SUBLINEAR ALGORITHMS FOR STATISTICS, MARKOV CHAIN AND BINPACKING PROBLEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Patrick Edward White

December 2019

SUBLINEAR ALGORITHMS FOR STATISTICS, MARKOV CHAIN AND BINPACKING PROBLEMS

Patrick Edward White, Ph.D.

Cornell University 2019

We consider the problem of how to construct algorithms which deal efficiently with large amounts of data. We give new algorithms which use time and communication resources that are sublinear in the problem size for problems in various domains including statistics and combinatorics.

We first consider properties of random variables. We begin with the problem of distinguishing whether two distributions over the same domain are close or far in both the L_1 and the L_2 norms. We investigate two models for representing a distribution. In one model, elements of a sample space are generated on request according to a fixed but unknown distribution. In the other, the probability assigned to each element is given explicitly in an array. We present algorithms in two settings: (1) when both distributions are represented in the first model; and, (2) when one of each representation is given. We show that the first setting is provably easier than the second setting. Next we give algorithms for testing whether two random variables are independent. In all of our algorithms, the number of samples required from the input distributions is sublinear in the domain size and nearly optimal.

We then consider properties of data. Specifically, we give an algorithm which determines if a Markov Chain is rapidly mixing in sublinear time, assuming the input is in a form which allows for easy generation of sequential nodes in a random walk. Our test distinguishes Markov chains which are rapidly mixing from those which cannot be made rapidly mixing by changing a small number of edges.

Finally we turn to a model in which the help of an untrusted entity is used to reliably solve a problem in sublinear time. For the problem of multidimensional bin-packing, we give an algorithm which can verify the goodness of a potential solution in sublinear time. To do this we develop tools which allow one to test that a function is approximately monotone.

BIOGRAPHICAL SKETCH

Patrick White grew up in the small town of Somerset, Kentucky. He attended public school in the Somerset Independent School district, graduating in 1992. He attended college as a Harold Stirling Vanderbilt Scholar at Vanderbilt University in Nashville, Tennessee, where he studied mathematics, computer science and piano performance, graduated *summa cum laude* and received his Bachelor of Science degree in May, 1996. In August of the same year he entered the College of Engineering at Cornell University and studied under Ronitt Rubinfeld in the Department of Computer Science. He received his Master's of Science degree in May, 2000 and his Ph.D. quite some time later, in 2019. He now resides in Leesburg, Virginia with his wife, Allyson and son, Nathaniel.

To Allyson

ACKNOWLEDGEMENTS

It is an unusual position to be in, submitting a dissertation so many years after the results contained within were first polished and published, because I no longer have to wonder if they will have an impact on the research community – if the theorems will flourish or flounder. I am quite aware that, luckily, a number of the results we obtained here have been regarded by some as worthy of continued study, and I couldn't be happier. But I am even more acutely aware that the only reason the questions we answered have been judged so kindly by our colleagues is because my advisor, Ronitt, has an astounding gift for knowing the right questions to ask.

Her name is, of course, attached to a bounty of results and papers that launched entire fields of research. My partner, Tuğkan, and I were just fortunate to be around when she, having read an unpublished manuscript about a conjecture related to expander graphs, noted that, just maybe, there was a tip-of-the-iceberg result hidden within, and if we just pointed our pick axes in the right direction and started digging, we might strike gold. She was, once again, exactly right. The field of Distribution Testing started to take shape and is yielding new results still today.

And while brilliance might suffice to produce a successful mathematician, it does not alone produce an advisor and mentor. Ronitt's compassion, generosity, kindness and humility are truly peerless among the faculty at the world's top universities. She always gives credit to her colleagues and companions, never seeks the spotlight, and yet, when necessary, asserts her correctness over your errors so pleasantly that you feel delighted to have been wrong. Her demeanor was just what I needed at Cornell, and was absolutely fundamental to my completion of this body of work.

Credit also goes to the (now-disbanded) algorithms research group at NEC Princeton, who, unencumbered by teaching loads and faculty meetings, were able to gather around a whiteboard and do research all day long, for days on end. It was here where I

felt most sincerely the true joy of pure research, enhanced by occasional sand-volleyball games and fueled by the famous blueberry pancakes of the Novotel Princeton.

Of course my teachers and especially my dissertation committee at Cornell are truly appreciated for the time they take designing challenging courses, supporting graduate student research and for the time they volunteer reading and reviewing my work here. They definitely deserve more sunshine than Ithaca is willing to let break through the clouds.

Beyond my academic colleagues, I would not have been able to complete this document without the continued support of my parents, Judy (*in memoriam*) and Bill White, my patient wife, Allyson, and my custom-minifig-crafting son, Nathaniel[1]. And finally, thanks to Jan, for helping me find myself after so many years of being lost.

To all who ponder the results contained herein, thank you for keeping the field alive, and for adding your part to what we all are creating.

Leesburg, VA

[1] *a.k.a.* epsilon-squirt

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In a peculiarly memorable episode of the 1970's American sit-com *Good Times*, student J.J. Evans advises a friend on how to quickly read any assigned novel. Demonstrating, he holds up a sample book, breaks the spine and leisurely fans through the pages in about three seconds. As he closes the book and nods in satisfaction, his friend, amazed, asks "Did you read that whole book that fast?" J.J. coolly replies "Nah. But when the teacher asks me tomorrow if I read it, I'll tell her that I laid an eye on every page."

J.J., in his own way, was prophetically describing a common sentiment which underlies the motivation for the field of sublinear algorithms: sometimes it just takes too long to read all the input. There are various motivations for algorithms which do not process all of their input. One obvious case is massive data sets, which are becoming increasingly common as computer processing power and storage space increase in efficiency. Vast data sets exist such as those used in the human genome project and the SETI cosmic background radiation project. With storage requirements ranging into terabytes, any algorithm which processes this data may find even linear time prohibitive. Moreover, data sets also exist which cannot be easily accessed or even stored, or are constantly changing. For example, the web graph (defined by the pages on the Internet and links between them) exists but is neither fixed nor written down in any one place. Any algorithm which takes the web graph as input can only use partial information.

Another motivation for sublinear algorithms arises in the field of result checking and verification. It is possible in a network for a client to have a computation performed remotely and the answer returned to the client. This may be desirable when a remote server has computational resources exceeding the client's, or a capability which the client does not have. Why, though, should the client trust the answer? The server could provide, along with the result, a transcript of the computations required, but verifying

that transcript would take as much time as performing the computation to begin with. It would be preferable if some way existed for the server to provide a certificate which could be quickly verified by the client who would then be able to trust the results of the computation.

While it may appear that sublinear algorithms must, by necessity, yield only approximate results, there are examples of problems that can be solved in deterministic sublinear time, in particular when one assumes the input comes in a special form. For example the classic binary search problem achieves sublinear running time on input which is a sorted list. Nevertheless, all of the algorithms given in this paper do exploit randomization and approximation to achieve their low time complexities.

This paper investigates sublinear algorithms in three arenas: testing properties of distributions, testing properties of data, and testing properties with untrusted help. When speaking of "sublinear" algorithms it is necessary to distinguish what kind of sublinearity is exhibited. We consider two types of sublinearity in this work: sublinear sample complexity and sublinear time complexity. An algorithm has sublinear time complexity if it is has time complexity $o(n)$ and sublinear sample complexity if it has sample complexity $o(n)$. All of the algorithms we give have sublinear sample complexity while many also have sublinear time complexity.

In testing properties of distributions, we look at the problems of testing closeness of two distributions and independence of random variables in Chapter 2. We consider two models in which distributions can be represented. The first model is a black-box distribution, that is, we assume we are given access to an oracle which outputs elements of a sample space according to a fixed but unknown probability function. Given two such oracles we show how to determine if their underlying distributions are close using only the samples which they output. We give tests which determine closeness in both the L_2 (Euclidean) norm (Section 2.1) and the L_1 (statistical) norm (Section 2.2), with

sample complexities on the order of $O(1)$ and $\tilde{O}(n^{2/3})$ respectively, over a sample space of n elements.

In the second model we consider an explicit distribution, in which the values a probability function assigns to the elements of a sample space are given directly in an array. We give in Section 2.4 an algorithm which accepts black boxes which output a distribution that is close to the explicitly given one and reject black boxes which output a distribution that is far from the explicit distribution. Our test has sample complexity $\tilde{O}(n^{1/2})$.

In Section 2.6 we consider the problem of testing independence of random variables. We define ϵ-independence, which is a quantification of the dependence of two random variables similar to the covariance. Given black-box distributions as input, we give an algorithm which accepts distributions which are independent and rejects distributions which are not ϵ-independent. If the distributions are over sets of size n and m, with $n \geq m$, then the sample complexity of our algorithm is $\tilde{O}(n^{2/3}m^{1/3})$.

In the domain of testing properties of data, we investigate in Chapter 3 the problem of testing for mixing properties of Markov chains. A Markov chain is said to be *rapidly-mixing* if it reaches a stationary distribution in logarithmically many iterations. We give in Section 3.3 a testing algorithm which samples random walks on the Markov chain and passes any Markov chain which is rapidly mixing, while rejecting any Markov chain which cannot be made rapidly mixing by slightly modifying its transition matrix. The time complexity of this test is sublinear in the number of edges in the Markov chain.

For testing with untrusted help, we give in Chapter 4 a sublinear algorithm for verifying the solution to a multidimensional bin packing problem. We give a protocol by which a remote untrusted host can certify a proposed solution of a multidimensional bin packing problem. The host provides along with a valid packing a table corresponding to a function which will be monotone if and only if the packing is valid. We then give

a test for monotonicity of a function over $\{1, \ldots, n\}^d$. This test samples an input function at several locations and passes any function which is monotone, while rejecting any function which cannot be made monotone by changing only a few of its values. Both the monotonicity tester and the verification algorithm we give for binpacking have time complexity sublinear in the size of their inputs.

Relationship to Property Testing The property testing model was introduced by Rubinfeld and Sudan [RS96] and Goldreich and Ron [GGR96]. In this model decision procedures are allowed to misclassify some inputs, where the probability and extent of possible misclassifications are bounded. Given a domain S with distance function Δ and a property $P \subset S$, a property test for P is an algorithm which, on input x, ϵ, δ will accept x if $x \in P$ and reject x if $\Delta(x, P) > \epsilon$ with probability of failure less than δ. If $0 < \Delta(x, P) \leq \epsilon$ then the behavior of the property testing algorithm is unspecified.

Property testing has proven a fruitful field of research and testing algorithms have been developed in numerous and various areas, including algebraic properties such as linearity and low degree polynomials, graph properties such as bipartiteness, connectivity, diameter, clique; group operations and regular languages, and many others. For results and an extensive bibliography, see the survey paper by Ron [Ron].

Although the algorithms we give in this paper are of a similar spirit, note that not all of them fit the original property testing model. The statistical tests in Chapter 2 test properties of distributions, which are not in themselves fixed inputs. The binpacking verification algorithm given in Chapter 4 is a property test that takes as additional input a proof generated by an untrusted party. We also give new property tests, according to the classic definition, for testing mixing of Markov chains in Chapter 3, and monotonicity in Chapter 4.

Preliminaries and Notation The following notation will be commonly used. The set of all natural numbers less than or equal to n, that is the set $\{1, \ldots, n\}$, is denoted as $[n]$. The notation $x \in_R [n]$ denotes that x is chosen uniformly at random from the set $[n]$. Vectors are assumed to be one-dimensional arrays of real numbers and are denoted by $\vec{v}$. The elements of a vector are denoted with subscripts, as in $\vec{v} = (v_1, \ldots, v_n)$. We employ standard vector norms, which are defined here for reference.

Definition 1 *Given any vector $\vec{v}$ with n elements, the L_1-**norm** of $\vec{v}$ is denoted as $\|\vec{v}\|_1$ and is defined as*

$$\|\vec{v}\|_1 = \sum_{i=1}^{n} |v_i|$$

.

Definition 2 *Given any vector $\vec{v}$ with n elements, the L_2-**norm** of $\vec{v}$ is denoted as $\|\vec{v}\|_2$ and is defined as*

$$\|\vec{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$$

.

Definition 3 *Given any vector $\vec{v}$ with n elements, the L_∞-**norm** of $\vec{v}$ is denoted as $\|\vec{v}\|_\infty$ and is defined as*

$$\|\vec{v}\|_\infty = \max_i |v_i|$$

The cardinality of a set is denoted either by $|S|$ or $\#\{S\}$, where the latter will sometimes be chosen for visual clarity. We also have notations for distributions and operations on them. A probability space is any set $\mathcal{A}$ and a function f which assigns to every element in $\mathcal{A}$ a value in the closed interval $[0, 1]$. Unless otherwise indicated, we restrict ourselves to discrete probability spaces, therefore, our sets can be taken as isomorphic to the set $[n]$ for some n. For us a probability function is usually a function over the set $[n]$ and is correspondingly represented as a vector $\vec{v}$ with elements in $[0, 1]$ with the

property that $\|\vec{v}\|_1 = 1$. Thus in the distribution $\vec{v} = (v_1, \ldots, v_n)$, we define v_k as the probability of selecting element k from the space of n elements.

CHAPTER 2

TESTING PROPERTIES OF DISTRIBUTIONS

An average-size department store needs to analyze recent sales patterns. They want to take a statistical approach to discerning when there is a change in sales trends in order to rethink how they restock their shelves. The number of items sold in a quarter, say, is relatively low when compared to the different types of products that have been kept in inventory. Stated as a statistical problem, then, the number of samples (sales) is small compared to the domain size (product types). In this case, traditional statistical tests may not be able to discover changes in sales trends due to the shortage of available data. Furthermore, since the only way to collect more samples is by selling more items, there is no easy way to increase the amount of data.

In many statistical problems of practical interest, it is highly desirable to minimize the number of samples required to achieve confidence that a given hypothesis or inference is correct. In this chapter we give new tests for several statistical properties which require significantly fewer samples in general than tests used currently in practice. To our knowledge, we give the first algorithms for these properties which require a sample size that is sublinear in the size of sample space. All of the algorithms we give in this chapter have sublinear sample complexity; many have sublinear time complexity as well.

Our Model All of the problems considered in this chapter involve distribution functions over a discrete probability space with n elements, which we usually take without loss of generality to be the set $[n] = \{1, \ldots, n\}$. Our algorithms interact with this probability space using two distinct models: a **Black box** model which generates elements of the sample space according to a fixed, but usually unknown, probability distribution, and an **Explicit** model in which the probability mass function is given explicitly as an array.

Definition 4 *A* **black-box distribution** *is a tuple $\langle \mathcal{A}, f, F \rangle$, where $\mathcal{A}$ is a discrete probability space, f is a density function over $\mathcal{A}$ and F is an oracle. On query to F the oracle emits an element in $\mathcal{A}$ such that* $\Pr[\mathcal{A} \text{ emits } x] = f(x)$.

Definition 5 *An* **explicit distribution** *is a tuple $\langle \mathcal{A}, f \rangle$, where $\mathcal{A}$ is a discrete probability space, f is a density function over $\mathcal{A}$. If $\mathcal{A} = [n]$ then there is a vector $\vec{f}$ whose elements correspond to the probabilities assigned by f such that $\forall i \in \mathcal{A},\ \vec{f_i} = f(i)$.*

Given an explicit distribution $\langle \mathcal{A}, f \rangle$ over n elements and $O(n)$ preprocessing time, one can simulate a black box distribution $\langle \mathcal{A}, f, F \rangle$ in $O(\log n)$ time per query. The preprocessing stage computes a *cumulative distribution* g where $g(x) = \sum_{i=1}^{x} f(i)$. For each query, oracle F now selects $y \in_R [0, 1]$ and by binary search over $[n]$ finds $i \in n$ such that $g(i-1) < y \leq g(i)$ outputs element i.

Our Results In this chapter we consider two kinds of statistical properties. The first property is closeness of distributions under various norms and the second is independence of random variables. For the closeness property we consider two models by which distributions can be represented.

In Section 2.1 we consider whether two distributions, both given as black-box distributions, are close in the L_2 (Euclidean) norm. We give an algorithm which can distinguish close distributions from those which are not close using a constant number of samples, that is, a sample size which does not depend on the size of the underlying domain. This result is a crucial element in the sections that follow.

Next in Section 2.2 we consider whether two distributions, again given as black-box distributions, are close in the L_1 (statistical) norm. We again distinguish close distributions from those which are not close. The number of samples required for a domain of n elements is on the order of $\tilde{O}(n^{2/3})$, in contrast to traditional techniques which require at least a linear number of samples.

Then in Section 2.4 we turn to the explicit distribution model. One might hope that testing a black box distribution against an explicit distribution would be easier than testing two black box distributions. We show that is in fact the case and give an algorithm which distinguishes black box distribution which are close to an explicit distribution from ones which are not by using only $\tilde{O}(n^{1/2})$ samples.

Finally in Section 2.6 we consider the problem of testing the independence of random variables. Given input of two black box distributions which output in tandem (or, equivalently, one black box which outputs pairs) we give an algorithm which accepts if the underlying distributions are independent, and rejects if they are not close to being independent. If the two distributions are over sets of size n and m respectively, then our sample complexity is $\tilde{O}(n^{2/3}m^{1/3})$. We use the results of previous sections in devising our independence testing algorithm.

Prior Work The earliest mention of property testing discrete distributions seems to come from a brief section in [GGR96], who consider distributions over $[0,1]^n$, following the model of Kearns *et al* [KR94] in their study of learning discrete distributions.

The kernel of the idea behind our distribution tests appears in an analysis of expander graphs by Goldreich and Ron [GR00]. They examine methods for approximating the L_2-distance between a given distribution and the uniform distribution to within a factor of $(1+\epsilon)$ in time $O(\sqrt{n})$ for constant ϵ. Their focus on quickly estimating collision probabilities underlies our techniques.

There is much work on the problem of estimating the distance between distributions in data streaming models where space is limited rather than time (cf. [GM99, AMS99, FKSV99]). Another line of work [BCFM00] estimates the distance in frequency count distributions on words between various documents, where again space is limited.

In an interactive setting, Sahai and Vadhan [SV97] show that given distributions p and q, generated by polynomial-size circuits, the problem of distinguishing whether p

and q are close or far in L_1-norm, is complete for statistical zero-knowledge.

There is a vast literature on testing statistical hypotheses. In these works, one is given examples chosen from the same distribution out of two possible choices, say p and q. The goal is to decide which of two distributions the examples are coming from. More generally, the goal can be stated as deciding which of two known classes of distributions contains the distribution generating the examples. This can be seen to be a generalization of our model as follows: Let the first class of distributions be the set of distributions of the form $q \times q$. Let the second class of distributions be the set of distributions of the form $q_1 \times q_2$ where the L_1 difference of q_1 and q_2 is at least ϵ. Then, given examples from two distributions p_1, p_2, create a set of example pairs (x, y) where x is chosen according to p_1 and y according to p_2. Bounds and an optimal algorithm for the general problem for various distance measures are given in [CT91, NP33, CM89, Csi67, Leh86]. None of these give sublinear bounds in the domain size for our problem. The specific model of singleton hypothesis classes is studied by Yamanishi [Yam95].

Subsequent Work Since the publication of the results presented here, a significant amount of literature has been published improving, enhancing and utilizing our results; see for example [BDKR05], [BKR04], [Pan08], [VV11], [VV17], [DDS$^+$13], [ADK15], [Ona09], [BFRV11], [CDGR18], [CDVV14], [DGPP17], [DK16], [DKN15], [Gol16], [LRR13]. The field of *distribution property testing* has grown into an active and established area of research. In addition, applications of these ideas have been found to problems disparate and far ranging, and many new algorithmic and analytic techniques have been discovered. Key contributions will be noted at the end of each section in this document as appropriate. For more information the interested reader is directed to the surveys [Can15], [Rub12], [Gol17].

2.1 Testing closeness in the L_2 norm

We show in this section how to determine if two distributions, each given as a black box, are close in the L_2 (Euclidean) norm. The basis of this test is a technique for counting "collisions" in sample sets from a distribution. We define two notions of collisions: one pertaining to a sample set from one black box, one for two different sample sets from two different black boxes.

Definition 6 *The* **self-collision probability** *of a black box distribution $\langle \mathcal{A}, f, F \rangle$ is the probability, given two independent random samples from F, that the two samples will be identical. Given a sample set S, a* self-collision *is a pair of indices $(i < j)$ such that $S_i = S_j$*

The self-collision probability is equal to the L_2 norm of the probability density function.

Lemma 7 *Given a black box distribution $\langle \mathcal{A}, f, F \rangle$ and a set M of m samples from F, If $i < j$ then $Pr[M_i = M_j] = \|\vec{f}\|_2^2$*

PROOF: $Pr[M_i = M_j] = \sum_{a \in \mathcal{A}} Pr[M_i = a] Pr[M_j = a] = \sum_{a \in \mathcal{A}} f(a)^2 = \|\vec{f}\|_2^2$

Definition 8 *The* **joint-collision probability** *of two black box distributions $\langle \mathcal{A}, f, F \rangle$ and $\langle \mathcal{A}, g, G \rangle$ is the probability, given a random sample from F and a random sample from G, that the two samples will be identical. Given two sample sets S and T, a* joint-collision *is a pair of indices (i, j) such that $S_i = T_j$.*

Lemma 9 *Given two black box distributions $\langle \mathcal{A}, f, F \rangle$ and $\langle \mathcal{A}, g, G \rangle$ and given a random sample s from F and a random sample t from G, then $Pr[s = t] = \sum_{a \in \mathcal{A}} f(a) g(a)$.*

PROOF: $Pr[x = y] = \sum_{a \in \mathcal{A}} Pr[x = a] Pr[y = a] = \sum_{a \in \mathcal{A}} f(a) g(a) = \vec{f} \cdot \vec{g}$

Counting the number of collisions in two sample sets from the same distribution yields a statistic that is an efficient estimator for the L_2 norm of the distribution.

Lemma 10 *Given a black box distribution $\langle \mathcal{A}, f, F \rangle$ and a set M of m samples from F, let X be the number of self-collisions in M. That is, $X = \#\{(i,j) | i < j, M_i = M_j\}$. Then* $\mathrm{E}[X] = \binom{m}{2} \|\vec{f}\|_2^2$.

