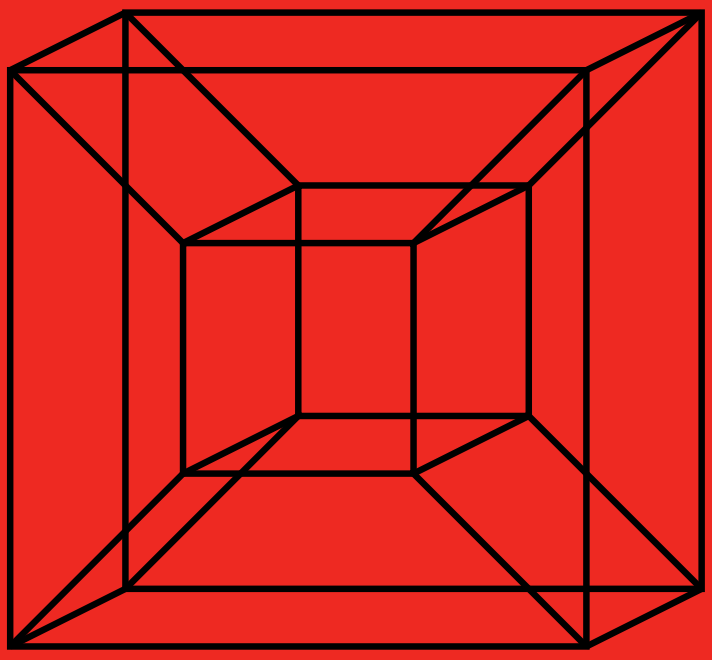
Query-Efficient Computation in Property Testing and Learning Theory



David García Soriano

Query-Efficient Computation in Property Testing and Learning Theory

ILLC Dissertation Series DS-2012-05

Institute for Logic, Language and Computation

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 904
1098 XH Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The investigations were performed at the Centrum Wiskunde & Informatica (CWI) and were supported by Vici grant 639.023.302 from the Netherlands Organization for Scientific Research (NWO).

Printed and bound by Ipskamp Drukkers.

ISBN: 978-94-6191-233-6

Query-Efficient Computation in Property Testing and Learning Theory

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Agnietenkapel
op woensdag 25 april 2012, te 10:00 uur

door

David García Soriano

geboren te Madrid, Spanje.

Promotor:	Prof. dr. H.M. Buhrman
Overige leden:	Prof. dr. E. Fischer
	Dr. A. Matsliah
	Prof. dr. A. Schrijver
	Prof. dr. R.M. de Wolf

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

a mis padres y hermano

Contents

Acknowledgments

First of all, I would like to thank my advisor, Harry Buhrman. He offered me a position in the Algorithms and Complexity group at CWI and provided me with a most pleasurable research environment for the last four years. I am also grateful for his guidance and good humor, and for allowing me complete freedom to pursue my own research interests (even though they didn't turn out to be quite as "quantum" as envisioned).

It goes without saying that this thesis is $\Omega(1)$-far from being the work of a single individual (or even a small junta). None of this would have been possible without the people I collaborated with: Noga Alon, Eric Blais, Jop Briët, Harry Buhrman, Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, Arie Matsliah, and Ronald de Wolf. I am especially indebted to Arie and Sourav, who sparked my interest in property testing and have since been an endless source of challenging problems and clever ideas. During the great time we had together they shared with me their contagious enthusiasm for research, as well as for the game of pocket billiards. Their advice on all kinds of matters, whether academic or personal, has been invaluable to me.

I am also thankful to the people who agreed to sit on my committee and proofread this dissertation, and who most definitely improved its quality through their comments and suggestions: Harry Buhrman, Eldar Fischer, Arie Matsliah, Lex Schrijver, and Ronald de Wolf. Ronald in particular has always been very kind in sharing his insight and expertise; this thesis has no doubt benefited greatly from his strive for rigor and his uncanny attention to detail. I thank Eldar also for inviting me for two research visits to the Israel Institute of Technology in Haifa, which proved very enjoyable and fruitful.

The pleasant atmosphere I found at the workplace I owe to my office mates Bruno Loff, Jop Briët, and Florian Speelman, with which I have engaged in many exciting discussions and debates, scientific as well as recreational. Florian deserves special mention for translating the abstract into the "samenvatting" in record time. I also very much enjoyed the time spent with my colleagues Peter van der Gulik, Giannicola Scarpa, and Christian Schaffner.

Finally, I wish to express my most sincere gratitude to my family, girlfriend and friends, for their continuous support and encouragement throughout the years.

David García Soriano
Amsterdam, March 2012.

Chapter 1

Query-efficient computation

1.1 Introduction

In this thesis we address the question of how to solve computational problems when we can only afford to look at a minuscule fraction of the data. The subject is easily motivated by the ever-increasing need to handle large datasets. For example, suppose we are interested in checking whether or not a list of numbers is sorted in increasing order (such a sequence is said to be monotonically increasing, or *monotone* for short). As far as algorithmic tasks go, this is usually considered one of the easiest because it can be solved in essentially the same time it takes to scan the input. But if the sequence is extremely long, just reading all the numbers will take precious time, and it is reasonable to wonder if there might be a faster way. To make any savings in running time, we would have to be able to ascertain if the sequence is monotone without inspecting the whole of it.

At first glance, this is not really possible: if we so much as miss a single element, we will be forced to wrongly accept some non-monotone sequences (or reject some monotone ones). This is because we will fail to see anything wrong with a sequence that is sorted *except* for the one element that we didn't look at. In view of this, we could ask if there is anything to be gained by allowing *randomized* algorithms that have a small probability of error, instead of returning the answer with absolute certainty. Unfortunately not much can be done, even in this case: it can be shown that for us to succeed on *every* sequence at least two thirds of the time, we need to look at half the sequence no matter what. If we didn't, an adversary could choose an arbitrary monotone sequence, change one randomly located element so that the new sequence is no longer monotone, and give it for us to test. Note that if we inspect less than half of the elements of such an "adversarial" sequence, it will look perfectly good to us more than half of the time, regardless of our strategy. When this happens we cannot hope to distinguish the sequence from a truly monotone one; since we need to accept monotone sequences with probability $\frac{2}{3}$, our best shot to decrease the probability

of being fooled in these situations is to flip a biased coin and accept the sequences that "look" monotone only with probability $\frac{2}{3}$. But then the error probability for the adversarial sequences would be greater than $\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$, which is more than we want.

What this tells us is that, unless we are willing to look at the entire data, we cannot hope to accept valid monotone sequences *and still reject those that are very "close" to being monotone*. What if we are promised that the sequence is either monotone or "far away" from it? Call a sequence "0.1-far from monotone" if at least 10% of the sequence needs to be modified in order to make it increasing; otherwise say that it is "0.1-close to monotone". Let $s = s_1 s_2 \dots s_n$ denote the sequence, and let us refer to each question for the element s_i at some particular position i as a *query*. An amazingly simple algorithm of Ergün et al. solves this problem with just about $\log n$ queries [EKK+00]: First, pick a random index $i \in \{1, 2, \dots, n\}$. Then query the ith element of s, which gives us the value of s_i. Next perform a binary search for s_i in the sequence, as though s were actually sorted. Finally, accept if the element found was s_i.

The algorithm just described is very efficient because the binary search takes time logarithmic in the sequence length. It clearly accepts if s is indeed monotone, because this is the only precondition for binary search to work. When s is 0.1-far from monotone, the test rejects with probability at least 0.1. The easiest way to see the latter claim is to rephrase it in an equivalent form: any sequence that the algorithm above accepts with probability greater than 0.9 must be 0.1-close to increasing. This is because the set of indices for which the binary search succeeds can be shown to be increasing, which means that we can make s monotone by modifying only the remaining indices. To see why, let i, j $(i < j)$ be two such indices and consider the point at which the binary search for s_i diverges from the search for s_j. Since both searches are successful, s_i must be less than or equal to the pivot element, and s_j must be greater than or equal to it; therefore $s_i \leq s_j$. Note that, while a rejection probability of 0.1 may look low, it can be easily amplified to take values arbitrarily close to 1 by performing a constant number of independent repetitions of this basic test.

Many other questions of this type are possible. In the example above we were given "black box" (or "oracle") access to the sequence and we wanted to minimize the number of elements inspected (queries). We can also consider properties of graphs; for example, we may want to check if a given graph is "essentially the same" as a known one, where "essentially" means that the two are considered equivalent if they are the same up to relabelling the vertices. Here each query could be a question about a particular entry of the adjacency matrix of the graph.

Usually, however, we will be exploring the difficulty of checking properties of functions. Our example problem of testing monotonicity of a sequence can be thought of as testing the monotonicity of a function from $[n]$ to $[n]$. Similarly, questions about graphs can be phrased in terms of functions on $[n] \times [n]$ representing adjacency matrices.

In computer science, *boolean*[1] functions play a prominent role (these are functions where the range of definition is the pair $\{0,1\}$). We will be interested in boolean functions mostly (but not exclusively). In general, a *property* of functions is simply a collection $\mathcal{P}$ of functions, and f is said to have property $\mathcal{P}$ iff $f \in \mathcal{P}$. The simplest example of a boolean property is that of being a constant boolean function on $\{0,1\}^n$; for f to satisfy this property we must have either $f \equiv 0$ or $f \equiv 1$, so $\mathcal{P}$ has only two elements in this case. The input to the tester would be the truth table of the function (i.e., its image on all elements of its domain), and each query would be a question for the value that f assumes on an element x chosen by the tester. A more interesting property could be that of being determined by just k out of n input variables; such functions are termed *k-juntas*.

1.2 Sublinear-time algorithms

In the classical decision setting, the goal of an algorithm is to determine whether the input has property $\mathcal{P}$ or not. Just how easy or hard it is to decide which case it is depends on $\mathcal{P}$ itself; this topic is studied by the theories of computability and complexity (a modern account of many of the highlights can be found in the book by Arora and Barak [AB09]). The complexity parameters under study are usually the running time of the algorithm, its space usage, the amount of randomness or advice it requires, or the possible tradeoffs among these measures; this reflects the intent to study the inherent computational hardness of the problem. Remarkably, the first research along these lines was carried out by Turing before the advent of digital computers, in his classic paper introducing the notion of computability [Tur36]. But with the tremendous increases in processors' speed and computational power, the idea of what may be calculated efficiently has undergone revisions and refinements in what might seem the "wrong" direction. Originally, the sole existence of an algorithm to solve a problem (i.e., its being computable) used to be considered good enough. Subsequent considerations of how running time scales with input size led Cobham [Cob65] and Edmonds [Edm65] to develop the more restrictive notion of polynomial-time computability. In recent times, the sheer amount of data available for many applications, such as computational biology, data mining, etc., has become so large that often only linear-time algorithms (or almost linear-time) can be considered efficient (although this is a much more model-dependent notion than that of polynomial-time computability).

Linear-time algorithms compute the answer in essentially the same time it takes to read the input. In this sense they are optimal because a correct algorithm for an ordinary problem is typically obliged to examine its entire input. As we

[1] It has been said that, while it is an honor for a mathematician to have an object named after him, the highest honor is conferred when the word ceases to be capitalized (as in, e.g., "abelian group"). Hence we will always write "boolean" in lowercase.

saw before, even if we allow randomization, a sizable portion of the input needs to be inspected. For these reasons, linear-time algorithms have long been thought to meet the "golden standard" of efficiency.

In spite of this, nowadays a good deal of research effort is being devoted to obtaining *sublinear-time* algorithms. These are programs that make assertions as to whether or not a certain property is satisfied by the input after reading only a very small portion thereof (see the recent survey of Ron and Shapira [RS12]). This entails weakening our requirements. In *Property Testing*, we no longer expect an algorithm to reject at the slightest discrepancy with the property $\mathcal{P}$. Instead, we ask it to distinguish between objects that have the property and those which are "far away" from it. In other words, we would like to accept inputs that satisfy $\mathcal{P}$, while ensuring that we reject "corrupt" inputs that cannot be made to satisfy $\mathcal{P}$ even after significant changes. It does not matter what the tester outputs for those functions in the gray area between "$\mathcal{P}$" and "far from $\mathcal{P}$". The main measure of efficiency for property testers is their *query complexity*, which indicates how much of the input an algorithm needs to inspect in order to reach a reliable decision. Surprisingly, many natural problems admit testers of *constant* query complexity, which are sometimes referred to as *local* testers.

Property testing has its origins in the classic paper of Blum, Luby and Rubinfeld [BLR90], who discovered the so-called "BLR test" for the class of linear functions (or, what amounts to the same thing, a local test for the Hadamard code). They were motivated by questions on program verification and self-testing/correcting. Shortly thereafter, Rubinfeld and Sudan [RS96] studied testers for the class of low-degree polynomials over a field (see also [AS03]). These results were instrumental in the early algebraic proofs of the celebrated PCP Theorem of Arora, Lund, Motwani, Sudan and Szegedy [ALM+98], and in the proof that $\mathsf{MIP} = \mathsf{NEXP}$ by Babai, Fortnow and Lund [BFL91]. The systematic study of property testing for discrete structures such as functions and graphs was initiated by Goldreich, Goldwasser, and Ron [GGR98]. Since then a great many properties have been found to admit efficient property testers; examples of well-studied problems include monotonicity [DGL+99, FLN+02, GGL+00], juntas [FKR+04, CG04, Bla09], halfspaces [MORS10, MORS09], and having concise representations [DLM+07]. The field has been extremely active over the last few years—see, e.g., the aforementioned survey [RS12] as well as those of Fischer [Fis01] and Ron [Ron08, Ron10].

Many uses of property testers have been found, as the subject arises naturally in a variety of contexts; for example, see [CSZ00] for applications in computational geometry. Property testers can be used to perform a quick, preliminary check that discards the input instances that are far from satisfying the property. Only when the input is close to satisfying it do we need to run a full-fledged, and potentially much more expensive, decision algorithm. The same ideas apply in the context of learning. Here one tries to construct a "learner" that, from sample values of the function, is able to come up with good predictors of the values of the function

on all possible inputs. The predictor itself will be a function that belongs to the so-called "hypothesis class" of the learner (see Chapter 7). If the function is far from the hypothesis class, then there is no such predictor and the learner is bound to be unsuccessful. A preliminary run of the property tester can detect this situation and spare us the unnecessary investment in resources.

Property testing of boolean functions will be our model of focus in most of this thesis. This being a work in theoretical computer science, it is sometimes deemed advisable to include a disclaimer that the algorithms presented here are mostly of theoretical interest, and in this sense this is no exception. The emphasis will be in understanding and not on the practicality or usefulness of these algorithms. However, just as it would be reckless to assume their practicality without fine-tuning them and/or studying their performance on real-world data, so it would be foolish to assume they are impractical only because they arose in a theoretical context. Some of them, such as those discussed in Chapter 6, definitely are in its current form; and some of them, such as those in Chapter 3 look like they are because of the large constants involved. However the latter are, more likely than not, an artifact of the proofs, and it seems entirely within the realm of possibility that many algorithms based on junta testing and core sample extraction can be efficiently implemented in practical applications.

1.3 Function isomorphism

In the next few chapters we will be studying the problem of boolean function isomorphism. We say that two boolean functions $f, g\colon \{0,1\}^n \to \{0,1\}$ are *isomorphic* if they are equal up to relabellings of the input variables, i.e., if it is possible to permute the n input variables of f so that the resulting function is equal to g. For instance, the functions

$$f(x_1x_2x_3) = (x_1 \wedge x_2) \vee x_3$$

and

$$g(x_1x_2x_3) = (x_3 \wedge x_1) \vee x_2$$

are isomorphic. On the other hand, a symmetric function, such as the majority function on n variables

$$\mathrm{Maj}_n(x_1x_2\ldots x_n) = \begin{cases} 1 & \text{iff } |x| \geq n/2 \\ 0 & \text{otherwise} \end{cases},$$

is only isomorphic to itself, because permuting variables has no effect on Maj_n. Function isomorphism is a well-studied problem since two functions being isomorphic means that they are "essentially the same" and have identical circuit realizations. (A related notion is that of *structural equivalence*, which allows for complementations of variables as well.) Moreover, many function properties can be cast in terms of isomorphism; some simple examples include the following:

- **Dictatorships**. The function g is a dictator if there is $i \in [n]$ such that $g(x_1x_2\ldots x_n) = x_i$ for all x. This amounts to saying that g is isomorphic to the fixed function $f(x_1x_2\ldots x_n) = x_1$.
- **Monomials**. The function g is a k-monomial if and only if it is isomorphic to $f(x_1x_2\ldots x_n) = x_1 \wedge x_2 \cdots \wedge x_k$.
- **Parities**. The function g is a k-parity if and only if it is isomorphic to $f(x_1x_2\ldots x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_k$.

The feature all these examples share is the fact that the function f is a small *junta* (that is to say, it depends on a small number of variables), and indeed the question of how the complexity is affected by the stipulation that f should be a small junta will play a major role in the sequel.

Although most of our focus will be on the query complexity of testing isomorphism, we start with a few words about the state of the art of the algorithms for deciding function isomorphism and their time complexity. When considering the decision version of the problem, we assume that truth tables for both functions are explicitly given. As explained in Chapter 4, there is a tight connection between function isomorphism and *hypergraph isomorphism*: broadly speaking, we can identify a boolean function f with the hypergraph with vertex set $[n]$ and edge set given by $f^{-1}(1)$ (where binary vectors $x \in \{0,1\}^n$ represent subsets of $[n]$ as usual). Seen this way, the problem of function isomorphism becomes a natural generalization of the analogous problem for graphs. The problem of graph isomorphism has no known polynomial-time solution, despite being strongly suspected *not* to be NP-complete [Sch88]; hence it is a natural candidate for an NP-intermediate problem. It should be noted that the input size for the hypergraph variant is $\Theta(2^n)$, whereas it is $\Theta\big(\binom{n}{2}\big) = \Theta(n^2)$ in the graph setting. The larger input size of the former can (and does) allow for faster solutions, relative to the input size. While the best known upper bounds for graph isomorphism are exponential (taking time $2^{O(\sqrt{n \log n})}$ on graphs with n nodes [BKL83]), a brute-force search over all permutations yields a solution for hypergraph isomorphism that runs in time $\widetilde{O}(n! \cdot 2^n) = 2^{O(n \log n)}$ for hypergraphs with n nodes, which is only quasipolynomial in the input size. The paper [Luk99] of Luks actually put the problem in P by providing a solution that runs in time $2^{O(n)}$. His algorithm proceeds by computing the *automorphism group* of a boolean function, a concept that will prove important to us later in Chapter 4. A recent paper of Babai and Codenotti [BC08a] considered the related problem of isomorphism of hypergraphs whose rank (maximum size of edges) is bounded by k, and gave a solution in time $\exp(\widetilde{O}(k^2\sqrt{n}\,))$.

It is also possible to consider variations where the input functions are implicitly encoded. For example, when f and g are given as boolean formulae, deciding isomorphism becomes coNP-hard. This problem is not known to be in coNP, although it is contained in Σ_2^{P}, the second level of the polynomial hierarchy. On

the other hand, Agrawal and Thierauf [AT00] showed that this problem is not complete for Σ_2^{P} under the assumption that the polynomial hierarchy does not collapse to the third level.

1.4 General notation

Let $n, k \in \mathbb{N}$ and $x \in \{0,1\}^n$. We write $[n] \triangleq \{1, \ldots, n\}$ and, when the symbol $[k, n]$ refers to a discrete set from context, we write $[k, n] \triangleq \{k, k+1, \ldots, n\}$. We refer to the elements of $\{0,1\}^n$ as n-bit binary strings or vectors, indistinctly. Whenever convenient, we identify a binary vector $x \in \{0,1\}^n$ with a subset of $[n]$ in the natural way, and vice versa. That is, $x \in \{0,1\}^n$ is identified with the set $\{i \in [n] \mid x_i = 1\}$, and $S \subseteq [n]$ is identified with its indicator string $x \in \{0,1\}^n$ satisfying $x_i = 1 \Leftrightarrow i \in S$. The *Hamming weight* of $x \in \{0,1\}^n$ is $|x| \triangleq |\{i \in [n] \mid x_i = 1\}|$.

Often we also identify $\{0,1\}^n$ with the direct product of n copies o $\mathbb{F}_2$ or $\mathbb{Z}_2$, and we write $x \oplus y$ (and sometimes $x + y$) for the sum of x and y over $\mathbb{F}_2^n$, which coincides with their bitwise XOR. In a similar manner, $x \wedge y$ denotes the bitwise AND of x and y, and $x \vee y$ their bitwise OR. Of course we may use the set-theoretic notation for these as well: $x \cap y$ and $x \cup y$.

Let S be a set and $k \in \mathbb{N}$. A k-set is a set of size k, and a k-subset of S is a k-set that is a subset of S. The collection of all k-subsets of S is denoted $\binom{S}{k}$, and $\binom{S}{\leq k}$ is the collection of all subsets of cardinality at most k; hence $\left|\binom{S}{k}\right| = \binom{|S|}{k}$. A similar notation is used for binomial coefficients:

$$\binom{m}{\leq k} \triangleq \sum_{i=0}^{k} \binom{m}{i}.$$

An *equipartition* of an s-set into t parts is a partition of the set into t parts of size $\lfloor s/t \rfloor$ or $\lceil s/t \rceil$; there must necessarily be $s \bmod t$ parts of the latter size.

The symbol log denotes logarithms to the base 2, and ln denotes the natural logarithm.

Restrictions and assignment manipulation

Let $a \in \{0,1\}^n, b \in \{0,1\}^m$. The symbol $a \sqcup b \in \{0,1\}^{n+m}$ denotes the concatenation of a and b.

Given $x \in \{0,1\}^n$ and a subset $I \subseteq [n]$, $x\restriction_I$ denotes the binary string obtained by restricting x to the indices in I, according to the natural order of $[n]$. Concretely, if $I = \{i_1, \ldots, i_t\}, i_1 \leq i_2 \leq \cdots \leq i_t$, then $x\restriction_I = x_{i_1} x_{i_2} \ldots x_{i_t}$. We also write $f\restriction_S$ for the restriction of a function to a set $S \subseteq \mathrm{dom}(f)$. For $y \in \{0,1\}^{|I|}$, $x_{I \leftarrow y}$ denotes the string z obtained by substituting y for the values in $x\restriction_I$, i.e., satisfying $z\restriction_I = y$ and $z\restriction_{[n]\setminus I} = x\restriction_{[n]\setminus I}$.

Parities

A *parity* is a linear form on $\mathbb{F}_2^n$, i.e., a function $f\colon \{0,1\}^n \to \{0,1\}$ that satisfies the identity $f(x \oplus y) = f(x) \oplus f(y)$. (To see why this condition corresponds to linearity, recall that the only non-zero scalar in $\mathbb{F}_2$ is 1, and that sum over $\mathbb{F}_2^n$ is the same as bitwise XOR.) Such a function is given by

$$f(x) = \langle x, v \rangle \bmod 2 = \bigoplus_{i \in [n]} x_i v_i$$

for some $v \in \{0,1\}^n$. The function f associated with v in this way is denoted v^*; we sometimes refer to $|v|$ as the *size* of the parity $f = v^*$. The set of all linear boolean functions on $\{0,1\}^n$ is denoted PAR^n.

We say that $f = v^*$ is a *k-parity* if its associated vector v has Hamming weight exactly k. The set of all k-parities on n variables is denoted PAR^n_k; the set of all parities of size at most k is denoted $\mathsf{PAR}^n_{\leq k}$. Sometimes we drop the superscript n.

Distributions

The term $x \sim \mathcal{D}$ represents a random variable x drawn from the distribution $\mathcal{D}$. Also, $e \in S$ under the probability symbol means that an element e is chosen uniformly at random from a set S. That is, it is understood that distributions are uniform by default, unless stated otherwise.

We use an algebraic notation to express convex combinations of distributions. For example, for two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, the distribution obtained by choosing a random element of $\mathcal{D}_1$ with probability half, and a random permutation of $\mathcal{D}_2$ with probability half is denoted $(\mathcal{D}_1 + \mathcal{D}_2)/2$.

We use the phrase "with high probability" to mean that some event happens with probability at least some constant $p > 1/2$.

The hypercube

The *directed n-dimensional hypercube* (or simply *n-cube*) is a directed graph $H_n = (V_n, E_n)$ with $V_n = \{0,1\}^n$ and $E_n = \{(x,y) \mid x \subseteq y \text{ and } |y| = |x| + 1\}$. The hth *layer*(or *level*) of H_n contains all $x \in V_n$ with $|x| = h$.

There is a natural partial order $\leq$ on the vertices of the hypercube: $x \leq y$ iff there is a directed path from x to y in H_n, i.e., when $(x_i = 1 \implies y_i = 1)$ for all $i \in n$. This ordering is the same as that of the inclusion poset of $[n]$: $x \leq y$ iff $x \subseteq y$.

Hypergraphs

Hypergraphs [Ber89, Bol86] are a straightforward generalization of graphs. Recall that the edge set of a graph is simply a collection of pairs of vertices. An *undirected*

hypergraph is a pair $H = (V, E)$, where V is a set of vertices and $E \subseteq \mathcal{P}(V)$ is an arbitrary collection of *hyperedges* (subsets of vertices). A hypergraph is thus essentially the same as a set system on V.

We define a *directed hypergraph* analogously, except that edges are now sequences (tuples) of elements of V rather than unordered sets. (Other definitions of directed hypergraphs are also used in the literature.)

The hypergraph H is *uniform* if all of its hyperedges have the same cardinality r; the number r is called the *rank* or *arity* of H.

Other

Expressions of the form $x = \Theta(y) \pm O(z)$ are taken to mean that there are constants $c > c' > 0, d \geq 0$ such that $c'y - dz \leq x \leq cy + dz$.

Tilde notation is used to hide polylogarithmic factors—for example $r(n) = \tilde{\Theta}(t(n))$ if there is a positive constant c such that $r(n) \geq \Omega\big(\frac{t(n)}{\log^c t(n)}\big)$ and $r(n) \leq O(t(n) \log^c t(n))$.

1.5 Property testing

We formalize here the concept of property testing outlined in Section 1.1, within the context of classical computation.[2]

1.5.1. Definition. Given a pair $f, g \colon D \to \{0,1\}$ of boolean functions defined on a domain D, the *distance* between them is

$$\operatorname{dist}(f, g) \triangleq \Pr_{x \in D}[f(x) \neq g(x)].$$

The distance from a function f to a property $\mathcal{P}$ is the minimum distance between f and g over all $g \in \mathcal{P}$, i.e.,

$$\operatorname{dist}(f, \mathcal{P}) \triangleq \min_{g \in \mathcal{P}} \operatorname{dist}(f, g).$$

For $\varepsilon \in \mathbb{R}^+$, f is *ε-far from $\mathcal{P}$* if $\operatorname{dist}(f, \mathcal{P}) \geq \varepsilon$, otherwise it is *$\varepsilon$-close to $\mathcal{P}$*.

1.5.2. Definition. A *(q, ε)-tester* for the property $\mathcal{P}$ is a randomized algorithm[3] $\mathcal{T}$ that queries an unknown function f on at most q different inputs in $\{0,1\}^n$ and then

[2] Property testing can also be studied within the context of quantum computation. This line of research was introduced by Buhrman, Fortnow, Newman and Röhrig [BFNR08] and continued in many other papers, but is beyond the scope of this thesis.

[3] No assumption of uniformity is implied by the use of the word "algorithm" here. In any case the distinction is inconsequential for us since all our lower bounds apply to non-uniform algorithms, and our upper bounds are usually uniform.

1. accepts f with probability at least $\frac{2}{3}$ when $f \in \mathcal{P}$ (**completeness**);
2. rejects f with probability at least $\frac{2}{3}$ when f is ε-far from $\mathcal{P}$ (**soundness**).

(If the property deals with a pair of input functions, the algorithm may query both.)

The query complexity of a (q, ε)-tester $\mathcal{T}$ is the worst-case number of queries it makes before making a decision, taken over all possible inputs and over the outcomes of all internal coin tosses of $\mathcal{T}$ (so it is at most q). The tester $\mathcal{T}$ is said to be *non-adaptive* if its choice of queries does not depend on the outcomes of earlier queries; otherwise it is *adaptive*.

The choice of 2/3 for the success probability is arbitrary, up to constant factors in the query complexity, by standard probability amplification arguments.

Note that a deterministic, non-adaptive tester is determined by a fixed set Q of $q = |Q|$ queries, together with a function from $\{0,1\}^q$ to $\{\mathbf{accept}, \mathbf{reject}\}$. A randomized tester can be specified with a distribution over deterministic testers, corresponding to the different outcomes of the internal coin tosses of the tester.

On the other hand, a deterministic, adaptive tester with q queries is specified by a decision tree of depth q. (See the paper [BW02] of Buhrman and de Wolf for a survey article on decision trees.) Each internal node is labelled by the next query and has two outgoing edges labelled 0 or 1, according to the outcome. Each leave is labelled by either **accept** or **reject**. Again, a randomized, adaptive tester can be viewed as a distribution over deterministic, adaptive testers.

A tester that always accepts functions in $\mathcal{P}$ has *one-sided error* (or *perfect completeness*); otherwise it has *two-sided error*.[4]

A (q, ε)-tester is said to be δ-*tolerant* if it also accepts when $\mathrm{dist}(f, \mathcal{P}) \leq \delta$ with high probability (where $\delta < \varepsilon$).

By default, in all testers (and bounds) discussed we assume adaptivity and two-sided error, unless mentioned otherwise. We assume without loss of generality that testers never query the same input twice.

The *query complexity* of a property $\mathcal{P}$ for a given $\varepsilon > 0$ is the minimum value of q for which there is a (q, ε)-tester for $\mathcal{P}$.

A property $\mathcal{P}$ is q-*testable* if, for some constant $\varepsilon > 0$, there is a (q, ε)-tester for $\mathcal{P}$. If $q = O(1)$, $\mathcal{P}$ is said to be *testable*.

1.5.1 Yao's principle and non-adaptive lower bounds

Most lower bound proofs for randomized algorithms involve Yao's principle [Yao77], directly or indirectly. Suppose we fix an input size n and construct a matrix

[4] The reader may wonder whether it is also possible to define one-sided testers with perfect soundness, which are allowed to err on the completeness side only. The reason these are not studied usually is that, for most natural properties, such testers would easily be seen to require $\Omega(2^n)$ queries.

whose rows correspond to all possible deterministic algorithms with a certain complexity c (say time complexity, or query complexity), and whose columns correspond to all possible input instances (functions in our case). Each entry in the matrix tells us whether a particular deterministic algorithm gives the correct answer on a particular input or not (although it could more generally contain some real value measuring in some way the performance of the algorithm). Then a randomized algorithm is the same as a distribution over the rows of the matrix, whereas a distribution over the columns corresponds to a distribution of input instances. Let $\mathcal{D}$ denote an arbitrary distribution on input instances. If there is a randomized algorithm $\mathcal{A}$ that always works with probability p, then in particular it works with probability at least p when fed with inputs from $\mathcal{D}$ (over the combined probability space of the algorithm's randomness and the input drawn from $\mathcal{D}$). But the success probability of $\mathcal{A}$ on inputs from $\mathcal{D}$ can be written as a convex combination of the success probabilities of deterministic algorithms (the rows of the matrix) on inputs from $\mathcal{D}$. Therefore simple averaging tells us that there must also be a *deterministic* algorithm $\mathcal{A}_{\mathcal{D}}$ (depending on $\mathcal{D}$) that works with probability at least p under $\mathcal{D}$. If we can prove that the latter is impossible, then we have shown that there is no good randomized algorithm with complexity c for the problem in question.

This proves the "easy direction" of Yao's principle, which is also the most widely used. The ubiquity of the principle in lower bound proofs can perhaps be explained by the fact that the reverse direction also holds: if there is no *randomized* algorithm with complexity c and success probability p on all inputs, then it is possible to concoct a distribution $\mathcal{D}$ such that no *deterministic* algorithm with complexity c can succeed with probability p when the inputs are drawn from $\mathcal{D}$. This can be proven via the minimax theorem of von Neumann [Neu28] for finite two-person, zero-sum games, which is also a consequence of the duality of linear programming. So from a theoretical standpoint every (non-uniform) lower bound can be shown using this method with a judicious choice of distribution $\mathcal{D}$.

In our context, this is easiest to apply to non-adaptive lower bounds, which are often proven by upper-bounding the statistical distance between two distributions $\mathcal{P}$ and $\mathcal{Q}$ of query responses: one where the input function has the property we are testing, and one where it is far from having the property.

1.5.3. Definition. The *statistical distance* (or *total variation distance*) between two probability measures $\mathcal{P}$ and $\mathcal{Q}$ is the largest possible difference between the probabilities that they can assign to the same event. If $\mathcal{P}$ and $\mathcal{Q}$ are discrete distributions on a countable set A, this is given by

$$\Delta(\mathcal{P}, \mathcal{Q}) \triangleq \max_{S \subseteq A} \left| \Pr_{P \sim \mathcal{P}} [P \in S] - \Pr_{Q \sim \mathcal{Q}} [Q \in S] \right|.$$

The L^1 distance between $\mathcal{P}$ and $\mathcal{Q}$ is

$$|\mathcal{P} - \mathcal{Q}|_1 \triangleq \sum_{x \in A} \left| \Pr_{P \sim \mathcal{P}}[P = a] - \Pr_{Q \sim \mathcal{Q}}[Q = a] \right|.$$

The L^∞ distance between $\mathcal{P}$ and $\mathcal{Q}$ is

$$|\mathcal{P} - \mathcal{Q}|_\infty \triangleq \max_{x \in A} \left| \Pr_{P \sim \mathcal{P}}[P = a] - \Pr_{Q \sim \mathcal{Q}}[Q = a] \right|.$$

The following well-known lemma relates these measures.

1.5.4. Lemma. *We have* $|\mathcal{P} - \mathcal{Q}|_1 \leq |A| \cdot |\mathcal{P} - \mathcal{Q}|_\infty$ *and* $\Delta(\mathcal{P}, \mathcal{Q}) = \frac{1}{2}|\mathcal{P} - \mathcal{Q}|_1$.

Proof. The first inequality is obvious. For the second, let

$$p_x = \Pr_{P \sim \mathcal{P}}[P = x]$$
$$P(S) = \Pr_{P \sim \mathcal{P}}[P \in S] = \sum_{x \in S} p_x$$

for $x \in A, S \subseteq A$, and define q_x and $Q(S)$ analogously. Observe that absolute values are not needed in the definition of $\Delta(\mathcal{P}, \mathcal{Q})$ because we can replace one event S with its complement $A \setminus S$. Let $S \subseteq A$ maximize $P(S) - Q(S)$. For all $x \in S$ we have $p_x \geq q_x$, otherwise $S \setminus \{x\}$ would increase this value; and for all $x \notin S$, we have $p_x \leq q_x$. Therefore the largest $P(S) - Q(S)$ is attained for $S = \{x \in A \mid p_x \geq q_x\}$. Also for any T we have $P(T) + P(A \setminus T) = Q(T) + Q(A \setminus T) = 1$, so plugging in our choice for S we obtain

$$\begin{aligned} P(S) - Q(S) &= Q(A \setminus S) - P(A \setminus S) \\ &= \frac{1}{2}(P(S) - Q(S) + Q(A \setminus S) - P(A \setminus S)) \\ &= \frac{1}{2} \sum_{x \in A} |p_x - q_x|, \end{aligned}$$

hence $\Delta(P, Q) = \frac{1}{2}|P - Q|_1$. □

Let $\mathcal{P}$ be a property of functions mapping T to $\{0, 1\}$. Let $\varepsilon > 0$ and

$$\mathcal{R} \subseteq \{f \colon T \to \{0, 1\} \mid \operatorname{dist}(f, \mathcal{P}) \geq \varepsilon\}$$

be non-empty. Any ε-tester for $\mathcal{P}$ should, with high probability, accept inputs from $\mathcal{P}$ and reject inputs from $\mathcal{R}$.

1.5.5. Corollary. *Let $\varepsilon, \mathcal{P}, \mathcal{R}$ be as in the preceding discussion, and let $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ be distributions over $\mathcal{P}$ and $\mathcal{R}$, respectively. If q is such that for all $Q \in \binom{T}{q}$,*

$$\sum_{a \in \{0,1\}^Q} \left| \Pr_{f \sim \mathcal{D}_{\text{yes}}} [f\restriction_Q = a] - \Pr_{f \sim \mathcal{D}_{\text{no}}} [f\restriction_Q = a] \right| < \alpha,$$

then any non-adaptive *tester for $\mathcal{P}$ with error probability $\leq 1/2 - \alpha/4$ must make more than q queries.*

With $\alpha = 2/3$ we obtain a bound for testers with success probability $2/3$.

Proof. The condition says that, for any deterministic tester, the statistical difference between the response vectors when f is drawn from $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ is less than $\alpha/2$. By definition of statistical distance, the probability of acceptance in both cases can only differ by less than $\alpha/2$, irrespective of the acceptance condition of the tester. But then the overall success probability of the tester when f is drawn from $(\mathcal{D}_{\text{yes}} + \mathcal{D}_{\text{no}})/2$ is less than $1/2 + \alpha/4$. To complete the proof, invoke Yao's lemma. □

1.5.2 A lemma for proving adaptive lower bounds

If we allow for adaptive algorithms, the condition of Corollary 1.5.5 no longer implies a lower bound of q queries. It does imply a lower bound of $\log(q+1)$ adaptive queries because for properties of boolean functions, the decision tree describing an algorithm is binary, so a depth-d decision tree can be replaced with a non-adaptive algorithm that queries in parallel all $2^d - 1$ query strings associated to its nodes, and then selects the ones that it actually needs.

However, the bounds obtained in this way are normally not very tight. We use the following lemma in various lower bound proofs for two-sided adaptive testing. It is proven implicitly in [FNS04], and a detailed proof appears in [Fis01]. Here we strengthen it somewhat, but the same proof still works (we reproduce it here for completeness).

1.5.6. Lemma. *Let $\mathcal{P}, \mathcal{R}$ be as in Corollary 1.5.5, and let $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ be distributions over $\mathcal{P}$ and $\mathcal{R}$, respectively. If q is such that for all $Q \in \binom{T}{q}$ and $a \in \{0,1\}^Q$ we have*

$$\alpha \Pr_{f \sim \mathcal{D}_{\text{yes}}} [f\restriction_Q = a] < \Pr_{f \sim \mathcal{D}_{\text{no}}} [f\restriction_Q = a] + \beta \cdot 2^{-q} \tag{1.1}$$

for some constants $0 \leq \beta \leq \alpha \leq 1$, then any tester for $\mathcal{P}$ with error probability $\leq (\alpha - \beta)/2$ must make more than q queries.

The hypothesis cannot be satisfied if $\alpha = 1$ and $\beta = 0$ because of the strict inequality in (1.1), so for any choice of α, β for which the lemma is applicable we have that $(\alpha - \beta)/2$ is bounded away from $1/2$, and this yields a lower bound of $\Omega(q)$ queries because the error probability of any tester can be reduced from $1/3$ to $(\alpha - \beta)/2$ by a constant number of repetitions (depending only on α, β).

Proof. Let $\mathcal{T}$ be any attempted tester that makes no more than q queries; without loss of generality it makes exactly q queries. Let $\mathcal{D} = (\mathcal{D}_{\text{yes}} + \mathcal{D}_{\text{no}})/2$ and fix a random seed such that the tester works correctly for $f \in \mathcal{D}$ with probability at least $1 - \frac{\alpha-\beta}{2}$; now the behaviour of the tester can be described by a deterministic decision tree of height q. Each leaf corresponds to a set $Q \in \binom{T}{q}$, along with an evaluation $a\colon Q \to \{0,1\}$; the leaf is reached if and only if f satisfies the evaluation. Consider the set L corresponding to accepting leaves; f is accepted if and only if there is $(Q, a) \in L$ such that $f\restriction_Q = a$. These $|L| \leq 2^q$ events are disjoint.

Let $p = \Pr_{f \sim \mathcal{D}_{\text{yes}}}[f \text{ is accepted}]$. By the observations above,

$$p = \sum_{(Q,a)\in L} \Pr_{f\sim\mathcal{D}_{\text{yes}}}[f\restriction_Q = a].$$

Likewise, let $r = \Pr_{f \sim \mathcal{D}_{\text{no}}}[f \text{ is accepted}]$ and write

$$r = \sum_{(Q,a)\in L} \Pr_{f\sim\mathcal{D}_{\text{no}}}[f\restriction_Q = a].$$

Conditioned on $f \in \mathcal{D}_{\text{yes}}$, the success probability of the tester is p. Conditioned on $f \in \mathcal{D}_{\text{no}}$, it is $1 - r$. Hence its overall success probability is $\frac{1+p-r}{2}$. By (1.1), we can perform a term-by-term comparison between the two sums in the expressions for p and r, which yields $\alpha p < r + \beta$, so $p - r < (1-\alpha)p + \beta \leq 1 - \alpha + \beta$. But then the overall success probability of $\mathcal{T}$ when f is taken from $\mathcal{D}$ is $\frac{1}{2} + \frac{p-r}{2} < 1 - \frac{\alpha-\beta}{2}$, so the error probability of $\mathcal{T}$ is larger than $(\alpha - \beta)/2$. □

In practice we sometimes make use of slightly different claims; their proof is still the same.

- The same conclusion holds if instead the inequality

  $$\alpha \Pr_{f\sim\mathcal{D}_{\text{no}}}[f\restriction_Q = a] < \Pr_{f\sim\mathcal{D}_{\text{yes}}}[f\restriction_Q = a] + \beta \cdot 2^{-q}$$

  is satisfied for all Q, a.

- If $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ are distributions of functions such that

  $$\Pr_{g\sim\mathcal{D}_{\text{yes}}}[f \in \mathcal{P}], \Pr_{g\sim\mathcal{D}_{\text{no}}}[g \in \mathcal{R}] = 1 - o(1),$$

the lemma is not quite applicable as stated. However, in that case the success probability of the tester can be no larger than

$$(1 + p - r + o(1))/2 < 1 - \frac{\alpha - \beta}{2} + o(1)$$

(where p and r are as in the proof of the lemma), so an $\Omega(q)$ lower bound still follows.

- Finally, note that the proof of the lemma is based on an indistinguishability result that a tester needs q queries to tell apart a random $f \sim \mathcal{P}$ from a random $f \sim \mathcal{R}$ (where $\mathcal{P}$ or $\mathcal{R}$ are chosen with probability half). If we drop the condition that $\mathcal{R}$ only contain functions far from $\mathcal{P}$, the implication for property testing lower bounds disappears, but the indistinguishability result still holds.

Chapter 2

Function isomorphism: first results

The content of this chapter is based on the papers

- S. Chakraborty, D. García–Soriano, and A. Matsliah. Nearly tight bounds for testing function isomorphism. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1683–1702, 2011.

- N. Alon, E. Blais, S. Chakraborty, D. García–Soriano and A. Matsliah. Nearly tight bounds for testing function isomorphism. Manuscript, 2011.

The second paper above is a combined version of the first and the article [AB10] of Alon and Blais, who independently obtained overlapping results. It is currently under journal review.

2.1 Property testing of isomorphism

Here we concern ourselves with property testing of boolean functions. Despite the progress in the study of the query complexity of many properties, our overall understanding of the testability of boolean function properties still lags behind our understanding of the testability of graph properties, whose study was initiated by Goldreich, Goldwasser, and Ron [GGR98].

A notable example that illustrates the gap between our understanding of graph and boolean function properties is isomorphism. There are three main variants to the isomorphism testing problem. (In the following list, an "object" refers to either a graph or a boolean function.)

1. **Testing isomorphism to a given object $\mathcal{O}$.** The query complexity required to test isomorphism in this variant depends on the object $\mathcal{O}$; the goal for this problem is to be able to characterize the query complexity of testing isomorphism to $\mathcal{O}$ in terms of some natural property of $\mathcal{O}$.

2. **Testing isomorphism to the hardest known object.** A less fine-grained variant of the first problem asks to determine the maximum query complexity of testing isomorphism to $\mathcal{O}$ over objects of a given size.

3. **Testing isomorphism of two unknown objects.** In this variant, the testing algorithm has query access to two unknown objects $\mathcal{O}_1$ and $\mathcal{O}_2$ and must distinguish between the cases where they are isomorphic to each other or far from isomorphic to each other.

Answering these questions, as suggested by [FKR$^+$04] and [BO10], is an important step in the research program of characterizing the testable properties of boolean functions, which constitutes the natural next step to take after the existing work on the testability of graph properties: In [AS08], Alon and Shapira proved that the family of "natural" graph properties testable with one-sided error is the class of *semi-hereditary* properties; and in [AFNS09], Alon, Fischer, Newman and Shapira showed that a graph property $\mathcal{P}$ can be tested with a constant number of queries iff testing $\mathcal{P}$ can be reduced to testing the property of satisfying one of finitely many Szemerédi partitions.

The problem of testing graph isomorphism was first raised by Alon, Fischer, Krivelevich, and Szegedy [AFKS00] (see also [Fis01]), who used a lower bound on testing isomorphism of two unknown graphs to give an example of a non-testable graph property of a certain type. Fischer [Fis05] studied the problem of testing isomorphism to a given graph G and characterized the class of graphs to which isomorphism can be tested with a constant number of queries. Tight asymptotic bounds on the (worst-case) query complexity of the problem of testing isomorphism to a known graph and testing isomorphism of two unknown graphs were then obtained by Fischer and Matsliah [FM08]. As a result, the graph isomorphism testing problem is well understood. To summarize,

- Graphs to which isomorphism can be tested with a constant number of queries are those which can be approximated by an "algebra" of constantly many cliques [Fis05]. This means that an approximation to the graph can be obtained from the cliques by applying set intersection, union and complementation operations (more on this in Chapter 4).

- The worst-case query complexity of testing isomorphism to a given graph on n nodes is $\widetilde{\Theta}(\sqrt{n}\,)$ [FM08].

Additionally, Babai and Chakraborty [BC08b] proved lower bounds for the query complexity of the problem of testing isomorphism between two uniform hypergraphs.

The picture is much less complete in the setting of boolean functions. The first problem on page 17 is particularly interesting because testing many function properties, like those mentioned in Section 1.3, are equivalent to testing isomorphism to some fixed function f. More general properties can often be reduced

to testing isomorphism to several functions (as a simple example, notice that testing whether g depends on a single variable can be done by first testing if g is isomorphic to $f(x) \equiv x_1$, then testing if g is isomorphic to $f(x) \equiv 1 - x_1$, and accepting if one of the tests accepts). The "Testing by Implicit Learning" approach of Diakonikolas et al. [DLM$^+$07] can also be viewed as a clever reduction from the task of testing a wide range of properties to simultaneous testing of function isomorphism against a number of functions. We elaborate on this technique and how our work relates to it in Chapter 6.

There are several classes of functions for which testing isomorphism is easy. For instance, if f is symmetric (invariant under permutations of variables), then f-isomorphism can be tested with a constant number of queries. (Since all permutations of a symmetric f are the same, the problem reduces to testing strict equivalence to a given function.) More interesting functions are also known to have testers with constant query complexity. Specifically, the fact that isomorphism to dictatorship functions and k-monomials can be tested with $O(1)$ queries follows from the work of Parnas, Ron and Samorodnitsky [PRS02].

The question of testing isomorphism against a known function f was first formulated explicitly by Fischer, Kindler, Ron, Safra, and Samorodnitsky [FKR$^+$04]. They gave a general upper bound on the problem, showing that for every function f that depends on k variables (that is, for every k-junta), the problem of testing isomorphism to f is solvable with $\mathrm{poly}(k/\varepsilon)$ queries. Conversely, they showed that when f is a parity function on $k < o(\sqrt{n})$ variables, testing isomorphism to f requires $\Omega(\log k)$ queries[1]. No other progress was made on the problem of testing isomorphism of boolean functions until recently, when Blais and O'Donnell [BO10] showed that for every function f that "strongly" depends on $k \leq n/2$ variables (meaning that f is far from all juntas on $k - O(1)$ variables), testing isomorphism to f requires $\Omega(\log k)$ non-adaptive queries, which implies a general lower bound of $\Omega(\log \log k)$ queries. They also proved that there is a k-junta (namely, a majority on k variables) against which testing isomorphism requires $\Omega(k^{1/12})$ non-adaptive queries, and therefore $\Omega(\log k)$ adaptive queries.

Taken together, the results in [FKR$^+$04, BO10] give only an incomplete solution to the problem of testing isomorphism to a given boolean function and provide only weak bounds on the other two versions of the isomorphism testing problem.

In this chapter and the next we settle questions 2 and 3 on page 18 up to logarithmic factors. The first question will be studied in subsequent chapters.

2.2 Notation

To fix some notational conventions, recall that $Sym(\Omega)$ denotes the symmetric group of all permutations of Ω, and Also, S_n denotes the group of permutations

[1]This was shown via an $\Omega(\sqrt{k})$ lower bound for non-adaptive testers obtained through the analysis of random walks in $\mathbb{Z}_2^q$.

on a set of size n such as $[n]$. For reasons that will become apparent shortly, the product operation we use for the elements of S_n is $\pi\sigma \triangleq \sigma \circ \pi$.

We consider the right action $\phi\colon S_n \to Sym(\{0,1\}^n)$ of S_n on $\{0,1\}^n$ defined in the following way: if $\pi \in S_n$, then $\phi(\pi) \in Sym(\{0,1\}^n)$ is the permutation mapping each $x = x_1x_2\dots x_n \in \{0,1\}^n$ to $\phi(\pi)(x) \triangleq x_{\pi(1)}x_{\pi(2)}\dots x_{\pi(n)}$. This is a faithful action, i.e., $|\operatorname{im}\phi| = |S_n| = n!$. We identify π and $\phi(\pi)$ and we write x^π (or $\pi(x)$) in place of $\phi(\pi)(x)$. Observe that $\phi(\pi)$ effectively sends the input at position i into position $\pi^{-1}(x)$, and as a result we have $(x^\sigma)^\pi = x^{\pi\circ\sigma} = x^{\sigma\pi}$ (note the order reversal). We also write f^π for the function on $\{0,1\}^n$ defined by $f^\pi(x) = f(x^\pi)$; by the observations above, $(f^\pi)^\sigma = f^{\pi\sigma}$. Similarly, for a set $Q \subseteq \{0,1\}^n$ we define $\pi(Q) \triangleq Q^\pi = \{x^\pi \mid x \in Q\}$.

In this language, the functions f and g are isomorphic (in short, $f \cong g$) if there is $\pi \in S_n$ with $f = g^\pi$. The set of all functions isomorphic to f is denoted

$$\operatorname{Isom}(f) \triangleq \{f^\pi \mid \pi \in S_n\}.$$

The *distance up to permutations of variables* between f and g is defined by

$$\operatorname{distiso}(f,g) \triangleq \min_{\pi\in S_n} \operatorname{dist}(f^\pi, g) = \operatorname{dist}(g, \operatorname{Isom}(f)).$$

Evidently, distiso is a metric.

Testing f-isomorphism is defined as the problem of testing the property $\operatorname{Isom}(f)$ in the usual property testing terminology. It is thus the task of distinguishing the case $f \cong g$ from the case $\operatorname{distiso}(f,g) \geq \varepsilon$.

2.3 Testing function isomorphism with one-sided error

As we shall see, the most straightforward tester for isomorphism against a boolean function is non-adaptive and has one-sided error. It is only natural to begin by studying testers that operate under such restrictions. Whereas the choice of adaptivity/non-adaptivity does not significantly affect the bounds (see Chapter 5), the fact that the one-sided error case is strictly harder than the two-sided error case was established in [FKR$^+$04]. In particular, they showed the impossibility of testing isomorphism to 2-juntas with one-sided error using a number of queries independent of n (their lower bound is $\Omega(\log\log n)$, which follows from an $\Omega(\log n)$ lower bound on non-adaptive testers). Here we show that the worst-case query complexity of testing isomorphism to a k-junta with one-sided error is $\Theta\big(\log\binom{n}{k}\big)$, up to $k = n^{1-\delta}$ (for any $\delta > 0$).

2.3.1. Theorem (Chakraborty et al. [CGM11c]).
For every integer $k \in [2,n]$ and every constant $0 < \varepsilon \leq \frac{1}{2}$, there is a k-junta

$f\colon \{0,1\}^n \to \{0,1\}$ for which ε-testing f-isomorphism with one-sided error requires $\Omega\big(\log \binom{n}{\leq k}\big)$ adaptive queries. On the other hand, for any k-junta there is a one-sided tester of isomorphism making $O\big(\frac{1}{\varepsilon} k \log n\big)$ non-adaptive queries.

Regarding the lower bound, note that for $k \geq n/2$ we have $\log \binom{n}{\leq k} = \Theta(n)$; and for $k < n/2$, $\log \binom{n}{\leq k} = \Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$. The range of k in the theorem is tight: when $k = 1$, as we mentioned in the introduction, testing isomorphism to any 1-junta with one-sided error can be done with $O(1/\varepsilon)$ many queries [PRS02].

The lower bound in Theorem 2.3.1 holds for the special case of k-parities (at least when $k \leq n/2$). Namely, we show that for any $2 \leq k \leq n-2$, the query complexity of testing with one-sided error whether a function is a k-parity (i.e, the XOR of *exactly* k indices of its input) is $\Theta\big(\log \binom{n}{k}\big)$. (This contrasts starkly with the situation for the problem of testing with one-sided error whether a function is a k-parity for *some* k, which can be solved with a constant number of queries by the classic test of Blum, Luby and Rubinfeld [BLR90].)

2.3.1 Upper bound

The upper bound of Theorem 2.3.1 can be seen as an instantiation of the "Occam razor" learning algorithm, which in this particular case maintains a collection of candidate permutations, and sufficiently many random samples are drawn to weed out the bad candidates so that only good ones remain eventually. It is well known that certain learning algorithms imply testing algorithms because proper learning is a strictly harder task than testing (see the survey by Ron [Ron08]). At any rate, the correctness of this procedure is easy to prove directly.

2.3.2. Proposition. *Isomorphism to any given $f\colon \{0,1\}^n \to \{0,1\}$ can be ε-tested with $O\Big(\frac{1+\log|\mathrm{Isom}(f)|}{\varepsilon}\Big)$ non-adaptive queries and one-sided error.*

Note that for any k-junta f, $|\mathrm{Isom}(f)| \leq \binom{n}{k} \cdot k! \leq n^k$. (For a k-parity f this bound can be strengthened to $|\mathrm{Isom}(f)| = |\mathsf{PAR}_k| = \binom{n}{k}$.)

Proof. Consider the simple tester described in Algorithm 1. It is plain that this

Algorithm 1 (Non-adaptive one-sided-error tester for the known-unknown setting)

let $q \leftarrow \frac{1}{\varepsilon}(2 + \ln|\mathrm{Isom}(f)|)$
for $i = 1$ to q **do**
    pick $x^i \in \{0,1\}^n$ uniformly at random
    query g on x^i
accept iff there exists $h \in \mathrm{Isom}(f)$ such that $g(x^i) = h(x^i)$ for all $i \in [q]$

is a non-adaptive one-sided error tester, and that it makes just $O((\log |\mathrm{Isom}(f)|)/\varepsilon)$ queries to g. So we only need to show that for any f and any g that is ε-far from f, the probability of acceptance is small. Indeed, for a fixed $h \in \mathrm{Isom}(f)$ the probability that $g(x^i) = h(x^i)$ for all $i \in [q]$ is at most $(1-\varepsilon)^q$. Applying the union bound on all functions $h \in \mathrm{Isom}(f)$, we can bound the probability of acceptance by $|\mathrm{Isom}(f)|\,(1-\varepsilon)^q \le |\mathrm{Isom}(f)|\,e^{-\varepsilon q} \le \exp(-2) < 1/3$. □

2.3.2 Lower bound

The lower bound of Theorem 2.3.1, the most interesting part, is obtained via the study of one-sided testers of isomorphism to k-parities. Note that if $f \in \mathsf{PAR}_k$, then testing isomorphism to f is the same as testing membership in PAR_k.

2.3.3. Lemma. *Let $f \in \mathsf{PAR}_k, g \in \mathsf{PAR}_{k'}$. Then*

$$\mathrm{distiso}(f,g) = \begin{cases} 0 & \text{if } k = k', \\ \frac{1}{2} & \text{if } k \neq k'. \end{cases}$$

Proof. It is easy to see that if $k = k'$ then $f \cong g$, whereas if $k \neq k'$,

$$\mathrm{distiso}(f,g) = \min_{\pi} \mathrm{dist}(f, g^\pi) = \min_{h \in \mathsf{PAR}_{k'}} \mathrm{dist}(f,h) = \min_{h \in \mathsf{PAR}_{k'}} \mathrm{dist}(f \oplus h, 0) = 1/2,$$

because whenever $h \in \mathsf{PAR}_{k'}, k' \neq k$, $f \oplus h$ is a parity on a non-empty set and therefore takes the value one on exactly half the inputs. □

Observe that testing isomorphism to k-parities is equivalent to testing isomorphism to $(n-k)$-parities. This is immediate from the fact that $p = p(x) \in \mathsf{PAR}_k$ if and only if $p'(x) \triangleq p(x) \oplus x_1 \oplus \cdots \oplus x_n \in \mathsf{PAR}_{n-k}$, making it possible to simulate queries to p by making queries to p' and vice versa.

2.3.4. Lemma. *Let $\varepsilon \in (0, \frac{1}{2}]$, $n \in \mathbb{N}$ and $k \in [0,n]$. Any ε-tester for PAR_k can be made into an ε-tester for PAR_{n-k}, preserving the query complexity, type of error, and adaptivity.*

Broadly speaking, we show that for a suitable choice of k', it is hard to tell $\mathsf{PAR}_{k'}$ and PAR_k apart. (Note however that sometimes this is a very easy task, even if k and k' are very close: if $k' = k+1$, a single query to the all-ones vector suffices.) But first we need to discuss one of the cornerstone results in extremal set theory.

Forbidden intersections

We say that r is a *forbidden intersection size* for a family of sets $\mathcal{F}$ if for no two distinct $A, B \in \mathcal{F}$ is $|A \cap B| = r$. If $\mathcal{F}$ is t-uniform (i.e., all its elements have size t), the following theorem puts an upper bound on the size of any such family. (Bounds for non-uniform families are also known, but are not strong enough for our intended usage.)

2.3.5. Theorem (Frankl & Wilson [FW81, Theorem 7b]).
Let $\frac{t}{2} \leq r < t \leq m$ be positive integers and suppose $t - r$ is a prime power. If $\mathcal{F} \subseteq \binom{[m]}{t}$ is a family of sets with forbidden intersection size r, then

$$|\mathcal{F}| \leq \frac{\binom{2t-1-r}{t}}{\binom{2t-1-r}{r}} \binom{m}{r}.$$

For the sake of completeness we present here the beautiful linear-algebraic proof of Theorem 2.3.5 found by Alon, Babai and Suzuki [ABS91] for the case of prime $t - r$ (which suffices for us); the general case where $t - r$ is a prime power is more complicated to prove. The interested reader can learn more about the so-called "linear algebra method" by consulting the unpublished manuscript of Babai and Frankl [BF92] and Chapter 13 of the book by Jukna [Juk11].

The proof is based on a certain "modular version" of the theorem, which generalizes a result of Ray–Chaudhuri and Wilson [RCW75]. Let $p \in \mathbb{N}$ and L a set of integers. We say that a family $\mathcal{F}$ is ($L \bmod p$)-intersecting if for all $a, b \in L$, $a \neq b$, it holds that $|a \cap b| \in L + p\mathbb{Z}$.

2.3.6. Theorem. *Let p be a prime and $\mathcal{F}$ be a t-uniform, ($L \bmod p$)-intersecting family of subsets of $[m]$, with $t \notin L + p\mathbb{Z}$. Then $\mathcal{F} \leq \binom{m}{|L|}$.*

Proof. Let us regard $\mathcal{F}$ as a subset of $\{0,1\}^m$ and associate with every $a \in \mathcal{F}$ the function $f_a \colon \{0,1\}^m \to \mathbb{F}_p$

$$f_a(x) = \prod_{l \in L} (\langle a, x \rangle - l).$$

Define, for each subset $I \subseteq [m]$ of size $|I| < |L|$, the function $g_I \colon \{0,1\}^m \to \mathbb{F}_p$

$$g_I(x) = \left(\sum_{j \in [m]} x_j - t \right) \prod_{i \in I} x_i.$$

Observe that

- For each $a \in \{0,1\}^m \subseteq \mathbb{F}_p$, $f_a(a) \neq 0$ (as $\langle a, a \rangle = |a| = t \notin L + p\mathbb{Z}$).

- For each $S \subseteq [m]$, $g_I(S) \neq 0$ if and only if $|S| \not\equiv t \pmod{p}$ and $S \supseteq I$. In particular $g_I(a) = 0$ for $a \in \mathcal{F}$.
- Both f_a and g_I can be represented as polynomials over $F_p[x_1, \ldots, x_m]$ by expanding the products. Moreover we can make these multilinear by replacing every occurrence of a power $x_i^j, j > 1$ in a monomial by x_i (which doesn't affect their evaluation on $\{0,1\}^m$).
- $\deg f_a, \deg g_I \leq |L|$.

The number of polynomials we have introduced is $|\mathcal{F}| + \binom{m}{\leq |L|-1}$. Below we show that they are all linearly independent over $\mathbb{F}_p$. Since the set of all multilinear polynomials of degree $\leq |L|$ over $\mathbb{F}_p$ is a vector space of dimension $\binom{m}{\leq |L|}$ (generated by the multilinear monomials of degree $\leq |L|$), we conclude

$$|\mathcal{F}| \geq \binom{m}{\leq |L|} - \binom{m}{\leq |L|-1} = \binom{m}{|L|}.$$

To prove linear independence, assume

$$\sum_{a \in \mathcal{F}} \lambda_a f_a + \sum_{|I| \leq |L|-1} \mu_I g_I = 0$$

for some sequences $\{\lambda_a\}, \{\mu_I\}$ of elements of $\mathbb{F}_p$. Evaluating at a_0 yields

$$\sum_{a \in \mathcal{F}} \lambda_a f_a(a_0) + \sum_{|I| \leq |L|-1} \mu_I g_I(a_0) = \lambda_{a_0} f_{a_0}(a_0) + 0 = 0,$$

i.e., $\lambda_{a_0} = 0$. So the linear dependence must occur among the g_I's. It is now easy to prove by induction on the inclusion poset (or in the size of I) that all μ_I's are zero as well. Indeed, let $|I| \leq |L| - 1$ and assume $\mu_J = 0$ for all $J \subset I$ (where the inclusion is strict). Evaluating now at I yields

$$\sum_{|J| \leq |L|-1} \mu_J g_J(I) = \sum_{|J| \subseteq I} \mu_J g_J(I) = 0 + \mu_I g_I(I) = 0,$$

so $\mu_I = 0$, completing the proof. □

Although we will not use this fact, it is not hard to see that this modular theorem can be applied directly to give the bound $|\mathcal{F}| \leq \binom{m}{t-r-1}$ in the case $t \geq 2r + 1$ of the non-modular problem, which is not covered by our statement of Theorem 2.3.5; see [FW81, Theorem 7a] for details. The case of interest to use requires a somewhat more involved argument:

Proof of Theorem 2.3.5 (for prime $t - r$). Assume otherwise. We intend to use the previous theorem and forbid intersection sizes that are a multiple of p

on a related set system. Write $d = 2r - t + 1 \in [1, t-1]$. As the family $\mathcal{F}$ is t-uniform, there must be a d-set $D \in \binom{[m]}{d}$ included in at least

$$|\mathcal{F}| \frac{\binom{t}{d}}{\binom{m}{d}}$$

elements of $\mathcal{F}$; to obtain this bound, calculate the size of the set

$$\left\{(A, D) \mid A \in \mathcal{F}, D \in \binom{A}{d}\right\}$$

and divide by the number of different d-subsets D of $[m]$. If we remove D from all elements of this subfamily, we obtain a $(t-d)$-uniform set system $\mathcal{G} \subseteq [m] \setminus D$ with forbidden intersection size $r - d = t - r - 1$. Let $a, b \in \mathcal{G}$ be distinct; then $0 \leq |a \cap b| < t - d = 2(t-r) - 1$. Since $t - r > 0$, in this setting $|a \cap b| \equiv t - r - 1 \mod (t-r)$ implies $|a \cap b| = t - r - 1 = r - d$. On applying Theorem 2.3.6 to $\mathcal{G}$ and $p = t - r$, $|L| = p - 1$, we find that

$$|\mathcal{G}| \leq \binom{m-d}{t-r-1} = \binom{m-d}{r-d},$$

implying

$$|\mathcal{F}| \leq \frac{\binom{m}{d}}{\binom{t}{d}} \binom{m-d}{r-d} = \binom{m}{r} \frac{r!(t-d)!}{t!(r-d)!} = \binom{m}{r} \frac{\binom{t+r-d}{t}}{\binom{t+r-d}{r}},$$

which simplifies to the expression given. □

It is interesting to note that the theorem ceases to hold when $t-r$ is not a prime power, as shown by Grolmusz [Gro99]. This suggests that this is an algebraic phenomenon rather than a purely combinatorial one. (However, in some special cases this assumption can be removed, as in the Frankl–Rödl theorems [FR87].)

Proof of the lower bound

The following result is implicit in [CGM11c]:

2.3.7. Theorem (Chakraborty et al.). *For all positive integers k, n such that $2 \leq k \leq n/2$, there exists $x \in \mathbb{F}_2^n$, $|x| \leq k/2$ with the following property:*

For any linear map $A\colon \mathbb{F}_2^n \to \mathbb{F}_2^q$, where $q \leq \frac{k}{8} \log\left(\frac{n}{k}\right)$, there is $y \in \mathbb{F}_2^n$, $|y| = k$ such that $Ax = Ay$.

Proof. We argue as follows. For any k we choose an appropriate $\ell = \ell(k) \in [k/2, k/4]$, define $k' = k - 2\ell$ and take an arbitrary $x \in \{0,1\}^n$ of weight k'. Set

$$Z = \{z \in \{0,1\}^n \mid |z| = 2\ell \text{ and } z \cap x = \emptyset\}.$$

We show that there are $z^1, z^2 \in Z$ with $Az^1 = Az^2$ and $|z^1 \cap z^2| = \ell$. The last equality implies $|z^1 \oplus z^2| = |z^1| + |z^2| - 2|z^1 \cap z^2| = 2\ell = k - k'$. This gives the result because then we can define $y = x \oplus z^1 \oplus z^2$ (the sum over $\mathbb{F}_2^n$, i.e., bitwise XOR), which satisfies $|y| = |x| + |z^1 \oplus z^2| = k$ and $Ay = Ax \oplus Az^1 \oplus Az^2 = Ax$.

If $k = 2$ we take $\ell = 1, k' = 0$ and the result follows from the pigeonhole principle: the image of each unit binary vector under A must fall into 2^q holes, and since $2^q < n$ (for small enough c), two of the elements of Z must have the same image. Likewise, if $k = 3$ we take $\ell = 1, k' = 1$ and use the fact that $2^q < \binom{n-1}{2}$.

So assume $k \geq 4$ and let ℓ be the largest prime $\leq k/2$; note that $\ell \geq k/4$ by Bertrand's postulate on the existence of a prime between $s+1$ and $2s-1$, proved by Chebyshev (see [AZ10, Chapter 2]). Alternatively, if we allow ourselves to use the full-fledged "prime power" version of Theorem 2.3.5, we can simply let ℓ be the largest power of two $\leq k/2$ and obtain the same bound. Also note $k' = k - 2\ell \leq \frac{k}{2}$.

Partition Z into disjoint subsets $\{Z_\alpha\}_{\alpha \in \{0,1\}^q}$ according to the different images under A:

$$Z_\alpha = \{z \in Z \mid Az = \alpha\}.$$

Fix α maximizing $|Z_\alpha|$. We can regard Z_α as a family of 2ℓ-subsets of $[m]$, where

$$m \triangleq n - k' = n - k + 2\ell \geq \frac{3}{4}n \geq 3\ell.$$

By our choice of α we must have

$$|Z_\alpha| \geq \frac{\binom{m}{2\ell}}{2^q}.$$

If ℓ were a forbidden intersection size for Z_α, Theorem 2.3.5 with $t = 2\ell, r = \ell$ would assert

$$|Z_\alpha| \leq \frac{\binom{3l-1}{2l}}{\binom{3l-1}{\ell}}\binom{m}{\ell} = \frac{1}{2}\binom{m}{\ell},$$

so

$$2^{q-1} \geq \frac{\binom{m}{2l}}{\binom{m}{\ell}} = \frac{(m-\ell)(m-\ell-1)\dots(m-2l+1)}{(2\ell)(2\ell-1)\dots(\ell+1)} \geq \left(\frac{m-\ell}{2\ell}\right)^\ell.$$

Since

$$\frac{m-\ell}{2\ell} = \frac{n-k+\ell}{2\ell} = \frac{n-k}{2\ell} + \frac{1}{2} \geq \frac{n-k}{k} + \frac{1}{2} = \frac{n}{k} - \frac{1}{2} \geq \frac{3}{4}\frac{n}{k}$$

(because $n/k \geq 2$), this would imply

$$q - 1 \geq \ell \log\left(\frac{m-\ell}{2\ell}\right) \geq \frac{k}{4}\log\left(\frac{3n}{4k}\right) \geq \frac{k}{8}\log\left(\frac{n}{k}\right),$$

contradicting our assumptions. □

Putting it all together we arrive at the following proposition, which implies Theorem 2.3.1. (The $\Omega(n)$ lower bound for k-juntas for $k \geq n/2$ is a consequence of the $\Omega(n)$ lower bound for $k' \triangleq \lfloor n/2 \rfloor$ because $\mathrm{Jun}_{k'} \subseteq \mathrm{Jun}_k$.)

2.3.8. Proposition. *Let $\varepsilon = \frac{1}{2}$. The following holds for all $n \in \mathbb{N}$:*

- *For any $k \in [2, n-2]$, the query complexity of testing PAR_k with one-sided error is $\Theta\big(\log \binom{n}{k}\big)$. Furthermore, the upper bound is obtainable with a non-adaptive tester, while the lower bound applies to adaptive tests, and even to the certificate size[2] for proving membership in PAR_k.*
- *For any $k \in \{0, 1, n-1, n\}$, the query complexity of testing PAR_k with one-sided error is $\Theta(1)$.*

Proof. Recall that we can assume $k \leq n/2$. The upper bound in the first item follows by Proposition 2.3.2. It is also easy to verify that the second item holds for $k = 0$ (i.e., the case when f is the constant zero function). For $k = 1$, the bound follows from [PRS02], who show that one-sided-error testing of functions for being a 1-parity (monotone dictatorship) can be done with $O(1)$ queries. (This also follows from the more general junta tests.)

Now we turn to the lower bound. We are left with the range $2 \leq k \leq n/2$. Let $a^1, \ldots, a^q$ be the (adaptive, random) queries made by a (q, ε)-tester on g and take for A the map $y \in \mathbb{F}_2^n \to (\langle a^1, y\rangle, \ldots, \langle a^q, y\rangle) \in \mathbb{F}_2^q$. If $q = o\big(\log \binom{n}{k}\big)$, we can pick x as in Theorem 2.3.7 and let $k' = |x| \neq k$. Then there is $y \in \mathbb{F}_2^n$, $|y| = k$ such that $Ax = Ay$, i.e., the result of the queries made on the parity $g = x^*$ are precisely the same as if the input were the k-parity $f = y^*$. Hence any q-query one-sided algorithm is forced to accept g, even though $\mathrm{distiso}(g, \mathsf{PAR}_k) = 1/2$. □

In fact this supplies a lower bound of $\Omega(\log \binom{n}{k})$ on the number of queries needed to provide a certificate that the number of influential variables of a parity is k rather than $k' \leq k/2$.

2.3.9. Remark. The reader may wonder about the query complexity for smaller ε. A lower bound of $\Omega(\frac{1}{\varepsilon})$ applies to testing any non-trivial property. In fact, it is not hard to conclude that, for $\varepsilon < \frac{1}{2}$, the testing query complexities in each of the two is are $\Theta\big(\log \binom{n}{k} + \frac{1}{\varepsilon}\big)$ and $\Theta(\frac{1}{\varepsilon})$, respectively. Only the $O(\log \binom{n}{k} + \frac{1}{\varepsilon})$ bound is not apparent. To see it, first run the BLR test to reject if f is 1/4-far from linear. This test takes $O(1)$ queries. If this test passes with high probability, then there is a unique parity g with $\mathrm{dist}(f, g) < 1/4$, therefore the distance from f to any other parity is larger than $1/2 - 1/4 \geq 1/4$. Now we make $O(\log \binom{n}{k})$ random queries; with high probability, no k-parity can agree with f on all of them, except possibly g (as the analysis of the algorithm of Proposition 2.3.2 with $\varepsilon = 1/4$

[2] By this we mean the size of the smallest set of inputs such that the evaluations of $f\colon \{0,1\}^n \to \{0,1\}$ on those inputs allow us to prove that $f \in \mathsf{PAR}_k^n$, assuming $f \in \mathsf{PAR}^n$.

shows). So if some parity is consistent with the responses to all these queries, it must be g and we can identify it. Finally, if this happens then we can take $O(1/\varepsilon)$ additional queries to test f for equality with g.

