On Tolerant Testing and Tolerant Junta Testing

by

Amit Levi

A thesis
presented to the University Of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Author's Declaration

Statement of contributions

This thesis describes results from the following papers:

1. [BCE$^+$19] *Tolerant junta testing and the connection to submodular optimization and function isomorphism.* Joint work with Eric Blais, Clément Canonne, Talya Eden and Dana Ron. ACM Transactions on Computation Theory (TOCT) 11 (4), 2019 (Preliminary version at SODA 2018).

2. [LW19] *Lower bounds for tolerant junta and unateness testing via rejection sampling of graphs.* Joint work with Erik Waingarten. Presented at ITCS 2019.

3. [BFLR20] *Hard properties with (very) short* PCPP*s and their applications.* Joint work with Omri Ben-Eliezer, Eldar Fischer and Ron D. Rothblum. Presented at ITCS 2020.

Abstract

Over the past few decades property testing has became an active field of study in theoretical computer science. The algorithmic task is to determine, given access to an unknown large object (e.g., function, graph, probability distribution), whether it has some fixed property, or it is far from any object having the property. The approximate nature of these algorithms allows in many cases to achieve a significant saving in running time, and obtain *sublinear* running time. Nevertheless, in various settings and applications, accepting only inputs that exactly have a certain property is too restrictive, and it is more beneficial to distinguish between inputs that are close to having the property, and those that are far from it. The framework of *tolerant* testing tackles this exact problem. In this thesis, we will focus on one of the most fundamental properties of Boolean functions: the property of being a *k-junta* (i.e., being dependent on at most k variables).

The first chapter focuses on algorithms for tolerant junta testing. In particular, we show that there exists a poly(k) query algorithm distinguishing functions close to k-juntas and functions that are far from $2k$-juntas. We also show how to obtain a trade-off between the "tolerance" of the algorithm and its query complexity.

The second chapter focuses on establishing a query lower bound for tolerant junta testing. In particular, we show that any non-adaptive tolerant junta tester, is required to make at least $\Omega(k^2/\text{polylog } k)$ queries.

The third chapter considers tolerant testing in a more general context, and asks whether tolerant testing is strictly harder than standard testing. In particular, we show that for any constant $\ell \in \mathbb{N}$, there exists a property $\mathcal{P}_\ell$ such that $\mathcal{P}_\ell$ can be tested in $O(1)$ queries, but any tolerant tester for $\mathcal{P}_\ell$ is required to make at least $\Omega(n/\log^{(\ell)} n)$ queries (where $\log^{(\ell)}$ denote the ℓ times iterated log function).

The final chapter focuses on applications. We show how to leverage the techniques developed in previous chapters to obtain results on tolerant isomorphism testing, unateness testing, and erasure resilient testing.

Acknowledgments

It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no knowing where you might be swept off to

J.R.R Tolkien

This marks the end of an adventure. One started long time ago across the oceans. This adventure could not have been completed without the help and support of many.

First and foremost, I want to thank my advisor Eric Blais for his encouragement and support during my years as a PhD student, and in particular, for allowing me to choose my own path in research. Whenever I knocked at his door, I was always greeted with a smile (and sometimes a fine cup of coffee). He never made me feel stupid (and I was at times), always offered good advice and an optimistic point of view.

Throughout my years in grad-school I had the honor of collaborating with some great mentors. Specifically, I want to thank Xi Chen, Eldar Fischer, Sofya Raskhodnikova, Dana Ron, and Yuichi Yoshida, whose immense knowledge and advice helped shape my research work. I offer a special thanks to Eldar Fischer, for multiple visit invitations to the Technion, for teaching me so much about math, and for being a great support in times of need.

I am also grateful to all my coauthors and collaborators: Michael Abebe, Omri Ben-Eliezer, Clément Canonne, Khuzaima Daudjee, Talya Eden, Brad Glasbergen, Rajesh Jayaram, Gautam Kamath, Ramesh Krishnan Pallavoor, Ron D. Rothblum, Nithin Varma and Erik Waingarten. It was truly a wonderful experience to work with each and every one of them. A special thanks goes to Erik Waingarten, my "academic bro", for his hospitality (either in NYC or Berkeley), numerous discussions about math and for being a great friend!

This thesis could not be completed without the support of my friends in Canada. I want to thank my fellow students at the department, Vedat Levi Alev, Abhinav Bommireddi, Nathan Harms, Anil Pacaci, Akshay Ramachandran and Hong Zhou, for many enjoyable conversations throughout the years. Special thanks for Victoria Sakhnini, for wonderful friday dinners, and for making me feel at home, away from home. To all of my friends in Waterloo, thank you for making my grad school experience so enjoyable.

Lastly, to Mom, Dad, Shenhav, Savti, Sabi and the rest of my family, thanks for your never ending support and encouragement, this journey could not have been completed without you.

New adventures are right around the corner...

Table Of Contents

List of Figures

Chapter 1

Introduction

The field of *property testing* is concerned with the analysis of the global structure of data. Property testing algorithms are algorithms that perform a type of approximate decision. As opposed to standard decision algorithms, which are required to determine whether a given input has some specified property or not, property testing algorithms are required to determine whether a given input has the property, or it is far from having the property. Informally, we say that an input is far from a property if it needs to be significantly modified in order to obtain the property (in particular, we will consider the Hamming distance). In the typical setting, property testing algorithms are given a *distance* parameter ε, should reject inputs that that are ε-far from having the property, and accept inputs having the property. However, if a given input is close to having the property, the algorithm can either accept or reject .[1]

The approximate nature of these algorithms allows us, in many cases, to go below the gold-standard linear time, and achieve a *sublinear* running time in the size of the input, and in some cases the running time is even *independent* of the input size. The fact that property testing algorithms may run in sublinear time suggests that the algorithm cannot even read the entire input, and instead is given *query access* to the input. Property testing algorithms provide a *tradeoff between accuracy and efficiency.* This tradeoff is beneficial in numerous scenarios such as:

1. When the exact decision problem is NP-hard, property testing algorithms provide a type of approximation which is necessary in such cases.
2. When reading the entire input is infeasible (for example, when the object is enormous).
3. When there is a need to quickly rule out far objects, and spend our resources only on objects that are close to having the property.

This tradeoff also raises the following high level question:

> *Is it possible to leverage the structure of the property in order to achieve testers with sublinear run time?*

In general, property testing algorithms come in two flavors, *non-adaptive* and *adaptive.* Non-adaptive algorithms need to fix their queries in advance and are not allowed to depend on answers to previous queries. Indeed, the fact that the algorithm is allowed to pick its queries depending on its previous ones, generally provides more power to the algorithm (see e.g., [GR11, STW15, CSTW, CG17]). We say that a testing algorithm has a *one-sided* error if it accepts every object that satisfies the property with probability 1, and otherwise, it has a

[1]For precise definitions, see Section 2.1

two-sided error. Similarly, allowing the algorithm to have a two-sided error generally decreases its query complexity (see e.g.,[Lev15]).

1.1 Testing juntas

A significant portion of this thesis will focus on one of the most fundamental properties of Boolean functions– the property of being a *junta*. A function $f\colon \{0,1\}^n \to \{0,1\}$ is a *k-junta* if it depends on at most k of its variables. Juntas are a central object of study in the analysis of Boolean functions, in particular since they can be used as a good approximation for many classes of (more complex) Boolean functions.

As an example, consider the class of s-term DNFs. That is, the class of all Boolean functions of the form $T_1 \vee \ldots \vee T_s$, where each T_i is of the form $\bigwedge_{j\in S} x_j$ for some $S \subseteq [n]$. It might be surprising at first, but this very expressive class can be well approximated by juntas of size independent of the dimension. Indeed, it is not hard to show that any s-term DNF is ε-close (in Hamming distance) to a $s\log(s/\varepsilon)$-junta, since any term with more than $\log(s/\varepsilon)$ variables can be removed contributing an error of at most ε/s. More generally, Friedgut [Fri98] showed that any boolean function f is close to a junta whose size depend on some “simplicity parameter” of f (the average sensitivity of f).

In the context of learning, the study of juntas was introduced by Blum et al. [Blu94, BL97] to model the problem of learning in the presence of irrelevant attributes. Blum asked whether there exists a poly(n) time algorithm that learns $O(\log n)$-juntas using samples from the uniform distribution. The current state of the art algorithm for the above problem is due to Mossel O'Donnell and Servedio. [MOS03] and has a running time of n^{ck} for some $c < 1$. Nevertheless, the question whether there exists a polynomial time algorithm remains open. Since then, juntas have been extensively studied both in computational learning theory (e.g., [MOS03, Val15]) and in applied machine learning (e.g., [JL10]).

The problem of deciding whether a given function $f : \{0,1\}^n \to \{0,1\}$ is a k-junta requires exponential running time. Indeed, this task may require going through all 2^n input strings, which is unfeasible especially when the dimension n is large. However, one might want to consider a possible relaxation of the above task to the: Is it possible to efficiently distinguish functions which are k-juntas from functions that are *far* from any k-junta? Namely, we consider property testing of juntas.

The problem of testing k-juntas was first addressed by Fischer et. [FKR$^+$04]. They designed an algorithm that queries the function on a number of inputs polynomial in k, and *independent of n*. A series of subsequent works essentially settled the optimal query complexity for this problem, establishing that $\Theta(k/\varepsilon \cdot \log(k/\varepsilon))$ queries are both necessary and sufficient [Bla08, Bla09, CG04, STW15, Sağ18].

Nevertheless, the setting of property testing is arguably fragile, since the testing algorithm is only guaranteed to accept all functions that exactly satisfy the property. In various settings and applications, accepting only inputs that exactly have a certain property is too restrictive, and it is more beneficial to distinguish between inputs that are close to having the property, and those that are far from it. To address this question, Parnas, Ron and Rubinfeld introduced in [PRR06] a natural generalization of property testing, where the algorithm is required to be *tolerant*.

1.2 Tolerant testing

A *tolerant property testing algorithm* is required to accept any function that is *close* to having the property, and as in the standard model, to reject any function that is far from the property.

Ideally, a tolerant testing algorithm should work for any given tolerance parameter $\varepsilon' < \varepsilon$ (that is, accept functions that are ε'-close to having the property), and have complexity that depends on $\varepsilon - \varepsilon'$. However, in some cases the relation between ε' and ε may be more restricted (e.g., $\varepsilon' = \varepsilon/c$ for a constant c). A closely related notion considered in [PRR06] is that of *distance approximation* where the goal is to obtain an estimate of the distance that the object has to a property.

As observed in [PRR06], any standard testing algorithm whose queries are uniformly (but not necessarily independently) distributed is inherently tolerant to some extent. More precisely, any q-query testing algorithm will accept objects which are $O(\varepsilon/q)$-close to the property. However, for many problems, strengthening the tolerance requires applying different methods and devising new algorithms (see e.g., [GR05, PRR06, FN07, ACCL07, KS09, MR09, FR10, CGR13, BMR16]).

In context of juntas, the problem of tolerant testing was previously considered by Diakonikolas et al. [DLM+07]. They applied the aforementioned observation from [PRR06] and showed that one of the junta testers from [FKR+04] actually accepts functions that are $\mathrm{poly}(\varepsilon, 1/k)$-close to k-juntas. Chakraborty et al. [CFGM12] observed that the analysis of the (standard) junta tester of Blais [Bla09] implicitly implies an $\exp(k/\varepsilon)$-query complexity tolerant tester which accepts functions that are ε/C-close to some k-junta (for some constant $C > 1$) and rejects functions that are ε-far from every k-junta.

It is natural to ask whether *in general* tolerant testing is strictly harder than standard testing. This question was explicitly studied by Fischer and Fortnow [FF06]. Using machinery common in the field of *probabilistically checkable proofs* (PCP), they designed a property $\mathcal{P} \subseteq \{0,1\}^n$ that admits a tester with constant query complexity, but such that every tolerant tester for $\mathcal{P}$ has query complexity $\Omega(n^c)$ for some $c \in (0,1)$. Using modern PCP tools [BSS08, Din07] combined with the techniques of [FF06] it is possible to construct a property demonstrating a better separation, of constant query complexity for standard testing versus $\Omega(n/\mathrm{polylog}\, n)$ for tolerant testing.

1.3 Our contributions

In this thesis, we present progress on the question of tolerant junta testing, and on tolerant testing in a more general context.

Chapter 3: Algorithms for tolerant junta testing

As observed in Section 1.2, it is not hard to obtain an algorithm that distinguishes functions which are ε/C-close to k-juntas and functions that are ε-far from k-juntas (for some constant $C > 1$), is relatively easy using $\exp(k)/\varepsilon$ queries (we provide a self contained proof in Section 3.1). Our goal in this chapter is to understand whether such a dependence can be avoided.

First, we wish to understand how the amount of tolerance of the algorithm effects its query complexity. The observation of [DLM+07] that it is possible to distinguish functions which are $\mathrm{poly}(\varepsilon, 1/k)$-close to k-juntas from those that are ε-far from k-juntas using $\mathrm{poly}(k, 1/\varepsilon)$ queries hints towards a *trade off* in terms of tolerance and query complexity. However, what about other settings of the closeness parameter? Is it possible to obtain an algorithm that interpolates the above two results in terms of the distance gap?