PROOF: Let $\chi_{i,j}$ be the indicator variable which equals 1 if and only if $M_i = M_j$ and 0 otherwise. Then $X = \sum_{i<j} \chi_{i,j}$. So

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i<j}^{m} \chi_{i,j}\right] = \sum_{i<j}^{m} \mathrm{E}[\chi_{i,j}] = \binom{m}{2} \sum_{a \in \mathcal{A}} f(a)^2 = \binom{m}{2} \|\vec{f}\|_2^2$$

The efficiency of the self-collision statistic as an unbiased estimator for the L_2 norm depends on the variance of the statistic. This was calculated in [GR00] and shown to yield an efficient estimator.

Lemma 11 (adapted from [GR00]) *Given a black box distribution $\langle \mathcal{A}, f, F \rangle$ and a set M of m samples from F, let X be the number of self-collisions in M. Then* $\mathrm{Var}[X] \leq \left\{ \binom{m}{2} \|f\|_2^2 \right\}^{3/2}$. *Moreover if $b = \|\vec{f}\|_\infty$ then there is a constant c such that* $\mathrm{Var}[X] \leq c(m^2 b + m^3 b^2)$.

An application of this Lemma is an efficient estimator for the L_2 norm of a given distribution.

Theorem 12 (adapted from [GR00]) *Given a black-box distribution $\langle \mathcal{A}, f, F \rangle$, there is a test using $O(\sqrt{|\mathcal{A}|}/\epsilon^2 \log(1/\delta))$ queries that estimates $\|\vec{f}\|_2^2$ to within a factor of $(1 \pm \epsilon)$, with probability at least $1 - \delta$.*

PROOF: Let $n = |\mathcal{A}|$. By Chebyshev,

$$\Pr[|X - E[X]| > \epsilon E[X]] < \frac{\mathrm{Var}[X]}{(\epsilon E[X])^2} \leq \frac{1}{\epsilon^2 m \|\vec{f}\|_2}$$

Since $\|\vec{f}\|_2 > \frac{1}{\sqrt{n}}$, this probability is less than $1/3$ when $m = \Omega(\sqrt{n}/\epsilon^2)$.[1] ■

We extend the techniques used in [GR00] to include joint-collision probabilities. An analysis of the variance of the joint-collision statistic produces an efficient algorithm for deciding if two different black-box distributions are close.

It is convenient to define two operators as a shorthand for counting the number of collisions in a sample from a black box distribution. One operator is used to count self-collisions and the other is used for joint-collisions.

Definition 13 *Given a black box distribution $B = \langle \mathcal{A}, f, F \rangle$ and a positive integer n, the operator* `Self-Col`(B, n) *signifies the number of self-collisions in a set of n independent samples drawn according to F.*

Given a vector of the frequencies, $\vec{v}$, of each element in an observed sample set from a sample space $\mathcal{A}$, $\vec{v} = \langle c_{a_1}, c_{a_2}, c_{a_3}, \ldots, c_{a_n} \rangle$, the self-collision count is equal to $\vec{v} \cdot \vec{v} - \vec{v}$. If $\vec{v}$ is sparse this can be computed in $O(k)$ time where k is the number of non-zero elements in $\vec{v}$.

Definition 14 *Given two black box distributions $B^f = \langle \mathcal{A}, f, F \rangle$ and $B^g = \langle \mathcal{A}, g, G \rangle$, and a positive integer n, the operator* `Joint-Col`(B^f, B^g, n) *signifies the number of joint-collisions in a set of n independent samples drawn according to F with a set of n independent samples drawn according to G.*

The joint collision count can similarly be computed as $\vec{v}_1 \cdot \vec{v}_2$, given two frequency vectors, in $O(k)$ time where k is the number of observed elements in either set.

[1] By using Cauchy-Schwarz, this result yields an $O(\sqrt{n}/\epsilon^4)$-query tester for ϵ-closeness to uniformity for constant ϵ in the L_1 norm. Perhaps suprisingly, it was eventually shown in [DGPP17] that these types of collision-testers are actually sample-optimal, using on a tighter analysis of the variance than we give here, and can achieve constant error with only $O(\sqrt{n}/\epsilon^2)$ samples.

2.1.1 L_2 distance between distributions

Using these techniques we give our first result on testing the closeness of two distributions.

Theorem 15 *Given parameter δ, and black-box distributions $\langle \mathcal{A}, f, F\rangle$ and $\langle \mathcal{A}, g, G\rangle$ there exists a test such that if $\|f-g\|_2 \leq \epsilon/2$ then the test passes with probability at least $1-\delta$. If $\|f-g\|_2 > \epsilon$ then the test passes with probability less than δ. The running time of the test is $O(\epsilon^{-4}\log\frac{1}{\delta})$.*

L_2**-Distance-Test** $(\langle \mathcal{A}, f, F\rangle, \langle \mathcal{A}, g, G\rangle, \epsilon, \delta)$
Let $m = O(\epsilon^{-4})$
Repeat $O(\log(\frac{1}{\delta}))$ times
  Let r_p = Self-Col(F, m)
  Let r_q = Self-Col(G, m)
  Let s_{pq} = Joint-Col(F, G, m)
  Let $r = \frac{2m}{m-1}(r_p + r_q)$
  Let $s = 2s_{pq}$
  If $r - s > m^2\epsilon^2/2$ then reject the iteration
End Repeat
Reject if the majority of iterations reject, accept otherwise

Figure 2.1: Algorithm L_2-Distance-Test

The test used to prove Theorem 15 is given in Figure 2.2. If f and g are close, we expect that the self-collision probabilities of either will be close to the joint-collision probability of the pair. We quantify this by re-normalizing r_p, r_q and s_{pq} to r and s and define a new statistic $r-s$ which we expect to be small if f is statistically close to g. We constructed r and s so that $r-s$ has nice properties

Lemma 16 *Let r and s be defined as in algorithm L_2-Distance-Test. Then $\mathrm{E}[r-s] = m^2\|f-g\|_2^2$.*

PROOF: From Lemma 10 and the definition of expectation,

$$\mathrm{E}[r] = \frac{2m}{m-1}(\mathrm{E}[r_p + r_q]) = \frac{2m}{m-1}\frac{m(m-1)}{2}(\|f\|_2^2 + \|g\|_2^2) = m^2(\|f\|_2^2 + \|g\|_2^2)$$

and

$$\mathrm{E}[s] = 2\mathrm{E}[s_{pq}] = 2m^2(f \cdot g)$$

so

$$\mathrm{E}[r - s] = m^2(\|f\|_2^2 - 2(f \cdot g) + \|g\|_2^2) = m^2(\|f - g\|_2^2)$$

■

To prove that our test is correct, we need to bound the variance of $r - s$ to show that it indeed is an efficient statistic. The variance bound is more complicated, and is given in terms of the largest weight in f and g, which we define as b.

Lemma 17 *There is a constant c such that* $\mathrm{Var}[r - s] \leq c(m^3b^2 + m^2b)$*, where* $b = \max(\|f\|_\infty, \|g\|_\infty)$.

PROOF: $\mathrm{Var}[r]$ follows from [GR00], so the work in this proof is bounding $\mathrm{Var}[s]$. Let S be the set $[m] \times [m]$, where m is the number of samples taken as defined in the algorithm. Let Q_F be the set of samples from F taken in the call to `Joint-Col`. Similarly let Q_G be the samples taken from G. For $(i, j) \in S$, define the indicator variable $\chi_{i,j} = 1$ if the i^{th} element of Q_F and the j^{th} element of Q_G are the same. It follows that the variable s_{pq} from the algorithm satisfies $s_{pq} = \sum_{i,j} \chi_{i,j}$. Also note that $f \cdot g = \sum_i f(i)g(i) \leq b\sum_i g(i) = b$

Now

$$\begin{aligned}
\mathrm{Var}\left[s_{pq}\right] &= \mathrm{Var}\left[\sum_{S}\chi_{i,j}\right] \\
&= \sum_{S\times S}\mathrm{Covar}\left[\chi_{i,j}\chi_{k,l}\right] \\
&= \sum_{S}\mathrm{Var}\left[\chi_{i,j}\right] + 2\sum_{\substack{(i,j),(k,l)\in S\times S\\(i,j)\neq(k,l)}}\mathrm{Covar}\left[(\chi_{i,j},\chi_{k,l})\right] \\
&= \sum_{S}\mathrm{E}\left[\chi_{i,j}^2\right] - \mathrm{E}\left[\chi_{i,j}\right]^2 + 2\sum_{\substack{(i,j),(k,l)\in S\times S\\(i,j)\neq(k,l)}}\mathrm{E}\left[\chi_{i,j}\cdot\chi_{k,l}\right] - \mathrm{E}\left[\chi_{i,j}\right]\mathrm{E}\left[\chi_{k,l}\right] \\
&\leq \sum_{S}\mathrm{E}\left[\chi_{i,j}^2\right] + 2\sum_{\substack{(i,j),(k,l)\in S\times S\\(i,j)\neq(k,l)}}\mathrm{E}\left[\chi_{i,j}\cdot\chi_{k,l}\right] \\
&= m^2(f\cdot g) + 2\sum_{\substack{i,j,l\in[m]\\j\neq l}}\mathrm{E}\left[\chi_{i,j}\cdot\chi_{i,l}\right] + 2\sum_{\substack{i,j,k\in[m]\\i\neq k}}\mathrm{E}\left[\chi_{i,j}\cdot\chi_{k,j}\right] \\
&= m^2(f\cdot g) + 2\sum_{\substack{i,j,l\in[m]\\j\neq l}}Pr[Q_f(i)=Q_g(j)=Q_g(l)] \\
&\quad + 2\sum_{\substack{i,j,k\in[m]\\i\neq k}}Pr[Q_f(i)=Q_g(j)=Q_f(k)] \\
&\leq m^2b + 4m^3b^2
\end{aligned}$$

Where the equality in the sixth line comes from removing terms in which $i \neq k$ and $j \neq l$ because these independent variables have zero covariance.

Therefore $\mathrm{Var}\left[s\right] = 4\mathrm{Var}\left[s_{pq}\right] \leq c(m^3b^2 + m^2b)$. Since from Lemma 11, $\mathrm{Var}\left[r\right] \leq c(m^3b^2 + m^2b)$, and the variance is additive for independent random variables, we can write $\mathrm{Var}\left[r-s\right] \leq c(m^3b^2 + m^2b)$. ■

To find the necessary number of samples to achieve constant error probability, we use Chebyshev's Inequality (Theorem 82). It follows from standard techniques that with $O(\log\frac{1}{\delta})$ iterations we can achieve an error probability at most δ. These calculations give us our main theorem on L_2 closeness testing:

PROOF OF THEOREM 15: For the statistic $A = (r-s)$, Chebyshev's inequality implies

that for some constant c,

$$\Pr\left[|A - \mathrm{E}\left[A\right]| > \rho\right] \leq \frac{\mathrm{Var}\left[A\right]}{\rho^2} = \frac{c(m^3b^2 + m^2b)}{\rho^2}$$

When $\|f - g\| \leq \epsilon/2$, the expected value of $r - s$ is $E[r - s] \leq m^2\epsilon^2/4$

$$\begin{aligned}
\Pr\left[\text{PASS}\right] &= \Pr\left[(r-s) < m^2\epsilon^2/2\right] \\
&\geq \Pr\left[|(r-s) - \mathrm{E}\left[r-s\right]| < m^2\epsilon^2/4\right] \\
&\geq 1 - 16c(m^3b^2 + m^2b)/(m^4\epsilon^4)
\end{aligned}$$

Solving this quadratic equation in m, we find that this probability will be at least $2/3$ whenever $m > k(b^2 + \epsilon^2\sqrt{b})/\epsilon^4$ for some constant k.

A similar analysis can be used to show the other direction. If $\|f - g\| > \epsilon$, then $E[r - s] > m^2\epsilon^2$ and

$$\begin{aligned}
\Pr\left[\text{FAIL}\right] &= \Pr\left[(r-s) > m^2\epsilon^2/2\right] \\
&\geq \Pr\left[|(r-s) - \mathrm{E}\left[r-s\right]| < m^2\epsilon^2/2\right] \\
&\geq 1 - 4c(m^3b^2 + m^2b)/(m^4\epsilon^4)
\end{aligned}$$

■

Efficient tests with sub-constant ϵ are also possible if the maximum probability in f and g is known before-hand. From the previous theorem it follows

Corollary 18 *If an upper bound on f and g is known to be $n^{-\alpha}$ then algorithm L_2-Bounded-Distance-Test passes with probability $> 2/3$ if $\|f - g\|_2 \leq \epsilon/2n^\beta$ and fails with probability $> 2/3$ if $\|f - g\|_2 > \epsilon/n^\beta$. The running time of this algorithm is $O(n^{4\beta-2\alpha}/\epsilon^4 + n^{2\beta-\alpha/2}/\epsilon^2)$.*

2.2 Testing closeness in the L_1-norm

It may seem surprising that the L_2 closeness test described in Theorem 15 has a sample complexity with no dependence on the domain size. This can be explained some-

```
L2-Bounded-Distance-Test(⟨A, f, F⟩, ⟨A, g, G⟩, ε, b, δ)
Let m = O((b^2 + ε^2 √b)/ε^4)
Repeat O(log(1/δ)) times
  Let r_p = Self-Col(F, m)
  Let r_q = Self-Col(G, m)
  Let s_pq = Joint-Col(F, G, m)
  Let r = (2m/(m−1))(r_p + r_q)
  Let s = 2 s_pq
  If r − s > m^2 ε^2/2 then reject the iteration
End Repeat
Reject if the majority of iterations reject,
accept otherwise
```

Figure 2.2: Algorithm L_2-Bounded-Distance-Test

what, though, by the nature of the L_2 norm on differences of probability distributions. Consider as an example two distributions which are each uniform over half of their domain, yet the support sets are disjoint. Specifically, let $\vec{x} = (\frac{2}{n}, \ldots, \frac{2}{n}, 0, \ldots, 0)$ and let $\vec{y} = (0, \ldots, 0, \frac{2}{n}, \ldots, \frac{2}{n})$. Then $\|\vec{x}\|_1 = \|\vec{y}\|_1 = 1$ and $\|\vec{x} - \vec{y}\|_1 = 2$, while $\|\vec{x} - \vec{y}\|_2 = 2/\sqrt{n} \ll \epsilon$. The problem is that $\vec{x}$ and $\vec{y}$ are relatively short in the L_2 norm, so $\vec{x} - \vec{y}$ must necessarily also be short. In the L_1 norm, however, all discrete finite probability functions result in unit vectors; there are no "short" probability vectors. Given this type of behavior, discerning constant differences between distributions in the Euclidean norm is not always enlightening. We will therefore in this section give a test for closeness in the L_1 norm.

We could try to bootstrap the L_2 test directly into an L_1 testing algorithm, with a sufficiently small ϵ. For example, since for any vector $\vec{v}$ we have a bound that $\|\vec{v}\|_2 \leq \|\vec{v}\|_1 \leq \sqrt{n}\|\vec{v}\|_2$, a simple way to extend the L_2 closeness test to an L_1 closeness test is by testing L_2 closeness with a parameter of $\frac{\epsilon}{\sqrt{n}}$. If, for example, $\|\vec{p} - \vec{q}\|_1 \leq \frac{\epsilon}{2\sqrt{n}}$ then $\|\vec{p} - \vec{q}\|_2 \leq \frac{\epsilon}{2\sqrt{n}}$, while if $\|\vec{p} - \vec{q}\|_1 > \epsilon$ then $\|\vec{p} - \vec{q}\|_2 > \frac{\epsilon}{\sqrt{n}}$.

According to Corollary 18, with $\beta = \frac{1}{2}$, the running time of our bootstrapped test would be $O(n^{2-2\alpha}/\epsilon^4 + n^{1-\alpha/2}/\epsilon^2)$. With no known bound on the maximum prob-

ability, this is $O(n^2/\epsilon^4)$. Yet the closer the two black-box distributions are known to be to a uniform distribution, the more efficient the test can be. If, for example, $b = \max(\|f\|_\infty, \|g\|_\infty) \leq n^{-2/3}$, we have an L_1 distance-testing algorithm with query complexity of $O(n^{2/3}/\epsilon^4)$: on inputs B_1, B_2, ϵ, call L_2-Distance-Test with distance parameter $(\epsilon/\sqrt{n})$ and return `PASS` if and only if the L_2 test returns `PASS`.

On the other hand, it is also possible to test L_1 distance efficiently if the distributions are highly skewed, that is, if all of the probability weight is concentrated over a small fraction of the domain. In this case, the elements which have positive probability occur frequently enough that their probability weight can be estimated by sampling and Chernoff Bounds (Theorem 83). For example, in a black box distribution where every member of the support set occurs with probability at least $n^{-2/3}$, the weights of each element can be estimated to within a multiplicative factor of $(1+\epsilon)$ with a sample set of size $O(\frac{n^{2/3}\log n}{\epsilon^2})$. This leads to an efficient algorithm for approximating the L_1 distance between distributions with a small support set.

To achieve an algorithm which works for all distributions, we show that one can combine the previous two specialized algorithms in a careful way. The key to the combination is the definition and identification of *big* elements.

Definition 19 *An element i is called **big** with respect to a distribution p if $p_i > \frac{1}{2n^{2/3}}$.*

By identifying the elements which are big in each distribution we are able to apply a different test to the two classes of elements. The result is given.

Theorem 20 *Given parameter δ, and distributions $\vec{p}, \vec{q}$ over a set of n elements, there is a test that runs in time $O(\epsilon^{-4}n^{2/3}\log n \log \frac{1}{\delta})$ such that if $\|\vec{p}-\vec{q}\|_1 \leq \max(\frac{\epsilon^2}{32\sqrt[3]{n}}, \frac{\epsilon}{4\sqrt{n}})$, then the test outputs `pass` with probability at least $1-\delta$ and if $\|\vec{p}-\vec{q}\|_1 > \epsilon$, then the test outputs `fail` with probability at least $1-\delta$.*

The L_1-closeness test proceeds in two stages. First it generates a sample set from each distribution of sufficient size that every big element should occur several times. If

L_1**-Distance-Test(**$\langle \mathcal{A}, p, F\rangle, \langle \mathcal{A}, q, G\rangle, \epsilon, \delta$**)**
(1) Take $M = O(\max(\epsilon^{-2}, 4)n^{2/3}\log n)$ samples from F and G
(2) Let S_p and S_q be the sample sets consisting of
(3) elements that occur at least $(1-\epsilon/63)Mn^{-2/3}$ times
(4) If S_p and S_q are empty
(5) L_2**-Distance-Test(**$\langle \mathcal{A}, p, F\rangle, \langle \mathcal{A}, q, G\rangle, \frac{\epsilon}{2\sqrt{n}}, \delta/2$)
else
(6) Let $\ell_i^p = \#$ times element i appears in S_p
(7) Let $\ell_i^q = \#$ times element i appears in S_q
(8) Let $\tilde{\Delta}_B = \sum_i |\ell_i^p - \ell_i^q|/M$
(9) Fail if $\tilde{\Delta}_B > \epsilon/8$.
(10) Define $\langle \mathcal{A}, p', F'\rangle$ as follows:
(11) sample an element from F
(12) if this sample is not in S_p output it,
(13) otherwise output an $x \in_R [n]$.
(14) Define $\langle \mathcal{A}, q', G'\rangle$ similarly.
(15) L_2**-Bounded-Distance-Test(**$\langle \mathcal{A}, p', F'\rangle$, $\langle \mathcal{A}, q', G'\rangle$, $\frac{\epsilon}{2\sqrt{n}}, 2n^{-2/3}, \delta/2$**)**

Figure 2.3: Algorithm L_1-Distance-Test

there are no big elements identified in either distribution then the two black boxes are passed directly to the L_2-Distance-Test. On the other hand, if big elements are identified then their probabilities are estimated directly from their frequencies in the sample set. Then the distance between the big elements in each distribution is approximately computed. Having identified the big elements, we then modify the input black boxes in a way which allows the distance between the non-big elements to be measured using the L_2-closeness test. The complete test is given in Figure 2.3.

In order to prove the correctness of the test we must first show that our method of sampling is sufficient to determine which elements are big in each distribution. In particular, we need to show that by using a sample of size $O(\epsilon^{-2}n^{2/3}\log n)$, we can estimate the weights of the big elements to within a multiplicative factor of $O(\epsilon)$.

Lemma 21 *Let p be a distribution function over $[n]$. Let $\mathcal{S}$ be a set of $M = \frac{6n^{2/3}\log n}{\epsilon^2}$ samples from p. For any i, define the sample probability $\bar{p}_i$ as $\bar{p}_i = \#\{j \in \mathcal{S} | \mathcal{S}_j = i\}/M$. Then $|\bar{p}_i - p_i| < max\{\epsilon p_i, \epsilon n^{-2/3}\}$ with probability at least $1 - 1/n^2$.*

PROOF: We will analyze three cases, each using the Chernoff bound given in Theorem 83. In particular we will show

1. If $p_i > n^{-2/3}$ then $|\bar{p}_i - p_i| < \epsilon p_i$
2. If $\epsilon^2 n^{-2/3} < p_i \leq n^{-2/3}$ then $|\bar{p}_i - p_i| < \epsilon n^{-2/3}$
3. If $p_i < \epsilon^2 n^{-2/3}$ then $\bar{p}_i < 4\epsilon^2 n^{-2/3}$

where each of the three items happens with error probability $1/n^2$.

The first case is a straightforward application of the Chernoff Bound.

$$\begin{aligned}\Pr\left[\bar{p}_i > (1+\epsilon)p_i\right] &< \exp\left(-\frac{\epsilon^2 p_i}{3}\cdot\frac{6n^{2/3}\log n}{\epsilon^2}\right)\\ &< \exp\left(-\frac{\epsilon^2 n^{-2/3}}{3}\cdot\frac{6n^{2/3}\log n}{\epsilon^2}\right) = 1/n^2\end{aligned}$$

The bound in the other direction is similar.

