at most $\left(1 - \frac{3\eta}{4b}\right) \cdot \left(1 + \frac{\eta}{4b}\right) < \left(1 - \frac{\eta}{2b}\right)$, causing the algorithm to reject. Thus the overall probability that the algorithm outputs REJECT is at least $8/10 - 1/(10s^2) > 2/3$, and the theorem is proved. ■

4.3 A $(\log N)^{\Omega(1)}$ lower bound for $\mathsf{PCOND}_{\mathsf{D}}$

In this subsection we prove that any $\mathsf{PCOND}_{\mathsf{D}}$ algorithm for testing equivalence to a known distribution must have query complexity at least $(\log N)^{\Omega(1)}$:

Theorem 4 *Fix $\epsilon = 1/2$. There is a distribution D^* over $[N]$ (described below), which is such that any $\mathsf{PCOND}_{\mathsf{D}}$ algorithm for testing whether $D = D^*$ versus $d_{\mathrm{TV}}(D, D^*) \geq \epsilon$ must make $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$ queries.*

The distribution D^*. Fix parameters $r = \Theta\left(\frac{\log N}{\log\log N}\right)$ and $K = \Theta(\log N)$. We partition $[N]$ from left (1) to right (N) into $2r$ consecutive intervals $B_1, \ldots, B_{2r}$, which we henceforth refer to as "buckets." The i-th bucket has $|B_i| = K^i$ (we may assume without loss of generality that N is of the form $\sum_{i=1}^{r} K^i$). The distribution D^* assigns equal probability weight to each bucket, so $D^*(B_i) = 1/(2r)$ for all $1 \leq i \leq 2r$. Moreover D^* is uniform within each bucket, so for all $j \in B_i$ we have $D^*(j) = 1/(2rK^i)$. This completes the specification of D^*.

To prove the lower bound we construct a probability distribution $\mathcal{P}_{\mathrm{No}}$ over possible "No"-distributions. To define the distribution $\mathcal{P}_{\mathrm{No}}$ it will be useful to have the notion of a "bucket-pair." A bucket-pair U_i is $U_i = B_{2i-1} \cup B_{2i}$, i.e. the union of the i-th pair of consecutive buckets.

A distribution D drawn from $\mathcal{P}_{\mathrm{No}}$ is obtained by selecting a string $\pi = (\pi_1, \ldots, \pi_r)$ uniformly at random from $\{\downarrow\uparrow, \uparrow\downarrow\}^r$ and setting D to be D_π, which we now define. The distribution D_π is obtained by perturbing D^* in the following way: for each bucket-pair $U_i = (B_{2i-1}, B_{2i})$,

- If $\pi_i = \uparrow\downarrow$ then the weight of B_{2i-1} is uniformly "scaled up" from $1/(2r)$ to $3/(4r)$ (keeping the distribution uniform within B_{2i-1}) and the weight of B_{2i} is uniformly "scaled down" from $1/(2r)$ to $1/(4r)$ (likewise keeping the distribution uniform within B_{2i}).
- If $\pi_i = \downarrow\uparrow$ then the weight of B_{2i-1} is uniformly "scaled down" from $1/(2r)$ to $1/(4r)$ and the weight of B_{2i} is uniformly "scaled up" from $1/(2r)$ to $3/(4r)$.

Note that for any distribution D in the support of $\mathcal{P}_{\mathrm{No}}$ and any $1 \leq i \leq r$ we have that $D(U_i) = D^*(U_i) = 1/r$.

Every distribution D in the support of $\mathcal{P}_{\mathrm{No}}$ has $d_{\mathrm{TV}}(D^*, D) = 1/2$. Thus Theorem 4 follows immediately from the following:

Theorem 5 *Let A be any (possibly adaptive) algorithm. which makes at most $q \leq \frac{1}{3} \cdot \sqrt{r}$ calls to $\mathsf{PCOND}_{\mathsf{D}}$. Then*

$$\left|\Pr_{D \leftarrow \mathcal{P}_{\mathrm{No}}}\left[A^{\mathsf{PCOND}_{\mathsf{D}}} \text{ outputs } \mathsf{ACCEPT}\right] - \Pr\left[A^{\mathsf{PCOND}_{\mathsf{D}^*}} \text{ outputs } \mathsf{ACCEPT}\right]\right| \leq 1/5. \tag{34}$$

Note that in the first probability of Equation (34) the randomness is over the draw of D from $\mathcal{P}_{\mathrm{No}}$, the internal randomness of A in selecting its query sets, and the randomness of the responses to the $\mathsf{PCOND}_{\mathsf{D}}$ queries. In the second probability the randomness is just over the internal coin tosses of A and the randomness of the responses to the $\mathsf{PCOND}_{\mathsf{D}}$ queries.

Intuition for Theorem 5. A very high-level intuition for the lower bound is that $\mathsf{PCOND}_{\mathsf{D}}$ queries are only useful for "comparing" points whose probabilities are within a reasonable multiplicative ratio of each other. But D^* and every distribution D in the support of P_{No} are such that every two points either have the same

probability mass under all of these distributions (so a $\mathsf{PCOND}_{\mathsf{D}}$ query is not informative), or else the ratio of their probabilities is so skewed that a small number of $\mathsf{PCOND}_{\mathsf{D}}$ queries is not useful for comparing them.

In more detail, we may suppose without loss of generality that in every possible execution, algorithm A first makes q calls to $\mathsf{SAMP}_{\mathsf{D}}$ and then makes q (possibly adaptive) calls to $\mathsf{PCOND}_{\mathsf{D}}$. The more detailed intuition for the lower bound is as follows: First consider the $\mathsf{SAMP}_{\mathsf{D}}$ calls. Since every possible D (whether D^* or a distribution drawn from $\mathcal{P}_{\mathrm{No}}$) puts weight $1/r$ on each bucket-pair $U_1, \ldots, U_r$, a birthday paradox argument implies that in both scenarios, with probability at least $9/10$ (over the randomness in the responses to the $\mathsf{SAMP}_{\mathsf{D}}$ queries) no two of the $q \leq \frac{1}{3}\sqrt{r}$ calls to $\mathsf{SAMP}_{\mathsf{D}}$ return points from the same bucket-pair. Conditioned on this, the distribution of responses to the $\mathsf{SAMP}_{\mathsf{D}}$ queries is exactly the same under D^* and under D where D is drawn randomly from $\mathcal{P}_{\mathrm{No}}$.

For the pair queries, the intuition is that in either setting (whether the distribution D is D^* or a randomly chosen distribution from $\mathcal{P}_{\mathrm{No}}$), making q pair queries will with $1 - o(1)$ probability provide no information that the tester could not simulate for itself. This is because any pair query $\mathsf{PCOND}_{\mathsf{D}}(\{\mathsf{x}, \mathsf{y}\})$ either has x, y in the same bucket B_i or in different buckets $B_i \neq B_j$ with $i < j$. If x, y are both in the same bucket B_i then in either setting $\mathsf{PCOND}_{\mathsf{D}}(\{\mathsf{x}, \mathsf{y}\})$ is equally likely to return x or y. If they belong to buckets B_i, B_j with $i < j$ then in either setting $\mathsf{PCOND}_{\mathsf{D}}(\{\mathsf{x}, \mathsf{y}\})$ will return the one that belongs to P_i with probability $1 - 1/\Theta(K^{j-i}) \geq 1 - 1/\Omega(K)$.

Proof of Theorem 5: As described above, we may fix A to be any $\mathsf{PCOND}_{\mathsf{D}}$ algorithm that makes exactly q calls to $\mathsf{SAMP}_{\mathsf{D}}$ followed by exactly q adaptive calls to $\mathsf{PCOND}_{\mathsf{D}}$.

A *transcript* for A is a full specification of the sequence of interactions that A has with the $\mathsf{PCOND}_{\mathsf{D}}$ oracle in a given execution. More precisely, it is a pair (Y, Z) where $Y = (s_1, \ldots, s_q) \in [N]^q$ and $Z = ((\{x_1, y_1\}, p_1), \ldots, (\{x_q, y_q\}, p_q))$, where $p_i \in \{x_i, y_i\}$ and $x_i, y_i \in [N]$. The idea is that Y is a possible sequence of responses that A might receive to the initial q $\mathsf{SAMP}_{\mathsf{D}}$ queries, $\{x_i, y_i\}$ is a possible pair that could be the input to an i-th $\mathsf{PCOND}_{\mathsf{D}}$ query, and p_i is a possible response that could be received from that query.