2.4 Linear lower bound for two-sided testers

We saw in Section 2.3.1 that, for the task of ε-testing isomorphism to a given function $f\colon \{0,1\}^n \to \{0,1\}$, $O(\frac{n \log n}{\varepsilon})$ queries always suffice. For constant ε, which is the primary focus here, this is $\widetilde{O}(n)$; our next result is a nearly matching lower bound of $\Omega(n)$ that applies to *almost all* functions f.

2.4.1. Theorem. *Fix a constant $0 < \varepsilon < \frac{1}{2}$. For a $1 - o(1)$ fraction of the functions $f\colon \{0,1\}^n \to \{0,1\}$, any algorithm for ε-testing isomorphism to f must make $\Omega(n)$ queries.*

As it turns out, these bounds can be generalized to $\tilde{\Theta}(k)$ for testing isomorphism to k-juntas (see Chapter 3).

A similar theorem was found by the independent works of Alon and Blais [AB10]; and Chakraborty, García–Soriano, and Matsliah [CGM11c] (albeit the results stated in both papers are somewhat weaker). We describe here the improved argument presented in [ABC+11].

The proof of Theorem 2.4.1 is non-constructive, but as will be shown in Chapter 6, the hardest functions to test isomorphism to may have relatively simple descriptions, which allows us to derive new lower bounds for other testing tasks. Moreover, in Chapter 5 we will see that k-parities exemplify the $\Omega(k)$ lower bound.

For the proof of Theorem 2.4.1, we fix a function f enjoying some "regularity" properties. Then we introduce two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ such that a function $g \sim \mathcal{D}_{\text{yes}}$ is isomorphic to f and a function $g \sim \mathcal{D}_{\text{no}}$ is ε-far from isomorphic to f with overwhelming probability, and then proceed to show indistinguishability of the two distributions with $o(n)$ adaptive queries. By the latter we mean that, when faced with a function h drawn from the mixed distribution $(\mathcal{D}_{\text{yes}} + \mathcal{D}_{\text{no}})/2$, the tester cannot tell if h belongs to $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$.

A first idea for $\mathcal{D}_{\text{no}}$ may be to make it uniform distribution over all boolean functions $\{0,1\}^n \to \{0,1\}$. However, it is possible for a tester to collect a great deal of information from inputs with very small or very large weight. In particular, just by querying the n-bit strings $\overline{0}$ and $\overline{1}$ we would obtain a tester that succeeds with probability 3/4 in distinguishing $\mathcal{D}_{\text{yes}}$ from $\mathcal{D}_{\text{no}}$ if $\mathcal{D}_{\text{no}}$ were completely uniform. This is because 0 and 1 remain invariant under permutations, and two random boolean functions agree on them with probability 1/4. To prevent an algorithm from gaining information by querying inputs of very small or very large weight, the functions appearing in both distributions are the same outside the middle layers of the hypercube. We remark that such a "truncation" is essential for this result to hold in full strength—one can prove that random permutations

of *any* f can be distinguished from completely random functions with $\widetilde{O}(\sqrt{n}\,)$ queries and high success probability. To do this one can use ideas from the graph isomorphism tester of Fischer and Matsliah [FM08], reducing the problem to testing the equivalence of a samplable distribution with an explicitly given one, which can be solved by an algorithm of Batu et al. [BFF$^+$01].

The indistinguishability result we prove is obtained via probabilistic techniques. We borrow ideas from the work of Babai and Chakraborty [BC08b], who proved query-complexity lower bounds for testing isomorphism to uniform hypergraphs. However, in order to be applicable to our problem, we have to extend the method of [BC08b] in several ways. One of the main differences is that, because of the need to consider trimmed functions, we have to deal with general sets of permutations (as opposed to groups of permutations) in the proof that a random permutation "shuffles" the values of a function uniformly. To compensate for this lack of structure, we show that any large enough set of permutations that are "independent" in some technical sense has the regularity property we need. Then the result for general sets is established by showing that any large enough set of permutations can be decomposed into a number of such sets. This can be deduced from the celebrated theorem of Hajnal and Szemerédi [HS69] on equitable colorings.

2.4.1 Regularity and additional definitions

To prove lower bounds for ε-testing isomorphism to a function f, it suffices to show the stronger claim that one can choose g such that both of the following conditions hold:

1. No tester can reliably distinguish between the cases where a function h is a random permutation of f or a random permutation of g;
2. $\operatorname{distiso}(f, g) \geq \varepsilon$.

2.4.2. Definition. Let $f, g\colon \{0,1\}^n \to \{0,1\}$ be boolean functions and $\varepsilon > 0$. Consider the distribution $\mathcal{D} = (\operatorname{Isom}(f) + \operatorname{Isom}(g))/2$ obtained by choosing a random permutation of f with probability half, and a random permutation of g with probability half.

We say that the pair (f, g) is (q, ε)-*hard* if $\operatorname{distiso}(f, g) \geq \varepsilon$ and no tester with oracle access to $h \sim \mathcal{D}$ can determine if $h \cong f$ or $h \cong g$ with overall success probability $\geq 2/3$ unless it makes more than q queries.

The existence of a q-hard pair f, g implies a lower bound of $q + 1$ on the query complexity of testing isomorphism to f (or to g, for that matter). The function g will be defined to agree with f on all unbalanced inputs, as defined below.

2.4.3. Definition. A query (or input) $x \in \{0,1\}^n$ is *balanced* if $\frac{n}{2} - 2\sqrt{n} \leq |x| \leq \frac{n}{2} + 2\sqrt{n}$. Otherwise, we say that x is *unbalanced*.

Note that the fraction of unbalanced inputs is

$$2^{-n} \sum_{|i-n/2|>2\sqrt{n}} \binom{n}{i} < 2\exp(-8) < 1/1000$$

by standard estimates on the tails of the binomial distribution.

2.4.4. Definition. For every f, a *random f-truncated function* is a random function uniformly drawn from the collection of all $g\colon \{0,1\}^n \to \{0,1\}$ satisfying $g(x) = f(x)$ for all unbalanced x.

2.4.5. Lemma. *Fix $0 < \varepsilon < \frac{1}{2}(1 - 10^{-3})$. For any function $f\colon \{0,1\}^n \to \{0,1\}$, a random f-truncated function g is ε-close to isomorphic to f with probability at most $o(1)$.*

We write

$$\{0,1\}^n_{\frac{n}{2}\pm h} \triangleq \{x \in \{0,1\}^n \mid \tfrac{n}{2} - h \le |x| \le \tfrac{n}{2} + h\}.$$

Proof. Let $N \triangleq \left|\{0,1\}^n_{\frac{n}{2}\pm 2\sqrt{n}}\right| = \Omega(2^n)$ and $\eta \triangleq 1 - (2^{n+1}/N)\varepsilon > 0$. For any $\pi \in S_n$, note that $\operatorname{dist}_{n/2\pm 2\sqrt{n}}(f^\pi, g) = (2^n/N)\operatorname{dist}(f^\pi, g)$, where the term on the left-hand side denotes the relative distance when the domain is $\{0,1\}^n_{n/2\pm 2\sqrt{n}}$ (and therefore is the expected sum of N independent unbiased binary random variables). Then, by the additive Chernoff bound,

$$\begin{aligned} \Pr\left[\operatorname{dist}_{n/2\pm 2\sqrt{n}}(f^\pi, g) < (2^n/N)\varepsilon\right] &= \Pr\left[\operatorname{dist}_{n/2\pm 2\sqrt{n}}(f^\pi, g) < (1-\eta)/2\right] \\ &\le \exp(-N\eta^2/4) \\ &\le o(\tfrac{1}{n!}). \end{aligned}$$

Taking the union bound over all choices of $\pi \in S_n$ completes the proof. □

In the rest of this section and all its subsections, we assume $\varepsilon < \frac{1}{2}(1 - 10^{-3})$. See Remark 2.4.16 in Section 2.4.2 for the details on how to deal with any $\varepsilon < \frac{1}{2}$.

Let $\mathcal{T}$ denote any deterministic, non-adaptive algorithm that attempts to test f-isomorphism with at most q queries to an unknown function g (where $q = \Omega(n)$ is a parameter to be determined later). Let $Q \subseteq \{0,1\}^n$ be the set of queries performed by $\mathcal{T}$ on f. We partition the queries in Q in two: the set Q_b of balanced queries, and the set Q_u of unbalanced queries.

The tester cannot distinguish f from g by making only unbalanced queries. Some unbalanced queries, however, could conceivably yield useful information to the tester and let it distinguish f from g with only a small number of balanced queries. The reason for this concern is that the set of responses to the unbalanced

queries might drastically reduce the number of candidate permutations that the tester needs to consider. The next proposition shows that this is not the case, and that little information is conveyed by the responses to unbalanced queries. We will apply it to the case where Q is a set of unbalanced queries.

2.4.6. Definition. For a fixed function $f\colon \{0,1\}^n \to \{0,1\}$, a set Q of queries, and $a\colon Q \to \{0,1\}$, the *set of permutations of f compatible with Q and a* is

$$\Pi_f(Q,a) \triangleq \{\pi \in S_n \mid f^\pi\restriction_Q = a\}$$

2.4.7. Lemma. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, any set Q of queries, and any $0 < t < 1$,*

$$\Pr_{\pi\in S_n}\left[\left|\Pi_f(Q, f^\pi\restriction_Q)\right| < t\cdot\frac{n!}{2^{|Q|}}\right] < t.$$

This implies that when the unknown function g is truncated according to f, with high probability the set $\Pi_f(Q_u, g^\pi\restriction_{Q_u})$ is large, which will be useful later.

Proof. For every $a \in \{0,1\}^{|Q|}$, let $S_a \subseteq S_n$ be the set of permutations σ for which $f^\sigma\restriction_Q = a$. A set S_a is *t-small* if $|S_a| < t\frac{n!}{2^{|Q|}}$. The union of all t-small sets covers fewer than $2^{|Q|}\cdot t\frac{n!}{2^{|Q|}} = tn!$ permutations, so the probability that a randomly chosen one belongs to a t-small set is less than t. □

We now examine the balanced queries.

2.4.8. Definition. Write any set Q of queries as $Q = Q_u \cup Q_b$, where the queries in Q_b are balanced and those in Q_u are not.

Let $n, q \in \mathbb{N}$. We say that a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is *q-regular* if for every $Q = Q_u \cup Q_b$ of total size at most q, and every pair of functions $a_b\colon Q_b \to \{0,1\}, a_u\colon Q_u \to \{0,1\}$ such that $|\Pi_f(Q_u, a_u)| \geq \frac{1}{3}\frac{n!}{2^{2q}}$,

$$\left|\Pr_{\pi\in\Pi_f(Q_u,a_u)}\left[f^\pi\restriction_{Q_b} = a_b\right] - 2^{-q}\right| < \tfrac{1}{6}\cdot 2^{-q}.$$

It is easy to see that "at most q" may be replaced with "exactly q" in the definition, as long as q does not surpass the total number of unbalanced inputs. Also note that whether f is regular or not depends only on the values it takes on balanced inputs. This restriction is necessary for $\Omega(n)$-regularity to be possible, since the condition implies in particular the existence of $\Omega(2^q)$ elements in the orbit of any 1-query set under S_n.

Definition 2.4.8 is useful because two functions f, g that are both regular and agree on unbalanced inputs will be hard to tell apart, as they both resemble random functions on balanced inputs. This holds no matter how f is defined on unbalanced inputs. This is formalized in the following lemma:

2.4.9. Lemma. *If f, g are q-regular with respect to, identical on unbalanced inputs, and satisfy* $\mathrm{distiso}(f, g) \geq \varepsilon$, *then the pair (f, g) is (q, ε)-hard.*

Proof. Consider the following two distributions:

- $\mathcal{D}_{\mathrm{yes}}$: pick $\pi \in S_n$ uniformly at random, and return f^π.
- $\mathcal{D}_{\mathrm{no}}$: pick $\pi \in S_n$ uniformly at random, and return g^π.

By definition, any $h_1 \in \mathcal{D}_{\mathrm{yes}}$ is isomorphic to f, whereas any $h_2 \in \mathcal{D}_{\mathrm{no}}$ is isomorphic to g and hence ε-far from isomorphic to f.

Let $Q = Q_u \cup Q_b$ be any set of at most q queries (where Q_u are unbalanced and Q_b balanced), and $a = (a_u, a_b)$ any set of $|Q|$ responses. We show that

$$\Pr_{\pi \in S_n}\left[f^\pi\restriction_Q = a\right] - \Pr_{\pi \in S_n}\left[g^\pi\restriction_Q = a\right] < \tfrac{1}{3}2^{-q}.$$

There are two cases to consider.

Case 1: $|\Pi_f(Q_u, a_u)| < \frac{1}{3}\frac{n!}{2^{2q}}$. In this case, we have $\Pr_\pi\left[f^\pi\restriction_{Q_u} = a_u\right] \leq \frac{1}{3}2^{-q}$ by Lemma 2.4.7. This immediately implies that $\Pr_\pi\left[f^\pi\restriction_Q = a\right] \leq \frac{1}{3}2^{-q}$ also.

Case 2: $|\Pi_f(Q_u, a_u)| \geq \frac{1}{3}\frac{n!}{2^{2q}}$. Note that

$$\begin{aligned}\Pr_\pi\left[f^\pi\restriction_Q = a\right] &= \Pr_\pi\left[f^\pi\restriction_{Q_u} = a_u\right] \cdot \Pr_\pi\left[f^\pi\restriction_{Q_b} = a_b \mid f^\pi\restriction_{Q_u} = a_u\right] \\ &= \Pr_\pi\left[f^\pi\restriction_{Q_u} = a_u\right] \cdot \Pr_{\pi \in \Pi_f(Q_u, a_u)}\left[f^\pi\restriction_{Q_b} = a_b\right] \\ &= (1 \pm \delta)2^{-q} \cdot \Pr_\pi\left[f^\pi\restriction_{Q_u} = a_u\right],\end{aligned}$$

where $\delta < 1/6$. The second equality transforms a probability expectation into a uniform probability over a subset of S_n, namely $\Pi_f(Q_u, a_u)$. The last line uses the lower bound on the size of this set and the regularity of f.

Similarly, by the regularity of g,

$$\begin{aligned}\Pr_\pi\left[g^\pi\restriction_Q = a\right] &= (1 \pm \delta)2^{-q}\Pr_\pi\left[g^\pi\restriction_{Q_u} = a_u\right] \\ &= (1 \pm \delta)2^{-q}\Pr_\pi\left[f^\pi\restriction_{Q_u} = a_u\right],\end{aligned}$$

because f and g are defined identically on unbalanced inputs. (We can choose the same $\delta < 1/6$ for both.) Therefore, for any $a\colon Q \to \{0, 1\}$,

$$\Pr_\pi\left[f^\pi\restriction_Q = a\right] - \Pr_\pi\left[g^\pi\restriction_Q = a\right] < \tfrac{1}{3}2^{-q}\Pr_\pi\left[f^\pi\restriction_{Q_u} = a_u\right] \leq \tfrac{1}{3}2^{-q},$$

and an appeal to Lemma 1.5.6 establishes the claim. □

The main step of the proof of existence of regular functions in the next section is to show that any sufficiently "uniform" family of functions contains regular functions.

2.4.10. Definition. A distribution $\mathcal{F}$ of boolean functions on $\{0,1\}^n$ is *r-uniform* if it is r-independent and uniform on sets of r balanced inputs, i.e., if for all $Q_b \in \binom{\{0,1\}^n}{\frac{n}{2}\pm 2\sqrt{n}}{r}$ and $a\colon Q_b \to \{0,1\}$,

$$\Pr_{f\in\mathcal{F}}\left[f\restriction_{Q_b} = a\right] = 2^{-r}.$$

For example, the uniform distribution over all boolean functions is 2^n-uniform. The reason we deal with more general distributions is that they are required to establish the existence of relatively simple functions that are hard to test isomorphism to (see Chapter 6).

2.4.2 Existence of regular functions

The main tool we need is the following:

2.4.11. Proposition. *Let $\mathcal{F}$ be an n^4-uniform distribution over boolean functions. Then a random element of $\mathcal{F}$ is $\left(\frac{n}{3} - 2\lceil\log n\rceil\right)$-regular with probability $1 - o(1)$.*

Before providing the proof, we show how it implies Theorem 2.4.1.

2.4.12. Theorem. *Fix any $0 < \varepsilon < \frac{1}{2}$. Let $f\colon \{0,1\}^n \to \{0,1\}$ be chosen at random from an n^4-uniform distribution $\mathcal{F}$, and let $g\colon \{0,1\}^n \to \{0,1\}$ be a random f-truncated function. Then with probability $1 - o(1)$, the pair (f,g) is $(\Omega(n),\varepsilon)$-hard.*

Hence, for most functions $f\colon \{0,1\}^n \to \{0,1\}$, testing f-isomorphism requires $\Omega(n)$ queries.

Proof. Recall that we are assuming $\varepsilon < \frac{1}{2}(1 - 10^{-3})$; see Remark 2.4.16 below to see how to handle larger ε. For some $q = \Omega(n)$ we can pick one q-regular function f from $\mathcal{F}$ by Proposition 2.4.11. The distribution of functions drawn from $\mathcal{F}$ and truncated according to f is also n^4-uniform, so a random such g is also q-regular with probability $1 - o(1)$. Also with probability $1 - o(1)$ we have $\operatorname{distiso}(f,g) = \Omega(1)$.[3] By the union bound, g satisfies both conditions with probability $1 - o(1)$. By Lemma 2.4.9, the pair f, g is (q,ε)-hard. The "hence"

[3]The proof is the same as that of Lemma 2.4.5, except that we use n^4-independence in place of full independence, and employ the variation of Chernoff bounds stated below in Theorem 2.4.15. This leads to a bound of $\exp(-\Omega(n^4))$ instead of $\exp(-\Omega(2^n))$, but is still $o(1/n!)$.

part follows by taking for $\mathcal{F}$ the uniform distribution among all functions. □

Proof of Proposition 2.4.11. Let $q \triangleq \frac{n}{3} - 2\log n$ and $r \triangleq n^4$. Fix a set $Q = Q_u \cup Q_b$ of q queries, and functions $a_u \colon Q_u \to \{0,1\}, a_b \colon Q_b \to \{0,1\}$. For any $f \colon \{0,1\}^n \to \{0,1\}$, let $S \triangleq \Pi_f(Q_u, a_u)$ and assume its size is $|S| \geq \frac{1}{3}\frac{n!}{2^{2q}}$. For each $\pi \in S$, define the indicator variable $X(f,\pi) \triangleq \mathbb{I}[f^\pi\restriction_{Q_b} = a_b]$ and define $A(f) \triangleq \Pr_{\pi\in S}[X(f,\pi) = 1]$. We aim to compute the probability, over a random $f \in \mathcal{F}$, that $A(f)$ deviates from $p \triangleq 1/2^q$ by $p/6$ or more. Notice that $\mathbb{E}_f[A(f)] = \mathbb{E}_\pi \mathbb{E}_f X(f,\pi) = \mathbb{E}_\pi p = p$, where we made use of the r-uniformity of $\mathcal{F}$ and the fact that $r \geq q$.

Consider any pair $\sigma_1, \sigma_2 \in S$ such that $\sigma_1(Q_b) \cap \sigma_2(Q_b) = \emptyset$. Since $2q \leq r$, a random function from $\mathcal{F}$ assigns values independently to each element of $\sigma_1(Q_b) \cup \sigma_2(Q_b)$, so the random variables $X(f,\sigma_1)$ and $X(f,\sigma_2)$, viewed as a function of f, are independent conditioned on the choice of σ_1, σ_2.

More generally, for any fixed set of s permutations $\sigma_1, \ldots, \sigma_s$ of S under which the images of Q_b are pairwise disjoint, the s random variables $X(f,\sigma_1), \ldots, X(f,\sigma_s)$ are $r/q \geq n^3$-wise independent. We show that S can be partitioned into a number of large sets of permutations, each of them satisfying the pairwise disjointness property. To establish this claim we use the celebrated theorem of Hajnal and Szemerédi. The interested reader can find an elementary proof of this theorem in the paper [KK08].

2.4.13. Theorem (Hajnal & Szemerédi [HS69]). *Let G be an undirected graph on n vertices with maximum vertex degree $\Delta(G) \leq d$. Then G has a $(d+1)$-coloring in which all the color classes have size $\lfloor \frac{n}{d+1} \rfloor$ or $\lceil \frac{n}{d+1} \rceil$.*

2.4.14. Lemma. *Let S be a set of permutations on $[n]$ (with $n \geq 30$) and let Q_b be a set of at most $q < n$ balanced queries. Then there exists a partition $S_1 \dot\cup \cdots \dot\cup S_m$ of the permutations in S such that for $i \in [m]$,*

(i) $|S_i| \geq \frac{|S|}{n!}\frac{2^n}{e^9 n^2\sqrt{n}} - 1$, *and*

(ii) The sets $T_i = \{Q_b^\pi\}_{\pi\in S_i}$, for $i \in [m]$, are pairwise disjoint.

Note that we have no control over the number m of elements of the partition, although an upper bound is straightforward.

Proof. Construct a graph G on S where two permutations σ, τ are adjacent iff there exist $x, y \in Q_b$ such that $\sigma(x) = \tau(y)$ or $\sigma(y) = \tau(x)$. By this construction, when T is a set of permutations that form an independent set in G, the sets $\{Q_b^\pi\}_{\pi\in T}$ are pairwise disjoint. Observe that G is a regular graph (all vertices have the same degree), since $\sigma(x) = \tau(y)$ iff $(\tau^{-1} \circ \sigma)(y) = x$.

Let $N \triangleq \binom{n}{n/2-2\sqrt{n}} \geq \frac{2^n}{e^9\sqrt{n}}$ for $n \geq 30$. Note that for any $x, y \in \{0,1\}^n_{\frac{n}{2}\pm 2\lceil\sqrt{n}\rceil}$,

$$\Pr_{\pi\in S_n}[x^\pi = y] = \left\{ \begin{array}{ll} 0, & |x| \neq |y| \\ \frac{1}{\binom{n}{|x|}}, & |x| = |y| \end{array} \right\} \leq \frac{1}{N}.$$

This holds because the orbit of x under S_n (the collection of all x^π as π ranges over S_n) is the set of all $\binom{n}{|x|}$ strings of the same weight. So by applying the union bound over all choices of $x, y \in Q_b$, we get

$$\begin{aligned} \Pr_{\pi\in S_n}[\exists\, x, y \in Q_b \mid x^\pi = y] &\leq \mathop{\mathbb{E}}_{\pi\in S_n}\Big[\sum_{x,y\in Q_b} \mathbb{I}[x^\pi = y]\Big] \\ &= \sum_{x,y\in Q_b} \Pr_{\pi\in S_n}[x^\pi = y] \\ &\leq \frac{q^2}{N}, \end{aligned}$$

which allows us to upper bound the degree of G by $d \triangleq q^2 n!/N < n^2 n!/N$. Therefore, by the Hajnal-Szemerédi Theorem, G can be colored so that each color class has size at least

$$\left\lfloor \frac{|S|}{d+1} \right\rfloor \geq \frac{|S|}{n!}\frac{2^n}{2n^2\sqrt{n}} - 1.$$

This completes the proof. □

In our case $|S|/n! \geq 2^{-2q}/3$, and by our choice of q we conclude that each of the elements of the partition has size at least $|S_i| \geq n^3 \cdot 2^q$ for large enough n. Since $A(f)$ is a weighted average of the m random variables $Y_i(f) \triangleq \mathbb{E}_{\pi\in S_i} X(f,\pi)$, it is enough to show that with probability $1 - o(1)$,

$$|Y_i(f) - 2^{-q}| < 2^{-q}/6$$

holds simultaneously for all $i = 1, \ldots, m$.

Each quantity $Y_i(f)$ is the average of $|S_i|$ random variables that are n^3-wise independent, each satisfying $\mathbb{E}_f X(f,\pi) = 2^{-q} = p$. We apply the following version of Chernoff bounds for k-wise independent random variables:

2.4.15. Theorem (Schmidt et al. [SSS95]). *Let X be the sum of s k-wise independent random variables in the interval $[0,1]$, and let $p = \frac{1}{s}\mathbb{E}[X]$. For any $0 \leq \delta \leq 1$,*

$$\Pr[|X - p| \geq \delta p] \leq e^{-\Omega(\min(k, \delta^2 ps))}.$$

Since $p|S_i| \geq n^3$ and $k = n^3$, using the theorem above with $\delta = \frac{1}{6}$ we obtain that for all $i \in [k]$,

$$\Pr_{f\sim\mathcal{F}}[|Y_i(f) - p| \geq p\delta] \leq 2^{-\Omega(n^3)},$$

hence we can bound

$$\begin{aligned}\Pr_{f\sim\mathcal{F}}[|A(f)-p|\geq p\delta] &\leq \Pr_{f\sim\mathcal{F}}\left[\exists i\in[m]\;\middle|\;|Y_i(f)-p|\geq p\delta\right]\\ &\leq m\cdot 2^{-\Omega(n^3)}\\ &\leq n!\cdot 2^{-\Omega(n^3)}.\end{aligned}$$

To conclude the proof we apply the union bound over all possible choices of Q and $a=(a_u,a_b)\in\{0,1\}^Q$, yielding

$$\Pr_{f\in\mathcal{F}}\left[\exists\, Q,a\;\middle|\;|A(f)-p|\geq p/6\right]\leq\binom{2^n}{q}2^q n!2^{-\Omega(n^3)}=o(1).$$

□

2.4.16. Remark. It is not difficult to see that if one replaces $\frac{n}{2}\pm 2\sqrt{n}$ in the definition of balanced inputs with $\frac{n}{2}\pm c\sqrt{n}$ for some other constant $c>2$, the result still holds for *the same* lower bound q and large enough n. We refrained from doing so because it would introduce an additional parameter in all the definitions and proofs. The only place where this matters is in claiming the $\Omega(n)$ lower bound for *any* fixed $\varepsilon<\frac{1}{2}$. The value $c=2$ only suffices for $\varepsilon<\frac{1}{2}(1-10^{-3})$ because of Lemma 2.4.5, but choosing larger values can prove the theorem for any constant $\varepsilon<\frac{1}{2}$.

2.5 Testing isomorphism between two unknown functions

Finally, we examine the problem of testing two unknown functions for the property of being isomorphic. Here the tester needs to make queries to both f and g.

2.5.1. Theorem (Alon & Blais; Chakraborty et al. [AB10, CGM11c]). *For any fixed $\varepsilon>0$, the query complexity of testing isomorphism of two unknown functions $f,g\colon\{0,1\}^n\to\{0,1\}$ is $\tilde{\Theta}(2^{n/2})$.*

More concretely,

1. *There exists a non-adaptive ε-tester with one-sided error for function isomorphism in the unknown-unknown setting with query complexity*

$$O(2^{n/2}\sqrt{n\log n/\varepsilon}\,).$$

2. *Any adaptive tester for function isomorphism in the unknown-unknown setting must have query complexity $\Omega(2^{n/2}/n^{1/4})$.*

Proof of the upper bound

The tester is described in Algorithm 2. (Recall that ε is a constant, so for large enough n we have $\sqrt{n \ln n/(\varepsilon 2^n)} < 1$.) Obviously it is non-adaptive, has one-

Algorithm 2 (Non-adaptive one-sided error tester for the unknown-unknown setting)

1: generate a set Q by including every $x \in \{0,1\}^n$ in Q with probability $\sqrt{\frac{n \ln n}{\varepsilon 2^n}}$ independently at random
2: **if** $|Q| > 10\sqrt{\frac{2^n}{\varepsilon} n \ln n}$ **then** accept
3: query both f and g on all inputs in Q
4: accept iff there exists π such that for all $x \in Q$, either $f(x) = g(x^\pi)$ or $x^\pi \notin Q$

sided error and makes $O(2^{n/2}\sqrt{n \log n/\varepsilon})$ queries. Let f and g be ε-far up to isomorphism; we prove that the probability of the tester accepting is $o(1)$. We may assume that the event $|Q| \leq 10\sqrt{2^n n \ln n/\varepsilon}$ holds, since it occurs with probability $1 - o(1)$. For any permutation $\pi \in S_n$ there are at least $\varepsilon 2^n$ inputs $x \in \{0,1\}^n$ for which $f(x) \neq g(x^\pi)$. When x satisfies this inequality, the probability that both x and x^π belong to Q is at least $\frac{n \ln n}{\varepsilon 2^n}$, so the permutation π passes the acceptance condition in the last line of Algorithm 2 with probability no more than $(1 - n \ln n/(\varepsilon 2^n))^{\varepsilon 2^n} \leq e^{-n \ln n} = n^{-n} = o(1/n!)$. The claim follows by taking the union bound over all $n!$ permutations. □

In the next chapter we will also study the running time of a generalization of this algorithm.

Proof of the lower bound

Again we apply the Yao principle via Lemma 1.5.6. We define two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ on pairs of functions such that the two elements of any pair drawn from $\mathcal{D}_{\text{yes}}$ are isomorphic, while the elements of any pair drawn from $\mathcal{D}_{\text{no}}$ are 1/8-far from isomorphic with probability $1 - o(1)$. The distribution $\mathcal{D}_{\text{yes}}$ is constructed by letting the pair of functions be (f, f^π), where $f \in \mathcal{F}_{\frac{n}{2} \pm 2\sqrt{n}}$ is a random 0-truncated function on $\{0,1\}^n$ (see Definition 2.4.4) and $\pi \in S_n$ is a uniformly random permutation.

For the distribution $\mathcal{D}_{\text{no}}$ the pair of functions are two independently chosen random 0-truncated functions f and g; with probability $1 - o(1)$, $\text{distiso}(f, g) \geq 1/8$ (Lemma 2.4.5). For any set $Q = \{x^1, \ldots, x^t\} \subseteq \{0,1\}^n$ of t queries and any $p, q \in \{0,1\}^t$ let $\Pr_{(f,g) \in \mathcal{D}_{\text{yes}}}[(f,g)\restriction_Q = (p,q)]$ be the probability that for all $1 \leq i \leq t$, $f(x^i) = p_i$ and $g(x^i) = q_i$ when f and g are drawn according to $\mathcal{D}_{\text{yes}}$. Similarly we define $\Pr_{(f,g) \in \mathcal{D}_{\text{no}}}[(f,g)\restriction_Q = (p,q)]$.

Without loss of generality we may assume that $|x^i| \in [\frac{n}{2} - 2\sqrt{n}, \frac{n}{2} + 2\sqrt{n}\,]$ for all $i \in [t]$, since functions drawn from $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$ always take the value 0 on all

other inputs. If the pair f, g is drawn from $\mathcal{D}_{\text{no}}$, the answers to the queries will be uniformly distributed by definition, so for any $p, q \in \{0,1\}^t$, we have

$$\Pr_{(f,g)\in\mathcal{D}_{\text{no}}}[(f,g)\restriction_Q = (p,q)] = 1/2^{2t}.$$

Now let the pair be drawn according to $\mathcal{D}_{\text{yes}}$ and let π be the permutation that defined the pair. Let E_Q denote the event that Q^π and Q are disjoint, i.e., that for all $i, j \in [t]$, the inequality $\pi(x^i) \neq x^j$ holds. Conditioned on E_Q, the answers to the queries will again be distributed uniformly, that is

$$\Pr_{(f,g)\in\mathcal{D}_{\text{yes}}}[(f,g)\restriction_Q = (p,q) \mid E_Q] = \Pr_{(f,g)\in\mathcal{D}_{\text{no}}}[(f,g)\restriction_Q = (p,q)].$$

(Note that the event in question is independent of E_Q when the pairs are drawn from $\mathcal{D}_{\text{no}}$.)

Let us now show that the E_Q occurs with probability at least $\frac{3}{4}$. For any fixed $i, j \in [t]$, we have $\Pr_{\pi\in S_n}[\pi(x^i) = x^j] \leq 1/\binom{n}{n/2-2\sqrt{n}} \leq \frac{e^9\sqrt{n}}{2^n}$ because x^i is balanced. So by the union bound, when $t \leq 2^{n/2}/(200n^{1/4})$ we have

$$\Pr[E_Q] = 1 - \Pr[\exists\, i, j \in [t] \mid \pi(x^i) = x^j] \geq 1 - \frac{e^9 t^2\sqrt{n}}{2^n} > \frac{3}{4}.$$

Therefore,

$$\begin{aligned}\Pr_{(f,g)\in\mathcal{D}_{\text{yes}}}[(f,g)\restriction_Q = (p,q)] &\geq \Pr[E_Q] \cdot \Pr_{(f,g)\in\mathcal{D}_{\text{yes}}}[(f,g)\restriction_Q = (p,q) \mid E_Q] \\ &> \tfrac{3}{4} \cdot \Pr_{(f,g)\in\mathcal{D}_{\text{no}}}[(f,g)\restriction_Q = (p,q)].\end{aligned}$$

By Lemma 1.5.6, this implies that the success probability of any tester that makes fewer than $2^{n/2}/(200n^{1/4})$ queries is at most $5/8 + o(1) < 2/3$ and completes the proof of the lower bound in Theorem 2.5.1. □

2.6 Summary

We studied the problem of testing isomorphism between two boolean functions. Our main focus is on the most well-studied case, where one of the functions is known explicitly in advance and the other one may be queried; in this setup we proved upper and lower bounds for testing isomorphism to functions, both with one-sided and two-sided error. We also studied the analogous problem when both functions are unknown. All our bounds are nearly tight.

Chapter 3
Testing and deciding junta isomorphism

The content of this chapter is based on the papers

- S. Chakraborty, D. García–Soriano, and A. Matsliah. Efficient sample extractors for juntas with applications. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 545–556, 2011.
- S. Chakraborty, D. García–Soriano, Y. Goldhirsh, and A. Matsliah. Deciding and approximating isomorphism efficiently for specific classes of boolean functions. Manuscript, 2011.

3.1 Introduction

In this chapter we investigate the problem of testing isomorphism to k-juntas. With this goal in mind we develop the notion of *sample extractors for juntas*, which have found additional applications (see Chapter 6).

Making a slight detour from property testing per se, we also study the complexity of the following problems (where f and g are k-juntas, or at least close to k-juntas):

- decide if f and g are isomorphic;
- compute $\text{distiso}(f, g)$;
- approximate $\text{distiso}(f, g)$.

We will examine both the time and query complexities of our algorithms. Again, both the known-known setting and the unknown-unknown setting are considered.

We comment that, in a recent work, Arvind and Vasudev [AV11] have also studied the time complexity of deciding and approximating isomorphism of two unknown functions from restricted function classes. They obtain bounds for the

class of constant-depth circuits, based on a classical result of Linial, Mansour and Nisan [LMN93]. Our results are incomparable to theirs.

3.2 Influence, juntas and cores

A central tool to analyze juntas is a measure of the influence that a set of coordinates has on a function.

3.2.1. Definition. The influence of the set of coordinates $A \subseteq [n]$ on a function $f\colon \{0,1\}^n \to \mathcal{R}$ is defined as

$$\mathrm{Inf}_f(A) \triangleq \Pr_{x\in\{0,1\}^n,\ y\in\{0,1\}^{|A|}} \left[f(x) \neq f(x_{A\leftarrow y}) \right].$$

Thus $\mathrm{Inf}_f(A)$ measures the probability that the value of f changes after taking a random input x and rerandomizing the bits inside A. Note that when A is a singleton, this value is half that of the most common definition of the influence of one variable (see, e.g., [Wol08]); for consistency we stick to Definition 3.2.1 in this case as well.

3.2.2. Definition. An index (variable) $i \in [n]$ is *relevant* for f if $\mathrm{Inf}_f(\{i\}) \neq 0$. A *$k$-junta* ($k \geq 1$) is a function that has at most k relevant variables; equivalently, a function f that satisfies $\mathrm{Inf}_f([n] \setminus J) = 0$ for some $J \in \binom{[n]}{k}$.

Jun_k will denote the class of k-juntas (on n variables), and for $A \subseteq [n]$, Jun_A will denote the class of juntas whose relevant variables are all contained in A.

To illustrate these two definitions, observe that every relevant variable of a k-parity ($k \geq 1$) has influence $\frac{1}{2}$ (which is as large as it can get).

3.2.3. Lemma. *Given $A \subseteq [n]$ and oracle access to $f\colon \{0,1\}^n \to \{0,1\}$, there is an $O(\log(1/\delta)/\eta)$-query one-sided test that accepts if $\mathrm{Inf}_f(A) = 0$ and rejects if $\mathrm{Inf}_f(A) \geq \eta$, with confidence $1-\delta$. Its running time is $O(n)$ times its query complexity.*

Proof. Note that $\mathrm{Inf}_f(A)$ is the expectation (over $x \in \{0,1\}^n, y \in \{0,1\}^{|A|}$) of the indicator function

$$\mathbb{I}\left[f(x) \neq f(x_{A\leftarrow y}) \right].$$

Suppose we draw a random pair of inputs $x, z \in \{0,1\}^n$ conditioned on $x_{[n]\setminus A} = z_{[n]\setminus A}$, and reject iff $f(x) \neq f(z)$. This step always accepts if $\mathrm{Inf}_f(A) = 0$, whereas if $\mathrm{Inf}_f(A) \geq \eta$ the pair drawn catches this fact with probability at least η. By drawing t independent pairs we can reduce the error probability to $(1-\eta)^t < e^{-\eta t}$. The lemma follows by taking $t = O(\log(1/\delta)/\eta)$. □

Such a test is called an *independence test* for A. Its outcome is said to be *positive* if it accepts, meaning that no evidence was found for $\mathrm{Inf}_f(A) \neq 0$.

Influence can be expressed in terms of the Fourier expansion of f. (See the survey by de Wolf [Wol08] for an introduction to Fourier analysis of boolean functions.) It is simpler for this purpose to work with the associated function h taking values in $\{-1, 1\}$ (for example, map 0 to 1 and 1 to -1 to obtain $h = (-1)^f$; this does not affect the outcome of equality comparisons). The multiplicative characters of $\mathbb{Z}_2^n$ are represented by the symbols χ_S (where $S \subseteq [n]$) and are defined by $\chi_S(x) = (-1)^{|x \cap S|}$. Let $x \subseteq [n], z \subseteq A$. Note that

$$\mathop{\mathbb{E}}_{x \subseteq [n]} [h(x) \cdot h(x \oplus z)] = (h \star h)(z) = \sum_S \widehat{h}(S)^2 \chi_S(z),$$

where "$\star$" denotes convolution; to see this, recall that the Fourier coefficient of a convolution $(f \star g)$ at S is $\widehat{f}(S) \cdot \widehat{g}(S)$. Hence

$$\mathop{\mathbb{E}}_{\substack{z \subseteq A \\ x \subseteq [n]}} [h(x)h(x \oplus z)] = \sum_{S \subseteq [n]} \widehat{h}(S)^2 \left(\mathop{\mathbb{E}}_{z \subseteq A} [\chi_S(z)] \right) = \sum_{S \subseteq [n] \setminus A} \widehat{h}(S)^2, \tag{3.1}$$

because the factor within brackets vanishes whenever S intersects A, and is 1 otherwise. Then

$$2 \mathop{\Pr}_{\substack{z \subseteq A \\ x \subseteq [n]}} [h(x) \neq h(x \oplus z)] = 1 - \mathop{\mathbb{E}}_{\substack{z \subseteq A \\ x \subseteq [n]}} [h(x)h(x \oplus z)] = \sum_{S \cap A \neq \emptyset} \widehat{h}(S)^2,$$

the last equality due to (3.1) and Parseval's identity. Therefore

$$\mathrm{Inf}_f(A) = \mathrm{Inf}_h(A) = \frac{1}{2} \sum_{\substack{S \subseteq [n] \\ S \cap A \neq \emptyset}} \widehat{h}(S)^2. \tag{3.2}$$

3.2.4. Lemma (Fischer et al. [FKR+04]). *Influence is monotone and subadditive.*

That is, for all $f \colon \{0,1\}^n \to \{0,1\}$ and $A, B \subseteq [n]$,

$$\mathrm{Inf}_f(A) \leq \mathrm{Inf}_f(A \cup B) \leq \mathrm{Inf}_f(A) + \mathrm{Inf}_f(B).$$

The proof is a simple consequence of (3.2). In fact something stronger than mere subadditivity holds:

3.2.5. Lemma (Fischer et al. [FKR+04]). *Influence is monotone submodular. That is, for all $A, B, C \subseteq [n]$,*

$$0 \leq \mathrm{Inf}_f(A \cup B \cup C) - \mathrm{Inf}_f(A \cup B) \leq \mathrm{Inf}_f(A \cup C) - \mathrm{Inf}_f(A).$$

This is also known as the law of diminishing marginal returns of influence.

Proof. For monotone functions, the condition above is equivalent to

$$\mathrm{Inf}_f(S \cup T) + \mathrm{Inf}_f(S \cap T) \leq \mathrm{Inf}_f(S) + \mathrm{Inf}_f(T)$$

for all $S, T \subseteq [n]$. The difference between the right and the left hand sides is a sum of non-negative numbers, as can be checked from (3.2):

$$\mathrm{Inf}_f(S) + \mathrm{Inf}_f(T) - (\mathrm{Inf}_f(S \cup T) + \mathrm{Inf}_f(S \cap T)) = \frac{1}{2} \sum_{\substack{Z \subseteq [n] \setminus (S \cap T) \\ Z \cap S \neq \emptyset \\ Z \cap T \neq \emptyset}} \widehat{h}(Z)^2,$$

because the Fourier mass of h at any Z not included in the right-hand sum is counted an equal number of times in the sums corresponding to $\mathrm{Inf}_f(S) + \mathrm{Inf}_f(T)$ and $\mathrm{Inf}_f(S \cup T) + \mathrm{Inf}_f(S \cap T)$. □

Any junta is determined by its set of relevant variables together with its *core*.

3.2.6. Definition. Fix an arbitrary ordering of $[n]$. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a k-junta on J. The *k-core* of f is the boolean function $\mathrm{core}_k(f)\colon \{0,1\}^k \to \{0,1\}$ obtained from f^* by dropping the (irrelevant) variables outside J, i.e., satisfying

$$f(x) = \mathrm{core}_k(f)(x\restriction_J).$$

Note that this definition depends on the fixed ordering of n, the number k, and possibly on the choice of J as well (if f is actually a k'-junta for $k' < k$). Sometimes we will simply refer to the core of f, where k is clear from context (and J is any k-set containing the relevant variables of f). Different choices for J lead to the same k-core, up to isomorphism.

3.3 From k-juntas to n-juntas

Proposition 2.3.2 showed how to test isomorphism to $f\colon \{0,1\}^n \to \{0,1\}$ with $O(n \log n)$ queries. Assume for the moment that both the known function f and the unknown function g are k-juntas. Then both f and g are determined by their cores, up to isomorphism. Suppose we could query the core of g, rather than g itself. This would enable us to approximate, in a similar way, the distance between $\mathrm{core}_k(f)$ and $\mathrm{core}_k(g)$, bringing the sample complexity down to $O(k \log k)$.

The quantity $\mathrm{distiso}(\mathrm{core}_k(f), \mathrm{core}_k(g))$ is a good approximation to $\mathrm{distiso}(f, g)$, although in general these two measures do not need to coincide.

3.3.1. Lemma (Chakraborty et al. [CGM11c]). *Let $f, g\colon \{0,1\}^n \to \{0,1\}$ be k-juntas, where $0 < k \leq n$. Then*

(a) $\operatorname{distiso}(\operatorname{core}_k(f), \operatorname{core}_k(g)) \in [\operatorname{distiso}(f,g), 2\operatorname{distiso}(f,g)]$.

(b) If $l \geq 2k$ *then* $\operatorname{distiso}(\operatorname{core}_l(f), \operatorname{core}_l(g)) = \operatorname{distiso}(f,g)$.

As regards the first part, note that both inequalities are tight. For example, if $k = n$ then obviously $\operatorname{distiso}(f,g) = \operatorname{distiso}(\operatorname{core}_k(f), \operatorname{core}_k(g))$. On the other hand, consider the case $n = k+1$, $f(x) = |x_1 + x_2 + \cdots + x_k| \bmod 2$ and $g(x) = 1 - f(x)$. Then $\operatorname{distiso}(f,g) = 1/2$ but $\operatorname{distiso}(\operatorname{core}_k(f), \operatorname{core}_k(g)) = 1$.

Proof.

We can assume without loss of generality that f and g are juntas on $[k]$, as distiso is permutation-invariant.

(a) Let $f' = \operatorname{core}_k(f), g' = \operatorname{core}_k(g)$. We prove

$$\operatorname{distiso}(f,g) \geq \frac{1}{2}\operatorname{distiso}(f',g')$$

(the other inequality $\operatorname{distiso}(f,g) \leq \operatorname{distiso}(f',g')$ being obvious).

Take π minimizing $\operatorname{dist}(f, g^\pi)$. The function f is a junta on $[k]$, while g^π is a junta on $\pi^{-1}([k])$. Let $A \triangleq \pi^{-1}([k]) \setminus [k]$, $B \triangleq \pi^{-1}([k]) \cap [k]$, $C \triangleq [k] \setminus B$; note that $|A| = |C|$ (because π is bijective and thus $|\pi^{-1}([k])| = |\pi(k)| = k$). Roughly speaking, if f and g^π are close then both must be close to a junta on B, because the coordinates outside B are irrelevant to either f or g^π.

Every input $x \in \{0,1\}^n$ is the interleaving of four strings

$$a \in \{0,1\}^A, b \in \{0,1\}^B, c \in \{0,1\}^C, r \in \{0,1\}^{[n]-(A \cup B \cup C)}$$

in the right order, i.e., $x = \sigma(a,b,c,r)$ for some permutation $\sigma\colon [n] \to [n]$, where $(a,b,c,r) = a \sqcup b \sqcup c \sqcup r$ denotes concatenation. Hence there are permutations σ_1, σ_2 of $[k]$, independent of x, for which

$$\begin{aligned} f(x) &= f'^{\sigma_1}(b,c), \\ g^\pi(x) &= g'^{\sigma_2}(b,a). \end{aligned}$$

For every $b \in \{0,1\}^B$ and $i,j \in \{0,1\}$, let

$$\begin{aligned} p^b_{ij} &\triangleq \Pr_a\left[f'^{\sigma_1}(b,a) = i \wedge g'^{\sigma_2}(b,a) = j\right] \\ &= \Pr_c\left[f'^{\sigma_1}(b,c) = i \wedge g'^{\sigma_2}(b,c) = j\right]. \end{aligned}$$

Obviously $p^b_{01} + p^b_{10} = \Pr_a\left[f'^{\sigma_1}(b,a) \neq g'^{\sigma_2}(b,a)\right] \leq 1$, so

$$p^b_{01} + p^b_{10} \geq (p^b_{01} + p^b_{10})^2 \geq 4p^b_{01}p^b_{10}.$$

For random x, the variables a, b, c are mutually independent. For every b we can compute

$$\begin{aligned}
\Pr_{a,c}\left[f'^{\sigma_1}(b,c) \neq g'^{\sigma_2}(b,a)\right] &= \Pr_{c}\left[f'^{\sigma_1}(b,c)=0\right]\cdot\Pr_{a}\left[g'^{\sigma_2}(b,a)=1\right] \\
&\quad + \Pr_{c}\left[f'^{\sigma_1}(b,c)=1\right]\cdot\Pr_{a}\left[g'^{\sigma_2}(b,a)=0\right] \\
&= (p^b_{00}+p^b_{01})(p^b_{01}+p^b_{11}) + (p^b_{10}+p^b_{11})(p^b_{00}+p^b_{10}) \\
&\geq p^b_{01}(p^b_{00}+p^b_{01}+p^b_{11}) + p^b_{10}(p^b_{00}+p^b_{10}+p^b_{11}) \\
&= p^b_{01}(1-p^b_{10}) + p^b_{10}(1-p^b_{01}) \\
&= \frac{p^b_{01}+p^b_{10}}{2} + \frac{p^b_{01}+p^b_{10}-4p^b_{01}p^b_{10}}{2} \\
&\geq \frac{p^b_{01}+p^b_{10}}{2} \\
&= \frac{\Pr_a\left[f'^{\sigma_1}(b,a) \neq g'^{\sigma_2}(b,a)\right]}{2}.
\end{aligned}$$

Hence, by taking expectations over b,

$$\begin{aligned}
\operatorname{dist}(f, g^\pi) &= \Pr_{a,b,c}\left[f'^{\sigma_1}(b,c) \neq g'^{\sigma_2}(b,a)\right] \\
&\geq \frac{1}{2}\Pr_{b,a}\left[f'^{\sigma_1}(b,a) \neq g'^{\sigma_2}(b,a)\right] \\
&= \frac{1}{2}\operatorname{dist}(f'^{\sigma_1}, g'^{\sigma_2}),
\end{aligned}$$

and we are done because $\operatorname{distiso}(f,g) = \operatorname{dist}(f, g^\pi) = \operatorname{dist}(f'^{\sigma_1}, g'^{\sigma_2})$.

(b) It is not hard to see that

$$\operatorname{distiso}(f,g) = \min_{\substack{C\subseteq[k] \\ |C|\leq n-k \\ \jmath:[k]\setminus C\to[k] \\ \jmath \text{ injective}}} \mathop{\mathbb{E}}_{c\colon C\to\{0,1\}}\left[\operatorname{dist}(\operatorname{core}_k(f), (\operatorname{core}_k(g)\restriction_{C\leftarrow c})^\jmath)\right], \quad (3.3)$$

where $h\restriction_{C\leftarrow c}\colon \{0,1\}^k \to \{0,1\}$ denotes the $(k-|C|)$-junta obtained from a function $h\colon \{0,1\}^k \to \{0,1\}$ by fixing the variable x_i to the constant value $c(i)$ for all $i \in C$. To verify this, observe that in trying to minimize $\operatorname{dist}(f, g^\pi)$ for two juntas f, g with relevant variables in $[k]$, we can select the optimal π in stages: first choose a set $C \subseteq [k]$ such that $|C| \leq n-k$, then pick an injection $\jmath$ from $[k]\setminus C$ to $[k]$ and finally let π satisfy $C = [k]\setminus\pi^{-1}([k])$ and $\pi\restriction_{[k]\setminus C} = \jmath$ (the concrete choice in the last step does not actually matter).

If we substitute l for n in (3.3), the right-hand side is the same for any $l \geq 2k$, for the condition $|C| \leq l-k$ becomes redundant. Hence

$$\operatorname{distiso}(\operatorname{core}_l(f), \operatorname{core}_l(g)) = \operatorname{distiso}(\operatorname{core}_n(f), \operatorname{core}_n(g)) = \operatorname{distiso}(f,g).$$

□

The implication is that, in order to approximate distiso(f, g) between two k-juntas f and g to within a factor of two, it suffices to compute the isomorphism distance between their k-cores. In fact, if an exact answer is needed instead of a 2-approximation, we can reduce the problem to a calculation of the isomorphism distance between their $2k$-cores.

As a corollary of item (a) we obtain the lower bound we promised the reader in Section 2.4.

3.3.2. Corollary. *Let $f, g\colon \{0,1\}^n \to \{0,1\}$ be k-juntas.*

If $(\mathrm{core}_k(f), \mathrm{core}_k(g))$ is (q, ε)-hard, then (f, g) is $(q, \varepsilon/2)$-hard. Hence for any $\varepsilon' \in (0, \frac{1}{4})$ there are pairs of r-juntas that are $(\Omega(k), \varepsilon')$-hard, and it takes $\Omega(k)$ queries to test isomorphism to them.

Proof. The "hence" part follows from the first statement on taking $\varepsilon = 2\varepsilon' \in (0, \frac{1}{2})$ and applying Theorem 2.4.1.

Let $f' = \mathrm{core}_k(f), g' = \mathrm{core}_k(g)$. It is clear that $f \cong g$ if $f' \cong g'$. Let there be an algorithm $\mathcal{A}$ capable of distinguishing a random permutation of f from a random permutation of g with high probability. Based on $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that makes the same number of queries and decides whether $h'\colon \{0,1\}^k \to \{0,1\}$ is a random permutation of f' or a random permutation of g', in the following manner. Let $h(x) \triangleq h'(x_1 \dots x_k)$. The algorithm $\mathcal{B}$ picks a uniformly random permutation $\sigma \in S_n$, and applies $\mathcal{A}$ to h^σ. Clearly, any query to h^σ can be simulated by one query to h' as σ is known to $\mathcal{B}$. The distribution of h^σ is a random permutation of either f or g.

Together with the inequality distiso$(f, g) \geq$ distiso$(f', g')/2$, this completes the proof. □

Likewise we obtain a lower bound for the unknown-unknown-case analogous to that of Theorem 2.5.1. We omit the proof, which mimics that of Corollary 3.3.2.

3.3.3. Corollary. *Testing isomorphism of two unknown k-juntas requires*

$$\Omega\left(\frac{2^{k/2}}{k^{1/4}}\right)$$

queries.

3.4 Exact algorithms for k-juntas

In this section we give algorithms for the isomorphism decision and isomorphism distance problem for k-juntas (where k is known). We start with a simple randomized solution and then discuss how to derandomize it.

One of the main issues we will be concerned with is how to obtain the truth table of $\mathrm{core}_k(g)$ without reading the truth table of g in its entirety (and of course, without knowing the influential variables in advance).

3.4.1 Randomized algorithm

3.4.1. Definition. Let $\ell, n > 0$. A *random (ordered) partition* $\mathcal{I} = (I_1, \ldots, I_\ell)$ of $[n]$ into ℓ sets is constructed by starting with ℓ empty buckets, and then putting each coordinate $i \in [n]$ into one of the buckets picked uniformly at random.

An *isolating partition* for a set $J \subseteq [n]$ is a partition of $[n]$ into sets, each of which contains at most one coordinate in J.

Sometimes we also refer to the components of a partition as *blocks* or *parts*. Note that the blocks in a random partition will usually have different sizes, although in expectation the size of a block will be n/ℓ. A random partition is well defined even if $\ell > n$ (in which case some elements of the partition will be empty no matter what). Unless explicitly mentioned otherwise, $\mathcal{I}$ will always denote a random partition $\mathcal{I} = (I_1, \ldots, I_\ell)$ of $[n]$ into ℓ subsets, where ℓ is even (the only significance of the last condition is that it allows for some cleaner equalities in Section 3.7.2); and $\mathcal{J} = (J_1, \ldots, J_k)$ shall denote an ordered k-subset of $\mathcal{I}$.

Suppose g is a junta on J, $|J| \leq k$. By elementary probability estimates, with high probability a random partition will isolate J.

3.4.2. Lemma. *The probability that a random partition $\mathcal{I}$ into ℓ buckets fails to isolate a given set J of size k is upper bounded by $k(k-1)/(2\ell)$.*

Proof. For any two distinct elements $a, b \in J$, where $|J| = k$, the probability that both a and b land into the same bucket of the partition is $1/\ell$ by definition. By the union bound, the probability that there exist two such elements of J is at most $\binom{k}{2}/\ell$. □

Let J be the set of relevant variables of a k-junta g. Given an isolating partition for J, we can extract the truth table of $\mathrm{core}_k(g)$ (up to some permutation of the variables) by making 2^ℓ queries to g — setting all of the variables in a block to the same value also guarantees that the influential variable inside this block (which is either unique or non-existent by assumption) was set to this value.

3.4.3. Definition. We say the string $y \in \{0,1\}^n$ is *$\mathcal{I}$-blockwise constant* if for every block I of $\mathcal{I}$, the restriction of y to I is constant; that is, if for all $i \in [\ell]$ and $j, j' \in I_i$, $y_j = y_{j'}$.

Given $z \in \{0,1\}^\ell$, define $\textsc{replicate}_{\mathcal{I}}(z)$ to be the $\mathcal{I}$-blockwise constant string $y \in \{0,1\}^n$ obtained by setting $y_j \leftarrow z_i$ for all $i \in \ell, j \in I_i$;

If we assume that

- $\mathcal{I}$ isolates J;

and

- we happen to know which components of $\mathcal{I}$ intersect J (say $I_{j_1}, \ldots, I_{j_{k'}}$),

then we can find the truth table of $\mathrm{core}_k(g)$ simply by looping over all $2^{k'}$ assignments to $z_{j_1}, \ldots, z_{j_{k'}}$, filling out the remaining bits of z arbitrarily, and making a query for $g(\textsc{replicate}_{\mathcal{I}}(z))$. The first condition above is true of a random partition, and it is also possible to satisfy the second:

3.4.4. Lemma. *Let g be a junta on an unknown k-set J. Given a partition $\mathcal{I}$ that isolates J, it is possible to identify all blocks of $\mathcal{I}$ that intersect J with $O(|\mathcal{I}| \log k \cdot 2^k)$ queries to g and constant success probability.*

Proof. Observe that the influence of every *relevant* variable of a k-junta is at least $2(1 - 2^{-k})2^{-k} \geq 2^{-k}$. So the influence of every block containing a relevant variable is also at least 2^{-k}. By Lemma 3.2.3, with $O(\log k \cdot 2^k)$ queries we can determine if a block is relevant with one-sided error probability $1/(10k)$, say. We perform this test for each component of $\mathcal{I}$. There are no more than k influential blocks, and by the union bound we can identify them all with overall error probability $1/10$ or less. □

(As a matter of fact, it is also possible to use a junta tester with $\varepsilon = 2^{-k}$ to solve this task with $O(k \cdot 2^k + k \log |I|)$ queries.)

In the next section we get around this lemma by derandomizing the algorithm.

3.4.5. Lemma. *Given access to a k-junta $g\colon \{0,1\}^n \to \{0,1\}$, it is possible to construct, with high probability, a complete truth table for (a permutation of)* $\mathrm{core}_k(g)$ *with $2^k \cdot \mathrm{poly}(k)$ queries and in time $n \cdot 2^k \cdot \mathrm{poly}(k)$.*

Proof. Take a random partition $\mathcal{I}$ into $\Theta(k^2)$ parts, which is isolating by Lemma 3.4.2. Then apply Lemma 3.4.4 to find the relevant blocks and finally make 2^k queries of the form $\textsc{replicate}_{\mathcal{I}}(z)$ as explained above.

The time bound is due to the time complexity of the tester of Lemma 3.4.4 being $n \cdot 2^k \cdot \mathrm{poly}(k)$, and the time required to query the entire truth table being $O(n \cdot 2^k)$. □

Note that we cannot hope to obtain the entire truth table of $\mathrm{core}_k(g)$ with fewer than 2^k queries (or time less than $\Omega(n + 2^k)$).

Now recall Luks's function isomorphism algorithm, mentioned in Section 1.3.

3.4.6. Theorem (Luks [Luk99]). *There is an algorithm* $\textsc{ExactIso}(f, g)$ *that decides isomorphism of two boolean functions $f, g\colon \{0,1\}^k \to \{0,1\}$ in time $2^{O(k)}$.*

Composing Lemma 3.4.5 with Theorem 3.4.6 yields the following theorem. We state it for the problem of testing isomorphism between two k-juntas, but all results of this type also hold for testing isomorphism to a fixed k-junta.

3.4.7. Theorem (Chakraborty et al. [CGGM11]). *There exists a randomized algorithm that given two k-juntas $f, g\colon \{0,1\}^n \to \{0,1\}$, accepts with probability at least $2/3$ if they are isomorphic and rejects with probability at least $2/3$ if they are not. The algorithm runs in time $n \cdot 2^{O(k)}$ and makes $2^k \cdot \mathrm{poly}(k)$ queries to f and g.*

Alternatively, composing it with an exhaustive search among all possible permutations of $[k]$ yields:

3.4.8. Theorem (Chakraborty et al. [CGGM11]). *There exists a randomized algorithm that given two k-juntas $f, g\colon \{0,1\}^n \to \{0,1\}$, outputs $\mathrm{distiso}(f,g)$ with probability at least $2/3$. It runs in time $n \cdot 2^{O(k)} + 2^{O(k \log k)}$ and makes $2^k \cdot \mathrm{poly}(k)$ queries.*

3.4.2 Deterministic algorithm

In light of the above, for a deterministic algorithm we need the truth table of the core of a k-junta while avoiding the use of influence tests or junta testers (which are randomized of necessity if their query complexity does not depend on n). We use a k-perfect family of hash functions $h\colon [n] \to [m]$ that are expected to map each influential variable into a different block. We remind the reader that we are assuming that k is a known parameter (although standard techniques based on binary search could be used to remove such assumption).

3.4.9. Definition. A family $\mathcal{F}$ of functions $h_1, \ldots, h_s\colon [n] \to [m]$ is a k-perfect hash family if for every subset $J \in \binom{[n]}{\leq k}$ there exists some $h_j \in \mathcal{F}$ that is injective on J.

Some constructions of k-perfect hash families are given in [NSS95, SS90]. Any function h in such a family gives rise to a partition $\mathcal{I}$ of $[n]$ into m parts, the ith block of which ($i \in [m]$) is $h^{-1}(i)$.

The algorithm works by iterating over all functions in the k-perfect hash family and building a truth table based on the assumption that the function separates the influential variables. It is given a k-junta $f\colon \{0,1\}^n \to \{0,1\}$ and a explicit k-perfect hash family $h_1, \ldots, h_s$, and finds a hash function in the family that is injective on the influential variables of f. Then it outputs a truth table for $\mathrm{core}_k(f)$.

Algorithm 3 GetCoreDet$(f, h_1, \ldots, h_s)$

```
for i = 1 to s do
    construct the h_i-truth table of f
    let k_i^f be the number of influential variables in the h_i-truth table
let i_0 be the index maximizing k_i^f
return the h_{i_0}-truth table of f
```

3.4.10. Lemma. *Given a k-junta $f\colon \{0,1\}^n \to \{0,1\}$ and a explicitly given k-perfect hash family $h_1, \ldots, h_s\colon [n] \to [k]$, GetCoreDet outputs a truth table for* $\mathrm{core}_k(f)$. *The time complexity of GetCoreDet is $O(sn \cdot 2^k)$, and the number of queries to f it makes is $O(s \cdot 2^k)$.*

Proof. First, since $h_1, \ldots, h_s$ is a k-perfect hash family and both the input functions are k-juntas, we are guaranteed that for each input function there is a hash function which separates its influential variables. This is the function which maximizes the number of influential variables in its h_i-truth table, which is therefore a truth table for a permutation of $\mathrm{core}_k(f)$. The truth table can be found after finding the partition of $[n]$ defined by h_i (which takes $O(n)$ time), and querying all 2^k blockwise-constant inputs (which takes $O(n \cdot 2^k)$ time). Then the number of influential variables can be found in time $O(k \cdot 2^k)$. The overall time and query complexities result from iterating over the members of the k-perfect hash family. □

Plugging in the construction of a globally explicit k-perfect hash family of Naor, Schulman and Srinivasan [NSS95, Theorem 3(iii)] we get $s = e^k k^{O(\log k)} \log n$ and the time complexity for constructing the family and evaluating each of its elements on every possible input is linear in the total description size. From these observations and Luks's algorithm the following follows.

3.4.11. Theorem (Chakraborty et al. [CGGM11]). *There exists a deterministic algorithm that given two k-juntas $f, g\colon \{0,1\}^n \to \{0,1\}$, accepts if they are isomorphic and rejects if they are not. The algorithm runs in time $n \log n \cdot 2^{O(k)} k^{O(\log k)}$.*

3.4.12. Theorem (Chakraborty et al. [CGGM11]). *There exists a deterministic algorithm that given two k-juntas $f, g\colon \{0,1\}^n \to \{0,1\}$, outputs the value* $\mathrm{distiso}(f, g)$. *It runs in time $n \log n \cdot 2^{O(k)} k^{O(\log k)} + 2^{O(k \log k)}$.*

3.5 Introducing sample extractors

3.5.1 Motivation

The algorithms described thus far suffer from several shortcomings. One is that their query complexity is exponential in k, while we strive for query complexity

$\widetilde{O}(k)$ if we are to match the lower bound of Corollary 3.3.2. (Even in the unknown-unknown setting, their query complexity, at least 2^k, goes well beyond the lower bound of roughly $2^{k/2}$ given by Corollary 3.3.3.) Finally, we have been assuming throughout that g is a k-junta, which needs not hold true (in which case the core of g is not even defined).

We attempt to apply these algorithms to the function

$$g^* = \text{the closest } k\text{-junta to } g.$$

This at least defines a k-junta whose core we would like to query, but a new problem is introduced as we do not have access to g^*, only to a "noisy" approximation thereof. The key to an efficient resolution is to notice that being able to query the core of g^* is overkill. All the tester from Proposition 2.3.2 needs is to draw $O(n \log n)$ random samples (x, a), where $x \in \{0,1\}^n$ is uniformly distributed and $a = g^*(x)$; this sufficed to obtain a good estimate for $\mathrm{distiso}(f, g^*)$. Therefore the task would be solved if we somehow managed to get samples from the *core* of g^*. This motivates the notion of "noisy sample extractor" (or "noisy sampler" for short).

3.5.1. Definition. Let $h\colon \{0,1\}^k \to \{0,1\}$ be a function, and let $\eta, \mu \in [0,1)$. An (η,μ)-*noisy sample extractor for* h is a probabilistic algorithm $\widetilde{h}$ that on each execution outputs $(x,a) \in \{0,1\}^k \times \{0,1\}$ such that

- for all $\alpha \in \{0,1\}^k$, $\Pr[x = \alpha] = \frac{1}{2^k}(1 \pm \mu)$;
- $\Pr[a = h(x)] \geq 1 - \eta$;
- the pairs output on each execution of $\widetilde{h}$ are mutually independent.

An η-noisy sampler is an $(\eta, 0)$-noisy sampler, i.e., one which on each execution picks a uniformly random x.[1]

The next theorem will be proved shortly.

3.5.2. Theorem (Chakraborty et al. [CGM11b]).
Construction of efficient noisy samplers:
Let $\theta_{3.5.2}(k,\varepsilon) = (\varepsilon/2400)^6/(10^{26}k^{10}) = \mathrm{poly}(k/\varepsilon)$.
There are algorithms A_P, A_S *(resp. preprocessor and sampler) that have oracle access to a function* $g\colon \{0,1\}^n \to \{0,1\}$, *and satisfy the following properties:*
The preprocessor A_P *takes as input* $\varepsilon > 0$ *and* $n, k \in \mathbb{N}$, *makes* $O(k/\varepsilon + k \log k)$ *adaptive queries to* g *and can either reject, or accept and return a state* $\alpha \in \{0,1\}^{O(n)}$. *Assuming* A_P *accepted, the sampler* A_S *can be called on demand, with*

[1] The reader familiar with [CGM11c] should beware that the usage of the parameter μ here is slightly different from the similar definition therein.

state α as an argument; in each call, A_S makes only one query to g and outputs a pair $(x, a) \in \{0,1\}^k \times \{0,1\}$.

On termination of the preprocessing stage A_P, all the following conditions are fulfilled simultaneously with probability at least $4/5$:

- *If g is $\theta_{3.5.2}(k,\varepsilon)$-close to a k-junta, A_P accepted g;*
- *If g is $\varepsilon/2400$-far from a k-junta, A_P rejected g;*
- *If A_P accepted, $A_S(\alpha)$ is an $\varepsilon/90$-noisy sampler for $\mathrm{core}_k(g^*)^\pi$ (where π is some permutation of $[k]$).*

The running time of A_P is $O\left(\frac{n\cdot k}{\varepsilon} + \mathrm{poly}(\frac{k}{\varepsilon})\right)$, and the running time of each call to A_S is $O(n)$.

The statement is somewhat technical and calls for careful reading. The fact that the last condition should be satisfied with high probability for *any* g plays a crucial role. When $\theta_{3.5.2}(k,\varepsilon) < \mathrm{dist}(g, \mathrm{Jun}_k) < \varepsilon/2400$, it might be the case that A_P always accepts g, or always rejects g, or anything in between, but with high probability either g has been rejected *or* an $\varepsilon/90$-noisy sampler for (a permutation of) $\mathrm{core}_k(g^*)$ has been constructed. We prove Theorem 3.5.2 in Section 3.7.

It is notable that such samples from the core of g^* can indeed be efficiently obtained (allowing for some noise), even though g is the only function we have access to. In fact even having query access to g^* itself would not seem to help much on initial consideration, because the location of the relevant variables of g^* is unknown to us, and cannot be found without introducing a dependence on n in the query complexity.

3.5.3. Remark. The preprocessing stage makes adaptive queries, while the sampler is non-adaptive. It is possible to make the preprocessor non-adaptive too by switching to the non-adaptive junta tester (see Section 3.6). This incurs an overhead of $O(k^4 \log k)$ in the query complexity of the preprocessor.

3.5.2 Approximating distiso

The $O(k \log k/\varepsilon)$-query upper bound for k-juntas would seem to be a matter of instantiating Proposition 2.3.1 and Theorem 3.5.2. This is not quite so because the sample extractor is allowed to make errors. So instead of Proposition 2.3.1 we need an isomorphism tester that is still guaranteed to work when f is merely close enough to a k-junta g. In future chapters we shall make use of a more general result, dealing with the estimation of the distance between f and the closest element of a class $\mathcal{S}$ of functions.

3.5.4. Lemma (Chakraborty et al. [CGM11b]). *There is an algorithm $\mathcal{A}$ that given $\varepsilon \in \mathbb{R}^+$, $k \in \mathbb{N}$, a set $\mathcal{S}$ of boolean functions on $\{0,1\}^k$, and an η-noisy sampler $\widetilde{g}$ for some $g\colon \{0,1\}^k \to \{0,1\}$, where $\eta \leq \varepsilon/90$, satisfies the following:*

- *if* $\operatorname{dist}(g, \mathcal{S}) < \varepsilon/10$, $\mathcal{A}$ *accepts with probability at least* $9/10$*;*
- *if* $\operatorname{dist}(g, \mathcal{S}) > 9\varepsilon/10$, $\mathcal{A}$ *rejects with probability at least* $9/10$*;*
- *Algorithm* $\mathcal{A}$ *makes* $O\left(\frac{1+\log|\mathcal{S}|}{\varepsilon}\right)$ *calls to the noisy sampler.*

If each call to the noisy sampler takes time $O(1)$ and every $f \in \mathcal{S}$ can be evaluated on any given input in $O(n)$, then $\mathcal{A}$ runs in time $O\left(\frac{n|\mathcal{S}|\log|\mathcal{S}|+n}{\varepsilon}\right)$.

Proof.

Consider the following variation of Algorithm 1 (on page 21). It is clear that

Algorithm 4 Tolerant, noise-resilient isomorphism tester

1: let $q \leftarrow \frac{1}{\varepsilon}(90 + 800 \ln|\mathcal{S}|)$.
2: obtain q independent samples $(x^1, a^1), \ldots, (x^q, a^q)$ from $\tilde{g}$
3: accept if and only if $\min_{h\in\mathcal{S}} \left|\{i \in [q] \mid h(x^i) \neq a^i\}\right| < \varepsilon q/2$

the time and query complexities are as stated.

For $h \in \mathcal{S}$, write $\delta_h \triangleq \operatorname{dist}(h, g)$ and let $\Delta_h \subseteq \{0,1\}^k$ be the set of inputs on which h and g disagree, where $|\Delta_h| = \delta_h 2^k$. Since the x's are independent and uniformly distributed random variables, we have $\Pr_x[x \in \Delta_h] = \delta_h$. Also let Λ_h be a random variable representing the fractional disagreement between h and g in the sample:

$$\Lambda_h = \frac{\left|\{i \in [q] \mid h(x^i) \neq g(x^i)\}\right|}{q}.$$

If $\operatorname{distiso}(g, \mathcal{S}) > 9\varepsilon/10$, then for any fixed $h \in \mathcal{S}$ the probability that Λ_h is at least $4\varepsilon/5$ can be bounded by using the Chernoff inequality in its multiplicative form:

$$\Pr\left[\Lambda_h < 8\varepsilon/10 \leq (1-1/9)\delta_h\right] = e^{-(1/9)^2(9/10)\varepsilon q/2} < \frac{1}{20|\mathcal{S}|}.$$

Hence with probability $19/20$, $\Lambda_h \geq 8\varepsilon/10$ for all $h \in \mathcal{S}$. To relate this to the fraction of samples (x, a) for which $h(x) \neq a$, we use Markov's inequality:

$$\begin{aligned}\Pr\left[\left|\{i \in [q] \mid a^i \neq g(x^i)\}\right| \geq (3/10)\varepsilon q\right] &\leq \Pr\left[\left|\{i \in [q] \mid a^i \neq g(x^i)\}\right| \geq 27\eta q\right] \\ &\leq 1/27. \qquad (3.4)\end{aligned}$$

Hence with probability at least $9/10$,

$$\min_{h\in\mathcal{S}} \left|\{i \in [q] \mid h(x^i) \neq a^i\}\right| > \varepsilon q/2.$$

On the other hand, if $\operatorname{distiso}(g, \mathcal{S}) < \varepsilon/10$, picking $h \in \mathcal{S}$ with $\operatorname{dist}(g, h) < \varepsilon/10$ we obtain in the same way

$$\Pr\left[\left|\{i \in [q] \mid h(x^i) \neq g(x^i)\}\right| > 2\varepsilon/10 \geq 2\delta_h q\right] \leq e^{-(1/10)\varepsilon q/3} < \frac{1}{20}$$

(no union bound is needed here). As (3.4) continues to hold, we conclude in this case that with probability at least $9/10$,

$$\min_{h \in \mathcal{S}} \left|\{i \in [q] \mid h(x^i) \neq a^i\}\right| < 2\varepsilon q/5 < \varepsilon q/2.$$

□

3.5.5. Remark. Note that this algorithm does **not** provide an estimate of $\operatorname{dist}(g, \mathcal{S})$ with additive accuracy $O(\varepsilon)$, because when $\operatorname{dist}(g, \mathcal{S})$ is large the approximation obtained is only good up to constant multiplicative factors. This meets our requirements. Nonetheless, it is equally easy to obtain an algorithm that estimates $\operatorname{dist}(g, \mathcal{S})$ up to, say, $\varepsilon/10$, by turning the $1/\varepsilon$ factor into $O(1/\varepsilon^2)$. The analysis would then use the additive Chernoff bounds.