For the second case we are given $\epsilon^2 n^{-2/3} < p_i \leq n^{-2/3}$. Define $k \in (\epsilon^2, 1)$ so that $p_i = kn^{-2/3}$. Then the additive error bound is

$$\begin{aligned}\Pr\left[\bar{p}_i - p_i > \epsilon n^{-2/3}\right] &= \Pr\left[\bar{p}_i > \left(1+\frac{\epsilon}{k}\right)p_i\right] < \Pr\left[\bar{p}_i > \left(1+\frac{\epsilon}{\sqrt{k}}\right)p_i\right]\\ &< \exp\left(\frac{-\epsilon^2 k}{3kn^{2/3}}\cdot\frac{6n^{2/3}\log n}{\epsilon^2}\right) = 1/n^2\end{aligned}$$

Note that we use the fact that $k < 1$ to replace it by $\sqrt{k} > k$. Then we use the fact that $k > \epsilon^2$ so that we may employ the Chernoff Bound. Again the bound in the other direction is similar.

For the last case, $p_i \leq \epsilon^2 n^{-2/3}$. Then $\Pr\left[\bar{p}_i > 4\epsilon^2 n^{-2/3}\right] \leq n^{-18}$ by a simple application of Chernoff bounds. ■

Since the big elements have sufficient probability that they can be learned efficiently, and the distance on small elements can be tested using the L_2 distance test, it follows that we can devise a test for the two extreme cases: namely when the distribution is close to uniform, or when the distribution has a small support set. The final proof of

correctness will show that the two tests can indeed be combined for the case in which a distribution is neither close to uniform, or highly skewed.

Lemma 22 *Let $S = S_p \cup S_q$. By our assumption, all the big elements of both $\vec{p}$ and $\vec{q}$ are in S, and no element with weight less than $\epsilon^2 n^{-2/3}$ (in either distribution) is in S. Let Δ_B be the L_1-distance attributed to the elements in S. Let $\Delta_S = |\vec{p'} - \vec{q'}|$, where $\vec{p'}$ and $\vec{q'}$ are defined as in the algorithm (Figure 2.3). Then*

$$\Delta_S \leq |\vec{p} - \vec{q}| \leq \Delta_S + 2\Delta_B.$$

PROOF: Notice that $\Delta_B \leq |\vec{p} - \vec{q}|$. Let $A = \sum_{i \in S} p_i$ and $B = \sum_{i \in S} q_i$. Then, $p'_i = p_i + A/n$ if $i \notin S$, and $p'_i = A/n$ otherwise. Similarly, $q'_i = q_i + B/n$ if $i \notin S$, and $q'_i = B/n$ otherwise. So,

$$\begin{aligned}
\Delta_S &= \sum_{i \notin S} |p'_i - q'_i| + \sum_{i \in S} |p'_i - q'_i| \\
&\leq \sum_{i \notin S} (|p_i - q_i| + |A - B|/n) + \sum_{i \in S} |A - B|/n \\
&= \sum_{i \notin S} |p_i - q_i| + |A - B| = \sum_{i \notin S} |p_i - q_i| + |\sum_{i \in S} p_i - \sum_{i \in S} q_i| \\
&\leq \sum_{i \notin S} |p_i - q_i| + \sum_{i \in S} |p_i - q_i| = |p - q|
\end{aligned}$$

and

$$\begin{aligned}
|p - q| &= \Delta_B + \sum_{i \notin S} |p_i - q_i| \leq \Delta_B + \sum_{i \notin S} |p'_i - q'_i| + \sum_{i \notin S} |A - B|/n \\
&\leq \Delta_B + \Delta_S + |A - B| \leq 2\Delta_B + \Delta_S
\end{aligned}$$

■

Our algorithm efficiently estimates Δ_B:

Lemma 23 *The algorithm estimates Δ_B from frequencies by sampling to within an additive error of $\epsilon/9$.*

PROOF: The error on the i^{th} term of the sum is bounded by $\frac{\epsilon}{63}(\max(p_i, n^{-2/3}) + \max(q_i, n^{-2/3})) \leq \frac{\epsilon}{63}(p_i + q_i + 2n^{-2/3})$. Consider the sum, over i, of these error terms. Notice that this sum is over at most $2n^{2/3}/(1 - \frac{\epsilon}{63})$ elements in S. Hence, the total additive error is bounded by

$$\sum_{i \in S} \frac{\epsilon}{63}(p_i + q_i + 2n^{-2/3}) \leq \frac{\epsilon}{63}(2 + 4/(1 - \epsilon/63)) \leq \epsilon/9.$$

■

PROOF OF THEOREM 20: Assume for all i such that $p_i > \epsilon^2 n^{-2/3}$, $|\bar{p}_i - p_i| < \frac{\epsilon}{21}\max(p_i, n^{-2/3})$ (similarly for $\vec{q}$). By Lemma 21, this event happens with probability at least $1 - \frac{2}{n}$.

Using the notation of L_1-*Distance-Test*, if S_p and S_q are empty then the theorem follows from the correctness of the L_2-*Distance-Test*, because by definition these sets are empty if $\max(\|\vec{p}\|_\infty, \|\vec{q}\|_\infty) < n^{-2/3}$.

Note that $\max(\|\vec{p'}\|_\infty, \|\vec{q'}\|_\infty) < n^{-2/3} + n^{-1}$. So, we can use the L_2-*Distance-Test* on $\vec{p'}$ and $\vec{q'}$ with $m = O(\epsilon^{-4}n^{2/3})$ as shown by Theorem 15. If $|\vec{p} - \vec{q}| < \frac{\epsilon^2}{32\sqrt[3]{n}}$ then so are Δ_B and Δ_S. The first phase of the algorithm clearly passes. By Fact 84, since $\|\vec{p'}\|_\infty, \|\vec{q'}\|_\infty \leq 2n^{-2/3}$ we can bound $\|\vec{p'} - \vec{q'}\|_2 \leq \frac{\epsilon}{4\sqrt{n}}$. Therefore, the L_2-*Distance-Test* passes. Similarly, if $\|\vec{p} - \vec{q}\|_1 > \epsilon$ then either $\Delta_B > \epsilon/4$ or $\Delta_S > \epsilon/2$. Either the first phase of the algorithm or the L_2-*Distance-Test* will fail.

For the sample complexity, note that the number samples required for the first phase is $O(\epsilon^{-2}n^{2/3}\log n)$ and that the samples for L_2-*Distance-Test* is $O(n^{2/3}\epsilon^{-4}\log\frac{1}{\delta})$. It is easy to see that our algorithm makes an error either when it makes a bad estimation of Δ_B or when L_2-*Bounded-Distance-Test* makes an error. So, the probability of error is bounded by δ. ■

2.3 Subsequent Work

When an upper bound is known on $\|f\|_\infty$ and $\|g\|_\infty$, an improved sample complexity and estimation gap have been shown by Levi, Ron and Rubinfeld [LRR13]. They add an additional check to our algorithm that the L_∞ bound of f is close to the squared L_2 norm of g. This allows them to tighten the variance bound and show

Theorem 24 ([LRR13]) *Given two black-box distributions $\langle \mathcal{A}, f, F\rangle, \langle \mathcal{A}, g, G\rangle$ with $\|f\|_\infty \leq \|g\|_\infty$, and $b_f = \|f\|_\infty, b_g = \|g\|_\infty$, there is a test requiring $O((|\mathcal{A}|^2 b_f b_g \epsilon^{-4} + \sqrt{|\mathcal{A}|} b_f \epsilon^{-2}) \log(1/\delta))$ samples such that (1) if $|X - Y| \leq \frac{\epsilon}{2\sqrt{|\mathcal{A}|}}$, it outputs PASS with probability at least $1 - \delta$ and (2) if $|X - Y| > \epsilon$, it outputs FAIL with probability at least $1 - \delta$.*

Our results were further improved and matching upper and lower bounds for testing closeness of two distributions were given by Chan, Diakonikolas, Valiant and Valiant [CDVV14].

Theorem 25 ([CDVV14]) *Given $\epsilon > 0$ and black box distributions $\langle \mathcal{A}, f, F\rangle, \langle \mathcal{A}, g, G\rangle$ over $[n]$, there is an algorithm which uses $O(\max(n^{2/3}/\epsilon^{4/3}, n^{1/2}/\epsilon^2))$ samples, runs in time linear in its sample size and with probability at least 2/3 distinguishes whether $f = g$ or $\|f - g\|_1 \geq \epsilon$. Additionally, $\Omega(\max(n^{2/3}/\epsilon^{4/3}, n^{1/2})/\epsilon^2)$ samples are information-theoretically necessary.*

Following the work of [CDVV14], a simpler testing algorithm was given by Diakonikolas and Kane [DK16] that matches the upper bound given above and also extends to the case of unequal sample sizes.

2.4 Testing Against An Explicit Distribution

We now turn to the problem of testing the L_1 distance between distributions when one of them is a black box distribution and the other is an explicit distribution. This problem has been studied in the special case that the explicit distribution is uniform by Goldreich and Ron[GR00], who give an algorithm for estimating the L_2 norm (Theorem 12). By using a sufficiently small ϵ, their algorithm yields an L_1 tester requiring $O(\sqrt{n}/\epsilon^4)$ samples (see for example [Can15].)

The general problem of testing against an arbitrary black-box distribution could be solved by the techniques given in Section 2.2 if we used the explicit distribution to simulate a black box distribution, as described in the discussion following Definition 5. The sample complexity would then be $\tilde{O}(n^{2/3})$, for constant ϵ. Intuitively, though, it would seem that a testing algorithm could sample from the black box and "compare" the output to the explicit distribution directly, yielding an improved running time. We will show that this is the case and that an algorithm exists which has sample complexity $\tilde{O}(\sqrt{n}/\epsilon^2)$.

Let $\langle \mathcal{A}, f, F \rangle$ be a black-box distribution and $\langle \mathcal{A}, g \rangle$ be an explicit distribution. Our task is to determine if $\|f - g\|_1 < \epsilon$ using as few samples (from F) as possible. Our algorithm works by decomposing an arbitrary black box distribution into several black box distributions, each of which will be close to uniform if the original distribution is close to uniform.

2.4.1 Flattening an Explicit Distribution

We will show in this section that any explicit distribution over n elements can be decomposed into $O(\log n)$ separate distributions, each of which is close to uniform. This decomposition, a process we refer to as "flattening," is the basis of our reduction from a

single black box to several black boxes which underlies our closeness test.

Motivation for the decomposition comes from an observation on the standard definition of conditional probability:

Definition 26 *Let g be any distribution over $\mathcal{A}$ and let $\mathcal{A}_i$ be any subset of of $\mathcal{A}$. Then the* **conditional distribution** *function $g|_{\mathcal{A}_i}$ is defined as*

$$g|_{\mathcal{A}_i}(x) = \frac{g(x)}{\sum_{y \in \mathcal{A}_i} g(y)}.$$

This function is also denoted by $g(x|\mathcal{A}_i)$.

This agrees with the common definition of conditional probability in that, if X is a random variable with distribution function g, then

$$\Pr[X = x \mid X \in \mathcal{A}_i] = \frac{\Pr[X = x]}{\Pr[X \in \mathcal{A}_i]}.$$

Moreover, if we rewrite the equation in Definition 26 we find simply that

$$g(x) = g|_{\mathcal{A}_i(x)} \cdot g(\mathcal{A}_i).$$

If the subset $\mathcal{A}_i$ were chosen such that for all $x, y \in \mathcal{A}_i$ the probabilities $g(x)$ and $g(y)$ were all very close, that is if the conditional distribution were close to uniform, then $g|_{\mathcal{A}_i}(x)$ would be close to $1/|\mathcal{A}_i|$. In the above equation we could replace $g|_{\mathcal{A}_i}(x)$ with $1/|\mathcal{A}_i|$ and have that

$$g(x) \cong \frac{1}{|\mathcal{A}_i|} g(\mathcal{A}_i).$$

for all $x \in \mathcal{A}_i$. Our "flattening" process partitions a probability space $\mathcal{A}$ in such a way that for every $\mathcal{A}_i$ this approximation is guaranteed to within a factor of $(1 + \epsilon)$.

Definition 27 *Let $\langle \mathcal{A}, g \rangle$ and ϵ be given. Let $n = \#\{\mathcal{A}\}$. Then* **Flatten**$(\langle \mathcal{A}, g \rangle, \epsilon)$ *is an operation that returns a partition of $\mathcal{A}$ into subsets $\{\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k\}$ with $k \leq \frac{\log n + \log\log n}{\log(1+\epsilon)} \leq \frac{2\log n}{\log(1+\epsilon)}$, $\mathcal{A}_0 = \{i \mid g(i) < 1/(n \log n)\}$, and for all i in $[k]$,*

$$\mathcal{A}_i = \left\{ j \;\middle|\; \frac{(1+\epsilon)^{i-1}}{n \log n} \leq g(j) < \frac{(1+\epsilon)^i}{n \log n} \right\}.$$

It is in fact the case that the conditional distributions over each of our partitions are close to uniform in both the L_1 and the L_2 norm.

Lemma 28 *Let $\langle \mathcal{A}, g \rangle$ be an explicit distribution over n elements and ϵ be given. Define the partition $\{\mathcal{A}_0, \ldots, \mathcal{A}_k\} = \texttt{Flatten}(\langle \mathcal{A}, g \rangle, \epsilon)$. Then for all $\mathcal{A}_{i>0}$ it is true that both $\|g|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_1 \leq \epsilon$ and $\|g|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 \leq \epsilon^2/|\mathcal{A}_i|$. Furthermore $g(\mathcal{A}_0) \leq 1/\log n$.*

PROOF: Since $g(x) < \frac{1}{n \log n}$ for all $x \in \mathcal{A}_0$, clearly $g(\mathcal{A}_0) \leq 1/\log n$. For $i \geq 1$, consider an arbitrary (non-empty) subset $\mathcal{A}_i$. Without loss of generality, assume that $\mathcal{A}_i = \{1, \ldots, \ell\}$ and $g(1) \leq \cdots \leq g(\ell)$. Let $h = g|_{\mathcal{A}_i}$. Then, $h(\ell)/h(1) < 1 + \epsilon$. Also, by averaging, $h(1) \leq 1/\ell \leq h(\ell)$. Hence $h(\ell) \leq (1+\epsilon)h(1) \leq (1+\epsilon)/\ell$. Similarly it can be shown that $h(1) \geq 1/(\ell(1+\epsilon)) > (1-\epsilon)/\ell$. It follows that $|h(j) - 1/\ell| \leq \epsilon/\ell$ for all $j = 1, \ldots, \ell$. The bounds on the norm follow from this component-wise bound: $\|g|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_1 = \sum_{j \in \mathcal{A}_i} |h(j) - U_{\mathcal{A}_i}| \leq \epsilon$ and $\|g|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 = \sum_{j \in \mathcal{A}_i} (h(j) - U_{\mathcal{A}_i})^2 \leq \epsilon^2/\ell = \epsilon^2/|\mathcal{A}_i|$. ■

In our applications of `Flatten`, we usually ignore the 0-th subset $\mathcal{A}_0$ since the total probability mass over this set, $g(\mathcal{A}_0) \leq \frac{1}{\log n}$, is negligible for our purposes.

The `Flatten` operation, in partitioning the sample space $\mathcal{A}$, implicitly defines a new distribution over the subsets $\mathcal{A}_i$ of the partition, where the weight associated with each subset is just $g(\mathcal{A}_i)$ – the probability that an element selected from $\mathcal{A}$ will be a member of $\mathcal{A}_i$. It is occasionally necessary for us to refer to this new distribution explicity, so we define it here:

Definition 29 *Given an explicit distribution $\langle \mathcal{A}, g \rangle$ and a constant parameter $\epsilon > 0$, the* **U-reduction** *is an explicit distribution $\langle \{\mathcal{A}_0, \ldots, \mathcal{A}_k\}, g_\downarrow \rangle$ where $\{\mathcal{A}_0, \ldots, \mathcal{A}_k\}$ is the output of $\texttt{Flatten}(\langle \mathcal{A}, g \rangle, \epsilon)$ and $g_\downarrow(i) = g(\mathcal{A}_i) = \sum_{x \in \mathcal{A}_i} g(x)$.*

2.4.2 Applying the approximation

The flattening procedure is useful to us because it suggests a two-level approach to testing closeness to an explicit distribution. Once $\mathcal{A}$ is flattened, it is easy to use the *Naive Algorithm* on black box F to determine if $f_\downarrow$ and $g_\downarrow$ are close. If they are, then we can test each conditional distribution $f|_{\mathcal{A}_i}$ separately for closeness to uniformity. The hope is that if f is close to uniform in each partition, and if $f_\downarrow$ and $g_\downarrow$ are close, then f and g will be close overall. This turns out to be the case:

Lemma 30 *Let f, g be distribution functions over $\mathcal{A}$ and let $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ be a partition of $\mathcal{A}$. If for all $\mathcal{A}_i$, $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 \leq \epsilon_1$ and $\|f_\downarrow - g_\downarrow\|_1 \leq \epsilon_2$, then $\|f - g\|_1 \leq \epsilon_1 + \epsilon_2$.*

PROOF: Using the hypothesis and Definition 26,

$$\begin{aligned}
\|f-g\|_1 &= \sum_{\mathcal{A}_i}\sum_{j\in\mathcal{A}_i} |f(j) - g(j)| \\
&= \sum_{\mathcal{A}_i}\sum_{j\in\mathcal{A}_i} |f(\mathcal{A}_i)\cdot f(j|\mathcal{A}_i) - g(\mathcal{A}_i)\cdot g(j|\mathcal{A}_i)| \\
&\leq \sum_{\mathcal{A}_i}\sum_{j\in\mathcal{A}_i} |f(\mathcal{A}_i)\cdot f(j|\mathcal{A}_i) - f(\mathcal{A}_i)\cdot g(j|\mathcal{A}_i)| \\
&\qquad + \sum_{\mathcal{A}_i}\sum_{j\in\mathcal{A}_i} |f(\mathcal{A}_i)\cdot g(j|\mathcal{A}_i) - g(\mathcal{A}_i)\cdot g(j|\mathcal{A}_i)| \\
&= \sum_{\mathcal{A}_i} f(\mathcal{A}_i) \sum_{j\in\mathcal{A}_i} |f(j|\mathcal{A}_i) - g(j|\mathcal{A}_i)| + \sum_{j\in\mathcal{A}_i} g(j|\mathcal{A}_i) \sum_{\mathcal{A}_i} |f(\mathcal{A}_i) - g(\mathcal{A}_i)| \\
&= \sum_{j\in\mathcal{A}_i} |f(j|\mathcal{A}_i) - g(j|\mathcal{A}_i)| + \sum_{\mathcal{A}_i} |f(\mathcal{A}_i) - g(\mathcal{A}_i)| \\
&\leq \epsilon_1 + \epsilon_2.
\end{aligned}$$

■

In addition, a valid testing algorithm requires the converse of the above to hold: if f and g are close, then so are their conditional distributions. This turns out to be true for the sufficiently "heavy" subsets of the probability space.

Lemma 31 *Let f, g be distribution functions over $\mathcal{A}$ and let $\mathcal{A}_j \subseteq \mathcal{A}$. Then the difference between the distributions $\|f|_{\mathcal{A}_j} - g|_{\mathcal{A}_j}\|_1$ is no more than $2\|f-g\|_1/f(\mathcal{A}_j)$.*

PROOF: Using the definition of conditional distributions,

$$\begin{aligned}
&\|f|_{\mathcal{A}_j} - g|_{\mathcal{A}_j}\|_1 = \\
&\quad \sum_{i\in\mathcal{A}_j} \left|\frac{f(i)}{f(\mathcal{A}_j)} - \frac{g(i)}{g(\mathcal{A}_j)}\right| = \sum_{i\in\mathcal{A}_j} \left|\frac{f(i)}{f(\mathcal{A}_j)} - \frac{g(i)}{f(\mathcal{A}_j)} + \frac{g(i)}{f(\mathcal{A}_j)} - \frac{g(i)}{g(\mathcal{A}_j)}\right| \\
\leq\ & \frac{1}{f(\mathcal{A}_j)} \sum_{i\in\mathcal{A}_j} |f(i)-g(i)| + \left|\frac{1}{f(\mathcal{A}_j)} - \frac{1}{g(\mathcal{A}_j)}\right| \sum_{i\in\mathcal{A}_j} g(i) \\
\leq\ & \frac{1}{f(\mathcal{A}_j)}\|f-g\|_1 + \left|\frac{1}{f(\mathcal{A}_j)} - \frac{1}{g(\mathcal{A}_j)}\right| g(\mathcal{A}_j) \\
=\ & \frac{1}{f(\mathcal{A}_j)} \left(\|f-g\|_1 + |g(\mathcal{A}_j) - f(\mathcal{A}_j)|\right) \\
\leq\ & \frac{1}{f(\mathcal{A}_j)} \left(\|f-g\|_1 + \|f-g\|_1\right) = 2\frac{\|f-g\|_1}{f(\mathcal{A}_j)}
\end{aligned}$$

■

The size restriction is required in this lemma because f and g could differ significantly (relative to ϵ) on an element x which is contained in a relatively small weight subset $\mathcal{A}_i$. In this case the relative error $x/\mathcal{A}_i$ could be very large. It is sufficient, though, to restrict our test to subsets with a large enough weight. Within each of these subsets we will test our black box F for closeness to uniformity in the L_2 norm using the result of Theorem 12. Since the preceding two lemmas require closeness in the L_1 norm, we need to show that testing for closeness in the L_2 norm is sufficient. First we give a useful lemma for uniform distributions:

Lemma 32 *For any distribution function f over $\mathcal{A}$, $\|f\|_2^2 - \|U_\mathcal{A}\|_2^2 = \|f - U_\mathcal{A}\|_2^2$.*

PROOF: $\|f\|_2^2 - \|U_\mathcal{A}\|_2^2 = \sum_{j\in\mathcal{A}} f(j)^2 - \frac{1}{|\mathcal{A}|} = \sum_{j\in\mathcal{A}} f(j)^2 - \frac{2}{|\mathcal{A}|}\sum_{j\in\mathcal{A}} f(j) + \frac{1}{|\mathcal{A}|} = \sum_{j\in\mathcal{A}}(f(j) - \frac{1}{|\mathcal{A}|})^2 = \|f - U_\mathcal{A}\|_2^2$. ■

We now show that a bound on the L_2-norm implies a certain bound on the L_1 norm for almost-uniform distributions.