We say that a *length-i transcript prefix* is a pair (Y, Z^i) where Y is as above and $Z^i = ((\{x_1, y_1\}, p_1), \ldots, (\{x_i, y_i\}, p_i))$. A PCOND algorithm A may be viewed as a collection of distributions over pairs $\{x, y\}$ in the following way: for each length-i transcript-prefix (Y, Z^i) $(0 \leq i \leq q-1)$, there is a distribution over pairs $\{x_{i+1}, y_{i+1}\}$ that A would use to select the $(i+1)$-st query pair for $\mathsf{PCOND}_{\mathsf{D}}$ given that the length-i transcript prefix of A's execution thus far was (Y, Z^i). We write $\mathrm{T}_{(Y,Z^i)}$ to denote this distribution over pairs.

Let P^* denote the distribution over transcripts induced by running A with oracle $\mathsf{PCOND}_{\mathsf{D}^*}$. Let P^{No} denote the distribution over transcripts induced by first (i) drawing D from $\mathcal{P}_{\mathrm{No}}$, and then (ii) running A with oracle $\mathsf{PCOND}_{\mathsf{D}}$. To prove Theorem 5 it is sufficient to prove that the distribution over transcripts of A is statistically close whether the oracle is D^* or is a random D drawn from P^{No}, i.e. it is sufficient to prove that

$$d_{\mathrm{TV}}(\mathrm{P}^*, \mathrm{P}^{\mathrm{No}}) \leq 1/5. \tag{35}$$

For our analysis we will need to consider variants of algorithm A that, rather than making q calls to $\mathsf{PCOND}_{\mathsf{D}}$, instead "fake" the final $q - k$ of these $\mathsf{PCOND}_{\mathsf{D}}$ queries as described below. For $0 \leq k \leq q$ we define $A^{(k)}$ to be the algorithm that works as follows:

1. $A^{(k)}$ exactly simulates the execution of A in making an initial q $\mathsf{SAMP}_{\mathsf{D}}$ calls and making the first k $\mathsf{PCOND}_{\mathsf{D}}$ queries precisely like A. Let (Y, Z^k) be the length-k transcript prefix of A's execution thus obtained.

2. Exactly like A, algorithm $A^{(k)}$ draws a pair $\{x_{k+1}, y_{k+1}\}$ from $\mathrm{T}_{(Y,Z^k)}$. However, instead of calling $\mathsf{PCOND}_{\mathsf{D}}(\{\mathsf{x}_{\mathsf{k+1}}, \mathsf{y}_{\mathsf{k+1}}\})$ to obtain p_{k+1}, algorithm $A^{(k)}$ generates p_{k+1} in the following manner:

   (i) If x_{k+1} and y_{k+1} both belong to the same bucket B_ℓ then p_{k+1} is chosen uniformly from $\{x_{k+1}, y_{k+1}\}$.

   (ii) If one of $\{x_{k+1}, y_{k+1}\}$ belongs to B_ℓ and the other belongs to $B_{\ell'}$ for some $\ell < \ell'$, then p_{k+1} is set to be the element of $\{x_{k+1}, y_{k+1}\}$ that belongs to B_ℓ.

   Let (Y, Z^{k+1}) be the length-$(k+1)$ transcript prefix obtained by appending $(\{x_{k+1}, y_{k+1}\}, p_{k+1})$ to Z^k. Algorithm A' continues in this way for a total of $q-k$ stages; i.e. it next draws $\{x_{k+2}, y_{k+2}\}$ from $\mathrm{T}_{(Y,Z^{k+1})}$ and generates p_{k+2} as described above; then (Y, Z^{k+2}) is the length-$(k+2)$ transcript prefix obtained by appending $(\{x_{k+2}, y_{k+2}\}, p_{k+2})$ to Z^{k+1}; and so on. At the end of the process a transcript (Y, Z^q) has been constructed.

Let $\mathrm{P}^{*,(k)}$ denote the distribution over final transcripts (Y, Z^q) that are obtained by running $A^{(k)}$ on a $\mathsf{PCOND}_{\mathsf{D}^*}$ oracle. Let $\mathrm{P}^{\mathrm{No},(k)}$ denote the distribution over final transcripts (Y, Z^q) that are obtained by (i) first drawing D from P_{No}, and then (ii) running $A^{(k)}$ on a $\mathsf{PCOND}_{\mathsf{D}}$ oracle. Note that $\mathrm{P}^{*,(q)}$ is identical to P^* and $\mathrm{P}^{\mathrm{No},(q)}$ is identical to P^{No} (since algorithm $A^{(q)}$, which does not fake any queries, is identical to algorithm A).

Recall that our goal is to prove Equation (35). Since $\mathrm{P}^{*,(q)} = \mathrm{P}^*$ and $\mathrm{P}^{\mathrm{No},(q)} = \mathrm{P}^{\mathrm{No}}$, Equation (35) is an immediate consequence (using the triangle inequality for total variation distance) of the following two lemmas, which we prove below:

Lemma 20 $d_{\mathrm{TV}}(\mathrm{P}^{*,(0)}, \mathrm{P}^{\mathrm{No},(0)}) \leq 1/10$.

Lemma 21 *For all* $0 \leq k < q$, *we have* $d_{\mathrm{TV}}(\mathrm{P}^{*,(k)}, \mathrm{P}^{*,(k+1)}) \leq 1/(20q)$ *and* $d_{\mathrm{TV}}(\mathrm{P}^{\mathrm{No},(k)}, \mathrm{P}^{\mathrm{No},(k+1)}) \leq 1/(20q)$.

Proof of Lemma 20: Define P_0^* to be the distribution over outcomes of the q calls to $\mathsf{SAMP}_{\mathsf{D}}$ (i.e. over length-0 transcript prefixes) when $D = D^*$. Define P^{No} to be the distribution over outcomes of the q calls to $\mathsf{SAMP}_{\mathsf{D}}$ when D is drawn from $\mathcal{P}_{\mathrm{No}}$. We begin by noting that by the data processing inequality for total variation distance, we have $d_{\mathrm{TV}}(\mathrm{P}^{*,(0)}, \mathrm{P}^{\mathrm{No},(0)}) \leq d_{\mathrm{TV}}(\mathrm{P}_0^*, \mathrm{P}_0^{\mathrm{No}})$ (indeed, after the calls to respectively $\mathsf{SAMP}_{\mathsf{D}}$ and $\mathsf{SAMP}_{\mathsf{D}^*}$, the same randomized function F – which fakes all remaining oracle calls – is applied to the two resulting distributions over length-0 transcript prefixes P_0^* and $\mathrm{P}_0^{\mathrm{No}}$). In the rest of the proof we show that $d_{\mathrm{TV}}(\mathrm{P}_0^*, \mathrm{P}_0^{\mathrm{No}}) \leq 1/10$.

Let E denote the event that the q calls to $\mathsf{SAMP}_{\mathsf{D}}$ yield points $s_1, \ldots, s_q$ such that no bucket-pair U_i contains more than one of these points. Since $D^*(U_i) = 1/r$ for all i,

$$\mathrm{P}_0^*(E) = \prod_{j=0}^{q-1} \left(1 - \frac{j}{r}\right) \geq 9/10\ , \tag{36}$$

where Equation (36) follows from a standard birthday paradox analysis and the fact that $q \leq \frac{1}{3}\sqrt{r}$. Since for each possible outcome of D drawn from $\mathcal{P}_{\mathrm{No}}$ we have $D(U_i) = 1/r$ for all i, we further have that also

$$\mathrm{P}_0^{\mathrm{No}}(E) = \prod_{j=0}^{q-1} \left(1 - \frac{j}{r}\right). \tag{37}$$

We moreover claim that the two conditional distributions $(\mathrm{P}_0^*|E)$ and $(\mathrm{P}_0^{\mathrm{No}}|E)$ are identical, i.e.

$$(\mathrm{P}_0^*|E) = (\mathrm{P}_0^{\mathrm{No}}|E). \tag{38}$$

To see this, fix any sequence $(\ell_1, \ldots, \ell_q) \in [r]^q$ such that $\ell_i \neq \ell_j$ for all $i \neq j$. Let $(s_1, \ldots, s_q) \in [N]^q$ denote a draw from $(\mathrm{P}_0^*|E)$, The probability that ($s_i \in U_{\ell_i}$ for all $1 \leq i \leq q$) is precisely $1/r^q$. Now given that $s_i \in U_{\ell_i}$ for all i, it is clear that s_i is equally likely to lie in $B_{2\ell_i - 1}$ and in $B_{2\ell_i}$, and given that it lies in a particular one of the two buckets, it is equally likely to be any element in that bucket. This is true independently for all $1 \leq i \leq q$.