3.5.6. Theorem (Chakraborty et al. [CGM11b]). *Let $\varepsilon > 0, k \in \mathbb{N}^+$ and let $\mathcal{C}_{[k]} \subseteq \mathrm{Jun}_{[k]}$ be a class of juntas on the first k variables that is closed under permutations of $[k]$. There is a randomized algorithm $A_{3.5.6}$ that given ε, k and oracle access to a function $f\colon \{0,1\}^n \to \{0,1\}$ does the following:*

- *if $\operatorname{distiso}(f, \mathcal{C}_{[k]}) < \theta_{3.5.2}(k, \varepsilon)$, $A_{3.5.6}$ accepts with probability at least $7/10$;*
- *if $\operatorname{distiso}(f, \mathcal{C}_{[k]}) \geq \varepsilon$, $A_{3.5.2}$ rejects with probability at least $7/10$;*
- *$A_{3.5.6}$ makes $O\left(k \log k + \frac{k + \log |\mathcal{C}_{[k]}|}{\varepsilon}\right)$ queries to f.*

If every $g \in \mathcal{C}_{[k]}$ can be evaluated on any given input in $O(k)$, then the algorithm runs in time

$$O\left(\frac{n(k + \log |\mathcal{C}_{[k]}|)}{\varepsilon} + \frac{k|\mathcal{C}_{[k]}| \log |\mathcal{C}_{[k]}|}{\varepsilon} + \operatorname{poly}\left(\frac{k}{\varepsilon}\right)\right).$$

Proof. Let $\tau \triangleq \theta_{3.5.6}(k, \varepsilon)$ and let f^* be the closest k-junta to f. Suppose first that $\operatorname{distiso}(f, \mathcal{C}_{[k]}) < \tau$. Then Theorem 3.5.2 asserts that, with probability at least $4/5$, we can construct an $\varepsilon/90$-noisy sampler for $\operatorname{core}_k(f^*)$. Since $\operatorname{dist}(f, f^*) \leq \operatorname{dist}(f, \mathcal{C}_{[k]}) \leq \tau$, we actually obtain an $\varepsilon/90$-noisy sampler for a function that is $2\tau < \varepsilon/10$-close to the core of some $g \in \mathcal{C}_{[k]}$. Using this noisy sampler we may apply the algorithm from Lemma 3.5.4 with

$$\mathcal{S} = \operatorname{core}_k(\mathcal{C}_{[k]}) \triangleq \{\operatorname{core}_k(f) \mid f \in \mathcal{C}_{[k]}\},$$

which in turn will accept with probability at least $9/10$. The overall acceptance probability in this case is at least $7/10$ by the union bound.

Now consider the case $\operatorname{distiso}(f, \mathcal{C}_{[k]}) \geq \varepsilon$. There are two possible sub-cases:

$\mathrm{dist}(f, \mathrm{Jun}_k) \geq \varepsilon/2400$**:** In this case f is rejected with probability at least $4/5 > 7/10$.

$\mathrm{dist}(f, \mathrm{Jun}_k) < \varepsilon/2400$**:** In this case, with probability at least $4/5$, either f is rejected (in which case we are done), or an $\varepsilon/90$-noisy sampler has been constructed for $\mathrm{core}_k(f^*)$. Since f^* is $\varepsilon/2400$-close to f, by the triangle inequality we have $\mathrm{dist}(\mathrm{core}_k(f^*), \mathrm{core}_k(\mathcal{C}_{[k]})) \geq \mathrm{distiso}(f, \mathcal{C}_{[k]}) - \mathrm{dist}(f, f^*) > 9\varepsilon/10$, and hence with probability at least $9/10$ the algorithm from Lemma 3.5.4 rejects. Thus the overall rejection probability in this case is at least $7/10$ too.

The assertion about the number of queries is easily seen to be correct, as it is the sum of the number of queries made in the preprocessing stage by A_P (Theorem 3.5.2), and the number of executions of the sampler A_S. As to the time complexity, recall that the preprocessing stage takes time $O(\frac{nk}{\varepsilon} + \mathrm{poly}(k/\varepsilon))$. Each of the calls to the noisy sampler takes $O(n)$, and the time complexity of the Algorithm of Lemma 3.5.4 excluding these calls is $O((k|\mathcal{S}| \log |\mathcal{S}| + k)/\varepsilon)$. □

The next few corollaries all have a similar flavor.

3.5.7. Corollary. *Let $\varepsilon > 0$ and suppose $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta. Then it is possible to ε-test isomorphism between $g\colon \{0,1\}^n \to \{0,1\}$ and f with $O(k \log k/\varepsilon)$ adaptive queries and time $\widetilde{O}(nk/\varepsilon) + \mathrm{poly}(k!/\varepsilon)$.*

Proof. Immediate from Theorem 3.5.6 on setting $C_{[k]} = \mathrm{Isom}(f) \cap \mathrm{Jun}_{[k]}$. □

3.5.8. Corollary. *Let $\varepsilon > 0$. There exists a non-adaptive ε-tester with one-sided error for function isomorphism in the unknown-unknown setting with query complexity*

$$\widetilde{O}\left(\frac{2^{k/2}}{\varepsilon^{\frac{1}{2}}}\right),$$

under the promise that both functions are k-juntas. It runs in time $n \cdot \mathrm{poly}(k!/\varepsilon)$.

Proof. Combine Theorem 2.5.1 with Theorem 3.5.2. □

Note that the tester of Corollary 3.5.7 is adaptive, in contrast with the tester of Theorem 2.3.1. (The tester of Corollary 3.5.8 can easily be made non-adaptive by switching to the non-adaptive junta tester, as the complexity of the preprocessing stage is subsumed by the number of samples taken.) We discuss in Chapter 5 how to obtain non-adaptive isomorphism testers.

3.6 Junta testing

As one may expect, the testers for the *property* of being a k-junta play an important role in constructing sample extractors. For this reason we need an overview of the internal workings of junta testers; see also the survey [Bla10]. The reader looking for full proofs should consult the original papers.

For simplicity we do not describe the testers in their full generality. Most of the time we will be dealing with boolean functions on $\{0,1\}^n$ and this is the setting we work with in this chapter, although the need will occasionally arise to consider arbitrary product domain distributions in subsequent chapters. We consider two different junta testers. The first one appeared in the original paper by Fischer, Kindler, Ron, Safra and Samorodnitsky [FKR$^+$04] and makes $O(k^4 \log(k+1)/\varepsilon)$ *non-adaptive* queries. The second one, due to Blais [Bla09], makes $O(k/\varepsilon + k \log k)$ *adaptive* queries.[2] The latter almost matches the $\Omega(k)$ lower bound of Chockler and Gutfreund [CG04]. There is also an improved non-adaptive tester [Bla08] with query complexity $\widetilde{O}(k^{3/2}/\varepsilon)$, which shall not be discussed here.

An essential ingredient of all junta testers is the construction of random partitions of the input variables, as in Definition 3.4.1. Let f be the function being tested, and if f is a k-junta, let $J \subseteq [n]$ denote its set of relevant variables. Roughly speaking, the junta testers take an isolating partition for J and then proceed to identify the blocks in $\mathcal{I}$ containing the most influential variables of g; it is important to note, however, that there is no guarantee that all influential variables will be detected. One way to decide if a block is relevant is to perform an influence test on it, with a low enough threshold η. The non-adaptive junta tester proceeds block by block in this fashion.

Of course, this only tells half of the story. When f is not a k-junta, there is no k-set J of relevant variables to start with. We make repeated use of the following lemma:

3.6.1. Lemma (Fischer et al. [FKR$^+$04]). *Let $f\colon \{0,1\}^n \to \{0,1\}, A \subseteq [n]$. Then*

(a) The closest A-junta to f is given by

$$J^*(x) = \mathrm{Maj}\,\{f(y) \mid y\restriction_A = x\restriction_A\}$$

(where ties can be arbitrarily broken).

(b)

$$\mathrm{dist}(f, \mathrm{Jun}_A) \leq \mathrm{Inf}_f([n] \setminus A) \leq 2 \cdot \mathrm{dist}(f, \mathrm{Jun}_A).$$

Proof.

[2] Interestingly, in the quantum world, the $\log k$ factor in the upper bound can be disposed of, as shown by Atıcı and Servedio [AS07].

(a) Clearly $\jmath^*$ is a junta on A by construction. Let $\jmath$ be an arbitrary junta on A. For every $x_2 \in \{0,1\}^A$, consider the nonnegative quantity

$$q(x_2) \triangleq \Pr_{x_1 \in \{0,1\}^{[n]\setminus A}} [f(x_1 \sqcup x_2) \neq \jmath(x_1 \sqcup x_2)].$$

As $\jmath(x_1 \sqcup x_2)$ does not depend on x_1, choosing $\jmath^*$ for $\jmath$ minimizes $q(x_2)$ over all A-juntas, for *any* x_2. Hence it also minimizes

$$\operatorname{dist}(f, \jmath) = \mathop{\mathbb{E}}_{x_2} [q(x_2)].$$

(b) Let $\jmath = \jmath^*$. We have, for every $x_2 \in \{0,1\}^A$,

$$\Pr_{y_1, x_1 \in \{0,1\}^{[n]\setminus A}} [f(x_1 \sqcup x_2) \neq f(y_1 \sqcup x_2)] = 2 \cdot q(x_2)(1 - q(x_2)) \in [q(x_2), 2q(x_2)]$$

because $q(x_2) \leq 1/2$ by our choice of $\jmath$. Since $\operatorname{dist}(f, \jmath) = \mathbb{E}_{x_2 \in \{0,1\}^A} [q(x_2)]$, by taking expectations we conclude that

$$\operatorname{Inf}_f([n] \setminus A) \in [\operatorname{dist}(f, \jmath), 2 \cdot \operatorname{dist}(f, \jmath)].$$

□

Consequently, the task can be reduced to accepting k-juntas and rejecting functions where the influence outside any set of size k is at least ε.

3.6.1 Non-adaptive junta tester

The non-adaptive tester uses its random coin flips to select a series of disjoint subsets $I_1, \ldots, I_r \subseteq [n]$ (where $r = O(k^2)$), and performs an independence test on each of them to some prescribed accuracy threshold η. These subsets are then shown to satisfy the following property with high probability:

- if $\operatorname{Inf}_f([n] \setminus A) \geq \varepsilon$ for all $A \subseteq [n], |A| = k$, then at least $k + 1$ of the independence tests will be positive.

This is enough to prove soundness (the rejection condition when f is ε-far from a k-junta), by the reasons explained right after Lemma 3.6.1. As for completeness, when f is a k-junta, at most k of the independence tests will be positive because $I_1, \ldots, I_r$ are disjoint.

We will not give a detailed account of how these tests are selected, but will be content to note that the proof in [FKR+04] uses the following results about how influence distributes over random subsets. We will make use of some of them later.

3.6.2. Definition. Let $f\colon \{0,1\}^n \to \{0,1\}$. The *unique influence* of the coordinate i on f with respect to the set $A \subseteq [n]$ is defined by

$$I_i^A \triangleq \operatorname{Inf}_f(A \cap [i]) - \operatorname{Inf}_f(A \cap [i-1]).$$

3.6.3. Lemma (Fischer et al. [FKR+04]). *Let $B \subseteq A \subseteq [n]$. Then*

(a) For all $i \in [n]$, we have $I_i^A \leq \operatorname{Inf}_i(f)$ (in particular $I_i^A = 0$ when $i \notin A$).

(b) For all $i \in B$, we have $I_i^A \leq I_i^B$.

(c) $\operatorname{Inf}_f(A) = \sum_{i\in A} I_i^A$ and $\operatorname{Inf}_f(B) = \sum_{i\in B} I_i^B \geq \sum_{i\in B} I_i^A$.

Proof.

(a) This is a special case of (b) when $B = \{i\}$.

(b) This is the submodularity of influence (Lemma 3.2.5).

(c) The first part holds because we can express the influence of A as a telescoping sum of unique influences:

$$\operatorname{Inf}_f(A) = \operatorname{Inf}_f(A \cap [n]) - \operatorname{Inf}_f(\emptyset) = \sum_{i\in[n]} I_i^A = \sum_{i\in A} I_i^A.$$

This also shows the equality in the second part; the inequality follows from item (b). □

The next lemma was key to the proofs of the non-adaptive junta tester. While we will only need the claim about the expected influence, we also include a concentration result.

3.6.4. Lemma (Fischer et al. [FKR+04]). *Let $f\colon \{0,1\}^n \to \{0,1\}$, $\tau > 0$ and let $A \subseteq [n]$ be a subset of the variables with individual influence $< \tau$ on f. Let $\rho \in [0,1]$ and $B \subseteq_\rho A$ denote a subset of A obtained by putting each element of A in B with probability ρ. Then*

$$\mathop{\mathbb{E}}_{B\subseteq_\rho A}[\operatorname{Inf}_f(B)] \geq \rho \operatorname{Inf}_f(A)$$

and

$$\Pr_{B\subseteq_\rho A}\left[\operatorname{Inf}_f(B) \leq \frac{\rho}{2}\operatorname{Inf}_f(A)\right] \leq e^{-\frac{\rho}{8\tau}\operatorname{Inf}_f(A)}.$$

Proof. When $B \subseteq_\rho A$, the quantity $\mathrm{Inf}_f(B)$ is bounded from below by a sum of independent random variables in the interval $[0, \tau)$, the ith of which is the product of the indicator function $\mathbb{I}[i \in B]$ and the unique influence I_i^A. To make the best use of concentration inequalities, we rescale them so that they assume values in $[0, 1)$. That is, note that $\tau > 0$ and for each $i \in A$ define

$$X_i \triangleq \frac{1}{\tau}\mathbb{I}[i \in B] \cdot I_i,$$

where $\mathbb{E}[\mathbb{I}[i \in B]] = \Pr[i \in B] = \rho$ for $i \in A$. Then we have by Lemma 3.6.3

$$\mathrm{Inf}_f(B) \geq \tau \sum_{i \in A} X_i,$$

where $X_i \in [0, 1), \mathbb{E}[X_i] = \frac{\rho}{\tau} I_i$, and

$$\mathbb{E}\left[\sum_{i \in A} X_i\right] = \frac{\rho}{\tau} \cdot \sum_{i \in A} I_i^A = \frac{\rho}{\tau} \mathrm{Inf}_f(A).$$

By Chernoff bounds, the probability that $\sum_{i \in A} X_i$ is at most half its expected value is bounded by $e^{-\frac{\rho}{8\tau}\mathrm{Inf}_f(A)}$, concluding the proof. □

From this one can show that if we take a random subset B of density $1/t$ of a set A and the influence of B decreases by a greater proportion than t, then most of the influence of A comes from a subset of size t.

3.6.5. Lemma. *Let $0 < t \leq n \in \mathbb{N}$, $f\colon \{0,1\}^n \to \{0,1\}$, $A \subseteq [n]$. Then either*

$$\Pr_{B \subseteq_{1/t} A}\left[\mathrm{Inf}_f(B) \geq \frac{\mathrm{Inf}(A)}{16t}\right] \geq 1 - \frac{1}{e}$$

or

$$\textit{there is } J \in \tbinom{A}{\leq t} \textit{ with } \mathrm{Inf}_f(J) \geq \tfrac{\mathrm{Inf}_f(A)}{2}.$$

Proof. Sort the elements of A by decreasing value of I_i^A. Without loss of generality, $A = \{1, 2, \ldots, m\}$ with $I_1^A \geq I_2^A \geq \ldots I_m^A$. By Lemma 3.6.3, $\mathrm{Inf}_f(A) = \sum_{i \in [m]} I_i^A$.

If $t \geq m$ or $\sum_{i=1}^t I_i^A \geq \mathrm{Inf}_f(A)/2$, then we can take $J = \{1, \ldots, t\}$. Otherwise we have

$$\mathrm{Inf}_f(A \setminus [t]) > \sum_{i > t} I_i^A \geq \mathrm{Inf}_f(A)/2, \tag{3.5}$$

and there are two cases to consider:

Case 1: $I_t^A \geq \frac{\mathrm{Inf}_f(A)}{16t}$. Then the probability that $B \subseteq_{1/t} A$ fails to intersect $\{1, \ldots, t\}$ is $(1-1/t)^t \leq \frac{1}{e}$. So with probability at least $1 - 1/e$, there is $i \in [t]$ with $i \in B$, hence

$$\mathrm{Inf}_f(B) \geq I_i^A \geq I_t^A \geq \frac{\mathrm{Inf}_f(A)}{16t}.$$

Case 2: $I_t^A < \frac{\mathrm{Inf}_f(A)}{16t}$. Any random subset $B \subseteq_{1/t} A$ contains a random subset $B' \subseteq_{1/t} A'$, where $A' \triangleq A \setminus [t]$. By (3.5), we can apply Lemma 3.6.4 for A' and $\rho = 1/t$, $\tau = \mathrm{Inf}_f(A')/(8t) \geq \rho\,\mathrm{Inf}_f(A)/(16t)$. We get

$$\Pr_{B' \subseteq_{1/t} A'}\left[\mathrm{Inf}_f(B') \leq \frac{\mathrm{Inf}_f(A')}{8t}\right] \leq e^{-1},$$

so

$$\Pr_{B \subseteq_{1/t} A}\left[\mathrm{Inf}_f(B) \leq \frac{\mathrm{Inf}_f(A)}{16t}\right] \leq e^{-1}.$$

□

3.6.2 Adaptive junta tester

The adaptive test takes a different route and we shall need a more detailed understanding of it. Given a random partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$, the tester starts with an empty set $\mathcal{J}' = \emptyset$, and gradually keeps adding to it the blocks I_i that have been found to contain a relevant variable, as follows. For each of $O(k/\varepsilon)$ rounds, it generates two random strings $x, y \in \{0,1\}^n$ and queries f on x and on

$$z \triangleq x_{\bar{S}} y_S = x_{S \leftarrow y\restriction_S},$$

where $S \triangleq [n] \setminus \bigcup_{I_i \in \mathcal{J}'} I_i$. (Picking x and y is the only place where randomness is used.) If $f(x)$ turns out to be different from $f(x_{\bar{S}} y_S)$, we know that there is at least one relevant block in $\mathcal{I} \setminus \mathcal{J}'$ yet to be found. In this case, we can find a relevant block by performing a binary search on the $|\mathcal{I} \setminus \mathcal{J}'| + 1$ *hybrid* strings between x and $x_{\bar{S}} y_S$ obtained in the following way:

Let $I_{i_1}, \ldots, I_{i_t}$ be the blocks in $\mathcal{I} \setminus \mathcal{J}'$, where $1 \leq i_1 \leq \cdots \leq i_t \leq \ell$. The jth hybrid z_j has the same values as y on all indices in $I_{i_1} \cup \cdots \cup I_{i_j}$, and its values elsewhere are the same as those of x. In particular, $z_0 = x$ and $z_t = z$. If we know $a < b$ with $f(z_a) \neq f(z_b)$, then for any $a \leq m \leq b$ at least one of $f(z_a) \neq f(z_m)$ or $f(z_m) \neq f(z_b)$ must hold, so if $f(z_0) \neq f(z_t)$ we can use binary search to find a relevant block I_{i_j} after making at most $\log(t+1) \leq \log(\ell+1)$ queries to f on the hybrid strings.

If at some stage the tester discovers more than k relevant blocks then it rejects; otherwise it accepts and outputs a (possibly extended) set $\mathcal{J} \supseteq \mathcal{J}'$ of size k (see Algorithm 5).

Algorithm 5 $\mathrm{T}^\star(k, \varepsilon, \mathcal{I})$: adaptive junta tester

$\mathcal{J}' \leftarrow \emptyset$
for $i = 1$ **to** $\lceil 40(k+1)/\varepsilon \rceil$ **do**
    $S \leftarrow [n] \setminus \bigcup_{I_i \in \mathcal{J}'} I_i$
    pick $x, y \in \{0,1\}^n$ uniformly at random
    if $f(x) \neq f(y_S x_{\bar{S}})$ **then**
        use binary search to find a block I_j containing a relevant variable
        $\mathcal{J}' \leftarrow \mathcal{J}' \cup \{I_j\}$
        if $|\mathcal{J}'| > k$, **reject** f
extend (if needed) $\mathcal{J}'$ to a set $\mathcal{J}$ of size k, by adding to it $k - |\mathcal{J}'|$ arbitrary blocks from $\mathcal{I} \setminus \mathcal{J}'$
accept f and return $\mathcal{J}$

Note that the algorithm needs time $O(n) + \mathrm{poly}(k/\varepsilon)$ to construct the partition and $O(nk/\varepsilon)$ to select and query all pairs x, y. To perform each of the $O(k)$ binary searches, the time required is only $O(n + n/2 + n/4 \dots) = O(n)$ (assuming we do not have to write the whole n-bit input on a new location for each oracle call to f).

3.6.6. Remark. There are a few minor differences between the original algorithm of [Bla09] and the one presented here:

- The constant factors have been modified for convenience.
- The original algorithm constructs the random partition $\mathcal{I}$ by itself; here we treat $\mathcal{I}$ as an argument passed to the algorithm (for convenience).
- The original algorithm does not actually output the set $\mathcal{J}$; instead, it identifies a set $\mathcal{J}'$ of at most k blocks containing relevant variables. Here $\mathrm{T}^\star$ always returns a set $\mathcal{J}$ of size *exactly* k, by extending (if necessary) the set $\mathcal{J}'$ arbitrarily; as we show later, precisely how this extension is performed is inconsequential.

Again, the tester's completeness is easy to show. Its soundness is proved through the next lemma.[3]

3.6.7. Lemma (Blais [Bla09, Main Lemma]). *Let $\mathcal{I} = (I_1, \dots, I_\ell)$ denote a random partition of $[n]$ into $\ell = 10^{20} k^9/\varepsilon^5$ parts. With probability at least $5/6$, a function $f\colon \{0,1\}^n \to \{0,1\}$ that is ε-far from being a k-junta also satisfies*

$$\mathrm{Inf}_f(A) \geq \varepsilon/2$$

for all *A formed by taking the union of k parts in $\mathcal{I}$.*

[3] Very recently, a new proof of a similar lemma has been found [BWY11] that shows that in fact we can take $\ell = O(k^2)$.

On a different note, observe that the lemma gives a *tolerant* junta tester with complexity $\exp(\mathrm{poly}(k/\varepsilon))$: simply go through all $\binom{\ell}{k}$ possible k-subsets of $\mathcal{I}$ and estimate the minimum influence among all of them.

3.6.3 Summary of junta testing

We summarize the junta testing results below. They are stated in greater generality than discussed so far. Briefly speaking, it is possible to tackle real-valued functions (as opposed to boolean functions) by replacing the notion of influence of a set A with that of *variation*, which is the variance of the function when the values inside A are random, averaged over all values outside A. This is the approach taken in [FKR$^+$04]. Alternatively, one can leave the definition of influence unaltered, and then the relationship between the influence of a set on f and the Fourier expansion of f is maintained if we replace the latter with the more general *Efron-Stein decomposition*, as shown in [Bla09].

Of greater interest to us is the fact that the testers work for an arbitrary product measure. Let $\Omega_1, \ldots, \Omega_n$ be finite sets and $\mu_1, \ldots, \mu_n$ distributions with μ_i supported on Ω_i. The distance between two functions $f, g : \Omega_1 \ldots \Omega_n \to \mathbb{R}$ is now defined by the measure under $\mu = \mu_1 \times \mu_2 \cdots \times \mu_n$ of the set $\{x \mid f(x) \neq g(x)\}$; this gives rise to a notion of distance from being a k-junta. (A k-junta from $\Omega_1 \ldots \Omega_n$ to $\mathbb{R}$ is still a function that depends on at most k variables.)

3.6.8. Theorem. *Let $\{\Omega_i\}_{i\in[n]}, \{\mu_i\}_{i\in[n]}, f\colon \prod_{i\in[n]} \Omega_i \to \mathbb{R}$ as before. Let $\varepsilon > 0$.*

For the property of being a k-junta, there exist the following ε-testers with constant success probability:

1. *A non-adaptive tester with query complexity $O(k^4 \log(k+1))$ [FKR$^+$04].*
2. *An adaptive tester with query complexity $O(k/\varepsilon + k \log k)$ [Bla09].*
3. *A tolerant, non-adaptive tester with query complexity $2^{\mathrm{poly}(k/\varepsilon)}$.*

All these testers have one-sided error (meaning they always accept k-juntas).

3.7 Construction of noisy sample extractors

Observe that Theorem 3.5.2 is stated for functions that are merely approximated by juntas. Although isomorphism testers do not need this, this will be of paramount importance for the applications discussed in Chapter 6. Unfortunately, the junta testers are not guaranteed to accept functions that are, say, $\varepsilon/10$ close to juntas, i.e., they are not tolerant. In fact, coming up with a tolerant junta tester with polynomial query complexity is listed as an open problem in [Bla10]. We observe, however, that the junta testers enjoy a certain (weak) form of *tolerance*; roughly

speaking, $\theta_{3.5.2}(k,\varepsilon)$ is a measure of the amount of tolerance of the adaptive tester, i.e., how close f must be to a k-junta in order to make sure it will be accepted with high probability. (This is Lemma 3.7.3 in Section 3.7.1.) We use the adaptive tester because of its smaller query complexity, but it is to be noted that this is not essential and our sampler construction can be analyzed in the same way if we use the non-adaptive tester instead. This would lead to a slower preprocessing stage but a better approximation threshold.

It may be useful to keep in mind that our final construction of the sample extractor will begin by calling the junta tester with parameter k. Let f be $\theta_{3.5.2}(k,\varepsilon)$-close to some k-junta f^*. The aforementioned tolerance implies that f is not rejected. (Note, however, that f may be $\theta_{3.5.2}(k,\varepsilon)$-far from any k-junta and still be accepted with high probability, as long as it is ε-close to some k-junta.) The tester also returns a set of k blocks that isolate the relevant variables of a junta h sufficiently close to f. (Clearly h must itself be close to f^* too.)

3.7.1 Smoothness and tolerance

Consider a property $\mathcal{P}$ of boolean functions on $\{0,1\}^n$ and an ε-tester $\mathcal{T}$ for $\mathcal{P}$ that makes q queries and has success probability $1-\delta$. Let r denote a random seed (so that we can view the tester as a deterministic algorithm with an additional input r) and let $Q(f,r) \subseteq \{0,1\}^n$ be the set of queries it makes on input f and seed r. Define $Q(r) \triangleq \bigcup_f Q(f,r)$; this is the set of all possible queries T may make as f runs through all possible functions, once r is fixed. We call $p \triangleq \max_r |Q(r)|$ the *non-adaptive complexity* of the tester. If $q = p$ then the tester is essentially non-adaptive; and clearly $p \leq 2^q$ holds for any tester of boolean function properties. We observe that for the adaptive junta tester, p is in fact polynomially bounded in q. (Without loss of generality we assume that $Q(r)$ is never empty.)

3.7.1. Definition. A tester is *p-smooth* if its non-adaptive complexity is at most p and for all $\alpha \in \{0,1\}^n$,

$$\Pr_{\substack{r \\ y \in Q(r)}} [y = \alpha] = \frac{1}{2^n}.$$

Notice that y is picked uniformly at random from the set $Q(r)$, regardless of the probability y would be queried by the tester $\mathcal{T}$ on any particular f. In other words, we are picking one random query of the non-adaptive version of $\mathcal{T}$ that queries all of $Q(r)$ in bulk, and requiring that the resulting string be uniformly distributed.

3.7.2. Lemma. *Let $\mathcal{T}$ be a p-smooth tester for $\mathcal{P}$ that accepts every $f \in \mathcal{P}$ with probability at least $1-\delta$. Then for every $f\colon \{0,1\}^n \to \{0,1\}$,*

$$\Pr[\mathcal{T} \text{ accepts } f] \geq 1 - \delta - p \cdot \operatorname{dist}(f, \mathcal{P}).$$

Proof. Choose any $f' \in \mathcal{P}$ and let $\Delta \triangleq \{y \in \{0,1\}^n \mid f(y) \neq f'(y)\}$. By the union bound, the probability (over r) that $Q(r)$ intersects Δ is at most $\mu \triangleq p \cdot \mathrm{dist}(f, f')$, and hence the probability is at least $1-\mu$ that the tester reaches the same decision about f as it does about f'. But the probability that f' is rejected is at most δ, hence the claim. □

3.7.3. Lemma. *The one-sided error junta tester* $\mathrm{T}^\star$ *of Algorithm 5 is* $\theta_{3.7.3}(k,\varepsilon)$*-smooth, where* $\theta_{3.7.3}(k,\varepsilon) \triangleq (10^{25}k^{10})/\varepsilon^6$*. Thus, by Lemma 3.7.2, it accepts functions that are* $\theta_{3.5.6}(k,\varepsilon)$*-close to* Jun_k *with probability at least* $9/10$ *(since* $10 \cdot \theta_{3.5.6}(k,\varepsilon) \leq 1/\theta_{3.7.3}(k,\varepsilon)$*). (It also rejects functions that are* ε*-far from* Jun_k *with probability at least* $2/3$*.)*

Proof. Note that once the randomness has been fixed, the number of possible queries that can be made in any given round is $|\mathcal{I} \setminus \mathcal{J}'| + 1 \leq \ell + 1$, so $|Q(r)| \leq 40\frac{k+1}{\varepsilon}(\ell+1)$ (recall that ℓ is the number of blocks in partition $\mathcal{I}$). Also, any hybrid z_j of two uniformly random strings $x, y \in \{0,1\}^n$ is itself uniformly random. These two things together mean that the tester is $40\frac{k+1}{\varepsilon}(\ell+1)$-smooth, and we can plug in the value of $\ell = O(k^9/\varepsilon^5)$ from Lemma 3.6.7. □

3.7.2 Extracting samples

Intuitively, the key idea to obtain the sample extractor is that the samples required may be obtained by making queries to f on certain blockwise-constant strings, so that we know the values that y sets on the (unknown) relevant variables of h (which is sufficiently close to both f and f^*). Although such strings are far from being uniformly distributed for a fixed partition, the approach can be shown to work most of the time if we settle for a small fraction of the samples being incorrectly labelled.

3.7.4. Definition. Given an $\mathcal{I}$-blockwise constant $y \in \{0,1\}^n$ and an ordered subset $\mathcal{J} = (J_1, \ldots, J_k)$ of $\mathcal{I}$, define $\textsc{extract}_{\mathcal{I},\mathcal{J}}(y)$ to be the string $x \in \{0,1\}^k$ where for every $i \in [k]$, $x_i = y_j$ if $j \in J_i$; and x_i is a uniformly random bit if $J_i = \emptyset$.

3.7.5. Definition. Let $\mathcal{I}$ denote a random partition of even size ℓ. For any $\mathcal{J} \subseteq \mathcal{I}$, we define a pair of distributions:

- The distribution $\mathcal{D}_{\mathcal{I}}$ on $\{0,1\}^n$: To obtain a random $y \sim \mathcal{D}_{\mathcal{I}}$,
  1. Pick $z \in \{0,1\}^\ell$ uniformly at random among all binary vectors of weight precisely $\ell/2$.
  2. Set $y \leftarrow \textsc{replicate}_{\mathcal{I}}(z)$.

- The distribution $\mathcal{D}_\mathcal{J}$ on $\{0,1\}^{|\mathcal{J}|}$: To obtain a random $x \sim \mathcal{D}_\mathcal{J}$,
  1. Draw $y \in \{0,1\}^n$ at random from $\mathcal{D}_\mathcal{I}$;
  2. Set $x \leftarrow \textsc{extract}_{\mathcal{I},\mathcal{J}}(y)$.

3.7.6. Lemma (Properties of $\mathcal{D}_\mathcal{I}$ and $\mathcal{D}_\mathcal{J}$).

(a) For all $\alpha \in \{0,1\}^n$, $\Pr_{\mathcal{I},y\sim\mathcal{D}_\mathcal{I}}[y = \alpha] = 1/2^n$;

(b) Assume $\ell > 4|\mathcal{J}|^2$. For every $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$, the L^∞ distance between $D_\mathcal{J}$ and the uniform distribution on $\{0,1\}^{|\mathcal{J}|}$ is at most $4|\mathcal{J}|^2/(\ell 2^{|\mathcal{J}|})$. Hence the statistical distance between $\mathcal{D}_\mathcal{J}$ and uniform is bounded by $2|\mathcal{J}|^2/\ell$.

Proof.

(a) Each choice of $z \in \{0,1\}^\ell$, $|z| = \ell/2$, in Definition 3.7.5 splits $\mathcal{I}$ into two equally-sized sets $\mathcal{I}^0$ and $\mathcal{I}^1$; $\mathcal{I}^b$ ($b \in \{0,1\}$) contains the blocks I_i that satisfy $z_i = b$. The bits corresponding to indices in $\mathcal{I}^b$ are set to b in the construction of y. For each index $i \in [n]$, the block it is assigned to is chosen independently at random from $\mathcal{I}$, and therefore falls within $\mathcal{I}^0$ (or $\mathcal{I}^1$) with probability $1/2$, independently of other $j \in [n]$. (This actually shows that the first item of the lemma still holds if z is an arbitrarily fixed string of weight $\ell/2$, rather than a randomly chosen one.)

(b) Let $k = |\mathcal{J}|$. Let us prove the claim on the L^∞ distance, which implies the other one. We may assume that all sets J_i in $\mathcal{J}$ are non-empty; having empty sets can only decrease the distance to uniform. Let $w \in \{0,1\}^k$. The choice of $y \sim \mathcal{D}_\mathcal{I}$, in the process of obtaining $x \sim \mathcal{D}_\mathcal{J}$, is independent of $\mathcal{J}$; thus, for every $i \in [k]$ we have

$$\Pr_{x\sim\mathcal{D}_\mathcal{J}}[x_i = w_i \mid x_j = w_j\ \forall j < i] \le \frac{\ell/2}{\ell - k} < \frac{1}{2} + \frac{k}{\ell},$$

and

$$\Pr_{x\sim\mathcal{D}_\mathcal{J}}[x_i = w_i \mid x_j = w_j\ \forall j < i] \ge \frac{\ell/2 - k}{\ell - k} > \frac{1}{2} - \frac{k}{\ell}.$$

Using the inequalities $1 - my \le (1-y)^m$ for all $y \le 1, m \in \mathbb{N}$ and $(1+y)^m \le e^{my} \le 1 + 2my$ for all $m \in [0, 1/(2y)]$, we conclude

$$\Pr_{x\sim\mathcal{D}_\mathcal{J}}[x = w] = \left(\frac{1}{2} \pm \frac{k}{\ell}\right)^k = \frac{1}{2^k}\left(1 \pm \frac{4k^2}{\ell}\right).$$

whereas a truly uniform distribution U should satisfy $\Pr_{x\sim U}[x = w] = 1/2^k$.

□

3.7.7. Definition. Given $\mathcal{I}, \mathcal{J}$ as above and oracle access to $f\colon \{0,1\}^n \to \{0,1\}$, we define a probabilistic algorithm $\textsc{sampler}_{\mathcal{I},\mathcal{J}}(f)$ that on each execution produces a pair $(x,a) \in \{0,1\}^{|\mathcal{J}|} \times \{0,1\}$ as follows: it first picks a random $y \sim \mathcal{D}_{\mathcal{I}}$, then queries f on y, and finally outputs the pair $(\textsc{extract}_{\mathcal{I},\mathcal{J}}(y), f(y))$.

Jumping ahead, we remark that the pair $\mathcal{I}, \mathcal{J}$ will be the information encoded in state α referred to in Lemma 3.5.2. In order to ensure that the last condition there is satisfied, we need to impose certain conditions on $\mathcal{I}$ and $\mathcal{J}$.

3.7.8. Definition. Given $\delta > 0$, a function $f\colon \{0,1\}^n \to \{0,1\}$, a partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ and a k-subset $\mathcal{J}$ of $\mathcal{I}$, we call the pair $(\mathcal{I}, \mathcal{J})$ *δ-good* (with respect to f) if there exists a k-junta $h\colon \{0,1\}^n \to \{0,1\}$ such that the following conditions are satisfied:

1. Conditions on h:
   (a) Every relevant variable of h is also relevant for f^* (recall that f^* denotes the k-junta closest to f);
   (b) $\operatorname{dist}(f^*, h) < \delta$.
2. Conditions on $\mathcal{I}$:
   (a) For all $j \in [\ell]$, I_j contains at most one variable of $\operatorname{core}_k(f^*)$;[4]
   (b) $\Pr_{y \sim \mathcal{D}_{\mathcal{I}}}[f(y) \neq f^*(y)] \leq 10 \cdot \operatorname{dist}(f, f^*)$;
3. Conditions on $\mathcal{J}$:
   (a) The set $\bigcup_{I_j \in \mathcal{J}} I_j$ contains *all* relevant variables of h;

3.7.9. Lemma. *Let $\delta, f, \mathcal{I}, \mathcal{J}$ be as in the preceding definition. If the pair $(\mathcal{I}, \mathcal{J})$ is δ-good (with respect to f), then $\textsc{sampler}_{\mathcal{I},\mathcal{J}}(f)$ is an (η, μ)-noisy sampler for some permutation of $\operatorname{core}_k(f^*)$, with $\eta \leq 4\delta + 4k^2/\ell + 10 \cdot \operatorname{dist}(f, f^*)$ and $\mu \leq 4k^2/\ell$.*

Proof. By item 2b in Definition 3.7.8, it suffices to prove that

$$\Pr_{y \sim \mathcal{D}_{\mathcal{I}}}[f^*(y) \neq \operatorname{core}_k(f^*)^{\pi}(\textsc{extract}_{\mathcal{I},\mathcal{J}}(y))] < 2\delta + 4k^2/\ell$$

for some π.

Let h be the k-junta witnessing the fact that the pair $(\mathcal{I}, \mathcal{J})$ is δ-good. Let $V \subseteq [n]$ be the set of k variables of $\operatorname{core}_k(f^*)$; recall that V may actually be a superset of the relevant variables of f^*. Let $\mathcal{J}' \triangleq \{I_j \in \mathcal{I}\colon I_j \cap V \neq \emptyset\}$ be an ordered subset respecting the order of $\mathcal{J}$, and let π be the permutation whose

[4]Note that this, together with 1a, implies that every block I_j contains at most one relevant variable of h, since the variables of $\operatorname{core}_k(f^*)$ contain all relevant variables of f^*.

inverse maps the i-th relevant variable of f^* (in the standard order) to the index of the element of $\mathcal{J}'$ in which it is contained. We assume without loss of generality that π is the identity map.

It follows from Definition 3.7.8 that $|\mathcal{J}'| = |V| = k$, since each block in $\mathcal{I}$ contains at most one variable of $\mathrm{core}_k(f^*)$. For any $\mathcal{I}$-blockwise $y \in \{0,1\}^n$, let $x \triangleq \textsc{extract}_{I,\mathcal{J}}(y)$ and $x' \triangleq \textsc{extract}_{I,\mathcal{J}'}(y)$ denote the k-bit strings corresponding to $\mathcal{J}$ and $\mathcal{J}'$. We have the equalities

$$f^*(y) = \mathrm{core}_k(f^*)(x') \qquad \text{by Definition 3.7.4,} \tag{3.6}$$

$$\mathrm{core}_k(h)(x) = \mathrm{core}_k(h)(x') \qquad \text{by Definition 3.7.8.} \tag{3.7}$$

From item 1b of Definition 3.7.8 we also have

$$\Pr_{r\in\{0,1\}^k}[\mathrm{core}_k(f^*)(r) \neq \mathrm{core}_k(h)(r)] < 2\delta, \tag{3.8}$$

where r is picked uniformly at random. However, by the second item of Lemma 3.7.6, the distribution $\mathcal{D}_{\mathcal{J}}$ is $2k^2/\ell$ close to uniform; combining this with (3.8) we also get

$$\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[\mathrm{core}_k(f^*)(x) \neq \mathrm{core}_k(h)(x)] < 2\delta + 2k^2/\ell. \tag{3.9}$$

Likewise, we have

$$\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[\mathrm{core}_k(f^*)(x') \neq \mathrm{core}_k(h)(x')] < 2\delta + 2k^2/\ell, \tag{3.10}$$

thus, using equations (3.7), (3.9), (3.10) and the union bound we get

$$\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[\mathrm{core}_k(f^*)(x') \neq \mathrm{core}_k(f^*)(x)] < 4\delta + 4k^2/\ell. \tag{3.11}$$

Combining (3.6) and (3.11) we conclude that

$$\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[f^*(y) \neq \mathrm{core}_k(f^*)(x)] < 4\delta + 4k^2/\ell,$$

and the claim follows. □

As the lemma suggests, our next goal is to obtain a good pair $(\mathcal{I}, \mathcal{J})$. For this we need to prove that the adaptive junta tester satisfies certain properties.

3.7.3 Obtaining a good pair $(\mathcal{I}, \mathcal{J})$

In the following proposition we claim that the tester $\mathrm{T}^\star$ satisfies several conditions that we need for obtaining the aforementioned sampler.

3.7.10. Proposition.
There is a tester $\mathrm{T}^\star$ *for* Jun_k *with query complexity* $O(k \log k + k/\varepsilon)$ *that takes a (random) partition* $\mathcal{I} = I_1, \ldots, I_\ell$ *of* $[n]$ *as input, where* $\ell = \Theta(k^9/\varepsilon^5)$ *is even, and outputs (in case of acceptance) a* k*-subset* $\mathcal{J}$ *of* $\mathcal{I}$ *such that for any* f *the following conditions hold (the probabilities below are taken over the randomness of the tester and the construction of* $\mathcal{I}$*):*

- *if* f *is* $\theta_{3.5.6}(k, \varepsilon/2400)$ *close to* Jun_k, $\mathrm{T}^\star$ *accepts with probability at least* $9/10$*;*
- *if* f *is* $\varepsilon/2400$*-far from* Jun_k, $\mathrm{T}^\star$ *rejects with probability at least* $9/10$*;*
- *for* any f*, with probability at least* $4/5$ *either* $\mathrm{T}^\star$ *rejects, or it outputs* $\mathcal{J}$ *such that the pair* $(\mathcal{I}, \mathcal{J})$ *is* $\varepsilon/600$*-good (as per Definition 3.7.8).*

In particular, if $\mathrm{dist}(f, \mathrm{Jun}_k) \le \theta_{3.5.6}(k, \varepsilon)$*, then with probability at least* $4/5$ $\mathrm{T}^\star$ *outputs a set* $\mathcal{J}$ *such that* $(\mathcal{I}, \mathcal{J})$ *is* $\varepsilon/600$*-good.*

Proof. By Lemma 3.7.3, the first two conditions are satisfied by the junta tester, called with a value of $\varepsilon' = \varepsilon/2400$.

Let $\mathcal{J}' = (I_{s_1}, \ldots, I_{s_{|\mathcal{J}'|}})$ be the set output by the original algorithm $\mathrm{T}^\star$ and let $S = \{s_1, \ldots, s_{|\mathcal{J}'|}\}$. Closer inspection of Algorithm 5 shows that, with probability at least $19/20$,

$(*)$ either f is rejected or the set S satisfies

$$\mathrm{Inf}_f\left([n] \setminus \left(\bigcup_{j \in S} I_j\right)\right) \le \varepsilon/4800.$$

This is because the main loop of algorithm runs for $40(k+1)/\varepsilon'$ rounds. Suppose that at any of these, the influence of the remaining blocks is always $\ge \varepsilon'/2$. Since the expected number of rounds to find $k+1$ relevant blocks is at most $2(k+1)/\varepsilon'$ in this case, it follows that with probability $19/20$, a $(k+1)$-th relevant block is found and f is rejected.

Recall that when $|S| < k$ the set $\mathcal{J}'$ is extended by putting in it $k - |S|$ additional "dummy" blocks from $\mathcal{I} \setminus \mathcal{J}'$ (some of them possibly empty), obtaining a set $\mathcal{J}$ of size exactly k.

Now we go back to proving the third item. Let $R \in \binom{[n]}{\le k}$ denote the set of relevant variables of f^* (the closest k-junta to f), and let $V \in \binom{[n]}{k}$, $V \supseteq R$, denote the set of variables of $\mathrm{core}_k(f^*)$. Assume[5] that $\mathrm{dist}(f, \mathrm{Jun}_k) \le \varepsilon/2400$, and $\mathrm{T}^\star$ did not reject. In this case,

[5]For other f's the third item follows from the second item.

- by $(*)$, with probability at least 19/20 the set $\mathcal{J}$ satisfies
$$\mathrm{Inf}_f\left([n] \setminus \left(\bigcup_{I_j \in \mathcal{J}} I_j\right)\right) \leq \mathrm{Inf}_f\left([n] \setminus \left(\bigcup_{j \in S} I_j\right)\right) \leq \varepsilon/4800;$$
- since $\ell \gg k^2$, with probability larger than 19/20 all elements of V fall into different blocks of the partition $\mathcal{I}$;
- by Lemma 3.7.6, $\Pr_{\mathcal{I}, y \sim \mathcal{D}_{\mathcal{I}}}[f(y) = f^*(y)] = \mathrm{dist}(f, f^*)$; hence by Markov's inequality, with probability at least 9/10 the partition $\mathcal{I}$ satisfies
$$\Pr_{y \sim \mathcal{D}_{\mathcal{I}}}[f(y) \neq f^*(y)] \leq 10 \cdot \mathrm{dist}(f, f^*).$$

So with probability at least 4/5, all three of these events occur. Now we show that conditioned on them, the pair $(\mathcal{I}, \mathcal{J})$ is $\varepsilon/600$-good.

Let $U = R \cap (\bigcup_{I_j \in \mathcal{J}} I_j)$. Informally, U is the subset of the relevant variables of f^* that were successfully "discovered" by $\mathrm{T}^\star$. Since $\mathrm{dist}(f, f^*) \leq \varepsilon/2400$, we have $\mathrm{Inf}_f([n] \setminus V) \leq \varepsilon/1200$ (by Lemma 3.6.1). By the subadditivity and monotonicity of influence we get

$$\begin{aligned}
\mathrm{Inf}_f([n] \setminus U) &\leq \mathrm{Inf}_f([n] \setminus V) + \mathrm{Inf}_f(V \setminus U) \\
&\leq \mathrm{Inf}_f([n] \setminus V) + \mathrm{Inf}_f\left([n] \setminus \left(\bigcup_{I_j \in \mathcal{J}} I_j\right)\right) \\
&\leq \varepsilon/960,
\end{aligned}$$

where the second inequality follows from $V \setminus U \subseteq [n] \setminus (\bigcup_{I_j \in \mathcal{J}} I_j)$. This means, by Lemma 3.6.1, that there is a k-junta h in Jun_U satisfying $\mathrm{dist}(f, h) \leq \varepsilon/960$, and by the triangle inequality, $\mathrm{dist}(f^*, h) \leq \varepsilon/2400 + \varepsilon/960 < \varepsilon/600$. Based on this h, we can verify that the pair $(\mathcal{I}, \mathcal{J})$ is $\varepsilon/600$-good by going over the conditions in Definition 3.7.8. □

3.7.4 Flattening out the distribution

We would like to obtain a perfectly uniform distribution for the first component of the samples (to comply with our definition of sample extractors, although allowing small deviations from uniformity would not affect any of our applications). Using rejection sampling, one can easily obtain an exactly uniform sampler from a slightly non-uniform sampler at the expense of a small increase in the error probability:

3.7.11. Lemma. *Let $\widetilde{g}$ be an (η, μ)-noisy sampler for $g\colon \{0,1\}^k \to \{0,1\}$, which on each execution picks $x \in \{0,1\}^k$ according to some fixed distribution D. Then we can construct an $(\eta + \mu)$-noisy sampler $\widetilde{g}_{uniform}$ for g that makes one query to $\widetilde{g}$ for each sample (and no queries to g itself).*

Proof. Let U denote the uniform distribution on $\{0,1\}^k$. The new sampler $\widetilde{g}_{uniform}$ acts as follows: first it obtains a sample (x,a) from $\widetilde{g}$, and proceeds as follows:

(acceptance) with probability $p_x \triangleq \frac{\Pr_{y\sim U}[y=x]}{(1+\mu)\Pr_{z\sim D}[z=x]}$ it outputs (x,a);

(rejection) with probability $1-p_x$ it picks an uniformly random $z \in \{0,1\}^k$ and outputs $(z,0)$.

(Note that $p_x \leq 1$ by definition of (n,μ)-noisy sampler.)

Let (x',a') denote the pairs output by $\widetilde{g}_{uniform}$. We can compute the overall acceptance probability as

$$\mathop{\mathbb{E}}_{x\sim D}[p_x] = \sum_{x\in\{0,1\}^k} \Pr_{z\sim D}[z=x]\cdot p_x = 1/(1+\mu).$$

Also note that for any x,

$$\Pr_{x'}[x'=x \text{ and the sample was accepted}] = \Pr_{z\sim D}[z=x]\cdot p_x = \frac{\Pr_{y\in U}[y=x]}{1+\mu}.$$

Therefore, conditioned on acceptance (which, as we just saw, happens with probability $1/(1+\mu)$), x is uniformly distributed. In case of rejection (which occurs with probability $\mu/(1+\mu)$) it is uniform by definition; hence the overall distribution of x is uniform too. Recalling that $\Pr[a \neq g(x)] \leq \eta$, we conclude that $\Pr[a' \neq g(x')] \leq \eta + \mu/(1+\mu) \leq \eta + \mu$. □

We remark that the conversion made in Lemma 3.7.11 is only possible when the distribution D is known. This is the case for the sampler that we construct here nonetheless. We note that in this case we have $\Pr_{y\sim U}[y=x] = 2^{-k}$ and

$$\Pr_{z\sim\mathcal{D}_{\mathcal{J}}}[z=x] = \frac{\binom{\ell/2}{|x|}\binom{\ell/2}{k-|x|}}{\binom{\ell}{k}\binom{k}{|x|}}$$

(which only depends on $|x|$); the preprocessor A_P can precompute these $k+1$ numbers in time $\mathrm{poly}(\ell) = \mathrm{poly}(k/\varepsilon)$ by, e.g., using dynamic programming to compute the binomial coefficients.

3.7.5 Putting it all together

Proof of Theorem 3.5.2. We start by describing how A_P and A_S operate: The preprocessor A_P starts by constructing a random partition $\mathcal{I}$ and calling the junta tester $\mathrm{T}^\star$. Then, in case $\mathrm{T}^\star$ accepted, A_P encodes in the state α the partition $\mathcal{I}$ and the subset $\mathcal{J} \subseteq \mathcal{I}$ output by $\mathrm{T}^\star$ (see Proposition 3.7.10). (The

state α, which has size $O(n)$, can also encode $O(n)$ precomputed values such as those needed by the conversion to uniform.) The sampler A_S, given α, obtains a pair $(x, a) \in \{0,1\}^k \times \{0,1\}$ by executing $\textsc{sampler}_{\mathcal{I},\mathcal{J}}(f)$ (once).

Now we show how Lemma 3.5.2 follows from Proposition 3.7.10. The first two items are immediate. As for the third item, notice that we only have to analyze the case where $\mathrm{dist}(f, f^*) \leq \varepsilon/2400$ and $\mathrm{T}^\star$ accepted; all other cases are taken care of by the first two items. By the third item in Proposition 3.7.10, with probability at least 4/5 the pair $(\mathcal{I}, \mathcal{J})$ is $\varepsilon/600$-good. If so, by Lemma 3.7.9 $\textsc{sampler}_{\mathcal{I},\mathcal{J}}(f)$ is an (η, μ)-noisy sampler for some permutation of $\mathrm{core}_k(f^*)$, with $\eta \leq \varepsilon/150 + 4k^2/\ell + 10 \cdot \mathrm{dist}(f, f^*) \leq \varepsilon/92 + 4k^2/\ell$ and $\mu \leq 4k^2/\ell$. The final step we apply is the conversion from Lemma 3.7.11, with which we obtain a $(\varepsilon/92 + 4k^2/\ell + 4k^2/\ell) \leq (\varepsilon/90)$-noisy sampler for some permutation of $\mathrm{core}_k(f^*)$.

Finally, apply Lemma 3.7.11 to turn it into a perfectly uniform sampler. □

3.8 Final remarks

We mention here one interesting avenue for further research. We have shown how to obtain random samples of the core of a k-junta, but there can be k-bit inputs on which the samples are always wrong. In line with the classical works on self-correction, one can ask whether it is possible to efficiently *self-correct* juntas. That is, given oracle access to f which is promised to be δ-close to a k-junta g, we would like an algorithm that outputs $g(x)$ with high probability for *any* x. (One could also ask the same question about the $\mathrm{core}_k(g)$ instead.) The problem is well defined if $\delta < 2^{-k-1}$.

This problem is solvable with $\exp(O(k))$ queries in several ways for $\delta < c2^{-k}$ and some constant c. For example, we could identify all relevant blocks and make blockwise-constant queries, much in the same way as in the exact randomized algorithm of Section 3.4.1. For small enough c, any assignment to the core of g gives rise to a subfunction that agrees on a noticeable majority with the same subfunction of f. Another solution would be to use the more general self-corrector for low degree polynomials of Alon et al. [AKK$^+$03], since k-juntas are in particular degree-k polynomials.

In the worst case, this bound cannot be improved upon, as noted by Alon and Weinstein [AW12]. They show that if g is an AND of k randomly chosen literals (positive or negative), self-correcting (functions close to) g requires $\exp(\Omega(k))$ queries. However, some juntas can be easily self-corrected. For example, if $\mathcal{C}$ is a class of k-juntas such that for every $f \in \mathcal{C}$, the influence of each relevant variable in f is greater than $1/\mathrm{poly}(k)$, then it is possible to self-correct $\mathcal{C}$ with $\mathrm{poly}(k)$ queries. It would be interesting to be able to pinpoint which functions or k-juntas can be efficiently self-corrected.

3.9 Summary

We developed a query-efficient algorithm that extracts labelled samples from the core of the closest k-junta to f. After a preprocessing step, which takes $\widetilde{O}(k)$ queries, generating each sample takes only one query to f. Using this sampler, we derived algorithms to test isomorphism to k-juntas, or to approximate the distance between two k-juntas up to isomorphism in various settings. Aside from being as efficient as possible in terms of query complexity, these algorithms also run in polynomial time for $k = O(\log n / \log \log n)$, and some of them even for $k = O(\log n)$.

Chapter 4

Junto-symmetric functions and hypergraph isomorphism

Now we touch upon the question of when it is possible to test isomorphism with constantly many queries. We prove a characterization of the class of hypergraphs of constant arity (rank) to which isomorphism can be efficiently tested, and make a step towards obtaining a similar characterization for general hypergraphs and boolean functions.

The content of this chapter is based on the paper

- S. Chakraborty, E. Fischer, D. García–Soriano, and A. Matsliah. Junto-symmetric functions, hypergraph isomorphism, and crunching. To appear in *Proceedings of the 27th IEEE Conference on Computational Complexity (CCC)*, 2012.

4.1 The size of invariance groups

The *automorphism group* of a function f, also known as its *symmetry group* or *invariance group*, is the group of permutations that leave f invariant:

$$\mathrm{Aut}(f) \triangleq \{\pi \in S_n \mid f^\pi = f\}.$$

Clearly $\mathrm{Aut}(f)$ is a subgroup of the symmetric group $S_n = Sym([n])$. Define an equivalence relation between permutations by $\pi \sim \sigma$ iff $f^\pi = f^\sigma$, and let

$$\mathrm{DifPerm}(f) = \{[\pi_1], \ldots, [\pi_t]\}$$

be the equivalence classes formed. There is a bijection between $\mathrm{DifPerm}(f)$ and the set $S_n : \mathrm{Aut}(f)$ of cosets of $\mathrm{Aut}(f)$; therefore the number $|\mathrm{DifPerm}(f)| = |\mathrm{Isom}(f)|$ of distinct permutations of f is equal to the index of $\mathrm{Aut}(f)$ in S_n, i.e., $|\mathrm{DifPerm}(f)| = |S_n : \mathrm{Aut}(f)| = n!/\,|\mathrm{Aut}(f)|$. The size of $\mathrm{Aut}(f)$ is a rough measure of the amount of symmetry that f possesses: the larger $\mathrm{Aut}(f)$, the more

symmetric f is. A symmetric function satisfies $\mathrm{Aut}(f) = S_n$ and $|\mathrm{DifPerm}(f)| = 1$, whereas a random function has, with high probability, a trivial automorphism group $\mathrm{Aut}(f) = \{1\}$ and $|\mathrm{DifPerm}(f)| = n!$ (for example, see [Cla92] for a simple proof of a stronger statement).

Not every group $G \leq S_n$ can arise as the automorphism group of a boolean function on n variables; those which can are called *2-representable*. For example, it is not hard to argue that if the alternating group A_n ($n \geq 3$) is contained in $\mathrm{Aut}(f)$, then $\mathrm{Aut}(f)$ is indeed the whole of S_n; as a result, A_n is not 2-representable. Indeed, take any $x, y \in \{0,1\}^n$ with $|x| = |y|$. Then there is a permutation $\pi \in S_n$ mapping x to y; if $n \geq 3$ then π can be assumed to be an even permutation by performing, if necessary, one additional swap between two distinct indices i, j with $y_i = y_j$. Then $\pi \in A_n \subseteq \mathrm{Aut}(f)$ and so $f(x) = f(y)$. Hence $A_n \leq \mathrm{Aut}(f)$ implies $f(x) = f(y)$ for all $|x| = |y|$, so f is actually symmetric.

The groups $G \leq S_n$ that can be represented as $\mathrm{Aut}(f)$ for some k-valued function $f\colon \{0,1\}^n \to [k]$ are called k-representable. The representability of k-valued functions and some generalizations are studied in [CK91, Kis98, Xia05] (see also Chapter 3 of [CK02]). A neat paper of Babai, Beals and Takácsi-Nagy [BBTN92] exposes a relationship between the circuit complexity of a function f and the number of orbits of the action of $\mathrm{Aut}(f)$ on $\{0,1\}^n$.

We know that f-isomorphism can always be tested with $O(\log |\mathrm{DifPerm}(f)|)$ queries for constant ε (Proposition 2.3.2), so symmetric functions are particularly easy to test isomorphism to (the query complexity becomes constant; in fact the problem reduces to testing equality in this case). What is the smallest size that $\mathrm{DifPerm}(f)$ can have for a non-symmetric function f? A moment's thought reveals that there are non-symmetric functions with only n different permutations, like any dictatorship $f(x_1 x_2 \dots x_n) = x_i$, and indeed this can be shown to be best possible.[1]

4.1.1. Proposition. *If $f\colon \{0,1\}^n \to \{0,1\}$ is not symmetric and $n \geq 5$, then $|\mathrm{DifPerm}(f)| \geq n$.*

Proof. The elements of S_n act on $\mathrm{DifPerm}(f)$ by multiplication in a natural way: for each $\pi \in S_n$ we define a permutation $\phi(\pi)$ of $\mathrm{DifPerm}(f)$ by

$$\phi(\pi)([\sigma]) = [\pi \circ \sigma]$$

The map $\phi\colon S_n \to Sym(\mathrm{DifPerm}(f))$ is well-defined since $[\sigma_1] = [\sigma_2]$ implies $f^{\sigma_1} = f^{\sigma_2}$ and hence $f^{\pi\circ\sigma_1} = f^{\pi\circ\sigma_2}$, so $[\pi \circ \sigma_1] = [\pi \circ \sigma_2]$. Moreover, it is a group homomorphism (where the product operation on both S_n and $Sym(\mathrm{DifPerm}(f))$ is the usual composition of permutations); this is because $\phi(1) = 1$, and $\phi(\pi_1) \circ \phi(\pi_2) = \phi(\pi_1 \circ \pi_2)$. Therefore its kernel $\ker \phi$ is a normal subgroup of S_n. The

[1] The claim fails for $n = 4$: the function $f(a,b,c,d) = (a \wedge b) \vee (c \wedge d)$ has three different permutations.

only normal subgroups of S_n ($n \geq 5$) are $1, A_n$ and S_n [Art10, Theorem 7.4.4]. Clearly $\ker \phi \leq \mathrm{Aut}(f)$ and since $\mathrm{Aut}(f)$ does not contain A_n (or else f would be symmetric as argued before), it follows that $\ker \phi = 1$, so ϕ is injective. Since the domain of ϕ is S_n, its image $Sym(Isom(f))$ must be at least as large, hence $|\mathrm{DifPerm}(f)| \geq n$. □

Even though the number of queries made by the trivial isomorphism tester is superconstant for a non-symmetric function, it is also possible to test isomorphism to dictatorships with $O(1)$ queries [PRS02], and more generally to $O(1)$-juntas [FKR$^+$04]. However, these two classes do not encompass all known easy-to-test functions. For example, consider the parity function on the first $n - t$ variables out of n, $\chi_{[n-t]}$.[2] The identity $\chi_{[n-t]}(x) = \chi_{[n]}(x) \oplus \chi_{[n]\setminus[n-t]}(x)$ makes it possible to transform the responses to all queries made for the t-junta $\chi_{[n]\setminus[n-t]}$ into the responses to queries for $\chi_{[n-t]}$. This transformation provides a reduction between the two testing problems. In particular, for constant t we can test isomorphism to $(n-t)$-parities with $O_t(1)$ queries. In the same vein, the majority function on the first $n - t$ variables $\mathrm{Maj}_{[n-t]}$ (for n large enough and $t \ll \sqrt{n}$) is very close to the symmetric majority $\mathrm{Maj}_{[n]}$, and it is not hard to see that the standard constant-query test for equality between the tested function and $\mathrm{Maj}_{[n]}$ yields a tester for isomorphism to $\mathrm{Maj}_{[n-t]}$ as well (because its queries are uniformly distributed).

We introduce a notion generalizing all these cases.

4.1.2. Definition. Let $J \subseteq [n]$. A function $f\colon \{0,1\}^n \to \{0,1\}$ is called *J-junto-symmetric* if it can be written in the form

$$f(x) = \tilde{f}\left(|x|, x\restriction_J\right) \tag{4.1}$$

for some $\tilde{f}\colon \{0,\dots,n\} \times \{0,1\}^{|J|} \to \{0,1\}$. Equivalently, this means that the restriction of f to any constant-weight layer of the cube is a junta on J.

The function f is called *k-junto-symmetric* if it is J-junto-symmetric on some subset J of size k.

The function $\tilde{f}$ above is not completely determined by f on inputs of very small or high weight. For example, let f be 1-junto-symmetric. Then one can define $\tilde{f}(0,1)$ in two different ways that give rise to the same function f.

Let $\mathcal{JS}_J$ denote the class of J-junto-symmetric functions, and $\mathcal{JS}_k$ the k-junto-symmetric functions. Note that the definition necessitates that the junta variables be the same on every layer, but the junta function is allowed to vary. Also variables outside J can have noticeable influence on a J-junto-symmetric function f.

[2]The symbol χ is usually reserved to a parity taking values in ± 1 so it is a character of $\mathbb{Z}_2^n$, but here we use it for $\{0,1\}$-valued functions.

Observe that any symmetric function is 0-junto-symmetric, and any k-junta is k-junto-symmetric. At the other extreme, every function is $(n-1)$-junto-symmetric. Additional examples of k-junto-symmetric functions are $\chi_{[n-k]}$ and $\mathrm{Maj}_{[n-k]}$; in fact, the reader may verify that any k-junta whose core function is symmetric must be $\min(k, n-k)$-junto-symmetric.

4.1.3. Definition. Let $\mathcal{F}$ denote a sequence $f_1, f_2, \ldots$ of boolean functions with $f_n \colon \{0,1\}^n \to \{0,1\}$ for each $n \in \mathbb{N}^+$.

We say that $\mathcal{F}$ is an *$O(1)$-junto-symmetric family* if there exists a constant k such that each f_i is k-junto-symmetric.

Interestingly, $O(1)$-junto-symmetric functions were studied by Shannon under the name "partially symmetric functions" [Sha49].

The size of $\mathrm{DifPerm}(f)$ for any k-junto-symmetric f is upper-bounded by $\binom{n}{k}k!$, because if f can be written in the form (4.1), then for any $\pi \in S_n$ there is a k-subset $T \subseteq [n]$ and a permutation $\sigma \in Sym(T) \cong S_k$ such that $f^\pi(x) = \tilde{f}(|x|, (x\restriction_T)^\sigma)$. This quantity is $n^{O(1)}$ for constant k. Families like this were given a name in [PS10]:

4.1.4. Definition. The family $\mathcal{F}$ is *poly-symmetric* if there exists a constant c such that $|\mathrm{DifPerm}(f_n)| \leq n^c$ for all n.

We will occasionally speak of such a family as an $O(1)$-junto-symmetric (or poly-symmetric function when the intended meaning is clear. As it turns out, the two notions just described are the same.

4.2 Characterizing $O(1)$-junto-symmetry

In the following we identify elements of $Sym(T)$, $T \subseteq [n]$, with elements of $Sym([n])$ that act as the identity outside T.

4.2.1. Theorem. *Let $\mathcal{F} = \{f_n \colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$. The following are equivalent:*

(a) $\mathcal{F}$ is a poly-symmetric family;

(b) There are sets $A_n \subseteq [n]$ of constant size such that $Sym([n] \setminus A_n) \leq \mathrm{Aut}(f_n)$ for all n;

(c) $\mathcal{F}$ is an $O(1)$-junto-symmetric family;

(d) Each f_n is a boolean combination of $O(1)$-many dictators and $O(1)$-many symmetric functions (with the same constants for all n).

To ease readability, we drop the subscripts in the proof, i.e., write f and A in place of f_n and A_n. All but one of the implications we need are straightforward:

- $(b) \implies (c)$: $Sym([n] \setminus A) \leq \mathrm{Aut}(f)$ means that f is invariant under permutations of $[n] \setminus A$, i.e., $f(x) = f(y)$ whenever $x\restriction_A = y\restriction_A$ and $|x\restriction_{[n]\setminus A}| = |y\restriction_{[n]\setminus A}|$. These conditions are equivalent to $x\restriction_A = y\restriction_A$ and $|x| = |y|$, so f has the form $f(x) = \tilde{f}(|x|, x\restriction_A)$ (where $|A| = O(1)$ by assumption).

- $(c) \implies (d)$: Let $f = \tilde{f}(|x|, x\restriction_A)$, $|A| = k = O(1)$. Define $\tilde{f}^{(i)}(x) = \tilde{f}(i, x\restriction_A)$. Each $\tilde{f}^{(i)}$ is a junta on A. The number of A-juntas is only $\ell = 2^{2^k} = O(1)$; let $\jmath_1, \ldots, \jmath_\ell$ be an enumeration of them and let

$$h_i(x) \triangleq \begin{cases} 1 & \text{if } \tilde{f}^{(|x|)} = \jmath_i \\ 0 & \text{otherwise} \end{cases}.$$

  Each h_i is a symmetric function, and f can be decomposed into

$$f(x) = \bigvee_{i \in [\ell]} h_i(x) \wedge \jmath_i(x),$$

  which is a boolean combination of ℓ symmetric functions and the $\{j_i\}$ functions, which are themselves a combination of the k dictators $\{x_i\}_{i \in A}$.

- $(d) \implies (b)$: Let $f(x) = \tilde{f}(s_1(x), \ldots, s_\ell(x), x_{i_1}, \ldots, x_{i_k})$, where $s_1, \ldots, s_\ell$ are symmetric. Set $A = \{i_1, \ldots, i_k\}$ and let $\pi \in Sym([n] \setminus A)$. Each function s_i remains invariant under $Sym([n])$, and each dictatorship x_{i_j} is invariant under $Sym([n] \setminus \{i\}) \supseteq Sym([n] \setminus A)$. Therefore $Sym([n] \setminus A) \leq \mathrm{Aut}(f)$.[3]

- $(c) \implies (a)$: As we just saw, if $f \in \mathcal{JS}_k$ and $k = O(1)$, then $|\mathrm{DifPerm}(f)| \leq \binom{n}{k} k! = n^{O(1)}$.

The only remaining implication, which will be shown next[4], is $(a) \implies (b)$.

[3] Note that the fact that $\ell = O(1)$ is immaterial here, and in fact yet another equivalent definition can be given by substituting "any number of symmetric functions" for "$O(1)$-many symmetric functions".

[4] This would be implied by the claim following Theorem 28 on page 586 of [CK91], but unfortunately this claim is in error (as can be seen by taking G_n to be the alternating group A_n). The mistake seems to lie near the end of the proof, after it is shown that $i_n \leq k$ and $|S_n : G_n| \leq n^k$, the claim that $V_n = S_{n-i_n}$ is unjustified. However, the lemma does hold for the automorphism groups of boolean functions however as we show. This is the case of interest in their paper and in this thesis.

4.2.1 Permutation groups

We need some basic notions from the theory of permutation groups (an exposition can be found in the books by Wielandt [Wie64] and Cameron[Cam99]). Let Ω be a set (which will be assumed finite here, and will often be equal to $[n]$ in our applications). $Sym(\Omega)$ denotes the symmetric group of all permutations of Ω, and $Alt(\Omega)$ is the subgroup of $Sym(\Omega)$ made up of even permutations. When $|\Omega| = n$, we occasionally write $A_n = Alt(\Omega)$ and $S_n = Sym(\Omega)$. The product operation we use in $Sym(\Omega)$ is $\pi\sigma \triangleq \sigma \circ \pi$.

A *permutation group* G on Ω is a subgroup of $Sym(\Omega)$, written $G \leq Sym(\Omega)$. The image $\pi(x)$ of $x \in \Omega$ under $\pi \in G$ is often written x^π; under our convention we have $(x^\pi)^\sigma = x^{\sigma\circ\pi} = x^{\pi\sigma}$ for $\pi, \sigma \in G$. The *orbit* of a set $\Delta \subseteq \Omega$ under an arbitrary collection $H \subseteq G$ is the set $\Delta^H = \{x^\pi \mid \pi \in H, x \in \Delta\}$. When $\Delta = \{x\}$ or $H = \{h\}$ are singletons we may simply write x^H or Δ^h.

G is called *transitive* if for every $x, y \in \Omega$ there is $\pi \in G$ with $x^\pi = y$. An intransitive group $G \leq Sym(\Omega)$ partitions Ω into orbits: these can be characterized as the equivalence classes of the relation $\sim$ given by $x \sim y$ iff there is $\pi \in G$ such that $x^\pi = y$, which occurs iff $x^G = y^G$.

A *group action* of a (general) group G on a set Ω is a homomorphism $\phi\colon G \to Sym(\Omega)$. (This is what is called a *right* action because of our convention on the composition law in $Sym(\Omega)$.) If $\ker\phi = 1_G$, the action is *faithful* and G is isomorphic (via ϕ) to a permutation group on Ω. It is customary to omit the explicit reference to the chosen ϕ and write x^g for $x^{\phi(g)}$ ($g \in G$). Given an action of G on Ω, we can naturally extend it to define an action on subsets of Ω: $g \in G$ acts on $\mathcal{P}(\Omega)$ by mapping $\Delta \subseteq \Omega$ to Δ^g as defined above.

A *block* of G is a subset Δ of Ω such that for every $\pi \in G$, either $\Delta^\pi = \Delta$ or $\Delta^\pi \cap \Delta = \emptyset$. Evidently, Ω, the empty set $\emptyset$ and each of the singletons $\{i\}_{i\in\Omega}$ are always blocks; we call these the *trivial blocks*. The permutation group G is said to be *primitive (group)* if it is transitive and has no non-trivial blocks. (Only transitive groups are classified as being primitive or imprimitive.) The intersection of any pair of blocks is itself a block. If Δ is a block of G, then Ω can be partitioned into a *complete block system*, where every block is of the form Δ^g for some $g \in G$ (so all blocks in a complete block system have the same cardinality). Any element of G permutes the blocks in a complete block system among themselves, and also the elements inside each block.

The *pointwise stabilizer* of $\Delta \subseteq \Omega$ with regard to G is the set

$$G_\Delta \triangleq \{\pi \in G \mid x^\pi = x \ \forall x \in \Delta\}.$$

4.2.2 Proof that poly-symmetric $\equiv O(1)-$junto-symmetric

First we need a handy result that provides a lower bound for the index of primitive groups. The proof can be found in [Wie64, Theorem 14.2] (asymptotically better bounds are available [Bab81, Cam81], but this one will suffice).

4.2.2. Theorem (Bochert's bound [Boc89]). *Let G be a primitive subgroup of S_n, other than S_n and A_n. Then*

$$[S_n : G] \geq \lfloor n/2 \rfloor !.$$

4.2.3. Lemma. *Let $n \geq 14$, $G \leq S_n$, $G \neq S_n, A_n$. Then*

(a) If G is transitive then

$$[S_n : G] \geq \frac{1}{2}\binom{n}{\lfloor n/2 \rfloor}.$$

(b) Suppose G is intransitive; let Δ be the longest orbit of an element of $[n]$ and $\ell = |\Delta| < n$ its size. Then

$$[S_n : G] \geq \binom{n}{\max(n/2, \ell)}.$$

(c) Under the same conditions as in (b), let

$$H \triangleq G \cap Sym(\Delta) = G \cap S_\ell$$

be the pointwise stabilizer of $[n] \setminus \Delta$ (we identify $Sym(\Delta)$ with S_ℓ). Then

$$[S_\ell : H] \leq \frac{[S_n : G]}{\binom{n}{\ell}}.$$

Proof.