Lemma 33 *Let f be a distribution function over $\mathcal{A}$ and let $\mathcal{A}_i \subseteq \mathcal{A}$. If $\|f|_{\mathcal{A}_i}\|_2^2 \leq \frac{1+\epsilon^2}{\#\{\mathcal{A}_i\}}$ then $\|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_1 \leq \epsilon$.*

PROOF: By Cauchy-Schwartz $\|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_1 \leq \sqrt{|\mathcal{A}_i|}\,\|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2$ which by Lemma 32, equals $\sqrt{|\mathcal{A}_i|}(\|f|_{\mathcal{A}_i}\|_2^2 - \|U_{\mathcal{A}_i}\|_2^2)^{1/2} \leq \sqrt{|\mathcal{A}_i|}\left(\frac{1+\epsilon^2}{|\mathcal{A}_i|} - \frac{1}{|\mathcal{A}_i|}\right)^{1/2} = \epsilon$. ■

If we have two distributions f and g over $\mathcal{A}$ and a subset $\mathcal{A}_i$ of $\mathcal{A}$ which was constructed so that g is known to be approximately uniform over $\mathcal{A}_i$, and we furthermore test that f is also close to uniform over $\mathcal{A}_i$, then we can conclude that f and g are close to each other over the subset. An immediate corollary to the previous lemma follows:

Corollary 34 *Let f, g be distributions over $\mathcal{A}$ and let ϵ be a constant. If $\{\mathcal{A}_0, \ldots, \mathcal{A}_k\} =$ `Flatten`$(\langle \mathcal{A}, g\rangle, \epsilon)$ and if $\|f|_{\mathcal{A}_i}\|_2^2 \leq \frac{1+\epsilon^2}{\#\{\mathcal{A}_i\}}$ then $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 \leq 2\epsilon$.*

PROOF: Using Lemma 28 and triangle inequality, $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 < \|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_1 + \|U_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 \leq 2\epsilon$. ■

TestIdentity$(\langle \mathcal{A}, f, F\rangle, \langle \mathcal{A}, g\rangle, \epsilon)$
(1) $\mathcal{A} \stackrel{\text{def}}{=} \{\mathcal{A}_0, \ldots, \mathcal{A}_k\} =$ *Flatten*$(\langle \mathcal{A}, g\rangle, \epsilon/\sqrt{2})$.
(2) Let M be a set of $O(\sqrt{n}\epsilon^{-2}\log n)$ samples from X.
(3) For each subset $\mathcal{A}_i$ do
(4) Let $\mathcal{S}_i = M \cap \mathcal{A}_i$ (preserving repetitions);
Let $\ell_i = |\mathcal{S}_i|$ (counting also repetitions).
(5) If $g(\mathcal{A}_i) \geq \epsilon/k$ then
(6) If $\ell_i < O(\sqrt{n}\epsilon^{-2})$ then FAIL.
(7) Let $\tilde{f}_i$ be an estimate of $\|X|_{\mathcal{A}_i}\|_2^2$ using $\mathcal{S}_i$.
(8) If $\tilde{f}_i > (1+2\epsilon^2)/|\mathcal{A}_i|$ then FAIL.
(9) If $\sum_{i\geq 1} |\ell_i/\#\{\mathcal{S}\} - g(\mathcal{A}_i)| > \epsilon$ then FAIL
(10) PASS.

Figure 2.4: Algorithm TestIdentity

We are now prepared to prove our main theorem.

Theorem 35 *Algorithm TestIdentity$(\langle \mathcal{A}, f, F\rangle, \langle \mathcal{A}, g\rangle, \epsilon)$ is such that: (1) if $\|f - g\|_1 \leq \frac{\epsilon^3}{4\sqrt{n}\log n}$, it outputs PASS with constant probability and (2) if $\|f - g\|_1 > 6\epsilon$, it outputs FAIL with constant probability. The algorithm uses $\tilde{O}(\sqrt{n}/\epsilon^2))$ samples.*

PROOF: Algorithm *TestIdentity* appears in Figure 2.4. Theorem 12 is employed in step (7) to guarantee the accuracy of our estimate. Suppose that the algorithm outputs PASS with high probability. This implies that for each partition R_i for which steps (6)–(8) were performed (which are those for which $g(\mathcal{A}_i) \geq \epsilon/k$), we have $\|f|_{\mathcal{A}_i}\|_2^2 \leq (1+\epsilon^2)/|\mathcal{A}_i|$. From the Corollary to Lemma 33, for each of these $\mathcal{A}_i$, $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 \leq 2\epsilon$.

The sum of $g(\mathcal{A}_i)$ over all R_i for which steps (6)–(8) were skipped is at most ϵ. Also, by step (9) the total difference between the weight in each subset and the expected weight $g(\mathcal{A}_i)$ is not more than ϵ. So the total difference between f and g coming from these conditions sums up to no more than 3ϵ. Adding this to the 3ϵ difference over the partitions that were not skipped in steps (6)–(8) (by applying Lemma 30 with $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 \leq 2\epsilon$), we get that $|f-g| \leq 6\epsilon$. Therefore if $|f-g| > 6\epsilon$, the algorithm fails with high probability.

On the other hand, suppose $|f-g| < \frac{\epsilon^3}{4\sqrt{n}\log n}$. From the definition of `Flatten`, step (1) will return a partition with $k \leq (2/\log(1+\epsilon/\sqrt{2})) \cdot \log n < (2\sqrt{2}/\epsilon) \cdot \log n$ elements. Using Lemma 31 for all subsets $\mathcal{A}_i$ with $g(\mathcal{A}_i) \geq \epsilon/k > \epsilon^2/(2\sqrt{2}\log n)$, we have $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_1 < \epsilon/(\sqrt{2n})$. This implies $\|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_2^2 < \epsilon^2/(2n) < \epsilon^2/(2|\mathcal{A}_i|)$. Since from Lemma 28, $\|g|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 < \epsilon^2/(2|\mathcal{A}_i|)$, then by the triangle inequality, $\|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 \leq \|f|_{\mathcal{A}_i} - g|_{\mathcal{A}_i}\|_2^2 + \|g|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 \leq \epsilon^2/|\mathcal{A}_i|$. So by Lemma 32, $\|f|_{\mathcal{A}_i}\|_2^2 = \|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 + \|U_{\mathcal{A}_i}\|_2^2 \leq (1+\epsilon^2)/|\mathcal{A}_i|$. Therefore the algorithm will pass with high probability on all such partitions. Due to the closeness of the input distributions, the algorithm will pass step (9) as well.

The sample complexity is $\tilde{O}(\sqrt{n}\epsilon^{-2})$ from step (2), which dominates the sample complexity of step (9) (no other samples are taken throughout the algorithm). ■

Algorithm *TestIdentity* will pass distributions which are close in L_1-norm and fail distributions which are not 6ϵ-close. It may also be desirable in certain cases to test the closeness of distributions in a pointwise fashion. We can show that algorithm *TestIden-*

tity has this application. Namely, distributions f and g will pass if the probability $f(x)$ of each element according to f is within $(1+\epsilon)$ of the probability $g(x)$ according to g. This particular property of the algorithm will be useful later in this paper. Essentially this theorem tells us that if you apply `Flatten` to a sample space based on an explicit distribution $\vec{g}$, then a probability distribution that approximates g on every element will also be approximately uniform in each subset of the computed partition. (Note the following theorem has been adapted by [LRR13] in their definition and implementation of a *Tolerant Identity Tester*).[2]

Theorem 36 *For algorithm TestIdentity*$(\langle \mathcal{A}, f, F\rangle, \langle \mathcal{A}, g\rangle, \epsilon)$*: (1) if* $(1-\epsilon/9)g(x) \leq f(x) \leq (1+\epsilon/9)g(x)$ *for all* i *in the sample space of* f *and* g *(disregarding the subset* $\mathcal{A}_0$*), then it outputs PASS with constant probability and (2) if* $\|f-g\|_1 > 6\epsilon$*, it outputs FAIL with constant probability. The algorithm uses* $\tilde{O}(\sqrt{n}\ \text{poly}(\epsilon^{-1}))$ *samples.*

PROOF: We only show the completeness case, since the other case is identical to the previous theorem. In particular we have shown above that the assumptions imply that $\|f|_{\mathcal{A}_i} - U|_{\mathcal{A}_i}\|_2^2 < \epsilon^2/|\mathcal{A}_i|$ for every subset $\mathcal{A}_i$. In order to get the same type of result from our current hypothesis, we compute the distance from uniformity of the conditional distributions, given the approximation assumptions in our theorem statement.

Let $\{\mathcal{A}_0, \ldots, \mathcal{A}_k\} = \texttt{Flatten}(\mathcal{A}, g, \epsilon/9)$. For each subset $\mathcal{A}_i$ we know that each element within this bucket is almost equi-probable, that is, the conditional probability of x given that x is in subset $\mathcal{A}_i$ is close to uniform. So for all $x \in \mathcal{A}_i$.

$$\frac{1-\epsilon/9}{\#\{\mathcal{A}_i\}} \leq \frac{g(x)}{g(\mathcal{A}_i)} \leq \frac{1+\epsilon/9}{\#\{\mathcal{A}_i\}}$$

Now from our hypothesis on the closeness of f and g we know that $(1-\epsilon/9)g(x) \leq f(x) \leq (1+\epsilon/9)g(x)$ and by summing over the subset, $(1-\epsilon/9)g(\mathcal{A}_i) \leq f(\mathcal{A}_i) \leq (1+\epsilon/9)g(\mathcal{A}_i)$. By substitution we can show that according to g, the conditional distributions

[2] See section 6, page 329 *et seq*

over $\mathcal{A}_i$ are almost-uniform. That is

$$\frac{(1-\epsilon/9)^3}{\#\{\mathcal{A}_i\}} \leq \frac{f(x)}{f(\mathcal{A}_i)} \leq \frac{(1+\epsilon/9)^3}{\#\{\mathcal{A}_i\}}$$

Since $(1-\epsilon/9)^3 > 1-\frac{2}{3}\epsilon$ and $(1+\epsilon/9)^3 < 1+\frac{2}{3}\epsilon$ we have shown, as in the proof of Theorem 35 that

$$\|f|_{\mathcal{A}_i} - U_{\mathcal{A}_i}\|_2^2 \leq \frac{\epsilon^2}{2 \cdot \#\{S_i\}}$$

The result follows immediately from the same calculations as in the proof of Theorem 35.

2.5 Subsequent Work

The problem of identity testing against a uniform distribution has been shown to have a tight sample complexity of $\Theta(\sqrt{n}/\epsilon^2)$, when $\epsilon > n^{-1/4}$, in [Pan08] and for any $\epsilon > 0$ in [DGPP17]. Additionally, recent improvements on these results have led to an upper bound of $O(\sqrt{n}/\epsilon^2)$ for identity testing against any known distribution [VV17], [DKN15]. Moreover an improved analysis of the collision-based technique we employed has shown that collision comparison algorithms using $\Omega(\sqrt{n}/\epsilon^2)$ samples are sample-optimal [DGPP17]. The problem of general identity testing has also recently been shown to be reducible to uniformity testing by Goldreich [Gol17], requiring only constant additional overhead, in contrast to our results requiring $\log(n)$ overhead.

■

2.6 Testing independence

The independence of random variables is a central concept in statistics and there exist many different procedures which test for it. Given two random variables over domains of size n and m, classical tests such as the χ^2 test or the Kolmogorov-Smirnoff test work well when n and m are small. For large n, m these tests in practice require more than $n \cdot m$ samples, which may be huge. Moreover it is often the case in statistical applications that the cost of acquiring a sample is the bottleneck in any procedure.

In this section we develop a general algorithm (Subsection 2.6.5) for the independence testing problem with a sublinear sample complexity (in the size of $[n] \times [m]$). To our knowledge, this is the first sublinear time test which makes no assumptions about the structure of the distribution.

2.6.1 Independence and approximate independence

We begin with some standard probability definitions which we use extensively in this section. First we define probability distributions over two-dimensional spaces and the associated marginal distributions which project these bivariate distributions into one dimension.

Definition 37 *Given random variables X over discrete space $\mathcal{A}$ and Y over discrete space $\mathcal{B}$ and a function $f : \mathcal{A} \times \mathcal{B} \to [0, 1]$ we say f is a* **joint probability distribution function (pdf)** *for X and Y if $f(x, y) = \Pr[X = x \text{ and } Y = y]$.*

The distribution of X or Y alone is a marginal distribution function.

Definition 38 *Given random variables X over $\mathcal{A}$ and Y over $\mathcal{B}$ with joint pdf f, the*

marginal distribution functions f_X *and* f_Y *are defined as follows:*

$$\begin{aligned} f_X(x) &= \Pr[X = x] = \sum_{y \in \mathcal{B}} f(x, y) \\ f_Y(y) &= \Pr[Y = y] = \sum_{x \in \mathcal{A}} f(x, y) \end{aligned}$$

Now we can define independence of random variables

Definition 39 *Given random variables* X *over* $\mathcal{A}$ *and* Y *over* $\mathcal{B}$ *with joint pdf* f*, we say* X *and* Y *are* **independent** *if*

$$\forall x \in \mathcal{A}, y \in \mathcal{B}\ f(x, y) = f_X(x) f_Y(y)$$

One functional shorthand which will be useful notation for us is that of a cross product. This combines two one-dimensional functions to produce one two-dimensional functions.

Definition 40 *Given two functions* f *and* g*, the* **cross-product** *function* $f \times g$ *is the function defined*

$$(f \times g)(x, y) = f(x) g(y)$$

over all x *in the domain of* f *and all* y *in the domain of* g*.*

Using this handy notation we can say that X, Y are independent if $f = f_X \times f_Y$. Finally we define the property of approximate independence in a natural way.

Definition 41 *Given random variables* X *over* $\mathcal{A}$ *and* Y *over* $\mathcal{B}$ *with joint pdf* f*, we say* X *and* Y *are* ϵ**-independent** *if there is a function* g *for which* $g = g_X \times g_Y$ *and* $\|f - g\|_1 \leq \epsilon$*. Otherwise, we say* X *and* Y *are* **not** ϵ**-independent** *or are* ϵ**-far from being independent**.

Using a slight abuse of nomenclature we will also sometimes say that distribution functions are independent, or ϵ-independent, when strictly we are referring to the random

variables with which they are associated. We will use the term "approximately independent" to refer to random variables that are ϵ-independent where ϵ, though not explicitly given in the context, is assumed to be small.

A few lemmas will prove helpful in our calculations. First, the distribution functions of independent random variables are close if their marginal distributions are close.

Lemma 42 ([SV99]) *Let f and g be joint distribution functions over $\mathcal{A} \times \mathcal{B}$. Then the marginal distributions satisfy $\|f_X \times f_Y - g_X \times g_Y\|_1 \leq \|f_X - g_X\|_1 + \|f_Y - g_Y\|_1$.*

PROOF:

$$\begin{aligned}
\|f_X \times f_Y - g_X \times g_Y\|_1 &= \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} \|f_X(i)f_Y(j) - g_X(i)g_Y(j)\|_1 \\
&\leq \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} \|f_X(i)f_Y(j) - f_X(i)g_Y(j)\|_1 + \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} \|f_X(i)g_Y(j) - g_X(i)g_Y(j)\|_1 \\
&= \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} f_X(i)\cdot\|f_Y(j) - g_Y(j)\|_1 + \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} g_Y(j)\cdot\|f_X(i) - g_X(i)\|_1 \\
&= \sum_{j\in\mathcal{B}} \|f_Y(j) - g_Y(j)\|_1 + \sum_{i\in\mathcal{A}} \|f_X(i) - g_X(i)\|_1
\end{aligned}$$

■

We also show a partial converse. If two joint distribution functions are close, then so are their marginal distributions.

Lemma 43 *Let f, g be distribution functions over $\mathcal{A}\times\mathcal{B}$. Then $\|f_X - g_X\|_1 \leq \|f - g\|_1$ and $\|f_Y - g_Y\|_1 \leq \|f - g\|_1$.*

PROOF: $\|f_X - g_X\|_1 = \sum_x |f_X(x) - g_X(x)| \leq \sum_{x,y} |f(x,y) - g(x,y)| = \|f - g\|_1$. The proof of the other half is symmetric. ■

Finally if f is a distribution function which is close to an independent distribution function g, then f is close to the product of its marginals.

Proposition 44 *Let f, g be distribution functions over $\mathcal{A} \times \mathcal{B}$. If g is the distribution function of independent random variables, then $\|f - f_X \times f_Y\|_1 \leq 3\|f - g\|_1$.*

PROOF: By independence, $g = g_X \times g_Y$. Using the triangle inequality, Lemma 42 and Lemma 43, $\|f - f_X \times f_Y\|_1 \leq \|f - g\|_1 + \|g - f_X \times f_Y\|_1$. Since g is independent this equals $\|f - g\|_1 + \|g_X \times g_Y - f_X \times f_Y\|_1 \leq \|f - g\|_1 + \|f_Y - g_2\|_1 + \|f_X - g_X\|_1 \leq 3\|f - g\|_1$. ■

2.6.2 Overview

We will test independence of random variables by testing whether the joint distribution is far from the product of the marginal distributions. The joint distribution is presented to our algorithm as a black-box which is used to construct and approximate marginal distributions and partitions of the probability space into almost-uniform regions. In the first algorithm, the marginal distributions are estimated and an estimate of the joint probability distribution computed. The second algorithm simulates the marginal distributions as separate black-box distributions by using samples from the joint distribution. Finally we give a general algorithm which combines these two.

Given a black-box distribution $\langle \mathcal{A} \times \mathcal{B}, F, f \rangle$, the algorithm in Section 2.6.3, estimates the values of $f_X(x)$ and $f_Y(y)$ for all $y \in \mathcal{B}$ and for all $x \in \mathcal{A}$ for which $f_X(x)$ is greater than a given minimum value. This estimate is obtained by simply sampling, according to *Naive Algorithm*, given in Theorem 85. This algorithm collects samples from a distribution g over S and, by using frequency counts, returns a subset $S' \in S$ and a function g' such that g' is a $(1 + \epsilon)$ approximation to g on every element in S', with high probability. Our algorithm uses the returned estimate directly to compute an approximation to $f_X \times f_Y$ and tests this distribution against f. If f is independent then it should be close to our approximation of $f_X \times f_Y$ and we show, using Theorem 36, that black box F will pass TestIdentity.

The second algorithm, given in Section 2.6.4 uses L_1-Distance-Test (Section 2.2) directly on two distributions. Given black box F which outputs pairs (x, y), we construct

a new black box $F_X \times F_Y$ which on a sample request takes two independent samples $(x, y), (w, z)$ from F and outputs the pair (x, z). If F generates samples according to an independent distribution then F and $F_x \times F_y$ will output the same distribution. In this case F and $F_x \times F_y$ will pass L_1-Distance-Test.

Finally we show in Section 2.6.5 how to combine these algorithms in order to optimize running time for the general case. The key to pairing these two is their complimentary characteristics: the first test is efficient when all elements have high probability (according to the first marginal) the second test is efficient when we have a small upper bound on the first marginal distribution. We will show how to partition the sample space $\mathcal{A} \times \mathcal{B}$ into $\mathcal{A}_0 \times \mathcal{B}$ and $\mathcal{A}_1 \times \mathcal{B}$, apply each test to a different region, and then combine the results.

2.6.3 The heavy prefixes

Given two black box distributions over sets of size n and m respectively, we show that by approximating the probabilities directly and using Theorem 36 we can test random variables X and Y for independence when we consider only a subset of $[n]$ over which X assigns a probability of at least b to each element. The sample complexity of our test is $\tilde{O}((b^{-1} + m)\text{poly}(\epsilon^{-1}))$.

Theorem 45 *There is an algorithm that given a black-box distribution $\langle \mathcal{A} \times \mathcal{B}, f, F \rangle$, a parameter b, and a subset $\mathcal{A}_1 \subset \mathcal{A}$ where $f_X(x) > b$ for all $(x, y) \in \mathcal{A}_1 \times \mathcal{B}$: (1) if f is independent over $(\mathcal{A}_1 \times \mathcal{B})$, it outputs PASS with high probability and (2) if f is not 6ϵ-independent over $(\mathcal{A}_1 \times \mathcal{B})$, it outputs FAIL with high probability. The algorithm uses $\tilde{O}((b^{-1} + |\mathcal{B}|)\text{poly}(\epsilon^{-1}))$ samples, where $m = |\mathcal{B}|$.*

PROOF: The test used is given in Figure 2.5. The correctness of this algorithm follows from the correctness of our test for explicit distributions. If f is independent, then

TestHeavyIndependence$(\langle \mathcal{A} \times \mathcal{B}, f, F \rangle, b, \epsilon)$
(1) Let $(\tilde{f}_X, \mathcal{A}_1)$ = *Naive Algorithm*$(f_X, \mathcal{A}_1, \epsilon/18, \epsilon b/|\log b|)$
(2) Let $(\tilde{f}_Y, \mathcal{B})$ = *Naive Algorithm*$(f_Y, \mathcal{B}, \epsilon/18, \epsilon/(|\mathcal{B}| \log |\mathcal{B}|))$
(3) Call TestIdentity$(f, \tilde{f}_X \times \tilde{f}_Y, \epsilon)$ with domain $\mathcal{A}_1 \times \mathcal{B}$
(4) PASS if TestIdentity returns PASS, otherwise FAIL

Figure 2.5: Algorithm TestHeavyIndependence

$f = f_X \times f_Y$. Since $\tilde{f}_X$ and $\tilde{f}_Y$ approximate the true marginal distributions on $\mathcal{A}_1 \times \mathcal{B}$ it follows that $(\tilde{f}_X \times \tilde{f}_Y)(x, y)$ is a $(1 + \epsilon/9)$ approximation to $f(x, y)$ on $\mathcal{A}_1 \times \mathcal{B}$. From Theorem 36, TestIdentity will pass with high probability. Conversely, if f passes this test with high probability, then we know that f is 6ϵ-close to the independent distribution function $\tilde{f}_X \times \tilde{f}_Y$ on $\mathcal{A}_1 \times \mathcal{B}$.