Now let $(s_1, \ldots, s_q) \in [N]^q$ denote a draw from $(\mathrm{P}_0^{\mathrm{No}}|E)$. Since each distribution D in the support of $\mathcal{P}_{\mathrm{No}}$ has $D(U_i) = 1/r$ for all i, we likewise have that the probability that ($s_i \in U_{\ell_i}$ for all $1 \leq i \leq q$) is precisely $1/r^q$. Now given that $s_i \in U_{\ell_i}$ for all i, we have that s_i is equally likely to lie in $B_{2\ell_i - 1}$ and in $B_{2\ell_i}$; this is because π_i (recall that π determines $D = D_\pi$) is equally likely to be $\uparrow\downarrow$ (in which case $D(B_{2\ell_i - 1}) = 3/(4r)$ and $D(B_{2\ell_i}) = 1/(4r)$) as it is to be $\downarrow\uparrow$ (in which case $D(B_{2\ell_i - 1}) = 1/(4r)$ and $D(B_{2\ell_i}) = 3/(4r)$). Additionally, given that s_i lies in a particular one of the two buckets, it is equally likely to be any element in that bucket. This is true independently for all $1 \leq i \leq q$ (because conditioning on E ensures that no two elements of $s_1, \ldots, s_q$ lie in the same bucket-pair, so there is "fresh randomness for each i"), and so indeed the two conditional distributions $(\mathrm{P}_0^*|E)$ and $(\mathrm{P}_0^{\mathrm{No}}|E)$ are identical.

Finally, the claimed bound $d_{\mathrm{TV}}(\mathrm{P}_0^*, \mathrm{P}_0^{\mathrm{No}}) \leq 1/10$ follows directly from Equations (36), (37) and (38). ■

Proof of Lemma 21: Consider first the claim that $d_{\mathrm{TV}}(\mathrm{P}^{*,(k)}, \mathrm{P}^{*,(k+1)}) \leq 1/(20q)$. Fix any $0 \leq k < q$. The data processing inequality for total variation distance implies that $d_{\mathrm{TV}}(\mathrm{P}^{*,(k)}, \mathrm{P}^{*,(k+1)})$ is at most the variation distance between random variables X and Y, where

- X is the random variable obtained by running A on COND_{D^*} to obtain a length-k transcript prefix (Y, Z^k), then drawing $\{x_{k+1}, y_{k+1}\}$ from $\mathrm{T}_{(Y,Z^k)}$, then setting p_{k+1} to be the output of $\mathsf{PCOND}_{\mathsf{D}^*}(\{\mathsf{x_{k+1}}, \mathsf{y_{k+1}}\})$; and
- Y is the random variable obtained by running A on COND_{D^*} to obtain a length-k transcript prefix (Y, Z^k), then drawing $\{x_{k+1}, y_{k+1}\}$ from $\mathrm{T}_{(Y,Z^k)}$, then setting p_{k+1} according to the rules 2(i) and 2(ii) given above.

Consider any fixed outcome of (Y, Z^k) and $\{x_{k+1}, y_{k+1}\}$. If rule 2(i) is applied (x_{k+1} and y_{k+1} are in the same bucket) then there is zero contribution to the variation distance between X and Y, because choosing a uniform element of $\{x_{k+1}, y_{k+1}\}$ is a perfect simulation of $\mathsf{PCOND}_{\mathsf{D}}(\{\mathsf{x_{k+1}}, \mathsf{y_{k+1}}\})$. If rule 2(ii) is applied then the contribution is at most $O(1/K) < 1/20q$, because $\mathsf{PCOND}_{\mathsf{D}^*}(\{\mathsf{x_{k+1}y_{k+1}}\})$ would return a different outcome from rule 2(ii) with probability $1/\Theta(K^{\ell' - \ell}) = O(1/K)$. Averaging over all possible outcomes of (Y, Z^k) and $\{x_{k+1}, y_{k+1}\}$ we get that the variation distance between X and Y is at most $1/20q$ as claimed.

An identical argument shows that similarly $d_{\mathrm{TV}}(\mathrm{P}^{\mathrm{No},(k)}, \mathrm{P}^{\mathrm{No},(k+1)}) \leq 1/(20q)$. The key observation is that for any distribution D in the support of P^{No}, as with D^* it is the case that points in the same bucket have equal probability under D and a point y that is $\ell' - \ell$ buckets lower than x has probability only $1/\Theta(K^{\ell' - \ell})$ of being returned by a call to $\mathsf{PCOND}_{\mathsf{D}}(\{\mathsf{x}, \mathsf{y}\})$. This concludes the proof of Lemma 21 and of Theorem 4. ■

5 Testing equality between two unknown distributions

5.1 An approach based on PCOND queries

In this subsection we consider the problem of testing whether two unknown distributions D_1, D_2 are identical versus ϵ-far, given PCOND access to these distributions. Although this is known to require $\Omega\big(N^{2/3}\big)$ many samples in the standard model [4, 30], we are able to give a $\mathrm{poly}(\log N, 1/\epsilon)$-query algorithm using PCOND queries, by taking advantage of comparisons to perform some sort of *clustering* of the domain.

On a high level the algorithm works as follows. First it obtains (with high probability) a small set of points R such that almost every element in $[N]$, except possibly for some negligible subset according to D_1, has probability weight (under D_1) close to some "representative" in R. Next, for each representative r in R it obtains an estimate of the weight, according to D_1, of a set of points U such that $D_1(u)$ is close to $D_1(r)$ for each u in U (i.e, r's "neighborhood under D_1"). This is done using the procedure ESTIMATE-NEIGHBORHOOD from Subsection 3.2). Note that these neighborhoods can be interpreted roughly as a succinct *cover* of the support of D_1 into (not necessarily disjoint) sets of points, where within each set the points have similar weight (according to D_1). Our algorithm is based on the observation that, if D_1 and D_2 are far from each other, it must be the case that one of these sets, denoted U^*, reflects it in one of the following ways: (1) $D_2(U^*)$ differs significantly from $D_1(U^*)$; (2) U^* contains a subset of points V^* such that $D_2(v)$ differs significantly from $D_2(r)$ for each v in V^*, and either $D_1(V^*)$ is relatively large or $D_2(V^*)$ is relatively large. (This structural result is made precise in Lemma 23). We thus take additional samples, both from D_1 and from D_2, and compare the weight (according to both distributions) of each point in these samples to the representatives in R (using the procedure COMPARE from Subsection 3.1). In this manner we detect (with high probability) that either (1) or (2) holds.
We begin by formalizing the notion of a cover discussed above:

Definition 6 (Weight-Cover) *Given a distribution D on $[N]$ and a parameter $\epsilon_1 > 0$, we say that a point $i \in [N]$ is* ϵ_1-covered *by a set $R = \{r_1, \ldots, r_t\} \subseteq [N]$ if there exists a point $r_j \in R$ such that $D(i) \in [1/(1+\epsilon_1), 1+\epsilon_1]D(r_j)$. Let the set of points in $[N]$ that are ϵ_1-covered by R be denoted by $C^D_{\epsilon_1}(R)$. We say that R is an* (ϵ_1, ϵ_2)-cover for D *if $D([N] \setminus C^D_{\epsilon_1}(R)) \leq \epsilon_2$.*

The following lemma says that a small sample of points drawn from D gives a cover with high probability:

Lemma 22 *Let D be any distribution over $[N]$. Given any fixed $c > 0$, there exists a constant $c' > 0$ such that with probability at least $99/100$, a sample R of size $m = c' \frac{\log(N/\epsilon)}{\epsilon^2} \cdot \log\left(\frac{\log(N/\epsilon)}{\epsilon}\right)$ drawn according to distribution D is an $(\epsilon/c, \epsilon/c)$-cover for D.*

Proof: Let t denote $\lceil \ln(2cN/\epsilon) \cdot \frac{c}{\epsilon} \rceil$. We define t "buckets" of points with similar weight under D as follows: for $i = 0, 1, \ldots, t-1$, define $B_i \subseteq [N]$ to be

$$B_i \stackrel{\text{def}}{=} \left\{ x \in [N] : \frac{1}{(1+\epsilon/c)^{i+1}} < D(x) \leq \frac{1}{(1+\epsilon/c)^i} \right\}.$$

Let L be the set of points x which are not in any of $B_0, \ldots, B_{t-1}$ (because $D(x)$ is too small); since every point in L has $D(x) < \frac{\epsilon}{2cN}$, one can see that $D(L) \leq \frac{\epsilon}{2c}$.