We answer this question affirmatively, and in Section 3.2 we show how to obtain a *smooth tradeoff* between the amount of tolerance and the query complexity of the algorithm.

Theorem 1.3.1. *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$, $\varepsilon \in (0,1)$ and $\rho \in (0,1)$, satisfies the following.*

- *If f is $\rho\varepsilon/16$-close to some k-junta, then the algorithm accepts with high constant probability.*
- *If f is ε-far from every k-junta, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $O\left(\frac{k\log k}{\varepsilon\rho(1-\rho)^k}\right)$.

Note that setting $\rho = O(1/k)$ results in a $\text{poly}(k, 1/\varepsilon)$ query algorithm and setting $\rho = O(1)$ results in a $\exp(k)/\varepsilon$ query algorithm, exactly matching the two extreme cases mentioned above.

Our second angle considers the question of whether a moderate relaxation of the tolerant testing model exhibits significant saving in query complexity. In Section 3.3, we show that if we only require our algorithm to reject functions that are ε-far from $2k$ juntas (while still accepting any function ε/C-close to a k-junta for some $C > 1$), one can achieve a tester with $\text{poly}(k, 1/\varepsilon)$ query complexity.

Theorem 1.3.2. *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$ and $\varepsilon \in (0,1)$, satisfies the following.*

- *If f is $\varepsilon/10$-close to some k-junta, then the algorithm accepts with high constant probability.*
- *If f is ε-far from every $2k$-junta, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $\text{poly}(k, \frac{1}{\varepsilon})$.

We note that similar relaxations have been considered both in the standard testing model (e.g., [PR02, KR98, KNOW14]) and in the tolerant testing model [PRR06]. Although one may hope for a stronger statement (distinguishing functions close to k-junta from those far from k-junta), for most practical purposes the relaxation above is more than sufficient. Among others, this is supported by the following two examples: (i) In the setting we are concerned about, k is to be thought of as very small (constant, or very slowly growing function of n), while n is dauntingly large. Thus, reducing the dimension of the problem from n features to $2k$ is already enough to break the curse of dimensionality, (ii) For *learning* algorithms, it is common to try and identify the smallest class of functions the unknown target belongs to, in order to apply an efficient learning algorithm tailored to it. In order to do so, one often uses a doubling search (or variant thereof) over the set of parameters (here, k) to identify the "best candidate parameter" while not spending too many resources in this preliminary phase. This implies that, inherently, any such search approach already loses a constant factor in this relevant parameter.

Chapter 4: Hardness results for tolerant junta testing

As noted in [PRR06], tolerant testing is not only a natural generalization of property testing, but is also very often the desirable attribute of testing algorithms. This motivates the high level question: how does the requirement of being tolerant affect the complexity of testing the properties studied? From previous works, it was conceivable that the problem of tolerant junta testing is not harder than standard junta testing. Our main result in this chapter shows that non-adaptive tolerant testing of juntas is significantly harder than standard non-adaptive junta testing. In particular, we prove the following theorem.

Theorem 1.3.3. *For any $\alpha < 1$, there exist constants $0 < \varepsilon_0 < \varepsilon_1 < 1$ such that for any $k = k(n) \leq \alpha n$, any non-adaptive algorithm that distinguish between functions $f : \{-1,1\}^n \to \{-1,1\}$ that are ε_0-close to k-junta and functions ε_1-far from any k-junta must make $\Omega(k^2/\text{polylog } k)$ queries.*

For the setting of a non-adaptive algorithms, Blais [Bla08] gave $O(k^{3/2} \cdot \operatorname{poly}\log k)$-query tester for (non-tolerant) testing of k-juntas, which was shown to be optimal for non-adaptive algorithms by Chen, Servedio, Tan, Waingarten and Xie in [CST$^+$17]. Combined with Theorem 1.3.3, this shows a polynomial separation in the query complexity of non-adaptive tolerant junta testing and non-adaptive junta testing.

Chapter 5: General separation between tolerant and intolerant testing

This chapter deals with the high level question described in the previous chapter. Namely, how hard can tolerant testing be compared to standard testing? As mentioned previously, Fischer and Fortnow [FF06] constructed a property of n-bits boolean strings which is testable with a number of queries independent of n, but require at least $n/\log n$ many queries to tolerantly test.

In this chapter, we provide an even stronger separation. We construct a family of properties, such that each property in the family is testable using constant (independent of n) queries, but require much more than $n/\log n$ in order to tolerantly test. Specifically, we show the following.

Theorem 1.3.4 (informal)**.** *For any constant integer $\ell \in \mathbb{N}$, there exists a property $\mathcal{P}_\ell$ such that $\mathcal{P}_\ell$ is testable with query complexity independent of n, but require $\Omega(n/\log^{(\ell)} n)$ to tolerantly test (where $\log^{(\ell)}$ denote the iterated* log *function).*

Chapter 6: Applications

Finally, we show how the techniques developed in previous chapters apply to other problems.

Tolerant isomorphism testing: In Section 6.1 we show how Theorem 1.3.1 can be applied to the problem of *isomorphism testing*, which we recall next. Given query access to two unknown Boolean functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$ and a parameter $\varepsilon \in (0,1)$, one has to distinguish between (i) f is equal to g up to some relabeling of the input variables; and (ii) $\operatorname{dist}(f, g\circ\pi) > \varepsilon$ for every such relabeling π (where $g \circ \pi$ denote the natural function composition). The worst-case complexity of this task is known, with $\Theta\big(2^{\frac{n}{2}}/\sqrt{\varepsilon}\big)$ queries being necessary (up to the exact dependence on ε) and sufficient [AB10, ABC$^+$13]. However, is the exponential dependence on n always necessary, or can we obtain better results for "simple" functions? Ideally, we would like our testers to improve on this worst-case behavior, and instead have an *instance-adaptive* query complexity, depending only on some intrinsic parameter of the functions f, g to be tested. This is the direction we pursue here. Let $k^* = k^*(f, g, \gamma)$ be the smallest k such that either f or g is γ-close to being a k-junta. We show that it is possible to achieve a query complexity only depending on this (unknown) parameter, namely of the form $O\big(2^{k^*(f,g,O(\varepsilon))/2}/\varepsilon\big)$.[2] Moreover, our algorithm offers a much stronger guarantee: it allows *tolerant* isomorphism testing.

Theorem 1.3.5 (Tolerant isomorphism testing)**.** *There exists an algorithm that, given query access to two functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$ and parameter $\varepsilon \in (0,1)$, satisfies the following, for some absolute constant $C \geq 1$.*

- *If f and g are $\frac{\varepsilon}{C}$-close to isomorphic, then the algorithm accepts with high constant probability.*

[2]It is worth noting that this parameter can be much lower than the actual number of relevant variables for either functions; for instance, there exist functions depending on *all* n variables, yet that are $o(1)$-close to $O(1)$-juntas.

- *If f and g are ε-far from isomorphic, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $O\big(2^{\frac{k^*}{2}}/\varepsilon\big)$ with high-probability (and $O\big(2^{\frac{n}{2}}/\varepsilon\ \big)$ in the worst case), where $k^* = k^*(f, g, \frac{\varepsilon}{C})$.

The above statement is rather technical, and requires careful parsing. In particular, the parameter k^* is crucially *not* provided as input to the algorithm: instead, it is discovered adaptively by invoking the tolerant tester of Theorem 1.3.1. This explains the high-probability bound on the query complexity: with some small probability, the algorithm may fail to retrieve the right value of k^* – in which case it may use instead a larger value, possibly up to n.

Lower bounds for tolerant unateness testing: In Section 6.2, we apply our techniques from Chapter 4 to the problem of *unateness* testing. A function $f\colon \{0,1\}^n \to \{0,1\}$ is *unate* if f is either non-increasing or non-decreasing in every variable. Namely, there exists a string $r \in \{0,1\}^n$ such that the function $f(x \oplus r)$ is monotone with respect to the bit-wise partial order on $\{0,1\}^n$.

The next two theorems concern tolerant testers for unateness.

Theorem 1.3.6. *There exist constants $0 < \varepsilon_0 < \varepsilon_1 < 1$ such that any (possibly adaptive) algorithm that distinguishes between functions ε_0-close to unate and functions ε_1-far from unate, must make $\Omega(n/\text{polylog } n)$ queries.*

Theorem 1.3.7. *There exist constant $0 < \varepsilon_0 < \varepsilon_1 < 1$ such that any non-adaptive algorithm that distinguishes between functions ε_0-close to unate and functions ε_1-far from unate, must make $\Omega(n^{3/2}/\text{polylog } n)$ queries.*

A similar separation in tolerant and non-tolerant testing occurs for the property of unateness as a consequence of Theorem 1.3.6 and Theorem 1.3.7. Recently, in [BCP$^+$17], Baleshzar, Chakrabarty, Pallavoor, Raskhodnikova, and Seshadhri gave a non-adaptive $\widetilde{O}(n)$-query tester for (non-tolerant) unateness testing, and Chen, Waingarten and Xie [CWX17a] gave an (adaptive) $\widetilde{O}(n^{3/4})$-query tester for (non-tolerant) unateness testing. We thus, conclude that by Theorem 1.3.6 and Theorem 1.3.7, tolerant unateness testing is polynomially harder than (non-tolerant) unateness testing, in both adaptive and non-adaptive settings.

Separating erasure resilient testing from property testing: In Section 6.3, we show how the tools developed in Chapter 5 allow us to separate the standard testing model from the *erasure resilient* model. This model was defined by Dixit et. al. [DRTV18] to address cases where data cannot be accessed at some domain points due to privacy concerns, or when some of the values were adversarially erased. More precisely, an α-erasure-resilient ε-tester gets as input parameters $\alpha, \varepsilon \in (0,1)$, as well as oracle access to a function f, such that at most an α fraction of its values have been erased. The tester has to accept with high probability if there is a way to assign values to the erased points of f such that the resulting function satisfies the desired property. The tester has to reject with high probability if for every assignment of values to the erased points, the resulting function is still ε-far from the desired property.

Similarly to the tolerant testing scenario, [DRTV18] show that there exists a property of Boolean strings of length n that has a tester with query complexity independent of n, but for any constant $\alpha > 0$ every α-erasure-resilient tester is required to query at least $\Omega(n^c)$ bits for some $c > 0$, thereby establishing a separation between the models. We give a stronger separation between the erasure resilient model and the standard testing model.

Theorem 1.3.8 (informal restatement of Theorem 6.3.2). *For any constant integer $\ell \in \mathbb{N}$, there exist a property of boolean strings $\mathcal{P} \subseteq \{0,1\}^n$ and a constant $\varepsilon_1 \in (0,1)$ such that $\mathcal{P}$ is ε-testable for any $\varepsilon > 0$ with number of queries independent of n, but for any $\alpha = \Omega(1/\log^{(\ell)} n)$ and $\varepsilon \in (0, \varepsilon_1)$ such that $\varepsilon < 1 - \alpha$, any α-erasure-resilient ε-tester is required to query $\Omega(n/\mathrm{polylog}^{(\ell)} n)$ many bits.*

Remark. Some of the results presented in this thesis have recently been extended or improved. In a followup work, De et al. [DMN19] show a tolerant junta testing algorithm that for any $0 \leq \varepsilon_0 < \varepsilon_1 < 1/2$, distinguishes between functions ε_0-close to k-junta and functions ε_1-far from any k-junta, with query complexity $2^k \cdot \mathrm{poly}(k, 1/(\varepsilon_1 - \varepsilon_0))$, providing a wider range of parameters. However, for some settings of ε_0 and ε_1 (e.g., $\varepsilon_0 = \varepsilon/\log k$ and $\varepsilon_1 = \varepsilon$) our algorithm from Theorem 1.3.1 provides a better query complexity than $O(2^k)$. Our lower bound presented in Theorem 1.3.3 has been improved recently by Pallavoor et al. [PRW19] to a super-polynomial in k and $1/\varepsilon$.

1.4 Related work

1.4.1 Property Testing of functions

Property testing was first explicitly defined in the work of Rubinfeld and Sudan [RS96], who considered testing whether a function is a low-degree (univariate) polynomial over a fixed finite field $\mathbb{F}$. The first time a question was phrased in terms of property testing was considered in the work of Blum, Luby and Rubinfeld [BLR90], where the authors consider the task of testing whether a function $f : \mathbb{F}^n \to \mathbb{F}$ is *linear*[3] or ε-far (in Hamming distance) from any linear function. The focus of the above was on testing algebraic properties of functions, which had an important role in the design of *Probabilistically Checkable Proofs (*PCP*)* systems [ALM+98, AS98, BFL91, BFLS91, GLR+91, FGL+91, FS95]. Since then, algebraic properties of functions continued to be a central object in the study of property testing, partly because of their connection to the area of *error correcting codes.* If we view functions as codeword, than the task is to distinguish between codewords and words that are far from any codeword. Examples of such codes which admit a sublinear time testers include Hadamard codes, Reed-Solomon codes and Reed-Muller codes (see e.g., [AKK+05, KR06, JPRZ09]).