For the sample complexity we need to sample $\tilde{O}(b^{-1} \cdot \text{poly}(\epsilon^{-1}))$ times for step (1) and $\tilde{O}(|\mathcal{B}| \cdot \text{poly}(\epsilon^{-1}))$ times for step (2). The sample complexity of TestIdent is $\tilde{O}((|\mathcal{B}|/b)^{1/2})$. This brings us to a total sample complexity of $\tilde{O}((b^{-1} + |\mathcal{B}|)\text{poly}(\epsilon^{-1}))$ samples. ■

2.6.4 The light prefixes

In a complimentary case, we now consider testing two black box distributions when we have a bound on the maximum probability of any element. In particular, given as input $\langle \mathcal{A} \times \mathcal{B}, f, F \rangle$ and a bound $\|f_X\|_\infty < 2b/\epsilon$, we give an independence test which uses test L_1-Distance-Test and has sample complexity $\tilde{O}((n^2 m^2 b + n^{2/3})\text{poly}(\epsilon^{-1}))$, where $n = |\mathcal{A}|$ and $m = |\mathcal{B}|$. Formally, we prove:

Theorem 46 *There is an algorithm that given a black-box distribution $\langle \mathcal{A} \times \mathcal{B}, f, F \rangle$ with $\|f_X\|_\infty \leq 2b/\epsilon$: (1) if f is independent, it outputs PASS with high probability and (2) if f is not 3ϵ-independent, it outputs FAIL with high probability. The algorithm uses $\tilde{O}((n^2 m^2 b + n^{2/3}) \cdot \text{poly}(\epsilon^{-1}))$ samples, where $n = |\mathcal{A}|, m = |\mathcal{B}|$.*

This algorithm and proof require us to define some new notation to signify a marginal distribution over a subset of the probability space $\mathcal{A} \times \mathcal{B}$.

Definition 47 *Let a black box distribution $\langle \mathcal{A} \times \mathcal{B}, f, F \rangle$ be given along with partitions $\{\mathcal{A}_0, \ldots, \mathcal{A}_k\}$ and $\{\mathcal{B}_0, \ldots, \mathcal{B}_k\}$. The distribution of f restricted to the subset $\mathcal{A}_i \times \mathcal{B}_j$ is defined as*

$$f^{(i,j)}(x,y) = f(x,y)/f(\mathcal{A}_i \times \mathcal{B}_j) \;\; \forall x, y \in \mathcal{A}_i \times \mathcal{B}_j$$

and the marginal distribution of f restricted to the subset $\mathcal{A}_i \times \mathcal{B}_j$ is defined as

$$\begin{aligned} f_X^{(i,j)}(x) &= \sum_{y \in \mathcal{B}_j} f^{(i,j)}(x,y) \\ f_Y^{(i,j)}(y) &= \sum_{x \in \mathcal{A}_i} f^{(i,j)}(x,y) \end{aligned}$$

When either index (i,j) is intended to include all subsets in a given partition, we signify this with a dot, as in

$$f_X^{(\cdot,j)}(x) = \frac{1}{f(\mathcal{A} \times \mathcal{B}_j)} \sum_{y \in \mathcal{B}_j} f(x,y).$$

TestLightIndependence$(\langle \mathcal{A} \times \mathcal{B}, f, F \rangle, \epsilon)$
(1) Let $(\tilde{f}_Y, \mathcal{B}_S)$ = *Naive Algorithm*$(f_Y, \mathcal{B}, \epsilon/75, (|\mathcal{B}| \log |\mathcal{B}|)^{-1})$.
(2) $\{\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_\ell\}$ =*Flatten*$(\langle \mathcal{B}_S, \tilde{f}_Y \rangle, \epsilon)$; $\mathcal{B}_0 \leftarrow \mathcal{B}_0 \cup (\mathcal{B} \setminus \mathcal{B}_S)$
(3) For $j = 1, \ldots, \ell$ do
(4) If $f(\mathcal{A} \times \mathcal{B}_j) \geq \epsilon/(2\ell)$ then
(5) If $\|f^{(\cdot,j)} - f_X^{(\cdot,j)} \times f_Y^{(\cdot,j)}\|_1 \geq \epsilon$, then FAIL.
(6) Let k be such that $f(\mathcal{A} \times \mathcal{B}_k) > \epsilon/(4\ell)$.
(7) For $j = 1, \ldots, \ell$ do
(8) If $f(\mathcal{A} \times \mathcal{B}_j) > \epsilon/(4\ell)$, then
(9) If $\|f_X^{(\cdot,j)} - f_X^{(\cdot,k)}\|_1 \geq \epsilon$, then FAIL.
(10) PASS.

Figure 2.6: Algorithm TestLightIndependence

PROOF OF THEOREM 46: The algorithm appears in Figure 2.6. In steps (4) and (8), we distinguish between $f(\mathcal{A} \times \mathcal{B}_j) \geq \epsilon/(2\ell)$ and $f(\mathcal{A} \times \mathcal{B}_j) \leq \epsilon/(4\ell)$ by taking $\tilde{O}(\ell/\epsilon)$ samples of F. This guarantees that we need to take only $O(\text{poly}(\log(nm))\ell/\epsilon)$ samples

of F for every sample of $f^{(\cdot,j)}$ required in step (5) and step (9), by re-sampling F until we obtain a member of the required set (similarly step (6) guarantees this for sampling $f^{(\cdot,k)}$.)

The projections appearing in step (5) are sampled by sampling the respective distribution from F and ignoring a coordinate. The k in step (6) can be obtained with $\tilde{O}(\text{poly}(m, \log n, 1/\epsilon))$ samples.

The test for the distribution difference in step (5) is done by using Theorem 24 with parameter ϵ and the distributions $f^{(\cdot,j)}$ and $f_1^{(\cdot,j)} \times f_2^{(\cdot,j)}$; the bound on the L_∞ norm of the distributions involved will be given below. The test for the difference in step (9) is done similarly, but this time using Theorem 20 with parameter ϵ.

Notice that $\|f^{(\cdot,j)}\|_\infty \leq 2b/\epsilon$ for every $\mathcal{B}_j$ (because of the bound on $\|f_X\|_\infty$), and that $\|f_2^{(\cdot,j)}\|_\infty \leq (1+3\epsilon)|\mathcal{B}_j|^{-1}$.

The total sample complexity for steps (3)–(5) is given by $\log|\mathcal{B}|$ times the sample complexity for iteration j. The sample complexity of the latter is given by Theorem 24, which is $\tilde{O}((1+3\epsilon)\cdot(|\mathcal{A}||\mathcal{B}_j|)^2(b\cdot b)|\mathcal{B}_j|^{-1}\cdot\epsilon^{-5})$, times the $\tilde{O}(\ell/\epsilon)$ for sampling from the restrictions to the buckets. This clearly dominates the sample complexity for step (6), and the sample complexity for steps (7)–(9), which is $\tilde{O}(|\mathcal{A}|^{2/3}\epsilon^{-5})$ by multiplying the estimate of Theorem 20, the sample complexity of the restricted distributions, and the number of iterations.

As for correctness, if f is independent then it readily follows that the algorithm accepts, while on the other hand, by an application of Lemma 33 to the partition defined in step (2), $\{\mathcal{B}_0, \ldots, \mathcal{B}_\ell\}$, if the distribution pairs compared in step (5) and step (9) are indeed all ϵ-close, then f is 3ϵ-independent. ■

2.6.5 Putting them together

We now show how algorithms TestHeavyIndependence and TestLightIndependence can be combined to give a general independence test with sublinear sample complexity.

Let ϵ be given and let $m = n^\beta$. Let $0 < \alpha < 1$ be a parameter to be determined later. Let S' denote the set of indices in the first coordinate with probability mass at least $n^{-\alpha}$, which we will also refer to as the heavy prefixes. Formally, let $S' = \{i \in [n] \mid f_X(i) \geq n^{-\alpha}\}$. We would like to be able to determine the set S' exactly, but we cannot do this by only sampling. So we define a new set which will contain all of set S' and no elements (with high probability) which are far from being in set S'. Define $S'' = \{i \in [n] \mid f_X(i) \geq \frac{1}{2}n^{-\alpha}\}$. Using a total of $O(n^\alpha \epsilon^{-2} \log n)$ samples, we can estimate $f_X(i), i \in S''$ by $\tilde{f_X}(i)$ to within an $\epsilon/75$ factor using Theorem 85. Let $\tilde{S}$ be the set of all i for which $\tilde{f_X}(i) \geq \frac{2}{3}n^{-\alpha}$. Then with high probability, $S' \subset \tilde{S} \subset S''$. We then take $\mathcal{A}_1 = \tilde{S}$ and $\mathcal{A}_0 = \mathcal{A} \setminus \tilde{S}$.

Our main idea is to first test that f is independent conditioned on the set of heavy prefixes and then to test that f is independent conditioned on the set of light prefixes. To create these conditionings, we first distinguish (using $\tilde{O}(\epsilon^{-1})$ samples) between $f_X(\tilde{S}) \geq \epsilon$ and $f_X(\tilde{S}) \leq \epsilon/2$. If the latter case occurs, then the distribution conditioned on the heavy prefixes cannot contribute more than $\epsilon/2$ to f's distance from independence. Otherwise, if we are guaranteed that the second case does not occur, we can simulate the distribution for $f^{(1,\cdot)}$ easily—we sample from F until we find a member of $\mathcal{A}_1 \times \mathcal{B}$ which we output; this takes $O(\epsilon^{-1} \log(nm))$ queries with a high enough success probability. We then apply an independence test that works well for heavy prefixes to $f^{(1,\cdot)}$.

Next we distinguish between $f_X(\mathcal{A}_0) \geq \epsilon$ and $f_X(\mathcal{A}_0) \leq \epsilon/2$. Again if the latter occurs, then the distribution conditioned on light elements can contribute at most $\epsilon/2$ to the distance from independence. Otherwise, if the latter does not occur, as before we simulate the distribution $f^{(0,\cdot)}$, and use it with a test that works well for distributions

restricted to light prefixes (they will still remain light enough provided that $f_X(\mathcal{A}_0) \geq \epsilon/2$).

Our test proceeds by merging the testing over light and heavy prefixes and then applying Theorem 20 to ensure the consistency of the distributions.

Theorem 48 *There is an algorithm that given a black box distribution $\langle \mathcal{A} \times \mathcal{B}, f, F \rangle$ and an $\epsilon > 0$: (1) if f is independent, it outputs PASS with high probability and (2) if f is not 10ϵ-independent, it outputs FAIL with high probability. The algorithm uses $\tilde{O}(n^{2/3}m^{1/3} \cdot \text{poly}(\epsilon^{-1}))$ samples, where $n = |\mathcal{A}|$ and $m = |\mathcal{B}|$, $n > m$.*

TestIndependence $(\langle \mathcal{A} \times \mathcal{B}, f, F \rangle, \epsilon)$
(0) $n = |\mathcal{A}|$, $m = |\mathcal{B}|$
(1) Let β be such that $m = n^{\beta}$, and set $\alpha = (2+\beta)/3$.
(2) Let $(\tilde{f}_1, \mathcal{A}_1) =$ *Naive Algorithm*$(f_X, \mathcal{A}, \epsilon/75, n^{-\alpha})$
    Let $\mathcal{A}_0 = \mathcal{A} \setminus \mathcal{A}_1$
(3) If $f_X(\mathcal{A}_1) > \epsilon$ then
(4) If *TestHeavyIndependence* $(\langle \mathcal{A}_1 \times \mathcal{B}, f, F \rangle, n^{-\alpha}, \epsilon)$ fails then FAIL
(5) If $f_X(\mathcal{A}_0) > \epsilon$ then
(6) If *TestLightIndependence* $(\langle \mathcal{A}_0 \times \mathcal{B}, f, F \rangle, \epsilon)$ fails then FAIL.
(7) If both $f_X(\mathcal{A}_0) > \epsilon$ and $f_X(\mathcal{A}_1) > \epsilon$ then
(8) If $\|f_2^{(0,\cdot)} - f_2^{(1,\cdot)}\|_1 > \epsilon$ then FAIL
(9) PASS

Figure 2.7: Algorithm TestIndependence

PROOF: The algorithm is given in Figure 2.7. Steps (3), (5) and (7) use sampling to distinguish between the cases where the respective quantities are at least ϵ and the cases where they are at most $\epsilon/2$. Step (4) (if required) is done by using Theorem 45, and step (6) is done by using Theorem 46; by the choice of α in step (1), the number of samples in both is $\tilde{O}(n^{2/3}m^{1/3} \cdot \text{poly}(\epsilon^{-1}))$ times the $O(\epsilon^{-1}\log(nm))$ samples required for choosing from the restricted distributions (a factor which does not change the above estimate).

In step (8) the two distributions are fed into the algorithm of Theorem 20, with parameters set to guarantee failure if these distributions are more than ϵ-apart; this uses a number of queries that is dominated by the terms in the rest of the algorithm.

It is clear that if f is independent then the test will accept with high probability. We now prove that if the test accepts with high probability then f is at least 10ϵ-independent.

If steps (4), (6) and (8) are performed and none of the above tests fails, then by a final application of Lemma 30, with a partition of $\{\mathcal{A}_0 \times \mathcal{B}, \mathcal{A}_1 \times \mathcal{B}\}$, we get that our distribution is at least 10ϵ-independent (because step (8) guarantees that the U-reduction is not more than ϵ-far from being independent). If steps (4) and (8) are not performed, then $f(\mathcal{A}_1 \times \mathcal{B}) < \epsilon$, so it contributes no more than ϵ to the distance of A from independence, and so step (6) is sufficient to guarantee 4ϵ-independence. Similarly 4ϵ-independence holds if steps (6) and (8) are not performed since in this case $f(\mathcal{A}_0 \times \mathcal{B})$ is small. This covers all possible cases and concludes the proof. ■

2.7 Subsequent Work

The independence problem has been extended to general product distributions and collections of distributions [ADK15], [LRR13], [DK16].

For the two-dimensional case, Diakonikolas and Kane give a lower bound

Theorem 49 ([DK16]) *Let $n \geq m \geq 2$ and $\epsilon > 0$ be a sufficiently small constant. Then, any algorithm that draws samples from a distribution p on $[n] \times [m]$ and, with probability at least $2/3$, distinguishes between the case that the coordinates of p are independent and the case where p is ϵ-far from any product distribution must use $\Omega(\max(\sqrt{nm}/\epsilon^2, n^{2/3}m^{1/3}/\epsilon^{4/3}))$ samples.*

They also give an algorithm that matches this lower bound for the two-dimensional case, and an extension of the algorithm to general dimensions which procedes by recur-

sively reducing the general problem to a two-dimensional one.

CHAPTER 3

TESTING PROPERTIES OF DATA

We now consider sublinear algorithms for testing properties of data. In particular, we consider Markov Chains given as input matrices and give tests for mixing properties of these input chains. We show how to test whether iterating a Markov chain for t steps causes it to reach a distribution close to the stationary distribution. Our testing algorithm works by following $\tilde{O}(tn^{5/3})$ edges in the chain. When the Markov chain is represented in a convenient way, this test remains sublinear in the size of a dense enough Markov chain for small t.We then investigate two notions of being *close* to a rapidly mixing Markov chain that fall within the framework of property testing, and show how to test that a Markov chain is close to a Markov chain that mixes in t steps by following only $\tilde{O}(tn^{2/3})$ edges. In the case of Markov chains that come from directed graphs and pass our test, our theorems show the existence of a directed graph that is close to the original one and rapidly mixing.

Related Work Our results fall within the various frameworks of property testing as defined in [RS96, GGR96, GR97, EKK+98, PR99]. A related work of Kannan and Yao [KY91] outlines a program checking framework for certifying the randomness of a program's output. In their model, one does not assume that samples from the input distribution are independent.

The conductance [SJ89] of a graph is known to be closely related to expansion and rapid-mixing properties of the graph [Kan94][SJ89]. Frieze and Kannan [FK99] show, given a graph G with n vertices and α, one can approximate the conductance of G to within additive error α in time $O(n2^{\tilde{O}(1/\alpha^2)})$. Their techniques also yield an $O(2^{\mathrm{poly}(1/\epsilon)})$ time test which determines whether an adjacency matrix of a graph can be changed in at most ϵ fraction of the locations to get a graph with high conductance. However, for the purpose of testing whether an n-vertex, m-edge graph is rapid mixing, we would need

to approximate its conductance to within $\alpha = O(m/n^2)$; thus only when $m = \Theta(n^2)$ would it run in $O(n)$ time.

It is known that mixing [SJ89, Kan94] is related to the separation between the two largest eigenvalues [Alo86]. Standard techniques for approximating the eigenvalues of a dense $n \times n$ matrix run in $\Theta(n^3)$ flops and consume $\Theta(n^2)$ words of memory [GV96]. However, for a sparse $n \times n$ *symmetric* matrix with m nonzero entries, $n \leq m$, "Lanczos algorithms" [Par98] accomplish the same task in $\Theta(n[m + \log n])$ flops, consuming $\Theta(n + m)$ storage. Furthermore, it is found in practice that these algorithms can be run for far fewer, even a constant number, of iterations while still obtaining highly accurate values for the outer and inner few eigenvalues. Our test for rapid mixing of a Markov chain runs more slowly than the algorithms that are used in practice except on fairly dense graphs ($m \gg tn^{5/3} \log n$). However, our test is more efficient than algorithms whose behavior is mathematically justified at every sparsity level. Our faster, but weaker, tests of various altered definitions of "rapid mixing," are more efficient than the current algorithms used in practice.

Outline of Chapter In Section 3.1 we give necessary notation and definitions. We also define our Markov chain model and notions of closeness. In Section 3.2 we give two simple tests for mixing and a relaxed mixing property, which we define. In Section 3.3 we give our main result of this chapter, a property test for mixing.

3.1 Preliminaries/Notation

Let a Markov chain be represented by the transition probability matrix M. The uth state of M corresponds to an n-vector $\vec{e}_u = (0, \ldots, 1, \ldots, 0)$, with a one in only the uth location and zeroes elsewhere. The distribution generated by t-step random walks starting at state u is denoted as a vector-matrix product $\vec{e}_u M^t$.

Instead of computing such products in our algorithms, we assume that our L_1-Distance-Test has access to an oracle, `next_node` which on input of the state u responds with the state v with probability $M(u,v)$. Given such an oracle, the distribution $\vec{e}_u^T M^t$ can be generated in $O(t)$ steps. Furthermore, the oracle itself can be realized in $O(\log n)$ time per query, given linear preprocessing time to compute the cumulative sums $M_c(j,k) = \sum_{i=1}^{k} M(j,i)$. The oracle can be simulated on input u by producing a random number α in $[0,1]$ and performing binary search over the uth row of M_c to find v such that $M_c(u,v) \leq \alpha \leq M_c(u,v+1)$. It then outputs state v. Note that when M is such that every row has at most d nonzero terms, slight modifications of this yield an $O(\log d)$ implementation consuming $O(n+m)$ words of memory if M is $n \times n$ and has m nonzero entries. Improvements of the work given in [Wal77] can be used to prove that in fact constant query time is achievable with space consumption $O(n+m)$ for implementing `next_node` given linear preprocessing time.

We say that two states u and v are (ϵ, t)-*close* if the distribution generated by t-step random walks starting at u and v are within ϵ in the L_1 norm, *i.e.* $|\vec{e}_u M^t - \vec{e}_v M^t| < \epsilon$.[1] Similarly we say that a state u and a distribution $\vec{s}$ are (ϵ, t)-close if $|\vec{e}_u M^t - \vec{s}| < \epsilon$. We say M is (ϵ, t)-*mixing* if all states are (ϵ, t)-close to the same distribution:

Definition 50 *A Markov chain M is* (ϵ, t)-mixing *if a distribution $\vec{s}$ exists such that for all states u, $|\vec{e}_u M^t - \vec{s}| \leq \epsilon$.*

For example, if M is $(\epsilon, O(\log n \log 1/\epsilon))$-mixing, then M is *rapidly-mixing* [SJ89]. It can be easily seen that if M is (ϵ, t_0)-mixing then it is (ϵ, t) mixing for all $t > t_0$.

If walks on a Markov chain do converge, it is to the stationary distribution. We define the notion of an average t-step distribution here, which is similar in spirit to a stationary distribution but has the advantage of being easily generated under our model:

[1] In this chapter we denote L_1 norm simply by $|x|$ since there is no ambiguity with other vector norms.

Definition 51 *The* average t-step distribution*,* $\vec{s}_{M,t}$ *of a Markov chain* M *with* n *states is the distribution*

$$\vec{s}_{M,t} = \frac{1}{n} \sum_u \vec{e}_u M^t.$$

This distribution can be easily generated by picking u uniformly from $[n]$ and walking t steps from state u. In an (ϵ, t)-mixing Markov chain, the average t-step distribution is ϵ-close to the stationary distribution. In a Markov chain that is not (ϵ, t)-mixing, this is not necessarily the case.

The tests given in Sections 3.2 and 3.3 assume access to L_1-Distance-Test(u, v, ϵ, δ) which given oracle access to distributions $\vec{e}_u, \vec{e}_v$ over the same n element set decides whether $|\vec{e}_u - \vec{e}_v| \leq f(\epsilon)$ or if $|\vec{e}_u - \vec{e}_v| > \epsilon$ with confidence $1 - \delta$. The time complexity of L_1_test is $T(n, \epsilon, \delta)$, and f is the *gap* of the tester. The implementation of L_1-Distance-Test given earlier in Section 2.1 has gap $f(\epsilon) = \epsilon/(4\sqrt{n})$, and time complexity $T = \tilde{O}(\frac{1}{\epsilon^4} n^{2/3}) \log(1/\delta)$.

3.2 A test for mixing and a test for almost-mixing

In this section, we show how to decide if a Markov chain is (ϵ, t)-mixing; then we define and solve a natural relaxation of that problem.