It is easy to see that if the sample R contains a point from a bucket B_j then every point $y \in B_j$ is $\frac{\epsilon}{c}$-covered by R. We say that bucket B_i is *insignificant* if $D(B_i) \leq \frac{\epsilon}{2ct}$; otherwise bucket B_i is *significant*. It

is clear that the total weight under D of all insignificant buckets is at most $\epsilon/2c$. Thus if we can show that for the claimed sample size, with probability at least $99/100$ every significant bucket has at least one of its points in R, we will have established the lemma.

This is a simple probabilistic calculation: fix any significant bucket B_j. The probability that m random draws from D all miss B_j is at most $(1 - \frac{\epsilon}{2ct})^m$, which is at most $\frac{1}{100t}$ for a suitable (absolute constant) choice of c'. Thus a union bound over all (at most t) significant buckets gives that with probability at least $99/100$, no significant bucket is missed by R. ■

Lemma 23 *Suppose $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$, and let $R = \{r_1, \ldots, r_t\}$ be an $(\tilde{\epsilon}, \tilde{\epsilon})$-cover for D_1 where $\tilde{\epsilon} \leq \epsilon/100$. Then, there exists $j \in [t]$ such that at least one of the following conditions holds for every $\alpha \in [\tilde{\epsilon}, 2\tilde{\epsilon}]$:*

1. *$D_1(U_\alpha^{D_1}(r_j)) \geq \frac{\tilde{\epsilon}}{t}$ and $D_2(U_\alpha^{D_1}(r_j)) \notin [1 - \tilde{\epsilon}, 1 + \tilde{\epsilon}] D_1(U_\alpha^{D_1}(r_j))$, or $D_1(U_\alpha^{D_1}(r_j)) < \frac{\tilde{\epsilon}}{t}$ and $D_2(U_\alpha^{D_1}(r_j)) > \frac{2\tilde{\epsilon}}{t}$;*
2. *$D_1(U_\alpha^{D_1}(r_j)) \geq \frac{\tilde{\epsilon}}{t}$, and at least a $\tilde{\epsilon}$-fraction of the points i in $U_\alpha^{D_1}(r_j)$ satisfy $\frac{D_2(i)}{D_2(r_j)} \notin [1/(1 + \alpha + \tilde{\epsilon}), 1 + \alpha + \tilde{\epsilon}]$;*
3. *$D_1(U_\alpha^{D_1}(r_j)) \geq \frac{\tilde{\epsilon}}{t}$, and the total weight according to D_2 of the points i in $U_\alpha^{D_1}(r_j)$ for which $\frac{D_2(i)}{D_2(r_j)} \notin [1/(1 + \alpha + \tilde{\epsilon}), 1 + \alpha + \tilde{\epsilon}]$ is at least $\frac{\tilde{\epsilon}^2}{t}$;*

Proof: Without loss of generality, we can assume that $\epsilon \leq 1/4$. Suppose, contrary to the claim, that for each r_j there exists $\alpha_j \in [\tilde{\epsilon}, 2\tilde{\epsilon}]$ such that if we let $U_j \stackrel{\mathrm{def}}{=} U_{\alpha_j}^{D_1}(r_j)$, then the following holds:

1. If $D_1(U_j) < \frac{\tilde{\epsilon}}{t}$, then $D_2(U_j) \leq \frac{2\tilde{\epsilon}}{t}$;
2. If $D_1(U_j) \geq \frac{\tilde{\epsilon}}{t}$, then:
   (a) $D_2(U_j) \in [1 - \tilde{\epsilon}, 1 + \tilde{\epsilon}] D_1(U_j)$;
   (b) Less than an $\tilde{\epsilon}$-fraction of the points y in U_j satisfy $\frac{D_2(y)}{D_2(r_j)} \notin [1/(1 + \alpha_j + \tilde{\epsilon}), 1 + \alpha_j + \tilde{\epsilon}]$;
   (c) The total weight according to D_2 of the points y in U_j for which $\frac{D_2(y)}{D_2(r_j)} \notin [1/(1 + \alpha_j + \tilde{\epsilon}), 1 + \alpha_j + \tilde{\epsilon}]$ is at most $\frac{\tilde{\epsilon}^2}{t}$;

We show that in such a case $d_{\mathrm{TV}}(D_1, D_2) < \epsilon$, contrary to the premise of the claim.

Consider each point $r_j \in R$ such that $D_1(U_j) \geq \frac{\tilde{\epsilon}}{t}$. By the foregoing discussion (point 2(a)), $D_2(U_j) \in [1 - \tilde{\epsilon}, 1 + \tilde{\epsilon}] D_1(U_j)$. By the definition of U_j (and since $\alpha_j \leq 2\tilde{\epsilon}$),

$$D_1(r_j) \in [1/(1 + 2\tilde{\epsilon}), 1 + 2\tilde{\epsilon}] \, \frac{D_1(U_j)}{|U_j|} \, . \tag{39}$$

Turning to bound $D_2(r_j)$, on one hand (by 2(b))

$$D_2(U_j) = \sum_{y \in U_j} D_2(y) \geq \tilde{\epsilon}|U_j| \cdot 0 + (1 - \tilde{\epsilon})|U_j| \cdot \frac{D_2(r_j)}{1 + 3\tilde{\epsilon}} \, , \tag{40}$$

and so

$$D_2(r_j) \leq \frac{(1 + 3\tilde{\epsilon}) D_2(U_j)}{(1 - \tilde{\epsilon})\,|U_j|} \leq (1 + 6\tilde{\epsilon}) \frac{D_1(U_j)}{|U_j|} \, . \tag{41}$$

On the other hand (by 2(c)),

$$D_2(U_j) = \sum_{y \in U_j} D_2(y) \leq \frac{\tilde{\epsilon}^2}{t} + |U_j| \cdot (1 + 3\tilde{\epsilon}) D_2(r_j) \ , \tag{42}$$

and so

$$D_2(r_j) \geq \frac{D_2(U_j) - \tilde{\epsilon}^2/t}{(1+3\tilde{\epsilon})\,|U_j|} \geq \frac{(1-\tilde{\epsilon})D_1(U_j) - \tilde{\epsilon} D_1(U_j)}{(1+3\tilde{\epsilon})\,|U_j|} \geq (1 - 5\tilde{\epsilon}) \frac{D_1(U_j)}{|U_j|} \ . \tag{43}$$

Therefore, for each such r_j we have

$$D_2(r_j) \in [1 - 8\tilde{\epsilon}, 1 + 10\tilde{\epsilon}] D_1(r_j) \ . \tag{44}$$

Let $C \stackrel{\text{def}}{=} \bigcup_{j=1}^{t} U_j$. We next partition the points in C so that each point $i \in C$ is assigned to some $r_{j(i)}$ such that $i \in U_{j(i)}$. We define the following "bad" subsets of points in $[N]$:

1. $B_1 \stackrel{\text{def}}{=} [N] \setminus C$, so that $D_1(B_1) \leq \tilde{\epsilon}$ (we later bound $D_2(B_1)$);
2. $B_2 \stackrel{\text{def}}{=} \{i \in C : D_1(U_{j(i)}) < \tilde{\epsilon}/t\}$, so that $D_1(B_2) \leq \tilde{\epsilon}$ and $D_2(B_2) \leq 2\tilde{\epsilon}$;
3. $B_3 \stackrel{\text{def}}{=} \{i \in C \setminus B_2 : D_2(i) \notin [1/(1+3\tilde{\epsilon}), 1+3\tilde{\epsilon}] D_2(r_{j(i)})\}$, so that $D_1(B_3) \leq 2\tilde{\epsilon}$ and $D_2(B_3) \leq \tilde{\epsilon}^2$.