In general, property testing can be seen as a relaxation of learning. Instead of asking that the algorithm output a good estimate of the function, which is assumed to belong to a particular class of functions $\mathcal{F}$, we require that the algorithm decide whether the function belongs to $\mathcal{F}$ or is far from any function in $\mathcal{F}$. With this view in mind, a natural motivation for property testing is to serve as a preliminary step before learning (and in particular, agnostic learning (e.g., [KSS94] where no assumption is made about the target function aside from it being a member of $\mathcal{F}$). We can first run the testing algorithm to decide whether to use a particular class of functions as our hypothesis class.

Additional properties of boolean functions of interest include monotonicity and unateness [GGLR98, DGL+99, FLN+02, EKK+00, BCGSM12, CC16, CS13, CS14, CWX17a, CWX17b], half-spaces [MORS10] and sparsity [GOS+11].

1.4.2 Property testing of graphs

The study of property testing in a more combinatorial context was initiated by Goldreich, Goldwasser and Ron [GGR98]. They gave several general results, among them results concerning testing properties of graphs. In this model the algorithm may perform queries of the form: "Is

[3]A function $f : \mathbb{F}^n \to \mathbb{F}$ is linear if and only if for every $x, y \in \mathbb{F}^n$, $f(x) + f(y) = f(x + y)$.

there an edge between vertices u and v in the graph?" That is, the algorithm may probe the adjacency matrix representing the tested graph $G = (V, E)$, which is equivalent to querying the function $f_G : V \times V \to \{0, 1\}$, where $f_G(u, v) = 1$ if and only if $(u, v) \in E$. We refer to such queries as *vertex-pair* queries. In the dense-graphs model, a graph is considered ε-far from having property $\mathcal{P}$ if more than εn^2 edges modifications should be performed on the graph so that it obtains the property.

In their seminal result, [GGR98] showed various testing algorithms for "partition problems" such as bipartiteness, k-Colorability and having a ρ-clique. All of which with query complexity of poly$(1/\varepsilon)$.

Further research shows that testable graph properties[4] in this model can be precisely characterized. A sequence of works [AFKS00, FN07, AS08] led to the work of Alon et al. [AFNS09] which gave a full characterization for graph properties that are testable in a number of queries independent of n, using variations of the Szemerédi regularity lemma. A different characterization, based on graph limits, was proved independently by Borgs et al. [BCL$^+$06]. Recently, it was shown that this result holds for a more general setting of *ordered graphs.*[5] In [ABEF17], the authors show that any hereditary property of ordered graphs is testable with a number of queries independent of the size of the graph. Similarly to the case of unordered graphs, this result has recently been obtained using a generalization of graph limits theory for ordered graphs [BEFLY18].

Other results in property testing in the dense graphs model include results for directed graphs (e.g. [BR02]) and extensions to hypergraphs (e.g. [KNR02]).

However, when dealing with sparse graphs, the above definition of distance makes little sense. To handle this case [GR02] introduced the *bounded-degree graph model.* In this model, the algorithm may perform queries of the form: "who is the i-th neighbor of a vertex v in the graph?" That is, the algorithm may probe the incidence lists of the vertices in the graph, where it is assumed that all vertices have degree at most d for some fixed degree-bound d. This is equivalent to querying the function $f_G : V \times [d] \to V \cup \perp$ that is defined as follows: For each $v \in V$ and $i \in [d]$, if the degree of v is at least i then $f_G(v, i)$ is the i-th neighbor of v (according to some arbitrary but fixed ordering of the neighbors), and if v has degree smaller than i, then $f_G(v, i) = \perp$.

In the bounded degree graph model, a graph is said to be ε-far from having property $\mathcal{P}$ if more than εdn edge modifications should be performed on the graph so that it obtains the property. In this case ε measures the fraction of entries in the incidence lists representation (which has size dn) that should be modified.

In [GR02], Goldreich and Ron showed various testing algorithms for properties such as connectivity, k-edge connectivity, cycle-freeness and being Eulerian. All of which have query complexity of poly$(1/\varepsilon)$. In this paper, Goldreich and Ron formulated the question of testing the expansion property, which resulted in a series of works [GR11, KS08, CS10, NS10]. In terms of characterization, a sequence of works [CSS09, BSS10, HKNO09, LR15] establish that every minor closed property is two-sided testable in a constant time. Recently, it was shown that for one-sided algorithms, the same task requires $\Theta(\sqrt{n})$ queries [CGR$^+$14, KSS18].

It is important to note that the number of queries required to test a property in the bounded-degree model may differ significantly from that required in the dense-graph model. For example, testing bipartiteness, that can be performed in the dense-graphs model with a number of queries independent of n ([GGR98]), but requires $\Omega(\sqrt{n})$ queries in the bounded-degree model ([GR02]).

[4]i.e., using number of queries independent of the size of the graph

[5]an ordered graph is a graph with a total order on its vertex set.

Chapter 2

Preliminaries and Tools

We start with some notations. We use boldfaced letters such as $\boldsymbol{a}, \boldsymbol{m}$ to denote random variables. Let $[n]$ the set of integers $\{1, \ldots, n\}$. Given a string $x \in \{0,1\}^n$ and $j \in [n]$, we write $x^{(j)}$ to denote the string obtained from x by flipping the j-th coordinate. An *edge* along the j-th direction in $\{0,1\}^n$ is a pair (x, y) of strings with $y = x^{(j)}$. In addition, for $\alpha \in \{0,1\}$ we use the notation $x^{(j \to \alpha)}$ to denote the string obtained from x by setting the jth coordinate to α. Given $x \in \{0,1\}^n$ and $S \subseteq [n]$, we use $x|_S \in \{0,1\}^S$ to denote the projection of x on S. For a distribution $\mathcal{D}$ we write $\boldsymbol{d} \sim \mathcal{D}$ to denote an element d drawn according to the distribution. In addition, given an event $\boldsymbol{\mathcal{E}}$ over some probability space, we use $\mathbb{1}_{\{\boldsymbol{\mathcal{E}}\}}$ to denote the indicator function for $\boldsymbol{\mathcal{E}}$.

We sometimes write $a \approx b \pm c$ to denote $b - c \leq a \leq b + c$. For a set A, we let 2^A denote the power-set of A. For two strings $x, y \in \{0,1\}^*$ we use $x \sqcup y$ to denote string concatenation.

For an integer k, a field $\mathbb{F} = \mathrm{GF}(2^k)$ and $\alpha \in \mathbb{F}$, we let $\langle\!\langle \alpha \rangle\!\rangle \in \{0,1\}^k$ denote the binary representation of α in some canonical way. For two sets of strings A and B we use $A \sqcup B$ to denote the set $\{a \sqcup b \mid a \in A,\ b \in B\}$. For a collection of sets $\{A(d)\}_{d \in D}$ we use $\bigsqcup_{d \in D} A(d)$ to denote the set of all possible concatenations $\bigsqcup_{d \in D} a_d$, where $a_d \in A(d)$ for every $d \in D$.

2.1 Property testing and Juntas

Definition 2.1.1 (Property and Distance)**.** A *property* $\mathcal{P}$ of Boolean functions is a subset of all these functions, and we say that a function f *has the property* $\mathcal{P}$ if $f \in \mathcal{P}$. The distance between two functions $f, g \colon \{-1,1\}^n \to \{-1,1\}$ is defined as their (normalized) Hamming distance $\operatorname{dist}(f, g) \stackrel{\text{def}}{=} \mathbf{Pr}_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$, where $\boldsymbol{x}$ is drawn uniformly at random.

Accordingly, for a function f and a property $\mathcal{P}$ we define the distance from f to $\mathcal{P}$ as $\operatorname{dist}(f, \mathcal{P}) \stackrel{\text{def}}{=} \min_{g \in \mathcal{P}} \operatorname{dist}(f, g)$. Given $\varepsilon \geq 0$ and a property $\mathcal{P}$, we will say that a function f is *ε-far* from $\mathcal{P}$ (resp. *ε-close* to $\mathcal{P}$) if $\operatorname{dist}(f, \mathcal{P}) > \varepsilon$ (resp. $\operatorname{dist}(f, \mathcal{P}) \leq \varepsilon$).

We consider the following definition of tolerant testing of parameterized properties, restated below.

Definition 2.1.2 (Tolerant Testing of Parameterized Properties)**.** Let $\mathcal{P} = (\mathcal{P})_{s \in \mathbb{N}}$ be a non-decreasing family of properties parameterized by $s \in \mathbb{N}$, i.e. such that $\mathcal{P}_s \subseteq \mathcal{P}_t$ whenever $s \leq t$; and $\sigma \colon \mathbb{N} \to \mathbb{N}$ be a non-decreasing mapping satisfying $\sigma(s) \geq s$ for all s. A *σ-tolerant testing algorithm for $\mathcal{P}$* is a probabilistic algorithm that gets three input parameters $s \in \mathbb{N}$ and $\varepsilon_1, \varepsilon_2 \in [0,1]$ such that $\varepsilon_1 < \varepsilon_2$, as well as oracle access to a function $f \colon \{-1,1\}^n \to \{-1,1\}$. The algorithm should output a binary verdict that satisfies the following two conditions.

- If $\mathrm{dist}(f, \mathcal{P}_s) \le \varepsilon_1$ then the algorithm accepts f with probability at least 2/3.
- If $\mathrm{dist}(f, \mathcal{P}_{\sigma(s)}) > \varepsilon_2$, then the algorithm rejects f with probability at least 2/3.

In some cases the algorithm is only given one parameter, ε_2, setting $\varepsilon_1 = r(\varepsilon_2)$ for some prespecified function $r\colon (0,1) \to (0,1)$.

Note that when σ is the identity function, the above definition corresponds to the tolerant testing model, defined in [PRR06]. We sometimes use the "non-uniform" variant of the above definition. Specifically, for parameters $0 < \varepsilon_0 < \varepsilon_1 < 1$, an $(\varepsilon_0, \varepsilon_1)$-tolerant testing algorithm is given an oracle access to the input, and is required to determine (with high probability) whether a given input is ε_0-close to the property or whether it is ε_1-far from it.

The next lemma will be useful to prove that some properties are hard to test. The lemma states that if we have two distributions whose restrictions to any set of queries of size at most q are identical, then no (possibly adaptive) algorithm making at most q queries can distinguish between them.

Definition 2.1.3 (Restriction)**.** Given a distribution $\mathcal{D}$ over functions $f : D \to \{0,1\}$ and a subset $Q \subseteq D$, we define the *restriction* $\mathcal{D}|_Q$ *of* $\mathcal{D}$ *to* Q to be the distribution over functions $g : Q \to \{0,1\}$, that results from choosing a function $f : D \to \{0,1\}$ according to $\mathcal{D}$, and setting g to be $f|_Q$, the restriction of f to Q.

Lemma 2.1.4 (e.g., [FNS04], special case)**.** *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distributions of functions over some domain D. Suppose that for any set $Q \subset D$ of size at most q, the restricted distributions $\mathcal{D}_1|_Q$ and $\mathcal{D}_2|_Q$ are identically distributed. Then, any (possibly adaptive) algorithm making at most q queries cannot distinguish $\mathcal{D}_1$ from $\mathcal{D}_2$ with any positive probability.*

The main focus of this work will be the property of being a *junta*, that is, a Boolean function that only depends on a (small) subset of its variables:

Definition 2.1.5 (k-Junta)**.** A Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ is a *k-junta* if there exists a set $T \subseteq [n]$ of size at most k, such that $f(x) = f(y)$ for every two assignments $x, y \in \{-1,1\}^n$ that satisfy $x_i = y_i$ for every $i \in T$. We let $\mathcal{J}_k$ denote the set of all k-juntas (over n variables).

2.2 Influence of variables

An important notion in this work is the *influence* of a set of variables of a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, which generalizes the standard notion of influence of a variable:

Definition 2.2.1 (Set-influence)**.** For a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$, the *set-influence* of a set $S \subseteq [n]$ is defined as

$$\mathbf{Inf}_f(S) \stackrel{\text{def}}{=} 2\,\mathbf{Pr}[f(\boldsymbol{x} \sqcup \boldsymbol{u}) \neq f(\boldsymbol{x} \sqcup \boldsymbol{v})],^{1}$$

where $\boldsymbol{x} \sim \{-1,1\}^{[n]\setminus S}$, $\boldsymbol{u}, \boldsymbol{v} \sim \{-1,1\}^S$.

The following properties of the influence function will be used extensively throughout this thesis.

[1]Here, we define the influence of a set with an additional factor 2, so that the (set-)influence of a singleton $\{i\}$ coincides with the standard definition of the influence of the i-th variable (as the latter definition asks that the i-th bit be flipped instead of re-randomized).