In order to test that M is (ϵ, t)-mixing, one can use L_1-Distance-Test to compare each distribution $\vec{e}_u M^t$ with $\vec{s}_{M,t}$, with error parameter ϵ and confidence δ/n. The running time is $O(nt \cdot T(n, \epsilon, \delta/n))$. If every state is $(f(\epsilon)/2, t)$-close to some distribution $\vec{s}$, then $\vec{s}_{M,t}$ is $f(\epsilon)/2$-close to $\vec{s}$. Therefore every state is (ϵ, t)-close to $\vec{s}_{M,t}$. On the other hand, if there is no distribution that is (ϵ, t)-close to all states, then, in particular, $\vec{s}_{M,t}$ is not (ϵ, t)-close to at least one state. This yields our first test for mixing:

Theorem 52 *Let* M *be a Markov chain. Given* L_1*-Distance-Test with time complexity* $T(n, \epsilon, \delta)$ *and gap* f *and an oracle for* next_node*, there exists a test with time com-*

plexity $O(nt \cdot T(n, \epsilon, \delta/n))$ *with the following behavior: If* M *is* $(f(\epsilon)/2, t)$*-mixing then* $\Pr[M \text{ passes}] > 1 - \delta$*; if* M *is not* (ϵ, t)*-mixing then* $\Pr[M \text{ passes}] < \delta$.

For the implementation of L_1-Distance-Test given in Section 2.1 the running time is $O(\frac{1}{\epsilon^4} n^{5/3} t \log n \log \frac{n}{\delta})$. It distinguishes between chains which are $\epsilon/(4\sqrt{n})$ mixing and those which are not ϵ-mixing. The running time is sublinear in the size of M if $t \in o(n^{1/3}/\log(n))$.

A relaxation of this procedure is testing that *most* starting states reach the same distribution after t steps. If $(1 - \rho)$ fraction of the states u of a given M satisfy $|\vec{s} - \vec{e}_u M^t| \leq \epsilon$, then we say that M is (ρ, ϵ, t)*-almost mixing*. By picking $O(1/\rho \cdot \ln(1/\delta))$ starting states uniformly at random, and testing their closeness to $\vec{s}_{M,t}$ we have:

Theorem 53 *Let* M *be a Markov chain. Given* L_1*-Distance-Test with time complexity* $T(n, \epsilon, \delta)$ *and gap* f *and an oracle for* `next_node`*, there exists a test with time complexity* $O((t/\rho) T(n, \epsilon, \delta\rho) \log \frac{1}{\delta})$ *with the following behavior: If* M *is* $(\rho, f(\epsilon)/2, t)$*-almost mixing then* $\Pr[M \text{ passes}] > 1 - \delta$*; If* M *is not* (ρ, ϵ, t)*-almost mixing then* $\Pr[M \text{ passes}] < \delta$.

3.3 A property tester for mixing

The main result of this section is a test that determines if a Markov chain's matrix representation can be changed in an ϵ fraction of the non-zero entries to turn it into a $(4\epsilon, 2t)$-mixing Markov chain. This notion falls within the scope of property testing [RS96, GGR96, GR97, EKK+98, PR99], which in general takes a set S with distance function Δ and a subset $P \subseteq S$ and decides if an elements $x \in S$ is in P or if it is far from every element in P, according to Δ. For the Markov chain problem, we take as our set S all matrices M of size $n \times n$ with at most d non-zero entries in each row. The distance function is given by the fraction of non-zero entries in which two matrices

differ, and the difference in their average t-step distributions.

Definition 54 *Let M_1 and M_2 be n-state Markov chains with at most d non-zero entries in each row. Define distance function $\Delta(M_1, M_2) = (\epsilon_1, \epsilon_2)$ iff M_1 and M_2 differ on $\epsilon_1 dn$ entries and $|\vec{s}_{M_1,t} - \vec{s}_{M_2,t}| = \epsilon_2$. We say that M_1 and M_2 are (ϵ_1, ϵ_2)-close if $\Delta(M_1, M_2) \leq (\epsilon_1, \epsilon_2)$.*[2]

A natural question is whether all Markov chains are ϵ_1, ϵ_2-close to an (ϵ, t)-mixing Markov chain, for certain parameters of ϵ. For constant ϵ and $t = O(\log n)$, one can show that every strongly-connected Markov chain is $(\epsilon, 1)$-close to another Markov chain which (ϵ, t)-mixes. However, the situation changes when asking whether there is an (ϵ, t)-mixing Markov chain that is close both in the matrix representation and in the average t-step distribution: specifically, it can be shown that there exist constants $\epsilon, \epsilon_1, \epsilon_2 < 1$ and Markov chain M for which no Markov chain is both (ϵ_1, ϵ_2)-close to M and $(\epsilon, \log n)$-mixing. In fact, when ϵ_1 is small enough, the problem becomes nontrivial even for $\epsilon_2 = 1$. The Markov chain corresponding to random walks on the n-cycle provides an example which is not $(t^{-1/2}, 1)$-close to any (ϵ, t)-mixing Markov chain.

Motivation As before, our algorithm proceeds by taking random walks on the Markov chain and comparing final distributions by using the L_1 distance tester. We define three types of states. First a *normal* state is one from which a random walk arrives at nearly the average t-step distribution. In the discussion which follows, t and ϵ denote constant parameters fixed as input to the algorithm `TestMixing`.

Definition 55 *Given a Markov Chain M, a state u of the chain is* normal *if it is (ϵ, t)-close to $\vec{s}_{M,t}$. That is if $|\vec{e}_u M^t - \vec{s}_{M,t}| \leq \epsilon$. A state is* bad *if it is not normal.*

Testing normality of a single state requires time $O(t \cdot T(n, \epsilon, \delta))$. Using this definition the first two algorithms given in this section can be described as testing whether all (*resp.*

[2]We say $(x, y) \leq (a, b)$ iff $x \leq a$ and $y \leq b$

most) states in M are *normal*. The test we give here additionally needs to distinguish states which not only produce random walks which arrive near $\vec{s}_{M,t}$ but which have low probability of visiting a bad state. We call such states *smooth* states:

Definition 56 *A state $\vec{e}_u$ in a Markov chain M is* smooth *if* (a) *u is (ϵ, τ)-close to $\vec{s}_{M,t}$ for $\tau = t, \ldots, 2t$ and* (b) *the probability that a $2t$-step random walk starting at $\vec{e}_u$ visits a bad state is at most ϵ.*

Testing smoothness of a state requires $O(t^2 \cdot T(n, \epsilon, \delta))$ time. Our property test merely verifies by random sampling that most states are smooth.

The test Figure 3.1 gives an algorithm which on input Markov chain M and parameter ϵ determines whether at least $(1 - \epsilon)$ fraction of the states of M are smooth according to two distributions: the uniform and the average t-step distribution. Assuming access to L_1-Distance-Test with complexity $T(n, \epsilon, \delta)$, this test runs in time $O(\epsilon^{-2} t^2 T(n, \epsilon, \frac{1}{6t}))$.

```
TestMixing(M, t, ε)
Let k = Θ(1/ε)
Select k states u_1,...,u_k uniformly
Select k states u_{k+1},...,u_{2k} according to s_{M,t}
For i = 1 to 2k
  u = e_{u_i}
  For w = 1 to O(1/ε)
    For j = 1 to 2t
      u = next_node(M, u)
      L_1-Distance-Test(e_u M^t, s_{M,t}, ε, 1/6t)
    End
  End
  For τ = t to 2t
    L_1-Distance-Test(e_{u_i} M^τ, s_{M,t}, ε, 1/3t)
  End
Pass if all tests pass
```

Figure 3.1: Algorithm TestMixing

The main lemma of this section says that any Markov chain which passes our test is $(2\epsilon, 2\epsilon)$-close to a $(4\epsilon, 2t)$-mixing Markov chain. First we give the modification

Definition 57 *F is a function from $n \times n$ matrices to $n \times n$ matrices such that $F(M)$ returns $\widetilde{M}$ by modifying the rows corresponding to bad states of M to $\vec{e}_u$ where u is a smooth state.*

An important feature of the transformation F is that it does not affect the distribution of random walks originating from smooth states very much.

Lemma 58 *Given a Markov chain M and any state $u \in M$ which is smooth. If $\widetilde{M} = F(M)$ then for any time $t \leq \tau \leq 2t$, $|\vec{e}_u M^\tau - \vec{e}_u \widetilde{M}^\tau| \leq \epsilon$ and $|\vec{s}_{M,t} - \vec{e}_u \widetilde{M}^\tau| \leq 2\epsilon$.*

PROOF: Define Γ as the set of all walks of length τ from u in M. Partition Γ into Γ_B and $\bar{\Gamma}_B$ where Γ_B is the subset of walks which visit a bad state. Let $\chi_{w,i}$ be an indicator function which equals 1 if walk w ends at state i, and 0 otherwise. Let weight function $W(w)$ be defined as the probability that walk w occurs. Finally define the primed counterparts Γ', etc. for the Markov chain $\widetilde{M}$. Now the ith element of $\vec{e}_u M^\tau$ is $\sum_{w \in \Gamma_B} \chi_{w,i} \cdot W(w) + \sum_{w \in \bar{\Gamma}_B} \chi_{w,i} \cdot W(w)$. A similar expression can be written for each element of $\vec{e}_u \widetilde{M}^\tau$. Since $W(w) = W'(w)$ whenever $w \in \bar{\Gamma}_B$ it follows that $|\vec{e}_u M^\tau - \vec{e}_u \widetilde{M}^\tau| \leq \sum_i \sum_{w \in \Gamma_B} \chi_{w,i} |W(w) - W'(w)| \leq \sum_i \sum_{w \in \Gamma_B} \chi_{w,i} W(w) \leq \epsilon$.

Additionally, since $|\vec{s}_{M,t} - \vec{e}_u M^\tau| \leq \epsilon$ by the definition of smooth, it follows that $|\vec{s}_{M,t} - \vec{e}_u \widetilde{M}^\tau| \leq |\vec{s}_{M,t} - \vec{e}_u M^\tau| + |\vec{e}_u M^\tau - \vec{e}_u \widetilde{M}^\tau| \leq 2\epsilon$. ■

We can now prove the main lemma:

Lemma 59 *If according to the uniform distribution and the distribution $\vec{s}_{M,t}$, $(1-\epsilon)$ fraction of the states of a Markov chain M are smooth, then the matrix M is $(2\epsilon, 2\epsilon)$-close to a matrix $\widetilde{M}$ which is $(4\epsilon, 2t)$-mixing.*

PROOF: Let $\widetilde{M} = F(M)$. $\widetilde{M}$ and M differ on at most $\epsilon n(d+1) < 2\epsilon(nd)$ entries. This gives the first part of our distance bound. For the second we analyze $|\vec{s}_{M,t} - \vec{s}_{\widetilde{M},t}| = \frac{1}{n} \sum_u |\vec{e}_u M^t - \vec{e}_u \widetilde{M}^t|$ as follows. We analyze two components of the sum separately: for each of the smooth nodes u, Lemma 58 says that $|\vec{e}_u M^t - \vec{e}_u \widetilde{M}^t| \leq \epsilon$; nodes which are

not smooth account for at most ϵ fraction of the nodes in the sum, and thus can contribute no more than ϵ absolute weight to the distribution $\vec{s}_{\widetilde{M},t}$. The sum can be bounded now by $|\vec{s}_{M,t} - \vec{s}_{\widetilde{M},t}| \leq \frac{1}{n}((1-\epsilon)n\epsilon + \epsilon n) \leq 2\epsilon$.

In order to show that $\widetilde{M}$ is $(4\epsilon, 2t)$-mixing, we prove that for every state u, $|\vec{s}_{M,t} - \vec{e}_u M^{2t}| \leq 4\epsilon$. The proof considers three cases: u smooth, u bad, and u normal. The last case is the most involved.

If u is smooth in the Markov chain M, then Lemma 58 immediately tells us that $|\vec{s}_{M,t} - \vec{e}_u \widetilde{M}^{2t}| \leq 2\epsilon$. Similarly if u is bad in the Markov chain M, then in the chain $\widetilde{M}$ any path starting at u transitions to a smooth state v in one step. Since $|\vec{s}_{M,t} - \vec{e}_v \widetilde{M}^{2t-1}| \leq 2\epsilon$ by Lemma 58, the desired bound follows.

If $\vec{e}_u$ is a normal state which is not smooth we need a more involved analysis of the distribution $|\vec{e}_u \widetilde{M}^{2t}|$. We divide Γ, the set of all $2t$-step walks in M starting at u, into three sets, which we consider separately.

For the first set take $\Gamma_B \subseteq \Gamma$ to be the set of walks which visit a bad node before time t. Let $\vec{d}_b$ be the distribution over endpoints of these walks, that is, let $\vec{d}_b$ assign to state i the probability that any walk $w \in \Gamma_B$ ends at state i. Let $w \in \Gamma_B$ be any such walk. If w visits a bad state at time $\tau < t$, then in the new Markov chain $\widetilde{M}$, w visits a smooth state v at time $\tau + 1$. Another application of Lemma 58 implies that $|\vec{e}_v \widetilde{M}^{2t-\tau-1} - \vec{s}_{M,t}| \leq 2\epsilon$. Since this is true for all walks $w \in \Gamma_B$, we find $|\vec{d}_b - \vec{s}_{M,t}| \leq 2\epsilon$.

For the second set, let $\Gamma_S \subseteq \Gamma \setminus \Gamma_B$ be the set of walks not in Γ_B which visit a smooth state at time t. Let $\vec{d}_s$ be the distribution over endpoints of these walks. Any walk $w \in \Gamma_S$ is identical in the chains M and $\widetilde{M}$ up to time t, and then in the chain $\widetilde{M}$ visits a smooth state v at time t. Thus since $|\vec{e}_v \widetilde{M}^t - \vec{s}_{M,t}| \leq 2\epsilon$, we have $|\vec{d}_s - \vec{s}_{M,t}| \leq 2\epsilon$.

Finally let $\Gamma_N = \Gamma \setminus (\Gamma_B \cup \Gamma_S)$, and let $\vec{d}_n$ be the distribution over endpoints of walks in Γ_N. Γ_N consists of a subset of the walks from a normal node u which do not visit a smooth node at time t. By the definition of normal, u is (ϵ, t)-close to $\vec{s}_{M,t}$ in the

Markov chain M. By assumption at most ϵ weight of $\vec{s}_{M,t}$ is assigned to nodes which are not smooth. Therefore $|\Gamma_N|/|\Gamma|$ is at most $\epsilon + \epsilon = 2\epsilon$.

Now define the weights of these distributions as ω_b, ω_s and ω_n. That is ω_b is the probability that a walk from u in M visits a bad state before time t. Similarly ω_s is the probability that a walk does not visit a bad state before time t, but visits a smooth state at time t, and ω_n is the probability that a walk does not visit a bad state but visits a normal, non-smooth state at time t. Then $\omega_b + \omega_s + \omega_n = 1$. Finally $|\vec{e_u}\widetilde{M}^{2t} - \vec{s}_{M,t}| = |\omega_b\vec{d_b} + \omega_s\vec{d_s} + \omega_n\vec{d_n} - \vec{s}_{M,t}| \leq \omega_b|\vec{d_b} - \vec{s}_{M,t}| + \omega_s|\vec{d_s} - \vec{s}_{M,t}| + \omega_n|\vec{d_n} - \vec{s}_{M,t}| \leq (\omega_b + \omega_s)\max\{|\vec{d_b} - \vec{s}_{M,t}|, |\vec{d_s} - \vec{s}_{M,t}|\} + \omega_n|\vec{d_n} - \vec{s}_{M,t}| \leq 4\epsilon.$ ■

Given this, we finally can show our main theorem:

Theorem 60 *Let M be a Markov chain. Given L_1-Distance-Test with time complexity $T(n, \epsilon, \delta)$ and gap f and an oracle for* `next_node`*, there exists a test such that if M is $(f(\epsilon), t)$-mixing then the test passes with probability at least $2/3$. If M is not $(2\epsilon, 1.01\epsilon)$-close to any $\widetilde{M}$ which is $(4\epsilon, 2t)$-mixing then the test fails with probability at least $2/3$. The runtime of the test is $O(\frac{1}{\epsilon^2} \cdot t^2 \cdot T(n, \epsilon, \frac{1}{6t}))$.*

PROOF: Since in any Markov chain M which is (ϵ, t)-mixing all states are smooth, M passes this test with probability at least $(1 - \delta)$. Furthermore, any Markov chain with at least $(1 - \epsilon)$ fraction of smooth states is $(2\epsilon, 2\epsilon)$-close to a Markov chain which is $(4\epsilon, 2t)$-mixing, by Lemma 59. ■

3.4 Extension to sparse graphs and uniform distributions

The property test can also be made to work for general sparse Markov chains by a simple modification to the testing algorithms. Consider Markov chains with at most $m \in o(n^2)$ nonzero entries, but with no nontrivial bound on the number of nonzero entries per row. Then the definition of the distance should be modified to $\Delta(M_1, M_2) = (\epsilon_1, \epsilon_2)$ if M_1

and M_2 differ on $\epsilon_1 \cdot m$ entries and $|\vec{s}_{M_1,t} - \vec{s}_{M_2,t}| = \epsilon_2$. The above test does not suffice for testing that M is (ϵ_1, ϵ_2)-close to an (ϵ, t)-mixing Markov chain $\widetilde{M}$, since in our proof, the rows corresponding to bad states may have many nonzero entries and thus M and $\widetilde{M}$ may differ in a large fraction of the nonzero entries. However, let D be a distribution on states in which the probability of each state is proportional to cardinality of the support set of its row. Natural ways of encoding this Markov chain allow constant time generation of states according to D. By modifying the test in Figure 3.1 to also test that most states according to D are smooth, one can show that M is close to an (ϵ, t)-mixing Markov chain $\widetilde{M}$.

Because of our ability to test ϵ-closeness to the *uniform* distribution in $O(n^{1/2}\epsilon^{-2})$ steps [GR00], it is possible to speed up our test for mixing for those Markov chains known to have uniform stationary distribution, such as Markov chains corresponding to random walks on regular graphs. An ergodic random walk on the vertices of an undirected graph instead may be regarded (by looking at it "at times $t + 1/2$") as a random walk on the *edge-midpoints* of that graph. The stationary distribution on edge-midpoints always exists and is uniform. So, for undirected graphs we can speed up mixing testing by using a tester for closeness to uniform distribution.

3.5 Reflections

We have given a test for rapid mixing of Markov chains represented by a directed graph, a test for a relaxation of mixing, and a property test for rapid mixing. Although tests for mixing based on eigenvalue representations are more efficient for certain problems than the test we gave in Section 3.2, the property test given in Section 3.3 is more efficient by a factor of $O(n)$ and tests a natural and interesting relaxation of the notion of mixing. In particular it may be the case that a network has been constructed and one wants to know how much effort must be expended in making the network into a rapidly mixing

Markov chain. Our test will give this information.

Natural questions still remain in this area, though. Our test is limited by dependence on directed edges: given an undirected input graph, we cannot test for closeness to a rapidly mixing undirected Markov chain. It would also be interesting to give a test for the related problem of whether an undirected graph is close to an expander or far from any expanding graph. We hope that the techniques given here provide help towards answering these questions.

3.6 Subsequent Work

Graph property testing is also a rich and productive field. For a survey of results see Goldreich's survey on Property Testing [Gol17]. Expansion of graphs in particular was cited as an interesting property for testing originally by Goldreich and Ron [GR00] and efficient testing algorithms have since been given by [NS10], [CS10] and [KS08].

We hope that continued research in this area will provide more results and give further applications of our results on distribution property testing.

CHAPTER 4

TESTING WITH UNTRUSTED HELP

Consider a scenario in which a company wants the optimal solution to a very large combinatorial optimization problem and hires an independent contractor to actually find the solution. The company would like to trust that the *value* of the solution is feasible, but might not care about the structure of the solution itself. In particular, they would like to have a quick and simple test that checks if the contractor has a good solution by only inspecting a very small portion of the solution itself.

For example, say a mail order company needs a trucking company to handle all of its shipping orders, which involves moving large numbers of boxes between several locations. The mail-order company wants to ensure that the trucking company has sufficient resources to handle the orders. In this case, large amounts of typical data on the shipping schedules might be presented to the contractor, which allows them to determine whether or not the load can be handled. The contractor, though, is an untrusted entity for purposes of verifying their competence to handle the job, and any claim they make must be verified (with hope, efficiently) by the mail order company.

The probabilistically checkable proof (PCP) techniques (c.f., [BFL91],[BFLS91], [ALM$^+$98]) offer ways of verifying such solutions quickly. In these protocols a proof is written down which a verifier can trust by inspecting only a constant number of bits of the proof. The PCP model offers efficient mechanisms for verifying any computation performed in NEXP with an efficient verifier. We note that the verifiers in the PCP results all require $\Omega(n)$ time. Approximate PCPs were introduced in [EKR99] for the case when the input data is very large, and even linear time is prohibitive for the verifier. Fast approximate PCPs allow a verifier to ensure that the solution to the optimization problem is at least *almost* correct. Approximate PCPs running in logarithmic or even constant time have been presented in [EKR99] for several combinatorial problems. For

example, a proof can be written in such a way as to convince a constant time verifier that there exists a bin packing which packs a given set of objects into a small number of bins. Other examples include proofs which show the existence of a large flow, a large matching, or a large cut in a graph to a verifier that runs in sublinear time.

Our Results. We consider approximate PCPs for multidimensional bin packing. In particular, we show how a verifier can be quickly convinced that a set of multidimensional objects can be packed into a small number of bins. Our results generalize the one-dimensional bin-packing results of [EKR99]. The approximate PCP protocols for the bin-packing problem are more intricate in higher dimensions; for example, the placements and orientations of the blocks within the bin must be considered more carefully. In the one-dimensional case, the approximate PCP protocol of [EKR99] makes use of a property called *heaviness* of an element in a list, introduced by [EKK+98]. Essentially, *heaviness* is defined so that testing if an element is heavy can be done very efficiently (logarithmic) in the size of the list and such that all heavy elements in the list are in monotone increasing order. We generalize this notion to the multidimensional case and give heaviness tests which determine the heaviness of a point $x \in [1, \ldots, n]^d$ in time $O(\log^d n)$. Then, given a heaviness tester which runs in time $T(n)$, we show how to construct an approximate PCP for bin packing in which the running time of the verifier is $O(T(n))$.