Let $B \stackrel{\text{def}}{=} B_1 \cup B_2 \cup B_3$. Observe that for each $i \in [N] \setminus B$ we have that

$$D_2(i) \in [1/(1+3\tilde{\epsilon}), 1+3\tilde{\epsilon}] D_2(r_{j(i)}) \subset [1 - 15\tilde{\epsilon}, 1 + 15\tilde{\epsilon}] D_1(r_{j(i)}) \subset [1 - 23\tilde{\epsilon}, 1 + 23\tilde{\epsilon}] D_1(i) \ , \tag{45}$$

where the first containment follows from the fact that $i \notin B$, the second follows from Equation (44), and the third from the fact that $i \in U_{j(i)}$. In order to complete the proof we need a bound on $D_2(B_1)$, which we obtain next.

$$\begin{aligned} D_2(B_1) &= 1 - D_2([N] \setminus B_1) \leq 1 - D_2([N] \setminus B) \leq 1 - (1 - 23\tilde{\epsilon}) D_1([N] \setminus B) \\ &\leq 1 - (1 - 23\tilde{\epsilon})(1 - 4\tilde{\epsilon}) \leq 27\tilde{\epsilon} \ . \end{aligned} \tag{46}$$

Therefore,

$$\begin{aligned} d_{\mathrm{TV}}(D_1, D_2) &= \frac{1}{2} \sum_{i=1}^{N} |D_1(i) - D_2(i)| \\ &\leq \frac{1}{2} \Big(D_1(B) + D_2(B) + \sum_{i \notin B} 23\tilde{\epsilon} D_1(i) \Big) < \epsilon \ , \end{aligned} \tag{47}$$

and we have reached a contradiction. ■

Theorem 6 *If $D_1 = D_2$ then with probability at least $2/3$ Algorithm* PCOND-TEST-EQUALITY-UNKNOWN *returns* ACCEPT*, and if $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$, then with probability at least $2/3$ Algorithm* PCOND-TEST-EQUALITY-UNKNOWN *returns* REJECT*. The number of* PCOND *queries performed by the algorithm is $\tilde{O}\big(\frac{\log^6 N}{\epsilon^{21}}\big)$.*

Algorithm 8: Algorithm $\mathsf{PCOND}_{\mathsf{D}_1,\mathsf{D}_2}$-TEST-EQUALITY-UNKNOWN

Input: PCOND query access to distributions D_1 and D_2 and a parameter ϵ.

1. Set $\tilde{\epsilon} = \epsilon/100$.
2. Draw a sample R of size $t = \tilde{\Theta}\left(\frac{\log N}{\epsilon^2}\right)$ from D_1.
3. For each $r_j \in R$:
   (a) Call ESTIMATE-NEIGHBORHOOD$_{D_1}$ on r_j with $\kappa = \tilde{\epsilon}$, $\eta = \frac{\tilde{\epsilon}}{8}$, $\beta = \frac{\tilde{\epsilon}}{2t}$, $\delta = \frac{1}{100t}$ and let the output be denoted by $(\hat{w}_j^{(1)}, \alpha_j)$.
   (b) Set $\theta = \kappa\eta\beta\delta/64 = \tilde{\Theta}(\epsilon^7/\log^2 N)$.
   (c) Draw a sample S_1 from D_1, of size $s_1 = \Theta\left(\frac{t}{\epsilon^2}\right) = \tilde{\Theta}\left(\frac{\log N}{\epsilon^4}\right)$.
   (d) Draw a sample S_2 from D_2, of size $s_2 = \Theta\left(\frac{t \log t}{\epsilon^3}\right) = \tilde{\Theta}\left(\frac{\log N}{\epsilon^5}\right)$.
   (e) For each point $i \in S_1 \cup S_2$ call COMPARE$_{D_1}(\{r_j\}, \{i\}, \theta/4, 4, 1/(200t(s_1+s_2)))$ and COMPARE$_{D_2}(\{r_j\}, \{i\}, \theta/4, 4, 1/(200t(s_1+s_2)))$, and let the outputs be denoted $\rho_{r_j}^{(1)}(i)$ and $\rho_{r_j}^{(2)}(i)$, respectively (where in particular these outputs may be High or Low).
   (f) Let $\hat{w}_j^{(2)}$ be the fraction of occurrences of $i \in S_2$ such that $\rho_{r_j}^{(1)}(i) \in [1/(1+\alpha_j+\theta/2), 1+\alpha_j+\theta/2]$.
   (g) If ($\hat{w}_j^{(1)} \leq \frac{3}{4}\frac{\tilde{\epsilon}}{t}$ and $\hat{w}_j^{(2)} > \frac{3}{2}\frac{\tilde{\epsilon}}{t}$) or ($\hat{w}_j^{(1)} > \frac{3}{4}\frac{\tilde{\epsilon}}{t}$ and $\hat{w}_j^{(2)}/\hat{w}_j^{(1)} \notin [1-\tilde{\epsilon}/2, 1+\tilde{\epsilon}/2]$), then output REJECT.
   (h) If there exists $i \in S_1 \cup S_2$ such that $\rho_{r_j}^{(1)}(i) \in [1/(\alpha_j+\tilde{\epsilon}/2), 1+\alpha_j+\tilde{\epsilon}/2]$ and $\rho_{r_j}^{(2)}(i) \notin [1/(\alpha_j+3\tilde{\epsilon}/2), 1+\alpha_j+3\tilde{\epsilon}/2]$, then output REJECT.
4. Output ACCEPT.

Proof: The number of queries performed by the algorithm is the sum of: (1) t times the number of queries performed in each execution of ESTIMATE-NEIGHBORHOOD (in Line 3-a) and (2) $t \cdot (s_1+s_2) = O(t \cdot s_2)$ times the number of queries performed in each execution of COMPARE (in Line 3-e). By Lemma 2 (and the settings of the parameters in the calls to ESTIMATE-NEIGHBORHOOD), the first term is $O\left(t \cdot \frac{\log(1/\delta)\cdot\log(\log(1/\delta)/(\beta\eta^2))}{\kappa^2\eta^4\beta^3\delta^2}\right) = \tilde{O}\left(\frac{\log^6 N}{\epsilon^{19}}\right)$, and by Lemma 1 (and the settings of the parameters in the calls to COMPARE), the second term is $O\left(t \cdot s_2 \cdot \frac{\log(t\cdot s_2)}{\theta^2}\right) = \tilde{O}\left(\frac{\log^6 N}{\epsilon^{21}}\right)$, so that we get the bound stated in the theorem.

We now turn to establishing the correctness of the algorithm. We shall use the shorthand U_j for $U_{\alpha_j}^{D_1}(r_j)$, and U'_j for $U_{\alpha_j+\theta}^{D_1}(r_j)$. We consider the following "desirable" events.

1. The event E_1 is that the sample R is a $(\tilde{\epsilon}, \tilde{\epsilon})$-weight-cover for D_1 (for $\tilde{\epsilon} = \epsilon/100$). By Lemma 22 (and an appropriate constant in the $\Theta(\cdot)$ notation for the size of R), the probability that E_1 holds is at least $99/100$.

2. The event E_2 is that all calls to the procedure ESTIMATE-NEIGHBORHOOD are as specified by Lemma 2. By the setting of the confidence parameter in the calls to the procedure, the event E_2 holds with probability at least $99/100$.

3. The event E_3 is that all calls to the procedure COMPARE are as specified by Lemma 1. By the setting of the confidence parameter in the calls to the procedure, the event E_3 holds with probability at least $99/100$.

4. The event E_4 is that $D_2(U'_j \setminus U_j) \leq \eta\beta/16 = \tilde{\epsilon}^2/(256t)$ for each j. If $D_2 = D_1$ then this event follows from E_2. Otherwise, it holds with probability at least $99/100$ by the setting of θ and the choice of α_j (as shown in the proof of Lemma 2 in the analysis of the event E_1 there).

5. The event E_5 is defined as follows. For each j, if $D_2(U_j) \geq \tilde{\epsilon}/(4t)$, then $|S_2 \cap U_j|/|S_2| \in [1 - \tilde{\epsilon}/10, 1 + \tilde{\epsilon}/10]D_2(U_j)$, and if $D_2(U_j) < \tilde{\epsilon}/(4t)$ then $|S_2 \cap U_j|/|S_2| < (1 + \tilde{\epsilon}/10)\tilde{\epsilon}/(4t)$. This event holds with probability at least $99/100$ by applying a multiplicative Chernoff bound in the first case, and Corollary 10 in the second.

6. The event E_6 is that for each j we have $|S_2 \cap (U'_j \setminus U_j)|/|S_2| \leq \tilde{\epsilon}^2/(128t)$. Conditioned on E_4, the event E_6 holds with probability at least $99/100$ by applying Corollary 10.