Fact 2.2.2 (See Chapter 5.3 in [Gol17])**.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a Boolean function.*

1. *(Monotonicity and subadditivity of influence) For any $S, T \subseteq [n]$*

$$\mathbf{Inf}_f(S) \leq \mathbf{Inf}_f(S \cup T) \leq \mathbf{Inf}_f(S) + \mathbf{Inf}_f(T).$$

2. *(Submodularity) For any $T \subset S$ and $i \notin S$, it holds that*

$$\mathbf{Inf}_f(T \cup \{i\}) - \mathbf{Inf}_f(T) \geq \mathbf{Inf}_f(S \cup \{i\}) - \mathbf{Inf}_f(S).$$

A direct implication of the definition of the influence function is an approximation algorithm for the influence of a set S.

Lemma 2.2.3 (Influence estimation)**.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a Boolean function. There exists an algorithm (Algorithm 1) receiving as an input oracle access to f, a set $S \subseteq [n]$, and $\gamma, \delta, \varepsilon \in (0,1)$, uses $m = O\left(\frac{\log(1/\delta)}{\varepsilon \cdot \gamma^2}\right)$ queries to f and satisfies the following with probability at least $1 - \delta$.*

1. *If $\mathbf{Inf}_f(S) \geq \varepsilon$, then the algorithm outputs an estimate $\widetilde{\mathbf{Inf}}_f(S)$ such that*

$$(1-\gamma) \cdot \mathbf{Inf}_f(S) \leq \widetilde{\mathbf{Inf}}_f(S) \leq (1+\gamma) \cdot \mathbf{Inf}_f(S).$$

2. *If $\mathbf{Inf}_f(S) < \varepsilon$, then the algorithm outputs an estimate $\widetilde{\mathbf{Inf}}_f(S)$ such that*

$$\widetilde{\mathbf{Inf}}_f(S) \leq (1+\gamma) \cdot \mathbf{Inf}_f(S).$$

Algorithm 1 Approximate Influence $(\varepsilon, \gamma, \delta, S)$

```
1: Set $m = \frac{C \cdot \ln(2/\delta)}{\gamma^2 \varepsilon}$, where $C \geq 1$ is an absolute constant.     ▷ $C \geq 3$ is sufficient.
2: for $i = 1$ to $m$ do
3:     Pick $\boldsymbol{x} \sim \{-1,1\}^{\overline{S}}$, $\boldsymbol{u}, \boldsymbol{v} \sim \{-1,1\}^S$ uniformly at random and independently.
4:     Set $\boldsymbol{\vartheta}_i \leftarrow \mathbb{1}_{\{f(\boldsymbol{x} \sqcup \boldsymbol{u}) \neq f(\boldsymbol{x} \sqcup \boldsymbol{v})\}}$.     ▷ $\boldsymbol{\vartheta}_i$ is 1 if $f(\boldsymbol{x} \sqcup \boldsymbol{u}) \neq f(\boldsymbol{x} \sqcup \boldsymbol{v})$, 0 otherwise.
5: end for
6: Let $\widetilde{\mathbf{Inf}}_f(S) \leftarrow \frac{2}{m} \sum_{i \in [m]} \boldsymbol{\vartheta}_i$ be the estimate of the influence of $S$.
```

Proof: We start by proving the first item. Note that by definition of the influence and the fact that $\mathbf{Inf}_f(S) \geq \varepsilon$,

$$\varepsilon < \mathbf{Inf}_f(S) = 2\,\mathbf{Pr}[f(\boldsymbol{x} \sqcup \boldsymbol{u}) \neq f(\boldsymbol{x} \sqcup \boldsymbol{v})] = 2 \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}} \left[\mathbb{1}_{\{f(\boldsymbol{x} \sqcup \boldsymbol{u}) \neq f(\boldsymbol{x} \sqcup \boldsymbol{v})\}}\right],$$

where $\boldsymbol{x} \sim \{-1,1\}^{\overline{S}}$, and $\boldsymbol{u}, \boldsymbol{v} \sim \{-1,1\}^S$. Therefore, by a Chernoff bound

$$\mathbf{Pr}\left[\left|\widetilde{\mathbf{Inf}}_f(S) - \mathbf{Inf}_f(S)\right| > \gamma \cdot \mathbf{Inf}_f(S)\right] \leq 2\exp\left(-\frac{m\gamma^2 \cdot \mathbf{Inf}_f(S)}{3}\right) \leq \delta.$$

The proof for the case where $\mathbf{Inf}_f(S) < \varepsilon$, follows similarly. ■

2.3 Probabilistically checkable proofs of proximity (PCPP)

A PCPP verifier for a property $\mathcal{P}$ is given access to an input x and a proof π, as well as a *detection radius* $\varepsilon > 0$ and *soundness error* $\delta > 0$. The verifier should make a constant number of queries (depending only on ε, δ) to the input x and the proof π, and satisfy the following. If $x \in \mathcal{P}$, then there exists π for which the verifier should always accept x. If $\mathrm{dist}(x, \mathcal{P}) > \varepsilon$, the verifier should reject x with probability at least δ, regardless of the contents of π. More formally, we define the following.

Definition 2.3.1 (PCPP)**.** For $n \in \mathbb{N}$, let $\mathcal{P} \subset \{0,1\}^n$ be a property of n-bit Boolean strings, and let $t \in \mathbb{N}$. We say that $\mathcal{P}$ has a *$q(\varepsilon, \delta)$-query, length-t Probabilistically Checkable Proof of Proximity (*PCPP*) system* if the following holds: There exists a verification algorithm V that takes as input $\varepsilon, \delta > 0$ and $n \in \mathbb{N}$, makes a total of $q(\varepsilon, \delta)$ queries on strings $w \in \{0,1\}^n$ and $\pi \in \{0,1\}^t$, and satisfies the following:

1. (Completeness) If $w \in \mathcal{P}$, then there exists a proof $\pi = \mathbf{Proof}_{\mathcal{P}}(w) \in \{0,1\}^t$ such that for every $\varepsilon, \delta > 0$, the verifier V accepts with probability 1.

2. (Soundness) If $\mathrm{dist}(w, \mathcal{P}) > \varepsilon$, then for every alleged proof $\pi \in \{0,1\}^t$, the verifier V rejects with probability greater than δ.

The following lemma, establishing the existence of a quasilinear PCPP for any property $\mathcal{P}$ that is verifiable in quasilinear time, will be an important tool throughout this work.

Lemma 2.3.2 (Corollary 8.4 in [Din07], see also [GM07])**.** *Let $\mathcal{P}$ be a property of Boolean strings which is verifiable by a size t Boolean circuit. Then, there exists a length-t'* PCPP *system for $\mathcal{P}$, that for every $\varepsilon, \delta > 0$ makes at most $q(\varepsilon, \delta)$ queries, where $t' = t \cdot \mathrm{polylog}\, t$.*

We note that maximally hard properties cannot have a constant query PCPP proof systems with a sublinear length proof string.

Proposition 2.3.3. *Let $\mathcal{P} \subseteq \{0,1\}^n$ and $\varepsilon > 0$ be such that any ε-tester for $\mathcal{P}$ has to make $\Omega(n)$ many queries. Then, any constant query* PCPP *system for $\mathcal{P}$ (for e.g., where $\delta = 1/3$) must have proof length of size $\Omega(n)$.*

Proof: Suppose that there exists a PCPP for $\mathcal{P}$ with $O(1)$ queries and proof length $t = o(n)$. Since the PCPP verifier has constant query complexity, we may assume that it is non adaptive and uses $q = O(1)$ queries. By a standard amplification argument, we can construct an amplified verifier that makes $O(q \cdot t) = o(n)$ queries, with soundness $2^{-t}/3$. By the fact that the verifier is non-adaptive, it has the same query distribution regardless of the proof string. Therefore, we can run 2^t amplified verifiers in parallel while reusing queries, one verifier for each of the 2^t possible proof strings. If any of the 2^t amplified verifiers accept, we accept the input. If the input belongs to $\mathcal{P}$, one of the above 2^t verifiers will accept (the one that used the correct proof). If the input was ε-far from $\mathcal{P}$, then by a union bound, the probability that there was any accepting amplified verifier is at most $1/3$. This yields an $o(n)$ tester for $\mathcal{P}$, which contradicts our assumption. ■

2.4 Probabilistic tools

Throughout this thesis, we use a generalization of Chernoff bounds for *negatively correlated* random variables.

Definition 2.4.1. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \{0,1\}$ be random variables. We say that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are *negatively correlated* if for all $I \subset [n]$ the following hold:

$$\mathbf{Pr}\left[\forall i \in I \ : \ \boldsymbol{x}_i = 0\right] \leq \prod_{i \in I} \mathbf{Pr}\left[\boldsymbol{x}_i = 0\right]$$
$$\mathbf{Pr}\left[\forall i \in I \ : \ \boldsymbol{x}_i = 1\right] \leq \prod_{i \in I} \mathbf{Pr}\left[\boldsymbol{x}_i = 1\right] \ .$$

Theorem 2.4.2 (Theorem 1.16 from [Doe11])**.** *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be negatively correlated binary random variables. Let $a_1, \ldots, a_n \in [0,1]$ and $\mathbf{X} = \sum_{i=1}^n a_i \boldsymbol{x}_i$. Then, for $\delta \in [0,1]$,*

$$\mathbf{Pr}\left[\boldsymbol{x} \geq (1+\delta)\,\mathbf{E}\left[\boldsymbol{x}\right]\right] \leq \exp(-\delta^2\,\mathbf{E}[\boldsymbol{x}]/2)$$
$$\mathbf{Pr}\left[\boldsymbol{x} \leq (1-\delta)\,\mathbf{E}\left[\boldsymbol{x}\right]\right] \leq \exp(-\delta^2\,\mathbf{E}[\boldsymbol{x}]/3) \ .$$

In addition, some of our proofs will use *hyper-geometric* random variables. Consider a population of size N that consists of K objects of a special type. Suppose n objects are picked without replacement. Let $\boldsymbol{x}$ be a random variable that counts the number of special objects picked in the sample. Then, we say that $\boldsymbol{x}$ is a hyper-geometric random variable, and we denote $\boldsymbol{x} \sim \mathrm{HG}(N, K, n)$. These hyper-geometric random variables enjoy tight concentration inequities (which are similar to Chernoff type bounds).

Theorem 2.4.3 ([Hoe63])**.** *Let $\boldsymbol{x} \sim \mathrm{HG}(N, K, n)$ and $\mu = K/N$. Then for any $t > 0$*

$$\mathbf{Pr}\left[\boldsymbol{x} \leq (\mu - t)n\right] \leq \exp(-2t^2 n)$$
$$\mathbf{Pr}\left[\boldsymbol{x} \geq (\mu + t)n\right] \leq \exp(-2t^2 n) \ .$$

2.5 Collection of covers

In this section we introduce a combinatorial claim that will be useful later on in Section 3.2.

Definition 2.5.1. Let X be a set of j elements, and for any $s \in [j]$ consider the family $\binom{X}{s}$ of all subsets of X that have size s. We shall say that $\mathcal{C} \subseteq \binom{X}{s}$ is a *s-cover* of X, if $\bigcup_{Y \in \mathcal{C}} Y = X$. We shall say that $\mathcal{C}_1, \ldots, \mathcal{C}_m$ is a *legal collection of s-covers for X*, if each $\mathcal{C}_t$ is a cover of X, and these covers are disjoint.

Claim 2.5.2. *For any set X of j elements, there exists a legal collection of s-covers for X of size at least*

$$m \geq \left\lfloor \frac{\binom{j}{s}}{\left\lceil \frac{j}{s} \right\rceil} \right\rfloor \ .$$

(Moreover, this bound is tight.)

Proof: This claim follows from a result due to Baranyai [Bar75] on factorization of regular hypergraphs. We state this result, and describe how to derive the claim from it, below (recall that K_n^h denotes the h-regular hypergraph K_n^h on n vertices):[2]

Theorem 2.5.3 (Baranyai's Theorem [Bar75, Theorem 1])**.** *Let n, h be integers satisfying $1 \leq h \leq n$, and $a_1, \ldots, a_\ell$ integers such that $\sum_{i=1}^{\ell} a_i = \binom{n}{h}$. Then the edges of K_n^h can be partitioned into hypergraphs $\mathcal{H}_1 \ldots, \mathcal{H}_\ell$ such that*

[2]An exposition of this result and the original proof as given by Baranyai can also be found in [Bra15, Theorem 4.1.1].