(a) If G is primitive, Bochert's theorem states the bound $[S_n : G] \geq \lfloor n/2 \rfloor !$, which is stronger for $n \geq 14$. So suppose G is transitive and imprimitive, with a block of imprimitivity of size a ($2 \leq a \leq n/2, a \mid n$), and hence $b = n/a \geq 2$ such blocks because of transitivity (see Section 4.2.1). Then

$$|G| \leq (a!)^b b! \leq 2\lfloor (ab/2) \rfloor ! \lceil (ab/2) \rceil ! = 2\lfloor n/2 \rfloor \lceil n/2 \rceil,$$

The first inequality holds because there are $b!$ ways of permuting the blocks among themselves, and $a!$ ways of permuting the elements inside a given block. To prove the last inequality, observe that for $a = 2$ it reduces to the triviality $b! \geq 2^{b-1}$. Hence it suffices to verify that for any $b \geq 2$, the quotient

$$q(a) \triangleq \frac{a!^b}{\lfloor (ab/2) \rfloor ! \lceil (ab/2) \rceil !}$$

is a decreasing function of a. Writing the factors in the numerator and denominator in decreasing order, we have

$$q(a) = \frac{\overbrace{a \cdot a \cdot \ldots \cdot a}^{b} \overbrace{(a-1) \cdot (a-1) \cdot \ldots \cdot (a-1)}^{b} \ldots}{\lceil ab/2 \rceil \lfloor ab/2 \rfloor \lceil ab/2 - 1 \rceil \lfloor ab/2 - 1 \rfloor \ldots}$$

Define the sequences $\{s_i\}, \{t_i\}, i \in [1, ab]$ by $s_i = \lfloor (i+b-1)/b \rfloor$ and $t_i = \lfloor (i+1)/2 \rfloor$. Then

$$\begin{aligned} q(a) &= \prod_{i=1}^{ab} \frac{s_i}{t_i} = \prod_{i=1}^{(a-1)b} \frac{s_i}{t_i} \cdot \prod_{j=(a-1)b+1}^{ab} \frac{s_j}{t_j} \\ &= q(a-1) \cdot \prod_{j=(a-1)b+1}^{ab} \frac{a}{t_j} \\ &\leq q(a-1), \end{aligned}$$

because $t_{(a-1)b+1} = \lfloor (a-1)b/2 \rfloor + 1 \geq a$ since $b \geq 2$. Therefore

$$[S_n : G] = \frac{n!}{|G|} \geq \frac{1}{2} \binom{n}{\lfloor n/2 \rfloor}.$$

(b) Let $A_1, \ldots, A_m$ ($m \geq 2$) be the orbits and $a_i = |A_i|$. Since G only maps elements of A_i to elements of the same A_i, we have $G \leq Sym(A_1) \times Sym(A_2) \times \cdots \times Sym(A_m)$ and therefore

$$|G| \leq \prod_{i \in [m]} a_i!.$$

Fix $n > \ell > 0$ and let us consider

$$r_n(\ell) \triangleq \max \Big\{ \prod_{i \in [m]} a_i! \mid m \geq 2, a_i \in \mathbb{N}, 0 \leq a_i \leq \ell, \sum_{i \in [m]} a_i = n \Big\}$$

Consider the expression inside the maximum in the definition of $r_n(\ell)$. Without loss of generality, we can take $m = n$. We claim that it attains its maximum for some solution with $a_i = \ell$ for at least one i. Take any optimal solution and sort the values in non-increasing order: $a_1 \geq a_2 \geq \cdots \geq a_t > a_{t+1} = \ldots a_m = 0$. If $a_1 = \ell$ we are done. Otherwise $a_1 < \ell$ and we must have $t > 1$ (with $a_t > 0$) since the total sum is at least ℓ. If we replace the pair (a_1, a_t) with $(a_1 + 1, a_t - 1)$ we obtain a feasible solution, and $\prod a_i!$ increases since $a_1! a_t! < (a_1+1)!(a_t-1)!$ (as $a_1 + 1 > a_t$). This is not possible for an optimal solution, so there is no such pair, meaning that $a_1 = \ell$.

Now observe that $\prod_{i \in [m]} a_i! \leq a_i!(n - a_i)!$ for any i. (For example, this can be seen by noting that the left-hand side is the size of the set of permutations $Sym(A_1) \times \cdots \times Sym(A_m)$, and this a subset of $Sym(A_i) \times Sym([n] \setminus A_i)$.) So using $a_i = \ell$ for some i we get

$$r_n(\ell) \leq \ell!(n-\ell)! = \frac{n!}{\binom{n}{\ell}}$$

for *any* ℓ. When ℓ is the size of the largest orbit, we have $|G| \leq r_n(\ell)$, and this shows that

$$[S_n : G] \geq \frac{n!}{r_n(\ell)} \geq \binom{n}{\ell},$$

On the other hand, $r_n(\ell)$ is by definition an increasing function of ℓ, so the inequality

$$[S_n : G] \geq \frac{n!}{r_n(\ell)} \geq \frac{n!}{r_n(n/2)} \geq \binom{n}{\lfloor n/2 \rfloor}$$

holds when the size of the longest orbit is $\ell \leq n/2$.

(c) Because $G \leq H \times Sym([n] \setminus \Delta)$, we can bound

$$|G| \leq |H||S_{n-\ell}| = |H|(n-\ell)!,$$

which yields

$$[S_\ell : H] = \frac{\ell!}{|H|} \leq \frac{\ell!(n-\ell)!}{|G|} = \frac{[S_n : G]}{\binom{n}{\ell}}.$$

□

4.2.4. Lemma. *Let $n \geq 14$, $t \leq n/2$, $[S_n : G] < \frac{1}{2}\binom{n}{t}$, and Δ, ℓ as before. Then $\ell > n - t$ and $Alt(\Delta) \leq G$.*

Proof. If the action of G is transitive on $[n]$ then $[S_n : G] \geq \frac{1}{2}\binom{n}{\lfloor n/2 \rfloor}$ by Lemma 4.2.3(a), which contradicts our assumptions. So G is not transitive and $\ell < n$. If $\ell \leq n/2$ we have, by Lemma 4.2.3(b), $[S_n : G] \geq \binom{n}{\lfloor n/2 \rfloor}$, which again is impossible.

We are left with the case $n/2 < \ell < n$. In accordance with Lemma 4.2.3(b),

$$\binom{n}{t} > [S_n/G] \geq \binom{n}{\ell} = \binom{n}{n-\ell},$$

so $t > n - \ell$ (since $n - \ell, t \leq n/2$). Let $H = G \cap S_\Delta$. This is actually the pointwise stabilizer of $[n] \setminus \Delta$ in G, and since Δ is an orbit of G it follows that H is normal in G [Wie64, Proposition 3.1]. We demonstrate that $A_\Delta \leq H$ by contradiction.

So assume $H \neq Sym(\Delta), Alt(\Delta)$. Then Lemma 4.2.3 applies to the group H acting on Δ. Let Δ' be the largest orbit of this action and $\ell' = |\Delta'|$. Since G is transitive on Δ and $H \lhd G$, it is not hard to see that the length of any orbit of H on Δ must divide ℓ, i.e., $\ell' \mid \ell$. We distinguish two cases:

- If $\ell' \leq \ell/2$, then

$$[S_\ell : H] \geq \binom{\ell}{\ell/2}$$

by part (b) of the "inner" application of Lemma 4.2.3.

- If $\ell' > \ell/2$, then as we observed that $\ell' \mid \ell$, we must in fact have $\ell' = \ell$, meaning that H is transitive on Δ and

$$[S_\ell : H] \geq \frac{1}{2}\binom{\ell}{\ell/2}$$

by part (a) of the "inner" application of Lemma 4.2.3.

In any case we have

$$[S_\ell : H] \geq \frac{1}{2}\binom{\ell}{\ell/2}.$$

Together with part (c) of the "outer" application, i.e.,

$$[S_\ell : H] \leq \frac{[S_n : G]}{\binom{n}{\ell}},$$

this yields

$$[S_n : G] \geq \frac{1}{2}\binom{\ell}{\ell/2}\binom{n}{\ell}.$$

Now we bound each of these two factors. Using the inequality

$$\binom{2(m+1)}{m+1} = 2\binom{2m}{m}\frac{2m+1}{m+1} \leq 4\binom{2m}{m},$$

it is possible to show that

$$\binom{\ell}{\ell/2} \geq \frac{1}{2^{n-\ell}}\binom{n}{n/2}.$$

Using the fact that $\ell > n/2$, we get

$$\binom{n}{\ell} = \binom{n}{n-\ell} \geq \left(\frac{n}{n-\ell}\right)^{n-\ell} \geq 2^{n-\ell}.$$

Multiplying these two bounds we obtain the contradiction

$$[S_n : G] \geq \frac{1}{2}\binom{\ell}{\ell/2}\binom{n}{\ell} \geq \frac{1}{2}\binom{n}{n/2}.$$

□

4.2.5. Corollary. *Let $n \geq 14$ and $f\colon \{0,1\}^n \to \{0,1\}$ be a boolean function with $|\mathrm{DifPerm}(f)| < \frac{1}{2}\binom{n}{t}$, $t \leq n/2$. Then there is a set Γ of size $|\Gamma| < t$ such that f is junto-symmetric on Γ. In particular, any poly-symmetric family is junto-symmetric on sets of size $O(1)$.*

Proof. Let $G = \mathrm{Aut}(f)$. Since $|\mathrm{DifPerm}(f)| = [S_n : G]$, the previous lemma states that if Δ is the largest orbit, then $|\Delta| \geq n/2 \geq 5$ and $Alt(\Delta) \leq \mathrm{Aut}(f)$. We show that this means that f is junto-symmetric on $\Gamma \triangleq [n] \setminus \Delta$. Indeed, for any $x \in \{0,1\}^\Gamma$, we can define a boolean function $g_x : \{0,1\}^\Delta \to \{0,1\}$ by $g(z) = f(z \sqcup x)$; then $Alt(\Delta) \leq \mathrm{Aut}(f) \cap Sym(\Delta) \leq \mathrm{Aut}(g_x)$, so g_x is a boolean function on more than 4 variables whose automorphism group contains the alternating group. Hence g_x is actually symmetric for all x, and f is junto-symmetric on Γ. □

This corollary is the last piece we needed to show Theorem 4.2.1.

4.3 Testers for junto-symmetric functions

One of the main results of this chapter is an extension of the junta tester and the isomorphism tester for juntas:

4.3.1. Theorem ([CFGM12, BWY11]). *Let $\varepsilon > 0$ and $1/\varepsilon^{1/4} < k < (2n)^{1/12}$. Let $f\colon \{0,1\}^n \to \{0,1\}$ and denote $f^* \in \mathcal{JS}_k$ the k-junto-symmetric function closest to f.*

There is a $\mathrm{poly}(k/\varepsilon)$*-query algorithm that takes ε, k and an oracle for f and satisfies:*

completeness *If* $\mathrm{dist}(f, f^*) \leq 1/k^5$*, the algorithm accepts with probability* $\geq 2/3$.

soundness *If* $\mathrm{dist}(f, f^*) \geq \varepsilon$*, the algorithm rejects with probability* $\geq 2/3$.

See Section 4.3.3 for the proof.

We can also obtain an $O(1)$-query algorithm for testing isomorphism to $O(1)$-junto-symmetric functions.

4.3.2. Theorem. *[CFGM12, BWY11] Let k, ε, f as before. There is a* $\mathrm{poly}(k/\varepsilon)$*-query ε-tester for testing isomorphism between f and a known function $g\colon \{0,1\}^n \to \{0,1\}$ that is $1/k^5$-close to k-junto-symmetric, with constant success probability.*

The proof is in Section 4.3.4.

4.3.3. Corollary. *Isomorphism to any poly-symmetric function can be ε-tested with* $\mathrm{poly}(1/\varepsilon)$ *queries.*

With a view toward obtaining a possible classification, it is best to state *tolerant* versions of these results. This is possible at the expense of an exponential blowup in the query complexities (see Section 4.3.5).

4.3.4. THEOREM. *There is a constant $0 < c < 1$ with the following property. Let k, ε, f as before.*

There is an $\exp(k/\varepsilon)$-query algorithm that, with high probability accepts if f is $(c\varepsilon)$-close to $\mathcal{JS}_k$ and rejects if it is ε-far from $\mathcal{JS}_k$.

Similarly, there is an $\exp(k/\varepsilon)$-query algorithm to test isomorphism to a function f that is $(c\varepsilon)$-close to $\mathcal{JS}_k$.

In an independent work simultaneous with ours, Blais, Weinstein and Yoshida have also proven the results stated in this section [BWY11]. (Their query complexities are better and the restrictions on the size of k are not present.)

It is possible to define a notion of "symmetric influence" that characterizes closeness to junto-symmetric functions up to a factor of two, just as influence does for closeness to juntas. The resulting definition does not enjoy the subadditivity property, which is crucial for the proofs of the standard junta testers (Section 3.6). Although this approach can be made to work with some technical work [BWY11], here we take a different route.

We present a reduction from testing the properties of being k-junto-symmetric, or being isomorphic to a given k-junto-symmetric function, to slight generalizations of the well-studied analogous problems for k-juntas. To this end we try to approximate the "junto-symmetric" components of the tested function f, i.e., the juntas determining the behaviour of f on each constant-weight layer of the boolean cube. However, each of these juntas is defined on a very small fraction of inputs; in order to define them on the whole of $\{0,1\}^n$ we attempt use a small "ballast" set $B \subseteq [n]$ of variables to enable us to balance weights as needed.

4.3.1 Preliminary observations

Let $\ell \in \mathcal{L} \triangleq \{0, 1, \ldots, n\}$ and $x \in \{0,1\}^n$. Write x^B for the string obtained from x by flipping the bits in $B \subseteq [n]$ and consider the set of *minimal* changes required to turn x into a string of weight ℓ:

$$\mathcal{B}_{\ell,x} \triangleq \left\{ B \subseteq [n] \;\middle|\; \left|x^B\right| = \ell \text{ and } |B| = \left|\ell - |x|\right| \right\}.$$

For any $B \in \mathcal{B}_{\ell,x}$, either $x^B \subseteq x$ or $x \subseteq x^B$ holds, depending on whether $|x| \geq \ell$ or $|x| \leq \ell$. The set $\mathcal{B}_{\ell,x}$ is always non-empty but consists of the single element 0^n when $\ell = |x|$.

Let $\mathcal{R}$ denote the set of all possible functions $r \colon \mathcal{L} \times \{0,1\}^n \to \{0,1\}^n$ with $r(\ell, x) \in \mathcal{B}_{\ell,x}$ for all ℓ, x. We need a lemma concerning the probability that $B = r(\ell, x)$ happens to intersect some small set A, when (ℓ, r, x) are drawn from the product distribution $\mu \triangleq \mathcal{L} \times \mathcal{R} \times \{0,1\}^n$. Here $\mathcal{L}$ is endowed with a binomial distribution $B(n, 1/2)$ and the uniform distribution is used in $\mathcal{R}$ and $\{0,1\}^n$.

4.3.5. Lemma. *Let $A \subseteq [n]$. Then*

$$\Pr_{\ell,x,B}[B \cap A \neq \emptyset] \leq \frac{|A|}{\sqrt{2n}}.$$

Proof. Observe that for any ℓ, the distribution of $B = r(\ell, x) \in \mathcal{B}_{\ell,x}$ over random x is symmetric under permutations, hence for all $i \in [n]$ we have

$$\Pr[i \in B] = \frac{1}{n}\sum_{j\in[n]} \Pr[j \in B] = \frac{1}{n}\mathbb{E}[|B|].$$

On the other hand, the size of any element B of $\mathcal{B}_{\ell,x}$ is $|\ell - |x||$ by definition. We can write $\ell = |y|$ for uniformly random $y \in \{0,1\}^n$, so $\mathbb{E}[|B|] = \mathbb{E}[||x| - |y||]$. Recalling that $\mathbb{E}[|x|] = \mathbb{E}[|y|] = n/2$, $\mathbb{E}[|x|^2] = \mathbb{E}[|y|^2] = \operatorname{Var}[|x|] + \mathbb{E}[|x|]^2 = \frac{1}{4}n(n+1)$ and applying Cauchy-Schwarz,

$$\left(\mathbb{E}\left[\left||x| - |y|\right|\right]\right)^2 \leq \mathbb{E}\left[(|x| - |y|)^2\right] = \mathbb{E}\left[|x|^2\right] + \mathbb{E}\left[|y|^2\right] - 2\mathbb{E}[|x|]\,\mathbb{E}[|y|] = \frac{n}{2}.$$

Hence $\mathbb{E}\left[\left||x| - |y|\right|\right] \leq \sqrt{n/2}$ and $\Pr[i \in B] \leq \sqrt{\frac{1}{2n}}$, so

$$\Pr[B \cap A \neq \emptyset] \leq \sum_{i\in A} \Pr[i \in B] \leq \frac{|A|}{\sqrt{2n}}.$$

□

Let us define a transformation T mapping each function $f\colon \{0,1\}^n \to \{0,1\}$ to $T(f)\colon \mathcal{L} \times \mathcal{R} \times \{0,1\}^n \to \{0,1\}^n$ given by

$$T(f)(\ell, r, x) = f(x^{r(\ell,x)}).$$

Thus the parameter r acts as a "random seed" selecting, for each pair (ℓ, x), one string $x^{r(\ell,x)}$ of Hamming weight ℓ with minimum distance to x; the choice is independent of all choices for any other pair when r ranges uniformly over $\mathcal{R}$.

We want to argue about $T(f)$ as a function in its own right, on a larger set of variables. We denote the input parameter variables of $T(f)$ by V_0, V_1 and V_2, in order; we identify V_2 with $[n]$, the input variables of f. The reader who so wishes may think of $T(f)$ as a function on $\{0,1\}^{\lceil\log(n+1)\rceil} \times \{0,1\}^{\lceil\log|\mathcal{R}|\rceil} \times \{0,1\}^n$, although this is not strictly necessary; in this case V_0, V_1, V_2 would indicate disjoint input bit variables with sizes

$$|V_0| = \log(n+1), |V_1| = \log|\mathcal{R}|, |V_2| = n$$

(but note that the input distribution on $\{0,1\}^{V_0}$ is not uniform).

If $g\colon \mathcal{L} \times \mathcal{R} \times \{0,1\}^n$ is a junta on $V_0 \cup V_2$ (that is to say, $g(\ell, r, x)$ depends only on ℓ and x, but not on r), we define the function $\psi(g)\colon \{0,1\}^n \to \{0,1\}$ by

$$\psi(g)(x) = g(|x|, \bullet, x),$$

where the dot emphasizes that the assignment to the second parameter is immaterial by assumption, i.e., the variables in V_1 are irrelevant to $\psi(g)$. The intuition is that T maps junto-symmetric functions f on A into functions that are close to juntas on $(V_0 \cup A$ because V_1 and $V_2 \setminus A$ will be nearly irrelevant to $T(f))$; while ψ maps these functions on an extended domain that are juntas on $V_0 \cup A$ into junto-symmetric functions on A defined on $\{0,1\}^n$.

We show that the task of testing junto-symmetry of f is closely related to that of testing $T(f)$ for being a junta, where distances are measured under μ. Let $\mathrm{Jun}_{V_0}(A) = \mathrm{Jun}(V_0 \cup A)$, and $\mathrm{Jun}_k(V_0) = \cup_{|A|\le k} \mathrm{Jun}_{V_0}(A)$.

In the next lemma, the variable symbols denote functions and sets of the following kind:

- $A \subseteq [n]$, $|A| = k$;
- f, g are arbitrary functions $\{0,1\}^n \to \{0,1\}$;
- $\jmath, \jmath_1, \jmath_2\colon \{0,1\}^n \to \{0,1\}$ are junto-symmetric on A;
- $\jmath'\colon \mathcal{L} \times \mathcal{R} \times \{0,1\}^n \to \{0,1\}$ is a member of $\mathrm{Jun}_{V_0}(A)$;
- $\pi \in 1_{V_0,V_1} \times Sym(V_2)$ (we identify π with an element of $Sym(V_2)$ as well).

4.3.6. Lemma. *The mappings T and ψ satisfy the following properties:*

(a) T preserves distances: $\mathrm{dist}(f,g) = \mathrm{dist}(T(f), T(g))$ *for all f, g.*

(b) For any $\jmath' \in \mathrm{Jun}_{V_0}(A)$, we have $\psi(\jmath') \in \mathcal{JS}(A)$ and

$$\mathrm{dist}(\jmath', T(\psi(\jmath'))) \le \frac{|A|}{\sqrt{2n}}.$$

(c) For any $\jmath \in \mathcal{JS}(A)$, $T(\jmath)$ is $|A|/\sqrt{2n}$-close to some $\jmath' \in \mathrm{Jun}_{V_0}(A)$. Moreover, we can take $\jmath'$ such that $\psi(\jmath') = \jmath$.

(d) $|\,\mathrm{dist}(f, \mathcal{JS}_k) - \mathrm{dist}(T(f), \mathrm{Jun}_k(V_0))\,| \le \frac{k}{\sqrt{2n}}$.

(e) ψ preserves permutations: for any π and $\jmath'$, $\psi(\jmath')^\pi = \psi(\jmath^\pi)$. Thus

$$\mathrm{distiso}(\jmath_1, \jmath_2) = \mathrm{distiso}(\psi(\jmath_1), \psi(\jmath_2)).$$

(f) The bounds

$$|\operatorname{distiso}(f,g) - d| \le \operatorname{dist}(f, \mathcal{JS}_k) + \operatorname{dist}(g, \mathcal{JS}_k) + \frac{2k}{\sqrt{2n}}.$$

hold for

$$d \triangleq \min_{\pi \in 1_{V_0,V_1} \times Sym(V_2)} \operatorname{dist}(T(f)^\pi, T(g)).$$

Proof.

(a) For any ℓ, the distribution of $x^{r(\ell,x)}$ for random x, r is uniform over all strings of weight ℓ. Since $\ell \sim B(n, 1/2)$ is distributed as the weight of a random element of $\{0,1\}^n$, it follows that the overall distribution of $x^{r(\ell,x)}$ is uniform, hence

$$\operatorname{dist}(T(f), T(g)) = \Pr[f(x^{r(\ell,x)}) \neq g(x^{r(\ell,x)})] = \Pr[f(x) \neq g(x)] = \operatorname{dist}(f,g).$$

(b) $\psi(\jmath')(x) = \jmath'(|x|, \bullet, x)$ is a function of $|x|$ and x_A, hence junto-symmetric on A. We have

$$\begin{aligned}\operatorname{dist}(\jmath', T(\psi(\jmath'))) &= \Pr\left[\jmath'(\ell, r, x) \neq \psi(\jmath')(x^{r(\ell,x)}) = \jmath'(\ell, \bullet, x^{r(\ell,x)})\right] \\ &\le \Pr\left[r(\ell, x) \cap A \neq \emptyset\right] \\ &\le \frac{|A|}{\sqrt{2n}}\end{aligned}$$

by Lemma 4.3.5.

(c) This follows from (b) because any $\jmath \in \mathcal{JS}(A)$ can be written in the form $\psi(\jmath')$ for some (in fact, many) $\jmath' \in \operatorname{Jun}_{V_0}(A)$.

(d) Let $\jmath$ be k-junto-symmetric and $\jmath' \in \operatorname{Jun}_k(V_0)$ with $\psi(\jmath') = \jmath$. Then by the triangle inequality and parts (c) and (a),

$$\operatorname{dist}(T(f), \jmath') \le \operatorname{dist}(T(f), T(\jmath)) + \operatorname{dist}(T(\jmath), \jmath') \le \operatorname{dist}(f, \jmath) + \frac{k}{\sqrt{2n}},$$

so $\operatorname{dist}(T(f), \operatorname{Jun}_k(V_0)) \le \operatorname{dist}(f, \mathcal{JS}_k) + k/(2\sqrt{n})$. Likewise, if $\jmath'$ is a junta on $V_0 \cup A$ where $|A| = k$, then

$$\begin{aligned}\operatorname{dist}(f, \psi(\jmath')) &= \operatorname{dist}(T(f), T(\psi(\jmath'))) \\ &\le \operatorname{dist}(T(f), \jmath') + \operatorname{dist}(\jmath', T(\psi(\jmath'))) \\ &\le \operatorname{dist}(T(f), \jmath') + \frac{k}{\sqrt{2n}},\end{aligned}$$

which proves the inequality $\operatorname{dist}(f, \mathcal{JS}_k) \le \operatorname{dist}(T(f), \operatorname{Jun}_k(V_0)) + k/(2\sqrt{n})$.

(e) Clear.

(f) Follows from (d), (e) and the triangle inequality for distiso.

□

4.3.2 Generalized junta testing

Now we describe a tester for the property $\mathrm{Jun}_k(V_0)$. Let $\mu = D_1 \times \cdots \times D_m$ be a product distribution, let us also denote by μ its support. Let $T \subseteq [m]$. (For our application we could take $D_1 = \mathcal{L}, D_2 = \mathcal{R}$, $T = \{1, 2\}$ and $D_3 \times \cdots \times D_m = \{0,1\}^n$.) Choose a confidence parameter $p \in (0,1)$ and a distance parameter $\varepsilon \in (0,1)$. Let $f\colon \mu \to \mathbb{R}$ denote a function.

4.3.7. Lemma. *For any product distribution μ and any constant $p < 1$, there is an algorithm*

$$\textsc{GeneralizedJuntaTester}_{\mu,p}(f, k, \varepsilon, T)$$

that, with probability at least p,

- *accepts if $f \in \mathrm{Jun}_k(T)$.*
- *rejects if $\mathrm{dist}(f, \mathrm{Jun}_k(T)) \geq \varepsilon$;*
- *makes $\Theta(k^4 \log(k+1)/\varepsilon)$ non-adaptive queries, and the marginal distribution of each query is μ.*

Note that standard junta testing corresponds to $T = \emptyset$.

Proof. All known junta testers can be used in a straightforward manner for this generalized property preserving the exact query complexity. One way to see this is to think about providing the junta tester with a set T of relevant variables for free, and instruct it to seek for relevant blocks outside T just as if the tester had found the variables of T by itself. (Note however that the "partitioning step" must be applied to $[m] \setminus T$.)

Recall from Section 3.6 that the non-adaptive junta tester produces a number of disjoints subsets $I_1, \ldots, I_r \subseteq [m]$ satisfying the property written on page 56. For any $B \subseteq [m]$, the same argument goes through to give a series of disjoint independence tests on $I'_1, \ldots, I'_r \subseteq B$ with the property

- if $\mathrm{Inf}_f(B \setminus A) \geq \varepsilon$ for all $A \subseteq B, |A| = k$, then at least $k+1$ of the independence tests will be positive.

(In fact, $I'_1, \ldots, I'_r$ are precisely the intervals the junta tester would use for testing k-juntas on $[m] \setminus B$).

To adapt these ideas to our task, note that if $\mathrm{dist}(f, \mathrm{Jun}_k(T)) \geq \varepsilon$ then $\mathrm{Inf}_f([m] \setminus (T \cup A)) \geq \varepsilon$ for any $A \in \binom{[m]\setminus T}{k}$ (Lemma 3.6.1). Let $B = [m] \setminus T$ and $I'_1, \ldots, I'_r \subseteq B$ as before. We simply perform the independence tests of f on $I'_1, \ldots, I'_r$ and reject if at least $k+1$ were positive; both soundness and completeness follow from the preceding comments. Finally, the query complexity remains the same as that of the standard junta tester, and the second part of the last item follows because it is true of the independence tests. □

4.3.3 Testing junto-symmetry

The procedure to ε-test the property of being k-junto-symmetric, for small enough k, is described next.

1. Let $q = \theta(k^4 \log(k+1)/\varepsilon)$ bound the query complexity of Step 3.
2. Make queries to $T(f)$ to test that $\mathrm{Inf}_{T(f)}(V_1) < \frac{1}{18q}$ with confidence $> 8/9$ by performing an independence test (Lemma 3.2.3): take $O(q)$ random pairs $(\ell, r, x), (\ell, r', x)$ and compare $T(f)$ on them. If it isn't, reject.
3. Reject iff $\textsc{GeneralizedJuntaTester}_{\mu,8/9}(T(f), k, \varepsilon/5, V_0 \cup V_1)$ rejects.

Proof of Theorem 4.3.1. The algorithm is clearly non-adaptive and its query complexity is $\Theta(q) = \Theta(k^4 \log(k+1)/\varepsilon)$. We assume that n is large enough for $2k/\sqrt{2n} < 1/(18q) < \varepsilon/5$ to hold (small constant values for n can be dealt with separately in the tester).

The probability that an incorrect assessment is given by either the junta tester in step 3 or the influence test in step 2 is less than $2/9 < 2/3$. So if the overall test accepts with probability $\geq 2/3$, then $T(f)$ must be $\varepsilon/5$-close to a junta $\jmath'$ on $V_0 \cup V_1 \cup A$, $|A| \leq k$. In particular $\mathrm{Inf}_{T(f)}(V_2 \setminus A) \leq \varepsilon/5$. Moreover, since the influence test succeeded we also have $\mathrm{Inf}_{T(f)}(V_1) < \varepsilon/5$. Therefore $\mathrm{Inf}_{T(f)}(V_1 \cup (V_2 \setminus A)) \leq 2\varepsilon/5$, which means (by Lemma 3.6.1) that $T(f)$ is in fact $4\varepsilon/5$-close to a junta on $V_0 \cup A$. Consequently, f is $4\varepsilon/5 + k/\sqrt{2n} < \varepsilon$-close to junto-symmetric on A (Lemma 4.3.6), proving soundness.

On the other hand, suppose f is $1/(18q)$-close to a junto-symmetric function $\jmath$. Then there is $\jmath' \in \mathrm{Jun}_k(V_0)$ with $\mathrm{dist}(T(f), \jmath') \leq 1/(18q) + k/\sqrt{2n} < 1/(9q)$. Recall that every query of the junta tester to $T(f)$ follows the distribution $\mathcal{L} \times \mathcal{R} \times \{0,1\}^n$ (third item of Lemma 4.3.7), and this translates into uniform queries to f (we showed that $x^{r(l,x)}$ is uniformly distributed during the course of the proof of Lemma 4.3.6(a)). As the tester is non-adaptive, this means that the expected number of queries exposing a difference between $T(f)$ and $\jmath'$ is $1/9$, so with probability $8/9$ the tester can't see the difference between $T(f)$ and $\jmath'$. Hence we are effectively testing $\jmath'$ for the property of being a $V_0 \cup V_1 \cup A$ junta for some $|A| \leq k$, which it is indeed. Therefore step 3 accepts with probability $8/9$; and since $\mathrm{Inf}_{\jmath}(V_1) = 0$, we also have $\mathrm{Inf}_{T(f)}(V_1) \leq 2 \cdot \mathrm{dist}(T(f), \jmath') \leq 2k/\sqrt{2n} < 1/(18q)$ and step 2 also accepts with probability $8/9$. This establishes completeness. □

4.3.4 Testing isomorphism to junto-symmetric functions

In an analogous fashion one can reduce the problem of testing isomorphism to g (when g is close enough to $\mathcal{JS}_k$) to testing isomorphism between k-juntas. For this we can use a tolerant tester of isomorphism, except that, in view of

Lemma 4.3.6(e), the set of permutations allowed must be restricted to those fixing V_0 and V_1:

1. Use the algorithm of Theorem 4.3.1 to accept if $f \in \mathcal{JS}_k$ and reject if $\operatorname{dist}(f, \mathcal{JS}_k) > \varepsilon/30$.

2. Perform a suitable test to accept if $d \leq \varepsilon/10$ and reject if $d \geq 9\varepsilon/10$, where

$$d \triangleq \min_{\pi \in 1_{V_0,V_1} \times Sym(V_2)} \operatorname{dist}(T(f), T(g)^\pi)$$

Ignoring for the moment the implementation details of the second test, we show that the algorithm outlined is an isomorphism tester for $\mathcal{JS}_k$:

Proof of Theorem 4.3.2. We use the algorithm just described. The claim about the query complexity is clear.

Suppose the test accepts with high probability. Then $\operatorname{dist}(f, \mathcal{JS}_k) \leq \varepsilon/30$ and $d \leq 9\varepsilon/10$. Since $\operatorname{distiso}(g, \mathcal{JS}_k) \leq 1/k^5$, we have

$$|\operatorname{distiso}(f, g) - d| \leq \varepsilon/30 + 1/k^5 + 2k/\sqrt{2n} \leq \varepsilon/20,$$

so $\operatorname{distiso}(f, g) < \varepsilon$, as it should.

On the other hand, if $f \cong g$ then $\operatorname{dist}(f, \mathcal{JS}_k) = \operatorname{dist}(g, \mathcal{JS}_k) < 1/k^5$ and $d \leq 2/k^5 + (2k)/\sqrt{2n} < 1/k^4$, meaning that both tests succeed. If $\operatorname{distiso}(f, g) < 1/k^5$, then it also accepts with high probability because we can argue as before that since the test makes $O(k^4)$ queries that are individually uniformly distributed. □

Step 2 can be implemented using sample extractors. Let $D = V_0 \times V_1$, $f\colon D \times \{0,1\}^n \to \{0,1\}$ and let $\jmath' \in \operatorname{Jun}_D(A), A \in \binom{[n]}{k}$ be the element of $\operatorname{Jun}_k(D)$ closest to f. Define $\operatorname{core}_{k,D}(\jmath')\colon D \times \{0,1\}^k \to \{0,1\}$ by

$$\operatorname{core}_{k,D}(\jmath')(x\restriction_D, x\restriction_A) = \jmath'(x).$$

A correct sample for $\operatorname{core}_{k,D}(\jmath')$ (with respect to $\sigma \in 1_D \times S_k$) is a pair (x, a) with $x \in D \times \{0,1\}^k$ and $\operatorname{core}_{k,D}(f)(x^\sigma) = a$. An *$\eta$-noisy sampler* for $\operatorname{core}_{k,D}(\jmath')$ is a procedure to obtain an unlimited sequence of independent samples (x, a) such that each one is correct with probability $1 - \eta$ with respect to some fixed σ, and x follows the distribution $D \times \{0,1\}^k$.

The following two lemmas are all we need.

4.3.8. Lemma. *Suppose* $\operatorname{dist}(f, \operatorname{Jun}_k(D)) < 1/k^5$. *Then there is a* $\operatorname{poly}(k, 1/\varepsilon)$*-query non-adaptive algorithm to construct an* $\varepsilon/100$*-noisy sampler for* $\operatorname{core}_{k,D}(\jmath')$.

Proof (sketch). This is essentially Theorem 3.5.2. We need two changes. The first is that we substitute the adaptive junta tester for the junta tester used in

the proof. The second one is the observation that we know how the variables in D map to the variables in $\mathrm{core}_{k,D}(j')$, so for any $z \in \{0,1\}^n$, we only need to "extract" the setting of the k relevant variables sitting outside A. □

4.3.9. Lemma. *Let $f, g\colon D \times \{0,1\}^n \to \{0,1\}$, $g \in \mathrm{Jun}_k(D)$. Write*

$$d = \min_{\pi \in 1_D \times S_n} \mathrm{dist}(f, g^\pi)$$

Assuming access to an $\varepsilon/100$-noisy sampler for f, there is a $\mathrm{poly}(k/\varepsilon)$-query tester that accepts if $d \le \varepsilon/10$ and rejects if $d \ge 9\varepsilon/10$.

Proof (sketch). This is essentially Lemma 3.5.4. Construct a sample for $\mathrm{core}_{k,D}(j')$ and take $O(\log k!/\varepsilon^2) = O(k \log k/\varepsilon^2)$ random samples. These are enough to estimate

$$d' = \min_{\pi \in S_k} \mathrm{dist}(\mathrm{core}_{k,D}(j'), \mathrm{core}_{k,D}(g)^\pi)$$

to within $O(\varepsilon)$ additive error. Finally recall that d' and d are the same up to constant factors (this follows from Lemma 3.3.1). □

4.3.5 Tolerant testers

Using the tolerant tester in the third item of Theorem 3.6.8 instead of the $\mathrm{poly}(k/\varepsilon)$-query junta tester in the proof of the previous theorems, we obtain Theorem 4.3.4.

4.4 Hypergraph isomorphism

It is possible to establish a link between function isomorphism and a generalized form of graph isomorphism. Recall that an *undirected hypergraph* is a pair $H = (V, E)$, where V is a set of vertices and $E \subseteq \mathcal{P}(V)$ is a collection of hyperedges. Isomorphism between hypergraphs is defined in the natural way.

Now define the distance between two hypergraphs $H = (V, E)$ and $H' = (V, E')$ on the same set of vertices by $\mathrm{dist}(H, H') = |E \oplus E'|/2^n$, where $E \oplus E'$ is the symmetric difference between their edge sets. Testing function isomorphism is easily seen to be equivalent to testing isomorphism between undirected hypergraphs under this distance measure (this is the "dense hypergraphs model"). Indeed, a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ can be identified with the hypergraph with vertex set $V = [n]$ and edge set

$$f^{-1}(1) = \{x \in \{0,1\}^n \mid f(x) = 1\},$$

where binary vectors $x \in f^{-1}(1) \subseteq \{0,1\}^n$ are themselves identified with subsets of $[n]$ in the natural way. Clearly this satisfies

$$f \cong g \iff f^{-1}(1) \cong g^{-1}(1) \text{ as hypergraphs,}$$

and moreover the distance between f and g coincide from both viewpoints.

Seen this way, the problem of function isomorphism becomes a natural generalization of the analogous problem for graphs. This raises the question of whether progress towards the characterization can be made by studying hypergraph isomorphism in the line of previous works on graph isomorphism. One possible line of work is the study of uniform hypergraphs. An *r-uniform hypergraph* is one in which every edge $e \in E$ has size precisely r; the number r is also said to be the *arity* of the hypergraph. The distance between two r-uniform hypergraphs $H = (V, E), H' = (V, E')$ on the same vertex set of size $|V| = n$ is defined as $|E \oplus E'|/\binom{n}{r}$. Babai and Chakraborty [BC08b] studied this question and obtained worst-case query-complexity bounds for the case of uniform hypergraphs. Yet a characterization of the testability of isomorphism between uniform hypergraphs remains to be found.

In this work we prove an extension of Fischer's result that resolves the problem for hypergraphs of constant arity. To state it, recall that a *homomorphism* between $H = (V, E)$ and $\tilde{H} = (\tilde{V}, \tilde{E})$ is a mapping $\Pi\colon V \to \tilde{V}$ such that for all $\{v_1, \ldots, v_r\} \in V$, the implication $\{v_1, \ldots, v_r\} \in E \implies \{\Pi(v_1), \ldots, \Pi(v_r)\} \in \tilde{E}$ holds. The homomorphism Π is called *full* (and H is said to be *fully homomorphic* to $\hat{H}$) if it holds in both directions, i.e., if

$$\{v_1, \ldots, v_r\} \in E \iff \{\Pi(v_1), \ldots, \Pi(v_r)\} \in \tilde{E}.$$

Note that the size of $\tilde{V}$ may be much smaller than the size of V.

4.4.1. Definition. An r-uniform hypergraph H is *k-crunchable* if it is fully homomorphic to an r-uniform hypergraph with $\leq k$ vertices.

The *crunching number* of H is the smallest k such that H is k-crunchable.

The *ε-approximate crunching number* of H, denoted $CrunchNum_\varepsilon(H)$, is the smallest k such that H is ε-close to a k-crunchable r-uniform hypergraph.

The *ε-testing number* of H, denoted $TestNum_\varepsilon(H)$, is the minimum q for which there exists an ε-tester with q queries for the property of being isomorphic to H.

For graphs, having a constant crunching number is essentially the same as being in the algebra of constantly many cliques, or close to it (see Lemma 4.4.6).

We prove the following.

4.4.2. Theorem (Chakraborty et al. [CFGM12]). *For every $r \in \mathbb{N}$, $\varepsilon > 0$ there exists a pair of functions $L_{\varepsilon,r}(t)$ and $U_{\varepsilon,r}(t)$, with $\lim_{t\to\infty} L_\varepsilon(t) = \infty$, such that for every r-uniform hypergraph H we have*

$$L_{\varepsilon,r}(CrunchNum_\varepsilon(H)) \leq TestNum_\varepsilon(H) \leq U_{\varepsilon,r}(CrunchNum_{\varepsilon/3}(H)).$$

The original proof of Fischer for (a statement equivalent to) the special case of Theorem 4.4.2 when $r = 2$ applied the highly acclaimed Szeméredi regularity lemma [Sze76] for the *lower* bound (which is somewhat unusual as its normal use in property testing is to obtain upper bounds). Our simpler proof shows that this can be avoided. The lower bound method, which we call *crunching*, has additional applications, as outlined in the next subsection.

Now we prove Theorem 4.4.2. The functions L_ε and U_ε can be extracted from the proofs of the lower bound and the upper bound, respectively.

4.4.1 Lower bound via crunching

4.4.3. Definition. Let $\Pi\colon V \to V$ denote a mapping from V to itself. A Π-*crunch of* H is a hypergraph $H^\Pi_{cr} = (V, E')$ where

$$E' = \Big\{\{v_1, \ldots, v_k\} \mid \{\Pi(v_1), \ldots, \Pi(v_k)\} \in E\Big\}.$$

A k-*crunch* of a hypergraph is a Π-crunch for some Π with an image of size $\leq k$.

Note that every k-crunch is a k-crunchable hypergraph (as witnessed by the same mapping Π). When Π is injective, a Π-crunch of H is a hypergraph isomorphic to H. For a hypergraph $H = (V, E)$ and $k \leq |V| = n$, we show that any tester will have a hard time distinguishing non-injective crunchs from injective ones (permutations). A *random* k-*crunch* of H is a random hypergraph on V obtained as follows:

1. pick a subset $W \subseteq V$ of size k uniformly at random;
2. pick a mapping $\Pi\colon V \to W$ uniformly at random and output the Π-crunch of H.

Now define the distribution $\mathcal{D}^k_H$ by drawing a random permutation of a random k-crunch of H. Also write $\mathcal{D}_H$ for the uniform distribution over all permutations of H.

4.4.4. Lemma. *Let H be an r-uniform hypergraph and define $\mathcal{D}_H$ and $\mathcal{D}^k_H$ as before. Then it is impossible to distinguish a random $\tilde{H} \sim \mathcal{D}_H$ from a random $\tilde{H} \sim \mathcal{D}^k_H$ with $o(\sqrt{k}/r)$ queries.*

Proof. Let $q = o(\sqrt{k}/r)$ and $e_1, \ldots, e_q$ be the (adaptive, random) edge queries made. Let $Q \subseteq V$ be the set of at most rq vertices involved in these queries. Conditioned on the event $E_Q(\Pi)$ that Π is injective on Q, the distribution of replies to queries $e_1, \ldots, e_q$ is identical for $\mathcal{D}_H$ and $\mathcal{D}^k_H$. But $E_Q(\Pi)$ occurs except with probability at most $|Q|^2/k = o(1)$ as the choice of Π is independent of Q. This means that for any sequence $e_1, \ldots, e_q$ of queries and any sequence $a_1, \ldots, a_q$

of answers, the probability of obtaining answer a_i to query e_i for all i is, up to a factor of $\Pr[E_Q(\Pi)] = 1 - o(1)$, the same when $\tilde{H}$ is drawn from $\mathcal{D}_H$ as when it is drawn from $\mathcal{D}_H^k$. We conclude by Lemma 1.5.6 that the tester cannot distinguish $\mathcal{D}_H$ from $\mathcal{D}_H^k$ with q queries and success probability $\geq 2/3$. □

4.4.5. Corollary. *If an r-uniform hypergraph is ε-far from being k-crunchable, then ε-testing isomorphism to it requires $\Omega\left(\sqrt{k}/r\right)$ queries.*

Together with the upper bound in the following subsection, this provides a characterization of hypergraphs of constant arity that can be tested for isomorphism with $O(1)$ queries. To see how this generalizes Fischer's result for graphs, we show that being $O(1)$-chunchable is equivalent to having "algebra number" $O(1)$ as well.

4.4.6. Definition. The *algebra number* of a graph G is the smallest number k for which there exist cliques $C_1, \ldots, C_k$ over subsets of the vertex set of G, such that G can be generated from the edge sets of $C_1, \ldots, C_k$ by taking set unions, intersections and complementations (the latter with respect to the edge set of a complete graph).

The *ε-approximate algebra number* is the smallest k such that H is ε-close to some graph whose algebra number is k.

We also define the *pairing number* as the smallest k for which there are k *vertex-disjoint* sets $A_1, \ldots A_k \subseteq V$ and a subset $S \subseteq [k] \times [k]$ such that the edge set of G is $E = \left\{\{v, w\} \mid v \in A_i, w \in A_j, (i, j) \in S, v \neq w\right\}$ (note that $i = j$ is allowed but loops are not). The *ε-approximate pairing number* of G is defined similarly.

4.4.7. Lemma.

1. *Any graph with pairing number k has algebra number $\leq k^2$.*
2. *Any graph with algebra number k has pairing number $\leq 2^k$.*
3. *Any k-crunchable graph has pairing number k. Conversely, any graph with pairing number k is ε-close to being k^2/ε-crunchable.*

Proof.

1. Let $cl(A)$ denote the edge set of the clique with vertex set $A \subseteq V$. It is enough to show that for disjoint $A_1, A_2 \subseteq V$, the set of edges between A_1 and A_2 is in the algebra generated by $cl(A_1), cl(A_2)$ and $cl(A_1 \cup A_2)$. This is easy to see because the set of edges in question is equal to $\overline{cl(A_1) \cup cl(A_2)} \cap cl(A_1 \cup A_2)$.

2. Let $G = (V, E)$ be generated from the edge sets of the cliques $C_1, \ldots, C_k \subseteq V$. For $S \subseteq [k]$, let $A_S = (\cap_{i \in S} C_i) \bigcap (\cap_{i \notin S} \overline{C_i})$. These 2^k sets are disjoint and contain all vertices incident with some edge in G. For all $S, T \subseteq [k]$, if $a_1, a_2 \in A_S$ and $b_1, b_2 \in A_T$, then $(a_1, b_1) \in E$ iff $(a_2, b_2) \in E$ (unless $a_1 = b_1$ or $a_2 = b_2$). This means G has pairing number k since it is possible to write E in the required form.

3. We prove the second statement (the first one is obvious). Suppose G has pairing number k and let $A_1, \ldots, A_k$ be as in Definition 4.4.6. The only reason G may not be k-crunchable is the possible existence of edges between vertices in the same A_i. Divide each A_i into $t = \lceil 1/\varepsilon \rceil$ subsets $A_{i1}, \ldots, A_{ik}$ of roughly equal size and remove the edges with both endpoints inside the same A_{ij}. If n is divisible by t, then from all $\binom{n}{2}$ possible edges, the removed ones constitute a fraction bounded by $t \cdot (1/t^2) = 1/t \leq \varepsilon$; a simple argument shows that the same bound still holds in the general case. Hence this graph is ε-close to the original graph, and is also k-crunchable by construction.

□

4.4.2 Upper bound via partition properties

For the upper bound we need to discuss "partition properties" of hypergraphs, which generalizes those discussed in the context of graphs by Goldreich, Goldwasser and Ron [GGR98]. A *graph partition instance* ψ is composed of an integer k specifying the number of sets in the required partition $V_1, \ldots, V_k$ of the graph's vertex set, and intervals specifying the allowed ranges for the number of vertices in every V_i and the number of edges between every V_i and V_j for $i \leq j$. Many problems, such as k-corolability and maximum clique, can be easily formulated in this framework. In [GGR98] the authors presented algorithms for testing if a graph satisfies a certain partition property. We use a similar notion for hypergraphs, taken from the work of Fischer, Matsliah and Shapira [FMS10]. They work with directed hypergraphs, but we state their results in terms of undirected hypergraphs.

Hypergraph partition property

Let $H = (V, E)$ be a directed r-uniform hypergraph. be a partition of V. Let us introduce a notation for counting the number of edges from E with a specific placement of their vertices within the partition classes of Π. We denote by Φ the set of all possible mappings $\phi\colon [r] \to [k]$. We think of every $\phi \in \Phi$ as mapping the vertices of an r-tuple to the components of Π. We denote by $E^{\Pi}_{\phi} \subseteq E$ the following collection of r-tuples:

$$E^{\Pi}_{\phi} = \Big\{ (v_1, \ldots, v_r) \in E \mid \forall j \in [r] :\ v_j \in V^{\Pi}_{\phi(j)} \Big\}.$$

4.4.8. Definition. A *density tensor* of *order* k and arity r is a sequence $\psi = \left\langle \langle \rho_j \rangle_{j\in[k]}, \langle \mu_\phi \rangle_{\phi\in\Phi} \right\rangle$ of reals between 0 and 1. (The interpretation is that they specify the presumed normalized sizes of $|V_i^\Pi|$ and $|E_\phi^\Pi|$ of a k-partition of a hypergraph of arity r.) Whenever k and r are clear from context, we call ψ simply a *density tensor*.

In particular, given a k-partition $\Pi = \{V_1^\Pi, V_2^\Pi, \ldots, V_k^\Pi\}$ of a hypergraph H, we set ψ^Π to be the density tensor $\langle \langle \rho_j^\Pi \rangle_{j\in[k]}, \langle \mu_\phi^\Pi \rangle_{\phi\in\Phi} \rangle$ with the property that for all j, $\rho_j^\Pi = \frac{1}{n} \cdot |V_j^\Pi|$ and for all ϕ, $\mu_\phi^\Pi = \frac{1}{n^r} \cdot |E_\phi^\Pi|$.

4.4.9. Definition. For a fixed hypergraph H of arity r, a set Ψ of density tensors (of order k and arity r) defines a property of the k-partitions of $V(H)$ as follows. We say that a partition Π of $V(H)$ (exactly) *satisfies* Ψ if there exists a density tensor $\psi = \left\langle \langle \rho_j \rangle_{j\in[k]}, \langle \mu_\phi \rangle_{\phi\in\Phi} \right\rangle \in \Psi$, such that ψ and the density tensor ψ^Π of Π are equal. Namely, Π *satisfies* Ψ if there is $\psi = \left\langle \langle \rho_j \rangle_{j\in[k]}, \langle \mu_\phi \rangle_{\phi\in\Phi} \right\rangle \in \Psi$ such that

- for all $j \in [k]$, $\rho_j^\Pi = \rho_j$;
- for all $\phi \in \Phi$, $\mu_\phi^\Pi = \mu_\phi$.

We extend this notion of satisfying partitions (and equivalence between density tensors) in two ways: one with respect to the edge density parameters $\langle \mu_\phi \rangle$, and the other with respect to the usual closeness measures between hypergraphs.

4.4.10. Definition. A k-partition Π *ε-approximately* satisfies Ψ if there is $\psi = \left\langle \langle \rho_j \rangle_{j\in[k]}, \langle \mu_\phi \rangle_{\phi\in\Phi} \right\rangle \in \Psi$ such that

- for all $j \in [k]$, $\rho_j^\Pi = \rho_j$;
- for all $\phi \in \Phi$, $\mu_\phi^\Pi = \mu_\phi \pm \varepsilon$.

In this case ψ^Π is *ε-approximate* to ψ.

By extension (and with a slight abuse of notation), we say that the hypergraph H itself *satisfies* the property Ψ if there exists a partition Π of H's vertices that satisfies Ψ, and similarly we say that H itself *ε-approximately* satisfies the property Ψ if there exists a partition of H's vertices that ε-approximately satisfies the property Ψ. In addition, we may consider a specific density tensor ψ as a singleton set $\Psi = \{\psi\}$, and accordingly as a property of partitions.

We define one additional measure of closeness to the property Ψ. The distance of a hypergraph H from the property Ψ is defined as $\mathrm{dist}(H, \Psi) = \min_{H'}\{\mathrm{dist}(H, H') \mid H' \text{ satisfies } \Psi\}$. For $\varepsilon > 0$ we say that H is *ε-far* from satisfying the property Ψ when $\mathrm{dist}(H, \Psi) \geq \varepsilon$, and otherwise, H is *ε-close* to Ψ. The testing algorithm follows immediately from the following theorem.

4.4.11. Theorem (Fischer, Matsliah & Shapira [FMS10]). *For any two $k, r \in \mathbb{N}$, and any set Ψ of density tensors of order k and arity r, there exists a randomized algorithm A_T taking as inputs two parameters $\varepsilon, \delta > 0$ and an oracle access to a hypergraph H of arity r, such that*

- *if H satisfies Ψ, then with probability at least $1-\delta$ the algorithm A_T outputs* ACCEPT*;*
- *if H does not even ε-approximately satisfy the property Ψ, then with probability at least $1-\delta$ the algorithm A_T outputs* REJECT*.*

The query complexity of A_T is bounded by $\log^3(\frac{1}{\delta}) \cdot \mathrm{poly}(k^r, \frac{1}{\varepsilon})$, and its running time is bounded by $\log^3(\frac{1}{\delta}) \cdot \exp\left((\frac{r}{\varepsilon})^{O(r \cdot k^r)}\right)$.

For us it is enough to consider set Ψ with a single partition property ψ.

Hypergraphs with small approximate crunching number

Let $H = (V, E)$ be a directed k-crunchable r-uniform hypergraph. We can define crunchings of directed hypergraphs in a similar manner, with the corresponding mapping $\Pi\colon V \to [k]$ and an hypergraph $\tilde{H} = ([k], \tilde{E})$ defining the edge patterns of H, i.e.,

$$\tilde{E} = \Big\{(v_1, \ldots, v_k) \mid (\Pi(v_1), \ldots, \Pi(v_k)) \in E\Big\}.$$

The algorithm above can be used as a testing algorithm in the traditional sense on account of the following observations.

4.4.12. LEMMA. *Let $\varepsilon_0 < \varepsilon/k^r$. Any directed hypergraph that ε_0-approximately satisfies a partition property Ψ is also ε-close to satisfying it.*

Proof. Let Π be a partition witnessing the fact that the hypergraph ε_0-approximately satisfies Ψ. For every $\phi \in \Phi$, we can add or remove $\varepsilon_0 n^r$ edges to/from E^{Π}_{ϕ} so that the resulting graph exactly satisfies Ψ. Since $|\Phi| = k^r$, this entails changing less than an ε-fraction of all possible edges. □

4.4.13. LEMMA. *Let H_0, H_1 denote directed hypergraphs on n vertices, where H_1 is the closest k-crunchable hypergraph to H_0. Suppose H_0 is $\varepsilon/3$-close to H_1 and the crunch is defined via the map $\Pi\colon V(H_0) \to V(H_1)$. We can assume $V(H_0) = V(H_1) = [n]$. Let $\psi = \big\langle \langle \rho_j \rangle_{j\in[k]}, \langle \mu_\phi \rangle_{\phi\in\Phi} \big\rangle$ denote the following density tensor of order k and arity r:*

- *for all $j \in [k]$, $\rho_j = \frac{\Pi^{-1}(j)}{n}$;*
- *for $\phi \in \Phi$, $\mu_\phi = \frac{E^{\Pi}_{\phi}(H_0)}{n^r}$.*

Let $\varepsilon_0 < 2\varepsilon/(9k^r)$. Then H_0 satisfies $\{\psi\}$, and every hypergraph that ε_0-approximately satisfies $\{\psi\}$ is $8\varepsilon/9$-close to being isomorphic to H_0.

Proof. Observe that in the density tensor above, the partition sizes ρ_j are defined by H_1, but the edge densities μ_ϕ are those of H_0. Note that H_0 satisfies $\{\psi\}$ by definition. The k-crunchable hypergraph H_1 does not satisfy the property ψ, but it does satisfy a related partition property ψ_1 with the same partition sizes $\{\rho_j\}$ but where the edge densities $\{\mu_\phi\}$ are all zero or one.

Take any hypergraph H_3 that ε_0-approximately satisfies ψ. By Lemma 4.4.12, it is $2\varepsilon/9$-close to some hypergraph H_2 that satisfies ψ. We show that H_2 is $\varepsilon/3$-close to satisfying ψ_1. The reason is that for any $\phi \in \Phi$, the number of edges of type ϕ that we need to change is $\min(\mu_\phi, 1-\mu_\phi)n^r$. So by modifying

$$\sum_{\pi\in\Phi} \min(\mu_\phi, 1-\mu_\phi)n^r$$

edges we can obtain a hypergraph that satisfies ψ_1. But this expression is also the number of edges that we need to change from H_0 so that it satisfies ψ_1, which is the distance between H_0 and H_1, hence at most $\varepsilon/3$.

Moreover, the only r-uniform directed hypergraph that satisfies ψ_1 is H_1, up to isomorphism. Therefore H_2 is $\varepsilon/3$-close to isomorphic to H_1, and $2\varepsilon/3$-close to H_0. Hence H_3 is $8/9$-close to begin isomorphic to H_0. □

For undirected hypergraphs, simply replace each edge $\{v_1, \dots, v_r\}$ with all $r!$ directed edges of the form $(v_{\pi(1)}, \dots, v_{\pi(r)})$ for a permutation $\pi\colon [r] \to [r]$, and test isomorphism to this directed version. The following follows.

4.4.14. Theorem. *Let $\varepsilon \in (0,1)$. Testing isomorphism to an r-uniform hypergraph that is $\varepsilon/3$-close to k-crunchable can be done with* $\mathrm{poly}(k^r/\varepsilon)$ *queries.*

4.4.3 Proof of the characterization

Proof of Theorem 4.4.2. By definition, any hypergraph H is ε-far from $(CrunchNum_\varepsilon(H)-1)$-crunchable, so by Lemma 4.4.5, we have

$$TestNum_\varepsilon(H) \geq L_{\varepsilon,r}(CrunchNum_\varepsilon(H))$$

for some $L_{\varepsilon,r}(t) = \Omega\big(\frac{\sqrt{t-1}}{r}\big)$. Clearly $\lim_{n\to\infty} L_\varepsilon(t) = \infty$.

For the upper bound, any hypergraph H is $\varepsilon/3$-close to $CrunchNum_{\varepsilon/3}(H)$-crunchable. Hence we have by Theorem 4.4.14 that

$$TestNum_\varepsilon(H) \leq U_{\varepsilon/3,r}(CrunchNum_{\varepsilon/3}(H)),$$

for some appropriate polynomial $U_{\varepsilon,r}(t) = \mathrm{poly}(t^r/\varepsilon)$. □

4.5 Junto-symmetric functions vs. layered juntas

Now we address the question of what happens when we generalize our definition of k-junto-symmetric to all functions that are k-juntas when restricted to any constant-weight layer of the cube (we call them *layered juntas*), and show that in general these functions are no longer testable for isomorphism. The proof applies the crunching method to boolean functions. In this setting the procedure resembles an idea used by Blais and O'Donnell [BO10].

4.5.1. Definition. A function $f\colon \{0,1\}^n \to \{0,1\}$ is called a *layered k-junta* if there are subsets $J_0, \dots, J_n \subseteq [n]$, each of size k, and functions $\tilde{f}_0, \dots \tilde{f}_n\colon \{0,1\}^k \to \{0,1\}$ so that for all $x \in \{0,1\}^n$,

$$f(x) = \tilde{f}_{|x|}\big(x\restriction_{J_{|x|}}\big).$$

Perhaps it should be stressed that layered k-juntas are not, in general, k-juntas. Let $\mathcal{LJ}_k$ denote the class of layered k-juntas, respectively. Note that $\mathcal{JS}_k \subseteq \mathcal{LJ}_k$.

We need a notion of random crunching for functions. The notion for hypergraphs provides a possible definition of function crunching via the equivalence discussed in Section 4.4, but unfortunately this kind of crunching would alter the Hamming weight of inputs, which could be easily detected by the tester for some functions. Here we give a slightly different definition that resolves this issue, but only applies to layered juntas, and also happens to depend on the particular choice of each $\tilde{f}_i$.

4.5.2. Definition. A *random t-crunch* of the function f defined by $f(x) = \tilde{f}_{|x|}(x\restriction_{J_{|x|}})$ is a function $g \in \mathcal{JS}_t$ obtained as follows:

1. pick, uniformly at random, a subset $J \subseteq [n]$ of size t and a mapping $\gamma\colon [n] \to J$;
2. for every $x \in \{0,1\}^n$, let $i_1, \dots, i_k$ denote the indices in $J_{|x|}$; set $g(x) = \tilde{f}_{|x|}(x_{\gamma(i_1)} \cdots x_{\gamma(i_k)})$ and return g.

4.5.3. Theorem (Chakraborty et al. [CFGM12]). *Fix $\varepsilon > 0$ and $Q\colon \mathbb{N} \to \mathbb{N}$, and suppose $f \in \mathcal{LJ}_k$. Then $\Omega(Q(k))$ queries are needed to distinguish a random permutation of f from a random permutation of a random $(k \cdot Q(k))^2$-crunch of f.*

In particular, if f is ε-far from $\mathcal{JS}_{(k\cdot Q(k))^2}$, then ε-testing isomorphism to f requires $\Omega(Q(k))$ queries.

Proof. Let $\mathcal{D}_{\text{yes}}$ denote the random permutations of f and $\mathcal{D}_{\text{no}}$ the distribution of random permutations of t-crunchs of f. Given $g \in \mathcal{D}_{\text{no}}$ and its corresponding mapping $\gamma\colon [n] \to J$, call a set of layers $\{\ell_1, \dots, \ell_m\}$ *collision-free* if γ is injective on

$\bigcup_{i\in[m]} J_{\ell_i}$. Let there be a deterministic tester that makes $q = o(Q(k))$ queries. For every $x^1, \dots, x^q \in \{0,1\}^n$ let $E_{x^1,\dots,x^q}$ denote the event that the set $\{|x^1|, \dots, |x^q|\}$ of layers containing inputs queried is collision-free with respect to the randomly chosen mapping γ of a function $g \sim \mathcal{D}_{\text{no}}$. Observe that for all $x^1, \dots, x^q \in \{0,1\}^n$ and $w \in \{0,1\}^q$, conditioned on $E_{x^1,\dots,x^q}$ we have

$$\Pr_{h\sim\mathcal{D}_{\text{yes}}}[h(x^1), \dots, h(x^q) = w] = \Pr_{h\sim\mathcal{D}_{\text{no}}}[h(x^1), \dots, h(x^q) = w].$$

By Lemma 1.5.6 (and the ensuing remark), it is enough to show that $E_{x^1,\dots,x^q}$ occurs with probability $> 2/3$.

The probability (over $g \in \mathcal{D}_{\text{no}}$) that $\gamma(i) = \gamma(j)$ for a specific pair $i \neq j$ is $(k \cdot Q(k))^{-2}$. The number of different pairs $i, j \in \bigcup_{i\in[q]} J_{|x^q|}$ is bounded by $(kq)^2 = o((k \cdot Q(k))^2)$, hence by the union bound the probability that the set $\{|x^1|, \dots, |x^q|\}$ of layers is collision-free is $1 - o(1)$. □

Note that this sort of argument admits certain generalizations. For example, we can consider functions that have few additional variables outside a known set A (as in Section 4.3.3), as the address function on n variables (which behaves as a 1-junta for any fixed setting of the $\approx \log n$ addressing variables). Choosing q small enough for address function to be far from isomorphic to a q^2-crunching of the addressee variables gives a lower bound of $\Omega(q)$ for testing isomorphism to it.

4.6 Linear isomorphism

We turn our attention now to a more general notions of isomorphism, namely equivalence up to transformations by an arbitrary invertible linear map over $\mathbb{F}_2^n$ (note that isomorphism in the usual sense corresponds to the linear application defined by a permutation matrix). We show that functions that are far from having constant Fourier dimension are hard to test for isomorphism.

4.6.1. Definition. Two boolean functions $f, g\colon \{0,1\}^n \to \{0,1\}$ are said to be *linearly isomorphic* if there exists a full-rank linear transformation $A\colon \{0,1\}^n \to \{0,1\}^n$ such that $f = g \circ A$.

This is an equivalence relation by virtue of the requirement that A have full rank.

4.6.2. Definition. Let $f(x) = \sum_{S\in\{0,1\}^n} \tilde{f}(S)\chi_S(x)$ be the Fourier expansion of the function $f\colon \{0,1\}^n \to \mathbb{R}$. Let

$$A \triangleq \big\{S \in \{0,1\}^n \mid \tilde{f}(S) \neq 0\big\}.$$

Then the dimension of the span of A is called the *Fourier dimension* of f.

4.6.3. Lemma. *The function f is linearly isomorphic to some k-junta iff its Fourier dimension is at most k.*

Proof. Suppose f has Fourier dimension k. Then it is a real linear combination of parities whose defining vectors lie on a k-dimensional vector space V. Here we take the parities χ_v to be ±1-valued. Each parity in V can be written as a product of some parities in a basis for V. It follows that f can be written as a function h (not necessarily linear) of $k' \leq k$ linearly independent parities:

$$f(x) = h\big(\chi_{v_1}(x), \dots, \chi_{v_{k'}}(x)\big) = g\big(\langle v_1, x\rangle, \langle v_2, x\rangle, \dots, \langle v_{k'}, x\rangle, \bullet\big),$$

where g is a junta on the first k' variables, the inner products are taken over $\mathbb{F}_2^n$, and $\bullet$ symbolizes that the remaining $n-k'$ variables are irrelevant. The function g can easily seen to be boolean-valued on $\{0,1\}^n$ if f is, because all $2^{k'}$ assignments to $\chi_{v_i}(x), i = 1 \dots k'$ are possible. Hence there is a k'-junta $g\colon \{0,1\}^n \to \{0,1\}$ and a change of basis $A\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that $f = g \circ A$ (take $v_1, \dots, v_k'$ as the first rows of the matrix associated with A).

Conversely, if $f = g \circ A$ for a k-junta g, then f is a junta on a set P of k parity functions and can be written as a polynomial on those parities. We can replace products of parities in P by a single parity whose defining vector is in the linear span of the defining vectors of the parities in the product. This means that f can also be written as a linear combination of the parities in the span of P, so f has Fourier dimension at most k. □

4.6.4. Theorem. *If $f\colon \{0,1\}^n \to \{0,1\}$ is ε-far from having Fourier dimension k then any adaptive ε-tester for linear isomorphism to f takes at least $k-1$ queries.*

Proof. Let $f\colon \{0,1\}^n \to \{0,1\}$ be ε-far from having Fourier dimension k. We will use Lemma 1.5.6 to prove the lower bound. We want to generate two distributions of functions D_Y and D_N for the yes-instances and no-instances respectively:

$\mathcal{D}_{\text{yes}}$: To generate a random function g_Y in $\mathcal{D}_{\text{yes}}$ pick a random linear transformation $L\colon \{0,1\}^n \to \{0,1\}^n$ of full rank and let $g_Y(x) \triangleq f(Lx)$.

$\mathcal{D}_{\text{no}}$: To generate a random function g_N in $\mathcal{D}_{\text{no}}$ pick a random linear transformation $R\colon \{0,1\}^n \to \{0,1\}^n$ of rank exactly k and let $g_N(x) \triangleq f(Rx)$.

Note that $\mathcal{D}_{\text{yes}}$ is a distribution supported on the set of those functions which are linearly isomorphic to f. On the other hand, $\mathcal{D}_{\text{no}}$ is supported on those functions that are ε-far from linearly isomorphic to f. This is because if $g_N \in \mathcal{D}_{\text{no}}$, then there exists a linear transformation R of rank k such that $g_N(x) = f(Rx)$, so g_N has Fourier dimension k.

Let $Q \subseteq \{0,1\}^n$ and let $q_1, \ldots, q_t$ be a basis of the span of Q. Note that when L is a random linear transformation of full rank then $L(q_1), L(q_2), \ldots, L(q_t)$ are linearly independent. In fact, given any t linearly independent vectors $v_1, \ldots, v_t$,

$$\Pr_L [\forall i : L(q_i) = v_i] = 1/M,$$

where M is the number of distinct sets of t independent vectors.

When R is a random linear transformation of rank k the set of vectors $\{R(q_1), \ldots, R(q_t)\}$ need not be linearly independent in general, but if $t < k$ they are independent with high probability.

4.6.5. Lemma. *If $\{q_1, \ldots, q_t\}$ is a set of linearly independent vectors then when R is a random linear transformation of rank k, then with probability $1 - 1/2^{k-t}$, the set $\{R(q_1), \ldots, R(q_t)\}$ is linearly independent.*

Proof. Let us assume that the set $\{R(q_1), \ldots, R(q_t)\}$ is not linearly independent. So there must be a linear combination of the vectors that add up to zero. That is there must be $a_1, \ldots, a_t \in \{0,1\}$ such that $\sum_{i=1}^{t} a_i R(q_i) = 0$. In other words, there exists a vector v in the span of $q_1, \ldots, q_t$ such that $R(v) = 0$.

Because R is a randomly chosen linear transformation of rank k,

$$\forall v \in \{0,1\}^n : \Pr_R [R(v) = 0] = \frac{1}{2^k}.$$

So the expected number of vectors in the span of $q_1, \ldots, q_t$ such that $R(v) = 0$ is $1/2^{k-t}$. And thus by Markov's Inequality,

$$\Pr [R(q_1), R(q_2), \ldots, R(q_t) \text{ are linearly independent}] \geq 1 - \frac{1}{2^{k-t}}.$$

□

In fact, conditioned on the event E that $\{R(q_1), \ldots, R(q_t)\}$ are linearly independent, any set of linearly independent vectors is equally likely. Thus, for any t linearly independent vectors $v_1, \ldots, v_t$,

$$\Pr_R [\forall i \; R(q_i) = v_i \mid E] = 1/M = \Pr_L [\forall i : L(q_i) = v_i].$$

And since Q is contained in the span of $q_1, \ldots, q_t$, for all $a \in \{0,1\}^{|Q|}$ we get

$$\begin{aligned} \Pr_{g_N \leftarrow \mathcal{D}_{\text{no}}} \left[g_N \restriction_Q = a \right] &\geq (1 - 2^{t-k}) \Pr_{g_N \leftarrow \mathcal{D}_{\text{no}}} \left[g_N \restriction_Q = a \mid E \right] \\ &= (1 - 2^{t-k}) \Pr_{g_Y \leftarrow \mathcal{D}_{\text{yes}}} \left[g_Y \restriction_Q = a \right]. \end{aligned}$$

Therefore, if $t \leq k - 2$ we have

$$\Pr_{g_N \leftarrow \mathcal{D}_{\text{no}}} \left[g_N \restriction_Q = a \right] \geq (3/4) \Pr_{g_Y \leftarrow \mathcal{D}_{\text{yes}}} \left[g_Y \restriction_Q = a \right].$$

By Lemma 1.5.6, if f is ε-far from having Fourier dimension k then any tester (even an adaptive one) for testing linear isomorphism to f must make at least $k-1$ queries. □

4.6.6. Remark. Gopalan et al [GOS+09] proved that if $f\colon \{0,1\}^n \to \{0,1\}$ has Fourier dimension k then testing linear isomorphism to f can be done using $O(k2^k)$ queries.

4.7 Summary

We proved that isomorphism to any boolean function on the n-dimensional hypercube with a polynomial number of distinct permutations can be tested with a number of queries that is independent of n. To do this we introduced the notion of junto-symmetric functions, proved its equivalence with poly-symmetric functions, and reduced the problem to certain generalizations of junta testing. We also showed some partial results in the converse direction. A complete characterization, however, remains open.

We also considered isomorphism testers against a *uniform* hypergraph that is given in advance. Our results regarding the latter topic generalize the known classification of the testability of graph isomorphism, and in the process we also provided a simpler proof of his original result which avoids the use of Szemerédi's regularity lemma.

Some related problems were also discussed, such as testing isomorphism up to linear transformations.

Chapter 5

Group testing, non-adaptivity, and explicit lower bounds

We have been making a thorough study of the difficulty of testing function isomorphism over the previous chapters. Although we know that the complexity is $\widetilde{\Omega}(n)$ in the worst case, we are yet to offer an explicit example of a function meeting the lower bound. We describe here a series of recent results that prove that k-parities are hard for testing isomorphism, and subsequently delve into the intimate connections between junta/isomorphism testing and the area of group testing. The content of this chapter is based on the manuscripts

- S. Chakraborty, D. García–Soriano and A. Matsliah. Notes on testing k-parities, 2011.
- D. García–Soriano, A. Matsliah and R. de Wolf. Untitled, 2011.

5.1 Measuring the size of a parity

Parities are a natural candidate of functions for which testing isomorphism is hard. Under the additional promise that the input function is linear, testing isomorphism to a k-parity reduces to checking if a given linear function has size precisely k. (In the absence of such a promise, one can always self-correct f a la BLR. This increases the complexity by a logarithmic factor, but as we shall see this is not needed.) Another way of looking at the problem is as determining, by making as few queries as possible to the Hadamard encoding of a word x, whether $|x| = k$ or not. Indeed, let $f = x^*$ be the parity of the bits in the support of x. For all $y \in \{0,1\}^n$, we have $f(y) = \langle x, y \rangle$, which is the yth bit of the Hadamard encoding of x. So the task is essentially how to compute $|x|$ efficiently[1] if we can query the XOR of arbitrary subsets of the bits of x. (Decision trees where the

[1]Remarkably, this task can be accomplished in the quantum setup with just *one* query, by deploying the Bernstein-Vazirani algorithm [BV97] (in fact, x itself can be found).

queries are allowed to be XORs of subsets of the inputs have appeared in the literature [ZS10].)