From this point on we assume that events $E_1 - E_6$ all hold. Note that in particular this implies the following:

1. By E_2, for every j:
   - If $D_1(U_j) \geq \beta = \tilde{\epsilon}/(2t)$, then $\hat{w}_j^{(1)} \in [1 - \eta, 1 + \eta]D_1(U_j) = [1 - \tilde{\epsilon}/8, 1 + \tilde{\epsilon}/8]D_1(U_j)$.
   - If $D_1(U_j) < \tilde{\epsilon}/(2t)$, then $\hat{w}_j^{(1)} \leq (1 + \tilde{\epsilon}/8)(\tilde{\epsilon}/(2t))$.
2. By E_3, for every j and for each point $i \in S_1 \cup S_2$:
   - If $i \in U_j$, then $\rho_{r_j}^{(1)}(i) \in [1/(1 + \alpha_j + \frac{\theta}{2}), 1 + \alpha_j + \frac{\theta}{2}]$.
   - If $i \notin U'_j$, then $\rho_{r_j}^{(1)}(i) \notin [1/(1 + \alpha_j + \frac{\theta}{2}), 1 + \alpha_j + \frac{\theta}{2}]$.
3. By the previous item and E_4–E_6:
   - If $D_2(U_j) \geq \tilde{\epsilon}/(4t)$, then $\hat{w}_j^{(2)} \geq (1-\tilde{\epsilon}/10)D_2(U_j)$ and $\hat{w}_j^{(2)} \leq (1+\tilde{\epsilon}/10)D_2(U_j)+\tilde{\epsilon}^2/(128t) \leq (1 + \tilde{\epsilon}/8)D_2(U_j)$.
   - If $D_2(U_j) < \tilde{\epsilon}/(4t)$ then $\hat{w}_j^{(2)} \leq (1 + \tilde{\epsilon}/10)\tilde{\epsilon}/(4t) + \tilde{\epsilon}^2/(128t) \leq (1 + \tilde{\epsilon}/4)(\tilde{\epsilon}/(4t))$.

Completeness. Assume D_1 and D_2 are the same distribution D. For each j, if $D(U_j) \geq \tilde{\epsilon}/t$, then by the foregoing discussion, $\hat{w}_j^{(1)} \geq (1 - \tilde{\epsilon}/8)D(U_j) > 3\tilde{\epsilon}/(4t)$ and $\hat{w}_j^{(2)}/\hat{w}_j^{(1)} \in [(1 - \tilde{\epsilon}/8)^2, (1 + \tilde{\epsilon}/8)^2] \subset [1 - \tilde{\epsilon}/2, 1 + \tilde{\epsilon}/2]$, so that the algorithm does not reject in Line 3-g. Otherwise (i.e., $D(U_j) < \tilde{\epsilon}/t$), we consider two subcases. Either $D(U_j) \leq \tilde{\epsilon}/(2t)$, in which case $\hat{w}_j^{(1)} \leq 3\tilde{\epsilon}/(4t)$, or $\tilde{\epsilon}/(2t) < D(U_j) < \tilde{\epsilon}/t$, and then $\hat{w}_j^{(1)} \in [1 - \tilde{\epsilon}/8, 1 + \tilde{\epsilon}/8]D_1(U_j)$. Since in both cases $\hat{w}_j^{(2)} \leq (1 + \tilde{\epsilon}/8)D(U_j) \leq 3\tilde{\epsilon}/(2t)$, the algorithm does not reject in Line 3-g. By E_3, the algorithm does not reject in Line 3-h either. We next turn to establish soundness.

Soundness. Assume $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$. By applying Lemma 23 on R (and using E_1), there exists an index j for which one of the items in the lemma holds. We denote this index by j^*, and consider the three items in the lemma.

1. If Item 1 holds, then we consider its two cases:

   (a) In the first case, $D_1(U_{j^*}) \geq \tilde{\epsilon}/t$ and $D_2(U_{j\text{-}*}) \notin [1-\tilde{\epsilon}, 1+\tilde{\epsilon}]D_1(U_{j^*})$. Due to the lower bound on $D_1(U_{j^*})$ we have that $\hat{w}_{j^*}^{(1)} \in [1-\tilde{\epsilon}/8, 1+\tilde{\epsilon}/8]D_1(U_{j^*})$, so that in particular $\hat{w}_{j^*}^{(1)} > 3\tilde{\epsilon}/(4t)$. As for $\hat{w}_{j^*}^{(2)}$, either $\hat{w}_{j^*}^{(2)} < (1-\tilde{\epsilon})(1+\tilde{\epsilon}/8)D_1(U_{j^*})$ (this holds both when $D_2(U_{j^*}) \geq \tilde{\epsilon}/(4t)$ and when $D_2(U_{j^*}) < \tilde{\epsilon}/(4t)$) or $\hat{w}_{j^*}^{(2)} > (1+\tilde{\epsilon})(1-\tilde{\epsilon}/10)D_1(U_{j^*})$. In either (sub)case $\hat{w}_{j^*}^{(2)}/\hat{w}_{j^*}^{(1)} \notin [1-\tilde{\epsilon}/2, 1+\tilde{\epsilon}/2]$, causing the algorithm to reject in (the second part of) Line 3-g.

   (b) In the second case, $D_1(U_{j^*}) < \tilde{\epsilon}/t$ and $D_2(U_{j^*}) > 2\tilde{\epsilon}/t$. Due to the lower bound on $D_2(U_{j^*})$ we have that $\hat{w}_{j^*}^{(2)} \geq (1-\tilde{\epsilon}/10)D_2(U_{j^*}) > (1-\tilde{\epsilon}/10)(2\tilde{\epsilon}/t)$, so that in particular $\hat{w}_{j^*}^{(2)} > (3\tilde{\epsilon}/(2t))$. As for $\hat{w}_{j^*}^{(1)}$, if $D_1(U_{j^*}) \leq \tilde{\epsilon}/(2t)$, then $\hat{w}_{j^*}^{(1)} \leq 3\tilde{\epsilon}/(4t)$, causing the algorithm to reject in (the first part of) Line 3-g. If $\tilde{\epsilon}/(2t) < D_1(U_{j^*}) \leq \tilde{\epsilon}/t$, then $\hat{w}_{j^*}^{(1)} \in [1-\tilde{\epsilon}/8, 1+\tilde{\epsilon}/8]D_1(U_{j^*}) \leq (1+\tilde{\epsilon}/8)(\tilde{\epsilon}/t)$, so that $\hat{w}_{j^*}^{(2)}/\hat{w}_{j^*}^{(1)} \geq \frac{(1-\tilde{\epsilon}/10)(2\tilde{\epsilon}/t)}{(1+\tilde{\epsilon}/8)\tilde{\epsilon}/t} > (1+\tilde{\epsilon}/2)$, causing the algorithm to reject in (either the first or second part of) Line 3-g.

2. If Item 2 holds, then by the choice of the size of S_1, which is $\Theta(t/\tilde{\epsilon}^2)$, with probability at least $99/100$, the sample S_1 will contain a point i for which $\frac{D_2(i)}{D_2(r_{j^*})} \notin [1/(1+\alpha_{j^*}+\tilde{\epsilon}), 1+\alpha_{j^*}+\tilde{\epsilon}]$, and by E_3 this will be detected in Line 3-h.

3. Similarly, if Item 3 holds, then by the choice of the size of S_2, with probability at least $99/100$, the sample S_2 will contain a point i for which $\frac{D_2(i)}{D_2(r_{j^*})} \notin [1/(1+\alpha_{j^*}+\tilde{\epsilon}), 1+\alpha_{j^*}+\tilde{\epsilon}]$, and by E_3 this will be detected in Line 3-h.

The theorem is thus established. ■

5.2 An approach based on simulating EVAL

In this subsection we present an alternate approach for testing whether two unknown distributions D_1, D_2 are identical versus ϵ-far. We prove the following theorem:

Theorem 7 COND-TEST-EQUALITY-UNKNOWN *is a*

$$\tilde{O}\left(\frac{(\log N)^5}{\epsilon^4}\right)$$

-query algorithm with the following properties: given COND_{D_1}, COND_{D_2} *oracles for any two distributions* D_1, D_2 *over* $[N]$*, it outputs* ACCEPT *with probability at least* $2/3$ *if* $D_1 = D_2$ *and outputs* REJECT *with probability at least* $2/3$ *if* $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$.

At the heart of this result is the efficient simulation of an "approximate EVAL_D oracle" using a COND_D oracle given by Theorem 1. (Recall that an EVAL_D oracle is an oracle which, given as input an element $i \in [N]$, outputs the numerical value $D(i)$.) We feel that this efficient simulation of an approximate EVAL oracle using a COND oracle is of independent interest since it sheds light on the relative power of the COND and EVAL models.