(i) $|\mathcal{H}_i| = a_i$ *for all* $i \in [\ell]$*;*

(ii) each $\mathcal{H}_i$ *is* almost regular*: the number of hyperedges any two vertices* $u, v \in \mathcal{H}_i$ *participate in differs by at most one (and here, specifically, is either* $\left\lceil \frac{a_i h}{n} \right\rceil$ *or* $\left\lfloor \frac{a_i h}{n} \right\rfloor$*).*

We apply Theorem 2.5.3 as follows: setting $m \overset{\text{def}}{=} \left\lfloor \frac{\binom{j}{s}}{\left\lceil \frac{j}{s} \right\rceil} \right\rfloor \leq \binom{j-1}{s-1}$ and $\ell \overset{\text{def}}{=} m+1$, we let $a_i \overset{\text{def}}{=} \left\lceil \frac{j}{s} \right\rceil$ for all $1 \leq i \leq m$, and $a_\ell \overset{\text{def}}{=} \binom{j}{s} - \sum_{i=1}^{m} a_i \geq 0$. By the theorem, we obtain a partition of K_j^s into $\ell = m+1$ hypergraphs $\mathcal{H}_1 \dots, \mathcal{H}_\ell$ such that the first m satisfy:

(i) $|\mathcal{H}_i| = \left\lceil \frac{j}{s} \right\rceil$ for all $i \in [m]$;

(ii) for any $i \in [m]$, any vertex $u \in \mathcal{H}_i$ participates in either 1 or 2 hyperedges;

(and we cannot say much about the "remainder" hypergraph $\mathcal{H}_\ell$). Condition (ii) ensures that each of the first m hypergraphs obtained indeed defines a cover of the set of j elements by s-element subsets, while by definition of the partition of the hypergraph we are promised that these m s-covers are disjoint. This proves the lemma, as $\mathcal{H}_1 \dots, \mathcal{H}_m$ then induce a legal cover of X.

As for the optimality of the bound, it follows readily from observing that one must have $m \leq \left\lfloor \frac{\binom{j}{s}}{\lceil j/s \rceil} \right\rfloor$ since for every cover $\mathcal{C}$ we must have $|\mathcal{C}| \geq \lceil j/s \rceil$, and $\left|\binom{X}{s}\right| = \binom{j}{s}$. ■

2.6 Error correcting codes and polynomials over finite fields

The relative Hamming distance of two strings $x, y \in \Sigma^n$ is defined as $\operatorname{dist}(x,y) = \frac{1}{n} \cdot |\{i \in [n] \mid x_i \neq y_i\}|$. For a string $x \in \Sigma^n$ and a non-empty set $S \subseteq \Sigma^n$, we define $\operatorname{dist}(x,S) = \min_{y \in S} \operatorname{dist}(x,y)$. The following plays a central role in many complexity-related works, including ours.

Definition 2.6.1. A *code* is an injective function $C : \Sigma^k \to \Sigma^n$. If Σ is a finite field and C is a linear function (over Σ), then we say that C is a *linear code*. The *rate* of C is defined as k/n, whereas the *minimum relative distance* is defined as the minimum over all distinct $x, y \in \Sigma^k$ of $\operatorname{dist}(C(x), C(y))$.

An *ε-distance code* is a code whose minimum relative distance is at least ε. When for a fixed $\varepsilon > 0$ we have a family of ε-distance codes (for different values of k), we refer to its members as *error correcting codes*.

In this work we use the fact that efficient codes with constant rate and constant relative distance exist. Moreover, there exist such codes in which membership can be decided by a quasi-linear size Boolean circuit.

Theorem 2.6.2 (see e.g., [Spi96])**.** *There exists a linear code* $\mathsf{Spiel} : \{0,1\}^k \to \{0,1\}^{100k}$ *with constant relative distance, for which membership can be decided by a* $k \cdot \operatorname{polylog} k$ *size Boolean circuit.*

Actually, the rate of the code in [Spi96] is significantly better, but since we do not try to optimize constants, we use the constant 100 solely for readability. In addition, the code described in [Spi96] is *linear time decodeable*, but we do not make use of this feature throughout this work.

We slightly abuse notation, and for a finite field $\mathbb{F}$ of size 2^k, view the encoding given in 2.6.2 as $\mathsf{Spiel} : \mathbb{F} \to \{0,1\}^{100k}$, by associating $\{0,1\}^k$ with $\mathbb{F}$ in the natural way. Note that for

$f : \mathbb{F} \to \mathbb{F}$, it holds that $\langle\langle f(\beta)\rangle\rangle \in \{0,1\}^k$ for every $\beta \in \mathbb{F}$, and therefore $\mathsf{Spiel}(f(\beta)) \in \{0,1\}^{100k}$. We slightly abuse notation, and for a function $f : \mathbb{F} \to \mathbb{F}$ we write $\mathsf{Spiel}(f)$ to denote the length $100k \cdot 2^k$ bit string $\bigsqcup_{\beta\in\mathbb{F}} \mathsf{Spiel}(f(\beta))$ (where we use the canonical ordering over $\mathbb{F}$).

Definition 2.6.3. Let $\mathcal{C}_\mathbb{F}$ denote the set of polynomials $g : \mathbb{F} \to \mathbb{F}$ such that $\deg(g) \le \frac{|\mathbb{F}|}{2}$.

The following lemma of [Hor72], providing a fast univariate interpolation, will be an important tool in this work.

Lemma 2.6.4 ([Hor72])**.** *Given a set of pairs $\{(x_1, y_1), \ldots, (x_r, y_r)\}$ with all x_i distinct, we can output the coefficients of $p(x) \in \mathbb{F}[X]$ of degree at most $r-1$ satisfying $p(x_i) = y_i$ for all $i \in [r]$, in $O(r \cdot \log^3(r))$ additions and multiplications in $\mathbb{F}$.*

The next lemma states that a randomly chosen function $\boldsymbol{\lambda} : \mathbb{F} \to \mathbb{F}$ is far from any low degree polynomial with very high probability.

Lemma 2.6.5. *With probability at least $1 - o(1)$, a uniformly random function $\boldsymbol{\lambda} : \mathbb{F} \to \mathbb{F}$ is $1/3$-far from $\mathcal{C}_\mathbb{F}$.*

Proof: Consider the size of a ball of relative radius $1/3$ around some function $\lambda : \mathbb{F} \to \mathbb{F}$ in the space of functions from $\mathbb{F}$ to itself. The number of points (i.e., functions from $\mathbb{F} \to \mathbb{F}$) contained in this ball is at most

$$\binom{|\mathbb{F}|}{|\mathbb{F}|/3} \cdot |\mathbb{F}|^{|\mathbb{F}|/3} \le (3e|\mathbb{F}|)^{|\mathbb{F}|/3}.$$

By the fact that the size of $\mathcal{C}_\mathbb{F}$ is $|\mathbb{F}|^{|\mathbb{F}|/2+1}$, the size of the set of points that are at relative distance at most $1/3$ from any point in $\mathcal{C}_\mathbb{F}$ is at most

$$|\mathbb{F}|^{|\mathbb{F}|/2+1} \cdot (3e|\mathbb{F}|)^{|\mathbb{F}|/3} = o(|\mathbb{F}|^{|\mathbb{F}|}).$$

The lemma follows by observing that there are $|\mathbb{F}|^{|\mathbb{F}|}$ functions from $\mathbb{F}$ to itself. ■

2.6.1 Dual distance of linear codes

We focus here specifically on a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$, and consider the linear subspace of its image, $V_C = \{C(x) : x \in \mathbb{F}^k\} \subseteq \mathbb{F}^n$. We define the *distance* of a linear space as $\mathrm{dist}(V) = \min_{v\in V\setminus\{0^n\}} \mathrm{dist}(v, 0^n)$, and note that in the case of V being the image V_C of a code C, this is identical to $\mathrm{dist}(C)$. For a linear code, it helps to investigate also *dual distances.*

Definition 2.6.6. Given two vectors $u, v \in \mathbb{F}^n$, we define their *scalar product* as $u \cdot v = \sum_{i\in[n]} u_i v_i$, where multiplication and addition are calculated in the field $\mathbb{F}$. Given a linear space $V \subseteq \mathbb{F}^n$, its *dual space* is the linear space $V^\perp = \{u : \forall v \in V, u \cdot v = 0\}$. In other words, it is the space of vectors who are orthogonal to all members of V. The *dual distance* of the space V is simply defined as $\mathrm{dist}(V^\perp)$.

For a code C, we define its *dual distance*, $\mathrm{dist}^\perp(C)$, as the dual distance of its image V_C. We call C an *η-dual-distance* code if $\mathrm{dist}^\perp(C) \ge \eta$. The following well-known lemma is essential to us, as it will relate to the "secret-sharing" property that we define later.

Lemma 2.6.7 (See e.g., [MS77, Chapter 1, Theorem 10])**.** *Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear η-dual distance code, let $Q \subset [n]$ be any set of size less than $\eta \cdot n$, and consider the following random process for picking a function $\boldsymbol{u} : Q \to \mathbb{F}$: Let $\boldsymbol{w} \in \mathbb{F}^k$ be drawn uniformly at random, and set $\boldsymbol{u}$ be the restriction of $C(\boldsymbol{w})$ to the set Q. Then, the distribution of $\boldsymbol{u}$ is identical to the uniform distribution over the set of all functions from Q to $\mathbb{F}$.*

Chapter 3

Algorithms for Tolerant Junta Testing

In this chapter our main focus is developing algorithms for tolerant k-junta testing, while aiming for query complexities lower than $\exp(k)$. As discussed in Chapter 1, it is relatively easy to obtain an algorithm which uses $\exp(k)/\varepsilon$ many queries and distinguishes with high constant probability between the case where the function is ε/C-close to some k-junta (for some constant $C > 1$), and the case where the function is ε-far from any k-junta (as a warm-up toward a more involved algorithm, we present a self contained proof for the above in section 3.1).

In contrast, [DLM+07] observed that distinguishing functions which are ε/k-close to some k-junta from functions which are ε-far from any k-junta, can be done using $\mathrm{poly}(k, 1/\varepsilon)$ queries. The above two extreme cases suggest a certain tradeoff between tolerance and query complexity. Our first main result in this chapter is an algorithm which leverages this tradeoff, and allows in some cases to avoid the exponential dependence in k in the query complexity.

Theorem 1.3.1. *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$, $\varepsilon \in (0,1)$ and $\rho \in (0,1)$, satisfies the following.*

- *If f is $\rho\varepsilon/16$-close to some k-junta, then the algorithm accepts with high constant probability.*
- *If f is ε-far from every k-junta, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $O\left(\frac{k \log k}{\varepsilon\rho(1-\rho)^k}\right)$.

The above statement provides a smooth trade-off between the tolerance and the query complexity and interpolates the above extreme cases.

Our second main result in this chapter, considers whether a slight relaxation of the tolerant testing model exhibits significant savings in terms of query complexity. In Section 3.3 we show that requiring the algorithm to reject functions which are ε-far from $2k$-juntas (as oppose to the original definition where the algorithm rejects functions which are far from k-juntas), results in an algorithm which uses $\mathrm{poly}(k, 1/\varepsilon)$ queries. Specifically,

Theorem 1.3.2. *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$ and $\varepsilon \in (0,1)$, satisfies the following.*

- *If f is $\varepsilon/10$-close to some k-junta, then the algorithm accepts with high constant probability.*
- *If f is ε-far from every $2k$-junta, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $\mathrm{poly}(k, \frac{1}{\varepsilon})$.

The proofs of Theorems 1.3.2 and 1.3.1 both rely on the notion of the *influence* of a set of variables. Given a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ and a set $S \subseteq [n]$, the influence of the set S (denoted $\mathbf{Inf}_f(S)$) is the probability that $f(x) \neq f(y)$ when x and y are selected uniformly subject to the constraint that for any $i \in \bar{S}$, $x_i = y_i$ (see Definition 2.2.1). In both of the above results, we consider a fixed partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ of $[n]$ into $\ell = O(k^2)$ parts and for $S \subseteq [\ell]$ we let $\phi_{\mathcal{I}}(S) \stackrel{\text{def}}{=} \bigcup_{i \in S} I_i$.

3.1 Warm-up: An $\exp(k \log k)/\varepsilon$ algorithm using dimension reduction

In this section we present an $\exp(k \log k)/\varepsilon$ query algorithm for tolerant junta testing.

Lemma 3.1.1. *There exists an algorithm that, given query access to a function* $f : \{-1,1\}^n \to \{-1,1\}$ *and parameters* $k \geq 1$ *and* $\varepsilon \in (0,1)$*, satisfies the following.*

- *If f is $\varepsilon/3$-close to k-junta, then the algorithm accepts with probability at least* $2/3$.
- *If f is ε-far from every k-junta, then the algorithm rejects with probability at least* $2/3$.

The query complexity of the algorithm is $2^{(1+o(1))k \log k}/\varepsilon$.

The ideas behind our algorithm are similar to those in [FKR+04, Bla09]; We partition the n variables into $\ell = O(k^2)$ parts, which removes the dependence on n. It is not hard to verify that if f is close to $\mathcal{J}_k$, then there exist k parts for which the following holds. If we denote by $T \subseteq [n]$ the union of variables in these k parts, then the complement set $\bar{T}$ has small influence.

On the other hand, Blais [Bla09] showed that if a function is far from $\mathcal{J}_k$, then a random partition into a sufficiently large number of parts ensures the following with high constant probability. For every union T of k parts, the complement set $\bar{T}$ will have large influence. The above gives rise to a $(2^{(1+(o(1))k \log k}/\varepsilon)$-query complexity algorithm that distinguishes functions that are $\frac{1}{3}\varepsilon$-close to $\mathcal{J}_k$ from functions that are ε-far from $\mathcal{J}_k$. The algorithm considers all unions $T \subseteq [n]$ of k parts, estimates the influence of $\bar{T}$ using Lemma 2.2.3, and accepts if there exists a set with sufficiently small estimated influence.