Deciding if the size of a parity is k is the same problem as deciding if it is $n-k$ because of the observation leading to Lemma 2.3.4. For even n, the case $k = n/2$ is particularly interesting because it enables us to verify the equality between the sizes of two unknown parities $f, g \in \mathsf{PAR}^n$. Indeed, define a parity on $2n$ variables by $h(x_1x_2) = f(1^n \oplus x_1) \oplus g(x_2)$, where $x_1, x_2 \in \{0,1\}^n$; then $h \in \mathsf{PAR}^{2n}_n$ if and only if f and g are isomorphic.

A related problem is determining if a parity has size *at most* k (naturally, this is equivalent to the problem of deciding if the size is at least $n-k$, or at most $n-k-1$). Upper bounds for this task imply upper bounds for testing isomorphism to k-parities (one can perform one test to verify the condition $|x| \leq k$ and another one for $|x| \leq k-1$). Lower bounds here do not immediately imply lower bounds for testing isomorphism, but they do imply lower bounds for testing k-juntas (because one way of checking if $f \in \mathsf{PAR}_{\leq k}$ is testing that f is linear and also a k-junta).

The first step towards analyzing the hardness of these problems was taken by Goldreich.

5.1.1. Theorem (Goldreich [Gol10, Theorem 4]). *Testing if a linear function $f \in \mathsf{PAR}^n$ (n even) is in $\mathsf{PAR}^n_{\leq n/2}$ requires $\Omega(\sqrt{n}\,)$ queries.*

5.1.2. Theorem (Goldreich). *Testing if a linear function $f \in \mathsf{PAR}^n$ (n even) is in $\mathsf{PAR}^n_{n/2}$ requires $\Omega(\sqrt{n}\,)$ queries.*

The second theorem was not stated in [Gol10], but follows from the proofs therein.

He conjectured that the true bound should be $\Theta(n)$, which will be confirmed in the next subsection. Remarkably, by using one of the other results in the very same paper, it is possible to strengthen the bound of Theorem 5.1.1 to $\widetilde{\Omega}(n)$, and to give a simpler proof of Theorem 5.1.2 up to polylogarithmic factors.

5.1.3. Theorem (Goldreich). *[Gol10, Corollary 2.2] At least $\Omega(n)$ queries are needed to distinguish a random parity on n variables from a random parity whose size is a multiple of three.*

5.1.4. Theorem (Chakraborty et al. [CGM11d]). *Testing if a linear function $f \in \mathsf{PAR}^n$ (n even) is in $\mathsf{PAR}^n_{n/2}$ requires $\widetilde{\Omega}(n)$ queries.*

Proof. Fix n and suppose that for every $k \leq n$ there is a tester $\mathcal{A}_k$ that can determine, under the assumption $f \in \mathsf{PAR}^n$, whether f is a parity of size $\leq k$ using $o(n/(\log n \log\log n))$ queries. Then, by standard binary search and probability amplification, one could find the exact number of influential variables of f in $o(n)$ steps. This contradicts Theorem 5.1.3, so for every n there must be some $k = k(n)$

such that deciding if $f \in \mathsf{PAR}^n_{\leq k}$ (for $f \in \mathsf{PAR}^n$) needs $\Omega(n/(\log n \log\log n))$ queries. (In fact, the additional $\log\log n$ factor in the denominator can be avoided because of the results of Feige et al. [FRPU94] about noisy boolean decision trees) If $k = n/2$ we are done. Since we can replace k with $n - k - 1$, we can assume $k > n/2$.

Now we prove that determining whether some function $g \in \mathsf{PAR}^{2k}$ belongs to PAR^{2k}_k requires $\widetilde{\Omega}(n/(\log n \log\log n))$ queries. We argue by contradiction. Take any tester for this property making $o(n/(\log n \log\log n))$ queries and, for each $f \in \mathsf{PAR}^n$, let the tester's input be the function $g_f \colon \{0,1\}^{2k} \to \{0,1\}$, where $g_f(x \sqcup y) = f(x)$ for any $x \in \{0,1\}^n$ and $y \in \{0,1\}^{2k-n}$. Clearly, the number of influential variables of f and g_f is the same and hence one would be able to determine for any $f \in \mathsf{PAR}^n$ whether the size of f is $\leq k$ or $> k$ using $o(n/(\log n \log\log n))$ queries. But this is not possible as shown in the previous paragraph.

Summarizing, for every n there exists some (unknown)

$$n' = \max(2k(n), 2(n - k(n) - 1)) = \Theta(n)$$

such that determining whether $g \in \mathsf{PAR}^{n'}$ is a parity of $\leq n'/2$ bits needs

$$\Omega(n/\log n \log\log n) = \Omega(\log n'/\log\log n')$$

queries. Hence the lower bound holds for infinitely many n. □

In fact, since Theorem 5.1.3 is a statement about random parities, and a random parity has size circa $n/2 \pm O(\sqrt{n})$ with high probability, one can argue (proof omitted) that this also implies an $\widetilde{\Omega}(\sqrt{n})$ lower bound for checking if the size of f is $n/2$, which almost matches Theorem 5.1.1.

5.1.1 Relationship to communication complexity

In a recent paper, Blais, Brody and Matulef noticed a nice connection with some well-studied problems in communication complexity. In this setup, introduced by Yao [Yao79], two parties, traditionally called Alice and Bob, each have an input and they need to devise a protocol to determine some property of the joint input. Unlimited access to their respective inputs and arbitrary computations are allowed, and the measure for the protocol's efficiency is provided by the amount of communication they need to transmit to each other. We consider the *public-coin* model, whereby Alice and Bob share a common source of randomness. (See the book by Kushilevitz and Nisan [KN97] for a comprehensive treatment.)

In the k-set disjointness problem, Alice and Bob receive two k-sets $x, y \in \binom{[n]}{k}$ and would like to determine if $x \cap y = \emptyset$ or not. Furthermore, they are guaranteed that either $x \cap y = \emptyset$ or $|x \cap y| = 1$. This problem is known to have communication complexity $\Theta(k)$. The upper bound is due to Håstad and Wigderson [HW07].

The lower bound was first established by Kalyanasundaram and Schnitger [KS92], and subsequent simplifications and generalizations of the proof were found by Razborov [Raz92] and Bar-Yossef et al. [BJKS04].

5.1.5. Theorem (Blais, Brody & Matulef [BBM11]). *Testing isomorphism to k-parities requires $\Omega(k)$ queries.*

Proof. Let k be even (a similar argument works for odd k). To solve a $k/2$-set disjointness problem with $2q$ bits of communication (and shared randomness), Alice and Bob can use a q-query solution for testing isomorphism to k-parities as indicated next. Alice forms the function $f = x^*$ and Bob forms the function $g = y^*$. Consider the function $f = (x \oplus y)^*$. Since $|x \oplus y| = |x| + |y| - 2|x \cap y|$, the function f is a k-parity if $x \cap y = \emptyset$, and a $(k-2)$-parity if $|x \cap y| = 1$. For every query $z \in \{0,1\}^n$ the testing algorithm makes, Alice and Bob can use two bits of communication to make sure they both know $h(z) = f(z) \oplus g(z)$. Then they both know which query the tester makes next, because it is determined by the shared randomness and the replies to previous queries. Hence they can simulate the tester of isomorphism. In particular they can find out whether h is a k-parity or a $(k-2)$-parity; equivalently, they can tell whether x and y intersect or not, under the assumptions made.

In the contrapositive form, what we have shown is that the $\Omega(k)$ lower bound for $k/2$-set disjointness also applies to telling the case where $h \in \mathsf{PAR}_k$ apart from the case where $h \in \mathsf{PAR}_{k-2}$, which is always possible by a tester of isomorphism to $(k-2)$-parities. □

5.1.6. Corollary. *The query complexity of testing isomorphism to k-parities is $O(k \log k)$ and $\Omega(k)$.*

Of interest to us in the rest of the chapter is what the situation looks like when we ask for non-adaptive algorithms. We will see that the query complexity becomes $\Theta(k \log k)$ in this case, and we conjecture that this bound remains valid in the adaptive case.

5.2 Background on group testing

The field of group testing[2] seeks efficient procedures to identify a set of defective items by performing "batch" tests, whereby a collection of items are tested together to determine if the batch contains a defective item or not. The subject originated during the Second World War as a means of detecting diseases in soldiers' blood samples [Dor43], without having to test each sample individually. The problem

[2]Despite its name, group testing is not concerned with property testing or group theory.

has since emerged time and again in a great many different contexts, theoretical as well as applied, including design theory, error-correcting codes, DNA screening, etc. A good textbook on the topic has been written by Du and Hwang [DH00]; somewhat related to our work are the papers [KS64, BGV05, PR08, INR10].

We formalize the problem as follows. We are given black-box access to a function $f\colon \{0,1\}^n \to \{0,1\}$ of the form

$$f(x_1, \ldots, x_n) = \bigvee_{i \in R} x_i,$$

for some unknown set R (the set of *relevant* variables of f). (These are the "defective items".) We abbreviate this as $f = \mathsf{OR}_R$ and also write

$$\mathsf{OR}_k = \bigcup_{R \subseteq [n], |R|=k} \{\mathsf{OR}_R\}$$

and

$$\mathsf{OR}_{\leq k} = \bigcup_{0 \leq k' \leq k} \{\mathsf{OR}_k\}.$$

One can regard queries to OR_R as questions of the form

$$\text{"Is } |X \cap R| = 0 \text{ or } |X \cap R| \geq 1\text{?"}$$

for any $X \subseteq [n]$. An upper bound k on the size of R is given, and the goal is to determine the elements of R while attempting to minimize the number of queries made.

Most prior work on group testing comes in two flavors, probabilistic and deterministic. The probabilistic view imposes a distribution on the location of the k relevant variables (see, e.g.,[Mac98]), while in the deterministic setting this is arbitrary. We work in the latter setting. A lot of effort has been put into optimizing the constants involved, but we will focus on asymptotics as usual.

In the adaptive case, tight bounds of $\Theta(k \log(n/k))$ can easily be obtained in various ways (we are assuming $k \leq n/2$, otherwise $\Theta(n)$ bounds are trivial). For example, for the upper bound one can query the sets $[1], [2], [4] \ldots$, until a relevant set has been found; then it is possible to perform a binary search for the index of the first relevant variable. The set R can thus be determined after k iterations of this process, and a somewhat careful analysis that we omit shows that this makes $O(k \log(n/k))$ queries. (Another approach would be to use the Winnow algorithm of Littlestone [Lit88] with random samples.) On the other hand, the lower bound is immediate from an information-theoretical argument, as there are $\binom{n}{k} = 2^{\Omega(k \log(n/k))}$ possible k-subsets of $[n]$.

Therefore, our discussion of standard group testing in this section is limited to non-adaptive algorithms. Associated with a deterministic, non-adaptive algorithm is a binary query matrix $M \in \mathcal{M}_{q \times n}$ whose rows $r_1, \ldots, r_q \in \{0,1\}^n$ are the

indicator vectors of each of the (non-adaptive) queries made. For two binary vectors $a, b \in \{0,1\}^n$, recall that $a \vee b$ represents their bitwise disjunction (which can be thought of as the union of two sets), and $a \wedge b$ represents their bitwise conjunction (the intersection of two sets). When $f = \mathsf{OR}_R$, the response vector

$$M \cdot R \triangleq (f(r_1), \ldots, f(r_q))^\top$$

is then simply the disjunction of the columns of M indexed by R, which can be interpreted as the boolean product of matrix M and a column vector R (the characteristic vector of the set of relevant variables), where $+, \cdot$ correspond to $\vee, \wedge$, respectively. (Note that this departs from our convention in the rest of the thesis that $a + b$ denotes addition over $\mathbb{F}_2^n$.) Observe that being able to determine R from $M \cdot R$ under the promise $|R| \leq k$ is equivalent to the following property being satisfied:

- the OR (union) of each set of $\leq k$ columns of M is unique, in the sense that $M \cdot R \neq M \cdot R'$ for any $R \neq R'$ with $|R|, |R'| \leq k$.

Such a matrix is called *k-separable*. Its columns are sometimes said to form a *uniquely decipherable code of order k*. This name reflects the connection, exploited by Kautz and Singleton [KS64], with certain kind of codes. Namely, the property above guarantees unique decodability of R from the response vector of the corresponding $\leq k$-OR, although the procedure may take as much as $\Omega(n^k)$ time. Most of the constructions of k-separable matrices possess a slightly stronger property that allows for quick decoding:

- the OR (union) of any set of $\leq k$ columns of M does not contain another column of M,

or equivalently

- every subset of $k+1$ columns of M contains a (permuted) $(k+1)$-sized identity submatrix.

Such a matrix M is called *k-strongly selective*, or *k-disjunct*. The set of its columns is also called a *k-cover-free family*, *k-union-free family*, or a *zero-false-drop code of order k* (more variations include "disjunctive codes" and "superimposed codes"). It is readily seen that any k-disjunct matrix is k-separable, and that any k-separable matrix is $(k-1)$-disjunct: if M is not $(k-1)$-disjunct, then it has a subset S of $k-1$ columns whose union includes the jth column for some $j \notin S$; but then $M \cdot S = M \cdot (S \cup \{j\})$, so M is not k-separable.

Given a k-disjunct matrix, an efficient decoding algorithm runs as follows. Observe that every negative answer ($f(r_i) = 0$) to a query r_i implies that no element of r_i (viewed as a subset of $[n]$) belongs to R as well. This allows us to label some columns as irrelevant, for each query with a negative answer. We can

safely discard these columns as no element of R will be removed in this way. The definition of k-disjunctness simply asserts that after going through the responses to all queries and discarding all such columns, we are left with precisely the columns associated with R. In short,

$$R = [n] \setminus \bigcup_{\substack{i \in [n] \\ OR_R(r_i)=0}} r_i$$

holds for any $R \in \binom{[n]}{k}$ (recall that we are identifying elements of $\{0,1\}^n$, such as r_i, with subsets of $[n]$).

Constructions are known for k-disjunct matrices where the number of rows (which translates into the number of queries) is $q = O(k^2 \log n)$ [KS64, BV03]. The best known lower bounds are $q = \Omega(k^2 \log n / \log k)$ [Für96, EFF82, Rus94]; the proof can also be found in [Juk11, p. 116].

5.3 Relaxed group testing: adaptive

We study the following relaxation of the group testing problem. Instead of trying to find R itself, suppose we only want to distinguish between the cases $|R| \leq k$ or $|R| > k$. We assume $k \leq n/2$ because tight bounds are easy otherwise. This relaxation does not appear to help much if we restrict ourselves to deterministic algorithms (see below), but the situation looks vastly different if randomization is allowed. As usual we consider algorithms with success probability, say, 2/3.

A straightforward $O(k \log k)$ adaptive upper bound for relaxed group testing follows by using the random partitions introduced in Definition 3.4.1: start from a random partition of $[n]$ into $\sim k^2$ buckets and then simulate the deterministic, adaptive group testing upper bound for $n = O(k^2)$, where single variables have been replaced with buckets.

By now the astute reader will have noted a striking resemblance between group testing and junta testing. All the testers we discussed in Section 3.6 used independence tests as a building block, *and this is the only way they accessed f* (except for the random choices of x and y with $f(x) \neq f(y)$). Suppose $f\colon \{0,1\}^n \to \{0,1\}$ is a junta with set R of relevant variables. With f we can associate the function $f' = \mathsf{OR}_R$ that takes the union of the bits in R. An independence test of f on A is roughly the same as an OR query for f on A, except that sometimes A may be incorrectly reported as non-influential by an independence test. So by replacing independence tests with calls to f', the one-sided junta testers immediately yield algorithms that accept if $f \in \mathsf{OR}_{\leq k}$. The rejection conditions being different, it would appear that there is no guarantee that they reject if $f = OR_R$ where $|R| > k$, but they actually do.

For example, take the adaptive junta tester from Section 3.6. As long as k or fewer relevant blocks have been detected, it conducts an influence test on the rest,

and if the rest is found to be relevant, it binary searches for one more additional block. If this process is repeated $k+1$ times, then it will find $k+1$ relevant blocks for any OR of more than k variables, provided the initial random partition separates at least $k+1$ of the elements of R, which occurs with high probability. Thus we arrive at a second $O(k \log k)$ adaptive solution for relaxed group testing; in fact, the algorithm finds a k-isolating partition for R (see Definition 3.4.1).

An $\Omega(k)$ adaptive lower bound can be proven for relaxed group testing in exactly the same way as for parities (Section 5.1.1). To see this, observe that to solve a set-disjointness problem, Alice can form the function $f = OR_x$ and Bob the function $g = OR_y$, and then simulate queries to $h \triangleq f \vee g$. (Of course, the same kind of argument works for ANDs just as well as it does for XORs and ORs.) All in all we have the following.

5.3.1. Theorem. *The adaptive query complexity of the relaxed group testing problem with parameter $k \leq n/2$ is $\Omega(k)$ and $O(k \log k)$.*

5.3.2. Remark. If we demand deterministic algorithms (which are always correct), then the adaptive complexity of the problem becomes $\Theta\big(\log \binom{n}{k}\big)$. The lower bound uses the same technique, since the reduction above then gives a deterministic protocol for the $k/2$-set disjointness problem, the communication complexity of which is $\lceil \log \binom{n}{k/2} \rceil = \Theta(\log \binom{n}{k})$ because of the rank bound [KN97, Example 2.12].

5.3.1 Interlude: a 3-way variant

Interestingly, were we endowed with the power to perform a somewhat stronger test of relevance, the problem would admit faster adaptive solutions. Namely, suppose that querying for a set A allowed us to distinguish with certainty from among the three cases $|A \cap R| = 0$, $|A \cap R| = 1$ and $|A \cap R| \geq 2$, instead of only the first from the other two. Then we claim that the problem could be solved with $O(k)$ queries, which gets rid of the $\log k$ factor. One notable consequence of this fact is that many lower bound techniques with an information-theoretic flavor cannot go beyond $\Omega(k)$ for this problem, as there is a randomized ternary decision tree with $O(k)$ depth that solves it.

To see this, observe that there is nothing to do if the whole set $[n]$ is not relevant, so assume it is. We keep a partition $\mathcal{P}$ of $[n]$ such that each element of $\mathcal{P}$ intersects R. Initially, $\mathcal{P}$ is the singleton $\{[n]\}$. At each step we take an arbitrary $A \in \mathcal{P}$ such that $|A \cap R| \geq 2$ (if there is one), and take a random partition of A into two sets A_1, A_2. With probability no less than $1/2$, at least one relevant index in A lands into each of A_1, A_2; otherwise we draw another random pair (this condition can be tested with two queries to the function). When we succeed in splitting A into two relevant sets A_1, A_2, we replace A with A_1 and A_2 in the partition to form $\mathcal{P}_{next} = \mathcal{P}_{prev} - \{A\} \cup \{A_1, A_2\}$, and go back to picking another element of $\mathcal{P} = \mathcal{P}_{next}$ if possible.

Intuitively, the reason why this works is that if we know for sure that $|A \cap R| \geq 2$, then we know that it pays to persist in attempting to split A: we will succeed promptly. Notice that if we have a test for $|A \cap R| \geq 2$ that works only with high constant probability, then $O(\log k)$ iterations decrease the error probability of each such test to $O(1/k)$ (this is necessary to handle those A with $|A \cap R| = 1$). This leads to a third constant-success probability solution to the relaxed group testing problem. Again, it has complexity $O(k \log k)$.

An unexpected application of these ideas, modulo some tailoring, will be exhibited in Chapter 9 on cycle finding.

5.4 Relaxed group testing: non-adaptive

For non-adaptive testers, these algorithmic techniques do not seem to work. Nevertheless, we show in the following section that, somewhat surprisingly, the task can still be accomplished non-adaptively with $O(k \log k)$ queries (Theorem 5.4.7). (Note that when $n = O(k^2)$, the best k-disjunct matrices have size between $O(k^2 \log k)$ and $\Omega(k^2)$, so applying the non-adaptive group testers to a random partition results in much worse query complexities.) Additionally, the algorithm can handle random errors in the response vector, in the form of false negatives. This sets the stage for applications to property testing (Section 5.5).

We can show that $\Omega(k \log k)$ non-adaptive queries are necessary (Theorem 5.4.1 and its corollary). This applies even to the noiseless setting, thus showing that our non-adaptive algorithm is optimal. It is also easy to derive from it an $\Omega(k \log k)$ lower bound for testing k-juntas non-adaptively. We should also point out that our upper bound for relaxed group testing actually *finds* k blocks each containing at most one relevant variable; it is also possible to argue that each of these blocks has size roughly n/k. Once we know a block has one relevant variable, we can use standard group testing with $k = 1$ to find the variable itself with $O(1 \log(n/1))$ non-adaptive queries. Hence our result also gives a 2-stage *randomized* solution to the standard group testing problem with complexity $O(k \log k + k \log(n/k)) = O(k \log n)$. However, this is already known, even deterministically (see [BGV05, BV06, MT11], which are based on [CGR02]).

5.4.1 Lower bound

The first observation we need for the non-adaptive lower bound is the following. In Section 5.3 we showed how to use a q-query solution to relaxed group testing to yield a solution to set disjointness with $2q$ bits of communication. But note that if the algorithm is non-adaptive, then we can obtain a solution to set disjointness with only $q + 1$ bits of communication, all of which directed from Alice to Bob. This is because there is no need for both parties to know $h(z)$, since the whole set of queries is determined already by the common source of randomness and

each of them knows what comes next. Bob can perform all the computations and figure out the answer. Hence it is enough to prove lower bounds for the **one-way** communication complexity of the set disjointness problem.

5.4.1. Theorem (de Wolf [Wol06]). *The one-way communication complexity of the k-set disjointness problem is $\Theta(k \log k)$ for $k = O(\sqrt{n}\,)$, and $\Theta(k \log(n/k))$ for $\Omega(\sqrt{n}\,) \leq k \leq n/2$.*

(It is $\Theta(n)$ for $k \geq n/2$.)

Proof. The upper bound follows from the relaxed group testing upper bounds, so we only need to prove the first part. We assume for simplicity that k is a power of two and divides n.

First consider the case $k \leq \sqrt{n/2}$. Let x be Alice's n-bit input. For Alice we restrict our attention to inputs of a particular structure. Namely, consider partition of $[n]$ into k consecutive sets of size $n/k \geq 2k$. The inputs we allow contain precisely a bit set to one inside each block of the partition, and moreover the offset of the unique index set to one within the ith block is a number between 0 and $2k-1$, inclusive. In this case, x describes a message M of k integers $m_1, \ldots, m_k$ in the interval $\{0, \ldots, 2k-1\}$. This is an m-bit long message, where $m = k \log(2k)$. We can write Alice's input as $x = u(m_1) \ldots u(m_k)$, where $u(m_i) \in \{0,1\}^{n/k}$ is the unary expression of the number m_i using n/k bits (where the rightmost $n/k - k$ bits of each $u(m_i)$ are always zero). For instance, the picture below illustrates the case where $n = 40$, $k = 4$, and $M = (1, 7, 0, 5)$:

$$x = \overbrace{\underbrace{0100000000}_{u(m_1)}}^{n/k}\ \overbrace{\underbrace{0000000100}_{u(m_2)}}^{n/k}\ \overbrace{\underbrace{1000000000}_{u(m_3)}}^{n/k}\ \overbrace{\underbrace{0000010000}_{u(m_4)}}^{n/k}$$

Let ρ_x be the q-bit message that Alice sends on this input. Below we show that the message is a *random-access code* for M, i.e., it allows a user to recover each bit of M with probability at least $1 - \delta$ (though not necessarily *all* bits of M simultaneously). Then our lower bound will follow from Nayak's random-access code lower bound [Nay99]. This says that

$$q \geq (1 - H(\delta))m,$$

where δ is the error probability of the protocol and $H(\delta)$ is its entropy.

Suppose Bob is given ρ_x and wants to recover some bit of x. Say this bit is the ℓth bit of the binary expansion of m_i. Then Bob completes the protocol using the following y: y is 0 everywhere except on the k bits in the ith block of size n/k whose offsets j (measured from the start of the block) satisfy the following: $0 \leq j < 2k$ and the ℓth bit of the binary expansion of j is 1.

Recall that Alice has a 1 in block i only at position m_i. Hence x and y will intersect iff the ℓth bit of the binary expansion of m_i is 1, and moreover, the

size of the intersection is either 0 or 1. Also, the Hamming weight of y is k by definition. Running the k-set disjointness protocol with confidence $1-\delta$ will now give Bob the sought-for bit of M with probability at least $1-\delta$, which shows that ρ_x is a random-access code for M.

If $k > \sqrt{n/2}$ then we can do basically the same proof, except that the integers m_i are now in the interval $\{0,\dots,n/k-1\}$, $m = k\log(n/k)$, and Bob puts only $n/2k < k$ ones in the ith block of y (he can put his remaining $k - n/2k$ indices somewhere at the end of the block, at an agreed place where Alice won't put 1s). This gives a lower bound of $\Omega(k\log(n/k)) = \Omega\big(\log\binom{n}{k}\big)$. □

5.4.2. Remark. The lower bound holds even for *quantum* one-way communication complexity. The proof remains intact except that if we wish to allow Alice and Bob to share entanglement, the random-access code bound needs to be replaced with Klauck's [Kla00] version, which is weaker by a factor of two.

5.4.3. Corollary. *Let $k \le \sqrt{n}$. At least $\Omega(k\log k)$ queries are needed for non-adaptive solutions to the following problems: relaxed group testing with parameter k, testing k-juntas and testing isomorphism to k-parities.*

5.4.2 Upper bound

Here we present an algorithm to solve the relaxed group testing problem in the presence of one-sided noise of some constant rate $\eta > 0$. Let $x \in \{0,1\}^n$ and $f = \mathsf{OR}_x$. Suppose that, instead of querying f, we can only query a "noisy version" $\tilde{f}$ of f. Let $\tilde{f}$ denote a function on $\{0,1\}^n$ whose image on any input is a $0-1$ random variable, and the variables $\{f(x)\}_{x\in\{0,1\}^n}$ are independent. Then $\tilde{f}$ is called a *η-noisy oracle* for f if the following two properties are satisfied:

1. $f(x) = 0$ implies $\tilde{f}(x) = 0$;
2. $f(x) = 1$ implies $\Pr[\tilde{f}(x) = 1] \ge 1-\eta$.

Different calls to $\tilde{f}'(x)$ for the same x return independent copies of the same random variable, may yield different values.

Recall that we denote the boolean product of a $q\times n$ matrix M and $x \in \{0,1\}^n$ by $M\cdot x$. When $f = \mathsf{OR}_x$, the response vector $a \in \{0,1\}^q$ under f will be $a = M\cdot x$. One equivalent way of modelling the response vector $\tilde{a}$ under a η-noisy oracle $\tilde{f}$ for f as defined above is as follows. Let $e \in \{0,1\}^q$ be a sequence q of independent $0-1$ variables such that $\Pr[e_i = 1] \ge 1-\eta$ and let $\tilde{a} = a \wedge e$. Our goal is to design M so as to allow, with high probability, determining if $|x| = k$ given $\tilde{a}$. We call this the *noisy relaxed group testing problem.*

We introduce the following variation of the notion of k-disjunct matrix.

5.4.4. Definition. A $q \times n$ binary matrix M is (δ, ζ)-*approximately k-disjunct* if, with probability at least $1 - \delta$, the union of a random k-subset of the columns of M and a vector in $\{0,1\}^q$ of weight at most ζq $\{0,1\}^q$ does not contain any other column of M.

The reader may verify that (δ, ζ)-approximate k-disjunctness implies (δ, ζ)-approximate $(k-1)$-disjunctness as well. The property can be rewritten as

$$(M \cdot x) \vee y \not\supseteq M \cdot z$$

with probability $1 - \delta$, for random $x \in \{0,1\}^n$ with $|x| = k$ (or $|x| \leq k$) and any $y, z \in \{0,1\}^q$ with $|y| \leq \zeta q$ and $z \subseteq [n] \setminus x$, $|z| = 1$.

We need the following lemma.

5.4.5. Lemma. *Let $n = O(k^2)$ and R be a k-subset of $[n]$. Let V be an $(60 \log k) \times n$ random matrix with entries in $[10]$.*

For every column index $j \in [n] \setminus R$, let X_j be the number of row indices $i \in [60 \log k]$ for which there exists another column $j' \in R$ with $V_{ij} = V_{ij'}$. Then

$$\Pr_V [\forall j \in [n] \setminus R : X_j < 15 \log k] = 1 - o(1).$$

Proof. Let $j \in [n] \setminus R$. For each $i \in [60 \log k]$,

$$\Pr [\forall j' \in R : V_{ij} \neq V_{ij'}] = (1 - 1/(10k))^k \geq 9/10.$$

Hence X_j is the sum of $60 \log k$ independent binary random variables with expectation $\leq 1/10$, and by the Chernoff bounds

$$\Pr [X_j \geq 15 \log k] \leq \exp(-(15/6 - 1)^2 (1/10) 60 \log k / 3) = o(k^{-2}).$$

The claim follows from the union bound over all $n - k = O(k^2)$ elements of $[n] \setminus R$. □

5.4.6. Corollary. *Let $n = O(k^2)$. There is a $(1 - o(1), 3/32)$-approximately k-disjunct matrix $M \in \mathcal{M}_{(600k \log k) \times n}$. Furthermore, each of the columns of M has Hamming weight $60 \log k$.*

Proof. We use the notation of Lemma 5.4.5. Let V be a matrix such that

$$\Pr_R [\forall j \in [n] \setminus R : X_j < 15 \log k] = 1 - o(1);$$

such V can be shown to exist by the said lemma and a straightforward averaging argument. Let V represent a code of n words of length $60 \log k$ over an $10k$-sized alphabet. Concatenate it with the "trivial" code mapping each symbol $x \in [10k]$ to the string $0^{x-1}10^{10k-x} \in \{0,1\}^{10k}$ and let M be an $(600k \log k) \times n$ matrix

whose columns are the words of the concatenated code. The claim follows by observing that

$$X_j = \left|\{\tilde{i} \in [600 \log k] \mid M_{\tilde{i}j} = 1 \wedge (\exists j' \in R \mid M_{\tilde{i}j} = 1)\}\right|.$$

□

5.4.7. THEOREM. *There is a non-adaptive algorithm making $O(k \log k)$ queries that solves the noisy relaxed testing problem with noise rate $\eta = 1/2$. It succeeds with high constant probability and has two-sided error.*

Proof. Let $\ell = 5(k+1)^2$. First we show how to reduce the general case to the case $n < \ell$. When $n \geq \ell$ the algorithm proceeds as follows:

1. Partition $[n]$ into ℓ buckets $I_1, \ldots, I_\ell$ at random.
2. Define the function REPLICATE: $\{0,1\}^\ell \to \{0,1\}^n$ by mapping $x \in \{0,1\}^\ell$ to the string $y \in \{0,1\}^n$ that is constant inside each bucket and takes value x_i on each element of I_i.
3. Solve the relaxed noisy group testing problem for the composition of f and REPLICATE.

Let $f = \mathsf{OR}_x$ for some $x \subseteq [n]$. Then $f \circ$ REPLICATE will be of the form $\mathsf{OR}_{x'}$ for some subset $x' \subseteq [\ell]$. The size of x' will always be at most that of x; in particular it will be bounded by k whenever $|x| \leq k$. When $|x| \geq k+1$, standard arguments based on the birthday paradox yield $|x'| \geq k+1$ with probability at least $9/10$. So the reduction does indeed work with high constant probability.

Let us deal first with the case $n = \ell$ (note we can always add irrelevant variables if $n < \ell$). We do the following.

1. Obtain a $(600k \log k) \times \ell$ matrix M as in Corollary 5.4.6.
2. Randomly permute the columns of M to obtain M^π.
3. For each row r_i of M^π, query f on r_i; let $\tilde{a}_i$ be the answer.
4. Compute $X_j \triangleq \left|\{i \in [600k \log k] \mid M^\pi_{ij} = 1 \wedge \tilde{a}_i = 1\}\right|$ for each column index $j \in [\ell]$.
5. Let $\tilde{x} \triangleq \{j \in [\ell] \mid X'_j \geq 15 \log k\}$.
6. Accept iff $|\tilde{x}| \leq k$ (and if so, return the nonzero indices of $\tilde{x}$).

The fact that a random permutation is being applied to each of the rows of M enables us to analyze x as a uniformly random $|x|$-set, while leaving M unpermuted. Because of the one-sided nature of the error, $\tilde{a}_i = 1$ implies $r_i \cap x \neq 0$. It is then immediate from the definition of approximate k-disjunct matrices that, with probability $1 - o(1)$ over x of size k, for any $j \in [\ell] \setminus x$, it holds that $X'_j < 15 \log k$ and hence $j \notin \tilde{x}$. This shows k-ORs are accepted with high probability.

We claim now that, for *any* $x \in [\ell]$, and with probability $1 - o(1)$, $x \subseteq \tilde{x}$ holds; in particular the algorithm rejects k'-ORs when $k' > k$. Indeed, each column of M has $60 \log k$ nonzero entries. For every $j \in x$ and row i containing a nonzero entry (that is, $M_{ij} = 1$), $\Pr[\tilde{a}_i = 1] \geq 1 - \eta \geq 1/2$. By the Chernoff bound, the probability that the number of i with $M_{ij} = 1$ and $\tilde{a}_i = 1$ is smaller than $15 \log k$ is bounded by $\exp(-(1/2)^2 30 \log k/2) = o(k^2)$. It follows by the union bound over all $j \in x$ that $x \subseteq \tilde{x}$, proving our claim. □

This procedure also can be used to distinguish k-ORs from any k'-OR with $k' \neq k$, whether $k' > k$ or $k' < k$.

5.5 Strong k-juntas

Recall that an independence test for $S \subseteq [n]$ is performed as follows. Pick $\ln(1/\eta)/\delta$ random pairs $x, y \in \{0,1\}^n$ conditioned on $x_{\restriction_{[n]\setminus S}} = y_{\restriction_{[n]\setminus S}}$, and return $\tilde{f}'(S) = 1$ if $f(x) \neq f(y)$ for some pair (x, y), or else return $\tilde{f}'(S) = 0$. Independence tests have one-sided error, so all errors are in the form of false negatives, i.e., the reported value $\tilde{f}'(x)$ is 0 but the true value of $f'(x)$ is 1. In fact it is clear that

1. $\mathrm{Inf}_f(S) = 0$ implies $\tilde{f}'(S) = 0$;
2. $\mathrm{Inf}_f(S) \geq \delta$ implies $\Pr[\tilde{f}'(S) = 1] \geq 1 - \eta$.

We call such $\tilde{f}'$ a *η-noisy relevance oracle for f*. As we saw, all known testers for k-juntas can be viewed as making queries to such a relevance oracle.

Observe that the second proviso is met whenever S contains a variable of influence $\geq \delta$, because influence is monotone. In fact, if f has a subset S of least k variables of influence $\geq \delta$ each, then $\tilde{f}$ is a η-noisy oracle for OR_S as defined in the previous section. (The oracle makes $O(\log(1/\eta)/\delta)$ queries to the function f.) This motivates the following definition:

5.5.1. Definition. A δ-strong k-junta is a function $f\colon \{0,1\}^n \to \{0,1\}$ with k variables of influence $\geq \delta$ and $n - k$ variables of influence 0.

Up to a factor of at most 2 in δ, this is essentially the same as a k-junta which is δ-far from all juntas on fewer than k variables (Lemma 3.6.1). It may be worthwhile to note that any k-junta with Fourier degree d is a $2^{-(d+1)}$-strong k-junta by a result of Nisan and Szegedy [NS92].)

As usual, when we speak of a δ-strong $\leq k$-junta, we mean a δ-strong k'-junta for some $k' \leq k$.

As a consequence of the discussion above we obtain the following result: there is an algorithm making $O(\frac{1}{\delta}k \log k)$ non-adaptive queries that, with high probability, accepts δ-strong $\leq k$-juntas and rejects functions with at least $k+1$ variables of influence $\geq \delta$. We can do better however:

5.5.2. THEOREM. *There is a non-adaptive $O\left(\frac{k \log k}{\min(\delta,\varepsilon)}\right)$-query tester that, with high probability, accepts δ-strong $\leq k$-juntas and rejects functions that are ε-far from all k-juntas.*

Roughly speaking, the key feature of strong k-juntas that allows simpler testing algorithms is that every relevant variable can be easily identified, eliminating the need to tackle sets of variables that account for a noticeable fraction of the function's overall influence despite each variable having individually low influence. This allows us to strengthen (and simplify) the acceptance conditions.

Proof. Suppose for the moment that $n = O(k^2)$ (so no partitioning is necessary). As we saw, it is possible to simulate a query to a (1/2)-noisy relevance oracle $\tilde{f}'$ for f with $O(1/\delta)$ queries to a δ-strong k-junta f. Then we apply the test of Theorem 5.4.7 to solve the noisy group testing problem with $O(k \log k)$ queries to $\tilde{f}'$ and reject if it rejects. This makes $O(k \log k/\delta)$ queries to f. Let $S \subseteq [\ell]$, $|S| \leq k$ be the set of relevant variables returned. If f is a δ-strong k-junta on R, then with high probability $R = S$, $f' = \mathsf{OR}_S$, and this first step of our algorithm accepts.

The second part is designed to make the algorithm reject when f is ε-far from a k-junta, in which case we have $\mathrm{Inf}_f([n] \setminus S) \geq \varepsilon$. (Of course, we also need to ensure that δ-strong k-juntas are accepted, but this will follow trivially from the fact that $\mathrm{Inf}([n] \setminus S) = 0$ then.) Let $A = [n] \setminus S$. We construct a (multi)set $T \subseteq \{0,1\}^n$ of $100ek \log k/\varepsilon$ inputs with density $1/k$, i.e., each of the queries B is drawn from $\subseteq_{1/k} [n]$ as in Lemma 3.6.4. For each $B \in T$, we query two random strings $x^B, y^B \in \{0,1\}^n$ conditioned on $x^B\restriction_{[n]\setminus B} = y^B\restriction_{[n]\setminus B}$ and reject if there is some B disjoint with S for which $f(x^B) \neq f(y^B)$.

We expect $100k \log k/\varepsilon$ of the elements of T to be all zero inside S, and by Chernoff bounds, with probability $1 - o(1)$ there are at least $96k \log k/\varepsilon$ like that. (We assume $k = \omega(1)$ or ist at least large enough; otherwise the theorem is easy.) Let $Q = \{B \in T \mid B\restriction_S = 0\}$ be this multiset of size $\geq 96k \log k/\varepsilon$. Note that the distribution of $B \subseteq_\rho [n]$ conditioned on $B\restriction_S = 0$ is exactly the same as the distribution $\subseteq_\rho [n] \setminus S$. Therefore the distribution of each element of Q follows exactly $\subseteq_{1/k} A$.

When taking B, x, y conditioned on $B\restriction_S = 0$ and $x^B\restriction_{[n]\setminus B} = y^B\restriction_{[n]\setminus B}$, note that the probability that $f(x^B) \neq f(y^B)$ is exactly equal to $\mathbb{E}_{B \subseteq_{1/k} [n]\setminus S} \mathrm{Inf}_f(B)$.

Therefore, by Lemma 3.6.5, if f is ε-far from being a junta on S, then each independence test associated with an element of B rejects with probability at least ε/k. Since $|B| = \Omega(k \log k/\varepsilon)$, in this case we reject with probability $1 - o(1)$, and we are done.

For larger n, the argument is the same except that we start with a random partition of n into $\ell = O(k^2)$ blocks and make blockwise-constant queries of density $1/k$. From then on we consider functions defined on ℓ-bit inputs, as in Chapter 3; there we used $\ell = O(k^9/\varepsilon)$, but a recent result of Blais et al. [BWY11] shows that we can in fact take $\ell = O(k^2)$. □

One consequence of the proof is that if f is a δ-strong k'-junta (with $k' \leq k$), we return k' blocks that isolate the relevant variables of f. Hence by using the techniques of Chapter 3 we can test isomorphism to to $\Omega(1)$-strong k-juntas with $O(k \log k/\varepsilon)$ non-adaptive queries.

However, it seems that the following recent result may yield non-adaptive testers of isomorphism to any function that make $O(k \log k/\varepsilon)$ queries.

5.5.3. Theorem (Ron & Tsur [RT11, Theorem 3.1]). *There is a non-adaptive algorithm that makes $O(k/\varepsilon)$ queries, accepts k-juntas and rejects functions far from $2k$-juntas (with high probability).*

The reason is that to be able to apply the techniques of Chapter 3, we do not need to reject functions that are far from k-juntas, as the junta testers do. We can get by rejecting functions that are far from $2k$-juntas, which is possible by Theorem 5.5.3. One can then relate in a similar way the distance from f to being a $2k$-junta to the distance to the "blockwise" version of f to being a $2k$-junta. However, the resulting testers would have two-sided error.

5.6 Parities and SMP complexity

There is a recent paper of Leung, Li, and Zhang [LLZ11] whose main result follows immediately from our $O(k \log k)$ non-adaptive upper bound for parities, and in fact is improved by it.

The problem is about communication complexity: Alice has a string $x \in \{0,1\}^n$, Bob has $y \in \{0,1\}^n$, and they want to compute some function f that depends only on $|x \oplus y|$, i.e., $f(x \sqcup y) = D(|x \oplus y|)$, for some known $D\colon [n] \to \{0,1\}$. They say such an f is a *symmetric XOR function*. The model is the Simultaneous Message Passing (SMP) model, which means Alice and Bob can't talk to each other; rather, they send messages to a referee, who computes the answer at the end.

Suppose first we are promised that $|x \oplus y| \leq k$. They proceed to give an $O(k \log^3 k/\log\log k)$ upper bound for the problem of finding out $|x \oplus y|$. It is not hard to see, however, that our non-adaptive algorithm for finding the size

of a XOR gives an $O(k \log k)$ solution. Take $Q = \{p_1, p_2, \ldots, p_q\}$ to be the set of $q = O(k \log k)$ random queries used to find the size of a parity, provided it is at most k. Then Alice sends the bits $\langle x, p_1\rangle, \langle x, p_2\rangle, ..., \langle x, p_q\rangle$ (where the inner products are taken modulo 2), and Bob sends $\langle y, p_1\rangle, \langle y, p_2\rangle, ..., \langle y, p_q\rangle$. The referee then computes from this $\langle x \oplus y, p_1\rangle, \langle x \oplus y, p_2\rangle, ..., \langle x \oplus y, p_q\rangle$, and this is enough to compute the size of $x \oplus y$, if it is $\leq k$.

They state the main result, when k is not known, in terms of a different parameter r, but in the paper it follows from a reduction to the previous problem with $r \leq k$. Using this reduction we obtain the following.

5.6.1. THEOREM. *Define r_0 and r_1 to be the minimum integers such that $r_0, r_1 \leq n/2$ and $D(k) = D(k+2)$ for all $k \in [r_0, n - r_1]$; set $r = \max(r_0, r_1)$ and $f(x \sqcup y) = D(|x \oplus y|)$ by a symmetric XOR function f.*

Then the randomized public-coin SMP complexity of computing f is $O(r \log r)$.

5.7 Summary

We have seen that several related problems, such as testing isomorphism to k-parities and some variations of group testing, have complexity $O(k \log k)$ and $\Omega(k)$. We also showed that the query complexity becomes $\Theta(k \log k)$ if non-adaptivity is required.

Chapter 6

Testing by implicit learning

Now we are ready to apply the techniques developed in previous chapters to a number of related property testing problems that fit into the "testing by implicit learning" framework of Diakonikolas et al. [DLM+07]. In particular, for some of these problems (including testing s-term DNF formulae, size-s decision trees, size-s Boolean formulae, s-sparse polynomials over $\mathbb{F}_2$, and size-s branching programs), we are able to improve on the query complexity of the best known testers by using sample extractors. It is worthy of note that our methods can lead to testers that have better query complexity than those tailored to a specific problem, such as the tester of Parnas et al. [PRS02] for the class of monotone s-term DNF formulae. We also establish new lower bounds for some of these problems.

The content of this chapter is based on the paper

- S. Chakraborty, D. García–Soriano, and A. Matsliah. Efficient sample extractors for juntas with applications. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 545–556, 2011.

6.1 Introduction

Suppose we wish to test for the property defined by a class $\mathcal{C}$ of Boolean functions over $\{0,1\}^n$; that is, we aim to distinguish the case $f \in \mathcal{C}$ from the case $\text{dist}(f,\mathcal{C}) \geq \varepsilon$. The class is parameterized by a "size" parameter s (e.g., the class of DNFs with s terms, or circuits of size s) and, as usual, our goal is to minimize the number of queries made to f. In particular we strive for query complexity independent of n whenever possible.

The main observation underlying the "testing by implicit learning" paradigm of Diakonikolas et al. [DLM+07] (see also [Ser11, DLM+08, GOS+09]) is that a large number of interesting classes $\mathcal{C}$ can be well approximated by (relatively) small juntas also belonging to $\mathcal{C}$. We explain these ideas by rephrasing them in term of testing approximate isomorphism against certain subsets of juntas.

The prototypical example is obtained by taking for $\mathcal{C}$ the class of s-term DNFs. Let $\tau > 0$ be an approximation parameter (which for our purpose should be thought of as polynomial in ε/s). Any DNF term involving more than $\log(s/\tau)$ variables may be removed from f, affecting only a τ/s fraction of its values; hence removing all of them results in an s-term DNF f' that is τ-close to f and *depends on only* $s\log(s/\tau)$ *variables* (equivalently, f' is a $s\log(s/\tau)$-junta). Let $\mathrm{Jun}_{[k]}$ denote the subset of (k-junta) functions $\{0,1\}^n \to \{0,1\}$ that depend only on the *first* k variables. Since the class $\mathcal{C}$ is permutation-invariant (closed under permutations of the variables), the foregoing observation can be rephrased as follows: for any $k \geq s\log(s/\tau)$, the subclass $\mathcal{C}_{[k]} \triangleq \mathcal{C} \cap \mathrm{Jun}_{[k]}$ is such that every $f \in \mathcal{C}$ is τ-close to being isomorphic to some $g \in \mathcal{C}_{[k]}$ (in short, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$).

On the other hand, for every f such that $\mathrm{dist}(f,\mathcal{C}) = \mathrm{distiso}(f,\mathcal{C}) \geq \varepsilon$ it also holds that $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \varepsilon$, since $\mathcal{C}_{[k]} \subseteq \mathcal{C}$. Hence, to solve the original problem, all we need is to differentiate the two cases (i) $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$ and (ii) $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \varepsilon$.

Let us denote by f^* the k-junta that is closest to f; f^* can be identified with its core, the Boolean function $\mathrm{core}_k(f^*)\colon \{0,1\}^k \to \{0,1\}$ obtained from f^* by dropping its irrelevant variables. We saw in Chapter 3 that, by getting random samples of the form $(x, \mathrm{core}_k(f^*)(x)) \in \{0,1\}^k \times \{0,1\}$, we can use standard learning algorithms to identify an element $g \in \mathcal{C}_{[k]}$ which is close to being isomorphic to f^* (if any), which essentially allows us to differentiate between the aforementioned cases. The number of such samples required for this is roughly logarithmic in $|\mathcal{C}_{[k]}|$.[1] An important observation is that the size of $\mathcal{C}_{[k]} \triangleq \mathcal{C} \cap \mathrm{Jun}_{[k]}$ is usually very small, even compared to the size of $\mathrm{Jun}_{[k]}$, which is 2^{2^k}. For instance, it is not hard to see that for the case of s-term DNFs, the size of $\mathcal{C}_{[k]}$ is bounded by $(2k)^k$, which is exponential in $k = s\log(s/\tau)$, rather than doubly exponential.

It is in the way to obtain these samples that our approach departs from that of [DLM+07]. We mention next the two main differences that, when combined together, lead to better query complexity bounds. The first difference is in the junta-testing part; both algorithms start with a junta tester to identify k disjoint subsets of variables (blocks), such that every "influential" variable of the function f being tested lies in one of these blocks. While [DLM+07] use the non-adaptive junta tester, we switch to the query-efficient adaptive junta tester. The key is the fact, proven in Section 3.7.1, that the tester of Blais is sufficiently tolerant (the level of tolerance of the tester determines how large τ can be, which in turn determines how small k can be). The second (and the main) difference is in sample extraction—the actual process that obtains samples from the core of f^*. In [DLM+07] sampling is achieved via independence tests with $\mathrm{poly}(k)$ queries, whereas we saw in Chapter 3 how to accomplish this task by making just *one* query to f (after some preprocessing).

[1] Issues of computational efficiency are usually disregarded here; however see [DLM+08].

6.2 Notation

Given a function $f\colon \{0,1\}^n \to \{0,1\}$, we denote by $f^*\colon \{0,1\}^n \to \{0,1\}$ the k-junta that is closest to f (if there are several k-juntas that are equally close, break ties using some arbitrarily fixed scheme). Clearly, if f is itself a k-junta then $f^* = f$.

Unless explicitly mentioned otherwise, $\mathcal{C}$ will always denote a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ that is closed under permutation of variables; that is, for any f and permutation π of $[n]$, $f \in \mathcal{C}$ if and only if $f^\pi \in \mathcal{C}$. For any $k \in \mathbb{N}$, let $\mathcal{C}_{[k]}$ denote the subclass $\mathcal{C} \cap \mathrm{Jun}_{[k]}$. Note that since $\mathcal{C}$ is closed under permutations of variables, $\mathcal{C}_{[k]}$ is closed under permutations of the first k variables.

6.3 Upper bounds

Coupled with the prior discussion on testing by implicit learning, Theorem 3.5.6 implies:

6.3.1. Corollary. *Let $\varepsilon > 0$ and let $\mathcal{C}$ be an permutation-invariant class of Boolean functions. In addition, let $k \in \mathbb{N}$ be such that for every $f \in \mathcal{C}$,* $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \theta_{3.5.6}(k, \varepsilon)$. *Then there is an algorithm that makes*

$$O\left(\frac{k(\log k + 1) + \log |\mathcal{C}_{[k]}|}{\varepsilon}\right)$$

queries and satisfies:

- *if $f \in \mathcal{C}$, it accepts with probability at least $7/10$;*
- *if* $\mathrm{dist}(f, \mathcal{C}) \geq \varepsilon$, *it rejects with probability at least $7/10$.*

To minimize the query complexity, we would like to pick k as small as possible, subject to the requirement of the theorem. Let $k^\star(\mathcal{C}, \tau)$ be the smallest $k \in \mathbb{N}$ such that for every $f \in \mathcal{C}$, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$; intuitively, this condition means that $\mathcal{C}$ is τ-approximated by $\mathcal{C}_{[k]}$. We take from [DLM+07] the bounds on $k^\star = k^\star(\mathcal{C}, \tau)$ and $|\mathcal{C}_{[k^\star]}|$ for the following classes of functions:

	$\mathcal{C}$ (class)	$k^\star \triangleq k^\star(\mathcal{C}, \tau) \leq$	$\lvert\mathcal{C}_{[k^\star]}\rvert \leq$
1	s-term DNFs	$s \log(s/\tau)$	$(2s \log(s/\tau))^{s \log(s/\tau))}$
2	size-s Boolean formulae	$s \log(s/\tau)$	$(2s \log(s/\tau))^{s \log(s/\tau)+s}$
3	size-s Boolean circuits	$s \log(s/\tau)$	2^{2s^2+4s}
4	s-sparse polynomials over $\mathbb{F}_2$	$s \log(s/\tau)$	$(2s \log(s/\tau))^{s \log(s/\tau))}$
5	size-s decision trees	s	$(8s)^s$
6	size-s branching programs	s	$s^s(s+1)^{2s}$
7	functions with Fourier degree at most d	$d2^d$	$2^{d^2 2^{2d}}$

These bounds hold for any approximation parameter $\tau \geq 0$. But to make Corollary 6.3.1 applicable, we need to pick τ and k such that the (circular) inequalities $\tau \leq \theta_{3.5.6}(k, \varepsilon)$ and $k \geq k^\star(\mathcal{C}, \tau)$ are satisfied.

For items $5, 6, 7$ setting $\tau = 0$ does the job; the reason these bounds are independent of τ is the fact that the corresponding classes contain only functions that actually are $k^\star$-juntas, rather than functions that can be well approximated by $k^\star$-juntas. (For example, the fact that functions with Fourier degree d are actually $(d \cdot 2^d)$-juntas follows from a seminal paper of Nisan and Szegedy [NS92].)

For the first 4 items we can set $\tau = \theta_{3.5.6}(s, \varepsilon)^2$. It is easy to verify that this satisfies the foregoing pair of inequalities. Furthermore, since $\theta_{3.5.6}(s, \varepsilon)$ is polynomial in ε/s, we get $k = O(s(\log s + \log 1/\varepsilon))$. Plugging the resulting values into Corollary 6.3.1, we obtain the following query-complexity bounds:

Class	This work	[DLM$^+$07], [PRS02]$^{(*)}$
s-term DNFs, size-s Boolean formulae, s-sparse polynomials over $\mathbb{F}_2$, size-s decision trees, size-s branching programs	$\widetilde{O}(s/\varepsilon)$	$\widetilde{O}(s^4/\varepsilon^2)$
size-s Boolean circuits	$\widetilde{O}(s^2/\varepsilon)$	$\widetilde{O}(s^6/\varepsilon^2)$
functions with Fourier degree at most d	$\widetilde{O}(2^{2d}/\varepsilon)$	$\widetilde{O}(2^{6d}/\varepsilon^2)$
s-term monotone DNFs	$\widetilde{O}(s/\varepsilon)$	$\widetilde{O}(s^2/\varepsilon)^*$

6.4 Lower bounds

In order to analyze how close to optimal our algorithms are, we in this section we discuss lower bounds concerning the problems studied here.

For ease of exposition we make a slight generalization of the notion of k-uniformity in Section 2.4.2. We identify the set $\{0,1\}^n$ with the field with 2^n elements.

6.4.1. Definition. Let D, S be sets, S finite, and let $\mathcal{C}$ denote a class of functions $f\colon D \to S$. We say that $\mathcal{C}$ *can generate k-wise independence* if there is a distribution $\mathcal{D}$ supported on elements of $\mathcal{C}$ such that the random variables $\{f(x)\}_{x \in D}$ are k-wise independent and each of them is uniformly distributed on S, i.e.,

$$\Pr_{f \sim \mathcal{D}}[f(x_1) = \alpha_1 \wedge f(x_2) = \alpha_2 \wedge \cdots \wedge f(x_k) = \alpha_k] = |S|^{-k}$$

for any k distinct $x_1, \ldots, x_k \in D$ and any $\alpha_1, \ldots, \alpha_k \in S$.

If $|S| = 2$ and $D = \{0,1\}^n$, this is the same as saying that some distribution over $\mathcal{C}$ is k-uniform. Clearly the class of all boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ can generate n-wise independence (here $S = \{0,1\}$).

The following observation is just a restatement of Definition 6.4.1, coupled with Yao's principle:

6.4.2. Lemma. *If $\mathcal{C}$ can generate k-wise independence under distribution $\mathcal{D}$, then at least $k+1$ adaptive queries are needed to distinguish, with probability $> 1/2$, between a function f drawn from $\mathcal{D}$ and a uniformly random $f\colon D \to S$.*

We say class $\mathcal{C}$ is *ε-far from uniform* if a uniformly random function $f\colon D \to S$ is ε-far from every element of $\mathcal{C}$ with probability larger than $2/3$.

It follows that if $\mathcal{C}$ is both ε-far from uniform and capable of generating k-wise independence, then more than k queries are necessary for testing membership in $\mathcal{C}$ to accuracy ε.

Now we give some examples of simple boolean functions classes on $\{0,1\}^n$ fitting these definitions.

6.4.1 Low-degree polynomials over $\mathbb{F}_2$

6.4.3. Lemma. *Let $\mathcal{F}_d$ be the set of all polynomials $p : \mathbb{F}_2^n \to \mathbb{F}_2$ of degree at most d. Then the uniform distribution over $\mathcal{F}_d$ is $(2^{d+1}-1)$-uniform.*

In other words, polynomials of degree $\log(k+1)-1$ over $\mathbb{F}_2$ can generate k-wise independence.

Proof. It is enough to prove the following claim: for any set $S \subseteq \mathbb{F}_2^n$ of size $|S| < 2^{d+1}$, and any function $f : S \to \mathbb{F}_2$, there is a polynomial $q \in \mathcal{F}_d$ such that $q\restriction_S = f$ (this fact has been generalized in the works [KS05, BEHL09]). Indeed, if the claim holds then

$$\Pr_{p\in\mathcal{F}_d}[p\restriction_S = f] = \Pr_{p\in\mathcal{F}_d}[(p\oplus q)\restriction_S = 0] = \Pr_{p'\in\mathcal{F}_d}[p'\restriction_S = 0],$$

since the distributions of p and $p' \triangleq p \oplus q$ are uniform over $\mathcal{F}_d$. Therefore this probability is the same for every f.

We prove now this claim by induction on $|S| + n$; it is trivial for $|S| = 0$ or $n = 0$. Suppose that, after removing the first bit of each element of S, we still get $|S|$ distinct vectors; then we can apply the induction hypothesis with S and $n-1$. Otherwise, there are disjoint subsets $A, B, C \subseteq \{0,1\}^{n-1}$ such that $S = \{0,1\} \times A \cup \{0\} \times B \cup \{1\} \times C$, and $A \neq \emptyset$.

We can find, by induction, a polynomial $p_{0A,0B,1C}$ of degree $\leq d$ on $n-1$ variables that computes f on $\{0\}\times A \cup \{0\}\times B \cup \{1\}\times C$. As $|S| = 2|A| + |B| + |C|$, either $|A| + |B|$ or $|A| + |C|$ is at most $\frac{|S|}{2} < 2^d$; assume the latter. Then any function $g : A \cup C \to \mathbb{F}$ can be evaluated by some polynomial $p_{AC}(y)$ of degree $\leq d-1$; consider $g(y) = 0$ if $y \in C$ and $g(y) = f(1,y) - p_{0A,0B,1C}(1,y)$ if $y \in A$. Then the polynomial $p(x,y) = p_{0A,0B,1C}(y) + x p_{AC}(y)$ does the job. □

6.4.2 Small circuits

6.4.4. Lemma. *Each of the following classes can generate n-wise independence: boolean formulae of size* $\mathrm{poly}(n)$, NC^1 *circuits, and branching programs of size* $\mathrm{poly}(n)$.

Proof. First observe that the class of all *univariate* polynomials of degree $\leq k-1$ over $\mathbb{F}$ can generate k-wise independence, because any degree $\leq k-1$ polynomial over a field can be interpolated from its values on any set of k distinct points, and the solution is unique. From this we can obtain a family of *boolean* functions on $\{0,1\}^n$ that generates k-wise independence in the following way: associate with each polynomial $p\colon GF(2^n) \to GF(2^n)$ of degree $k-1$ the function that, on input $x \in \{0,1\}^n$, returns the last bit of $p(x)$, by which we mean the representation of $p(x) \in GF(2^n)$ as an n-bit string, which is uniformly distributed among all field elements. (A different, slightly more efficient way, would be to work on a field of size roughly $2^n/n$.) Clearly the resulting family can generate k-wise independence.

By a result of Healy and Viola [HV06], field arithmetic over $GF(2^n)$ is in TC^0 (see also [Ebe89] for a weaker result). This is the class of polynomial-size, constant-depth circuits with unbounded fan-in built up from threshold gates (as well as AND, NOT and OR gates). It is known that TC^0 is contained in NC^1, which corresponds to fan-in 2 circuits of polynomial size and logarithmic depth. Altogether this means that it is possible to evaluate any t-term polynomial $p \in GF(2^n)[x]$ with an NC^1 circuit of size $\mathrm{poly}(n,t)$, which is $\mathrm{poly}(n)$ if $t = \mathrm{poly}(n)$. It is also known that NC^1 corresponds precisely to the set of Boolean formulae of polynomial size, and also to the set of functions computed by width-5 branching programs of polynomial size by Barrington's theorem [Bar89]. Summarizing, the last bit of polynomial functions over $GF(2^n)$ of degree n (and therefore $n+1$ terms) can be computed by boolean formulae of size n^c and branching programs of size n^c for some (small) c. □

As a corollary, we obtain a result of independent interest that says that functions for which isomorphism testing is hard can have relatively simple descriptions, providing a partial derandomization of Theorem 2.4.1.

6.4.5. Theorem. *Let $0 < \varepsilon < \frac{1}{2}$ and $r \in \mathbb{N}$. For large enough n, among the functions $f\colon \{0,1\}^n \to \{0,1\}$ for which ε-testing isomorphism requires $\Omega(r)$ queries there are:*

- *functions of degree $O(r)$ (over any field $\mathbb{F}$ fixed in advance);*
- *n-variate polynomials of degree $O(\log r)$ over $\mathbb{F}_2$;*
- *functions computed by circuits of size $\mathrm{poly}(r)$ and depth $O(\log r)$; or boolean formulae of size $\mathrm{poly}(r)$; or branching programs of size $\mathrm{poly}(r)$.*

Proof.

- By Corollary 3.3.2, there are r-juntas with property stated. Any r-junta has degree r.
- Let us first consider the case $r = n$. In this case we can prove the result for $\varepsilon < \frac{1}{2}$. By Theorem 2.4.12, it is enough to show that there is an n^4-uniform distribution among polynomials of degree $O(\log n)$; this is Lemma 6.4.3. To show the general statement for $r < n$ and $\varepsilon < \frac{1}{4}$, consider the set of functions that depend on the first r variables. Among them there are pairs of polynomials of degree $O(\log r)$ that are $(r, 2\varepsilon)$-hard; then we can extend them to functions on n variables and use Corollary 3.3.2. This extension does not change the degree.
- Similarly, we can consider functions that depend on the first r variables and then extend them to n; this does not affect circuit size, etc. So we need to prove the case $r = n$, and again this is implied if we show that NC^1 circuits (resp. formulae, branching programs) can generate n^4-independence. This is Lemma 6.4.4.

□

6.4.3 The bounds

The next theorem summarizes the lower bounds known.

6.4.6. Theorem. *The following lower bounds hold for the respective testing problems (for some constant ε and size parameter up to some polynomial of n):*

1. *size-s boolean formulae, branching programs and boolean circuits:* $\mathrm{poly}(s)$.
2. *functions with $\mathbb{F}_2$ degree d:* $2^{\Omega(d)}$.
3. *s-sparse polynomials over $GF(2)$:* $\Omega(s)$.
4. *functions with Fourier degree d:* $\Omega(d)$.
5. *s-term DNFs, size-s decision trees:* $\Omega(\log s)$.

The first bound appears in [CGM11b]. The second one was previously known [AKK$^+$03], but we give a new proof. The bounds 3-5 are in [BBM11].

Proof.

1. By Lemmas 6.4.2 and 6.4.4, we only need to prove that these classes are $\Omega(1)$-far from uniform. This is easy to verify by counting.

2. By Lemmas 6.4.2, 6.4.3 and the fact that a random function is $\Omega(1)$-far from having degree $< n/3$.

3. This follows from the lower bounds for s-parities in Chapter 5, as any s-parity is an s-sparse polynomial over $\mathbb{F}_2$, and also far from being $(s-1)$-sparse [DLM$^+$07, BBM11]. A (weaker) $\Omega(\sqrt{s}\,)$ lower bound is a consequence of the lower bound of Theorem 5.1.3, and an $\Omega(s)$ bound follows from the $\Omega(s)$ bound for distinguishing s-parities from $(s-2)$-parities (Theorem 5.1.5).

4. Again, it is enough to show that s-parities are far from having Fourier degree $< s$; see [BBM11].

5. See [BBM11].

□

6.5 Summary

We plugged in our sample extractor in the "testing by implicit learning" framework of Diakonikolas et al. [DLM$^+$07], improving the query complexity of testers for various Boolean function classes. In particular, for some of the classes considered in [DLM$^+$07], such as s-term DNF formulae, size-s decision trees, size-s Boolean formulae, s-sparse polynomials over $\mathbb{F}_2$, and size-s branching programs, the query complexity is reduced from $\widetilde{O}(s^4/\varepsilon^2)$ to $\widetilde{O}(s/\varepsilon)$. For most of these problems we also supply lower bounds that are polynomially related to the query complexity of our algorithms.

Chapter 7

Learning parities

The content of this chapter is based on the paper

- H. Buhrman, D. García–Soriano and A. Matsliah. Learning parities in the mistake-bound model. *Information Processing Letters*, 111(1):16–21, 2010.

7.1 Overview of learning theory

Just as we studied the problem of isomorphism to parities under the property testing viewpoint in Chapters 2 and 5, we consider now the similar problem from the *learning* perspective (in the noiseless setting). We assume some familiarity with computational learning theory; refer to the book of Kearns and Vazirani [KV94] for an easily readable introduction to the field. Nevertheless, for the benefit of the reader we give here a condensed and somewhat informal account.

7.1.1 Mistake-bounded learning and equivalence queries

In learning one is interested in a certain *concept class* $\mathcal{C}$ of objects to which the learning algorithm applies. Here we will always take $\mathcal{C}$ to be a subset of boolean functions. The *mistake-bound model* of learning is an online model introduced by Littlestone in [Lit88]. In this setting learning proceeds in rounds. A target function $f \in \mathcal{C}$ is selected arbitrarily by the "teacher" and is unknown to the "learner". In each round the teacher provides an *unlabelled* example $x \in \{0,1\}^*$, and the learner must predict the value $f(x)$. Then the learner is told the correct value of $f(x)$, according to which it can update its current hypothesis, i.e., the process whereby the future predictions will be made. The *mistake bound* of the learner, with respect to a target function f, is the worst-case number of mistakes that it makes over all (arbitrary, possibly infinite) sequences of examples. The mistake bound on a concept class $\mathcal{C}$ is the maximum of the mistake bounds taken over all possible target functions $f \in \mathcal{C}$.

The discussion so far ignores the running time of the learner. In order to quantify the behaviour of learning algorithms with respect to the input size, we need to consider a sequence of concept classes $\mathcal{C}_n, n \in \mathbb{N}$, one for each input size n (the value of n being known by the learner). In this chapter we will always assume $\mathcal{C}_n$ is a collection of boolean functions on $\{0,1\}^n$.

7.1.1. Definition. We say that the class $\mathcal{C} = \bigcup_{n\in\mathbb{N}} \mathcal{C}_n$ is *learnable with mistake bound $M = M(n)$ and running time per round $t = t(n)$* if there is a learner L that on each example $x \in \{0,1\}^n$ runs in time $t(n)$ and outputs its guess for $f(x)$, and has mistake bound at most $M(n)$ on $\mathcal{C}_n$.

We say that $\mathcal{C}$ is *efficiently learnable with mistake bound M* if in addition $t(n) = \mathrm{poly}(n)$.

It is easily seen that, up to one additional query, this model coincides with that of Angluin [Ang88] on learning with *equivalence queries*.[1]

Note that the learner may never know if its current hypothesis is a good approximation to f in some sense (unless M mistakes have already been made). For instance, suppose that the teacher gives the same example over and over again. For this particular sequence of examples, any non-trivial learner will make one mistake at most, but usually this will not allow us to learn anything about how f behaves on other inputs.

7.1.2 PAC learning

The Probably Approximately Correct (PAC) model was introduced in the seminal paper of Valiant [Val84] and has since attracted a lot of attention from researchers in learning theory. In spite of being the first model defined it is a bit more complicated to describe. Here we additionally need to refer to a *hypothesis class* $\mathcal{H}$ of boolean functions; it represents the possible hypotheses the learning algorithm may output. We assume that $\mathcal{H} \supseteq \mathcal{C}$, but in general $\mathcal{H}$ does not need to coincide with $\mathcal{C}$, and the choice of $\mathcal{H}$ has a significant impact on the learning complexity. (When $\mathcal{H} = \mathcal{C}$, the learner is called *proper*.)

The PAC model is a model for "batch" learning: the learner trains on some fixed dataset, and with this information it constructs a hypothesis that is hopefully "good". The dataset consists of *labelled* samples of some concept $f \in \mathcal{C}_n$, by which we mean a pair $(x, f(x))$, where $x \in \{0,1\}^n$. The input x in these samples come from some unknown distribution $\mathcal{D}$ over $\{0,1\}^n$. When the learner L has collected enough data, it outputs a representation of a hypothesis $h \in \mathcal{H}$ that must, with high probability, be close to f. (The representation of h must be polynomial-time evaluatable: given x, the value of $h(x)$ must be computable in polynomial time.)

[1] Here the learner does not receive a particular input x to try to guess $f(x)$; instead, it is allowed to ask "equivalence queries" whereby it tells the teacher a representation of its guess h for the entire function f. The teacher then tells the learner whether the guess was correct ($h = f$), and if not the teacher provides a counterexample, meaning an input x with $f(x) \neq h(x)$.

A confidence parameter $\delta \in (0,1)$ controls the error probability of the algorithm, whereas an error parameter ε controls the error of the hypothesis h. Since we trained the algorithm with examples from $\mathcal{D}$, we can only hope for h to be a good approximation to f for random inputs drawn from $\mathcal{D}$. In other words, we want to have $\mathrm{dist}_{\mathcal{D}}(f,h) \leq \varepsilon$ with probability at least $1-\delta$, where

$$\mathrm{dist}_{\mathcal{D}}(f,h) = \Pr_{x\sim\mathcal{D}}[f(x) \neq h(x)].$$

7.1.2. Definition. We say that the class $\mathcal{C}$ is PAC-*learnable using* $\mathcal{H}$ if there exists an algorithm L that is provided with two real numbers δ, ε and has the following property:

For every concept $f \in \mathcal{C}$, every input distribution $\mathcal{D}$, and all $0 < \varepsilon, \delta < 1/2$, L outputs a hypothesis concept $h \in H$ that, with probability at least $1-\delta$, satisfies $\mathrm{dist}_{\mathcal{D}}(f,h) \leq \varepsilon$.

The probability is taken over the random samples and the internal random coin flips of A.

In order to define efficient PAC learning, we fix a "reasonable" representation scheme for the elements of $\mathcal{H}$ and let $\mathrm{size}(h)$ denote the length of the shortest description of $h \in H$ under this scheme. Note that $\mathrm{size}(f)$ is a lower bound on the running time of a learner when presented with target f, because at the end of the day the learner must output a representation of its hypothesis.

7.1.3. Definition. We say that the class $\mathcal{C} = \bigcup_{n\in\mathbb{N}} \mathcal{C}_n$ is *efficiently* PAC-*learnable using* $\mathcal{H}$ if there exists a learner L for $\mathcal{C}$ that, when learning a target function $f \in \mathcal{C}_n$, runs in time $\mathrm{poly}(n, 1/\varepsilon, 1/\delta, \mathrm{size}(f))$; and moreover the hypothesis $h \in \mathcal{H}$ output by L can be evaluated on any given input in time polynomial in n and $\mathrm{size}(h)$.

Note that the learning must work under any distribution $\mathcal{D}$. Sometimes this requirement is weakened so the tester only needs to work for a specific $\mathcal{D}$.

If we disregard the efficiency requirements, there is a tight characterization of the complexity of PAC learning in terms of the so-called Vapnik-Chervonenkis dimension of the class.

7.1.4. Definition. Let $\mathcal{C}$ be a concept class of boolean functions on some domain A. For a sequence of inputs $S = (x_1, \ldots, x_m) \in A^m$, define

$$\Pi_{\mathcal{C}}(S) = \{(f(x_1), \ldots, f(x_m)) \mid f \in \mathcal{C}\}$$

Then S is said to be *shattered* by $\mathcal{C}$ if $\Pi_{\mathcal{C}}(S) = \{0,1\}^m$, i.e., any assignment of answers to the elements of S is consistent with some $f \in \mathcal{C}$.

(If we agree on an ordering of the input domain A, we can speak of sets of inputs instead of sequences, as is usually done.)

7.1.5. Definition. The VC dimension of $\mathcal{C}$ is the cardinality of the largest set of inputs shattered by $\mathcal{C}$, or ∞ if there are arbitrarily large shattered sets.

7.1.6. Theorem (see [KV94, Theorems 3.3 and 3.4]). *Let $\mathcal{C}$ be a concept class of VC dimension d. Then $\mathcal{C}$ can be* PAC*-learned with sample complexity*

$$O\left(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon}+\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$$

On the other hand, any PAC *learner for $\mathcal{C}$ must use $\Omega(d/\varepsilon)$ examples in the worst case.*

There are standard conversion techniques that can be used to transform any mistake-bounded algorithm into a PAC learning algorithm. These transformations preserve the running time of the mistake-bounded algorithm, and the sample size required by the PAC algorithm is equal to the mistake bound, up to constant factors that depend on its approximation and confidence parameters.