In more detail, our approach to prove Theorem 7 is based on a simple algorithm from [25] that uses an EVAL_D oracle to test equality between D and a known distribution D^*. We show (see Theorem 8) that

a modified version of the algorithm, which uses a SAMP oracle and an "approximate" EVAL oracle, can be used to efficiently test equality between two unknown distributions D_1 and D_2. Theorem 7 follows straightforwardly by combining Theorems 8 and 1.

5.2.1 Testing equality between D_1 and D_2 using an approximate EVAL oracle.

We now show how an approximate $\mathsf{EVAL}_{\mathsf{D}_1}$ oracle, an approximate $\mathsf{EVAL}_{\mathsf{D}_2}$ oracle, and a $\mathsf{SAMP}_{\mathsf{D}_1}$ oracle can be used together to test whether $D_1 = D_2$ versus $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$. As mentioned earlier, the approach is a simple extension of the EVAL algorithm given in Observation 24 of [25].

Theorem 8 *Let* ORACLE_1 *be an* $(\epsilon/100, \epsilon/100)$*-approximate* $\mathsf{EVAL}_{\mathsf{D}_1}$ *simulator and let* ORACLE_2 *be an* $(\epsilon/100, \epsilon/100)$*-approximate* $\mathsf{EVAL}_{\mathsf{D}_2}$ *simulator. There is an algorithm* TEST-EQUALITY-UNKNOWN *with the following properties: for any distributions* D_1, D_2 *over* $[N]$*, algorithm* TEST-EQUALITY-UNKNOWN *makes* $O(1/\epsilon)$ *queries to* ORACLE_1*,* ORACLE_2 *and* $\mathsf{SAMP}_{\mathsf{D}_1}$*, and it outputs* ACCEPT *with probability at least* $7/10$ *if* $D_1 = D_2$ *and outputs* REJECT *with probability at least* $7/10$ *if* $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$*.*

Algorithm 9: TEST-EQUALITY-UNKNOWN

Input: query access to ORACLE_1, to ORACLE_2, and access to $\mathsf{SAMP}_{\mathsf{D}_1}$ oracle

1: Call the $\mathsf{SAMP}_{\mathsf{D}_1}$ oracle $m = 5/\epsilon$ times to obtain points $h_1, \ldots, h_m$ distributed according to D_1.
2: Call the $\mathsf{SAMP}_{\mathsf{D}_2}$ oracle $m = 5/\epsilon$ times to obtain points $h_{m+1}, \ldots, h_{2m}$ distributed according to D_2.
3: **for** $j = 1$ to $2m$ **do**
4: Call $\mathsf{ORACLE}_1(\mathsf{h_j})$. If it returns UNKNOWN then output REJECT, otherwise let $v_{1,i} \in [0, 1]$ be the value it outputs.
5: Call $\mathsf{ORACLE}_2(\mathsf{h_j})$. If it returns UNKNOWN then output REJECT, otherwise let $v_{2,i} \in [0, 1]$ be the value it outputs.
6: **if** $v_{1,j} \notin [1 - \epsilon/8, 1 + \epsilon/8]v_{2,j}$ **then**
7: output REJECT and exit
8: **end if**
9: **end for**
10: output ACCEPT

It is clear that TEST-EQUALITY-UNKNOWN makes $O(1/\epsilon)$ queries as claimed. To prove Theorem 8 we argue completeness and soundness below.

Completeness: Suppose that $D_1 = D_2$. Since ORACLE_1 is an $(\epsilon/100, \epsilon/100)$-approximate $\mathsf{EVAL}_{\mathsf{D}_1}$ simulator, the probability that any of the $2m = 10/\epsilon$ points $h_1, \ldots, h_{2m}$ drawn in Lines 1 and 2 lies in $S^{(\epsilon/100, D_1)}$ is at most $1/10$. Going forth, let us assume that all points h_i indeed lie outside $S^{(\epsilon/100, D_1)}$. Then for each execution of Line 4 we have that with probability at least $1 - \epsilon/100$ the call to $\mathsf{ORACLE}(\mathsf{h_i})$ yields a value $v_{1,i}$ satisfying $v_{1,i} \in [1 - \frac{\epsilon}{100}, 1 + \frac{\epsilon}{100}]D_1(i)$. The same holds for each execution of Line 5. Since there are $20/\epsilon$ total executions of Lines 4 and 5, with overall probability at least 7/10 we have that each $1 \leq j \leq m$ has $v_{1,j}, v_{2,j} \in [1 - \frac{\epsilon}{100}, 1 + \frac{\epsilon}{100}]D_1(i)$. If this is the case then $v_{1,j}, v_{2,j}$ pass the check in Line 6, and thus the algorithm outputs ACCEPT with overall probability at least $7/10$.

Soundness: Now suppose that $d_{\mathrm{TV}}(D_1, D_2) \geq \epsilon$. Let us say that $i \in [N]$ is *good* if $D_1(i) \in [1 - \epsilon/5, 1 + \epsilon/5]D_2(i)$. Let BAD $\subseteq [N]$ denote the set of all $i \in [N]$ that are not good. We have

$$2d_{\mathrm{TV}}(D_1, D_2) = \sum_{i \text{ is good}} |D_1(i) - D_2(i)| + \sum_{i \text{ is bad}} |D_1(i) - D_2(i)| \geq 2\epsilon.$$

Since

$$\sum_{i \text{ is good}} |D_1(i) - D_2(i)| \leq \sum_{i \text{ is good}} \frac{\epsilon}{5}|D_2(i)| \leq \frac{\epsilon}{5},$$

we have

$$\sum_{i \text{ is bad}} (|D_1(i)| + |D_2(i)|) \geq \sum_{i \text{ is bad}} |D_1(i) - D_2(i)| \geq \frac{9}{5}\epsilon.$$

Consequently it must be the case that either $D_1(\text{BAD}) \geq \frac{9}{10}\epsilon$ or $D_2(\text{BAD}) \geq \frac{9}{10}\epsilon$. For the rest of the argument we suppose that $D_1(\text{BAD}) \geq \frac{9}{10}\epsilon$ (by the symmetry of the algorithm, an identical argument to the one we give below but with the roles of D_1 and D_2 flipped throughout handles the other case).

Since $D_1(\text{BAD}) \geq \frac{9}{10}\epsilon$, a simple calculation shows that with probability at least $98/100$ at least one of the $5/\epsilon$ points $h_1, \ldots, h_m$ drawn in Line 1 belongs to BAD. For the rest of the argument we suppose that indeed (at least) one of these points is in BAD; let h_{i^*} be such a point. Now consider the execution of Line 4 when ORACLE_1 is called on h_{i^*}. By Definition 3, whether or not i^* belongs to $S^{(\epsilon/100, D_1)}$, with probability at least $1 - \epsilon/100$ the call to ORACLE_1 either causes TEST-EQUALITY-UNKNOWN to REJECT in Line 4 (because ORACLE_1 returns UNKNOWN) or it returns a value $v_{1,i^*} \in [1 - \frac{\epsilon}{100}, 1 + \frac{\epsilon}{100}]D_1(i^*)$. We may suppose that it returns a value $v_{1,i^*} \in [1 - \frac{\epsilon}{100}, 1 + \frac{\epsilon}{100}]D_1(i^*)$. Similarly, in the execution of Line 5 when ORACLE_2 is called on h_{i^*}, whether or not i^* belongs to $S^{(\epsilon/100, D_2)}$, with probability at least $1 - \epsilon/100$ the call to ORACLE_2 either causes TEST-EQUALITY-UNKNOWN to reject in Line 5 or it returns a value $v_{2,i^*} \in [1 - \frac{\epsilon}{100}, 1 + \frac{\epsilon}{100}]D_2(i^*)$. We may suppose that it returns a value $v_{2,i^*} \in [1 - \frac{\epsilon}{100}, 1 + \frac{\epsilon}{100}]D_2(i^*)$. But recalling that $i^* \in \text{BAD}$, an easy calculation shows that the values v_{1,i^*} and v_{2,i^*} must be multiplicatively far enough from each other that the algorithm will output REJECT in Line 7. Thus with overall probability at least $96/100$ the algorithm outputs REJECT. ■

References

[1] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating the entropy. *SICOMP*, 35(1):132–150, 2005.

[2] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proceedings of FOCS*, pages 442–451, 2001.

[3] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *Proceedings of FOCS*, pages 189–197, 2000.

[4] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing closeness of discrete distributions. Technical Report abs/1009.5397, 2010. This is a long version of [3].

[5] T. Batu, R. Kumar, and R. Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of STOC*, pages 381–390, 2004.