We start with a useful definition of *k-part juntas*, and two lemmas regarding their properties with respect to random partitions of the domain. Recall that for a set S, we denote by $\binom{S}{r}$ the set of all subsets of S of size r. Given a partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ of $[n]$ and a set $J \subseteq [\ell]$, we denote by $\phi_{\mathcal{I}}(J)$ the union $\bigcup_{i \in J} I_i$.

Definition 3.1.2 (Partition juntas [Bla12, Definition 5.3], extended)**.** Let $\mathcal{I}$ be a partition of $[n]$ into ℓ parts, and $k \geq 1$. The function $f\colon \{-1,1\}^n \to \{-1,1\}$ is a *k-part junta with respect to $\mathcal{I}$* if the relevant coordinates in f are all contained in at most k parts of $\mathcal{I}$. Moreover,

(i) f is said to *ε-approximate being a k-part junta with respect to $\mathcal{I}$* if there exists a set $J \in \binom{[\ell]}{\ell-k}$ satisfying $\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)) \leq 2\varepsilon$.

(ii) Conversely, f is said to *ε-violate being a k-part junta with respect to $\mathcal{I}$* if for every set $J \in \binom{[\ell]}{\ell-k}$, $\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)) > 2\varepsilon$.

Lemma 3.1.3 ([Bla12, Lemma 5.4])**.** *For* $f\colon \{-1,1\}^n \to \{-1,1\}$ *and* $k \geq 1$*, let* $\alpha \stackrel{\text{def}}{=} \mathrm{dist}(f, \mathcal{J}_k)$*. Also, let $\boldsymbol{\mathcal{I}}$ be a random partition of $[n]$ with $\ell \stackrel{\text{def}}{=} 24k^2$ parts obtained by uniformly and independently assigning each coordinate to a part. With probability at least $5/6$ over the choice of the partition $\boldsymbol{\mathcal{I}}$, the function f $\frac{\alpha}{2}$-violates being a k-part junta with respect to $\boldsymbol{\mathcal{I}}$.*

Lemma 3.1.4. *For $f\colon \{-1,1\}^n \to \{-1,1\}$ and $k \geq 1$, let $\alpha \overset{\text{def}}{=} \operatorname{dist}(f, \mathcal{J}_k)$ and let $\mathcal{I}$ be any partition of $[n]$ into $\ell \geq k$ parts. Then f 2α-approximates being a k-part junta with respect to $\mathcal{I}$.*

Proof: Let $g \in \mathcal{J}_k$ be such that $\operatorname{dist}(f,g) = \operatorname{dist}(f,\mathcal{J}_k) = \alpha$. Let $I_{i_1},\dots,I_{i_r}$ be the $r \leq k$ parts of $\mathcal{I}$ containing the relevant variables of g. Then, for any set $J \subset [\ell]$ of size $\ell - k$ such that $\{i_1,\dots,i_r\} \subseteq \bar{J}$, we have that when drawing $\boldsymbol{x} \sim \{-1,1\}^{\phi_{\mathcal{I}}(\bar{J})}$, and $\boldsymbol{u},\boldsymbol{v} \sim \{-1,1\}^{\phi_{\mathcal{I}}(J)}$ the following holds.

$$\begin{aligned}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)) &= 2\,\mathbf{Pr}[f(\boldsymbol{x}\sqcup\boldsymbol{u}) \neq f(\boldsymbol{x}\sqcup\boldsymbol{v})] \leq 2\,\mathbf{Pr}[f(\boldsymbol{x}\sqcup\boldsymbol{u}) \neq g(\boldsymbol{x}\sqcup\boldsymbol{u}) \text{ or } f(\boldsymbol{x}\sqcup\boldsymbol{v}) \neq g(\boldsymbol{x}\sqcup\boldsymbol{v})] \\ &\leq 2\left(\mathbf{Pr}[f(\boldsymbol{x}\sqcup\boldsymbol{u}) \neq g(\boldsymbol{x}\sqcup\boldsymbol{u})] + \mathbf{Pr}[f(\boldsymbol{x}\sqcup\boldsymbol{v}) \neq g(\boldsymbol{x}\sqcup\boldsymbol{v})]\right) \leq 2\,(\alpha+\alpha) = 4\alpha\ ,\end{aligned}$$

where the first inequality follows from observing that (as g does not depend on variables in $\phi_{\mathcal{I}}(J)$) one can only have $f(\boldsymbol{x}\sqcup\boldsymbol{u}) \neq f(\boldsymbol{x}\sqcup\boldsymbol{v})$ if f disagrees with g on at least one of the two points; and the third inequality holds since both $\boldsymbol{x}\sqcup\boldsymbol{u}$ and $\boldsymbol{x}\sqcup\boldsymbol{v}$ are uniformly distributed. ■

The above two lemmas suggest the following approach for distinguishing between functions that are ε'-close to some k-junta and functions that are ε-far from every k'-junta. Suppose we select a random partition of $[n]$ into $O(k^2)$ parts. Then, with high probability over the choice of the partition, it is sufficient to distinguish between functions that $2\varepsilon'$-approximate being a k-junta and functions that $\varepsilon/2$-violate being a k'-part junta. Specifically, we get the proposition below, which we apply throughout this work:

Proposition 3.1.5 (Reduction to part juntas)**.** *Let $\mathcal{T}$ be an algorithm that is given query access to a function $f : \{-1,1\}^n \to \{-1,1\}$, a partition $\mathcal{I} = \{I_1,\dots,I_\ell\}$ of $[n]$ into ℓ parts, and parameters $k \in \mathbb{N}$ and $\varepsilon \in (0,1)$. Suppose that $\mathcal{T}$ performs $q(k,\varepsilon,\ell)$ queries to f and satisfies the following guarantees, for a pair of functions $r\colon (0,1)\times\mathbb{N} \to (0,1)$ and $r'\colon \mathbb{N} \to \mathbb{N}$.*

- *If f ε'-approximates being a k-part junta with respect to $\mathcal{I}$ and $\varepsilon' \leq r(\varepsilon,k)$, then $\mathcal{T}$ returns* accept *with probability at least $5/6$;*
- *If f ε-violates being a k'-part junta with respect to $\mathcal{I}$ and $k' \geq r'(k)$, then $\mathcal{T}$ returns* reject *with probability at least $5/6$.*

Then there exists an algorithm $\mathcal{T}'$, that given query access to f and parameters $k \in \mathbb{N}$ and $\varepsilon \in (0,1)$, satisfies the following.

- *If $\operatorname{dist}(f,\mathcal{J}_k) \leq \frac{\varepsilon'}{2}$ and $\varepsilon' \leq r(\varepsilon,k)$, then $\mathcal{T}'$ outputs* accept *with probability at least $2/3$;*
- *If $\operatorname{dist}(f,\mathcal{J}_{k'}) > 2\varepsilon$ and $k' \geq r'(k)$, then $\mathcal{T}'$ outputs* reject *with probability at least $2/3$.*

Moreover, the algorithm $\mathcal{T}'$ has query complexity $q(k,\varepsilon,\ell)$.

Proof of Proposition 3.1.5: The algorithm $\mathcal{T}'$ first obtains a random partition $\boldsymbol{\mathcal{I}}$ of $[n]$ into $\ell \overset{\text{def}}{=} 24(k')^2$ parts by uniformly and independently assigning each coordinate to a part. $\mathcal{T}'$ then invokes $\mathcal{T}$ with parameters ε, k, ℓ and the partition $\boldsymbol{\mathcal{I}}$. By Lemma 3.1.3 and the choice of ℓ, with probability at least $5/6$ the partition $\boldsymbol{\mathcal{I}}$ is *good* in the following sense. For $\alpha = \operatorname{dist}(f,\mathcal{J}_{k'})$, it holds that f $\frac{\alpha}{2}$-violates being a k'-part junta with respect to $\boldsymbol{\mathcal{I}}$. Conditioned on $\boldsymbol{\mathcal{I}}$ being good, and by Lemma 3.1.4, we are guaranteed that the following holds.

(i) If $\operatorname{dist}(f,\mathcal{J}_k) \leq \frac{\varepsilon'}{2}$, then f ε'-approximates being a k-part junta with respect to $\boldsymbol{\mathcal{I}}$;

(ii) If $\operatorname{dist}(f,\mathcal{J}_{k'}) > 2\varepsilon$, then f ε-violates being a k'-part junta with respect to $\boldsymbol{\mathcal{I}}$.

Therefore, $\mathcal{T}$ will answer as specified by the proposition with probability at least 5/6, making $q(\varepsilon, k, \ell)$ queries. Overall, by a union bound, $\mathcal{T}'$ is successful with probability at least 2/3. $\blacksquare$

Proof of Lemma 3.1.1: Given a partition $\mathcal{I}$ of $[n]$ into ℓ parts, $\mathcal{T}$ considers all $\binom{\ell}{\ell-k}$ sets of variables that result from taking the union of k parts. For each such set T, it uses Lemma 2.2.3 to obtain an estimate $\widetilde{\mathbf{Inf}}_f(T)$ of the influence of T, by performing $O\left(\frac{\ell \log \ell}{\varepsilon}\right)$ queries to f . $\mathcal{T}$ accepts if for at least one of the sets T, $\widetilde{\mathbf{Inf}}_f(T)$ is at most $\frac{3}{2}\varepsilon$. Performing $O(\frac{\ell \log \ell}{\varepsilon})$ queries to the oracle for each set, ensures that the following holds with high constant probability. For every set T such that $\mathbf{Inf}_f(T) \leq \frac{4}{3}\varepsilon$, $\widetilde{\mathbf{Inf}}_f(T) \leq \frac{3}{2}\varepsilon$ and for every set T such that $\mathbf{Inf}_f(T) > 2\varepsilon$, $\widetilde{\mathbf{Inf}}_f(T) > \frac{3}{2}\varepsilon$. Hence, the algorithm $\mathcal{T}$ fulfills the requirements stated in Proposition 3.1.5 (for $r(\varepsilon, k) = \frac{2}{3}\varepsilon$ and $r'(k) = k$), and it follows that:

- If f is $\frac{1}{3}\varepsilon$-close to some k-junta then $\mathcal{T}'$ accepts with probability at least 2/3.
- If f is ε-far from every k-junta then $\mathcal{T}'$ rejects with probability at least 2/3.

Since $\ell = 24k^2$, the query complexity of the algorithm is $\binom{\ell}{\ell-k} \cdot O(\frac{\ell \log \ell}{\varepsilon}) = 2^{(1+o(1))k \log k}/\varepsilon$. $\blacksquare$

3.2 A tradeoff between tolerance and query complexity

In this section, we show how to obtain a smooth tradeoff between the amount of tolerance and the query complexity. Formally, we prove Theorem 1.3.1, restated below.

Theorem 1.3.1. *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$, $\varepsilon \in (0,1)$ and $\rho \in (0,1)$, satisfies the following.*

- *If f is $\rho\varepsilon/16$-close to some k-junta, then the algorithm accepts with high constant probability.*
- *If f is ε-far from every k-junta, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $O\left(\frac{k \log k}{\varepsilon\rho(1-\rho)^k}\right)$.

As discussed in the introduction, this in particular implies the two following results. Setting $\rho = \Omega(1)$, we obtain a tolerant tester that distinguishes between functions $O(\varepsilon)$-close to $\mathcal{J}_k$ and functions ε-far from $\mathcal{J}_k$, with query complexity $2^{O(k \log k)}/\varepsilon$, an improvement over the naive tester from section 3.1. On the other hand, choosing $\rho = O(1/k)$ yields a weakly tolerant tester that distinguishes functions $O(\varepsilon/k)$-close to $\mathcal{J}_k$ from those ε-far from $\mathcal{J}_k$, with query complexity $\tilde{O}(k^2/\varepsilon)$ – thus matching the guarantees provided in [FKR+04].

The key idea behind our approach is the following. The exhaustive search algorithm, presented in Section 3.1 estimates the influence of the set of variables $\phi_{\mathcal{I}}(J)$ for every set of indices $J \subset [\ell]$ such that $|J| = \ell - k$ by performing pairs of queries specifically designed for J. Namely, it queries the value of the function on pairs of points in $\{-1,1\}^n$ that agree on the set $\bar{J}$. If it were possible to use the same queries for estimating the influence of $\phi_{\mathcal{I}}(J)$ for different choices of J, then we could reduce the query complexity. We show that this can be done if we consider the *ρ-biased subset influence* of a set $J \subset [\ell]$, defined next.

Given a partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$, a parameter $\rho \in (0,1)$, and a set $J \subset [\ell]$, a random ρ-biased subset $\mathbf{S} \sim_\rho J$ is a subset of J resulting from taking every index in J to S with probability ρ. The expected influence of a random ρ-biased subset of J, referred to as the ρ-subset influence of J, is $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))]$. We prove that for every set $J \subseteq [\ell]$, its ρ-subset influence is in $[\frac{\rho}{3}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)), \mathbf{Inf}_f(\phi_{\mathcal{I}}(J))]$.