7.1.7. Theorem (Angluin [Ang88]; Littlestone [Lit89]). *Any algorithm A that learns $\mathcal{C}$ in the mistake-bound model with mistake bound M and maximum running time per round t can be converted into an algorithm A' that learns $\mathcal{C}$ in the* PAC *model using a sample set of size $O(\frac{1}{\varepsilon}M+\frac{1}{\varepsilon}\log\frac{1}{\delta})$ and running time $O(\frac{1}{\varepsilon}Mt\log\frac{M}{\delta})$, where ε and δ are the approximation and confidence parameters of A'.*

Let us briefly mention how the PAC learner constructed operates. First, it takes $\Theta(\frac{M}{\varepsilon}+\frac{1}{\varepsilon}\log\frac{1}{\delta})$ random samples, and runs the mistake-bounded learner on them. Without loss of generality, the learner only changes hypotheses after a mistake, so at most M hypotheses are produced. The PAC learner then draws a new set of $O(\frac{1}{\varepsilon}\log\frac{M}{\delta})$ samples, uses it to test the validity of each hypothesis, and selects the best. We omit the proof.

These conversion techniques imply that positive results for mistake-bound learning, in particular those given in this chapter, directly yield corresponding positive results for PAC learning. We mention here that no such conversion is known in the opposite direction. In fact, Blum [Blu94] proved that under widely held assumptions (namely, the existence of one-way functions) the mistake-bound model is strictly harder than the PAC model.

7.1.3 Attribute-efficient learning

The study of attribute-efficient learning was initiated in the same paper introducing the mistake-bound model [Lit88]. A learning algorithm L for $\mathcal{C}$ in the mistake-bound model is *attribute-efficient* if all the foregoing conditions are met and the mistake bound of A on any $f \in C$ is bounded by a polynomial in size(f), independent of n. Similarly, an algorithm A for learning $\mathcal{C}$ in the PAC model is attribute-efficient if the sample size required for A to learn the target function $f \in \mathcal{C}$ is polynomial in size$(f), 1/\delta, 1/\varepsilon$.

7.2 General observations

One of the long-standing open questions in both the mistake-bound and the PAC learning models is whether parities can be learned attribute-efficiently in polynomial time [Blu96]. With each vector $\tilde{f} \in \{0,1\}^n$ we associate a parity function $f\colon \{0,1\}^n \to \{0,1\}$ defined by $f(x) = \langle \tilde{f}, x \rangle \triangleq \sum_{i=1}^{n} \tilde{f}_i x_i \bmod 2$ for all $x \in \{0,1\}^n$. As any two parities are $1/2$-far apart, learning a parity function with $\varepsilon < 1/2$ can be thought of as identifying the corresponding n-bit vector $\tilde{f}$. With a slight abuse of notation, from now on we will denote by f both the parity function $f\colon \{0,1\}^n \to \{0,1\}$ and its corresponding vector $\tilde{f} \in \{0,1\}^n$; in particular we may for instance write $|f|$ for the Hamming weight of $\tilde{f}$. The concept class $\mathsf{PAR}^n_{\leq k}$ is defined as the class of all parity functions of Hamming weight at most k. We will also refrain from explicitly mentioning n throughout the chapter so we will write $\mathsf{PAR}_{\leq k}$ instead. The description length of any function $f \in \mathsf{PAR}_{\leq k}$ is $O(k \log n)$, and thus ideally we would like to have poly(n)-time algorithms that learn $\mathsf{PAR}_{\leq k}$ with a mistake bound (respectively sample size) of poly$(k \log n)$. This would correspond to attribute-efficient learning as defined above.

It is well known that, in exponential time, $\mathsf{PAR}_{\leq k}$ can be learned attribute-efficiently in the mistake-bound model (and hence in the PAC model too). A simple algorithm with mistake bound at most $k \log n$ is the *halving algorithm*. It maintains a set $\mathcal{H} \subseteq \mathsf{PAR}_{\leq k}$ of candidate parity functions, and given an example x, it predicts $\mathrm{Maj}\{h(x) \mid h \in \mathcal{H}\}$. Whenever a mistake is made, all "wrong" candidates (of which there are at least $|\mathcal{H}|/2$) are removed from $\mathcal{H}$. If initially the set $\mathcal{H}$ was set to be $\mathsf{PAR}_{\leq k}$, then after at most $\log |\mathsf{PAR}_{\leq k}| \leq k \log n$ mistakes the function f is learned. The running time of the halving algorithm is dominated by the time needed to compute the predicate $\mathrm{Maj}\{h(x) \mid h \in \mathcal{H}\}$. A naive computation of this predicate requires $|\mathsf{PAR}_{\leq k}| \geq \binom{n}{k}$ steps, and in fact the running time of all known algorithms that try to compute or approximate this predicate is super-polynomial for any $k = \omega(1)$.

The question of learning juntas efficiently in the PAC setting under the uniform distribution was raised by Mossel, O'Donnell and Servedio [MOS04]. They showed that k-juntas are PAC-learnable under the uniform distribution in time roughly

$n^{\frac{\omega}{\omega+1}k} \cdot \mathrm{poly}(2^k, n)$, which is approximately $n^{0.704}$ (here $\omega < 2.373$ is the exponent in the time bound for matrix multiplication [CW90, Sto10, Wil11]). As PAR_k is a subset of the class of all juntas, this applies to parities too and yields, in this particular case, an improvement over the halving algorithm in terms of running time.

On the other hand, if we pay no attention to efficiency issues it is possible to show that the halving algorithm is optimal for parities with regard to the sample complexity/mistake bound (though it is known not to be optimal in general; see [Lit88] for counterexamples). The reason is that any subset of PAR^n forms a list-decodable code (for more detailed accounts on what this means see, e.g., the surveys of Trevisan [Tre04] and Sudan [Sud01]).

7.2.1. Theorem.
Let $\mathcal{C} \subseteq \mathsf{PAR}^n$ and $0 < \varepsilon < 1/2, \delta < 1/6$. Then any PAC learner for $\mathcal{C}$ must draw $\Omega\left(\log\left(|\mathcal{C}| \cdot \delta(1/2 - \varepsilon)\right)\right)$ samples, even under the uniform distribution.

In particular, the VC dimension of PAR_k is $\Theta(\log \binom{n}{k})$, and any mistake-bounded learner for PAR_k must make $\Omega(\log \binom{n}{k})$ mistakes in the worst case.

Proof. For the first part, assume there is a PAC learner for $\mathcal{C}$ using $q(\delta)$ samples. With probability $1 - \delta$, its hypothesis h is ε-close to $\mathcal{C}$ (although it need not be a linear function). Let $\alpha = 1/2 - \varepsilon$. There is a well known bound of $1/(4\alpha^2)$ on the number of Hadamard codewords (elements of PAR^n) at distance $\leq \varepsilon$ from any given h; it can be easily proven by analyzing the Fourier coefficients of h (see [Wol08]). By Occam's razor (Theorem 2.2 in [KV94]), we can reduce the number of candidates from $O(1/\alpha^2)$ to one with $O(\log(1/(\delta\alpha^2))$ additional samples and error probability δ. Finally we can test for equality with the single remaining candidate with confidence δ with $O(\log(1/\delta))$ samples. All in all we get an algorithm that draws $q(\delta) + O(\log(1/(\alpha\delta))$ samples and with confidence $1 - 3\delta > 1/2$ manages to *exactly identify* $f \in \mathcal{C}$. Yet a straightforward information-theoretic argument shows that the latter task needs at least $\log|\mathcal{C}| - 1$ samples, hence the lower bound.

For the "in particular" part, note that $\mathrm{VCdim}(\mathcal{C}) \leq \log|\mathcal{C}|$ always holds, and the inequality $\mathrm{VCdim}(\mathcal{C}) \geq \Omega(\log|\mathsf{PAR}_k|)$ follows from the first part and the upper bound of Theorem 7.1.6 after setting constant values for δ, ε. Finally recall that PAC learning lower bounds imply mistake-bound learning lower bounds by virtue of Theorem 7.1.7. □

On the other hand, with a mistake bound of n (respectively, a sample set of size $O(n)$), parities can be learned straightforwardly in polynomial time: one can check, for each new example, whether it is a linear combination of the previous ones; if it is, we output the appropriate linear combination of the previous answers. At most n mistakes will be made since this is the size of a maximum linearly independent subset of $\{0,1\}^n$. We will call this the *trivial algorithm* (see also [Blu96]). Just which tradeoffs are attainable between a learner's running time and its mistake bound is one of the driving questions we investigate in this chapter.

7.3 Results and related work

Despite the simplicity of these algorithms, no other methods for learning parities in the mistake-bound model were known prior to this work. In particular, it was unknown whether $\omega(1)$-parities could be learned in polynomial time with $o(n)$ mistakes. Our main result (stated next) is the first step in this direction.

7.3.1. Theorem (Buhrman et al. [BGM10, Main result]).
Let $k, t\colon \mathbb{N} \to \mathbb{N}^+$ be two functions satisfying $k(n) \le t(n) \le n$.[2] For every $n \in \mathbb{N}$ (and the corresponding integers $k = k(n)$ and $t = t(n)$) there is a deterministic algorithm that learns $\mathsf{PAR}_{\le k}$ in the mistake-bound model, with mistake bound $k\lceil \frac{n}{t} \rceil + \lceil \log \binom{t}{k} \rceil$ and running time per round $O\Big(\binom{t}{k}(kn/t)^2\Big)$.

Let us examine a few interesting values for the parameters in Theorem 7.3.1. For example, putting $k = \log n / \log\log n$ and $t = \log n$ yields mistake bound of $O(n/\log\log n)$ and running time per round $O(n^{2+o(1)})$. More generally, it is interesting to find out when $\mathsf{PAR}_{\le k}$ can be efficiently learned with $o(n)$ mistakes. On the other hand, we saw in Section 7.2 (from arguments using the VC dimension) that for $k = \Omega(n)$ it is impossible to learn $\mathsf{PAR}_{\le k}$ with sublinear mistake bound, even disregarding computational efficiency. So we only need to consider the case $k = o(n)$. Recall that the running time of the halving algorithm is at least $\binom{n}{k}$, which is super-polynomial for any super-constant k, and is $2^{\Omega(k \log n)}$ for any positive $k = n^{1-\Omega(1)}$. In the following we show that, with appropriate parameters, our main theorem can be used to outperform the halving algorithm. Specifically,

- for any $k = o(\log n)$, $\mathsf{PAR}_{\le k}$ can be learned with $o(n)$ mistakes in polynomial time;
- for any $k = o(n)$, $\mathsf{PAR}_{\le k}$ can be learned with $o(n)$ mistakes in time $\approx 2^{O(k+\log n)}$.

The two items above are formalized next.

7.3.2. Corollary (Case $k = O(\log n)$).
For any $\omega(1) \le k = O(\log n)$ and $c \in \mathbb{N}$, define $t = t(n) = \lceil \frac{kn^{c/k}}{e} \rceil$ (for large enough n). Then $\mathsf{PAR}_{\le k}$ can be learned deterministically with mistake bound $O(n^{1-c/k})$ and running time per round $O(n^{c+2-2c/k})$. Consequently (see Theorem 7.1.7), $\mathsf{PAR}_{\le k}$ can be learned deterministically in the PAC model with $O(n^{1-c/k})$ samples and running time $\widetilde{O}(n^{3+c-3c/k})$.

In particular, if $k = o(\log n)$ then the mistake bound (sample size) is $o(n)$.

[2] We assume that the functions $k(n), t(n)$ are computable in $O(n^2)$ time.

7.3.3. Corollary (Case $k = o(n)$). *For any $\omega(1) \leq k = o(n)$ let $t = t(n) = o(n)$ be an arbitrary function with growth rate $\omega(k)$. Then $\mathsf{PAR}_{\leq k}$ can be learned deterministically with mistake bound $O(kn/t + k \log \frac{t}{k}) = o(n)$, and total running time $2^{O(k \log \frac{t}{k} + \log n)}$.*

Consequently (see Theorem 7.1.7), $\mathsf{PAR}_{\leq k}$ can be learned deterministically in the PAC model with $O(kn/t + k \log \frac{t}{k}) = o(n)$ samples and running time $2^{O(k \log \frac{t}{k} + \log n)}$.

For example, if $t = k \log k$ then the running time is $2^{O(k \log \log k + \log n)}$.

In addition to the corollaries above, observe that Theorem 7.3.1 with $t = \frac{n}{\log(n/k)}$ gives the same mistake bound as the halving algorithm with slightly better running time. Similarly, we can obtain the features of the trivial algorithm by setting $k = t = n$.

7.3.1 Learning parities in the PAC model

In the PAC model, Klivans and Servedio [KS06] were the first to show non-trivial algorithms for learning parities with sample sets of sublinear size. (They attribute the second item of the following theorem to Spielman, although it seems to have previously appeared in the literature, e.g., [HB01].)

7.3.4. Theorem (Klivans & Servedio [KS06]).

1. *$\mathsf{PAR}_{\leq k}$ can be learned in the PAC model with $O(n^{1-1/k} \log n)$ samples in time $O(n^4)$.*
2. *$\mathsf{PAR}_{\leq k}$ can be learned in the PAC model with $O(k \log n)$ samples in time $\widetilde{O}(n^{\lceil k/2 \rceil})$.*

Since our main theorem holds in the harder mistake-bound model, using the standard conversion techniques (Theorem 7.1.7) it also implies results similar to those in Theorem 7.3.4, even with improved parameters. In particular, from Corollary 7.3.2 (with $c = 1$) and Corollary 7.3.3 we get the following.

7.3.5. Theorem (Buhrman et al. [BGM10]).

1. *$\mathsf{PAR}_{\leq k}$ can be learned in the PAC model with $O(n^{1-1/k})$ samples in time $O(n^{4-3/k} \log n)$.*
2. *$\mathsf{PAR}_{\leq k}$ can be learned in the PAC model with $o(n)$ samples in time $\approx 2^{O(k + \log n)}$.*

In the first item, the number of samples required by our algorithm is improved by a factor of $\log n$, and the running time is improved by a factor of $n^{3/k}$. As for Item 2, our algorithm requires more than $O(k \log n)$ samples (although still

$o(n)$), but its running time is reduced to $\approx 2^{O(k+\log n)}$, compared to the $2^{\Omega(k \log n)}$ time required by both the halving algorithm and the algorithm from Item 2 of Theorem 7.3.4. In addition to these features, our algorithms are *deterministic* whereas the algorithms from [KS06] are probabilistic.

7.3.2 Extending the $\tilde{O}(n^{k/2})$ algorithm to the MB model

The second item in Theorem 7.3.4 improves on the halving algorithm, bringing down the running time to roughly $O(n^{k/2})$, but it still uses a sample set of the same size (up to constant factors). It is natural to ask whether such an improvement is attainable in the mistake-bound model too. Our main result does not directly imply such an improvement; however, using similar ideas it is possible to extend Item 2 of Theorem 7.3.4 to the mistake-bound model as well. Specifically, the following theorem is proved in Section 7.5.

7.3.6. THEOREM. *$\mathsf{PAR}_{\leq k}$ can be learned in the mistake-bound model with mistake bound $O(k \log n)$ and maximum running time per round $O(n^{\lceil k/2 \rceil})$.*

After our work, algorithms with similar time complexities for the noisy version of this problem have been studied by Grigorescu et al. [GRV11] and by Valiant [Val12].

7.4 Proof of the main theorem

The algorithm from Theorem 7.3.1 is based on an idea that was recently used by Alon, Panigrahy and Yekhanin, who gave elegant deterministic approximation algorithms for the Nearest Codeword and Remote Point problems (see [APY09] for details). First we outline the main idea in this algorithm, and then provide its formal description together with the proof.

7.4.1 Informal description of the algorithm

Recall that, in the halving algorithm, a set $\mathcal{H}$ of candidate parity functions is maintained, and given an example x, the prediction of the learner is

$$\mathrm{Maj}\,\{h(x) \mid h \in \mathcal{H}\}.$$

The problem with this method is that for any $k = \omega(1)$, the initial set $\mathcal{H} = \mathsf{PAR}_{\leq k}$ is of super-polynomial size, and we have no efficient algorithm to compute the majority vote.

In order to overcome this problem, we use a special set of affine spaces that enables a compact representation of (a superset of) the candidate parity functions, while at the same time enabling efficient approximation of their majority vote,

for any example x. Specifically, our learning algorithm begins by obtaining a set of affine spaces $N_1, N_2, \cdots \subseteq \{0,1\}^n$, at least one of them containing the target parity function f. In every step of the the learning process, these sets of affine spaces are updated according to the response given by the teacher. The way these updates are performed guarantees:

- the running time is polynomial in n and linear in the number of affine spaces N_i;
- after every mistake, some sets N_i get shrunk, so that the quantity $\sum_i |N_i|$ is at least halved (this is ensured by approximating the majority vote);
- the target function f is never removed from any N_i.

Since $\sum_i |N_i| \geq |\bigcup_i N_i|$, after at most a logarithmic (in $\sum_i |N_i|$) number of mistakes the target function f is the only element left in $\bigcup_i N_i$, and hence f is learned.

7.4.2 Formal description and proof

Fix n and $t = t(n)$. Define $S \subseteq 2^{[t]}$ as $S = \{s \subseteq [t] \mid |s| = k\}$, hence $|S| = \binom{t}{k}$. Let $\pi = C_1, \ldots, C_t$ be an equipartition of $\{e_1, e_2 \ldots, e_n\}$ (the standard basis of $\{0,1\}^n$) into t parts. For every $s \in S$ we define the linear subspace (over $\mathbb{F}_2^n$) $M_s = \mathrm{span}(U_s)$, where U_s is a set of unit vectors defined as

$$U_s \triangleq \bigcup_{i \in s} C_i.$$

That is, M_s consists of all binary vectors whose nonzero entries belong to the parts that are indexed by the elements of s. Notice that for every $s \in S$, M_s is a span of at most $k\lceil \frac{n}{t} \rceil$ vectors, and hence

$$|M_s| \leq 2^{k\lceil n/t \rceil}.$$

Let $\ell = k\lceil \frac{n}{t} \rceil$. For every affine space $N \subseteq \{0,1\}^\ell$, $x \in \{0,1\}^\ell$ and $z \in \{0,1\}$, we define the affine space $N(x,z) \triangleq \{y \in N \mid \langle y, x \rangle = z \mod 2\}$. Given $x \in \{0,1\}^\ell$, $z \in \{0,1\}$ and a representation for N as a system Lin^N of independent linear equations in triangular form, the corresponding representation of $N(x,z)$ (and the cardinality $|N(x,z)|$) can be computed in time $O(\ell^2)$. This is done by adding $x' = x \sqcup z \in \{0,1\}^{\ell+1}$ as a row to Lin^N and performing only one step of the Gaussian elimination procedure to bring the matrix back into triangular form. Notice that this procedure has three possible outcomes:

(i) x' is inconsistent with Lin^N, and hence $|N(x,z)| = 0$;

(ii) x' is a linear combination of equations in Lin^N, and hence $|N(x,z)| = |N|$;

(iii) x' is linearly independent of Lin^N, and hence $|N(x,z)| = |N|/2$.

7.4.1. Proposition.

1. *Every $f \in \{0,1\}^n$ with $|f| \leq k$ is contained in $\bigcup_{s\in S} M_s$;*
2. $|\bigcup_{s\in S} M_s| \leq \binom{t}{k} 2^{k\lceil n/t\rceil}$.
3. *Let $\{N_s \mid s \in S\}$ be a family of affine subspaces of $\{0,1\}^n$. For any $x \in \{0,1\}^n$ there exists $z \in \{0,1\}$ for which $\sum_{s\in S} |N_s(x,z)| \geq \frac{1}{2}\sum_{s\in S} |N_s|$.*

Proof.

1. This follows from the fact that every set of k unit vectors is contained in the union of some $d \leq k$ subsets $C_{i_1}, \ldots, C_{i_d}$ in the partition π. Let $s \subseteq [t]$, $|s| = k$ be a set that contains $i_1, \ldots, i_d$. Then $f \in M_s$. .
2. $|\bigcup_{s\in S} M_s| \leq \sum_{s\in S} |M_s| \leq \binom{t}{k} 2^{k\lceil n/t\rceil}$.
3. This is a consequence of the equality $|N_s(x,0)| + |N_s(x,1)| = |N_s|$.

□

The learner proceeds as follows:

Initialization:
Obtain a system of equations describing each of the linear spaces M_s as defined above; and then initialize the affine spaces $N_s = M_s$ for all $s \in S$.

On example $x \in \{0,1\}^n$:
Compute $n_0 = \sum_{s\in S} |N_s(x,0)|$ and $n_1 = \sum_{s\in S} |N_s(x,1)|$. If $n_0 \geq n_1$ output 0, else output 1.

On answer $l = \langle f, x\rangle$:
Update $N_s := N_s(x,l)$ for each $s \in S$.

It might be helpful for the reader to think that each M_s runs an independent instance of the trivial algorithm of Section 7.3. Each instance assumes that all relevant parity bits of f belong to the corresponding M_s, and N_s is the set of candidates (parities consistent with the answers to previous examples) left under this assumption. Some of these candidate sets will vanish as new values of f are learned, but at least one of them contains f and hence will always remain non-empty. The prediction made by the algorithm can be viewed as a weighted majority of all "surviving" instances, where the weight of an instance is proportional to the number of candidates left for it. Thus, whenever a new sample is, when restricted to a set M_s, linearly independent of the prior ones,

we "penalize" the sth instance by halving its weight; while if the new example is inconsistent we remove the sth instance from consideration.

Proof of Theorem 7.3.1. First notice that the invariant $f \in \bigcup_{s\in S} N_s$ holds at any stage of the learning algorithm. Initially it holds by Item 1 of Proposition 7.4.1, and every time the algorithm shrinks the sets N_s, only elements that are not equal to f are removed.

Since all the subspaces N_s contain vectors of Hamming weight at most $\ell = k\lceil \frac{n}{t} \rceil$, we can treat them as affine subspaces of $\{0,1\}^\ell$ by truncating all their irrelevant coordinates. In addition, for any N_s, an example $x \in \{0,1\}^n$ can be truncated to the corresponding ℓ-bit vector by removing all the irrelevant coordinates (with respect to N_s). Making this observation, the $O\Big(\binom{t}{k}(kn/t)^2\Big)$ bound on the running time (per round) of the algorithm now follows from the remarks above on Gaussian elimination and the fact that $|S| \le \binom{t}{k}$.

Finally, we have to show that the number of mistakes that the learner makes is bounded by $k\lceil \frac{n}{t} \rceil + \lceil \log \binom{t}{k} \rceil$. Notice that by the definition of the output value, and by Item 3 of Proposition 7.4.1, every time the learner makes a mistake the quantity $\sum_{s\in S} |N_s|$ reduces by a factor of at least 2. Since at every step $0 < \left|\bigcup_{s\in S} N_s\right| \le \sum_{s\in S} |N_s|$, and since initially we started with $\sum_{s\in S} |N_s| = \sum_{s\in S} |M_s| \le \binom{t}{k} 2^{k\lceil n/t \rceil}$ (see Item 2 of Proposition 7.4.1), after at most

$$\log\Big(\sum_{s\in S} |M_s|\Big) \le k\left\lceil \frac{n}{t} \right\rceil + \left\lceil \log \binom{t}{k} \right\rceil$$

mistakes the size of $\bigcup_{s\in S} N_s$ will decrease to 1, which by the invariant above will imply that $\bigcup_{s\in S} N_s = \{f\}$, and the learner will no longer make any errors. □

7.4.3 Optimality of the system of affine spaces

To recap, we have constructed a set $\mathcal{A}$ of $m = \binom{t}{k}$ affine spaces of dimension $d = kn/t \le \frac{3n}{m^{1/k}}$ that together "cover" all vectors of weight $\le k$, in that every such vector belongs to one of the elements of $\mathcal{A}$. The mistake bound we get is $\log \sum_{A\in\mathcal{A}} |A| = \log m + d \le \log m + 3n/m^{1/k}$. One may ask whether this value can be improved upon by finding a better system of affine spaces. It turns out that such a possibility can be ruled out:

7.4.2. Proposition. *Let $1 \le k \le n/100$ and suppose $\mathcal{A}$ is a collection of $m \le \left(\frac{n}{100k}\right)^k$ affine spaces over $\{0,1\}^n$ such that every $x \in \{0,1\}^n$ with $|x| \le k$ belongs to some $A \in \mathcal{A}$. Then*

$$\log\left(\sum_{A\in\mathcal{A}} |A|\right) \ge \log m + \frac{n}{3m^{1/k}}.$$

7.4.3. Lemma. *If $V \in \{0,1\}^n$ is an affine subspace of dimension d, then the number of vectors in V of weight at most k is upper bounded by*

$$\binom{d}{\leq k} = \binom{d}{0} + \binom{d}{1} + \cdots + \binom{d}{k}.$$

Proof. As V is a d-dimensional affine subspace, there is a set $D \subseteq [n]$, $|D| = d$ such that the projection $\pi_D(x)$ of each element $x \in V$ onto the coordinates of D uniquely determines x. Consider the set $V_k \triangleq \{x \in V \mid |x| \leq k\}$ and write $\pi_D(V_k) \triangleq \{\pi_D(x) \mid x \in V_k\}$; we have just seen that $|\pi_D(V_k)| = |V_k|$. But no element of $\pi_D(V_k)$ has weight greater than k by construction, so $|\pi_D(V_k)| \leq \sum_{i=0}^{k} \binom{d}{i}$. □

Proof of Proposition 7.4.2. Write $\mathcal{A} = \{A_1, \ldots, A_m\}$ and $d_i \triangleq \dim A_i$, so $|A_i| = 2^{d_i}$. In order to cover all vectors of weight at most k, simple counting (along with the preceding lemma) tells us that the following inequality must hold:

$$\sum_{i=1}^{m} \binom{d_i}{\leq k} \geq \binom{n}{\leq k}.$$

Suppose there are r subspaces of dimension larger than $2k$ and let $d_1, \ldots, d_r > 2k \geq d_{r+1}, \ldots, d_m$. For the small subspaces we can compute $\sum_{d_i \leq 2k} \binom{d_i}{\leq k} \leq m4^k$. Hence

$$\sum_{i=1}^{r} \binom{d_i}{\leq k} \geq \binom{n}{\leq k} - m4^k \geq \binom{n}{\leq k} - \left(\frac{n}{25k}\right)^k$$

by our bound on m. Together with the well-known inequality $(a/b)^b \leq \binom{a}{\leq b} \leq (ae/b)^b$, this implies that

$$\sum_{i=1}^{r} d_i^k \geq \left(\frac{n}{e}\right)^k - \left(\frac{n}{25e}\right)^k \geq \left(\frac{n}{3}\right)^k.$$

The function $f\colon [e^{k-1}, \infty] \to \mathbb{R}^+$ defined by $f(x) = (\log x)^k$ is concave if $k \geq 1$, therefore Jensen's inequality [CT91] applied to the sequence $\{2^{d_i}\}_{i \in [r]}$ yields

$$f\left(\frac{1}{r}\sum_{i=1}^{r} x_i\right) \geq \frac{1}{r}\sum_{i=1}^{r} f(x_i) \geq \frac{1}{r}\left(\frac{n}{3}\right)^k,$$

from which one obtains (by taking kth roots)

$$\log\left(\sum_{d_i > 2k} 2^{d_i}\right) \geq \log r + \frac{n}{3r^{1/k}} \geq \log m + \frac{n}{3m^{1/k}},$$

where the last step made use of the fact that the function $g(r) = \log r + n/(3r^{1/k})$ is decreasing in a range containing $[1, m]$. □

7.5 Analysis of the $\tilde{O}(n^{k/2})$ algorithm

As mentioned previously, the possibility to improve the running time of the halving algorithm to roughly $O(n^{k/2})$ has been noted by several authors [KS06, HB01]. In this section we explain how to extend the ideas to derive a mistake-bounded algorithm.

Proof of Theorem 7.3.6. Let $A = \mathsf{PAR}_{\leq\lceil k/2\rceil} \times \mathsf{PAR}_{\leq\lceil k/2\rceil}$. Then $|A| = O(n^{k+1})$. We can associate each element $(p, q) \in A$ with the "parity pair" $p \oplus q$; each parity $r \in \mathsf{PAR}_{\leq k}$ will then correspond to several pairs in A, namely those such that $r(x) = p(x) \oplus q(x)$ for all $x \in \{0,1\}^n$. Thus, we can view A as a multiset of $\leq (k+1)$-parities (as well as a set of parity pairs). The *answer* of a parity pair (p, q) on x is defined as $p(x) \oplus q(x)$.

We will show that, given any input x, we can compute the majority vote of the answers of all parity pairs in A that agree with all previous examples in $O(n^{\lceil k/2\rceil})$ time, effectively simulating the halving algorithm over the multiset A. This implies that the number of mistakes will be bounded by $\log |A| = O(k \log n)$.

In order to compute this majority, it is enough to know how many parity pairs in A are consistent with all the examples seen so far and would output 0 for the new example (and how many of them would output 1). Assume we have been given the examples $x = x_1, x_2, \ldots, x_{m-1} \in \{0,1\}^{n\times(m-1)}$ with answers $y = y_1, y_2, \ldots, y_{m-1} \in \{0,1\}^{m-1}$, together with the new example $x_m \in \{0,1\}^n$, and we are required to output our prediction for $f(x_m)$, where f is the unknown parity function. Let $a \triangleq y_1 y_2 \ldots y_{m-1} 0$ be the m-bit vector that contains the answers to all previous $m-1$ examples and whose last entry is 0 (representing that we are trying to count how many consistent parity pairs would answer 0 for x_m). Each parity $p \in \mathsf{PAR}_{\leq\lceil k/2\rceil}$ will give an answer for examples $x_1, \ldots, x_{m-1}, x_m$; let $p(x) \triangleq p(x_1)p(x_2)\ldots p(x_m) \in \{0,1\}^m$ be their concatenation. Consider the multisets

$$V \triangleq \{p(x) \mid p \in \mathsf{PAR}_{\leq\lceil k/2\rceil)}\}$$

and

$$W_a \triangleq \{p(x) \oplus a \mid p \in \mathsf{PAR}_{\leq\lceil k/2\rceil)}\}$$

(where $\oplus$ denotes bitwise addition mod 2). Sort the multiset $V \cup W_a$ in, say, lexicographical order, keeping track of whether each vector comes from V or from W_a. For each range of (consecutive) equal elements in the sorted sequence $V \cup W_a$ (equal to some vector $c \in \{0,1\}^m$), count how many of them are from V and how many are from W_a; call these numbers r and s respectively. What this means is that there are $\lceil k/2\rceil$-parities $p_1, p_2, \ldots, p_r, q_1, \ldots, q_s$ such that

$$c = p_1(x) = p_2(x) = \cdots = p_r(x) = q_1(x) \oplus a = q_2(x) \oplus a = \cdots = q_s(x) \oplus a,$$

and $p_1, \ldots, p_r$ are distinct (as are $q_1, \ldots, q_s$).

Thus, there are exactly rs pairs of parities in $\mathsf{PAR}_{\leq\lceil k/2\rceil}$ such that $p(x)\oplus q(x) = a$ and $p(x) = c$. For each range of equal elements in the sorted sequence $V \cup W_a$, we will find a possible value of c. Summing rs over all these ranges we obtain the number of pairs $(p, q) \in A$ such that $p(x) \oplus q(x) = a$. We can compute this in linear time by making one pass over the sorted sequence. We can similarly compute the number of parity pairs consistent with previous examples that output 1, and then predict the bit that agrees with the majority of consistent parity pairs.

For the implementation, note that we can go through all $\mathsf{PAR}_{\leq\lceil k/2\rceil}$ parities and compute their answers on x_m in $O\left(\binom{n}{\leq\lceil k/2\rceil}\right)$ time (a naive implementation might give an additional factor of n, but this factor can be avoided with some care, for example by backtracking). Note also that before any example has been given, the sequence $V \cup W_a$ can be regarded as a multiset of empty vectors, and is thus sorted; and given a new example, if we keep the multiset $V \cup W_a$ corresponding to our answer from the previous round, we can update the sequence in $O(|V|)$ time by performing one step of radix sort, since it is already sorted with respect to the first $m-1$ bits and we only need to sort it with respect to the newly computed bit (the answer of the parity to x_m), which we can consider the most significant one.

Hence, the total running time per round is $O(|V|) = O\left(\binom{n}{\lceil\leq k/2\rceil}\right)$. □

7.6 Summary

We developed new deterministic algorithms for learning parities in both the mistake-bound and the PAC models of learning. For the mistake-bound model we showed the first efficient algorithm that learns k parities for non-constant k while making a sublinear number of mistakes.

The mistake bound of our algorithm is still far from the value achieved by the halving algorithm. It remains a major open problem to determine whether parities can be learned attribute-efficiently in polynomial time. The halving algorithm has no known efficient implementation, but if $\mathsf{P} = \mathsf{NP}$ it can be converted into one that runs in polynomial time, and has approximately the same mistake bound. (This follows from the result of Stockmeyer [Sto83] that if one is provided with access to an NP oracle, then it is possible to use a randomized polynomial-time algorithm to approximate, to within a constant factor, the number of solutions to an NP predicate.) Two possible avenues of research remain open: either construct an efficient algorithm with improved mistake bound (ideally approaching the bounds of the halving algorithm), or show that the existence of such an algorithm is unlikely by relating it to hardness assumptions from classical complexity theory.

Chapter 8

Monotonicity testing and shortest-path routing

The content of this chapter is based on the paper

- J. Briët, S. Chakraborty, D. García–Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32:1–19, 2012. Earlier version in *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 462–475, 2010.

8.1 Introduction

Testing monotonicity of functions [DGL+99, Ras99, GGL+00, EKK+00, Fis04, FLN+02, AC06, Bha08, HK08] is one of the oldest and most studied problems in Property Testing. The problem is defined as follows: Let $\mathcal{D}$ be a finite, partially ordered set (poset) and let $\mathcal{R} \subseteq \mathbb{Z}$. A function $f\colon \mathcal{D} \to \mathcal{R}$ is *monotone* if for every (comparable) pair $x, y \in \mathcal{D}$, $x \leq y$ implies $f(x) \leq f(y)$. By the standard definitions, a function f is ε-far from monotone if it has to be changed on at least an ε-fraction of the domain $\mathcal{D}$ to become monotone. A one-sided (q, ε)-monotonicity tester for domain $\mathcal{D}$ and range $\mathcal{R}$ is a probabilistic algorithm that, given oracle access to a function $f\colon \mathcal{D} \to \mathcal{R}$, satisfies the following: (a) it makes at most q queries to f; (b) it accepts with probability at least $2/3$ if f is monotone; (c) it rejects with probability at least $2/3$ if f is ε-far from monotone.

The simplest monotonicity testers are those which specify all their queries in advance (non-adaptively) and reject if and only if the responses reveal a violation, i.e., if $f(x) > f(y)$ for some comparable pair $x \leq y$ of points from $\mathcal{D}$. These *non-adaptive* testers with *one-sided error* are the only ones considered in this chapter, unless explicitly stated otherwise. We note that nearly all known monotonicity testers are non-adaptive and have one-sided error. Furthermore, it is also known

that if $\mathcal{D}$ is totally ordered then non-adaptive testers with one-sided error are as powerful (in terms of query complexity) as general ones [Fis04].

For general domains $\mathcal{D}$, Fischer et al. [FLN+02] proved that testing monotonicity is equivalent to several natural problems, including testing certain graph properties and testing assignments for boolean formulae. Domains of the form $\{0, 1, \ldots, m\}^n$, however, received most of the attention [DGL+99, EKK+00, GGL+00, Fis04, Ras99, Bha08, BGJ+09a, BGJ+09b]. Here the order relation $x \leq y$ is defined to hold for $x, y \in \{0, \ldots, m\}^n$ when $x_i \leq y_i$ for all $i \in [n]$. In this chapter we focus on a well-studied subcase of the above, where $m = 1$: integer-valued functions with domain $\{0, 1\}^n$.

8.2 Preliminaries

Recall from Section 1.4 that $H_n = (V_n, E_n)$ is the graph of the directed n-dimensional hypercube.

Given $\mathcal{R} \subseteq \mathbb{Z}$, a finite poset $(\mathcal{D}, \leq)$, and a function $f \colon \mathcal{D} \to \mathcal{R}$, we say that a pair $(x, y) \in \mathcal{D} \times \mathcal{D}$ is *violated* by f if $x \leq y$ and $f(x) > f(y)$. An *edge* is a pair $(x, y) \in \mathcal{D} \times \mathcal{D}$ with $x < y$ and such that there is no z with $x < z < y$; when $(\mathcal{D}, \leq) = (V_n, \subseteq)$, this is tantamount to saying that $(x, y) \in E_n$. The set of all violated pairs of f is denoted $\mathsf{Viol}(f)$, and the set of all violated edges is denoted $\mathsf{EdgeViol}(f)$. Clearly, the function f is monotone if and only if $\mathsf{Viol}(f) = \mathsf{EdgeViol}(f) = \emptyset$.

We denote by $\varepsilon_M(f) \in [0, 1]$ the relative distance of f from being monotone, i.e., the minimum of $\Pr_{x \in \mathcal{D}}[f(x) \neq g(x)]$ taken over all monotone functions $g \colon \mathcal{D} \to \mathcal{R}$ (the minimum exists even if $\mathcal{R}$ is infinite, but we shall not need this). Let $\delta_M(f) \in [0, 1]$ denote the fraction of edges violated by f; for the hypercube poset this is $|\mathsf{EdgeViol}(f)| / |E_n| = |\mathsf{EdgeViol}(f)| / (n2^{n-1})$.

8.2.1. Definition. A non-empty set $\mathcal{P} \subseteq V_n \times V_n$ of ℓ pairs $\{(s^i, t^i)\}_{i=1}^{\ell}$ is called a *source-sink pairing* (of size ℓ), with *sources* $s^1, \ldots, s^\ell$ and *sinks* $t^1, \ldots, t^\ell$, if

- $s^i \subsetneq t^i$ for all $i \in [\ell]$ and
- $s^i \neq s^j$, $s^i \neq t^j$ and $t^i \neq t^j$ for all $i, j \in [\ell]$, $i \neq j$.

$\mathcal{P}$ is *aligned* if in addition $|s^i| = |s^j|$ and $|t^i| = |t^j|$ for all $i, j \in [\ell]$.

Notice that $\mathcal{P}$ is a source-sink pairing if and only if it forms a (partial) matching in the transitive closure of H_n. Throughout this chapter we denote by $\mathcal{P}$ only sets of pairs that form a source-sink pairing, even when not explicitly mentioned.

A (directed) path in H_n is called a *$\mathcal{P}$-path* if it connects some source s^i from $\mathcal{P}$ to its sink t^i. A subset $C \subseteq E_n$ is called a *$\mathcal{P}$-cut* if every $\mathcal{P}$-path in H_n uses at least one edge from C. Similarly, a subset $S \subseteq V_n$ is called a *$\mathcal{P}$-vertex-cut* if every $\mathcal{P}$-path uses at least one vertex from S. We write $\mathsf{maxflow}(\mathcal{P})$ for the size of the

largest set of edge-disjoint $\mathcal{P}$-paths, $\mathsf{mincut}(\mathcal{P})$ for the size of the smallest $\mathcal{P}$-cut and $\mathsf{minvertexcut}(\mathcal{P})$ for the size of the smallest $\mathcal{P}$-vertex-cut. Clearly $\mathsf{mincut}(\mathcal{P})$ is an upper bound on both $\mathsf{minvertexcut}(\mathcal{P})$ and $\mathsf{maxflow}(\mathcal{P})$. Unlike the case with a single pair in $\mathcal{P}$, the quantities $\mathsf{mincut}(\mathcal{P})$ and $\mathsf{maxflow}(\mathcal{P})$ need not coincide.

We define the terms *sparsity* and *meagerness* as in [RL05], [ABY08], [HKL06]. The *sparsity* of $\mathcal{P}$ is the ratio $\mathsf{mincut}(\mathcal{P})/|\mathcal{P}|$, and the *vertex sparsity* of $\mathcal{P}$ is the ratio $\mathsf{minvertexcut}(\mathcal{P})/|\mathcal{P}|$. In other words, *sparsity* is the average number of edges per source-sink pair that one has to remove to disconnect every source from its sink, whereas *vertex sparsity* is the average number of vertices per source-sink pair that one has to remove to disconnect every source from its sink. The definitions of *meagerness* and *vertex meagerness* are similar, except for the stronger requirement that the corresponding cuts disconnect *all* sources s^i from *all* sinks t^j. The *sparsity* and the *vertex sparsity* of H_n are defined as $\min_{\mathcal{P}}\{\mathsf{mincut}(\mathcal{P})/|\mathcal{P}|\}$ and $\min_{\mathcal{P}}\{\mathsf{minvertexcut}(\mathcal{P})/|\mathcal{P}|\}$, respectively (where $\mathcal{P}$ ranges over all pairings $\mathcal{P} \subseteq V_n \times V_n$).

Observe that

1. sparsity $\geq$ vertex sparsity;
2. meagerness $\geq$ vertex meagerness;
3. meagerness $\geq$ sparsity;
4. vertex meagerness $\geq$ vertex sparsity.

8.3 From sparsity bounds to monotonicity testers

One of the earliest upper bounds on the query complexity of monotonicity testing on the hypercube used an approach based on the concepts of meagerness and sparsity. In particular, Goldreich et al. [GGLR98] observed that if the meagerness of H_n is at least 1, then monotonicity of boolean functions would be testable with $O(n/\varepsilon)$ queries. They reasoned as follows. It can be shown that a function f which is $\Omega(1)$-far from monotone induces a pairing $\mathcal{P}$ of violated pairs of cardinality $\Omega(2^n)$. Being a boolean function, each of these pairs (s^i, t^i) must then satisfy $f(s^i) = 1$ and $f(t^i) = 0$. By transitivity, if $s^i \subseteq t^j$, then there must be some violated edge in any path from s^i to t^j, even if $i \neq j$. If the meagerness of $\mathcal{P}$ is at least one, then there are at least $|\mathcal{P}| = \Omega(2^n)$ edges witnessing the non-monotonicity of f: at least one per path in the optimal set of paths that disconnect all sources from all sinks in $\mathcal{P}$. This would mean that a random edge in H_n belongs to $\mathsf{Viol}(f)$ with probability at least $|\mathcal{P}|/|E_n| = \Omega(1/n)$. Then the following simple algorithm would be a one-sided tester of monotonicity: pick an edge from E_n at random, reject if it is violated, and repeat $O(n)$ times.

What they proved is that vertex meagerness (and hence meagerness too) is 1 if the possible pairings $\mathcal{P}$ are restricted to *aligned* sets, satisfying $|s^i| = |s^j|$ and

$|t^i| = |t^j|$ for all i, j (see also [LR01] for a detailed proof). This was still good enough to derive an upper bound for boolean monotonicity testing of $O(n/\varepsilon)$, and an upper bound of $O(n^2/\varepsilon)$ for general ranges.

While a lower bound on meagerness implies query-complexity upper bounds for boolean functions, a lower bound on sparsity implies query-complexity upper bounds for functions with general range. (In this case we are only guaranteed violations between s^i and t^i but not between s^i and t^j.) In particular, we will presently see that if the sparsity of H_n is at least $\mu = \mu(n)$, then monotonicity of functions with any linearly ordered range can be tested with $O(n/(\varepsilon\mu))$ queries. In [LR01], Lehman and Ron asked whether the sparsity of any $\mathcal{P}$ (or even just of the aligned ones) is at least 1, noting that this would imply the existence of efficient monotonicity testers as well as progress on some long-standing questions regarding routing on the hypercube network. As they wrote, *it appears that a counterexample must be sizable, if one exists at all.* We prove that a counterexample does exist and the answer to both of their questions is **no**.

The basic combinatorial interpretation of $\varepsilon_M(f)$ is given in the following lemma:

8.3.1. Lemma ([FLN+02, Corollary 2];[DGL+99, Lemma 7]). *Let $f\colon \mathcal{D} \to \mathcal{R}$ be a function, and define the* violation graph *of f as the undirected graph $G = (\mathcal{D}, E)$, where $\{x, y\} \in E$ if either (x, y) or (y, x) is in $\mathsf{Viol}(f)$. Then $\varepsilon_M(f)2^n$ is exactly the size of a minimum vertex cover of G. Consequently, there is a matching in G of size at least $\varepsilon_M(f)2^{n-1}$.*

Proof. Let g be monotone and $\varepsilon_M(f)$-close to f, and write $T \triangleq \{x \in \mathcal{D} \mid f(x) \neq g(x)\}$; we have $|T| = \varepsilon_M(f)2^n$. Let C be a minimum vertex cover of the violation graph G. We show that $|C| = |T|$; the "consequently" part then follows from the easy fact that the size of any maximal matching in a graph is at least half the size of the minimum vertex cover, as the endpoints of any maximal matching form a vertex cover.[1]

Clearly T must be a vertex cover of G, otherwise g wouldn't be monotone. Hence $|T| \geq |C|$. To prove $|C| \leq |T|$, we show how that if S is a vertex cover of the violation graph of f (not necessarily smallest), then f can be made monotone by redefining it on S. We proceed by induction on the size of S. The base case, $S = \emptyset$, is trivial. If S is nonempty, take any minimal element $x \in S$ (according to the ordering of $\mathcal{D}$). Consider the sets $x_< \triangleq \{y \in \mathcal{D} \mid y < x\}$ and $x_> \triangleq \{y \in \mathcal{D} \mid y > x\}$. Modifying $f(x)$ only affects the violations occurring among elements of $x_< \cup \{x\} \cup x_>$. Let us define $\tilde{f}$ to be equal to f except on input x, where we let

$$\tilde{f}(x) \triangleq \max\{f(y) \mid y \in x_<\},$$

[1] If the function is boolean, something stronger holds. In this case G is bipartite because violations only occur among pairs $x, y \in \mathcal{D}$ with $x \leq y$, $f(x) = 1$ and $f(y) = 0$. Therefore, by König's Theorem [Die05, Theorem 2.1.1], the size of the maximum matching equals $|C|$, the size of the minimum cover.

where we adopt the convention that the maximum of the empty set is the smallest element of the range $\mathcal{R}$. By definition there are no violations in the new function $\tilde{f}$ between the elements of $x_<$ and the input x. Nor are there any violations between $x_<$ and $x_>$, because $\tilde{f}$ equals f on them and S is a vertex cover of the violation graph of f that does not intersect $x_<$. There cannot be a violation either between x and $x_>$ because by definition, $f(x) = f(y)$ for some $y \in x_<$, which means there would be a violation between $x_<$ and $x_>$ too. Hence $S - \{x\}$ is a vertex cover of the violation graph of $\tilde{f}$, and we are done by induction. □

Regarding functions defined on the hypercube, an important observation is that since G is a subgraph of the transitive closure of H_n, the matching of violated pairs in Lemma 8.3.1 forms a source-sink pairing $\mathcal{P}$ (see Definition 8.2.1) of size $\varepsilon_M(f)2^{n-1}$.

As we mentioned earlier, the best known upper bounds for testing monotonicity over hypercubes are obtained by a simple edge-tester, which picks a set of edges from H_n uniformly at random, queries f on their endpoints, and rejects if one of them is violated. Recall that $\delta_M(f)$ denotes the fraction of edges in H_n that are violated by f; thus the success probability of the edge-tester is determined by $\delta_M(f)$. Goldreich et al. prove the following:

8.3.2. Theorem. *[GGLR98, GGL+00] For any $f\colon \{0,1\}^n \to \{0,1\}$,*

$$\delta_M(f) \geq \frac{\varepsilon_M(f)}{n}.$$

More generally, [DGL+99] use their range-reduction lemma to conclude that for any $f\colon \{0,1\}^n \to \mathcal{R}$, $\delta_M(f) \geq \frac{\varepsilon_M(f)}{n \log |\mathcal{R}|}$. Since without loss of generality $|\mathcal{R}| \leq 2^n$, this gives an upper bound of $O(n^2/\varepsilon)$ queries for testing monotonicity of all functions $f\colon \{0,1\}^n \to \mathcal{R}$.

Clearly, obtaining better lower bounds on $\delta_M(f)$ is sufficient for improving the upper bounds on the query complexity of testing monotonicity. (It may even be the case that Theorem 8.3.2 holds for any $\mathcal{R}$.) The next lemma states that this can also be done by proving lower bounds on the sparsity of H_n.

8.3.3. Lemma. *Let $\mu(n)$ denote the sparsity of H_n. For any $\varepsilon > 0$ and $\mathcal{R} \subseteq \mathbb{Z}$, monotonicity of functions $f\colon \{0,1\}^n \to \mathcal{R}$ can be tested with $O(\frac{n}{\varepsilon\mu(n)})$ queries.*

Proof. Let $\varepsilon > 0$ and let $f\colon \{0,1\}^n \to \mathcal{R}$ be ε-far from monotone. Let $\mathcal{P}$ be the set of $\varepsilon_M(f)2^{n-1} \geq \varepsilon 2^{n-1}$ vertex-disjoint violated pairs promised by Lemma 8.3.1. By definition, $\mathcal{P}$ is a source-sink pairing. Notice that since every $(s^i, t^i) \in \mathcal{P}$ is violated, every path from s^i to t^i must contain at least one violated edge. It follows that the set $\mathsf{EdgeViol}(f)$ is a $\mathcal{P}$-cut and $|\mathsf{EdgeViol}(f)|/|\mathcal{P}| \geq \mu(n)$. Hence $\delta_M(f) = \frac{|\mathsf{EdgeViol}(f)|}{|E_n|} \geq \frac{\varepsilon\mu(n)}{n}$. We can thus conclude that $O\big(\frac{n}{\varepsilon\mu(n)}\big)$ edge queries suffice to find an edge-violation with constant probability. □

8.4 Upper bounds on sparsity

We prove the following theorem.

8.4.1. Theorem (Briët et al. [BCGM12]). *The sparsity of H_n is at most $n^{-\frac{1}{2}+o(1)}$. Furthermore, this upper bound on sparsity can be demonstrated both with aligned sets and with $\Omega(2^n)$-sized sets:*

- *for any $\delta > 0$ and large enough n there is an* aligned *pairing $\mathcal{P}$ in H_n with sparsity at most $n^{-\frac{1}{2}+\delta}$;*
- *for any $\delta > 0$ there is $\varepsilon > 0$, such that for large enough n there is a pairing $\mathcal{P}$ in H_n of size $|\mathcal{P}| \geq \varepsilon 2^n$ with sparsity at most $n^{-\frac{1}{2}+\delta}$.*

We use a number of properties of the structure of perfect binary Hamming codes (see, e.g., [MS77, Lin98]), which we now describe. For an integer $k \geq 1$, let the strings $y \in \{0,1\}^k \setminus \{0\}^k$ represent indices to bit positions of binary strings of length $n = 2^k - 1$. The Hamming code is a linear code consisting of the n-bit strings $x \in \{0,1\}^n$ that, for every $i \in [k]$, have an even number of positions y for which $y_i = 1$ and $x_y = 1$. We are, however, more interested in the properties of the parity check matrix p of the code. This is a $k \times n$ binary matrix whose columns are all possible nonzero k-bit vectors y; it represents a linear map $p\colon \{0,1\}^n \to \{0,1\}^k$ over GF(2). Therefore, for any unit vector e_y (i.e., the element of $\{0,1\}^n$ having 1 at position y and 0 elsewhere), $p(e_y) = y$. Consequently, for all x, y, $p(x \oplus e_y) = p(x) \oplus y$.

Codewords of the Hamming code correspond to strings satisfying $p(x) = 0$ (here and in what follows we use 0 to denote the all-zero vector of the appropriate size). We refer to the k positions of the form 2^i (i.e., $1, 2, 4, \ldots, (n+1)/2$, corresponding to vectors of the form $e_i = 0^{i-1}10^{k-i}$), as the *redundancy bits* of the code; in a codeword x, the k values $x_{e_i}, i = 1 \ldots k$ are determined by the remaining $n - k$ bits of x. Moreover, for general $a \in \{0,1\}^n$, they are determined by the remaining $n - k$ bits and the *parity vector* $p(a)$.

8.4.1 Warm-up

To showcase the main ideas in the construction, we first show that the sparsity of the hypercube is at most $O\big(\frac{1}{n^{1/3}}\big)$; better bounds are derived later in this section.

8.4.2. Proposition. *Let $k > 0$ be a multiple of three, and $n = 2^k - 1$. There is a pairing $\mathcal{P} \subseteq V_n \times V_n$ in H_n of size $|\mathcal{P}| = \Omega(2^n)$ that admits a $\mathcal{P}$-cut $C \subseteq E_n$ of size $|C| = O(2^n/n^{1/3})$.*

Proof. For $a \in \{0,1\}^n$, consider the k parity bits $p(a)$ and divide them into three groups of size $k/3$ each, denoted $x(a), y(a)$ and $z(a)$. For convenience, we will write (v_1, v_2, v_3) to denote the concatenation of three vectors $v_1, v_2, v_3 \in \{0,1\}^{k/3}$,

and whenever no confusion may arise, we interpret every $v \in \{0,1\}^k$ as an element of $\{0\} \cup [n]$. With this convention, we have $p(a) = (x(a), y(a), z(a))$, and if at least one of $v_1, v_2, v_3 \in \{0,1\}^{k/3}$ is nonzero, then $(v_1, v_2, v_3) \in [n]$.

The set S of sources of $\mathcal{P}$ is the set of all $s \in \{0,1\}^n$ that satisfy

$$\Big(x(s) \neq 0 \wedge y(s) \neq 0 \wedge z(s) \neq 0\Big) \wedge \Big(s_{(x(s),y(s),0)} = s_{(x(s),0,z(s))} = s_{(0,y(s),z(s))} = 0\Big).$$

For each source $s \in S$, we define its sink $t = t(s)$ as

$$t = s \cup \{(x(s), y(s), 0), (x(s), 0, z(s)), (0, y(s), z(s))\}.$$

Note t is at distance 3 from s, and the three directions leading from s to t are $(x(s), y(s), 0)$, $(x(s), 0, z(s))$ and $(x(s), 0, z(s))$. The first clause in the conditions on a member s of S ensures that all three directions are (1) distinct; and (2) have a k-bit binary representation with Hamming weight strictly greater than one (in particular they represent proper directions, i.e., they are nonzero). The second clause ensures that the relevant bits of s take the value zero.

The pairing $\mathcal{P}$ will be given by all pairs (s,t) defined in this way. Clearly $s \subseteq t$ and $|t - s| = 3$. It is easy to verify that

$$|S| = \left(2^{k/3} - 1\right)^3 2^{n-k-3} = 2^{n-3} \frac{(2^{k/3} - 1)^3}{2^k} = \Omega(2^n),$$

as follows. There are $(2^{k/3}-1)^3$ ways to pick $x, y, z \in \{0,1\}^{k/3} - \{0\}$. For each such choice, we can construct a source s by letting $s_{(x,y,0)} = s_{(x,0,z)} = s_{(0,y,z)} = 0$ and setting the remaining $n-k-3$ non-redundant bits arbitrarily; there are 2^{n-k-3} ways to accomplish this. (Note none of the directions used corresponds to a redundancy bit, i.e., none of them is a power of 2 because their binary representations have at least two ones.) Finally, the values s takes on the redundancy bits are now determined by the values already set and the condition $p(s) = (x, y, z)$. The last equality also implies that different choices of x, y, z, along with the remaining non-redundant bits, lead to different sources; putting all together we get the equality stated on the size of S.

To prove that $\mathcal{P}$ is a pairing, it remains to show that all sinks are distinct, and that no source is also a sink. Recall that one of the properties of map p is that $p(a \oplus e_{(x,y,z)}) = p(a) \oplus (x, y, z)$. So after flipping, e.g.,bit $(x, y, 0)$ from a source s with parity vector (x, y, z), we reach a vertex with parity vector $(0, 0, z)$. Thus, we see that the parity vectors of the eight vertices in the cube from s to t are:

- Level 3 (sink): (x, y, z).
- Level 2: $(x, 0, 0), (0, y, 0), (0, 0, z)$.
- Level 1: $(0, 0, z), (0, y, 0), (x, 0, 0)$.

- Level 0 (source): (x, y, z).

Observe that, while the source s is different from its sink t, the two parity vectors are the same. Also notice that the parity vectors at level 1 are distinct, as are the parity vectors at level 2.

The three directions from s to t, i.e., the indices of the support of $t - s$, are determined by $p(s) = (x, y, z) = p(t)$. By construction, $t_{(x,y,0)} = t_{(x,0,z)} = t_{(0,y,z)} = 1$, implying $t \notin S$ as it does not meet the requirements for being a source. Likewise, if two different sources s_1 and s_2 were associated with the same sink $t \supset s_1, s_2$, we would get $p(s_1) = p(t) = p(s_2)$, so $t - s_1 = t - s_2$, implying $s_1 = s_2$. Hence $\mathcal{P}$ is indeed a pairing of size $|\mathcal{P}| = |S| = \Omega(2^n)$.

Now we argue that $\mathcal{P}$ admits a small cut. Let Q_s be the set of vertices at level 1 or 2 in the subcube from a source $s \in S$ to its sink t (that is, lying on one of the six paths from s to t and different from s and t). Let $Q = \bigcup_{s \in S} Q_s$. All vertices in Q have parity vectors of one of the forms $(0, 0, z), (0, y, 0), (x, 0, 0)$, hence $|Q| = O(2^n/n^{2/3})$. Now take the set $C \subseteq E_n$ of all (directed) edges of H_n with both endpoints in Q; it is clearly a $\mathcal{P}$-cut. Besides, each vertex of Q is incident with at most $3 \cdot 2^{k/3} = O(n^{1/3})$ edges from C. To see this, consider an arbitrary element of $q \in Q$ with parity vector $p(q) = (x, 0, 0)$, say. It can be incident only with those edges in C that have directions corresponding to vectors of the form $(x, y, 0)$, $(x, 0, z)$ or $(x', 0, 0)$, for various $y, z, x' \in \{0, 1\}^{k/3}$. Since x is fixed for this particular vertex q, there are at most $3 \cdot 2^{k/3} = O(n^{1/3})$ edges in C going out of q. Therefore $|C| \leq |Q| \cdot O(n^{1/3}) = O(2^n/n^{1/3})$, concluding the proof. □

8.4.2 Improved upper bound on the edge sparsity of H_n

In the main construction, we divide the length-k strings into m equally-sized parts, we let d be the distance between pairs in the pairing and w be the number of nonzero length-(k/m) parts of the parity strings of the direction vectors. The main tool is the following lemma about certain sets of vectors used to generalize the proof in the warm-up. The reader should keep in mind that an example of such a set of vectors for $m = 3$, $d = 3$, $w = 2$, is $V = \{110, 101, 011\}$, and was implicitly used in the previous proof.

For our purposes, all parameters involved except k and n should be thought of as constants.

8.4.3. Lemma. *Suppose $V \subseteq \{0, 1\}^m$, $d = |V| > 0$, and $w \in \mathbb{N}$ are such that:*

1. *$2 \leq |v| \leq w$ for all $v \in V$,*
2. *$\bigoplus_{v \in V} v = 0$, and*
3. *for all $W \subseteq V$ of size $|W| = \lfloor d/2 \rfloor$, $\left|\bigoplus_{v \in W} v\right| \geq \lceil m/2 \rceil$.*

Let $k \geq m \log m$ be a positive multiple of m and $n = 2^k - 1$. Then there is a pairing $\mathcal{P} \subseteq V_n \times V_n$ of vertices of H_n of size

$$|\mathcal{P}| \geq \frac{1}{4} 2^{n-d}$$

that has a $\mathcal{P}$-cut $C \subseteq E_n$ of size

$$|C| \leq \frac{2^n}{\sqrt{n+1}} \, (n+1)^{w/m} \, \sqrt{d} \, 2^d$$

and with the additional property that each source in $\mathcal{P}$ is at distance exactly d from its sink.

Proof. Partition $[k]$ into m disjoint subsets $G_1, \ldots, G_m \subseteq [k]$ of size k/m; e.g., $G_i = \{(i-1)k/m + 1, \ldots, ik/m\}$. For $a \in \{0,1\}^n$, consider the k parity bits $p(a) \in \{0,1\}^k$ of a, and split them into m blocks according to $G_1, \ldots, G_m$;[2] let us call each of the corresponding k/m-bit substrings $x_1(a), \ldots, x_m(a)$. Thus, $p(a)$ is the concatenation of $x_1(a), x_2(a), \ldots, x_m(a)$.

For a subset $v \subseteq [m]$, let $Z_v = \bigcup_{i \in v} G_i \subseteq [k]$. Given $r \subseteq [k]$, define the *projection* of r on v to be $\Pi_v(r) = r \cap Z_v$, (remember that r and $\Pi_v(x)$ can be interpreted as strings in $\{0,1\}^k$ as well). For example, in the preceding subsection, we would write $\Pi_{110}((x,y,z)) = (x,y,0)$. Consider the set

$$S \triangleq S_V = \{a \in \{0,1\}^n \mid \forall i \in [m] : \; x_i(a) \neq 0 \text{ and } \forall_{v \in V} \; a_{\Pi_v(p(a))} = 0\}.$$

This will be our set of sources in $\mathcal{P}$. Note that the expression $a_{\Pi_v(p(a))}$, referring to bit number $\Pi_v(p(a))$ of a, is well-defined, because the condition $\forall i : x_i(a) \neq 0$, along with $v \neq 0$, implies $\Pi_v(p(a)) \neq 0$. Moreover, $\Pi_v(p(a)) \neq \Pi_w(p(a))$ for $v \neq w$.

The set of d directions between a source s and the corresponding sink t will be determined by the parity vector of s alone. This set will be $D(p(s))$ for a function D defined in the following way: for $r \in \{0,1\}^k$, let $D(r) = \bigcup_{v \in V} \{\Pi_v(r)\}$. Condition 1 in the hypothesis of the lemma implies that for $s \in S$, all elements of $D(p(s))$ have weight ≥ 2; note also that s is disjoint with $D(p(s))$ and that $|D(p(s))| = |V| = d$. For each source $s \in S$, we define the sink $t = s \cup D(p(s))$; by construction $s \subseteq t$, and $|t - s| = |D(p(s))| = d$. The cut $\mathcal{P}$ is defined as the union of all such ordered pairs (s,t): $\mathcal{P} \triangleq \bigcup_{s \in S} \{(s, s \cup D(p(s)))\}$. Note that

$$|\mathcal{P}| = |S| = \left(2^{k/m} - 1\right)^m 2^{n-k-d}.$$

We can bound

$$n + 1 = 2^k \geq (2^{k/m} - 1)^m = \left(1 - \frac{1}{2^{k/m}}\right)^m \cdot 2^k \geq \frac{1}{4} 2^k, \tag{8.1}$$

[2] Actually, in order to do this we first impose an arbitrary ordering on the elements of each G_i.

since $k \geq m \log m$ and $m \geq 2$. Hence

$$|\mathcal{P}| = |S| \geq \frac{1}{4} 2^{n-d}.$$

We prove now that $\mathcal{P}$ forms a pairing: the set of sinks is disjoint from the set of sources, and no two different sources have the same sink. Because of the aforementioned properties of the parity check p, for any source-sink pair (s, t) we have

$$p(t) = p(s) \oplus \bigoplus_{v \in V} \Pi_v(p(s)) = p(s) \oplus \Pi_{\bigoplus_{v \in V}}(p(s)) = p(s),$$

where we used the second property of V and simple properties of the projection operator. Since for every $i \in D(p)$, $i \notin s$ but $i \in t$, it follows that no sink is a source too. Likewise, if two sinks t_1 and t_2 (corresponding to sources s_1 and s_2) were the same ($t_1 = t_2$), we would have $p(s_1) = p(s_2)$, which implies $D(p(s_1)) = D(p(s_2))$ and therefore $s_1 = s_2$.

To conclude, we only need to upper-bound the size of a smallest $\mathcal{P}$-cut. Consider the set of vertices halfway between a source and a sink:

$$Q \triangleq \{x \in \{0,1\}^n \mid \exists\, (s,t) \in \mathcal{P} \text{ s.t. } s \subseteq x \subseteq t \text{ and } |x - s| = \lfloor d/2 \rfloor\}$$

(note the slightly different definition of Q, compared to that in the warmup at the start of the section).

The set Q is a $\mathcal{P}$-vertex-cut; we use it to construct an edge cut for $\mathcal{P}$. Due to the third property of V and the definition of $D(p(s))$, it follows that for $b \in Q$, at least half of $x_1(b), \ldots, x_m(b)$ are zero. For any $b \in \{0,1\}^n$, let $r(b)$ be the m-bit string such that for all $i \in [m]$, the equality $x_i(b) = 0^{k/m}$ holds iff $r(b)_i = 0$. Then the set $\{r(b) \mid b \in Q\}$ has size bounded by $\binom{d}{d/2}$: for all $s \in S$, $r(s)$ is the all-ones string and for any $b \in Q$, $r(b)$ is $r(s)$ XOR-ed with some $d/2$ vectors in V. So the set $\{p(b) \mid b \in Q\}$ has size at most $\binom{d}{d/2}(2^{k/m} - 1)^{m/2}$, and does not contain unit vectors; therefore using (8.1) we obtain

$$|Q| \leq 2^{n-k} \binom{d}{\lfloor d/2 \rfloor} (2^{k/m} - 1)^{m/2} \leq \frac{2^n}{\sqrt{n+1}} \frac{2^d}{\sqrt{d}}.$$

An edge cut is given by

$$C \triangleq \{(b, c) \in E_n \mid b \in Q \wedge c - b \in D(p(S))\},$$

where $D(p(S)) = \bigcup_{s \in S} \{D(p(s))\}$. To verify this, observe that any path from a source s to its sink t must go through some vertex $q \in Q$ at distance $\lfloor d/2 \rfloor$ from s, and then take one of the directions in the set $D(p(s))$ (or else the sink would not be reachable from the next vertex in the path). Finally, observe that

$$|D(p(S))| \leq d(2^{k/m} - 1)^w \leq d(n+1)^{w/m},$$

because every element of $D(p(s))$ is determined by the projection of $p(s)$ on some $v \in V$ (with weight at most $|v| \leq w$). The claim follows from our bounds on $|Q|$ and $|D(p(s))|$, since $|C| \leq |Q| \cdot |D(p(S))|$. □

8.4.4. Lemma. *Let $w \in \mathbb{N}, w \geq 2$ and set $m = w^2$, $d = 2w$. Then there is a set $V \subseteq \{0,1\}^m$ of d vectors satisfying the three conditions in Lemma 8.4.3.*

Proof. Arrange the w^2 elements of $[m]$ into a square matrix $A \in \{0,1\}^{w \times w}$. For each row and each column of A we form an element of $\{0,1\}^w$ whose support is that row or column (there are $2w$ vectors in total). The i-th row is then associated with the subset (or vector in $\{0,1\}^w$)

$$R_i \triangleq \{r \in [m] \mid (i-1)w < r \leq iw\};$$

the j-th column will correspond to the subset

$$C_j \triangleq \{r \in [m] \mid (r-1) \bmod w = j-1\}.$$

Let

$$V \triangleq \bigcup_{i \in [w]} \{R_i, C_i\}.$$

Clearly, $|V| = 2w$ and for all $v \in V$, we have $|v| = w > 1$. It is also apparent that $\bigoplus_{v \in V} v = 0$, because any $k \in [m]$ belongs to exactly two vectors in V, namely R_i and C_j, where $k = (i-1)w + j$ with $i, j \in [w]$. (This is a rephrasing of the fact that every entry of the matrix A is in the intersection of precisely one row and one column.)

Finally, we show that, for any $W \subseteq V$ with $|V| = d/2 = w$,

$$\left| \bigoplus_{v \in W} v \right| \geq \frac{m}{2} = \frac{w^2}{2}.$$

Suppose that

$$W = \{R_{i_1}, R_{i_2}, \ldots, R_{i_a}, C_{j_1}, C_{j_2}, \ldots, C_{j_{(w-a)}}\};$$

then

$$\left| \bigoplus_{v \in W} v \right| = a^2 + (w-a)^2 \geq \frac{w^2}{2}$$

by the quadratic mean-arithmetic mean inequality. □

Proof of Theorem 8.4.1. Let $\delta \in (0,1)$, $n > (4/\delta^2)^{4/\delta^2+1}$ and set

$$w \triangleq \lceil 1/\delta \rceil, m \triangleq w^2, d \triangleq 2w.$$

Let k be the largest multiple of m which is at most $\log(n+1)$. Note that $w \leq 2/\delta, n > m^{m+1}$ and

$$k > \log(n+1) - m \geq \log(n/m) \geq m \log m.$$

Now set $n_0 \triangleq 2^k - 1 > n/2^{m+1}$. By Lemmas 8.4.4 and 8.4.3, there is a pairing $\mathcal{P}_{n_0} \in V_{n_0} \times V_{n_0}$ of size 2^{n_0-d-2} which admits a cut C_{n_0} of size

$$\frac{2^{n_0}}{\sqrt{n_0+1}} \, (n_0+1)^{1/w} \, \sqrt{d}\, 2^d$$

and thus H_{n_0} has sparsity at most

$$\frac{|C_{n_0}|}{|\mathcal{P}_{n_0}|} \leq (n_0+1)^{\frac{1}{w}-\frac{1}{2}} \cdot 2^{2(d+1)}\sqrt{d} \leq \frac{1}{\sqrt{n}} \cdot n^{\delta} \cdot 2^{O\left(\frac{1}{\delta^2}\right)}.$$

Observe that H_n can embed 2^{n-n_0} disjoint copies of H_{n_0}, for example, according to each of the settings of the last $n-n_0$ bits of a vertex label. We can thus embed 2^{n-n_0} copies of the pairing and its cut, and still obtain a pairing of size $\Omega(2^{n-d})$ and a cut of sparsity $n^{-1/2+\delta} \cdot 2^{O(1/\delta^2)}$. This may look slightly weaker than the second part of the theorem, but it implies it by choosing for example $\delta_0 = \delta/2$ and $n = 2^{\Omega(1/\delta^3)}$.

For the first part, recall that the pairing $\mathcal{P}$ we obtain has the additional property that all pairs in $\mathcal{P}$ have distance exactly d. Knowing this, the first part of Theorem 8.4.1 can be proved in the following way. Let C be a $\mathcal{P}$-cut of sparsity $\leq n^{-1/2+\delta}$. Partition the pairs of $\mathcal{P}$ into $d+1$ parts $\mathcal{P}_1, \ldots, \mathcal{P}_{d+1}$ according to the Hamming weight of their source modulo $d+1$. At least one of them, say $\mathcal{P}_i$, has size $\geq |\mathcal{P}|/(d+1)$, so C is a $\mathcal{P}_i$-cut with sparsity $\leq (d+1) \cdot n^{-1/2+\delta}$. If $(s^j, t^j), (s^k, t^k) \in \mathcal{P}_i$ and $|s^j| \neq |s^k|$, then $|s^j| - |s^k|$ is a multiple of $d+1$, and since $|t^j - s^j| = |t^k - s^k| = d$, we conclude that no edge in any path from s^j to t^j lies in a path from s^k to t^k as well.

Now, for $j = 0, \ldots, \lceil (n-i)/(d+1) \rceil - 1$, let

$$A_j = \{x \in \{0,1\}^n \mid j(d+1) \leq |x| - i < (j+1)(d+1)\}.$$

It follows that we can partition $\mathcal{P}_i$ into $\lceil n/d \rceil$ parts, according to which set A_j their sources (and sinks) belong to. Moreover, there is a subset $C' \subseteq C$ that is a $\mathcal{P}_i$-cut and only contains edges both of whose endpoints are inside the same A_j (possibly different for different pairs), i.e., $C' = \bigcup C'_j$ where $C'_j \subseteq A_j \times A_j$. (The reason is that we can safely remove any edges that do not satisfy this condition.) From $\sum_j |C_j| = |C'|$ and $\sum_j |P_i \cap (V_j \times V_j)| = |\mathcal{P}_i|$, it follows that there is some j with $|C_j| \,/\, |\mathcal{P}_i \cap (V_j \times V_j)| \leq |C'|/|\mathcal{P}| \leq |C|/|\mathcal{P}|$, i.e., there is an *aligned* pairing $\mathcal{P}_i \cap (V_j \times V_j)$ with sparsity at most $(d+1) \cdot n^{-1/2+\delta}$. Again, choosing a smaller δ proves the claim. □

8.4.3 Upper bound on the vertex sparsity of H_n

While it is edge sparsity that bears the strongest relationship with monotonicity testing, it is natural to study the related quantity of vertex sparsity as well. Here we present a simple result on the vertex sparsity of H_n.

8.4.5. Theorem (Briët et al. [BCGM12]). *The vertex sparsity of H_n is $O(1/n)$.*

Proof. Let $n \geq 4$ be even. We construct an aligned source-sink pairing $\mathcal{P} \subseteq V_n \times V_n$ of $2^{n/2}(n/4)$ disjoint pairs (s^i, t^i), such that for all i, j, $|t^j| = |s^i| + 2$. Then we exhibit a $\mathcal{P}$-vertex-cut M of size $2^{n/2}$.

Consider the following set of pairs:

$$\mathcal{P}_0 = \{(0001, 1011), (1000, 1101), (0010, 0111), (0100, 1110)\}.$$

Any $\mathcal{P}_0$-path (of length 2) goes through one of the following vertices:

$$M_0 = \{1001, 0011, 1100, 0110\}.$$

Using this small example, we construct the large one recursively. For $i \geq 1$, we set

$$\mathcal{P}_i = \{(01a, 01b) \mid (a, b) \in \mathcal{P}_{i-1}\} \cup \{(10a, 10b) \mid (a, b) \in \mathcal{P}_{i-1}\} \cup \{(00a, 11a) \mid a \in M_{i-1}\}.$$

Let M_i denote the set of all internal vertices that lie on some $\mathcal{P}_i$-path. Notice that $M_i = \{01a \mid a \in M_{i-1}\} \cup \{10a \mid a \in M_{i-1}\}$. So we have:

- $|\mathcal{P}_i| = 2|\mathcal{P}_{i-1}| + |M_{i-1}|$;
- $|M_i| = 2|M_{i-1}|$.