Given function f from $[1, \ldots, n]^d$ to $[1, \ldots, r]$, a multidimensional monotonicity tester passes functions f that are monotone and fails functions f if no way of changing the value of f at no more than ϵ fraction of the inputs will turn f into a monotone function. In [GGLR98], a monotonicity tester with query complexity $\tilde{O}(d^2n^2r)$ is given. Our multidimensional heaviness tester can also be used to construct a multidimensional monotonicity tester which runs in time $O(T(n))$. However, Dodis et al. [DGL+99] independently give monotonicity testers that are as efficient as ours for two dimensions

and greatly improve on our running times for dimension greater than two.

Outline of Chapter We give definitions of our model and of heaviness tests in Section 4.1. Then in Section 4.2 we give a sublinear algorithm for verifying a solution to the multidimensional bin-packing problem. In Section 4.3 we give two tests for heaviness, which may be of independent interest as these tests also yield sublinear tests for monotonicity properties.

4.1 Preliminaries

Notation. We use the notation $x \in_R S$ to indicate x is chosen uniformly at random from the set S. The notation $[n]$ indicates the interval $[1, \ldots, n]$.

We define a partial ordering relation $\prec$ over integer lattices such that if x and y are d-tuples then $x \prec y$ if and only if $x_i \leq y_i$ for all $i \in \{1, \ldots, d\}$. Consider a function $f : \mathcal{D}^d \to \mathcal{R}$, where $\mathcal{D}^d$ is a d-dimensional lattice. For $x, y \in \mathcal{D}^d$ such that $x \prec y$, we say that x and y are in *monotone order* if $f(x) \leq f(y)$. We say f is *monotone* if for all $x, y \in \mathcal{D}^d$ such that $x \prec y$, x and y are in monotone order.

Approximate PCP. The approximate PCP model is introduced in [EKR99]. The verifier has access to a theorem and a possibly-valid proof Π. It can query Π in order to determine whether the theorem is close to a true theorem. More specifically, if on input x, the prover claims $f(x) = y$, then the verifier wants to know if y is close to $f(x)$.

Definition 61 (Ergun et al. [EKR99]) *Let $\Delta(\cdot, \cdot)$ be a distance function. A function f is said to have a* $t(\epsilon, n)$-approximate probabilistically checkable proof system *with distance function Δ if there is a randomized verifier V with oracle access to the words of a proof Π such that for all inputs ϵ, and x of size n, the following holds. Let y be the contents of the output tape, then*

1. *If* $\Delta(y, f(x)) = 0$*, there is a proof,* Π*, such that* V^{Π} *outputs pass with probability at least 3/4 (over the internal coin tosses of* V*);*

2. *If* $\Delta(y, f(x)) > \epsilon$*, for all proofs* Π'*,* $V^{\Pi'}$ *outputs fail with probability at least 3/4 (over the internal coin tosses of* V*); and*

3. V *runs in* $O(t(\epsilon, n))$ *time.*

The probabilistically checkable proof protocol can be repeated $O(\lg 1/\delta)$ times to get confidence at least $1 - \delta$. We occasionally describe the verifier's protocol as an interaction with a prover. In this interpretation, it is assumed that the prover's answers are fixed functions of the verifier's queries that are determined before the interaction. It is known that this model is equivalent to the PCP model [FRS88]. The verifier is a RAM machine which can read a word in one step.

We refer to PCP using the distance function $\Delta(y, f(x)) = \max\{0, 1 - (f(x)/y)\}$as an *approximate lower bound* PCP : if $f(x) \geq y$ then Π causes V^{Π} to pass; if $f(x) < (1 - \epsilon)y$ then no proof Π' convinces $V^{\Pi'}$ with high probability. This distance function applied to our bin-packing protocol will show that if a prover claims to be able to pack all of the n input objects, the verifier can trust that at least $(1 - \epsilon)n$ of the objects can be packed.

It also follows from considerations in [EKR99] that the protocols we give can be employed to prove the existence of suboptimal solutions. In particular, if the prover knows a solution of value v, it can prove the existence of a solution of value at least $(1 - \epsilon)v$. Since v is not necessarily the optimal solution, these protocols can be used to trust the computation of approximation algorithms to the NP-complete problems we treat. This is a useful observation since the prover may not have computational powers outside of deterministic polynomial time, but might employ very good heuristics. In addition, since the prover is much more powerful than V it may use its computational abilities to get surprisingly good, yet not necessarily optimal, solutions.

Heaviness Testing. Our methods rely on the ability to define an appropriate *heaviness* property on the domain elements of a function f. In the multidimensional case, heaviness is defined so that testing if a domain element is heavy can be done very efficiently in the size of the domain, and such that all heavy elements in the domain which are comparable according to $\prec$ are in monotone order.

We give a simple motivating example of a heaviness test for $d = 1$ from [EKK+98]. This one-dimensional problem can be viewed as the problem of testing if a list $L = (f(1), f(2), \ldots, f(n))$ is mostly sorted. Here we assume that the list contains distinct elements (a similar test covers the non-distinct case). Consider the following for testing whether such a list L is mostly sorted: pick a point $x \in L$ uniformly and at random. Perform a binary search on L for the value x. If the search finds x then we call x *heavy*. It is simple to see that if two points x and y are heavy according to this definition, then they are in correct sorted order (since they are each comparable to their least common ancestor in the search tree). The definition of a heaviness property is generalized in this paper. We call a property a *heaviness property* if it implies that points with that property are in monotone order.

Definition 62 *Given a domain $\mathcal{D} = [1, \ldots, n]^d$, a function $f : \mathcal{D} \to \mathcal{R}$ and a property H over $\mathcal{D}$, we say that H is a* heaviness property *if*

1. $\forall x \prec y,\ H(x) \wedge H(y)$ *implies* $f(x) \leq f(y)$, *and*
2. *In a monotone function all points have property H.*

If a point has a heaviness property H then we say that point is *heavy*. There may be many properties which can be tested of points of a domain which are valid heaviness properties. A challenge of designing heaviness tests is to find properties which can be tested efficiently. A heaviness test is a probabilistic procedure which decides the heaviness property with high probability. If a point is not heavy, it should fail this test with high probability, and if a function is perfectly monotone, then every point should

pass. Yet it is possible that a function is not monotone, but a tested point is actually heavy. In this case the test may either pass or fail.

Definition 63 *Let $\mathcal{D} = [1, \ldots, n]^d$ be a domain, and let $f : \mathcal{D} \to \mathcal{R}$ be a function. Let $S(\cdot, \cdot)$ be a randomized decision procedure on $\mathcal{D}$. Given security parameter δ, we will say S is a* heaviness test *for x if*

1. *If for all $z \prec y$, $f(z) \leq f(y)$, $S(x, \delta) = \text{Pass}$.*
2. *If x is not heavy then $\Pr(S(x, \delta) = \text{Fail}) > 1 - \delta$.*

The heaviness tests we consider enforce, among other properties, local multidimensional monotonicity of certain functions computed by the prover. It turns out that multidimensional heaviness testing is more involved than the one dimensional version considered in earlier works, and raises a number of interesting questions.

Our results on testing bin-packing solutions are valid for any heaviness property, and require only a constant number of applications of a heaviness test. We give sample heaviness properties and their corresponding tests in Section 4, yet it is an open question whether heaviness properties with more efficient heaviness tests exist. Such tests would immediately improve the efficiency of our approximate PCP verifier for bin packing.

Permutation Enforcement. Suppose the values of a function f are given for inputs in $[n]$ in the form of a list $y_1, \ldots, y_n$. Suppose further that the prover would like to convince the verifier that the y_i's are distinct, or at least that there are $(1 - \epsilon)n$ distinct y_i's. In [EKR99], the following method is suggested: The prover writes array A of length n. $A(j)$ should contain i when $f(i) = j$ (its preimage according to f). We say that i is *honest* if $A(f(i)) = i$. Note that the number of honest elements in $[n]$ lower bounds the number of distinct elements in $y_1, \ldots, y_n$ (even if A is written incorrectly). Thus, sampling $O(1/\epsilon)$ elements and determining that all are honest suffices to convince the

verifier that there are at least $(1-\epsilon)n$ distinct y_i's in $O(1/\epsilon)$ time. We refer to A as the *permutation enforcer*.

4.2 Multidimensional Bin Packing

We consider the d-dimensional bin-packing problem. We assume the objects to be packed are d-dimensional rectangular prisms, which we will hereafter refer to as blocks. The blocks are given as d-tuples (in $\mathbb{N}^d$) of their dimensions. Similarly, the bin size is given as a d-tuple, with entries corresponding to the integer width of the bin in each dimension. When we say a block with dimensions $w = (w_1, \ldots, w_d) \in \mathbb{N}^d$ is located at position $x = (x_1, \ldots, x_d)$, we mean that all the locations y such that $x \prec y \prec x + w - \vec{1}$ are occupied by this block. The problem of multidimensional bin packing is to try to find a packing of n blocks which uses the least number of bins of given dimension $D = (N_1, \ldots, N_d)$.

It turns out to be convenient to cast our problem as a maximization problem. We define the d-dimensional bin-packing problem as follows: given n blocks, the dimensions of a bin, and an integer k, find a packing that packs the largest fraction of the blocks into k bins. It follows that if $1-\epsilon$ fraction of the blocks can be packed in k bins, then at most $k + \epsilon n$ bins are sufficient to pack all of the blocks, by placing each of the remaining blocks in separate bins.

We give an approximate lower bound PCP protocol for the maximization version of the d-dimensional bin-packing problem in which the verifier runs in $O((1/\epsilon)T(N,d))$ time where $T(N,d)$ is the running time for a heaviness tester on $\mathcal{D} = [N_1] \times \cdots \times [N_d]$ and we take $N = \max_i N_i$. In all of these protocols, we assume that the block and bin dimensions fit in a word.

In this protocol, we assume that the prover is trying to convince the verifier that all the blocks can be packed in k bins. We require that the prover provides an encoding

of a feasible packing of the input blocks in a previously agreed format. This format is such that if all the input blocks can be packed in the bins used by the prover, the verifier accepts. If only less than $1-\epsilon$ fraction of the input blocks can be simultaneously packed, the verifier rejects the proof with some constant probability. In the intermediate case, the verifier provides no guarantees.

4.2.1 A First Representation of a Packing

We represent a bin as a d-dimensional grid with the corresponding length in each dimension. The prover will label the packed blocks with unique integers and then label the grid elements with the label of the block occupying it in the packing. In Figure 4.1, we illustrate one such encoding. The key to this encoding is that we give requirements by which the prover can define a monotone function on the grid using these labels only if he knows a feasible packing. To show such a reduction exists, we first define a relation on blocks.

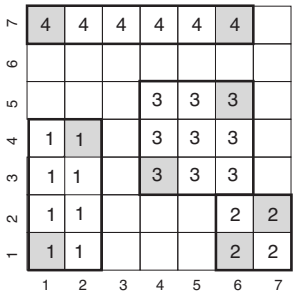

Figure 4.1: A 2D Encoding

Definition 64 *For a block b,* the highest corner of b*, denoted $h^{(b)}$, is the corner with the largest coordinates in the bin it is packed with respect to the $\prec$ relation. Similarly,* the lowest corner of b*, denoted $l^{(b)}$, is the corner with the smallest coordinates.*

In our figure, $l^{(1)} = (1, 1)$ and $h^{(1)} = (2, 4)$. We can order blocks by only considering the relative placement of these two corners.

Definition 65 *Let b_1 and b_2 be two blocks packed in the same bin. Block b_1* precedes *block b_2 in a packing if $l^{(b_1)} \prec h^{(b_2)}$.*

Note that for a pair of blocks in dimension higher than one it may be the case that neither of the two blocks precedes the other. This fact along with the following observation makes this definition interesting.

Observation 66 *For two blocks, b_1 and b_2, such that b_1 precedes b_2, b_1 and b_2 overlap if and only if b_2 precedes b_1.*

Surely if b_1 precedes b_2 and this pair overlaps it must be the case that $l^{(b_2)} \prec h^{(b_1)}$. It follows that the precedence relation on blocks is a reflexive antisymmetric ordering precisely when the packing of blocks is feasible. Given such an ordering, it is easy to construct a monotone function.

Lemma 67 *Given a feasible packing of a bin with blocks, we can label the blocks with distinct integers such that when we assign each occupied grid element in the d-dimensional grid (of the bin) with the label of the block occupying it, we get a monotone partial function, which can be extended to monotone total function, on this grid.*

PROOF: The relation from Definition 65 gives a relation on the blocks that is reflexive and antisymmetric. Therefore we can label the blocks according to this relation such that a block gets a label larger than those of all its predecessors. This labeling gives us a monotone partial function on the grid. To extend this partial function to a total function, each unoccupied grid element can be assigned the smallest possible.

Now we can describe the proof that the prover will write down. The proof will consist of three parts: the first one is a table which will have an entry for each block containing the label assigned to the block; a pointer to the bin where the object was assigned and the locations of the two (lowest and highest) corners of the block in this bin. The second part is a permutation enforcer on the blocks and the labels of the

blocks. Finally, the third part consists of a d-dimensional grid with dimensions of size $N_1 \times \ldots \times N_d$ for each bin used that numbers each grid element with the label of the block occupying it.

4.2.2 Testing Multidimensional Bin-Packing Using Heaviness

The heaviness test we have defined can be used to test that the prover's labeling agrees with a monotone function. We will show that if all the defining corners of a pair of blocks are heavy then they cannot overlap.

Protocol. We will define "good" blocks such that all good blocks can be packed together feasibly. Our notion of good should have the properties that (1) a good block is actually packed inside a bin, and it is not overlapping any other good block; and (2) we can efficiently test a block for being good. Then, the verifier will use sampling to ensure that at least $1 - \epsilon$ fraction of the blocks are good in the protocol.

Definition 68 *The block i with dimensions $\vec{w} = (w_1, \ldots, w_d)$ is* good *with respect to an encoding of a packing if it has the following properties:*

- *Two corners defining the block in the proof have positive coordinates with values inside the bin.*
- *The distance between these corners exactly fits the dimensions of the block, i.e., $\vec{w} = h^{(i)} - l^{(i)} + \vec{1}$.*
- *The grid elements at $l^{(i)}$ and $h^{(i)}$ are heavy.*
- *The block is assigned a unique label among the good blocks, i.e., it is honest with respect to the permutation enforcer.*

Given this definition, we can prove that two good blocks cannot overlap.

Lemma 69 *If two blocks overlap in a packing, then both of the blocks cannot be good with respect to this packing.*

PROOF: Note that when two blocks overlap, according to Definition 65, they must both precede each other, that is, $l^{(b_1)} \prec h^{(b_2)}$ and $l^{(b_2)} \prec h^{(b_1)}$. We know, by the definition of a heaviness property, that two comparable heavy points on the grid do not violate monotonicity. Since both defining corners of a good block must have the same label, either $l^{(b_1)}$ and $h^{(b_2)}$, or $l^{(b_2)}$ and $h^{(b_1)}$ violates monotonicity.

Corollary 70 *There is a feasible packing of all the good blocks in an encoding using k bins.*

The verifier's protocol can be given as follows: The verifier chooses a block randomly from the input, and using the encoding described above, confirms that the block is good. Testing a block for being good involves $O(d)$ comparisons for the first two conditions in the definition, O(1) time for checking the unique labeling using the permutation enforcer, and 2 heaviness tests for the third condition. The verifier repeats this $O(1/\epsilon)$ times to ensure at least $(1-\epsilon)$ fraction of the blocks are good. We have proven:

Theorem 71 *There is an $O((1/\epsilon)T(N,d))$-approximate lower bound PCP for the d-dimensional bin packing problem where $T(N,d)$ is the running time for a heaviness tester on $\mathcal{D} = [N_1] \times \cdots \times [N_d]$ and $N = \max_i\{N_i\}$.*

4.2.3 A Compressed Representation of a Packing

The previous protocol requires the prover to write down a proof whose size depends on the *dimensions* of the bins to be filled, since the values N_i were based on the actual size of the bins given. We show here how the prover may write a proof which depends only on the number, of objects to be packed. In the protocol from the previous section the

verifier calls the heaviness tester only on grid elements which correspond to the lowest or the highest corners of the blocks. We use this observation to get a compressed proof.

The prover constructs a set of *distinguished coordinate* values S_k for each dimension $k = 1, \ldots, d$. Each set is initially empty. The prover considers each block i and does the following: for the lower corner, $l^{(i)} = (c_1, \ldots, c_d)$, and higher corner, $h^{(i)} = (e_1, \ldots, e_d)$, of block i, the prover updates $S_i \leftarrow S_i \cup \{c_i\} \cup \{e_i\}$. After all the blocks are processed, $|S_i| \leq 2n$. The *compressed grid* will be a sublattice of $\mathcal{D}$ with each dimension restricted to these distinguished coordinates, that is the set $\{\langle x_1, \ldots, x_d \rangle | x_i \in S_i\}$. This grid will contain in particular all the corners of all the blocks and the size of this proof will be at most $O((2n)^d)$. The fact that this new compressed encoding is still easily testable does not trivially follow from the previous section. In particular, we must additionally verify that the prover's compression is valid.

The proof consists of four parts. First the prover implicitly defines the proof from the previous section, which we refer to as the *original grid*. The prover then writes down a table containing the *compressed grid*. In each axis, the prover labels the coordinates $[1, \ldots, 2n]$ and provides a *lookup-table* (of length $2n$) for each axis which maps compressed grid coordinates to original grid coordinates. Finally the prover writes down the list of objects with pointers to the compressed grid, and a permutation enforcer as before. In Figure 4.2 , we give the compressed encoding of the packing from Figure 4.1.

6	4	4	4	4	
5			3	3	
4	1	1	3	3	
3	1	1			
2	1	1		2	2
1	1	1		2	2
	1	2	3	4	5

Figure 4.2: A Compressed Encoding

Protocol. By making the prover write only a portion of the proof from the first protocol, we provide more opportunities for the prover to cheat. For example, even if the prover uses the correct set of hyperplanes for the compression, he may reorder them in the compressed grid to hide overlapping blocks. The conversion tables we introduced to our proof will allow the verifier to detect such cheating.

The definition of a good block is extended to incorporate the lookup tables. In a valid proof, the lookup tables would each define a monotone function on $[2n]$. We will check that the entries in the lookup tables which are used in locating a particular block are *heavy* in their respective lookup tables. Additionally we test a that a block is good with respect to Definition 68 in the compressed grid.[1] A block which passes both phases is a *good* block.

Our new protocol is then exactly as before. The verifier selects $O(1/\epsilon)$ blocks and tests that each is good and if so concludes that at least $1 - \epsilon$ fraction of the blocks are good.

Correctness. Any two good objects do not overlap in the compressed grid, by applying Lemma 69. Furthermore, since the labels of good objects in the lookup table are heavy, it follows that two good objects do not overlap in the original grid either. Certainly, since the corresponding values in the lookup table form a monotone sequence, the prover could not have re-ordered the columns during compression to untangle an overlap of blocks. It also follows from the earlier protocol that good blocks are the right size and are uniquely presented. We have shown our main theorem:

Theorem 72 *There is an $O((1/\epsilon)T(n,d))$-approximate lower bound PCP for the d-dimensional bin-packing problem with proof size $O((2n)^d)$, where $T(n,d)$ is the running time for a heaviness tester on $\mathcal{D} = [2n]^d$.*

[1] Except when we test the size of the block, for which we refer to the original coordinates via the lookup table.

4.2.4 An extension to recursive bin packing

At the simplest level the recursive bin-packing problem takes as input a set of objects, a list of container sizes (of unlimited quantity), and a set of bins. Instead of placing the objects directly in the bins, an object must first be fit into a container (along with other objects) and the containers then packed in the bin. The goal is to minimize the total number of bins required for the packing. We can give a protocol by which a prover can convince a verifier that a good solution exists by applying an extension of our multidimensional bin-packing tester. In particular, we define an object as *good* if it passes the goodness test (with respect to its container) given in Section 4.2 and furthermore if the container it is in passes the same goodness test (with respect to the bin). After $O(1/\epsilon)$ tests we can conclude that most objects are good and hence that $(1-\epsilon)$ fraction of the objects can be feasibly packed. For a k-level instance of recursive bin packing, therefore, the prover will write k compressed proofs and $O(k/\epsilon)$ goodness tests will be needed.

4.2.5 Can Monotonicity Testing Help?

Given the apparent similarities between heaviness testing and monotonicity testing, it may seem that a monotonicity test could be used to easily implement our multidimensional bin packing protocol. The obvious approach, though, does not seem to work. The complications arise because we are embedding n objects in a $(2n)^d$ sized domain. If a monotonicity tester can determine that the domain of our compressed proof has $(1-\epsilon')$ of its points in a monotone subset, we can only conclude that at least $n - \epsilon' \cdot (2n)^d$ boxes are "good", by distributing the bad points among the corners of the remaining boxes. Thus a direct application of monotonicity testing on this domain seems to need an error parameter of $O(\epsilon/(n^d))$. If the running time of the monotonicity tester is linear in ϵ then this approach requires at least $O((2n)^{d-1})$ time.

In this section we describe two heaviness tests for testing functions over a domain isomorphic to an integer lattice. We represent our domains as $\mathcal{D} = [1, \ldots, n]^d$. The range of the function can be any partial order. Here we describe the range as $\mathbb{R}$, the set of real numbers. Both tests which follow can determine that a point is heavy in $O((2 \log n)^d)$ time. These running times yield efficient bin packing tests for small values of d. In particular, the example applications of bin packing which we have cited typically have dimension less than 3.

4.3 Tests for heaviness properties

In this section we give two tests for heaviness properties which are used in our bin-packing test. As discussed in section 4.1 these tests can also be used to give general monotonicity testers for functions over a d dimensional lattice.