A crucial element in our proof is a combinatorial result due to Baranyai [Bar75] on factorization of regular hypergraphs (see Section 2.5) . With this fact in hand, we then present an algorithm that allows to simultaneously estimate the ρ-subset influence of all sets $J \subset [\ell]$ of size $\ell - k$. The query complexity of the algorithm is $O\left(\frac{k \log k}{\varepsilon\rho(1-\rho)^k}\right)$.

3.2.1 Useful bounds on the expected influence of a random ρ-subset of a set

In this subsection we formally define the ρ-subset influence of a set and prove that for every set $J \subseteq [\ell]$, its ρ-subset influence is at least $\frac{\rho}{3} \cdot \mathbf{Inf}_f(\phi_{\mathcal{I}}(J))$ and at most $\mathbf{Inf}_f(\phi_{\mathcal{I}}(J))$. Then in the next subsection we provide an algorithm that simultaneously estimates the ρ-subset influence of all subsets J of $[\ell]$ of size $\ell - k$. The query complexity of the algorithm is $O\left(\frac{k \log k}{\varepsilon\rho(1-\rho)^k}\right)$. We start with a few definitions and notations.

Definition 3.2.1. For any $\rho \in (0, 1)$ and any set R, we denote by $\mathbf{S} \sim_\rho R$ the random ρ-biased subset of R, resulting from including independently each $i \in R$ in $\mathbf{S}$ with probability ρ. We refer to such a set $\mathbf{S}$ as a *random ρ-subset of R.*

Definition 3.2.2. For a partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ and a set $J \subseteq [\ell]$ we refer to the expected value of the influence of a random ρ-biased subset of J, $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))]$, as the *$\rho$-subset influence* of J (with respect to $\mathcal{I}$).

The next lemma describes the connection between the influence of a set J and its ρ-subset influence.

Lemma 3.2.3. *Let $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ be a partition of $[n]$. Then, for any $J \subseteq [\ell]$ and $\rho > 0$,*

$$\frac{\rho}{3}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)) \leq \mathop{\mathbf{E}}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] \leq \mathbf{Inf}_f(\phi_{\mathcal{I}}(J)).$$

Proof: The upper bound is immediate by monotonicity of the influence, as $\mathbf{Inf}_f(\phi_{\mathcal{I}}(S)) \leq \mathbf{Inf}_f(\phi_{\mathcal{I}}(J))$ for all $S \subseteq J$ (see Fact 2.2.2, item 1). As for the lower bound, let $j = |J|$ and observe that

$$\mathop{\mathbf{E}}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] = \sum_{s=1}^{j} \sum_{S\subseteq J:|S|=s} \rho^s(1-\rho)^{j-s} \cdot \mathbf{Inf}_f(\phi_{\mathcal{I}}(S)). \tag{3.1}$$

We will lower bound the sum $\sum_{S\subseteq J:|S|=s} \mathbf{Inf}_f(\phi_{\mathcal{I}}(S))$ for each s separately. In order to do so we use Definition 2.5.1 regarding *legal collection of s-covers for a set J, denoted by $\mathscr{C}_J$.*

We are interested in showing that there exists a legal collection of s-covers for J whose size $m = |\mathscr{C}_J|$ is "as big as possible". This is established in Claim 2.5.2 and gives

$$m \geq \left\lfloor \frac{\binom{j}{s}}{\left\lceil \frac{j}{s} \right\rceil} \right\rfloor.$$

This claim is obtained from a result due to Baranyai [Bar75] on factorization of regular hypergraphs: for completeness, we state this result, and describe how to derive the claim from it, in Section 2.5.

Observe that if s divides j then $\left\lfloor \frac{\binom{j}{s}}{\lceil \frac{j}{s} \rceil} \right\rfloor = \binom{j-1}{s-1}$; and otherwise

$$\left\lfloor \frac{\binom{j}{s}}{\left\lceil \frac{j}{s} \right\rceil} \right\rfloor = \left\lfloor \frac{\frac{j}{s}}{\left\lceil \frac{j}{s} \right\rceil}\binom{j-1}{s-1} \right\rfloor \geq \left\lfloor \frac{\frac{j}{s}}{\frac{j}{s}+1}\binom{j-1}{s-1} \right\rfloor \geq \left\lfloor \frac{1}{2}\binom{j-1}{s-1} \right\rfloor \geq \frac{1}{3}\binom{j-1}{s-1}.$$

Therefore, by using the monotomicity of the influence (see Fact 2.2.2, item 1)

$$\sum_{S\subseteq J:|S|=s} \mathbf{Inf}_f(\phi_{\mathcal{I}}(S)) = \sum_{S\in\binom{J}{s}} \mathbf{Inf}_f(\phi_{\mathcal{I}}(S)) \geq \sum_{\mathcal{C}\in\mathscr{C}_J}\sum_{S\in\mathcal{C}} \mathbf{Inf}_f(\phi_{\mathcal{I}}(S))$$
$$\geq |\mathscr{C}_J|\cdot \mathbf{Inf}_f(\phi_{\mathcal{I}}(J)) \geq \frac{1}{3}\binom{j-1}{s-1}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)).$$

Plugging the above into Equation (3.1), we obtain that

$$\begin{aligned}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)) &\geq \sum_{s=1}^{j} \rho^s(1-\rho)^{j-s}\cdot\left(\frac{1}{3}\binom{j-1}{s-1}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J))\right)\\ &= \frac{\rho}{3}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J))\sum_{s=1}^{j}\binom{j-1}{s-1}\rho^{s-1}(1-\rho)^{j-s}\\ &= \frac{\rho}{3}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J))(\rho+(1-\rho))^{j-1} = \frac{\rho}{3}\mathbf{Inf}_f(\phi_{\mathcal{I}}(J)),\end{aligned}$$

which concludes the proof. ■

3.2.2 Approximation of the ρ-subset influences

We now describe and analyze an algorithm that given a partition $\mathcal{I} = \{I_1,\dots,I_\ell\}$, allows to simultaneously get good estimates of the ρ-subset influences of all subset $J \in \binom{[\ell]}{\ell-k}$. This algorithm is the main building block of the tolerant junta tester of Theorem 1.3.1.

Algorithm 2 Simultaneously Approximate ρ-subset Influence $(\rho,\varepsilon,\gamma,k,\ell,\mathcal{I})$

```
1: Set $m = \frac{C\cdot k\log\ell}{\gamma^2\varepsilon\rho(1-\rho)^k}$, where $C \geq 1$ is an absolute constant.   ▷ $C \geq 256\ln 2$ is sufficient.
2: for $i = 1$ to $m$ do
3:     Let $\mathbf{S}_i \sim_\rho [\ell]$.
4:     Pick $x^i \in \{-1,1\}^n$ uniformly at random
5:     Let $z^i$ obtained by flipping independently each coordinate of $x^i_{\phi_{\mathcal{I}}(\mathbf{S}_i)}$ w.p 1/2
6:     Set $y^i \leftarrow x^i_{\phi_{\mathcal{I}}(\bar{\mathbf{S}}_i)} \sqcup z^i$.
7:     Set $\vartheta_{\mathbf{S}_i} \leftarrow \mathbb{1}_{\{f(x^i)\neq f(y^i)\}}$ .   ▷ $\vartheta_{\mathbf{S}_i}$ is 1 if $f(x^i) \neq f(y^i)$, 0 otherwise.
8: end for
9: Let $\boldsymbol{\mathcal{S}}$ be the multiset of subsets $\mathbf{S}_1,\dots,\mathbf{S}_m$.
10: for every $J \in \binom{[\ell]}{\ell-k}$ do
11:     Let $\boldsymbol{\mathcal{S}}_J \subseteq \boldsymbol{\mathcal{S}}$ denote the subset of sets $\mathbf{S} \in \boldsymbol{\mathcal{S}}$ such that $\mathbf{S} \subseteq J$ .
12:     Let $\boldsymbol{\nu}^\rho_J \leftarrow \frac{1}{|\boldsymbol{\mathcal{S}}_J|}\sum_{\mathbf{S}\in\boldsymbol{\mathcal{S}}_J}\vartheta_{\mathbf{S}}$ be the estimate of the $\rho$-subset influence of $J$.
13: end for
```

Lemma 3.2.4. *Let $\mathcal{I} = \{I_1,\dots,I_\ell\}$ be a partition of $[n]$. For every $\varepsilon \in (0,1)$ and $\rho \in (0,1)$, Algorithm 2 satisfies that, with probability at least $1-o(1)$, the following holds simultaneously for all sets $J \in \binom{[\ell]}{\ell-k}$:*

1. *If $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] > \frac{\rho\varepsilon}{3}$, then the estimate $\boldsymbol{\nu}^\rho_J$ is within a multiplicative factor of $(1\pm\gamma)$ of the ρ-subset influence of J.*

2. *If $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] \leq \frac{\rho\varepsilon}{4}$, then the estimate $\boldsymbol{\nu}^\rho_J$ does not exceed $(1+\gamma)\frac{\rho\varepsilon}{4}$.*

Proof: Let $m' \overset{\text{def}}{=} \frac{1}{2}(1-\rho)^k \cdot m = \frac{Ck\log\ell}{2\gamma^2\varepsilon\rho}$. We first claim that for any fixed set $J \subseteq [\ell]$ of size $\ell - k$, with probability at least $1 - o(\ell^{-2k})$, $|\boldsymbol{\mathcal{S}}_J| \geq m'$. To see why this is true, fix some $J \subset [\ell]$ of size $\ell - k$. For every $i \in [m]$, let $\mathbb{1}_{\{\mathbf{S}_i \subseteq J\}}$ be an indicator variable which is equal to 1 if and only if $\mathbf{S}_i \subseteq J$. Then, for every $i \in [m]$, $\mathbf{Pr}[\mathbb{1}_{\{\mathbf{S}_i \subseteq J\}} = 1] = (1-\rho)^k$. By a Chernoff bound,

$$\mathbf{Pr}\left[\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{\mathbf{S}_i \subseteq J\}} < \frac{1}{2}\cdot(1-\rho)^k\right] \leq e^{-\frac{m}{8}(1-\rho)^k} = e^{-\frac{C\cdot k\log\ell}{8\varepsilon\rho\gamma^2}} < 2^{-4k\log\ell},$$

for a suitable choice of $C \geq 1$. Therefore, by a union bound over all $\binom{\ell}{\ell-k} = \binom{\ell}{k} = 2^{k\log\ell}$ sets $J \in \binom{[\ell]}{\ell-k}$, it holds that with probability $1 - o(1)$, for every such J, $|\boldsymbol{\mathcal{S}}_J| \geq m'$. We hereafter condition on this.

We now turn to prove the two items of the lemma. Let $\mathbf{X} = \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^m\}$ and $\mathbf{Z} = \{\boldsymbol{z}^1, \dots, \boldsymbol{z}^m\}$. For a set $\mathbf{S}_i$, $\mathbf{E}_{\boldsymbol{x}^i,\boldsymbol{z}^i}[\vartheta_{\mathbf{S}}] = \mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))$. Hence, by the definition of $\boldsymbol{\nu}_J^\rho$ in Step 12 of the algorithm,

$$\begin{aligned}\mathbf{E}[\boldsymbol{\nu}_J^\rho] &= \mathop{\mathbf{E}}_{\boldsymbol{\mathcal{S}},\mathbf{X},\mathbf{Z}}\left[\frac{1}{|\boldsymbol{\mathcal{S}}_J|}\sum_{\mathbf{S}\in\boldsymbol{\mathcal{S}}_J}\vartheta_{\mathbf{S}}\right] = \mathop{\mathbf{E}}_{\boldsymbol{\mathcal{S}}}\left[\frac{1}{|\boldsymbol{\mathcal{S}}_J|}\sum_{\mathbf{S}_i\in\boldsymbol{\mathcal{S}}_J}\mathop{\mathbf{E}}_{\boldsymbol{x}^i,\boldsymbol{y}^i}[\vartheta_{\mathbf{S}_i}]\right] \\ &= \sum_{S\subseteq J}\mathbf{Pr}[S\in\boldsymbol{\mathcal{S}}]\cdot\mathbf{Inf}_f(\phi_{\mathcal{I}}(S)) = \mathop{\mathbf{E}}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))]. \end{aligned} \tag{3.2}$$

Consider a set J with $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] > \frac{\rho\varepsilon}{3}$. By Equation (3.2), $\mathbf{E}[\boldsymbol{\nu}_J^\rho] > \frac{\rho\varepsilon}{3}$. Therefore, by a Chernoff bound, and since for every J, $|\boldsymbol{\mathcal{S}}_J| \geq m' = \frac{Ck\log\ell}{2\gamma^2\varepsilon\rho}$,

$$\begin{aligned}\mathbf{Pr}\left[\left|\boldsymbol{\nu}_J^\rho - \mathop{\mathbf{E}}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))]\right| > \gamma\mathop{\mathbf{E}}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))]\right] &\leq 2e^{-\frac{|\boldsymbol{\mathcal{S}}_J|\gamma^2\cdot\mathbf{E}_{\mathbf{S}\sim_\rho J}\left[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))\right]}{3}} \\ &\leq 2e^{-\frac{m'\gamma^2\varepsilon\rho}{9}} < 2^{-4k\log\ell},\end{aligned}$$

again for a suitable choice of the constant $C \geq 1$. By taking a union bound over all subsets $J \in \binom{[\ell]}{\ell-k}$, we get that, with probability at least $1-o(1)$, for every J such that $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] > \frac{\rho\varepsilon}{3}$, it holds that $\boldsymbol{\nu}_J^\rho \in (1\pm\gamma)\cdot\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))]$.