Solving these recurrence relations we get $|M_i| = 2^i|M_0|$ and $|\mathcal{P}_i| = 2^i|\mathcal{P}_0| + i2^{i-1}|M_0|$. Let $\mathcal{P} \triangleq \mathcal{P}_{(n-4)/2} \subseteq V_n \times V_n$ and $M \triangleq M_{(n-4)/2} \subseteq V_n$. Since M is a $\mathcal{P}$-vertex-cut by definition, we only need to show that the sizes of M and $\mathcal{P}$ are as claimed. Indeed, $|M| = 2^{(n-4)/2}|M_0| = 2^{n/2}$ and

$$|\mathcal{P}| = 2^{(n-4)/2}|\mathcal{P}_0| + 2^{(n-4)/2-1}\left(\frac{n-4}{2}\right)|M_0| = 2^{n/2}(n/4).$$

□

8.5 Interlude: Routing on the hypercube

The hypercube is a natural and well-studied architecture for multiprocessor systems and networks. The ability to route arbitrary permutations on it models flow of information in a network of processors. In this context, a doubly-directed version of H_n is usually considered, where each edge in E_n is replaced with a pair of anti-parallel edges. Let us denote the doubly-directed version of H_n by $H_n^{\uparrow\downarrow}$. A permutation π of V_n is 1-*realizable* if there exists a set of pairwise edge-disjoint paths in $H_n^{\uparrow\downarrow}$ such that for every v, there is a path in the set that connects v with $\pi(v)$. Similarly, a permutation π is k-*realizable* if there exist paths connecting every v with $\pi(v)$ such that each edge is used in at most k paths. In terms of the circuit-switching capability of interconnection networks this means the following. If a computer is located at every node of the directed hypercube and two neighboring vertices can communicate and send at most k messages simultaneously, then for any permutation π it is possible to set up connection paths allowing us to send messages from every computer v to its addressee $\pi(v)$ without needing to send more than k simultaneous messages between any pair of computers. Szymanski [Szy89] conjectured that any permutation π of V_n is 1-realizable with *shortest paths*. It was proved that the conjecture holds up to dimension 3, but later Lubiw [Lub90] provided a counterexample in dimension 5 that is not 1-realizable using shortest paths. While it is still unknown whether or not every permutation is 1-realizable *without* requiring shortest paths[3], the fact that any permutation is 2-realizable follows from the classical work of Beneš [Ben65] (see [Lub90] for details). In contrast, we prove that if we insist on the shortest-path condition, there are permutations that are not k-realizable for any k significantly smaller than $\sqrt{n}$. Specifically, the construction of Theorem 8.4.1 can be used to prove the following.

8.5.1. Theorem (Briët et al. [BCGM12]). *For any $\delta > 0$ and large enough n, there are permutations of V_n that cannot be $n^{\frac{1}{2}-\delta}$-realized in $H_n^{\uparrow\downarrow}$ with shortest paths.*

Proof. Let $\mathcal{P}$ and C be the pairing and cut constructed in the proof of Theorem 8.4.1. Let π be any permutation of V_n that maps each source in $\mathcal{P}$ to its sink. Notice that any shortest path in $H_n^{\uparrow\downarrow}$ that connects a source of $\mathcal{P}$ to its sink must also be a directed path in H_n, because the elements on each pair are comparable. So such a path must intersect C. Hence any realization of $\mathcal{P}$ with shortest paths must use some edge in C at least $|\mathcal{P}|/|C| = n^{1/2-\delta}$ times. □

8.5.2. Remark. Any upper bound $\mu(n)$ on the sparsity of H_n can be used to show that $H_n^{\uparrow\downarrow}$ is not $1/\mu(n)$-realizable with shortest paths. But the opposite is

[3]Since the original conjecture was shown to be false, the weaker version not requiring shortest paths is now sometimes called Szymanski's conjecture.

not true; in particular, the counterexample from [Lub90] does not imply that the sparsity of H_5 is less than 1.

8.6 New bounds on testing monotonicity

At the time of writing the best known query-complexity bounds for testing monotonicity (non-adaptively with one-sided error) of functions $f\colon \{0,1\}^n \to \mathcal{R}$ were:

- an upper bound of $O(\frac{n}{\varepsilon}\log|\mathcal{R}|)$ for any range $\mathcal{R}$, by Dodis et al. [DGL+99];
- a lower bound of $\Omega(\sqrt{n}/\varepsilon)$ for boolean ranges (and hence for wider ranges too), by Fischer et al. [FLN+02].

The tester used in the upper bound of [DGL+99] is perhaps the most natural one: it picks an edge $(x,y) \in E_n$ uniformly at random, and rejects if $f(x) > f(y)$. Let us call this an *edge-test*. [DGL+99] prove that the probability that a single execution of an edge-test rejects is $\Omega\big(\frac{\varepsilon_M(f)}{n\log|\mathcal{R}|}\big)$, by relating the distance of a function from monotone to the number of edges that it violates.

It is an interesting open question whether the general upper bound of [DGL+99] can be improved into one that is independent of $|\mathcal{R}|$ (or at least has a better dependence on it). Since we can assume without loss of generality that $|\mathcal{R}| \le 2^n$, any upper bound of $o(n^2/\varepsilon)$ queries would be an improvement. We make a small step in this direction. Call a function $f\colon \{0,1\}^n \to \mathcal{R}$ *dist-k monotone* if $f(y) \ge f(x)$ for every $y > x$ with $|y| > |x| + k$. In this terminology, "dist-0 monotone" simply means "monotone".

In Section 8.6.1 we prove that given a dist-3 monotone function f, we can test if f is monotone with $O(n^{3/2}/\varepsilon)$ queries. The reasons for considering dist-3 monotonicity here are twofold. Firstly, it is the first non-trivial case: It is easy to see that the sparsity of any pairing contained in the set of violated pairs of a dist-2 monotone function is one, because to disconnect any pair of points at distance 2 from each other at least two edges must be removed, but in this case any edge can only be involved in two pairs (and a similar argument takes care of the pairs at distance one too). This implies (see Section 8.3) that 2-monotone functions can be tested for monotonicity with $O(n/\varepsilon)$ queries. Secondly, we saw in Section 8.4 that non-trivial sparsity upper bounds can already hold for pairings in which every source is at distance 3 from its sink.

In Section 8.6.2 we also extend the lower bound of $\Omega(\sqrt{n}/\varepsilon)$ of [FLN+02] to $\Omega(n/\varepsilon)$, for large enough $|\mathcal{R}|$. Using the "Range-Reduction Lemma" of [DGL+99], the new bound implies an improved lower bound of $\Omega(n/(\varepsilon\log n))$ for the boolean range, in the special case of pair testers whose query complexity can be written as $q(n)/\varepsilon$ for some function q. (A *pair tester* picks independent pairs of comparable vertices according to some distribution, and rejects if and only if one of the pairs

forms a violation.) We note that such testers are not overly restricted: essentially all known query-complexity upper bounds for monotonicity testing use (or can be easily converted into ones that use) pair tests of this kind. Furthermore, the new lower bound almost matches the aforementioned upper bound of $O(n/\varepsilon)$ achieved by edge-tests (a special case of pair tests).

Section 8.6.3 describes the recent work of Blais et al., who used a clever reduction to well-studied problems in communication complexity to show that the $\Omega(n/\varepsilon)$ query complexity lower bound holds even for adaptive, two-sided-error testers of monotonicity.

8.6.1 Sparsity and dist-3-monotone functions

8.6.1. Theorem (Briët et al. [BCGM12]). *Let $\varepsilon > 0$, $\mathcal{R} \subseteq \mathbb{Z}$ and consider a dist-3 monotone function $f\colon \{0,1\}^n \to \mathcal{R}$. If f is ε-far from being monotone then $|\mathsf{EdgeViol}(f)| \geq \Omega\left(\frac{2^n}{\varepsilon\sqrt{n}}\right)$.*

Proof. Let $\varepsilon > 0$, $\mathcal{R} \subseteq \mathbb{Z}$ and consider a dist-3 monotone function $f\colon \{0,1\}^n \to \mathcal{R}$. If f is ε-far from being monotone, then by Lemma 8.3.1 there is a set $\mathcal{P}$ of $\varepsilon 2^{n-1}$ vertex-disjoint pairs in H_n that are violated by f. Furthermore, since f is dist-3 monotone, for every $(s^i, t^i) \in \mathcal{P}$ we have $|t^i| \leq |s^i| + 3$. To prove Theorem 8.6.1 we show that the sparsity of such $\mathcal{P}$ must be $\Omega(1/\sqrt{n}\,)$.

Let C be a smallest $\mathcal{P}$-cut, and let us prove that $|C|/|\mathcal{P}| = \Omega(1/\sqrt{n}\,)$. There is nothing to prove if $|C| \geq |\mathcal{P}|/2$, so assume the opposite. It is possible to assume further that C has *no* edges incident with any source s^i or sink t^j from $\mathcal{P}$ (and in particular, this will mean that no pair in $\mathcal{P}$ has distance 1 or 2): Let $p \leq |C| < |\mathcal{P}|/2$ be the number of edges in C that are incident to some source or sink of a pair in $\mathcal{P}$. Removing these p edges from C and the corresponding pairs from $\mathcal{P}$ leaves a set C' of size $|C| - p \geq 0$ that cuts a subset $\mathcal{P}' \subseteq \mathcal{P}$ of at least $|\mathcal{P}| - 2p > 0$ pairs. This is due to the fact that the pairs in $\mathcal{P}$ are disjoint, and hence each edge can be incident with at most two pairs. We have

$$(|C| - p)|\mathcal{P}| = |C||\mathcal{P}| - |\mathcal{P}|p \leq |C||\mathcal{P}| - 2|C|p = (|\mathcal{P}| - 2p)|C|,$$

so the sparsity of the $\mathcal{P}'$-cut C' is

$$\frac{|C'|}{|\mathcal{P}'|} = \frac{|C| - p}{|\mathcal{P}| - 2p} \leq \frac{|C|}{|\mathcal{P}|}.$$

Therefore it is enough to prove the claim for $C \triangleq C'$ and $\mathcal{P} \triangleq \mathcal{P}'$.

For $0 \leq h \leq n-3$, let $\mathcal{P}^h \subseteq \mathcal{P}$ be the set of pairs $(s^i, t^i) \in \mathcal{P}$ with $|s^i| = h$ (and $|t^i| = h + 3$). Clearly C is a $\mathcal{P}^h$-cut for every h. Let $C^h \subseteq C$ denote the set of edges in C that lie on some $\mathcal{P}^h$-path. Since C^h has no edges incident with any s^i or t^j, in order to cut $\mathcal{P}^h$ we must use exactly those edges between levels $h+1$

and $h+2$ that lie on some $\mathcal{P}^h$-path. So the sets C^h, $0 \le h \le n-3$, are in fact disjoint. Therefore it is sufficient to prove that $C^h/|\mathcal{P}^h| = \Omega(1/\sqrt{n})$ for all h.

Fix h. Each pair $(s^i, t^i) \in \mathcal{P}^h$ defines a (directed) subcube graph $H_3^i = (V(H_3^i), E(H_3^i))$ of dimension 3, This subcube contains all vertices and edges that belong to one of the six possible paths from s^i to t^i.

8.6.2. Lemma. *For any two pairs $(s^i, t^i), (s^j, t^j) \in \mathcal{P}^h$, $|E(H_3^i) \cap E(H_3^j)| \le 1$.*

Proof. Assume that $|E(H_3^i) \cap E(H_3^j)| \ge 2$ for some $i \ne j$, and let $e = (a, b)$ and $e' = (a', b')$ be two edges in $E(H_3^i) \cap E(H_3^j)$. Since the pairs (s^i, t^i) and (s^j, t^j) are disjoint, both e and e' should connect layers $h+1$ and $h+2$. Therefore, $a = a' = s^i \cup s^j$ and $b = b' = t^i \cap t^j$, contradicting the assumption that $e \ne e'$. □

Consider the directed graph $G^h = (V^h, E^h)$ with $V^h = \bigcup_{(s^i,t^i)\in\mathcal{P}^h} V(H_3^i)$ and $E^h = \bigcup_{(s^i,t^i)\in\mathcal{P}^h} E(H_3^i)$. Since every s^i has out-degree 3 in G^h (and in-degree 0), the number of edges between layers h and $h+1$ of H_n that belong to G^h is exactly $3|\mathcal{P}^h|$. Let $A = \{a_1, \dots, a_k\}$ be the set of vertices in layer $h+1$ of H_n that belong to G^h, let $\alpha_1, \dots, \alpha_k$ denote their in-degrees in G^h and let $\beta_1, \dots, \beta_k$ denote their out-degrees in G^h. We have $\sum_{i\in[k]} \alpha_i = 3|\mathcal{P}^h|$, and our goal is to prove that $|C^h| = \sum_{i\in[k]} \beta_i = \Omega(|\mathcal{P}^h|/\sqrt{n})$.

Consider vertex $a_i \in A$. For every pair $(s^j, t^j) \in \mathcal{P}^h$ such that $a_i \in V(H_3^j)$ there are two edges in H_3^j going out of a_i. Since for any two pairs $(s^j, t^j), (s^{j'}, t^{j'}) \in \mathcal{P}^h$ we have $|E(H_3^j) \cap E(H_3^{j'})| \le 1$, it follows that the source-sink pair (s^j, t^j) is completely determined by the vertex a_i and the two possible edges to take from a^i to reach the sink. In other words, for any $a_i \in A$, any pair of two distinct outgoing edges from a_i "encodes" a unique ingoing edge. This implies $\binom{\beta_i}{2} \ge \alpha_i$. So $\beta_i > \sqrt{2\alpha_i}$ for all i and hence

$$|C^h| = \sum_{i\in[k]} \beta_i > \sum_{i\in[k]} \sqrt{\alpha_i} = \sum_{i\in[k]} \frac{\alpha_i}{\sqrt{\alpha_i}} \ge \frac{3|\mathcal{P}^h|}{\sqrt{n}},$$

as $\alpha_i \le n$. □

8.6.2 A lower bound for general functions

8.6.3. Theorem (Briët et al. [BCGM12]). *Let $\mathcal{R} \subseteq \mathbb{Z}$, $|\mathcal{R}| > 2\sqrt{n}$. Testing monotonicity of functions $f\colon \{0,1\}^n \to \mathcal{R}$ non-adaptively with one-sided error requires $\Omega(n/\varepsilon)$ queries.*

Proof. We first prove a lower bound of $\Omega(n)$ for some constant ε and argue at the end how we can achieve the promised lower bound of $\Omega(n/\varepsilon)$.

A non-adaptive q-query monotonicity tester with one-sided error queries f on a set Q of at most q vertices and rejects if and only if one of the comparable

pairs in Q is violated. Hence, it is sufficient to show a family $\mathcal{F}_n$ of functions $f\colon \{0,1\}^n \to \mathcal{R}$ that are ε-far from monotone (for a fixed $\varepsilon > 0$ and all n) and such that, for any fixed set $Q \subseteq \{0,1\}^n$ of size $o(n)$, a uniformly random $f \sim \mathcal{F}_n$ induces a violated pair in Q with probability less than $1/3$.

For every n, we will define a family $\mathcal{F}_n = \{f_1, \ldots, f_n\}$ of n functions

$$f_i\colon \{0,1\}^n \to \mathcal{R}$$

with the following properties:

- every f_i is ε-far from monotone, for some absolute constant $\varepsilon > 0$;
- for any set $Q \subseteq \{0,1\}^n$, $\Pr_{i\in[n]}[(Q \times Q) \cap \mathsf{Viol}(f_i) \neq \emptyset] \leq \frac{|Q|-1}{n}$.

This implies any tester making fewer than $\frac{2n}{3}$ queries will fail with probability $> 1/3$.

As in the proof of the lower bound for boolean functions given in [FLN+02], each $f_i \in \mathcal{F}_n$ will violate only pairs differring in the i-th coordinate. But unlike their construction, where distant vertices may cause violations, ours will take advantage of the larger range size to make sure that only the actual *edges* of H_n are violated, making it more difficult to catch violated pairs.

We formally define $\mathcal{F}_n$ formally. Without loss of generality, let the range be $\mathcal{R} = \{0, 1, \ldots, 2\sqrt{n}\}$ (the labels can be chosen freely, and a lower bound for range $\mathcal{R}$ also holds for ranges containing $\mathcal{R}$). Let $h(x) \triangleq |x| - n/2 + \sqrt{n}$ for all $x \in \{0,1\}^n$. For each $i \in [n]$ we define $f_i\colon \{0,1\}^n \to \mathcal{R}$ as follows:

$$f_i(x) = \begin{cases} 0, & h(x) < 0 \\ 2\sqrt{n}, & h(x) > 2\sqrt{n} \\ h(x), & h(x) \in \mathcal{R} \text{ and } x_i \neq h(x) \bmod 2 \\ h(x) + (-1)^{x_i}, & h(x) \in \mathcal{R} \text{ and } x_i = h(x) \bmod 2. \end{cases}$$

Observe that for all $i \in [n]$, $\mathsf{Viol}(f_i) = \mathsf{EdgeViol}(f_i)$ (all violated pairs in f_i are neighbouring vertices of the hypercube), and the edges in $\mathsf{EdgeViol}(f_i)$ are vertex-disjoint. So by Lemma 8.3.1, the functions $f_i \in \mathcal{F}_n$ are ε-far from monotone (for some fixed $\varepsilon > 0$) if $|\mathsf{EdgeViol}(f_i)| \geq \varepsilon 2^n$. Indeed, $|\mathsf{EdgeViol}(f_i)|$ equals the number of points $x \in \{0,1\}^n$ such that: $h(x) \in \mathcal{R}$, $h(x) \equiv 0 \pmod 2$ and $x_i = 0$. Notice that for $n > 10$, these constitute roughly a quarter of all points $y \in \{0,1\}^n$ with $h(y) \in \mathcal{R}$. On the other hand, it follows from Chernoff bounds that for some constant $\rho > 0$ and for all $n > 10$, the number of points $y \in \{0,1\}^n$ with $h(y) \in \mathcal{R}$ is at least $\rho 2^n$. Setting $\varepsilon = \rho/5$, we conclude that all functions $f_i \in \mathcal{F}_n$ are ε-far from monotone.

Now we prove that $\Pr_{i\in[n]}[(Q \times Q) \cap \mathsf{Viol}(f_i) \neq \emptyset] \leq \frac{|Q|-1}{n}$. Fix Q and consider the *undirected* graph $G = (V, E)$, where $V = Q$ and

$$E = \Big\{\{x, y\} \in Q \times Q \mid (x, y) \in E_n\Big\}.$$

In other words, G is the undirected skeleton of the subgraph of H_n induced on Q. For $x, y \in \{0,1\}^n$ we write $x = y^{(j)}$ if x equals y in all coordinates except j. Let $T \subseteq [n]$ be a set of directions spanned by E, namely,

$$T = \{j \in [n] \mid \text{there exists } \{x, y\} \in E \text{ such that } x = y^{(j)}\}.$$

Clearly, the success probability of the test is bounded by $|T|/n$. To finish the proof, we show that $|T| \leq |Q| - 1$.

Consider a minimal subgraph G' of G that spans all directions in T. Then $|E(G')| = |T|$. Since any cycle in the undirected skeleton of H_n travels along any direction an even number of times, G' is acyclic. So $|T| = |E(G')| \leq |V(G')| - 1 = |Q| - 1$.

We proved a lower bound of $\Omega(n)$ queries for some constant $\varepsilon > 0$. To get a lower bound of $\Omega(n/\varepsilon)$ for any $\varepsilon = \varepsilon(n)$ we need to compose our lower bound with a simple "hiding" procedure. Namely, we define a distribution $\mathcal{F}'_n$ that fools any deterministic tester with $o(n/\varepsilon)$ queries as follows: first, partition H_n into disjoint subcubes, each of size $\varepsilon 2^n$ (for simplicity we assume that $1/\varepsilon$ is a power of two); then pick a random subcube C in this partition, and value it with a random $f_i \in \mathcal{F}_{n-\log 1/\varepsilon}$; value the other subcubes so that there are no violations outside C. Now for any fixed set Q of $o(n/\varepsilon)$ queries, the expected number of queries that hit C is $o(n)$, and we know that with $o(n)$ queries it is impossible to find a violation in a random f_i. □

Note the range $\mathcal{R}$ of the functions f_i is of size $O(\sqrt{n}\,)$ - much smaller than the 2^n different values a function on the hypercube may take. Our result then implies a query complexity lower bound of $\Omega(n \log n/\varepsilon)$ for pair testers of *boolean* monotonicity (see Section 8.6), by dint of the range reduction lemma of [DGL+99]. Although the lemma holds for every poset, we formulate it here for the particular case of the n-cube:

8.6.4. Lemma (Dodis et al.: Range reduction [DGL+99, Theorem 3]). *Let $c\colon \mathbb{N} \to \mathbb{N}^+$ and suppose for some distribution $\mathcal{D}_n$ on pairs $(x, y) \in V_n \times V_n$ with $x \leq y$, and for every function $f\colon V_n \to \{0,1\}$,*

$$\Pr_{(x,y)\sim\mathcal{D}_n}[f(x) > f(y)] \geq \frac{\varepsilon_M(f)}{c(n)}.$$

Then, for every $\mathcal{R}$ and every function $g\colon V_n \to \mathcal{R}$,

$$\Pr_{(x,y)\sim\mathcal{D}_n}[f(x) > f(y)] \geq \frac{\varepsilon_M(f)}{c(n)\log|\mathcal{R}|}.$$

8.6.5. Corollary. *Suppose there is a pair tester of boolean monotonicity over H_n whose query complexity, for each distance parameter $\varepsilon > 0$, is upper bounded by $q(n)/\varepsilon$. Then $q(n) = \Omega(n/\log n)$.*

This is tight up to the $\log n$ factor in view of the existing $O(n/\varepsilon)$ upper bound via pair testers.

Proof. Fix a distribution $\mathcal{D}_n$ and pick a sufficiently large range $\mathcal{R}$. For each function $f\colon V_n \to \{0,1\}$, let $p(f) \triangleq \Pr_{(x,y)\sim\mathcal{D}_n}[f(x) > f(y)]$; define in the same way $p(g)$ for each $g\colon V_n \to \mathcal{R}$. The probability that a violation is detected between the values of f on two elements of the same pair after $q(n)/\varepsilon$ samples is bounded by $p(f)q(n)/\varepsilon$. Putting $\varepsilon = \varepsilon_M(f)$, the existence of a tester implies $p(f) \geq (2\varepsilon_M(f))/(3q(n))$ for any f. It follows that the range reduction lemma applies for $c(n) = O(q(n))$. Choosing $|\mathcal{R}| > 2\sqrt{n}$, we obtain $p(g) = \Omega(\varepsilon_M(g)/(q(n)\log n))$ for every $g\colon V_n \to \mathcal{R}$.

The expected number of draws of pairs from $\mathcal{D}_n$ before a violation of g is caught is $1/p(g)$, which is the expectation of a geometric random variable with parameter $p(g)$. By Markov's inequality, $3/p(g)$ samples suffice with probability at least $2/3$. This quantity is bounded by $O(q(n)\log n/\varepsilon)$ whenever $\varepsilon_M(g) \geq \varepsilon$, so we obtain a pair tester of monotonicity of functions with range $\mathcal{R}$ with query complexity $O(q(n)\log n/\varepsilon)$. This contradicts Theorem 8.6.3 unless $q(n) = \Omega(n/\log n)$. □

8.6.3 Recent developments

After completion of this work, the lower bound has been greatly extended in a recent paper that exploits connections with some problems in communication complexity (c.f. Section 5.1).

8.6.6. Theorem (Blais, Brody & Matulef [BBM11]). *Any adaptive, two-sided error ε-tester of monotonicity over the hypercube must make $\Omega(n/\varepsilon)$ queries.*

(As before, the range of f can be taken to be $O(\sqrt{n}\,)$; see their paper for details.)

Proof. Let $f\colon H_n \to \mathbb{Z}$. We apply a reduction from the n-disjointness problem. Let $A, B \in \{0,1\}^n$ be Alice's and Bob's strings, respectively. As before, Alice builds the parity defined on A, but here it is best to write it as a ± 1 valued parity χ_A that maps z to $(-1)^{\sum_{i\in A} z_i}$. Likewise, Bob builds the character χ_B. Now they communicate to test whether the function $h\colon \{0,1\}^n \to \mathbb{Z}$ defined by

$$h(z) = 2\cdot|x| + \chi_A(z) + \chi_B(z)$$

is monotone.

Now we study when the function h is monotone. Suppose $i \notin z$ and let us determine when $h(z\cup\{i\}) - h(z)$ is negative. The term $2\cdot|x|$ increases by two when going from z to $z\cup\{i\}$. If $i \notin A$ then $\chi_A(z) = \chi_A(z\cup\{i\})$ and the second summand stays the same; otherwise it changes by ± 2. Similarly for $\chi_B(z)$.

Consequently, the only way for $h(z \cup \{i\})$ to be less than $h(z)$ is for i to belong to both A and B and to have $\chi_A(z) = \chi_B(z) = 1$.

Clearly this means that h is monotone when A and B are not disjoint. It remains to be seen that h is $\Omega(1)$-far from monotone when $A \cap B = \{i\}$. Consider the 2^{n-1} edges of H_n in the ith direction. Exactly $1/4$ of these edges $(x, x \cup \{i\})$ satisfy the condition $\chi_A(x) = \chi_B(x) = 1$, and the set of those which do is vertex-disjoint. For each of these pairs of vertices, either $h(x)$ or $h(x \cup \{i\})$ needs to be modified to make h monotone, because $h(x) > h(x \cup \{i\})$. Therefore, when $A \cap B \neq \emptyset$, the function h is $1/8$-far from monotone, as we wished to show. □

8.7 Summary

We studied the problem of monotonicity testing over the hypercube. As previously observed in several works, a positive answer to a natural question about routing properties of the hypercube network would imply the existence of efficient monotonicity testers. We constructed a family of instances of $\Omega(2^n)$ pairs in n-dimensional hypercubes such that no more than roughly a $\frac{1}{\sqrt{n}}$ fraction of the pairs can be simultaneously connected with edge-disjoint paths. This answers an open question of Lehman and Ron [LR01], and suggests that the aforementioned appealing combinatorial approach for deriving query-complexity upper bounds from routing properties cannot yield, by itself, query-complexity bounds better than $\approx n^{3/2}$. Additionally, our construction can also be used to obtain a strong counterexample to Szymanski's conjecture about routing on the hypercube.

We also proved a lower bound of $\Omega(n/\varepsilon)$ queries for one-sided non-adaptive testing of monotonicity over the n-dimensional hypercube, as well as additional bounds for specific classes of functions and testers.

We suggest two open problems related to this line of research. The first one is to find better upper bounds for the special case of testing monotonicity of dist-k monotone functions, for some $k \geq 3$. As we saw in Section 8.4.1, non-trivial sparsity upper bounds can be found even if we restrict ourselves to pairings in which all pairs are at distance 3. This seems to indicate, in our opinion, that a better understanding of the small-distance situations will yield new insights that may be applicable in the general case.

As to the second one, recall from Section 8.6.1 that for $k \leq 3$, dist-k monotonicity can be tested with $O(n^{3/2})$ queries; on the other hand, the construction in Section 8.4.1 shows that sparsity considerations alone will never yield upper bounds better than this. In view of these results, it is natural to ask whether these two measures need to coincide for larger k; that is, whether the complexity of edge-testers may be better than the values derived from sparsity upper bounds.

Chapter 9

Cycle detection with jumps

In this chapter we consider the following related questions:

- Suppose you are in front of a huge roulette wheel, so large that you are only able to see one slot at a time. The numbers written on each slot are all different, but other than that they are arbitrary. You can turn the roulette clockwise and perform a rotation by j slots, where $j \leq m$ and m is some known quantity. You cannot read any of the numbers in between while the rotation is underway. What is the minimum number of turns needed to determine the total number of slots in the wheel?

- How many elements of a sequence is it necessary to inspect to decide whether or not a cycle exists (containing distinct elements), and if so determine its length and location?

- What is the most efficient way to identify a number n by making queries of the form $\pi(x \bmod n)$, where π is an unknown permutation of the residues modulo n?

The content of this chapter is based on the paper

- S. Chakraborty, D. García–Soriano and A. Matsliah. Cycle detection, order finding, and discrete log with jumps. In *Proceedings of the second Symposium on Innovations in Theoretical Computer Science (ITCS)* (formerly known as *Innovations in Computer Science*), pages 284–297, 2011.

9.1 Introduction

Let S be a finite set. Given a function $f\colon S \to S$ and an element $a_0 \in S$, define the ith iteration f^i of f by $f^0(a_0) = a_0$ and $f^i(a_0) = f(f^{i-1}(a_0))$ for all $i \geq 1$. As S is finite, at some point there will be repeated elements in the sequence

$a \triangleq f^0(a_0)f^1(a_1)\ldots$ of values obtained this way. Let $s \geq 0$ and $r > 0$ be the smallest integers such that $f^s(a_0) = f^{s+r}(a_0)$; it is readily seen that both can be simultaneously minimized, and the sequence a will consist of an initial "tail" of length s, followed by unlimited repetitions of a cycle of length r. Determining s and r, given $a_0 \in S$ and a black-box oracle to f, is the *cycle-detection problem*. When f is bijective (i.e., f is a permutation of S), then the tail is empty ($s = 0$) and the *order-finding problem* is to find the smallest $r > 0$ such that $f^r(a_0) = a_0$, and the *discrete-log problem* is, given an additional element $b_0 \in S$, to find the smallest $k \geq 0$ such that $f^k(a_0) = b_0$.

Cycle detection, order finding and discrete log are well-studied problems in various settings and models. There are plenty of algorithms, lower bounds and more general time-space tradeoff results known for these problems. Many applications have been found, ranging from Pollard's ρ algorithm for integer factoring [Pol75], through detecting infinite loops in programs, to measuring the period length of pseudorandom number generators or performing shape analysis of data structures (some of the highlights can be found on their Wikipedia pages as of the time of writing [Wika, Wikb]).

In most of the relevant literature, time and space complexity are the main measures of efficiency for algorithms solving these problems. The classical "tortoise and hare" algorithm of Floyd [Flo67] is probably the best example of a cycle-detecting algorithm with optimal space complexity: it uses only two pointers to elements in S, which move through the sequence $a = f^0(a_0)f^1(a_0)\ldots$ at different speeds, and detects a cycle after $O(s+r)$ steps (and function evaluations). (Specifically, it finds the smallest $i > 0$ for which $a_i = a_{2i}$. Such i must satisfy $i \geq s$ and $r \mid i$; the precise values of r and s can subsequently be easily determined.)

In the present chapter the main measure of efficiency considered is the query complexity, i.e., the number of elements of the sequence a inspected. (This corresponds to the number of wheel turns in our example.) Clearly, with the standard oracle, which only allows to evaluate f on a certain input, one cannot do better than evaluating f at least $s + r$ times. Here we consider more powerful oracles, which allow longer "jumps" in the sequence a at unit cost.

There are various scenarios in which our objective to minimize the number of such queries may make sense. One example is when S is the set of possible states of a system and f corresponds to a program being running on it; that is, f maps a given state a to the state $f(a)$ reached on completion of the next execution step. In this setting, running the program for $i > 1$ steps and then reading the state $f^i(a)$ may be almost as fast as reading just the next state $f(a)$. We are aware of a few works that are directly related to the model we study here. First is the work of Cleve [Cle04], where a query-complexity lower bound is shown for order-finding. Second is the more recent work of Lachish and Newman [LN11], who study the related problem of periodicity testing. Finally there is the paper [CFMW10] of Chakraborty et al., who study the complexity of testing periodicity in *quantum* property testing.

Also somewhat related are the works in which S corresponds to a group, and the complexity of these problems is measured by the number of group operations required before obtaining the result. More on this in Section 9.5.2.

9.2 Preliminaries

Recall that the symbol log denotes logarithms to the base 2, and ln denotes the natural logarithm. For notational brevity, instead of writing $\max\{\log x, 1\}$, we redefine $\log x$ to be 1 when $x < 2$ in order for expressions such as $\log\log n$ to be defined for all n. Whenever we write such phrases as "a prime power ${p_1}^{\alpha_1}$" or "the prime factorization of t is $\prod {p_i}^{\alpha_i}$" we assume implicitly that each of these p_i is prime.

We write $\operatorname{lcm}(S)$ for the least common multiple of all elements of a set $S \subseteq \mathbb{N}$. If $a \geq 0, b > 0$, we also write $a \bmod b$ for the unique $0 \leq r < b$ such that $a \equiv r \pmod b$.

The set of primes is denoted $\mathcal{P}$ and we write $\mathcal{P}_n \triangleq \mathcal{P} \cap \{1, \ldots, n\}$ for $n \in \mathbb{N}$. The set of prime divisors of n is denoted $\mathcal{PD}(n)$. Finally, denote by $\nu_p(x)$ the largest power of p that divides x, and if $D \subseteq \mathcal{P}$, let $\nu_D(x) = \prod_{p \in D} \nu_p(x)$, which is the part of the prime factorization of x that uses primes from D.

We make use of the following results from number theory:

9.2.1. Theorem (Chinese remainder theorem). *Let $m_1, \ldots, m_k, a_1, \ldots, a_k$ be integers. The system of congruences $x \equiv a_1 \pmod{m_1}, \ldots, x \equiv a_k \pmod{m_k}$ has a unique solution modulo $\operatorname{lcm}(m_1, \ldots, m_k)$ iff for all i, j, the congruence $a_i \equiv a_j \pmod{\gcd(m_i, m_j)}$ holds.*

9.2.2. Theorem (Prime number theorem). *Let $\pi(n) = |\mathcal{P}_n|$ be the number of primes up to n. Then*

$$\pi(n) = \frac{n}{\ln n} \pm O\left(\frac{n}{(\ln n)^2}\right).$$

In fact the weaker statement that $|\mathcal{P}_n| = \Theta(n/\log n)$ is enough for us. This is known as Chebyshev's theorem and admits a much simpler proof by analyzing the prime factors of binomial coefficients (see Theorem 7 of the book by Hardy and Wright [HW08]).

The following result is also Theorem 317 of [HW08].

9.2.3. Theorem (Wigert's Divisor bound [Wig07]). *Let $\tau(n)$ be the number of positive divisors of $n \in \mathbb{N}$. Then*

$$\tau(n) \leq 2^{O(\log n \,/\, \log\log n)}.$$

It may be worth noting that this worst-case bound is tight in that there are n for which $\tau(n) = 2^{\Omega(\log n \,/\, \log\log n)}$, even though the average order of $\tau(n)$ is just $\log n$.

9.2.1 Sequence oracles: restricted vs. m-restricted

Here S is a finite set and f an arbitrary function mapping S to itself.

9.2.4. Definition. An *unrestricted* oracle $O_f^\infty \colon S \times \mathbb{N} \to S$ for f maps every query (a, i) to $f^i(a)$.

Let $m \in \mathbb{N}, m > 0$. An *m-restricted* oracle $O_f^m \colon S \times [m] \to S$ for f is defined similarly, except the restriction $0 \le i < m$ must hold.

When we want to impose the additional constraint that f be a permutation of S, we may write π instead of f.

9.2.2 The problems

- **Cycle detection:** Given $a_0 \in S$ and oracle access to f, find the smallest $s \ge 0$ and $r > 0$ such that $f^s(a) = f^{s+r}(a)$. Considering the infinite sequence $a = a_0 a_1 \dots$ given by $a_i = f^i(a)$, it is easily seen that $a_0, \dots, a_{r+s-1}$ are distinct and $a_i = a_{i+r}$ whenever $i \ge s$. In this case an equivalent definition avoiding an explicit mention of the function f is an oracle that allows probing a sequence $a \in S^\infty$ having the property that $a_i = a_j$ implies $a_{i+1} = a_{j+1}$. The integer r is called the *length* of the cycle, and s its *starting position*.

- **Order finding:** Given $a_0 \in S$ and oracle access to $\pi \in Sym(S)$, find the smallest $r > 0$ such that $\pi^r(a) = a$; this is the length of the cycle to which a belongs in the cycle decomposition of π. Similarly, one can view this as the problem of finding the period length r in a purely periodic sequence a, in which $a_0, \dots, a_{r-1}$ are distinct and $a_i = a_{i+r}$ for all $i \ge 0$ (i.e., $s = 0$).[1] The m-restricted oracle is viewed in this setting as allowing one to query position $p + i$ of a (where $0 \le i < m$), provided that $p = 0$ or is a previously queried position.

- **Discrete log:** Given $a_0, b_0 \in S$ and oracle access to π, find the smallest $k > 0$ such that $\pi^k(a_0) = b_0$. If no such k exists (i.e., a_0 and b_0 belong to different cycles), output ∞.

9.3 Order finding with unrestricted oracles

Consider the sequence $a \in [n]^\infty$ associated with some $f \colon [n] \to [n]$. Take a randomized q-query algorithm $\mathcal{A}$ for order finding; without loss of generality we

[1] One may also consider the problem of finding the period of a general sequence (not arising from a permutation), where the same value may appear several times within each period. In this case, upper and lower bounds of $\Theta(r)$ queries are straightforward (for any type of oracle). However, in the property-testing setting, where the task is to distinguish periodic sequences from those which are "far from periodic", polylogarithmic bounds were obtained in [LN11].

assume it always makes q queries (some of which may be redundant). At any stage $0 \leq i < q$, its behaviour is determined by its random seed t and the *history sequence*

$$\mathcal{H}_i = ((q_0, \alpha_0), \ldots, (q_{i-1}, \alpha_{i-1}))$$

of query/answer pairs received thus far (where $\alpha_i = a_{q_i}$). $\mathcal{H}_0$ starts out empty. On history $\mathcal{H}_i$, where $i = |\mathcal{H}| < q$, $\mathcal{A}$ queries position $q_i \triangleq \textsc{query}_{\mathcal{A},t}(\mathcal{H}_i)$, obtaining $\alpha_i \triangleq a_{q_i}$ as a response, and then the new pair (q_i, α_i) is appended to $\mathcal{H}_i$ to form $\mathcal{H}_{i+1}$. After the last query, $\mathcal{A}$ guesses at the period of a based on $\mathcal{H}_q$ and t. For *any* a, the value returned by $\mathcal{A}$ is correct with probability at least $2/3$ (over t).

Suppose now we are promised that the period r belongs to a known set S_n. Construct the set $Cand(\mathcal{H}_i)$ of *candidate periods* that are consistent with $\mathcal{H}_i$. It is easy to see that

$$Cand(\mathcal{H}_i) = \{r \in S_n \mid \forall j, k \in [i] : \alpha_j = \alpha_k \Leftrightarrow r \mid (q_j - q_k)\}.$$

This set depends only on the pairs of queries that got the same answer, and not on the answer labels themselves. What this means is that each of the possible responses to the $i + 1st$ query falls into at most $i + 1$ equivalence classes: the answer can either be one of $\alpha_0, \ldots, \alpha_{i-1}$, or be a new one, but aside from that the values of $\alpha_0, \ldots, \alpha_{i-1}$ are irrelevant. This motivates the introduction of a new kind of oracle.

9.3.1 Divisibility oracles

9.3.1. Definition. A *divisibility oracle* for an unknown integer $r > 0$ is an oracle $DivO(r)$ that keeps a record of the sequence of queries $q_0, \ldots, q_{i-1}$ previously made, and on the ith query q_i returns the smallest index $j \leq i$ such that $q_j \equiv q_k \pmod{r}$.

It must be emphasized that, even though the indices returned by $DivO(r)$ on a given set of queries depends on the order in which the queries are placed, the result of any set of equality comparisons between those indices remains the same, as they are simply determined by equality comparisons modulo r. The results to a series of i calls to $DivO(r)$ induce a partition of $[i]$, where the indices of queries that received the same answer are put into the same component of the partition.

Divisibility oracles can be much weaker than unrestricted oracles for sequences of a special form. For example, if a is determined from r by $a_i = i \bmod r$ and we have $r \leq n$ for some known n, then there is a deterministic algorithm to find r that makes the single query for position $n! - 1$ and returns $a_{n!-1} + 1$. This works because $n! \equiv 0 \pmod{r}$, so $a_{n!-1} + 1 = ((n! - 1) \bmod r) + 1 = (r - 1) + 1 = r$. It is not possible to do this with a divisibility oracle; this algorithm "cheats" in that it takes advantage of some properties of the sequence that are not shared for all r-periodic sequences. In fact, this kind of algorithm is bound to fail in general: if

we look for algorithms that are required to work for *all* sequences, then divisibility oracles turn out to be every bit as powerful as unrestricted oracles.

9.3.2. THEOREM. *Suppose n is known. Any order-finding algorithm for cyclic sequences $a \in [n]^\infty$ using an unrestricted oracle gives rise to an algorithm to determine $r \le n$ by calls to a divisibility oracle to r, and vice versa.*

For any r, the transformations preserve the worst-case query complexity (over all r-periodic sequences a) and the success probability.

Proof. One direction is trivial, since a query to a divisibility oracle can be simulated with one query to an unrestricted oracle, followed by an examination of the labels of all previous queries to determine if one of them matches the latest one. We argue the converse.

We model the labels examined by the algorithm as sequences of length bounded by n (as n queries always suffice). We say that two sequences $\alpha, \beta \in [n]^{\le n}$ are *equivalent* if they are the same up to a relabelling, i.e., if there is a permutation $\sigma\colon [n] \to [n]$ such that $\beta = \alpha^\sigma$, where $(\alpha^\sigma)_{i\in[n]} = (\sigma(\alpha_i))_{i\in[n]}$. Choose one representative for each equivalence class $[\alpha]$; we denote it by $[\alpha]$ as well. Then α and β are equivalent iff $[\alpha] = [\beta]$. For any permutation $\pi\colon [n] \to [n]$, we also consider the permuted representative function $[]^\pi$ mapping α to $[\alpha]^\pi$.

We start with an arbitrary order-finding algorithm $\mathcal{A}$ and define a new algorithm $\mathcal{B}$ that simulates $\mathcal{A}$, except that it normalizes the labels "on the fly" according to a random representative function $[]^\pi$, so that the sequence $\alpha_0, \dots, \alpha_{i-1}$ of responses kept in $\mathcal{H}_i$ is always "normalized" this way. The new algorithm $\mathcal{B}$ will still use unrestricted oracles, albeit in a limited way. Concretely, $\mathcal{B}_{t,\pi}$ takes the random seed t and an additional uniformly random permutation π, and on history $\mathcal{H}_i$ does the following:

1. Make the same query $q_i = \textsc{query}_{\mathcal{A},t}(\mathcal{H}_i)$ that $\mathcal{A}$ would make; let $\beta_i \triangleq a_{q_i}$ be its answer.

2. Find the unique α_i satisfying

$$[(\alpha_0, \dots, \alpha_{i-1}, \beta_i)]^\pi = (\alpha_0, \dots, \alpha_{i-1}, \alpha_i).$$

3. Append (q_i, α_i) to $\mathcal{H}_i$ to form $\mathcal{H}_{i+1}$ (that is, pretend that the answer received was α_i instead of β_i).

It is important to note that for any t, a, the sequence $(\alpha_0, \alpha_1, \dots)$ so constructed follows the same distribution over random π as as the sequence of labels received by $\mathcal{A}$ on input a^σ over a random permutation $\sigma = \sigma(t, \pi)$. Also recall that all a^σ have the same period. These observations imply that $\mathcal{B}_{t,\pi}$ has the same query

complexity, and also has success probability 2/3 for any a:

$$\begin{aligned}\Pr_{t,\pi}[\mathcal{B}_{t,\pi} \text{ succeeds on } a] &= \Pr_{t,\sigma}[\mathcal{A}_t \text{ succeeds on } a^\sigma] \\ &= \mathop{\mathbb{E}}_{\sigma} \Pr_{t}[\mathcal{A}_t \text{ succeeds on } a^\sigma] \\ &\geq \frac{2}{3}.\end{aligned}$$

By construction, the decisions of $\mathcal{B}_{t,\pi}$ depend only on t and $[a]^\sigma$, and therefore once the random coin tosses for t and π have been made it behaves the same for equivalent sequences in $[n]^\infty$, i.e., sequences having the same period, say r. This entails that the decisions of $\mathcal{B}$ can be written as a random variable of the responses of $DivO(r)$, as can be easily shown by induction on the number of queries. □

9.3.2 Lower bound

9.3.3. Theorem (Chakraborty et al. [CGM11a]). *Order finding requires* $\Omega(\log r \,/\, \log\log r)$ *queries with an unrestricted oracle* O_f^∞.

Proof. We prove that for every n, order finding with $DivO(r)$ under the promise that the period r belongs to a known set $S_n \subseteq [n/2, n]$ of polynomial density (i.e., $m \triangleq |S_n| = n^{\Omega(1)}$) requires $\Omega(\log m \,/\, \log\log m) = \Omega(\log r \,/\, \log\log r)$ queries. We apply Yao's principle and show that no deterministic decision tree of small depth can succeed with probability at least 2/3 over random $r \in S_n$.

Take a depth-q decision tree $\mathcal{T}$ with ℓ leaves, each of whose internal nodes is associated with a query to $DivO(r)$. To the ith leaf ($i \in [\ell]$) corresponds a unique normalized history sequence $\mathcal{H}_\mathcal{T}(i)$ leading to it (as defined above) and a candidate set $\mathcal{C}_\mathcal{T}(i) = \mathcal{C}(\mathcal{H}_\mathcal{T}(i))$. The normalized history sequence, and hence also $\mathcal{C}_\mathcal{T}(i)$, is determined by a partition of $[q]$. Therefore $\ell \leq B_q$, where Bell's numberB_q is the number of partitions of an n-set (see, e.g., [LW01]). It is not hard to see that $B_q \leq q!$.

On the other hand, $\{\mathcal{C}_\mathcal{T}(i)\}_{i\in[\ell]}$ is a family of disjoint subsets of $[n]$ (for a given period, precisely one of the leaves will be reached). Since the tree is deterministic, there is only one $r = r_i$ on each leaf for which the correct period r_i is returned. As r is chosen uniformly at random from a set of cardinality m, the probability of picking some $r \in \bigcup_{i\in[\ell]}\{r_i\}$ is exactly ℓ/m. Hence for the success probability to be no smaller than 2/3, we need to have $\ell \geq (2/3)m$. Then $q! \geq B_q \geq \ell \geq (2/3)m$ and $q \geq \Omega(\log m \,/\, \log\log m)$, concluding the proof. □

9.3.4. Remark. This is in essence an information-theoretical lower bound, and also holds if r is drawn from any set of size n^ε. In particular the lower bound still applies to the special case where r is promised to be a prime power. In this

case the algorithms in this chapter provide a matching $O(\log r / \log\log r)$ upper bound.

9.3.3 Upper bound

The main result of this section is stated next.

9.3.5. Theorem (Chakraborty et al. [CGM11a]). *Assume we are given an upper bound n on the cycle length r. Then there is a randomized algorithm that finds r by making $O(\log n / \log\log\log n)$ queries to a divisibility oracle for r.*

Therefore there is a randomized order-finding algorithm with an unrestricted oracle making the same queries. We will tackle the question of how to obtain a good upper bound n in Section 9.5.3. Our techniques are inspired by junta testers: we think of r as a junta and we attempt to find the set $\mathcal{PD}(r)$ of "relevant variables". (Here the junta core would be the product of the elements of $\mathcal{PD}(n)$ raised to the appropriate powers.)

For starters observe that it is easy to find out whether a given prime power p^α divides r:

9.3.6. Lemma. *Let t be the largest divisor of $\operatorname{lcm}([n])$ that is not a multiple of p^α. If $r \le n$, then $p^\alpha \mid r \Leftrightarrow r \nmid t$.*

Proof. The prime factorization of t is the product of all prime powers that are at most n, except for p, which appears with exponent $\min(\alpha - 1, \lfloor \log_p(n) \rfloor)$ instead. So any prime power appearing in the factorization of r appears in t as well, except possibly for p^α. Therefore $r \mid t$ implies that p appears with exponent at most $\alpha - 1$ in r, so $p^\alpha \nmid r$; conversely, $p^\alpha \mid r$ implies $r \nmid t$. □

Since $r \mid t$ is the same thing as $\textsc{query}(0) = \textsc{query}(t)$, we can perform this check easily. This is done by Procedure IsDivisor, which admits a straightforward generalization HasFactor to sets D of prime powers.

If we knew a prime factor p of r, we could use IsDivisor and binary search to compute the exponent α of p in the prime factorization of r by making at most $1 + 2\log\alpha$ queries. Procedure FindExponents takes as input $D = \mathcal{PD}(r)$, computes the exponents for all prime factors of r and returns r itself. Therefore, if we somehow managed to know the set $\mathcal{PD}(r)$, we could find the precise value of r with no more than $\sum_{i=1}^{k}(1+2\log(\alpha_i))$ additional queries, where $r = \Pi_i^k p_i^{\alpha_i}$ has been written according to its prime factorization. In analyzing these query complexities we shall make repeated use of the following bound on the size of $\mathcal{PD}(n)$:

9.3.7. Lemma. $|\mathcal{PD}(n)| \le 4\log n / \log\log n$.

Algorithm 6 – return **true** if r is divisible by some element of D

Require: $D = \{p_1^{\alpha_1}, \ldots, p_k^{\alpha_k}\}$ $(\alpha_i \geq 1)$ is a set of powers of distinct primes;
$r \leq n$

1: **procedure** HasFactor(D, n)
2: find the prime factorization of $\operatorname{lcm}([n]) = p_1^{\beta_1} \ldots p_m^{\beta_m}$
(where the first k factors correspond to the primes in D)
3: $t \leftarrow \prod_{i=1}^{k} p_i^{\alpha_i - 1} \prod_{i=k+1}^{m} p_i^{\beta_i}$
4: **return true** iff query$(0) \neq$ query(t)

5: **procedure** IsDivisor(p^α, n) ▷ is r divisible by prime power p^α?
6: **return** HasFactor$(\{p^\alpha\}, n)$

Proof. Up to constants, the result follows from Theorem 9.2.3 because $\tau(n) \geq 2^{|\mathcal{PD}(n)|}$, but it can also be shown directly. Let b be the least integer satisfying $b^b \geq n$; then $b = (1 - o(1)) \log n \,/\, \log\log n$ and in fact one can check that $b \leq 2 \log n \,/\, \log\log n$ for all n. There are at most b primes in $[2, b]$ and since $b^b \geq n$, there are at most b prime factors of n in $[b, n]$. Hence there are at most $2b$ primes that divide n. □

Observe that we also have $|\mathcal{PD}(r)| \leq 4 \log n \,/\, \log\log n$ if $r \leq n$ since the function $x \to \log x \,/\, \log\log x$ is increasing, irrespective of whether $|\mathcal{PD}(r)| < |\mathcal{PD}(n)|$.

9.3.8. Lemma. *Let $\Pi_{i=1}^{k} p_i^{\alpha_i}$ be the prime factorization of r. Then*

$$\sum_{i=1}^{k} (1 + \log(\alpha_i)) = O(\log r \,/\, \log\log r).$$

Proof. The number of divisors of r is $\tau(r) = \prod_i (\alpha_i + 1)$. By the divisor bound (Theorem 9.2.3), $\tau(r) = 2^{O(\log r \,/\, \log\log r)}$, whence

$$\sum_{i=1}^{k} \log \alpha_i = \log \prod_{i=1}^{k} \alpha_i < \log \tau(n) = O(\log r \,/\, \log\log r).$$

Since $k = |\mathcal{PD}(r)| = O(\log r \,/\, \log\log r)$, we get

$$\sum_{i=1}^{k} (1 + \log \alpha_i) = k + \sum_{i=1}^{k} \log \alpha_i = O(\log r \,/\, \log\log r).$$

□

9.3.9. Corollary. *The number of queries Procedure* FindExponents *makes when its requirements are met is $O(\log r \,/\, \log\log r)$.*

Algorithm 7 – return r, given the set D of its prime divisors

Require: $D = \mathcal{PD}(r)$

```
1: procedure FindExponents(D, n)
2:     r ← 1
3:     for all p ∈ D do                                  ▷ compute exponent
4:         α, β ← 1
5:         while IsDivisor(p^β, n) do                    ▷ find a strict upper bound β
6:             β = 2β
7:         while β − α ≠ 1 do                            ▷ binary search
8:             γ ← ⌊(α+β)/2⌋
9:             if IsDivisor(p^γ, n) then α = γ else β = γ
10:        r ← r · p^α
11:    return r
```

Thus the main task is finding the set $\mathcal{PD}(r)$ using $O(\log n \,/\, \log\log\log n)$ queries. We divide this task into three sub-tasks. First we present an algorithm that, given a subset D that is known to contain *exactly* one prime factor of r, finds it. Building on this we give a method to isolate all prime factors of r into disjoint subsets of $[n]$. Then, in order to reduce the query complexity, we partition $[n]$ into intervals of increasing length and find those prime factors of r inside each. These intervals will be carefully chosen so as to guarantee that the overall query complexity of finding all the prime factors of r remains small.

Locating one prime factor

If we are given a subset D containing precisely one prime divisor of r, it is not difficult to see that binary search and Procedure HasFactor can be used to find p with $O(\log |D|)$ queries. Unfortunately, this is too expensive for our purposes; we show how to do better. The key idea is the following. Suppose that, over the previous queries, we have obtained t distinct responses from the divisibility oracle, corresponding to t different remainders modulo r. Further, assume for simplicity that r is a prime contained in some set D. Then we can split D into $t+1$ parts and query some number x that is guaranteed to leave a different remainder modulo t depending on which of these $t+1$ parts r belongs to (this is possible via the Chinese remainder theorem). Hence we can manage to divide the search range by a factor of $t+1$, and possibly learn a new remainder modulo r. In contrast, a binary search would always divide the size of the search range by two.

9.3.10. Lemma. *Procedure* FindUniquePrimeDivisor(D, n) *finds the unique* $p \in \mathcal{PD}(r) \cap D$ *and makes* $O(\log |D| \,/\, \log\log |D|)$ *queries.*

Proof. First we show that the query complexity is $O(\log |D| \,/\, \log\log |D|)$. The **for** loop in line 4 makes $O(\log |D| \,/\, \log\log |D|)$ queries. Every iteration of the

Algorithm 8 – find the unique prime factor of r in D

Require: $r \leq n$ and $|\mathcal{PD}(r) \cap D| = 1$

1: **procedure** FINDUNIQUEPRIMEDIVISOR(D, n)
2: $k \leftarrow 2\lfloor \log|D| \,/\, \log\log|D| \rfloor$
3: $T \leftarrow \mathcal{P}_n \setminus D$; $a \leftarrow \nu_T(\mathrm{lcm}([n]))$ ▷ a = prime powers outside of D
4: **for** $i = 0$ **to** $k-1$ **do** ▷ loop through small candidates for p
5: $v_i \leftarrow \textsc{query}(i \cdot a)$
6: **if** $i > 0$ **and** $v_i = v_0$ **then**
7: **return** the only prime divisor of i
8: **while** $|D| \geq 2$ **do**
9: construct an equipartition of D into k parts $D_1, \ldots, D_k$
10: $D \leftarrow$ WHICHISRELEVANT($\{D_1, \ldots, D_k\}, n, a, \{v_0, \ldots, v_{k-1}\}$)
11: **return** the unique $p \in D$

12: **procedure** WHICHISRELEVANT($\{D_1, \ldots, D_k\}, n, a, \{v_0, \ldots, v_{k-1}\}$)
13: **for** $i = 1$ **to** k **do** $a_i \leftarrow a \cdot \nu_{D_i}(\mathrm{lcm}([n]))$
14: find x such that $x \equiv i \cdot a \pmod{a_i}$ for all $i \in [1, k]$ using the CRT
15: $y \leftarrow \textsc{query}(x)$
16: find $1 \leq i \leq k$ such that $v_i = y$, or **return fail** if none exists
17: **return** D_i

while loop in line 8 divides the size of D by a factor of k (up to a floor function). The maximum number of iterations increases with $|D|$ and since $|D| \leq k^k$ at the outset, the body of the loop is executed k times at most. As only one query is made inside each iteration, the total query complexity of the **while** loop is $O(\log|D| \,/\, \log\log|D|)$.

Now we show the correctness of the algorithm. Let $\mathcal{PD}(r) \cap D = \{p\}$ and the exponent of p in r be t. The condition $v_i = v_0$ in line 6 is satisfied when $r \mid i \cdot a$, which by our choice of a is equivalent to $p^t \mid i$. So the first time line 7 will be reached is when i attains the value p^t, provided $p^t < k$. In this case p is the only prime divisor of i and the return value will be p. So all we need to show is that if $p^t \geq k$, the **while** loop finds the correct D_i to which p belongs.

In line 14 we need to find x such that $x \equiv i \cdot a \pmod{a_i}$ for all i. The existence of such x is guaranteed by the Chinese Remainder Theorem because for all $i \neq j$, we have $\gcd(a_i, a_j) = a$ and $i \cdot a \equiv j \cdot a \equiv 0 \pmod{a}$ (some of the sets D_i may be empty, in which case $a_i = a$). If $p \in D_i$, then $p^t \mid \nu_{D_i}(\mathrm{lcm}([n]))$ and so $r \mid a_i$. This gives $x \equiv i \cdot a \pmod{r}$, implying $\textsc{query}(x) = \textsc{query}(i \cdot a) = v_i$. To complete the proof we have to show that no $j \neq i$ can satisfy $\textsc{query}(x) = \textsc{query}(j \cdot a)$, which is the same thing as $x \equiv j \cdot a \pmod{r}$. This is because $x \equiv j \cdot a \pmod{r}$ and $x \equiv i \cdot a \pmod{r}$ together imply $a(i-j) \equiv 0 \pmod{r}$, so $a(i-j) \equiv 0 \pmod{p^t}$ (as $p^t \mid r$); and since $\gcd(a, p^t) = 1$ we can divide by a to get $i \equiv j \pmod{p^t}$ and $k > |i-j| \geq p^t$, which is a contradiction. □

Isolating the prime factors of r

Recall from Section 5.3.1 that it is possible to isolate the relevant variables of a strong k-junta with $O(k \log k)$ queries. Here we take $k = \Theta(\log n / \log\log n)$ to be the largest possible size of $\mathcal{PD}(r)$. As we hope to obtain an upper bound that is almost linear in k, we need some improvements. One of the observations of Section 5.3 is that, if we could distinguish between a set D having two relevant variables or only one, then we could actually solve this problem with $O(k)$ queries. So going back to our idea of thinking of prime powers as "relevant variables" of r, we would like to have a procedure HasTwoPrimeDivisors to determine whether a given set D contains at least two prime factors of r. One solution would be to call FindUniquePrimeDivisor and check the result, but this makes more queries than we can afford for large D. In fact we do not know of any query-efficient deterministic solution, but a randomized one can be designed along the lines of Procedure FindUniquePrimeDivisor. It performs better than FindUniquePrimeDivisor does when $|D| = \Omega(1/\delta)$.

Algorithm 9 – determine if D contains ≥ 2 prime divisors of r with confidence $1 - \delta$

Require: $r \leq n$

1: **procedure** HasTwoPrimeDivisors(D, n, δ)
2: $k \leftarrow 2\lceil \log(\delta^{-1}) / \log\log(\delta^{-1}) \rceil$
3: $F \leftarrow \{p \in \mathcal{P}_{k-1} \mid \text{IsDivisor}(p, n)\} = \mathcal{PD}(r) \cap [k]$
4: **if** $F \neq \emptyset$ **then**
5: **return** $|F| \geq 2$ **or** ($|F| = 1$ **and** HasFactor($D \setminus F, n$))
6: compute $a, \{v_0, \ldots, v_{k-1}\}$ as in FindUniquePrimeDivisor
7: **for** $i = 1$ **to** k **do**
8: split D into k disjoint sets $D_1, \ldots, D_k$ by placing each $p \in D$ in a randomly selected D_j
9: **if** WhichIsRelevant($\{D_1, \ldots, D_k\}, n, a, \{v_0, \ldots, v_{k-1}\}$) **fails then**
10: **return true**
11: **return false**

9.3.11. Lemma. *Let* $d \triangleq |D \cap \mathcal{PD}(r)|$.

If $d \leq 1$, *Procedure* HasTwoPrimeDivisors(D, n, δ) *always returns* ***false****. If* $d \geq 2$ *it returns* ***true*** *except with error probability* $\delta^{d-1} \leq \delta$. *It makes* $O(\log \delta^{-1} / \log\log \delta^{-1})$ *queries.*

Proof. Clearly, the query complexity is as stated, and if $d \leq 1$ then the algorithm always returns **false**. Also the correct decision is always made in line 5 if there is some element of $\mathcal{PD}(r)$ smaller than k.

So assume that $d \geq 2$ and $\mathcal{PD}(r) \cap [k] = \emptyset$. With probability $1 - (1/k)^{d-1}$, at least two $p, q \in D \cap \mathcal{PD}(r)$ land into different sets D_i, D_j in line 8, where $1 \leq i, j \leq k$. We claim that conditioned on this, WhichIsRelevant fails. This happens when $x \not\equiv m \cdot a \pmod{r}$ for any $1 \leq m \leq k$.

Indeed, consider the case $x \equiv i \cdot a \pmod{p}$, $x \equiv j \cdot a \pmod{q}$ and suppose for a contradiction that $x \equiv m \cdot a \pmod{r}$. Noting that $x \equiv 0 \pmod{a}$ and $pq \mid r$, this implies $x/a \equiv i \pmod{p}$, $x/a \equiv j \pmod{q}$ and $x/a \equiv m \pmod{pq}$. Therefore $i \equiv m \pmod{p}$, and from $1 \leq i, m \leq k \leq p$ we deduce $i = m$. Likewise, $j = m$, implying $i = j$.

We have seen that each iteration of the **for** loop in line 7 returns **true** except with probability $(1/k)^{d-1}$. Since k independent iterations are run, the error probability is $1/k^{(d-1)k} \leq \delta^{d-1}$ because $k^k \geq 1/\delta$. □

Algorithm 10 – isolate prime factors with probability $1 - \delta$

Require: $r \leq n$ and $D \subseteq \mathcal{P}_n$

```
 1: procedure IsolateFactors(D, n, δ)
 2:     if HasTwoPrimeDivisors(D, n, δ/3) then
 3:         repeat
 4:             split D into two sets D_1 and D_2 at random
 5:         until HasFactor(D_1, n) and HasFactor(D_2, n)
 6:         return IsolateFactors(D_1, n, δ/3) ∪ IsolateFactors(D_2, n, δ/3)
 7:     else
 8:         if HasFactor(D, n) then
 9:             return {D}
10:         else
11:             return ∅
```

9.3.12. Lemma. *Let $d = |\mathcal{PD}(r) \cap D|$. When* IsolateFactors *succeeds, it returns d disjoint sets containing one element of $\mathcal{PD}(r) \cap D$ each. The error probability of* IsolateFactors(D, n, δ) *is at most δ, and its expected query complexity is $O(d \log(d/\delta) / \log\log(d/\delta))$.*

Proof. Lines 10 and 11 are there only to cover the trivial case $d = 0$, so let $d \geq 1$. The probability that some error is ever made by some of the probabilistic procedures employed is, using the union bound, at most the sum of the error probabilities of the call to HasTwoPrimeDivisors in line 2 (which is at most $\delta/3$) and both recursive calls to FindPrimeDivisors in line 6. By induction on the call depth, each of the latter have error probability $\delta/3$, summing up to $3\delta/3 = \delta$.

Provided the algorithm terminates and all these probabilistic calls were correct, the set returned has size d and isolates $\mathcal{PD}(r) \cap D$. The call tree is then a complete binary tree with d leaves. This means that the number of times lines

4 and 8 are executed is precisely $d-1$ and d respectively, and that the smallest confidence parameter used is at least $\delta/d^{\log 3} = (\delta/d)^{O(1)}$ as the depth is bounded by $\lceil \log d \rceil$. Each execution of line 2 contributes $O(\log(d/\delta)/\log\log(d/\delta))$ to the query complexity, and each execution of line 8 takes $O(1)$ queries. Hence the number of queries due to lines 2 and 8, which are run $O(d)$ times, is $O(d\log(d/\delta)\,/\log\log(d/\delta))$.

It remains to bound the query complexity of line 5, which is bounded by a constant times the expected number of iterations of the **repeat** loop. Because of the one-sidedness of HasTwoPrimeDivisors, the loop is only run when $|\mathcal{PD}(r) \cap D| \geq 2$. Then the expected number of tries before a successful split is at most two, and by linearity of expectation the total number of iterations is bounded by $2(d-1)$. □

Partitioning

Remember that r can have as many as $\Omega(\log n\,/\log\log n)$ prime divisors. Therefore an invocation of IsolateFactors with $D = [n]$ followed by calls to FindUniquePrimeDivisor might require as many as $\Omega((\log n\,/\log\log n)^2)$ queries on average. To overcome this difficulty, we partition $[n]$ into consecutive intervals of increasing length; the intuition being that if $a < b < c < d$, the maximum number of prime divisors of r the interval $[c,d]$ can contain is smaller than the number of divisors of r that the interval $[a,b]$ can contain. The key is to choose a judicious division of $[n]$ into a sequence of intervals.

Algorithm 11 – finds the period r

Require: $r \leq n$

1: **procedure** FindPeriod(n)
2: $a = \min\{i \mid 2^{i\log i} \geq \log n\}$
3: $b = \min\{i \mid 2^{i\log i} > n\}$
4: $A = \mathcal{P}_{2^{a\log a}-1}$
5: **for** $i = a$ **to** $b-1$ **do**
6: $I = [2^{i\log i}, 2^{(i+1)\log(i+1)} - 1] \cap [n]$
7: $A \leftarrow A \cup$ IsolateFactors$(I, n, 1/\log n)$
8: $\mathcal{PD} = \{$FindUniquePrimeDivisor$(D,n) \mid D \in A\}$
9: **return** FindExponents$(\mathcal{PD}, n)$

Theorem 9.3.5 is a consequence of the next result.

9.3.13. Lemma. *Procedure* FindPeriod(n) *makes an expected number of*

$$O(\log n\,/\log\log\log n)$$

queries and returns r with high probability.

Recall that standard techniques can convert average-case complexity into worst-case complexity at the expense of a constant factor in the error probability owing to Markov's inequality.

Proof. First we show correctness. Note that $a < b = \Theta(\log n \,/\, \log\log n)$. Define the sequence

$$l_i \triangleq 2^{i \log i} = i^i.$$

It is easy to see that $l_a = \Theta(\log n)$ and $l_{b-1} = n^{\Theta(1)}$. The interval $[n]$ is partitioned into $O(\log n \,/\, \log\log n)$ parts: namely $\pi(l_a - 1)$ parts containing small primes, then the intervals $[l_i, l_{i+1} - 1]$ for $a \le i < b-1$ and finally the interval $[l_b, n]$. Each call to IsolateFactors is made with error parameter $\delta = 1/\log n$, and since there are $o(\log n)$ such calls, with probability $1 - o(1)$ all of them work correctly. Then lines 4 to 6 find a collection A of sets containing at most one prime factor of r, and lines 8 and 9 locate these factors and compute the actual period.

Now we prove the bound on the query complexity. We have $l_a = \Theta(\log n)$, hence by the prime number theorem there are $\Theta(\log n \,/\, \log\log n)$ primes in the interval $[0, l_a - 1]$. Thus we only need to consider the other intervals. Consider an interval $D \subseteq [n]$ of size $|D| \ge \log n$ and set $d = |\mathcal{PD}(r) \cap D|$. For $\delta = 1/\log n$, a call to IsolateFactors(D, n, δ) followed by the at most d calls to FindUniquePrimeDivisor(D_i, n) for each of the subintervals D_i returned takes

$$O\left(\frac{d \log |D|}{\log\log n}\right)$$

queries on expectation. Indeed, the expected cost of IsolateFactors(D, n, δ) is

$$O\left(\frac{d \log(d/\delta)}{\log\log(d/\delta)}\right) \le O\left(\frac{d \log(d/\delta)}{\log\log n}\right) = O\left(\frac{d \log d}{\log\log n} + \frac{d \log\log n}{\log\log n}\right), \tag{9.1}$$

and the total cost of the d calls to FindUniquePrimeDivisor(D_i, n) is

$$O\left(\frac{d \log |D|}{\log\log n}\right). \tag{9.2}$$

The expression (9.2) dominates the right-hand side of (9.1) term by term. Hence the cost of the calls to FindUniquePrimeDivisor dominate (on average).

Let k_i be the size of the interval $I_i = [l_i, l_{i+1} - 1]$ and $n_i = |\mathcal{PD}(r) \cap [l_i, l_{i+1} - 1]|$; then $k_i \triangleq l_{i+1} - l_i = \Theta(l_{i+1})$. Let $n_i \triangleq |\mathcal{PD}(r) \cap [l_i, l_{i+1} - 1]|$. Since $|I_i| \ge \log n$ and linearity of expectation, it follows that the expected number of queries is bounded by $O(\sum_{i=1}^{b-1} n_i \log k_i \,/\, \log\log n)$. Note that

$$n \ge r \ge \prod_{i=1}^{b-1} l_i^{n_i} = \prod_{i=1}^{b-1} 2^{n_i \, i \log i},$$

so by taking logarithms on both sides we obtain

$$\sum_{i=1}^{b-1} n_i \, i \log i \leq \log n.$$

These inequalities, together with $\log k_i = O(i \log i)$, can be used to bound the total expected number of queries made in the second **for** loop by

$$\begin{aligned}\sum_{i=a}^{b-2} n_i \frac{\log k_i}{\log \log n} &\leq \frac{O(1)}{\log \log \log n} \sum_{i=a}^{b-2} n_i \, i \log i \\ &\leq O\left(\frac{\log n}{\log \log \log n}\right),\end{aligned}$$

as we set out to prove. □

9.4 Order finding with restricted oracles

9.4.1 Lower bound

9.4.1. Theorem (Chakraborty et al. [CGM11a]). *Order finding with an m-restricted oracle requires*

$$\Omega\left(\frac{\log r}{\log \log r} + \sqrt{\frac{r}{\log m \log r}} + \frac{r}{m}\right)$$

queries.

Recall that taking maximums is the same as taking sums from an asymptotic point of view. The term $\log r \,/\, \log \log r$ follows from the bound for unrestricted oracles, which are a special case. So we focus our efforts on analyzing how much a restriction a small jump size m is by studying the largest index that needs to be queried. The term r/m is clear from the mere fact that the algorithm needs to reach position $r - 1$ to detect a cycle of length r. Hence, it suffices to prove a lower bound of

$$q(n, m) \triangleq \sqrt{\frac{n}{\log m \log n}}$$

queries for order finding under the promise that $r = \Theta(n)$. Cleve [Cle04] proved that the query complexity of order finding with an m-restricted oracle is $\Omega\left(\frac{|S|^{1/3}}{\sqrt{\log m}}\right)$, and if $m \geq 2|S|$ then it is $O(|S|^{1/2})$. Since $n = |S|$ is clearly an upper bound on r, we improve Cleve's lower bound by a factor of roughly $\frac{n^{1/6}}{\sqrt{\log n}}$. In particular, for any $m = \text{poly}(n)$ Cleve's bound is $\widetilde{\Omega}(n^{1/3})$ and ours is $\widetilde{\Omega}(n^{1/2})$, which we will see is nearly optimal.

For $n \in \mathbb{N}$ and $1 < r < n/2$, we denote by $\mathcal{G}_n^r$ the set of all permutations $\pi\colon [n] \to [n]$ consisting of two disjoint cycles, one of length r and the other of length $n - r > n/2 > r$. Given $R \subseteq [n]$, define $\mathcal{G}_n^R \triangleq \bigcup_{r \in R \cap (1,n/2)} \mathcal{G}_n^r$.

Recall that for every permutation π, the m-restricted oracle O_π^m maps $[n] \times [m]$ to $[n]$ according to π; namely, $O_\pi^m(i,j) = \pi^j(i)$. Given access to O_π^m corresponding to some $\pi \in \mathcal{G}_n^{[n]}$, an order-finding algorithm should be able to compute r such that $\pi \in \mathcal{G}_n^r$. We show the existence of a pair of disjoint sets $R_1, R_2 \subseteq [n/2]$, and a distribution $\mathcal{D}$ on permutations from $\mathcal{G}_n^{R_1} \cup \mathcal{G}_n^{R_2}$, such that no deterministic algorithm can tell if a random $\pi \sim \mathcal{D}$ belongs to $\mathcal{G}_n^{R_1}$ or $\mathcal{G}_n^{R_2}$ unless it makes $\Omega(q(n,m))$ queries to O_π^m.

Formal statement

Let $Q = \{(i_1, j_1), \ldots, (i_q, j_q)\} \subseteq [n] \times [m]$ be a set of q queries (each being a pair (i,j) fed as input to oracle O_π^m). Let $R_1, R_2 \subseteq (\frac{1}{5}n, \frac{1}{2}n)$ be a pair of disjoint non-empty sets of integers. For $a \in \{1,2\}$, let D_a denote the uniform distribution over all permutations $\pi \in \mathcal{G}_n^{R_a}$.