The first test We extend the protocol of [EKK$^+$98] to multidimensional arrays. On input x our test compares x to several random elements y selected from a set of carefully chosen neighborhoods around x. It is tested that x is in order with a large fraction of points in each of these neighborhoods. From this we can conclude that any two comparable heavy points a and b can be ordered by a mutually comparable point c such that $a < c < b$ and $f(a) < f(c) < f(b)$. The test is shown in Figure 4.3

We consider a set of $\log^d n$ carefully chosen neighborhoods around a point x. We say that x is *heavy* if for a large fraction of points y in each of these neighborhoods, $f(x)$ and $f(y)$ are monotonically ordered. We are able to show from this that for any two heavy points x and y, two of these regions can be found whose intersection contains a point z with the property that $x < z < y$ and $f(x) < f(z) < f(y)$. Hence this defines a valid heaviness property.

```
HeavyTest(D, f, x, δ)
 for k_1 ← 0 ... log x_1,
      ⋮
     k_d ← 0 ... log x_d  do
        repeat  t = O(2^d log(1/δ))  times
           choose  h_i ∈_R [1, 2^{k_i}]  1 ≤ i ≤ d
           h ← (h_1, ..., h_d)
           if  (f(x) < f(x − h))  return  FAIL
 for  k_d ← 0 ... log(n − x_1),
      ⋮
     k_d ← 0 ... log(n − x_d)  do
        repeat  t  times
           choose  h_i ∈_R [1, 2^{k_i}]  1 ≤ i ≤ n
           h ← (h_1, ..., h_d)
           if  (f(x) > f(x + h))  return  FAIL
return  PASS
```

Figure 4.3: Algorithm HeavyTest

Theorem 73 *Algorithm* `Heavy-Test` *is a (randomized) heaviness tester with query complexity* $O(\log(1/\delta)2^d \log^d n)$ *and error probability* δ.

In order to prove this theorem, we consider a special graph induced by a function f over a partially ordered domain. The vertices in the graph correspond to points of the domain while edges are inserted between points which are monotonically ordered ordered according to f.

Definition 74 *The graph* G_f *induced by a function* $f : D^n \to R$ *is a directed graph where* $V(G_f) = D^n$ *and* $E(G_f) = \{(x, y) | x \prec y$ *and* $f(x) \leq f(y)\}$.

Given this graph G_f, a point x, and a deviation h we are interested in the number of points in the intervals $[x, x+h]$ and $[x-h, x]$ which are monotonically ordered according to f. In terms of G_f we want to know how many of the out-edges originating at x terminate in the subgraph of points $[x, x+h]$ and how many of the in-edges to x originate in the subgraph of points $[x - h, x]$. We define these points as functions of x and h.

Definition 75 *$\Gamma^+_{h_1,\ldots,h_n}(x)$ is the set of points y in the domain such that $x \prec y$, $y \prec x+h$, and $(x,y) \in E(G_f)$. Similarly, $\Gamma^-_{h_1,\ldots,h_n}(x)$ is the set of points y in the domain such that $y \prec x$, $x \prec y+h$, and $(y,x) \in E(G_f)$.*

Using these definitions, we can formalize the notion of a "good" point. Intuitively, these "good" points have lots of incoming and outgoing arcs from and to every neighborhood around them.

Definition 76 *A point x in the graph G_f is* η-good *if for all i, all k_i, $0 \leq k_i \leq \log x_i$, $|\Gamma^-_{2^{k_1},\ldots,2^{k_d}}(x)| \geq \eta 2^{\sum_i k_i}$, and for all k_i, $0 \leq k_i \leq \log(n - x_i)$, $|\Gamma^+_{2^{k_1},\ldots,2^{k_d}}(x)| \geq \eta 2^{\sum_i k_i}$.*

Note that our definition of "good" requires that x satisfy requirements over $O(\log^d n)$ subsets of $\mathcal{D}$. The following lemma states that a pair of "good" points does not violate monotonicity.

Lemma 77 *If $x \prec y$ and x and y are each η-good, where $\eta = 1 - 2^{-(d+1)}$, then $(x,y) \in E(G_f)$.*

PROOF: Fix $x \prec y$. Let x and y be opposite endpoints of a d-dimensional rectangular hyperprism. Let I denote the space of points in $\mathcal{D}$ within this closed hyperprism. Let Δ_i denote the lengths of each of the axes of this hyperprism. Define m_i so that $\Delta_i \leq m_i < 2\Delta_i$ and $m_i = 2^{k_i}$ for some integer k_i. Now extend I to a new hypercube S such that the lengths of the axes of S are given by the set of m_i's defined above, that is, intuitively "round up" each side length to the next power of two.

By definition of η-*goodness* and of S, we can now calculate the number of edges from x into I. We can bound $|\Gamma^+_{m_1,\ldots,m_d}(x) \cap I|$ such that

$$|\Gamma^+_{m_1,\ldots,m_d}(x) \cap I| \geq \eta|S| - |S \setminus I| = \eta|S| - |S| + |I| = |I| - (1-\eta)|S|$$

and similarly

$$|\Gamma^-_{m_1,\ldots,m_d}(y) \cap I| \geq |I| - (1-\eta)|S|$$

If we can show that $|\Gamma^{+}_{m_1,\ldots,m_d}(x) \cap I| + |\Gamma^{-}_{m_1,\ldots,m_d}(y) \cap I| > |I|$ then the pigeonhole principle can be applied to find some vertex z with $(x,z) \in E(G_f)$ and $(z,y) \in E(G_f)$. By transitivity, we would have shown that $(x,y) \in E(G_f)$. We solve the equation given above

$$2\left(|I| - (1-\eta)|S|\right) > |I|$$

if and only if

$$|I| > 2(1-\eta)|S| = 2(2^{-(d+1)})|S| = 2^{-d}|S|$$

This last line is true since $|I|$ and $|S|$ are d-dimensional and every side of S is less than twice the length of its corresponding side in I. The correctness of Algorithm Heavy Test follows immediately.

PROOF: [of Theorem 73] Given a function f over $[n]^d$ and a point x, this algorithm constructs neighborhoods around x and explicitly checks (by sampling $\log(1/\delta)$ times) that x is η-good, with $\eta = 1 - 2^{-(d+1)}$.

The Second Heaviness Test This algorithm is based on a recursive definition of heaviness. Namely a point x is heavy in dimension d if a certain set of projections of x onto hyperplanes are each heavy in dimension $d-1$. We are able to use the heaviness of these projection points to conclude that d-dimensional heavy points are appropriately ordered.

Given a dimension *d* hypercube, C , consider a subdividing operation ϕ which maps C into 2^d congruent subcubes. This operation passes through the center *d* hyperplanes parallel to each of the axes of the hypercube. This is a basic function in our algorithm. For notational convenience, we extend ϕ to Φ which acts on sets of cubes such that $\Phi(\{x_1,\ldots,x_n\}) = \{\phi(x_1),\ldots,\phi(x_n)\}$. It is now possible to compose Φ with itself. We also define a function $\hat{\Phi}(x,C) = S \Rightarrow x \in S \in \Phi(C)$. This function is also a notational convenience which identifies the subcube a point lies in after such a division.

Now consider any two distinct points in the hypercube, x and y. We wish to apply Φ to the cube repeatedly until x and y are no longer in the same cube. To quantify this we define a new function $\varrho : C^2 \rightarrow \mathcal{Z}$ such that $\varrho(x,y) = r \Rightarrow \hat{\Phi}^r(x,C) = \hat{\Phi}^r(y,C)$ and $\hat{\Phi}^{r+1}(x,C) \neq \hat{\Phi}^{r+1}(y,C)$. That is, the $r+1$st composition of Φ on C separates x from y.

Definition 78 *A point x is heavy in a domain $\mathcal{D} = [n]^d$ if the $2d$ perpendicular projections of x onto each cube in the series $\hat{\Phi}(x,\mathcal{D}), \ldots, \hat{\Phi}^{\log n}(x,\mathcal{D})$ of shrinking cubes are all heavy in dimension $d-1$. The domains $\mathcal{D}'$ for these recursive tests are the respective faces of the cubes. When $d = 1$ we use the test of [EKK+98].*

We can now give the heaviness test for a point. Let C be a d-dimensional integer hypercube with side length n. Let x be some point in C. Construct the sequence $\{s_1, \ldots, s_k\} = \{\hat{\Phi}^1(x,C), \ldots, \hat{\Phi}^k(x,C)\}$ where $k = \lceil \log(n) \rceil$. Note that $s_k = x$. At each cube s_k perform the following test: (1) Compute the $2d$ perpendicular projections $\{p_1, \ldots, p_{2d}\}$ of x onto the $2d$ faces of s_k. (2) Verify that f is consistent with a monotone function on each of the $2d$ pairs (x, p_k). (3) If $d > 1$ recursively test that each of the points p_i is heavy over the reduced domain of its corresponding face on s_k. If $d = 1$, we use the heaviness test of [EKK+98]. This test is shown in Figure 4.4.

Theorem 79 *If x and y are heavy and $x < y$ then $f(x) < f(y)$.*

PROOF: *(by induction on d).* Let $r = \varrho(x,y)$. Let $S = \hat{\Phi}^r(x,C)$. Let $s_x = \hat{\Phi}^{r+1}(x,C)$ and $s_y = \hat{\Phi}^{r+1}(y,C)$. There is at least one plane perpendicular to a coordinate axes passing through the center of S which separates x and y. This plane also defines a face of s_x and of s_y, which we denote as f_x and f_y respectively. By induction we know the projections of x and y onto these faces are heavy. Since y dominates x in every coordinate, we know that $p_x < p_y$. Inductively we can conclude from the heaviness of the projection points that $f(p_x) < f(p_y)$. Since we have previously tested that $f(x) < f(p_x)$ and $f(p_y) < f(y)$ we conclude $f(x) < f(y)$.

```
RecursiveHeavyTest(D, f, x, δ)
 let d = dimension of D
 if d = 1
    δ' ← δ/(d log^d n)
    return EKKRV-HeavyTest(f, D, δ')
 else
    for i = 1 ... log n
       Φ = Φ̂^i(x, C)
       {p_1, ..., p_d} = projections of x onto Φ
       for k = 1 ... d
          C ← the face of Φ̂^i(x, D) containing p_k
          RecursiveHeavyTest(Φ, f, p_k, δ)
       end
    end
 end
 return PASS
```

Figure 4.4: Algorithm RecursiveHeavyTest

Running time analysis

If we let $H_d(n)$ be the number of queries made by our algorithm in testing that a point of the function $f : \mathcal{Z}_n^d \to \mathcal{S}$ is heavy, then we can show

Lemma 80 *For all $d > 1$, for sufficiently large n, $H_d(n) \leq (d-1)\log^d(n)\log(1/\delta)$.*

PROOF: We use proof by induction. For the case $d = 1$ we employ the spot checker algorithm from [EKK+98], which performs $\log(1/\delta)\log(n)$ queries to determine that a point is heavy. We shall define $H_1(n) = \log(1/\delta)\log(n)$. For our base case we compute $d = 2$

$$\begin{aligned}
H_2(n) &= \log(1/\delta) \cdot 2(H_1(n/2) + H_1(n/4) + \ldots + H_1(2) + H_1(1)) \\
&= \log(1/\delta) \cdot 2(\log(n) + \log(n/2) + \log(n/4) + \ldots + \log(2) + \log(1)) \\
&= 2\log(1/\delta) \sum_{i=1}^{\log(n)} i \\
&= 2\log(1/\delta) \frac{\log(n)(\log(n)+1))}{2} \\
&= \log(1/\delta)\log^2(n) + O(\log(n))
\end{aligned}$$

We now assume $H_d(n)$ is of the correct form, and prove $H_{d+1}(n)$ is as claimed.

$$\begin{aligned}
H_{d+1}(n) &= \log(1/\delta) \cdot 2(H_d(n/2) + H_d(n/4) + \ldots + H_d(2) + H_d(1)) \\
&= d \sum_{k=1}^{n} (d-1) \log^d(2^k) \log(1/\delta) \\
&= d(d-1) \log(1/\delta) \sum_{i=1}^{\log n} i^d \\
&= \frac{d(d-1)}{d+1} \log(1/\delta) \log^{d+1}(n) + O(\log^d(n)) \\
&\leq d \log^{d+1}(n) \log(1/\delta)
\end{aligned}$$

Theorem 81 *Algorithm* `RecursiveTest` *given above is a heaviness tester that requires* $O((d\log(d) + d\log\log(n) + log(1/\delta))(d-1)\log^d(n))$ *queries.*

PROOF: The confidence parameter $\delta' = \delta/(d\log^d(n))$ which appears in Figure 4.4 arises because the probability of error accumulates at each recursive call. Now apply Lemma 80.

APPENDIX A

TOOLS

There are several standard results we will use frequently. Those are given here for reference. The first two are tail inequalities for random variables. Given an arbitrary random variable X with expectation μ and variance σ^2, Chebyshev's inequality bounds the likelihood of X taking on values more than ρ away from μ.

Theorem 82 *Let X be any random variable with mean μ and variance σ^2. Then*

$$\Pr\left[|X-\mu| > \rho\right] \leq \frac{\sigma^2}{\rho^2}.$$

Chernoff Bounds give much tighter results if the random variable are known to be sums of independent Bernoulli trials. These bounds exist in many forms, but those most useful to us in this paper are

Theorem 83 *For $M > 0$ if $X_1, \ldots, X_M$ are independent $0/1$ random variables where $\Pr\left[X_i = 1\right] = p_i$ and $0 < p_i < 1$. Let $p = \sum_i \frac{p_i}{M}$. Then for $0 < \alpha \leq 1$ we have*

$$\begin{aligned}
\Pr\left[\frac{\sum_{i=1}^{M} X_i}{M} > (1+\alpha)p\right] &< e^{-\frac{1}{3}\alpha^2 pM} \\
\Pr\left[\frac{\sum_{i=1}^{M} X_i}{M} < (1-\alpha)p\right] &< e^{-\frac{1}{2}\alpha^2 pM}
\end{aligned}$$

We also use bounds on vector norms.

Lemma 84 *If x is an n-element vector then*

$$\begin{aligned}
\|\vec{x}\|_2 &\leq \|\vec{x}\|_1 \leq \sqrt{n}\|\vec{x}\|_2 \\
\|\vec{x}\|_2^2 &\leq \|\vec{x}\|_\infty \|\vec{x}\|_1
\end{aligned}$$

The following theorem states that all sufficiently large entries of a probability vector can be estimated efficiently.

Theorem 85 *Given a black-box distribution X over R, a threshold t and an accuracy $\epsilon > 0$, there is an algorithm,* Naive Algorithm*, that requires $O(t^{-1}\epsilon^{-2}\log|R|\log(1/\delta))$ samples and outputs an estimate $\tilde{X}$ such that with probability at least $1-\delta$, for every $i \in R$ with $X(i) \geq t$ we have $(1-\epsilon)X(i) \leq \tilde{X}(i) \leq (1+\epsilon)X(i)$; the algorithm also outputs a set $R' \subseteq R$ that includes $\{i \in R \mid X(i) \geq t\}$ and on which the above approximation is guaranteed.*

The proof (omitted) of the above theorem is a simple application of a Chernoff bound to several independent samples from X.

BIBLIOGRAPHY

[ACM91] *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, New Orleans, Louisiana, 6–8 May 1991.

[ACM97] *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, El Paso, Texas, 4–6 May 1997.

[ADK15] Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal testing for properties of distributions. In *Advances in Neural Information Processing Systems*, pages 3591–3599, 2015.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[Alo86] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *JCSS*, 58, 1999.

[BCFM00] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *JCSS*, 60, 2000.

[BDKR05] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM Journal on Computing*, 35(1):132–150, 2005.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In ACM [ACM91], pages 21–31.

[BFRV11] Arnab Bhattacharyya, Eldar Fischer, Ronitt Rubinfeld, and Paul Valiant. Testing monotonicity of distributions over general partial orders. In *ICS*, pages 239–252, 2011.

[BKR04] Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 381–390. ACM, 2004.

[Can15] Clément L Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22(63):1–1, 2015.

[CDGR18] Clément L Canonne, Ilias Diakonikolas, Themis Gouleakis, and Ronitt Rubinfeld. Testing shape restrictions of discrete distributions. *Theory of Computing Systems*, 62(1):4–62, 2018.

[CDVV14] Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1203. SIAM, 2014.

[CM89] N. Cressie and P.B. Morgan. Design considerations for Neyman Pearson and Wald hypothesis testing. *Metrika*, 36(6):317–325, 1989.

[CS10] Artur Czumaj and Christian Sohler. Testing expansion in bounded-degree graphs. *Combinatorics, Probability and Computing*, 19(5-6):693–709, 2010.

[Csi67] I. Csiszár. Information-type measures of difference of probability distributions and indirect observations. *Studia Scientiarum Mathematicarum Hungarica*, 1967.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, 1991.

[DDS+13] Constantinos Daskalakis, Ilias Diakonikolas, Rocco A Servedio, Gregory Valiant, and Paul Valiant. Testing k-modal distributions: Optimal algorithms via reductions. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1833–1852. Society for Industrial and Applied Mathematics, 2013.

[DGL+99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In Dorit Hochbaum, Klaus Jensen, José D.P. Rolim, and Alistair Sinclair, editors, *Randomization, Approximation, and Combinatorial Optimization*, volume 1671 of *Lecture Notes in Computer Science*, pages 96–108, Berkeley, California, 8–11 August 1999. Springer-Verlag.

[DGPP17] Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Sample-optimal identity testing with high probability. *arXiv preprint arXiv:1708.02728*, 2017.

[DK16] Ilias Diakonikolas and Daniel M Kane. A new approach for testing properties of discrete distributions. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 685–694. IEEE, 2016.

[DKN15] Ilias Diakonikolas, Daniel M Kane, and Vladimir Nikishkin. Testing identity of structured distributions. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1841–1854. Society for Industrial and Applied Mathematics, 2015.

[EKK+98] Funda Ergün, Sampath Kannan, S. Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 259–268, 1998.

[EKR99] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate PCPs. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, May 1999.

[FK99] Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *COMBINAT: Combinatorica*, 19, 1999.

[FKSV99] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L^1-difference algorithm for massive data streams (extended abstract). In *FOCS 40*, 1999.

[FRS88] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multiprover interactive protocols. In *Proceedings, Structure in Complexity Theory, Third Annual Conference*, pages 156–161, Georgetown University, Washington D.C., 14–17 June 1988. IEEE Computer Society Press.

[GGLR98] O. Goldreich, S. Goldwasser, E. Lehman, and D. Ron. Testing monotonicity. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS-98)*, pages 426–435, Los Alamitos, CA, November8–11 1998. IEEE Computer Society.

[GGR96] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. In *37th Annual Symposium on Foundations of Computer Science*, pages 339–348, Burlington, Vermont, 14–16 October 1996. IEEE.

[GM99] Phillip B. Gibbons and Yossi Matias. Synopsis data structures for massive data sets. In *SODA 10*, pages 909–910. ACM-SIAM, 1999.

[Gol16] Oded Goldreich. The uniform distribution is complete with respect to testing identity to a fixed distribution. *Electronic Colloquium on Computational Complexity (ECCC)*, 23(15):1, 2016.

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[GR97] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In ACM [ACM97], pages 406–415.

[GR00] Oded Goldeich and Dana Ron. On testing expansion in bounded-degree graphs. Technical Report TR00-020, Electronic Colloquium on Computational Complexity, El Paso, Texas, 4–6 May 2000.

[GV96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[Kan94] R. Kannan. Markov chains and polynomial time algorithms. In Shafi Goldwasser, editor, *Proceedings: 35th Annual Symposium on Foundations of Computer Science, November 20–22, 1994, Santa Fe, New Mexico*, pages 656–671, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1994. IEEE Computer Society Press.

[KR94] Michael Kearns and Ronitt Rubinfeld. On the learnability of discrete distributions. In *In The 25th Annual ACM Symposium on Theory of Computing*, pages 273–282. ACM Press, 1994.

[KS08] Satyen Kale and C Seshadhri. Testing expansion in bounded degree graphs. *35th ICALP*, pages 527–538, 2008.

[KY91] Sampath Kannan and Andrew Chi-Chih Yao. Program checkers for probability generation. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *ICALP 18*, volume 510 of *Lecture Notes in Computer Science*, pages 163–173, Madrid, Spain, 8–12 July 1991. Springer-Verlag.

[Leh86] E. L. Lehmann. *Testing Statistical Hypotheses*. Wadsworth and Brooks/Cole, Pacific Grove, CA, second edition, 1986. [Formerly New York: Wiley].

[LRR13] R. Levi, D. Ron, and R. Rubinfeld. Testing properties of collections of distributions. *Theory of Computing*, 9(8):295–347, 2013.

[NP33] J. Neyman and E.S. Pearson. On the problem of the most efficient test of statistical hypotheses. *Philos. Trans. Royal Soc. A*, 231:289–337, 1933.

[NS10] Asaf Nachmias and Asaf Shapira. Testing the expansion of a graph. *Information and Computation*, 208(4):309–314, 2010.

[Ona09] Krzysztof Onak. Testing distribution identity efficiently. *arXiv preprint arXiv:0910.3243*, 2009.

[Pan08] Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

[Par98] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*, volume 20 of *Classics in applied mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[PR99] Michal Parnas and Dana Ron. Testing the diameter of graphs. In Dorit Hochbaum, Klaus Jensen, José D.P. Rolim, and Alistair Sinclair, editors, *Randomization, Approximation, and Combinatorial Optimization*, volume 1671 of *Lecture Notes in Computer Science*, pages 85–96, Berkeley, California, 8–11 August 1999. Springer-Verlag.

[Ron] Dana Ron. Property testing. http://www.eng.tau.ac.il/ danar/Public-pdf/tut.pdf.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, April 1996.

[Rub12] Ronitt Rubinfeld. Taming big probability distributions. *ACM Crossroads*, 19(1):24–28, 2012.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, July 1989.

[SV97] Amit Sahai and Salil Vadhan. A complete promise problem for statistical zero-knowledge. In *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science*, pages 448–457. IEEE, 20–22 October 1997.

[SV99] Amit Sahai and Salil Vadhan. Manipulating statistical difference. In Panos Pardalos, Sanguthevar Rajasekaran, and José Rolim, editors, *Randomization Methods in Algorithm Design (DIMACS Workshop, December 1997)*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 251–270. American Mathematical Society, 1999.

[VV11] Gregory Valiant and Paul Valiant. Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new clts. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 685–694. ACM, 2011.

[VV17] Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 46(1):429–455, 2017.

[Wal77] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM trans. math. software*, 3:253–256, 1977.

[Yam95] Kenji Yamanishi. Probably almost discriminative learning. *Machine Learning*, 18(1):23–50, 1995.