[6] A. Bhattacharyya, E. Fischer, R. Rubinfeld, and P. Valiant. Testing monotonicity of distributions over general partial orders. In *Proceedings of ITCS*, pages 239–252, 2011.

[7] C. Canonne, D. Ron, and R. Servedio. Testing probability distributions using conditional samples. Technical Report http://arxiv.org/abs/1211.2664, 12 Nov 2012.

[8] S. Chakraborty, E. Fischer, Y. Goldhirsh, and A. Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of ITCS*, 2013. Arxiv posting http://arxiv.org/abs/1210.8338 31 Oct 2012.

[9] C. Daskalakis, I. Diakonikolas, R. Servedio, G. Valiant, and P. Valiant. Testing k-modal distributions: Optimal algorithms via reductions. In *Proceedings of SODA*, 2013.

[10] D. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, Cambridge, 2009.

[11] E. Fischer. The art of uninformed decisions: A primer to property testing. *BEATCS*, 75:97–126, 2001.

[12] O. Goldreich, editor. *Property Testing: Current Research and Surveys*. Springer, 2010. LNCS 6390.

[13] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998.

[14] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. Technical Report TR00-020, ECCC, 2000.

[15] P. Indyk, R. Levi, and R. Rubinfeld. Approximating and Testing k-Histogram Distributions in Sub-linear Time. In *Proceedings of PODS*, pages 15–22, 2012.

[16] S. K. Ma. Calculation of entropy from data of motion. *J. Stat. Phys.*, 26(2), 1981.

[17] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[18] Jerzy Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[19] L. Paninski. Estimating entropy on m bins given fewer than m samples. *IEEE-IT*, 50(9):2200–2203, 2004.

[20] L. Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE-IT*, 54(10):4750–4755, 2008.

[21] S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bonds for approximating distributions support size and the distinct elements problem. *SICOMP*, 39(3):813–842, 2009.

[22] L. Reyzin. Extractors and the leftover hash lemma. http://www.cs.bu.edu/ reyzin/teaching/s11cs937/notes-leo-1.pdf, March 2011.

[23] D. Ron. Property Testing: A Learning Theory Perspective. *FnTML*, 1(3):307–402, 2008.

[24] D. Ron. Algorithmic and analysis techniques in property testing. *FnTCS*, 5:73–205, 2010.

[25] R. Rubinfeld and R. A. Servedio. Testing monotone high-dimensional distributions. *RSA*, 34(1):24–44, January 2009.

[26] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SICOMP*, 25(2):252–271, 1996.

[27] G. Valiant and P. Valiant. A CLT and tight lower bounds for estimating entropy. Technical Report TR10-179, ECCC, 2010.

[28] G. Valiant and P. Valiant. Estimating the unseen: A sublinear-sample canonical estimator of distributions. Technical Report TR10-180, ECCC, 2010.

[29] G. Valiant and P. Valiant. Estimating the unseen: an $n/\log(n)$-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of STOC*, pages 685–694, 2011. See also [27] and [28].

[30] P. Valiant. Testing symmetric properties of distributions. *SICOMP*, 40(6):1927–1968, 2011.

[31] Wikipedia contributors. Stratified Sampling. http://en.wikipedia.org/wiki/Stratified_sampling, accessed July 1, 2013.

A Useful tools from probability

On several occasions we will use the *data processing inequality for variation distance*. This fundamental result says that for any two distributions D, D', applying any (possibly randomized) function to D and D' can never increase their statistical distance; see e.g. part (iv) of Lemma 2 of [22] for a proof of this lemma.

Lemma 24 (Data Processing Inequality for Total Variation Distance) *Let D, D' be two distributions over a domain Ω. Fix any randomized function*[10] *F on Ω, and let $F(D)$ be the distribution such that*

[10] Which can be seen as a distribution over functions over Ω.

a draw from $F(D)$ is obtained by drawing independently x from D and f from F and then outputting $f(x)$ (likewise for $F(D')$). Then we have

$$d_{\mathrm{TV}}(F(D), F(D')) \leq d_{\mathrm{TV}}(D, D').$$

We next give several variants of Chernoff bounds (see e.g. Chapter 4 of [17]).

Theorem 9 *Let $Y_1, \ldots, Y_m$ be m independent random variables that take on values in $[0,1]$, where $\mathrm{E}[Y_i] = p_i$, and $\sum_{i=1}^m p_i = P$. For any $\gamma \in (0,1]$ we have*

$$\textit{(additive bound)} \qquad \Pr\left[\sum_{i=1}^m Y_i > P + \gamma m\right], \Pr\left[\sum_{i=1}^m Y_i < P - \gamma m\right] \leq \exp(-2\gamma^2 m) \tag{48}$$

$$\textit{(multiplicative bound)} \qquad \Pr\left[\sum_{i=1}^m Y_i > (1+\gamma)P\right] < \exp(-\gamma^2 P/3) \tag{49}$$

and

$$\textit{(multiplicative bound)} \qquad \Pr\left[\sum_{i=1}^m Y_i < (1-\gamma)P\right] < \exp(-\gamma^2 P/2). \tag{50}$$

The bound in Equation (49) is derived from the following more general bound, which holds from any $\gamma > 0$:

$$\Pr\left[\sum_{i=1}^m Y_i > (1+\gamma)P\right] \leq \left(\frac{e^\gamma}{(1+\gamma)^{1+\gamma}}\right)^P, \tag{51}$$

and which also implies that for any $B > 2eP$,

$$\Pr\left[\sum_{i=1}^m Y_i > B\right] \leq 2^{-B}. \tag{52}$$

The following extension of the multiplicative bound is useful when we only have upper and/or lower bounds on P (see Exercise 1.1 of [10]):

Corollary 10 *In the setting of Theorem 9 suppose that $P_L \leq P \leq P_H$. Then for any $\gamma \in (0,1]$, we have*

$$\Pr\left[\sum_{i=1}^m Y_i > (1+\gamma)P_H\right] < \exp(-\gamma^2 P_H/3) \tag{53}$$

$$\Pr\left[\sum_{i=1}^m Y_i < (1-\gamma)P_L\right] < \exp(-\gamma^2 P_L/2) \tag{54}$$

We will also use the following corollary of Theorem 9:

Corollary 11 *Let $0 \leq w_1, \ldots, w_m \in \mathbb{R}$ be such that $w_i \leq \kappa$ for all $i \in [m]$ where $\kappa \in (0,1]$. Let $X_1, \ldots, X_m$ be i.i.d. Bernoulli random variables with $\Pr[X_i = 1] = 1/2$ for all i, and let $X = \sum_{i=1}^m w_i X_i$ and $W = \sum_{i=1}^m w_i$. For any $\gamma \in (0,1]$,*

$$\Pr\left[X > (1+\gamma)\frac{W}{2}\right] < \exp\left(-\gamma^2 \frac{W}{6\kappa}\right) \textit{ and } \Pr\left[X < (1-\gamma)\frac{W}{2}\right] < \exp\left(-\gamma^2 \frac{W}{4\kappa}\right),$$

and for any $B > e \cdot W$,

$$\Pr[X > B] < 2^{-B/\kappa}.$$

Proof: Let $w'_i = w_i/\kappa$ (so that $w'_i \in [0,1]$), let $W' = \sum_{i=1}^m w'_i = W/\kappa$, and for each $i \in [m]$ let $Y_i = w'_i X_i$, so that Y_i takes on values in $[0,1]$ and $\mathrm{E}[Y_i] = w'_i/2$. Let $X' = \sum_{i=1}^m w'_i X_i = \sum_{i=1}^m Y_i$, so that $\mathrm{E}[X'] = W'/2$. By the definitions of W' and X' and by Equation (49), for any $\gamma \in (0,1]$,

$$\Pr\left[X > (1+\gamma)\frac{W}{2}\right] = \Pr\left[X' > (1+\gamma)\frac{W'}{2}\right] < \exp\left(-\gamma^2\frac{W'}{6}\right) = \exp\left(-\gamma^2\frac{W}{6\kappa}\right), \tag{55}$$

and similarly by Equation (50)

$$\Pr\left[X < (1-\gamma)\frac{W}{2}\right] < \exp\left(-\gamma^2\frac{W}{4\kappa}\right) . \tag{56}$$

For $B > e \cdot W = 2e \cdot W/2$ we apply Equation (52) and get

$$\Pr\left[X > B\right] = \Pr\left[X' > B/\kappa\right] < 2^{-B/\kappa}, \tag{57}$$

as claimed. ■