For the lower bound, it suffices to prove the following (see Lemma 1.5.6):

9.4.2. Proposition. *There are R_1, R_2, with corresponding distributions D_1 and D_2, satisfying the following property: for every set $Q = \{(i_1, j_1), \ldots, (i_q, j_q)\}$ of $q = o(q(n,m))$ (distinct) queries and every $\alpha \in [n]^q$, we have*

$$\Pr_{\pi \sim D_1}[O_\pi^m(Q) = \alpha] = (1 \pm o(1)) \cdot \Pr_{\pi \sim D_2}[O_\pi^m(Q) = \alpha],$$

where $O_\pi^m(Q)$ denotes the string

$$O_\pi^m(i_1, j_1) \cdots O_\pi^m(i_q, j_q) \in [n]^q.$$

Outline of the proof

First, we can assume without loss of generality that any order-finding algorithm finds a collision in π; namely, it makes a pair of queries (i,j) and (i',j') such that $O_\pi^m(i,j) = O_\pi^m(i',j')$. Indeed, once r has been determined, one additional query suffices to find a collision.

Second, we also observe that the actual values returned by oracle O_π^m are irrelevant. Namely, as long as the algorithm finds no collisions, the values obtained from earlier queries are just random elements from $[n]$ (distributed uniformly without repetitions). Therefore, we may assume that the choice of queries is non-adaptive while no collision has been found.

Having made these observations, all we need to show is that for any fixed set Q of $o(q(n,m))$ queries and $a \in \{1,2\}$, the probability that Q contains a collision with regard to $\pi \sim D_a$ is $o(1)$.

Core lemmas

Fix Q as above, and let $Q_1, \dots, Q_\ell$ be the partition of Q where $(i,j), (i',j') \in Q$ belong to the same Q_h if and only if $i = i'$. Clearly $\ell \le q$. Given π, a subset $Q' \subseteq Q$ is called *π-collision-free* if $O^m_\pi(i,j) \neq O^m_\pi(i',j')$ for all $(i,j) \neq (i',j') \in Q'$. The query set Q' is *r-collision-free* if it is π-collision-free for all $\pi \in \mathcal{G}^r_n$. We say that $Q = Q_1 \cup \dots \cup Q_\ell$ is *component-wise r-collision-free* if Q_h is r-collision-free for every $h \in [\ell]$.

In the following lemmas we let Q be an arbitrary set of size $q = o(q(n,m))$, and by $Q_1, \dots, Q_\ell$ we denote the foregoing partition of Q.

9.4.3. Lemma. *For infinitely many $n \in \mathbb{N}$ there exists a pair of non-empty disjoint sets $R_1, R_2 \subseteq (\frac{1}{5}n, \frac{1}{2}n)$ such that for $a \in \{1,2\}$,*

$$\Pr_{r \in R_a}\Big[Q \text{ is component-wise } r\text{-collision-free}\Big] \ge 1 - o(1).$$

Given π and $h \neq h' \in [\ell]$, we say that Q_h and $Q_{h'}$ are *π-disjoint* if for all $(i,j) \in Q_h$ and $(i',j') \in Q_{h'}$, $O^m_\pi(i,j) \neq O^m_\pi(i',j')$. We say that Q is π-disjoint if for all $h \neq h' \in [\ell]$, Q_h and $Q_{h'}$ are π-disjoint.

9.4.4. Lemma. *For every (sufficiently large) n and r, $\frac{1}{5}n < r < \frac{1}{2}n$,*

$$\Pr_{\pi \in \mathcal{G}^r_n}\Big[Q \text{ is } \pi\text{-disjoint}\Big] \ge 1 - o(1).$$

Observe that if for some $\pi \in \mathcal{G}^r_n$, Q is both π-disjoint and component-wise r-collision-free, then it is π-collision-free (with regard to that particular π). Hence, by these two lemmas we get the following.

9.4.5. Corollary. *For infinitely many $n \in \mathbb{N}$ there exists a pair of non-empty disjoint sets $R_1, R_2 \subseteq (\frac{1}{5}n, \frac{1}{2}n)$ such that for $a \in \{1,2\}$,*

$$\Pr_{\pi \in \mathcal{G}^{R_a}_n}\Big[Q \text{ is } \pi\text{-collision-free}\Big] \ge 1 - o(1).$$

Proposition 9.4.2 follows from Corollary 9.4.5, as sketched in the proof outline.

Proof of Lemma 9.4.3

We start with an auxiliary lemma, the statement of which should not be feared despite its intimidating look.

9.4.6. Lemma. *There exist absolute constants $\delta > 0$ and $n_0 \in \mathbb{N}$ such that for any $\hat{n} \ge n_0$ there is $n = (1 \pm \frac{1}{12})\hat{n}$ and α, β, γ, where $\frac{1}{5} < \alpha < \beta < \gamma < \frac{1}{2}$, for which the following holds. There exist $2k \ge \delta n / \log^2 n$ pairs*

$$(p_1, t_1), \dots, (p_k, t_k), (p'_1, t'_1), \dots, (p'_k, t'_k)$$

such that for all $i \in [k]$ the following holds:

- *p_i, t_i, p'_i and t'_i are all primes;*
- *$p_i \neq p_j$, $t_i \neq t_j$, $p'_i \neq p'_j$ and $t'_i \neq t'_j$ for all $j \in [k] \setminus \{i\}$;*
- *$p_i + t_i = n$ and $p'_i + t'_i = n$.*
- *$\alpha n < p_i < \beta n$ and $\beta n < p'_i < \gamma n$ (consequently, $p_i < t_i$ and $p'_i < t'_i$);*

Proof. By the Prime Number Theorem, there exists $\varepsilon > 0$ and $n_0 \in \mathbb{N}$ such that for any $\hat{n} \geq n_0$, the number of primes $\hat{p} \in (\frac{1}{4}\hat{n}, \frac{1}{3}\hat{n})$, as well as the number of primes $\hat{t} \in (\frac{2}{3}\hat{n}, \frac{3}{4}\hat{n})$, is at least $\hat{k} \triangleq \varepsilon\hat{n} \,/\, \log \hat{n}$. Let $\hat{p}_1, \ldots, \hat{p}_{\hat{k}}$ and $\hat{t}_1, \ldots, \hat{t}_{\hat{k}}$ denote these primes, and consider the multiset $N = \{\hat{p}_i + \hat{t}_j \mid i, j \in [\hat{k}]\}$. Notice that N contains $\hat{k}^2$ elements, each of them between $\frac{11}{12}\hat{n}$ and $\frac{13}{12}\hat{n}$. Therefore, there must exist some $n \in \mathbb{N}$ appearing in N at least $\ell \triangleq \frac{\hat{k}^2}{\hat{n}/6} = \frac{6\varepsilon^2\hat{n}}{\log^2 \hat{n}}$ times.

Let $(\hat{p}_{i_1}, \hat{t}_{j_1}), \ldots, (\hat{p}_{i_\ell}, \hat{t}_{j_\ell})$ be the pairs corresponding to this n, namely, $\hat{p}_{i_h} + \hat{t}_{j_h} = n$ for all $h \in [\ell]$. It is clear that $\hat{p}_{i_h} \neq \hat{t}_{j_{h'}}$ for all $h \neq h' \in [\ell]$, since the ranges of $\hat{p}$'s and $\hat{t}$'s are disjoint. Notice that $\hat{p}_{i_h} \neq \hat{p}_{i_{h'}}$ and $\hat{t}_{i_h} \neq \hat{t}_{i_{h'}}$ also hold for all $h \neq h' \in [\ell]$, since all pairs must sum to n. Let β be such that exactly[2] $k \triangleq \ell/2$ of the pairs $(\hat{p}_{i_h}, \hat{t}_{j_h})$ satisfy $\hat{p}_{i_h} < \beta n$.

Denote those k pairs by $(p_1, t_1), \ldots, (p_k, t_k)$, and the remaining k pairs by $(p'_1, t'_1), \ldots, (p'_k, t'_k)$. Let α be such that $\alpha n = \min_{i\in[k]} p_i - 1$, and let γ be such that $\gamma n = \max_{i\in[k]} p'_i + 1$. Clearly, $\alpha < \beta < \gamma$. Since $n \in (\frac{11}{12}\hat{n}, \frac{13}{12}\hat{n})$ and $\max_{i\in[k]} p'_i < \hat{n}/3$ we also have $\gamma < 1/2$. Similarly, $\min_{i\in[k]} p_i \geq \hat{n}/4$ and so $\alpha > 1/5$. Setting $\delta = 5\varepsilon^2$, the bound $2k \geq \delta n \,/\, \log^2 n$ follows from $n \in (\frac{11}{12}\hat{n}, \frac{13}{12}\hat{n})$ as well. □

Proof of Lemma 9.4.3. Let $n \in \mathbb{N}$ be one of those for which Lemma 9.4.6 holds. Let $R_1 = \{p_1, \ldots, p_k\}$, $T_1 = \{t_1, \ldots, t_k\}$, $R_2 = \{p'_1, \ldots, p'_k\}$ and $T_2 = \{t'_1, \ldots, t'_k\}$. The conditions in Lemma 9.4.6 imply $R_1, R_2 \subseteq (\frac{1}{5}n, \frac{1}{2}n)$ and $R_1 \cap R_2 = \emptyset$.

Let $a \in \{1, 2\}$ and $r \in R_a$. Consider a single component

$$Q_h = \{(i, j_1), \ldots, (i, j_{|Q_h|})\}$$

in the partition of Q. Notice that if Q_h is *not* r-collision-free, then there must be a pair $j \neq j' \in \{j_1, \ldots, j_{|Q_h|}\}$ that satisfies either $j - j' \equiv_r 0$ or $j - j' \equiv_{n-r} 0$ (depending on which cycle contains element i). Let R be the set of all $r \in R_a \cup T_a$ for which some pair j, j' satisfies $j - j' \equiv_r 0$. Since R contains only primes that are greater than $n/5$ and $j \neq j'$, the inequality

$$|j - j'| \geq \prod_{r\in R} r \geq (n/5)^{|R|}$$

[2] We assume without loss of generality that ℓ is even. If not, drop one pair.

must hold. On the other hand $|j - j'| \leq m$, so $|R| \leq \frac{\log m}{\log(n/5)}$.

Consequently, the number of different $r \in R_a$ for which *some* pair $j, j' \in \{j_1, \ldots, j_{|Q_h|}\}$ satisfies $j - j' \equiv_r 0$ or $j - j' \equiv_{n-r} 0$ is bounded by $2|Q_h|^2 \frac{\log m}{\log(n/5)}$. This means that for a random $r \in R_a$, the probability that any particular Q_h is not r-collision-free is at most $2\frac{|Q_h|^2 \log m}{|R_a| \log(n/5)}$, and by the union bound,

$$\Pr_{r \in R_a}\left[Q \text{ is } \textit{not} \text{ component-wise } r\text{-collision-free}\right] \leq$$

$$\frac{2(\sum_{h\in[\ell]} |Q_h|^2) \log m}{|R_a| \log(n/5)} \leq \frac{2|Q|^2 \log m}{|R_a| \log(n/5)}.$$

The lemma follows since $|R_a| = \Omega(n \,/\, \log^2 n)$ and $|Q| = o\left(\sqrt{\frac{n}{\log m \log n}}\right)$. □

Proof of Lemma 9.4.4

Let $n \in \mathbb{N}$ be large enough, and let $r \in (\frac{1}{5}n, \frac{1}{2}n)$. Fix a pair of components $Q_h = \{(i, j_1), \ldots, (i, j_{|Q_h|})\}$ and $Q_g = \{(i', j'_1), \ldots, (i', j'_{|Q_g|})\}$ in the aforementioned partition of Q. We now bound the probability, taken over random $\pi \in \mathcal{G}_n^r$, that Q_h and Q_g are *not* π-disjoint.

Notice that when picking a random $\pi \in \mathcal{G}_n^r$, either i and i' belong to different cycles in π (and therefore Q_h and Q_g are π-disjoint), or else π locates both i and i' on the same cycle, where the positions of i and i' are distributed uniformly at random. Both cycles in π are of length greater than $n/5$, hence the probability that Q_h and Q_g are not disjoint is at most $\frac{|Q_h||Q_g|}{n/5}$.

Taking the union bound on all pairs of components we derive an upper bound on $\Pr_{\pi\in\mathcal{G}_n^r}\left[Q \text{ is not } \pi\text{-disjoint}\right]$ of

$$\sum_{h\neq g\in[\ell]} \frac{|Q_h||Q_g|}{n/5} \leq \frac{(\sum_{h\in[\ell]} |Q_h|)(\sum_{g\in[\ell]} |Q_g|)}{n/5}$$
$$= 5q^2/n.$$

The lemma follows by plugging in the value of q.

9.4.7. Remark. Notice that the lower bound we proved does not work for *any* large enough n. But since the n's for which it works are densely spread (for any $\hat{n} \geq n_0$ there is $n = (1 \pm \frac{1}{12})\hat{n}$ for which it works), we can extend the lower bound to work for all sufficiently large n by padding with unit cycles (fixed points).

9.4.2 Upper bound

We deal now with the restricted-oracle case.

9.4.8. Theorem (Chakraborty et al. [CGM11a]). *Assume we are given an upper bound n on the cycle length r. Then there is a randomized algorithm that finds r by making*

$$O\left(\min\left(\frac{n}{m}+\sqrt{n}, \frac{n}{\log m}+\frac{\log n}{\log\log\log n}\right)\right)$$

queries to an m-restricted oracle.

The left bound in the min expression is better when $m = 2^{O(\sqrt{n})}$. The proof of the theorem will follow from the next few lemmas.

9.4.9. Lemma. *Let $\mathcal{A}$ be an adaptive algorithm that finds r by making q queries to the unrestricted oracle and let g be the largest position that $\mathcal{A}$ queries. Then for any $m > 0$ there is an algorithm that finds r with $q + \frac{g}{m}$ queries to the m-restricted oracle.*

Proof. Let us start assuming, for the sake of simplicity, that $\mathcal{A}$ is non-adaptive. Thus g is known at the start of the algorithm. Let $g = am + b$, where $b < m$. The algorithm can make queries to all positions of form cm for all $c \le a$ at the start. This takes $\frac{g}{m}$ many queries. Once all these queries are made, by the definition of the m-restricted oracle, for any $i \le g$ the algorithm can obtain the value of QUERY(i) with at most one call to the m-restricted oracle. Since all the queries that the algorithm $\mathcal{A}$ makes are at most g, the new algorithm can simply simulate the algorithm $\mathcal{A}$ while making queries to the m-restricted oracle. Thus the total query complexity will be $q + \frac{g}{m}$.

Note that the algorithm does not have to know g beforehand, as it can query position cm only when the algorithm $\mathcal{A}$ queries some i such that $i > cm$. Therefore the conversion works too when $\mathcal{A}$ is adaptive. □

We present two algorithms for the restricted-m case. The first one is better for $m < 2^{O(\sqrt{n})}$; the second one is better for $m \ge 2^{\Omega(\sqrt{n})}$.

Small jump length m

This algorithm uses the baby steps-giant steps method of Shanks [Sha71, Sut07].

9.4.10. Lemma. *Let $n > 0$ and define $a \triangleq \lceil \sqrt{n} \rceil$, $B \triangleq \{0, 1, 2, \ldots, a-1\}$ and $G \triangleq \{a, 2a, 3a, \ldots, a^2\}$. If $r \le n$, then there are $b \in B$ and $g \in G$ such that $g - b = r$.*

Algorithm 12 – find r

```
1: procedure FindPeriodByShortJumps(n)
2:     a ← ⌈√n⌉
3:     for i ∈ {0, ..., a − 1} do
4:         b_i ← i
5:         g_i ← (i + 1) · a.
6:     query each element of ⋃_i {b_i, g_i}
7:     return the smallest positive value of (a · (j + 1) − i) among the pairs
               {(i, j) | query(b_i) = query(g_j)}
```

Proof. Clear if $r \in G$. Otherwise write $r = aq + t$, where $0 \le q < a$ and $0 < t < a$. Taking $b = a - t \in B$ and $g = a(q+1) \in G$, we have $r + b = g$. □

Procedure FindPeriodByShortJumps(n) needs to make $|B|+|G| = O(\sqrt{n}\,)$ queries. Note that the maximum position queried is $a^2 \le n + 2\sqrt{n}$. So from Lemma 9.4.9 we have the following lemma.

9.4.11. Lemma. *For any $m > 0$ there is a deterministic, non-adaptive order-finding algorithm that makes $O\big(\sqrt{n} + \frac{n}{m}\big)$ queries to an m-restricted oracle.*

Large jump length $m \ge 2^{\Omega(\sqrt{n}\,)}$

To begin we need the following lemma.

9.4.12. Lemma. *Let c be a large enough constant. If $m \ge 2^{c\sqrt{n}}$, there exists a partition of $\mathcal{P}_n$ into $A_0, \ldots, A_k$ with the following properties:*

1. *$A_0 \triangleq \mathcal{P}_n \cap [1, \sqrt{n}\,]$.*
2. *For all $0 \le i \le k$, $\nu_{A_i}(\mathrm{lcm}([n])) < \sqrt{m}$.*
3. *$k = O(n \,/\, \log m)$.*

Proof. For any set $A \subseteq \mathcal{P}_n$, we have

$$\nu_A(\mathrm{lcm}([n])) = \prod_{p \in A} p^{\lfloor \log_p(n) \rfloor} \le n^{|A|},$$

because $p^{\lfloor \log p(n)}$ is the highest power of p that divides some integer in $[n]$. So if we take $A_1, A_2, \ldots,$ to be consecutive subsets of $\mathcal{P}_n \setminus A_0$, each containing $\lfloor \log m/(2 \log n) \rfloor - 1$ primes (except possibly the last), the second condition is satisfied for all $i > 0$. Now the Prime Number Theorem implies that the number of sets in such a partition is $|\mathcal{P}_n|/(\log m/(2 \log n)) = O(n \,/\, \log m)$ (which gives us the third condition), and also that $|A_0| = O(\sqrt{n} \,/\, \log n)$, implying $\nu_{A_0}(\mathrm{lcm}([n])) = 2^{O(\sqrt{n}\,)} \le \sqrt{m}$ for large enough c. □

Algorithm 13 find r when $m \geq 2^{c\sqrt{n}}$

```
1: procedure FindPeriodByLongJumps(n)
2:     find the partition of P_n into A_0, ..., A_k (as in Lemma 9.4.12)
3:     for i = 1 to k do
4:         if query(ν_{A_0 ∪ A_i}(lcm([n]))) = query(0) then
5:             N ← ν_{A_0 ∪ A_i}(lcm([n]))
6:             use the algorithm FindPeriod(n) from Section 9.3.3 to find r,
                   replacing each query(i) with query(i mod N).
7:             return r
8:
```

9.4.13. Lemma. *Let c be as in Lemma 9.4.12. If $m \geq 2^{c\sqrt{n}}$ then Algorithm 13 makes $O(n / \log m + \log n / \log\log\log n)$ queries to an m-restricted oracle and outputs the period r.*

Proof. The first property of the partition implies that at most one A_i with $i > 0$ contains a prime divisor of r. In any case the **for** loop will find an i such that $r \mid \nu_{A_0 \cup A_i}(\mathrm{lcm}([n]))$. So after a suitable i has been found, we know that r divides $N \triangleq \nu_{A_0 \cup A_I}(\mathrm{lcm}([n])) < \sqrt{m} \times \sqrt{m} = m$ by the second property, and therefore for all i, if $i \equiv i' \pmod{N}$ then $\textsc{query}(i) = \textsc{query}(i')$. Hence by using the algorithm of Section 9.3.3, with each query to position i replaced by a query to position $(i \bmod N)$, r is found with $O(\log n / \log\log\log n)$ additional queries.

It is clear that the main loop spends at most $O(k) = O(n / \log m)$ queries (by the third property) on checking the **if** condition inside the **for** loop. So the total number of queries made by FindPeriodByLongJumps(n) is $O(n / \log m + \log n / \log\log\log n)$. Also note that the maximum position the algorithm queries is

$$\max_i \{\nu_{A_0 \cup A_i}(\mathrm{lcm}([n]))\} < m.$$

So from Lemma 9.4.9 we have an algorithm that finds r by making at most $O(n / \log m + \log n / \log\log\log n)$ queries to an m-restricted oracle. □

9.5 Extensions and consequences

9.5.1 Cycle finding

In the cycle finding problem, we do not know the starting position s of the sequence, which may be non-zero. We prove the following.

9.5.1. Theorem (Chakraborty et al. [CGM11a]). *The query complexity of cycle finding for m-restricted oracles O_f^m is between*

$$\Omega(\log s + s/m + \log r / \log\log r + \sqrt{r/(\log m \log r)} + r/m)$$

and

$$O\left(\log s + s/m + \log r \,/\, \log\log\log r + r \,/\, \log m\right).$$

The query complexities for unrestricted oracles can be obtained by eliminating the terms involving m.

Proof. In view of our bounds for order finding, we only need to justify the added term $\Theta(\log s + s/m)$ in both the upper and lower bounds.

The term $\Omega(\log s)$ is clear, even under the promise that the period is $r = 1$, since the problem of determining the index of the first '1' in a sequence consisting of s zeroes followed by an infinite number of ones can be reduced to it.[3] The term $\Omega(s/m)$ is clear from the mere fact that the algorithm needs to reach position $s + r > s$ to detect any cycle.

For the upper bound, assume temporarily that s is known and we can get tight upper bounds on n within the stated query complexity, where b is said to be a tight upper bound for a if $a \leq b = O(a)$. (We will see how this can be done Section 9.5.3). Then we can replace each QUERY(i) by FINDPERIOD to QUERY$(i + s)$, which results in the same outcome, and find r with $O(\log n \,/\, \log\log\log n + n \,/\, \log m)$ queries, which is of the right order as $n = O(r)$. □

9.5.2 Lower bound for discrete log via the generic group model

In earlier work the query complexity of group properties has been studied in a different (but related) model – the *generic group* model [BB99, Sut07]. In this setting, one has access to a "black box" that allows one to find the identity element of the group, compute the inverse of an element, and multiply two elements. The black box returns certain labels to which the algorithm is not allowed to ascribe any meaning, except for equality comparisons (the label determines uniquely the group element, but the precise bijection is a priori unknown to the algorithm).

The best known algorithm to compute the order of an element in this model was found by Sutherland [Sut07]. His algorithm runs in time $O(\sqrt{r \,/\, \log\log r}\,)$, where r is the order of the group. In particular, its query complexity is bounded by $O(\sqrt{r \,/\, \log\log r}\,)$. A lower bound that is polynomial in r was shown by Babai and Beals [BB99], and Sutherland shows a lower bound of $\Omega(r^{1/3})$ [Sut07]. In contrast, for the similar problem of discrete log over generic groups there are tight $\Theta(\sqrt{r})$ bounds (the lower bound is by Shoup [Sho97], and the upper bound by Shanks [Sha71]). Therefore, discrete log is strictly harder than order finding in the

[3] Note that it is required here to identify the precise value of s. If an upper bound on s were all that is needed, the query complexity could grow arbitrarily slowly as a function of s (albeit not necessarily of r).

generic group model, but their complexities are polynomially related. In contrast, there is an exponential separation between the two in our model.

Indeed, given $a \in G$, we know that it is possible to find the order r of the (cyclic) group generated by a with $O(\sqrt{r \,/\, \log\log r}\,)$ queries in the generic group model. Afterwards, any of the "jumps" allowed in our model (of the form a^i for some $i \geq 0$ and a that was obtained previously) can be simulated with $O(\log r)$ queries to a generic group oracle by the standard logarithmic exponentiation algorithm, after reducing i modulo r. So the existence of an algorithm making q queries for the discrete-log problem in our model implies an algorithm for the discrete-log problem in the generic group model making $O(\sqrt{r \,/\, \log\log r} + q \log r)$ queries; by the $\Omega(\sqrt{r})$ lower bound of Shoup, one gets $q = \Omega(\sqrt{r} \,/\, \log r)$. We have thus proved the following.

9.5.2. Theorem (Chakraborty et al. [CGM11a]). *The discrete-log problem with an unrestricted oracle requires $\Omega(\sqrt{r}/\log r)$ queries.*

Interestingly, this is exponentially larger than the $O(\log r \,/\, \log\log\log r)$ upper bound we prove for the order-finding problem.

To apply any of the algorithms presented so far, we need to be able to have at our disposal a good upper bound r' on r and know the value of s. We say that a pair (r', s') is a candidate if $r \leq r'$ and $s \leq s'$. Given a candidate pair r' and s', Procedures FindPeriodByShortJumps(s, n) and FindPeriodByLongJumps(s, n) can be used to build a deterministic procedure check(r', s') to decide if the bounds r' and s' are valid, i.e., $r \leq r'$ and $s \geq s'$, and the same can be said of FindPeriodByLongJumps(s, n)

9.5.3 Obtaining good upper bounds on r and s

To apply any of the algorithms presented so far, we need to be able to have at our disposal a good upper r' on r and know the value of s. Given a candidate pair r' and s', Procedures FindPeriodByShortJumps(s, n) and FindPeriodByLongJumps(s, n) can be used to build a deterministic procedure ValidBounds(r', s') to check if r' and s' are valid bounds, i.e., $r \leq r'$ and $s \geq s'$. We omit the code; note that the probabilistic part of FindPeriod(s', r') only comes into play when s' and r' are good bounds, and could in fact be skipped for this check. The query complexity of this check is $O(f(r') + g(s'))$, where

$$f(r') \triangleq \min(r'/m + \sqrt{r'}, r' \,/\, \log m + \log r' \,/\, \log\log\log r')$$

and

$$g(s') = s'/m.$$

We show that with $O(f(r) + g(s) + \log s)$ queries we can determine the precise values of s and r.

Algorithm 14 – find r and s

```
1: procedure FindPeriodAndStart
2:     for i = 1 to ∞ do
3:         r' = ⌈f^{-1}(2^i)⌉
4:         s' = ⌈g^{-1}(2^i)⌉
5:         if ValidBounds(s', r') then
6:             r ← FindPeriod(s', r')
7:             binary search for s on [0, ∞) by comparing query(s') and query(s'+r)
8:             return (r, s)
```

Note that $f(r)$ and $g(s)$ are strictly increasing functions whose growth rate is bounded above by a polynomial. In particular their inverses satisfy $f(f^{-1}(x)+1) = O(x)$ and $g(g^{-1}(y) + 1) = O(y)$. (We are viewing f and g as continuous, increasing functions defined over the reals). Consider Algorithm 14. Clearly the call to FindPeriod(s', r') will succeed when i reaches the value $i_0 = \max(\lceil \log f(r) \rceil, \lceil \log g(s) \rceil)$. The number of queries made at this point is

$$\sum_{i=1}^{i_0} f(\lceil f^{-1}(2^i) \rceil) + g(\lceil g^{-1}(2^i) \rceil) = \sum_{i=1}^{i_0} O(2^i + 2^i),$$

which is $O(2^{i_0}) = O(\max(f(r), g(s))$. So we can determine r with $O(f(r) + g(s))$ queries. The next line finds s by binary search with $O(\log s)$ queries. Here Lemma 9.4.9 is implicitly used, coupled with the fact that position $r' \geq r$ has been already inspected at this point. (Notice that the binary search must start from scratch and find a tight upper bound on s, as the prior bound s' might be much larger than poly(s).) Therefore the precise values of r and s can be determined after $O(f(r) + g(s) + \log s)$ queries with high probability.

9.6 Summary

We showed that the query complexity of cycle detection (and related problems), when one is allowed to inspect any element of the sequence at will, is sublogarithmic in the period length. The bounds do not quite match however; it would be interesting to close the gap between $\Omega(\log r / \log \log r)$ and $O(\log r / \log \log \log r)$. I strongly suspect that the lower bound is tight in this case. (Of course, it would also be interesting to try to close the bigger, quadratic gap that appears in one of the terms once the parameter m is introduced.)

We also provided algorithms and lower bounds for these problems in a model when the "jump length" is limited; the lower bounds here improve on the existing work.

Appendix A

Some frequently used estimates

We use the following well-known inequalities [Juk11].

$$e^t \geq 1 + t$$

$$(1+x)^n \geq 1 + xn \qquad (n \in \mathbb{Z}, x \in \mathbb{R}, x \geq 1)$$

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) \qquad \text{(Stirling's formula)}$$

$$\frac{2^n}{\sqrt{n}}\left(\sqrt{\frac{2}{\pi}} - O\left(\frac{1}{n}\right)\right) \leq \binom{n}{\lfloor n/2 \rfloor} \leq \frac{2^n}{\sqrt{n}} \cdot \sqrt{\frac{2}{\pi}}$$

$$\binom{n}{k} = \sqrt{\frac{n}{2\pi k(n-k)}}\left(\frac{n}{k}\right)^k \left(\frac{n}{n-k}\right)^{n-k}(1 - O(k^{-1})) \qquad (1 \leq k \leq n/2)$$

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \binom{n}{\leq k} \leq \left(\frac{en}{k}\right)^k \qquad (1 \leq k)$$

$$\frac{e^{-k^2/n - 1/(6k)}}{\sqrt{2\pi k}}\left(\frac{en}{k}\right)^k \leq \binom{n}{k} \qquad (1 \leq k \leq n/2)$$

$$\binom{n}{\leq k} \leq \binom{n}{k}\left(1 + \frac{k}{n - 2k + 1}\right) \qquad (1 \leq k \leq n/2)$$

$$2^{n \cdot H(k/n) - O(\log n)} \leq \binom{n}{\leq k} \leq 2^{n \cdot H(k/n)} \qquad (1 \leq k \leq n/2)$$

$$\binom{n}{\alpha n} = \frac{1 + o(1)}{\sqrt{2\pi\alpha(1-\alpha)n}} \cdot 2^{n \cdot H(\alpha)} \qquad (\alpha n \in \mathbb{N}, 0 < \alpha < 1)$$

Here $H(x)$ stands for Shannon's entropy function:

$$H(x) = x \log\left(\frac{1}{x}\right) + (1-x)\log\left(\frac{1}{1-x}\right)$$

for $x \in (0,1)$, and $H(0) = H(1) = 0$.

Chernoff bounds

Let $X_1, \ldots, X_n$ be $[0,1]$-valued independent random variables with $\mathbb{E}[X_i] = p_i$. Define $X \triangleq \sum_{i=1}^n X_i$ and $p \triangleq \frac{1}{n}\mathbb{E}[X] = \frac{1}{n}\sum p_i$. Then for any $\delta > 0$ we have the following inequalities:

Multiplicative bounds

$$\begin{aligned}
\Pr[X \geq (1+\delta)pn] &\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{pn} \\
&\leq e^{-pn\delta^2/3} && \text{if } \delta \leq 1 \\
\Pr[X \leq (1-\delta)pn] &\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{pn} \leq e^{-pn\delta^2/2} \\
\Pr[|X - pn| \geq \delta pn] &\leq 2e^{-pn\delta^2/3} && \text{if } \delta \leq 1
\end{aligned}$$

Additive bounds

$$\begin{aligned}
\Pr[X \geq n(p+\delta)] &\leq e^{-2n\delta^2} \\
\Pr[X \leq n(p-\delta)] &\leq e^{-2n\delta^2}
\end{aligned}$$

The difference between the multiplicative and the additive forms of the bounds is a ratio of $O(p)$ in the exponent; multiplicative bounds are better for $p = o(1)$, and almost the same for constant p.

If we know $\mathbb{E}[X_i] \leq p_i$ instead, then the bound for $\Pr[X \geq (1+\delta)pn]$ continues to hold; similarly in the opposite direction. See [DP09, MU05, MR95] for in-depth treatments of concentration inequalities and their use in the analysis of randomized algorithms.

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[AB10] N. Alon and E. Blais. Testing boolean function isomorphism. In *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 394–405, 2010.

[ABC+11] N. Alon, E. Blais, S. Chakraborty, D. García–Soriano, and A. Matsliah. Nearly tight bounds for testing function isomorphism. Manuscript, 2011.

[ABS91] N. Alon, L. Babai, and H. Suzuki. Multilinear polynomials and Frankl-Ray-Chaudhuri-Wilson type intersection theorems. *Journal of Combinatorial Theory, Series A*, 58:165–180, 1991.

[ABY08] A. Al-Bashabsheh and A. Yongaçoglu. On the k-pairs problem. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, pages 1828–1832, 2008.

[AC06] N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Information and Computation*, 204(11):1704–1717, 2006.

[AFKS00] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.

[AFNS09] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009.

[AKK+03] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing low-degree polynomials over GF(2). In *Proceedings of the seventh International Workshop on Randomization and Computation (RANDOM)*, pages 188–199, 2003.

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[APY09] N. Alon, R. Panigrahy, and S. Yekhanin. Deterministic approximation algorithms for the nearest codeword problem. In *Proceedings of the 13rd International Workshop on Randomization and Computation (RANDOM)*, pages 339–351, 2009.

[Art10] M. Artin. *Algebra*. Prentice Hall, second edition, 2010.

[AS03] S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23:365–426, 2003.

[AS07] A. Atıcı and R. A. Servedio. Quantum algorithms for learning and testing juntas. *Quantum Information Processing*, 6:323–348, 2007.

[AS08] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM Journal on Computing*, 37:1703–1727, 2008.

[AT00] M. Agrawal and T. Thierauf. The formula isomorphism problem. *SIAM Journal on Computing*, 30:990–1009, 2000.

[AV11] V. Arvind and Y. Vasudev. Isomorphism testing of boolean functions computable by constant depth circuits. In *Proceedings of the The sixth International Conference on Language and Automata Theory and Applications*, 2011.

[AW12] N. Alon and A. Weinstein. Local correction of juntas. *Information Processing Letters*, 112(6):223–226, 2012. To appear.

[AZ10] M. Aigner and G. M. Ziegler. *Proofs from the book*. Springer, fourth edition, 2010.

[Bab81] L. Babai. On the order of uniprimitive permutation groups. *Annals of Mathematics*, 113(3):553–568, 1981.

[Bar89] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BB99] L. Babai and R. Beals. A polynomial-time theory of black-box groups. In *Groups St Andrews 1997 in Bath, I*, London Mathematical Society Lecture Notes 260, pages 30–64, 1999.

[BBM11] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. In *Proceedings of the 26th IEEE Conference on Computational Complexity (CCC)*, 2011.

[BBTN92] L. Babai, R. Beals, and P. Takácsi-Nagy. Symmetry and complexity. In *Proceedings of the 24th ACM Symposium on Theory of Computing (STOC)*, pages 438–449, 1992.

[BC08a] L. Babai and P. Codenotti. Isomorphism of hypergraphs of low rank in moderately exponential time. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 667–676, 2008.

[BC08b] L. Babai and S. Chakraborty. Property testing of equivalence under a permutation group action. Technical report, Electronic Colloquium on Computational Complexity (ECCC) TR–08–04, 2008. To appear in *ACM Transactions on Computation Theory*.

[BCGM12] J. Briët, S. Chakraborty, D. García–Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32:1–19, 2012. Earlier version in *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 462–475, 2010.

[BEHL09] I. Ben-Eliezer, R. Hod, and S. Lovett. Random low degree polynomials are hard to approximate. In *Proceedings of the 13rd International Workshop on Randomization and Computation (RANDOM)*, pages 366–377, 2009.

[Ben65] V. Beneš. Mathematical theory of connecting networks and telephone traffic. *Academic Press*, 1965.

[Ber89] C. Berge. *Hypergraphs*. North-Holland, Amsterdam, The Netherlands, 1989.

[BF92] L. Babai and P. Frankl. *Linear Algebra Methods in Combinatorics, with Applications to Geometry and Computer Science*. University of Chicago, 1992. Unpublished manuscript, available at http://www.cs.uchicago.edu/research/publications/combinatorics.

[BFF+01] T. Batu, L. Fortnow, E. Fischer, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, page 442, 2001.

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

[BFNR08] H. Buhrman, L. Fortnow, I. Newman, and H. Röhrig. Quantum property testing. *SIAM Journal on Computing*, 37(5):1387–1400, 2008.

[BGJ+09a] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. P. Woodruff. Transitive-closure spanners. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 932–941, 2009.

[BGJ+09b] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. P. Woodruff. Transitive-closure spanners of the hypercube and the hypergrid. Technical report, Electronic Colloquium on Computational Complexity (ECCC) TR–09–046, 2009.

[BGM10] H. Buhrman, D. García–Soriano, and A. Matsliah. Learning parities in the mistake-bound model. *Information Processing Letters*, 111(1):16–21, 2010.

[BGV05] A. de Bonis, L. Gasieniec, and U. Vaccaro. Optimal two-stage algorithms for group testing problems. *SIAM Journal on Computing*, 34:1253–1270, 2005.

[Bha08] A. Bhattacharyya. A note on the distance to monotonicity of boolean functions. Technical report, Electronic Colloquium on Computational Complexity (ECCC) TR–08–012, 2008.

[BJKS04] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68:702–732, 2004.

[BKL83] L. Babai, W. M. Kantor, and E. M. Luks. Computational complexity and the classification of finite simple groups. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–171, 1983.

[Bla08] E. Blais. Improved bounds for testing juntas. In *Proceedings of the 12nd International Workshop on Randomization and Computation (RANDOM)*, pages 317–330, 2008.

[Bla09] E. Blais. Testing juntas nearly optimally. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 2009.

[Bla10] E. Blais. Testing juntas: A brief survey. In O. Goldreich, editor, *Property Testing*, pages 32–40, 2010.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC)*, pages 73–83, 1990.

[Blu94] A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM Journal on Computing*, 23(5):990–1000, 1994.

[Blu96] A. Blum. On-line algorithms in machine learning. In *Proceedings of the Workshop on On-Line Algorithms*, pages 306–325, Dagstuhl, 1996.

[BO10] E. Blais and R. O'Donnell. Lower bounds for testing function isomorphism. In *Proceedings of the 25th IEEE Conference on Computational Complexity (CCC)*, pages 235–246, 2010.

[Boc89] A. Bochert. Über die Zahl verschiedener Werte, die eine Funktion gegebener Buchstaben durch Vertauschung derselben erlangen kann. *Mathematische Annalen*, pages 584–590, 1889.

[Bol86] B. Bollobás. *Combinatorics: set systems, hypergraphs, families of vectors and combinatorial probability.* Cambridge University Press, 1986.

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[BV03] A. de Bonis and U. Vaccaro. Constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. *Theoretical Computer Science*, 306(1–3):223–243, 2003.

[BV06] A. de Bonis and U. Vaccaro. Optimal algorithms for two group testing problems, and new bounds on generalized superimposed codes. *IEEE Transactions on Information Theory*, 52(10):4673–4680, 2006.

[BW02] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21–43, 2002.

[BWY11] E. Blais, A. Weinstein, and Y. Yoshida. Partially symmetric functions are efficiently isomorphism-testable. Technical report, Computing Research Repository, 2011.

[Cam81] P. J. Cameron. Finite permutation groups and finite simple groups. *Bulletin of the London Mathematical Society*, 13:1–22, 1981.

[Cam99] P. J. Cameron. *Permutation groups*, volume 45 of *London Mathematical Society Student Texts*. Cambridge University Press, New York and London, 1999.

[CFGM12] S. Chakraborty, E. Fischer, D. García–Soriano, and A. Matsliah. Junto-symmetric functions, hypergraph isomorphism, and crunching. In *Proceedings of the 27th IEEE Conference on Computational Complexity (CCC)*, 2012. To appear.

[CFMW10] S. Chakraborty, E. Fischer, A. Matsliah, and R. de Wolf. New Results on Quantum Property Testing. In *Proceedings of the 30th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 145–156, 2010.

[CG04] H. Chockler and D. Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90:301–305, 2004.

[CGGM11] S. Chakraborty, D. García–Soriano, Y. Goldhirsh, and A. Matsliah. Deciding and approximating isomorphism efficiently for specific classes of boolean functions. Manuscript, 2011.

[CGM11a] S. Chakraborty, D. García–Soriano, and A. Matsliah. Cycle detection, order finding and discrete log with jumps. In *Proceedings of the second Symposium on Innovations in Theoretical Computer Science (ITCS)*, pages 284–297, 2011.

[CGM11b] S. Chakraborty, D. García–Soriano, and A. Matsliah. Efficient sample extractors for juntas with applications. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 545–556, 2011.

[CGM11c] S. Chakraborty, D. García–Soriano, and A. Matsliah. Nearly tight bounds for testing function isomorphism. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1683–1702, 2011.

[CGM11d] S. Chakraborty, D. García–Soriano, and A. Matsliah. Notes on testing k-parities. Manuscript, 2011.

[CGR02] M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.

[CK91] P. Clote and E. Kranakis. Boolean functions, invariance groups, and parallel complexity. *SIAM Journal on Computing*, 20(3):553–590, 1991.

[CK02] P. Clote and E. Kranakis. *Boolean Functions and Computation Models*. EATCS Series. Springer-Verlag, 2002.

[Cla92] M. Clausen. Almost all boolean functions have no linear symmetries. *Information Processing Letters*, 41:291–292, 1992.

[Cle04] R. Cleve. The query complexity of order-finding. *Information and Computation*, 192(2):162–171, 2004.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the Second Inetrnational Congress of Logic, Methodology and Philosophy of Science*, Amsterdam, 1965. North-Holland.

[CSZ00] A. Czumaj, C. Sohler, and M. Ziegler. Property testing in computational geometry. In *Proceedings of the 8th Annual European Symposium on Algorithms*, ESA '00, pages 155–166, 2000.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[DGL+99] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of the third International Workshop on Randomization and Computation (RANDOM)*, pages 97–108, 1999.

[DH00] D. Du and F. H. Hwang. *Combinatorial group testing and its applications*. Series on applied mathematis. World Scientific Publishing Co. Pte. Ltd., 2000.

[Die05] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition, 2005.

[DLM+07] I. Diakonikolas, H. K. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. A. Servedio, and A. Wan. Testing for concise representations. In *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 549–558, 2007.

[DLM+08] I. Diakonikolas, H. K. Lee, K. Matulef, R. A. Servedio, and A. Wan. Efficiently testing sparse GF(2) polynomials. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 502–514, 2008.

[Dor43] R. Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):pp. 436–440, 1943.

[DP09] D. P. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, New York, USA, 2009.

[Ebe89] W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18:955–976, 1989.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[EFF82] P. L. Erdős, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of two others. *Journal of Combinatorial Theory, Series A*, 33(2):158–166, 1982.

[EKK+00] F. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000.

[Fis01] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the EATCS*, 75:97, 2001.

[Fis04] E. Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004.

[Fis05] E. Fischer. The difficulty of testing for isomorphism against a graph that is given in advance. *SIAM Journal on Computing*, 34(5):1147–1158, 2005.

[FKR+04] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

[FLN+02] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002.

[Flo67] R. W. Floyd. Nondeterministic algorithms. *Journal of the ACM*, 14(4):636–644, 1967.

[FM08] E. Fischer and A. Matsliah. Testing graph isomorphism. *SIAM Journal on Computing*, 38(1):207–225, 2008.

[FMS10] E. Fischer, A. Matsliah, and A. Shapira. Approximate hypergraph partitioning and applications. *SIAM Journal on Computing*, 39:3155–3185, 2010.

[FNS04] E. Fischer, I. Newman, and J. Sgall. Functions that have read-twice constant width branching programs are not necessarily testable. *Random Structures and Algorithms*, 24(2):175–193, 2004.

[FR87] P. Frankl and V. Rödl. Forbidden intersections. *Transactions of the American Mathematical Society*, pages 259–286, 1987.

[FRPU94] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

[Für96] Z. Füredi. On r-cover-free families. *Journal of Combinatorial Theory, Series A*, 73:172–173, 1996.

[FW81] P. Frankl and M. Wilson. Intersection theorems with geometric consequences. *Combinatorica*, 1(4):357–368, 1981.

[GGL+00] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

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

[GGR98] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

[GMW11] D. García–Soriano, A. Matsliah, and R. de Wolf. Manuscript. 2011.

[Gol10] O. Goldreich. On testing computability by small width OBDDs. In *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 574–587, 2010.

[GOS+09] P. Gopalan, R. O'Donnell, R. A. Servedio, A. Shpilka, and K. Wimmer. Testing Fourier dimensionality and sparsity. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 500–512, 2009.

[Gro99] V. Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit Ramsey graphs. *Combinatorica*, 20:2000, 1999.

[GRV11] E. Grigorescu, L. Reyzin, and S. Vempala. On noise-tolerant learning of sparse parities and related problems. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, pages 413–424, 2011.

[HB01] N. J. Hopper and M. Blum. Secure human identification protocols. In *Proceedings of the seventh International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 52–66, 2001.

[HK08] S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. *Random Structures and Algorithms*, 33(1):44–67, 2008.

[HKL06] N. J. A. Harvey, R. Kleinberg, and A. R. Lehman. On the capacity of information networks. *IEEE/ACM Transactions on Networking*, 14:2345–2364, 2006.

[HS69] A. Hajnal and E. Szemerédi. Proof of a conjecture of Paul Erdős. In *Combinatorial Theory and its Applications*, pages 601–623, 1969.

[HV06] A. Healy and E. Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 672–683, 2006.

[HW07] J. Håstad and A. Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[HW08] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, New York, sixth edition, 2008.

[INR10] P. Indyk, H. Q. Ngo, and A. Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1126–1142, 2010.

[Juk11] S. Jukna. *Extremal Combinatorics: with applications in computer science*. Springer, second edition, 2011.

[Kis98] A. Kisielewicz. Symmetry groups of boolean functions and constructions of permutation groups. *Journal of Algebra*, 199(2):379–403, 1998.

[KK08] H. A. Kierstead and A. V. Kostochka. A short proof of the Hajnal-Szemerédi theorem on equitable colouring. *Combinatorics, Probability and Computing*, 17:265–270, 2008.

[Kla00] H. Klauck. On quantum and probabilistic communication: Las Vegas and one-way protocols. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, pages 644–651, 2000.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KS64] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964.

[KS92] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5:545–557, 1992.

[KS05] P. Keevash and B. Sudakov. Set systems with restricted cross-intersections and the minimum rank of inclusion matrices. *SIAM Journal on Discrete Mathematics*, 18(4):713–727, 2005.

[KS06] A. R. Klivans and R. A. Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7:587–602, 2006.

[KV94] M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994.

[Lin98] J. von Lint. *Introduction to Coding Theory*. Springer, third edition, 1998.

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[Lit89] N. Littlestone. From on-line to batch learning. In *Proceedings of the second Conference on Learning Theory (COLT)*, pages 269–284, 1989.

[LLZ11] M. L. Leung, Y. Li, and S. Zhang. Tight bounds on the randomized communication complexity of symmetric XOR functions in one-way and SMP models. Technical report, Electronic Colloquium on Computational Complexity (ECCC) TR–11-011, 2011.

[LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40:607–620, 1993.

[LN11] O. Lachish and I. Newman. Testing periodicity. *Algorithmica*, 60(2):401–420, 2011.

[LR01] E. Lehman and D. Ron. On disjoint chains of subsets. *Journal of Combinatorial Theory, Series A*, 94(2):399–404, 2001.

[Lub90] A. Lubiw. Counterexample to a conjecture of Szymanski on hypercube routing. *Information Processing Letters*, 35(2):57–61, 1990.

[Luk99] E. M. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. In *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, pages 652–658, 1999.

[LW01] J. Lint and R. Wilson. *A course in combinatorics*. Cambridge University Press, 2001.

[Mac98] A. J. Macula. Probabilistic Nonadaptive and Two-Stage Group Testing with Relatively Small Pools and DNA Library Screening. *Journal of Combinatorial Optimization*, 2(4):385–397, 1998.

[MORS09] K. Matulef, R. O'Donnell, R. Rubinfeld, and R. A. Servedio. Testing ± 1-weight halfspaces. In *Proceedings of the 13rd International Workshop on Randomization and Computation (RANDOM)*, pages 646–657, 2009.

[MORS10] K. Matulef, R. O'Donnell, R. Rubinfeld, and R. A. Servedio. Testing halfspaces. *SIAM Journal on Computing*, 39:2004–2047, 2010.

[MOS04] E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004. Also known as "Learning juntas".

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[MS77] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[MT11] M. Mézard and C. Toninelli. Group testing with random pools: Optimal two-stage algorithms. *IEEE Transactions on Information Theory*, 57(3):1736–1745, 2011.

[MU05] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[Nay99] A. Nayak. Optimal lower bounds for quantum automata and random access codes. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 369–376, 1999.

[Neu28] J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[NS92] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:462–467, 1992.

[NSS95] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 182–191, 1995.

[Pol75] J. M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 15:331–334, 1975.

[PR08] E. Porat and A. Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 748–759, 2008.

[PRS02] M. Parnas, D. Ron, and A. Samorodnitsky. Testing basic boolean formulae. *SIAM Journal on Discrete Mathematics*, 16(1):20–46, 2002.

[PS10] T. Pitassi and R. Santhanam. Effectively polynomial simulations. In *Proceedings of the first Symposium on Innovations in Theoretical Computer Science (ITCS)*, pages 370–382, 2010.

[Ras99] S. Raskhodnikova. Monotonicity testing. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1999.

[Raz92] A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106:385–390, 1992.

[RCW75] D. Ray-Chaudhuri and R. Wilson. On t-designs. *Osaka Journal of Mathematics*, pages 737–744, 1975.

[RL05] A. Rasala-Lehman. *Network coding.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 2005.

[Ron08] D. Ron. Property testing: A learning theory perspective. *Foundations and Trends in Theoretical Computer Science*, 1:307–402, 2008.

[Ron10] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5:73–205, 2010.

[RS96] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25:252–271, 1996.

[RS12] R. Rubinfeld and A. Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, pages 1562–1588, 2012.

[RT11] D. Ron and G. Tsur. On approximating the number of relevant variables in a function. In *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM)*, pages 676–687, 2011.

[Rus94] M. Ruszinkó. On the upper bound of the size of the r-cover-free families. *Journal of Combinatorial Theory, Series A*, 66:302–310, 1994.

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1988.

[Ser11] R. A. Servedio. Testing by implicit learning: A brief survey. In O. Goldreich, editor, *Property Testing*, volume 6390 of *Lecture Notes in Computer Science*, pages 197–210. Springer Berlin / Heidelberg, 2011.

[Sha49] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28:59–98, 1949.

[Sha71] D. Shanks. Class number, a theory of factorization, and genera. In *Analytic Number Theory, Proceedings of Symposia on Pure Mathematics*, volume 20, pages 415–440, 1971.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th international conference on theory and application of cryptographic techniques (EUROCRYPT)*, pages 256–266, 1997.

[SS90] J. P. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal on Computing*, 19(5):775–786, 1990.

[SSS95] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[Sto83] L. Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th ACM Symposium on Theory of Computing (STOC)*, pages 118–126, 1983.

[Sto10] A. Stothers. *On the Complexity of Matrix Multiplication*. PhD thesis, University of Edinburgh, 2010.

[Sud01] M. Sudan. Coding theory: Tutorial and survey. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.

[Sut07] A. V. Sutherland. *Order computations in generic groups*. PhD thesis, MIT, 2007.

[Sze76] E. Szemerédi. Regular Partitions of graphs. *Colloques Internationaux CNRS*, 260:399–401, 1976.

[Szy89] T. H. Szymanski. On the permutation capability of a circuit-switched hypercube. In *Proceedings of the first International Conference on Parallel Processing (ICPP)*, pages 103–110, 1989.

[Tre04] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[Val12] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas with noise. Technical report, Electronic Colloquium on Computational Complexity (ECCC) TR–12-006, 2012.

[Wie64] H. Wielandt. *Finite permutation groups*. Academic Press, New York and London, 1964.

[Wig07] S. Wigert. Sur l'order de grandeur du nombre des diviseurs d'un entier. *Arkiv för Matematik, Astronomi och Fysik*, 3:1–9, 1907.

[Wika] Wikipedia, The Free Encyclopedia. Cycle detection. Available online at `http://en.wikipedia.org/wiki/Cycle_detection`. Accessed November 2nd, 2011.

[Wikb] Wikipedia, The Free Encyclopedia. Discrete log. Available online at `http://en.wikipedia.org/wiki/Discrete_log`. Accessed November 2nd, 2011.

[Wil11] V. V. Williams. Breaking the Coppersmith-Winograd barrier. Unpublished manuscript, Nov 2011.

[Wol06] R. de Wolf. The one-way and SMP communication complexity of k-disjointness. Manuscript, 2006.

[Wol08] R. de Wolf. *A Brief Introduction to Fourier Analysis on the Boolean Cube*. Number 1 in Graduate Surveys. Theory of Computing, 2008.

[Xia05] W. Xiao. Linear symmetries of boolean functions. *Discrete Applied Mathematics*, 149(13):192–199, 2005.

[Yao77] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[Yao79] A. C.-C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11st ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

[ZS10] Z. Zhang and Y. Shi. On the parity complexity measures of boolean functions. *Theoretical Computer Science*, 411:2612–2618, 2010.

Name index

Subject index

Samenvatting

Hoe kunnen we rekenkundige problemen oplossen wanneer we simpelweg niet genoeg tijd hebben om alle invoer te verwerken? Bijvoorbeeld, gegeven een rij getallen, kunnen we bepalen of deze zich gaan herhalen door slechts naar een paar getallen te kijken? Of, gegeven toegang tot een booleaanse functie f, hoeveel aanroepen van f hebben we nodig om te testen of deze functie monotoon is? En hoe zouden we kunnen ontdekken of f vrijwel gelijk is aan een andere, vooraf bekende, functie g?

Centrale doelen van de theoretische informatica zijn het inzicht krijgen in de grenzen van de rekenkracht van verschillende rekenmodellen, en het karakteriseren van de middelen die nodig zijn om bepaalde problemen op te lossen. Het soort problemen hierboven genoemd is bij uitstek geschikt om bestudeerd te worden in het *property testing* model (het testen van eigenschappen), en het is door deze lens dat wij die vraagstukken onderzoeken. Bij property testing moeten algoritmen onderscheid maken tussen objecten die een gewenste eigenschap hebben, en objecten die daar ver vandaan zijn. We zoeken gerandomiseerde testers die de problemen met zo min mogelijk functie-aanroepen oplossen (*bovengrenzen*), samen met rigoureuze bewijzen die laten zien waarom er geen significant betere oplossingen kunnen bestaan (*ondergrenzen*). De resultaten maken gebruik van technieken uit kansrekening, grafentheorie, extremale combinatoriek, de studie van permutatiegroepen, coderingstheorie, de analyse van booleaanse functies en getallentheorie.

Het beginpunt is ons werk aan het probleem van het testen van functie-isomorfisme in hoofdstuk 2. Twee booleaanse functies met n-bit invoer zijn isomorf wanneer ze hetzelfde zijn, na een zekere permutatie van de n invoervariabelen. Er wordt vooral gekeken naar de situatie wanneer g bekend is en f aangeroepen wordt. Het is bekend dat deze taak uitgevoerd kan worden met $\widetilde{O}(n)$ aanroepen, maar dit is exponentieel groter dan de beste ondergrenzen van voorgaand werk. Hier sluiten we de kloof door een bijna-optimale adaptieve ondergrens te geven van $\Omega(n)$ aanroepen voor de slechtst-mogelijke functies f. Verscheidene varianten

van het probleem worden ook besproken.

Hiernaast laten we in hoofdstuk 3 zien dat wanneer f een k-junta is (dat betekent dat de functiewaarde bepaald wordt door slechts k van de n invoervariabelen), de complexiteit van isomorfisme testen met f is gereduceerd naar $\widetilde{O}(k)$. (In contrast, één van onze andere resultaten is dat wanneer we de schijnbaar zwakke restrictie opleggen dat de tester slechts eenzijdige fouten kan maken, de complexiteit van isomorfisme testen met k-juntas wordt ruwweg $\log \binom{n}{k}$, wat veel groter is voor kleine k.) Hierbij construeren we onafhankelijk interessante objecten met de naam *sample extractors*. Dit zijn efficiënte algoritmen die ons in staat stellen om steekproeven te doen van de waarheidstabel van de "kern"-functie op k variabelen die de gegeven k-junta functie bepaalt.

Vervolgens geven we een gedeeltelijke karakterisatie van de verzameling functies waartegen het onmogelijk is om isomorfisme te testen met een constant aantal functie-aanroepen. We laten in hoofdstuk 4 zien dat, voor *elke* functie f met polynomiaal veel verschillende permutaties, isomorfisme met betrekking tot f testbaar is met een constant aantal aanroepen. Deze stelling is een uitbreiding op voorgaande resultaten over het testen van junta's, en beschrijft alle functies waarvan tot nu toe bekend was dat isomorfisme testen makkelijk is. Gerelateerd hieraan, en kijkende naar de overeenkomst tussen het testen van functie-isomorfisme en het testen van *hypergraaf-isomorfisme*, richten we onze aandacht naar het testen van isomorfisme voor uniforme hypergrafen. We karakteriseren de klasse van hypergrafen van constante rang waarvoor isomorfisme efficiënt getest kan worden, dit is een generalisatie van een resultaat van Fischer (STOC'04) over isomorfismen in grafen.

In hoofdstuk 5 leggen we een verbinding aan met *groeptesten*, en zien dat ideeën uit het testen van isomorfisme gebruikt kunnen worden bij het bestuderen van een natuurlijke relaxatie van problemen uit groeptesten; zowel de methoden voor het verkrijgen van ondergrenzen en de algoritmen zijn bruikbaar. We bepalen de precieze complexiteit van het gerelaxeerde groeptest probleem voor niet-adaptieve algoritmen, op een constante factor na. Het vraagstuk van het krijgen van expliciete ondergrenzen voor de problemen wordt ook aangepakt; het blijkt dat pariteitsfuncties (XOR's van deelverzamelingen van de invoervariabelen) een voorbeeld zijn van de ongunstigste ondergrenzen voor het testen van functie-isomorfisme, zowel voor eenzijdige als tweezijdige testers.

We vervolgen met het bespreken van andere property testing problemen in hoofdstuk 6. Het blijkt dat onze sample extractors gebruikt kunnen worden ter verbetering van de beste bekende algoritmen voor vele andere problemen, welke bepaald worden door de eigenschap van het hebben van een *beknopte representatie*, zoals het testen of f berekend kan worden door kleine circuits of door kleine beslissingsbomen. We geven ook nieuwe ondergrenzen voor sommige van deze problemen, waarmee we diverse open vragen gesteld door Diakonikolas et al. (FOCS'07) oplossen.

In hoofdstuk 7 onderzoeken we pariteitsfuncties op een andere manier: *com-*

putationele leertheorie. Testen en leren zijn twee nauwverwante gebieden. In plaats van testen of de functie f een pariteit is op een klein aantal variabelen, nemen we bij testen aan dat dat zo is. Dan probeert het algoritme dan $f(x)$ te voorspellen voor andere invoer x met hoge nauwkeurigheid, op basis van een aantal voorbeeldwaarden van f. We werken in het mistake-bound leermodel, welke sterker is dan het (meer gebruikelijke) PAC-model. Het is een welbekend feit dat pariteiten geleerd kunnen worden met foutgrens (of steekproefcomplexiteit) $O(k \log n)$, maar geen implementatie in polynomiale tijd van een dergelijk leeralgoritme is bekend. We ontwerpen een simpel, deterministisch, polynomiale-tijd algoritme om k-pariteiten te leren met foutgrens $O(n^{1-\frac{1}{k}})$. Dit is het eerste polynomiale-tijd algoritme wat $\omega(1)$-partiteiten kan leren met foutgrend $o(n)$, en met standaard conversie-technieken impliceert dit een verbetering van de resultaten van Klivans and Servedio (COLT'04) voor het leren van k-pariteiten in het PAC-model.

Hiernaast beschouwen we ook één van de fundamentele problemen in property testing: monotoniciteit van functies (niet noodzakelijk booleaanse functies). In hoofdstuk 8 onderzoeken we functies gebaseerd op de n-dimensionale hyperkubus en geven we een $\Omega(n)$ ondergrens voor eenzijdige, niet-adaptieve testers van monotoniciteit. Omdat er nog steeds een aanzienlijke kloof zit tussen deze ondergrens en de beste bekende bovengrenzen, kijken we naar een natuurlijke aanpak voor het verkrijgen van bovengrenzen: namelijk het bestuderen van de combinatorische eigenschappen van de hyperkubus. Er was al eerder opgemerkt dat wanneer een verzameling van bron-uitgang paren op de gerichte hyperkubus (waarbij alle bronnen en uitgangen verschillend zijn) verbonden kan worden met rand-disjuncte paden, de monotoniciteit van functies op de n-dimensionale hyperkubus testbaar is met $O(n)$ functie-aanroepen. Bepalen of deze eigenschap geldt, is als open vraag gesteld door Lehman en Ron (J. Comb. Theory, Ser. A, 2001), maar het antwoord was al bijna een decennium ongrijpbaar. Door het analyseren van de combinatorische eigenschappen van de hyperkubus laten we zien dat het antwoord negatief is, en dat deze aanpak altijd tekort schiet om de huidige ondergrenzen te bereiken, of zelfs maar te benaderen.

In het laatste hoofdstuk gaan we in op het probleem van detecteren van cyclussen, geschetst in het begin. We bewijzen dat, misschien contra-intuïtief, wanneer het kleinste herhalende segment van de reeks verschillende elementen bevat, de periode r bepaald kan worden door een aantal elementen van de rij op te vragen, waarbij dat aantal sublogaritmisch in r is; dit is niet ver van optimaal. We bestuderen ook varianten van het probleem waarbij directe toegang tot de reeks niet mogelijk is, maar we in plaats daarvan kunnen "springen" tussen posities die niet te ver van elkaar liggen. Dit is een verbetering op de verwante resultaten van Cleve (CCC'00).

Abstract

How can we solve computational tasks when we simply don't have enough time to process all the data? For example, given a sequence of numbers, can we determine if they start repeating by looking at just a few of them? Or, given query access to some boolean function f, how many queries to f do we need to test whether it is monotone? How about deciding if f is "essentially" the same as another, known, function g?

One of the central goals of theoretical computer science is to understand the limits of computation in a variety of models, and to characterize the resources required to solve certain tasks. The kinds of problems outlined above are particularly amenable to study in the *property testing* model, and it is under this prism that we investigate them. In property testing, algorithms are required to distinguish those objects which satisfy the desired property from those which are far from it. We seek query-efficient randomized testers to solve them (*upper bounds*), along with rigorous proofs explaining why no significantly better solutions are possible, even in principle (*lower bounds*). The results presented here draw from techniques in probability theory, graph theory, extremal combinatorics, the theory of permutation groups, coding theory, the analysis of boolean functions, and number theory.

The starting point is our work on the problem of testing *function isomorphism* in Chapter 2. Two boolean functions on n-bit inputs are isomorphic if they are the same up to permutations of the n input variables. The main case of interest is when g is known and f needs to be queried. It is known that the task can be accomplished with $\widetilde{O}(n)$ queries, but this is exponentially larger than the best lower bounds from prior work. Here we bridge the gap by contributing an almost-optimal adaptive lower bound of $\Omega(n)$ queries for worst-case functions f. Several other variants of the problem are also discussed.

In addition, we show in Chapter 3 that when f is a k-junta (meaning that it is determined by just k of the n input variables), the complexity of testing isomorphism to f is reduced to $\widetilde{O}(k)$. (In contrast, another result of ours is that

if we impose the seemingly weak restriction that the tester have one-sided error, the complexity of testing isomorphism to k-juntas becomes roughly $\log \binom{n}{k}$, which is much larger for small k.) In the process we construct objects of independent interest called *sample extractors*. These are efficient algorithms that allow us to draw samples from the truth table of the "core" function on k variables that determines a given k-junta function.

Next we give a partial characterization of the set of functions against which it is possible to test isomorphism with constant query complexity. We show in Chapter 4 that, for *any* function f with polynomially many different permutations, isomorphism to f is testable with constantly many queries. This theorem extends previous results from junta testing and in fact covers all functions known to date for which isomorphism testing is easy. On a related note, observing the connection between testing function isomorphism and testing *hypergraph isomorphism*, we turn our attention to testing isomorphism for uniform hypergraphs. We give a characterization of the class of hypergraphs of constant rank for which isomorphism can be efficiently tested, generalizing a result of Fischer (STOC'04) regarding graph isomorphism.

In Chapter 5 we establish links with the field of *group testing*, and observe that ideas from isomorphism testing can be applied to the study of a naturally-arising relaxation of group testing problems; both the lower bound methods and the algorithms are of use. We determine the exact query complexity of the relaxed group testing problem for non-adaptive algorithms, up to constant factors. The question of obtaining explicit lower bounds for these problems is also addressed; it turns out that parity functions (XORs of subsets of the input variables) exemplify the worst-case lower bounds for testing function isomorphism, both for one-sided and two-sided testers.

We move on to discuss other property testing problems in Chapter 6. It turns out that our sample extractors can be used to enhance the best algorithms known for many other testing problems, specifically those defined by the property of having *concise representations*, such as testing if f can be computed by small circuits, or decision trees of small size. We also provide new lower bounds for some of them, resolving several open questions posed by Diakonikolas et al. (FOCS'07).

In Chapter 7 we examine parity functions under a different light: *computational learning theory*. Testing and learning are two closely related areas. Instead of testing whether the function f is a parity on a small number of variables, in testing we assume that it is. Then, based on some example values of f, the learner tries to predict $f(x)$ for other inputs x with good accuracy. We work in the mistake-bound model of learning, which is stronger than the (more usual) PAC model. It is a standard fact that parities can be learned with mistake bound (or sample complexity) $O(k \log n)$, but no polynomial-time implementation of such a learning algorithm is known. We design a simple, deterministic, polynomial-time algorithm for learning k-parities with mistake bound $O(n^{1-\frac{1}{k}})$. This is the first polynomial-time algorithm to learn $\omega(1)$-parities with mistake bound $o(n)$, and

using standard conversion techniques it implies an improvement over the results of Klivans and Servedio (COLT'04) for learning k-parities in the PAC model.

Apart from this, we also consider one of the fundamental problems in property testing: monotonicity of functions (not necessarily boolean). In Chapter 8 we consider functions defined on the n-dimensional hypercube, and give an $\Omega(n)$ lower bound for one-sided, non-adaptive testers of monotonicity. As there is still a sizable gap between this and the best upper bounds known, we look at a natural approch to obtaining upper bounds via the study of routing properties of the hypercube. It had been previously observed that if any set of source-sink pairs on the directed hypercube (where all sources and sinks are distinct) could be connected with edge-disjoint paths, then monotonicity of functions on the n-dimensional hypercube would be testable with $O(n)$ queries. Determining whether this property holds was posed as an open problem by Lehman and Ron (J. Comb. Theory, Ser. A, 2001), but the question had remained elusive for nearly a decade. By analyzing the combinatorial properties of the hypercube, we show that the answer is negative, and that this approach must fail short of matching the current lower bounds, or even approaching them.

In the final chapter we address the cycle detection problem. Perhaps counter-intuitively, we prove that if the smallest periodic part of the sequence contains distinct elements, the period length r can be determined by making a number of probes to the sequence that is sublogarithmic in r; moreover, this is not too far from optimal. We also study variants of the problem where random access to the sequence is not allowed, but we can instead "jump" between positions that are not too far away from each other. The latter improves on the related results of Cleve (CCC'00).

Titles in the ILLC Dissertation Series:

ILLC DS-2006-01: **Troy Lee**
Kolmogorov complexity and formula size lower bounds

ILLC DS-2006-02: **Nick Bezhanishvili**
Lattices of intermediate and cylindric modal logics

ILLC DS-2006-03: **Clemens Kupke**
Finitary coalgebraic logics

ILLC DS-2006-04: **Robert Špalek**
Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

ILLC DS-2006-05: **Aline Honingh**
The Origin and Well-Formedness of Tonal Pitch Structures

ILLC DS-2006-06: **Merlijn Sevenster**
Branches of imperfect information: logic, games, and computation

ILLC DS-2006-07: **Marie Nilsenova**
Rises and Falls. Studies in the Semantics and Pragmatics of Intonation

ILLC DS-2006-08: **Darko Sarenac**
Products of Topological Modal Logics

ILLC DS-2007-01: **Rudi Cilibrasi**
Statistical Inference Through Data Compression

ILLC DS-2007-02: **Neta Spiro**
What contributes to the perception of musical phrases in western classical music?

ILLC DS-2007-03: **Darrin Hindsill**
It's a Process and an Event: Perspectives in Event Semantics

ILLC DS-2007-04: **Katrin Schulz**
Minimal Models in Semantics and Pragmatics: Free Choice, Exhaustivity, and Conditionals

ILLC DS-2007-05: **Yoav Seginer**
Learning Syntactic Structure

ILLC DS-2008-01: **Stephanie Wehner**
Cryptography in a Quantum World

ILLC DS-2008-02: **Fenrong Liu**
Changing for the Better: Preference Dynamics and Agent Diversity

ILLC DS-2008-03: **Olivier Roy**
Thinking before Acting: Intentions, Logic, Rational Choice

ILLC DS-2008-04: **Patrick Girard**
Modal Logic for Belief and Preference Change

ILLC DS-2008-05: **Erik Rietveld**
Unreflective Action: A Philosophical Contribution to Integrative Neuroscience

ILLC DS-2008-06: **Falk Unger**
Noise in Quantum and Classical Computation and Non-locality

ILLC DS-2008-07: **Steven de Rooij**
Minimum Description Length Model Selection: Problems and Extensions

ILLC DS-2008-08: **Fabrice Nauze**
Modality in Typological Perspective

ILLC DS-2008-09: **Floris Roelofsen**
Anaphora Resolved

ILLC DS-2008-10: **Marian Counihan**
Looking for logic in all the wrong places: an investigation of language, literacy and logic in reasoning

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic

ILLC DS-2009-05: **Andreas Witzel**
Knowledge and Games: Theory and Implementation

ILLC DS-2009-06: **Chantal Bax**
Subjectivity after Wittgenstein. Wittgenstein's embodied and embedded subject and the debate about the death of man.

ILLC DS-2009-07: **Kata Balogh**
Theme with Variations. A Context-based Analysis of Focus

ILLC DS-2009-08: **Tomohiro Hoshi**
Epistemic Dynamics and Protocol Information

ILLC DS-2009-09: **Olivia Ladinig**
Temporal expectations and their violations

ILLC DS-2009-10: **Tikitu de Jager**
"Now that you mention it, I wonder...": Awareness, Attention, Assumption

ILLC DS-2009-11: **Michael Franke**
Signal to Act: Game Theory in Pragmatics

ILLC DS-2009-12: **Joel Uckelman**
More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains

ILLC DS-2009-13: **Stefan Bold**
Cardinals as Ultrapowers. A Canonical Measure Analysis under the Axiom of Determinacy.

ILLC DS-2010-01: **Reut Tsarfaty**
Relational-Realizational Parsing

ILLC DS-2010-02: **Jonathan Zvesper**
Playing with Information

ILLC DS-2010-03: **Cédric Dégremont**
The Temporal Mind. Observations on the logic of belief change in interactive systems

ILLC DS-2010-04: **Daisuke Ikegami**
Games in Set Theory and Logic

ILLC DS-2010-05: **Jarmo Kontinen**
Coherence and Complexity in Fragments of Dependence Logic

ILLC DS-2010-06: **Yanjing Wang**
Epistemic Modelling and Protocol Dynamics

ILLC DS-2010-07: **Marc Staudacher**
Use theories of meaning between conventions and social norms

ILLC DS-2010-08: **Amélie Gheerbrant**
Fixed-Point Logics on Trees

ILLC DS-2010-09: **Gaëlle Fontaine**
Modal Fixpoint Logic: Some Model Theoretic Questions

ILLC DS-2010-10: **Jacob Vosmaer**
Logic, Algebra and Topology. Investigations into canonical extensions, duality theory and point-free topology.

ILLC DS-2010-11: **Nina Gierasimczuk**
Knowing One's Limits. Logical Analysis of Inductive Inference

ILLC DS-2011-01: **Wouter M. Koolen**
Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice

ILLC DS-2011-02: **Fernando Raymundo Velazquez-Quesada**
Small steps in dynamics of information

ILLC DS-2011-03: **Marijn Koolen**
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

ILLC DS-2011-04: **Junte Zhang**
System Evaluation of Archival Description and Access

ILLC DS-2011-05: **Lauri Keskinen**
Characterizing All Models in Infinite Cardinalities

ILLC DS-2011-06: **Rianne Kaptein**
Effective Focused Retrieval by Exploiting Query Context and Document Structure

ILLC DS-2011-07: **Jop Briët**
Grothendieck Inequalities, Nonlocal Games and Optimization

ILLC DS-2011-08: **Stefan Minica**
Dynamic Logic of Questions

ILLC DS-2011-09: **Raul Andres Leal**
Modalities Through the Looking Glass: A study on coalgebraic modal logic and their applications

ILLC DS-2011-10: **Lena Kurzen**
Complexity in Interaction

ILLC DS-2011-11: **Gideon Borensztajn**
The neural basis of structure in language

ILLC DS-2012-01: **Federico Sangati**
Decomposing and Regenerating Syntactic Trees

ILLC DS-2012-02: **Markos Mylonakis**
Learning the Latent Structure of Translation

ILLC DS-2012-03: **Edgar José Andrade Lotero**
Models of Language: Towards a practice-based account of information in natural language

ILLC DS-2012-04: **Yurii Khomskii**
Regularity Properties and Definability in the Real Number Continuum: idealized forcing, polarized partitions, Hausdorff gaps and mad families in the projective hierarchy.

ILLC DS-2012-05: **David García Soriano**
Query-Efficient Computation in Property Testing and Learning Theory