Now consider a set $J \subseteq [\ell]$ such that $|J| > \ell - k$ and $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] \leq \frac{\rho\varepsilon}{4}$. By a Chernoff bound:

$$\mathbf{Pr}\left[\boldsymbol{\nu}_J^\rho > (1+\gamma)\frac{\rho\varepsilon}{4}\right] \leq e^{-\frac{\gamma^2}{3}\frac{\rho\varepsilon}{4}|\boldsymbol{\mathcal{S}}_J|} \leq e^{-\frac{\gamma^2\rho\varepsilon}{12}m'} < 2^{-4k\log\ell}.$$

The claim follows by taking a union bound over all subsets $J \in \binom{[\ell]}{\ell-k}$ for which $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] \leq \rho\varepsilon/4$. Overall, the conclusions above hold with probability at least $1 - o(1)$, as stated. ■

3.2.3 Tradeoff between tolerance and query complexity

We now describe how the algorithm from the previous section lets us easily derive the tolerant tester of Theorem 1.3.1.

Algorithm 3 ρ-Tolerant Junta Tester (ε, ρ, k)

```
1: Create a random partition $\boldsymbol{\mathcal{I}}$ of $\ell = 24k^2$ parts by uniformly and independently assigning
   each coordinate to a part.
2: Run Algorithm 2 with the partition $\boldsymbol{\mathcal{I}}$, $\ell = 24k^2$ and $\gamma = 1/8$.
3: if there is a set $J \subset [\ell]$ of size $\ell - k$ such that $\boldsymbol{\nu}_J^\rho \leq \frac{9\rho\varepsilon}{32}$ then
4:     return accept .
5: end if
6: return reject .
```

Proof of Theorem 1.3.1: Given Proposition 3.1.5 it is sufficient to consider a partition $\boldsymbol{\mathcal{I}}$ of size $\ell = 24k^2$ and show that Algorithm 3 distinguishes with probability at least 5/6 between the following two cases.

1. f $\frac{\rho\varepsilon}{8}$-approximates being a k-part junta with respect to $\boldsymbol{\mathcal{I}}$;
2. f $\frac{\varepsilon}{2}$-violates being a k-part junta with respect to $\boldsymbol{\mathcal{I}}$

Suppose first that f $\frac{\rho\varepsilon}{8}$-approximates being a k-part junta with respect to $\boldsymbol{\mathcal{I}}$. Then by Definition 3.1.2, there exists a set $J \in \binom{[\ell]}{\ell-k}$ such that $\mathbf{Inf}_f(\phi(J)) \leq \frac{\rho\varepsilon}{4}$. By Lemma 3.2.3, $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] \leq \frac{\rho\varepsilon}{4}$, and by Lemma 3.2.4, we have that with probability at least $1 - o(1)$, the estimate ν_J^ρ is at most $(1 + 1/8)\frac{\varepsilon\rho}{4} \leq \frac{9\varepsilon\rho}{32}$. Therefore, Algorithm 3 will return accept when considering J.

Consider now the case where f $\frac{\varepsilon}{2}$-violates being a k-part junta with respect to $\boldsymbol{\mathcal{I}}$. Hence, by Definition 3.1.2, every set $J \in \binom{[\ell]}{\ell-k}$ is such that $\mathbf{Inf}_f(\phi(J)) > \varepsilon$, and by Lemma 3.2.3, we have that $\mathbf{E}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] \geq \frac{\rho}{3}\mathbf{Inf}_f(\phi(J)) > \frac{\rho\varepsilon}{3}$. Therefore, by Lemma 3.2.4, with probability at least $1 - o(1)$, for every $J \in \binom{[\ell]}{\ell-k}$

$$\boldsymbol{\nu}_J^\rho \geq \frac{7}{8} \mathop{\mathbf{E}}_{\mathbf{S}\sim_\rho J}[\mathbf{Inf}_f(\phi_{\mathcal{I}}(\mathbf{S}))] > \frac{9\rho\varepsilon}{32} .$$

Thus, with probability at least $1 - o(1)$, Algorithm 3 will reject f. ■

3.3 Polynomial bi-criteria algorithm via submodular minimization

In this section, we show how a mild relaxation of the tolerant testing model, allows dramatic savings in the query complexity. We restate our main theorem proved in this section.

Theorem 1.3.2. *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$ and $\varepsilon \in (0,1)$, satisfies the following.*

- *If f is $\varepsilon/10$-close to some k-junta, then the algorithm accepts with high constant probability.*
- *If f is ε-far from every $2k$-junta, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $\mathrm{poly}(k, \frac{1}{\varepsilon})$.

In order to describe the algorithm referred to in Theorem 1.3.2, it will be useful to introduce the following function. For a Boolean function f and a partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$, we let $h\colon 2^{[\ell]} \to [0,1]$ be defined as $h(J) \overset{\text{def}}{=} \mathbf{Inf}_f(\phi_{\mathcal{I}}(J))$. The starting point of our approach is the observation that the exhaustive search algorithm described previously can be seen as performing a brute-force minimization of h, under a cardinality constraint. Indeed, it effectively goes over all sets $J \subseteq [\ell]$ of size $\ell - k$, estimates $h(J)$, and accepts if there exists a set J for which the estimated value is sufficiently small. With this view, it is natural to ask whether this minimization can be performed more efficiently, by exploiting the fact that by the diminishing marginal property of the influence, h is submodular (see Fact 2.2.2 item 2). That is, for every two sets $J_1 \subseteq J_2$ and variable $i \notin J_2$, it holds that $h(J_1 \cup \{i\}) - h(J_1) \geq h(J_2 \cup \{i\}) - h(J_2)$. While it is possible to find the minimum value of a submodular function in polynomial time given query access to the function, if a cardinality constraint is introduced, then even finding an approximate minimum is hard [SF11]. In light of the hardness of the problem, we design an algorithm for the following *bi-criteria relaxation*. Given oracle access to a non-negative submodular function $h\colon 2^{[\ell]} \to \mathbb{R}$ and input parameters $\varepsilon \in (0,1)$ and $k \in \mathbb{N}$, the algorithm distinguishes between the following two cases:

- There exists a set J such that $|J| \geq \ell - k$ and $h(J) \leq \varepsilon$;
- For every set J such that $|J| \geq \ell - 2k$, $h(J) > 2\varepsilon$.

Moreover, the algorithm can be adapted to the case where it is only granted access to an approximate oracle for h (for a precise statement, see Theorem 3.3.3). This is critical in our setting, since $h(J) = \mathbf{Inf}_f(\phi_{\mathcal{I}}(J))$, and we can only estimate the influence of sets of variables.

3.3.1 Approximate submodular minimization under a cardinality constraint

In this section we show how a certain bi-criteria approximate version of submodular minimization with a cardinality constraint can be reduced to approximate submodular minimization with no cardinality constraint. This reduction holds even when given *approximate oracle access* to the submodular function, and is meaningful when the cardinality constraint is sufficiently large. Precise details follow.

Definition 3.3.1 (Approximate oracle)**.** Let $h\colon 2^{[\ell]} \to \mathbb{R}$ be a function. An *approximate oracle* for h, denoted $\mathcal{O}^{\pm}_{h}$, is a randomized algorithm that, for any input $J \subseteq [\ell]$ and parameters $\tau, \delta \in (0,1)$, returns a value $\tilde{h}(J)$ such that $|\tilde{h}(J) - h(J)| \leq \tau$ with probability at least $1 - \delta$.

Definition 3.3.2 (Approximate submodular minimization algorithm)**.** Let $h\colon 2^{[\ell]} \to \mathbb{R}$ be a non-negative submodular function and let $\mathcal{O}^{\pm}_{h}$ denote an approximate oracle for h. An *approximate submodular function minimization* algorithm (ASFM) is an algorithm that, when given access to $\mathcal{O}^{\pm}_{h}$ and called with input parameters ξ and δ, returns a value $\boldsymbol{\nu}$ such that $|\boldsymbol{\nu} - \min_{J \subseteq [\ell]}\{h(J)\}| \leq \xi$ with probability at least $1 - \delta$.

In Lemma 3.3.10 in Section 3.3.2 we establish the existence of such an ASFM algorithm whose running time is polynomial in ℓ, logarithmic in the maximal value of the function and linear in the running time of the approximate oracle. We next present an algorithm for approximate submodular minimization under a cardinality constraint.

Algorithm 4 Approximate Submodular Minimization under a Cardinality Constraint($\mathcal{O}^{\pm}_{h}, \varepsilon, \delta, \xi, k$)

```
1: Let $h'(J) = h(J) - \frac{\varepsilon}{k}|J|$ so that for every $\tau', \delta'$ $\mathcal{O}^{\pm}_{h'}(J, \tau', \delta') = \mathcal{O}^{\pm}_{h}(J, \tau', \delta') - \frac{\varepsilon}{k}|J|$.
2: Let $\boldsymbol{\nu}$ be the returned value from invoking an ASFM algorithm with access to $\mathcal{O}^{\pm}_{h'}$ and parameters $\xi$ and $\delta$.
3: Accept if and only if $\boldsymbol{\nu} \leq (2 - \frac{\ell}{k}) \cdot \varepsilon + \xi$.
```

Theorem 3.3.3. *For a non-negative submodular function h, Algorithm 4 satisfies the following conditions:*

1. *If there exists a set $J \subseteq [\ell]$ such that $|J| \geq \ell - k$ and $h(J) \leq \varepsilon$, then the algorithm accepts with probability at least $1 - \delta$.*

2. *If for every set $J \subseteq [\ell]$ such that $|J| \geq \ell - 2(1 + \frac{\xi}{\varepsilon})k$, we have $h(J) > 2\varepsilon + 2\xi$, then the algorithm rejects with probability at least $1 - \delta$.*

Moreover, the second item can be strengthened so that it holds for functions h that satisfy the following: (i) for every set $J \subseteq [\ell]$ such that $|J| \geq \ell - k$, $h(J) > 2\varepsilon + 2\xi$ and (ii) for every set $J \subseteq [\ell]$ such that $|J| \geq \ell - 2(1 + \frac{\xi}{\varepsilon})k$, $h(J) > \varepsilon + 2\xi$.

Proof: First of all, note that the function h' defined in Step 1 is indeed submodular, as it is the sum of the submodular function h and a modular function (scalar multiple of the cardinality function). By Definition 3.3.2, with probability at least $1 - \delta$ the value $\boldsymbol{\nu}$ defined in Step 2 of the algorithm thus satisfies

$$|\boldsymbol{\nu} - \min_{J \subseteq [\ell]}\{h'(J)\}| \leq \xi \ . \tag{3.3}$$

We start with proving the first item in Theorem 3.3.3. If there exists a set $J^* \subseteq [\ell]$ such that $|J^*| \geq \ell - k$ and $h(J^*) \leq \varepsilon$, then

$$\min_{J \subseteq [\ell]}\{h'(J)\} \leq h'(J^*) \leq \varepsilon - \frac{\varepsilon}{k}(\ell - k) = \left(2 - \frac{\ell}{k}\right) \cdot \varepsilon,$$

and therefore, by Equation (3.3), with probability at least $1 - \delta$,

$$\boldsymbol{\nu} \leq \min_{J \subseteq [\ell]}\{h'(J)\} + \xi \leq \left(2 - \frac{\ell}{k}\right) \cdot \varepsilon + \xi,$$

and the algorithm accepts.

We divide the analysis of the second item in the theorem into two cases depending on $|J|$.

- For sets J such that $|J| \geq \ell - 2(1 + \frac{\xi}{\varepsilon})k$,

$$h'(J) > 2\varepsilon + 2\xi - \frac{\varepsilon}{k} \cdot \ell = \left(2 - \frac{\ell}{k}\right)\varepsilon + 2\xi,$$

  where in the first inequality we used the fact that for all sets J, $|J| \leq \ell$.

- For sets J such that $|J| < \ell - 2(1 + \frac{\xi}{\varepsilon})k$, it holds that

$$h'(J) > 0 - \frac{\varepsilon}{k} \cdot \left(\ell - 2\left(1 + \frac{\xi}{\varepsilon}\right)k\right) = \left(2 - \frac{\ell}{k}\right) \cdot \varepsilon + 2\xi,$$

  where in the first inequality we used the fact that for all sets J, $h(J) \geq 0$.

Hence,

$$\min_{J \subseteq [\ell]}\{h'(J)\} > \left(2 - \frac{\ell}{k}\right) \cdot \varepsilon + 2\xi,$$

and by Equation (3.3), with probability at least $1 - \delta$, it holds that

$$\boldsymbol{\nu} \geq \min_{J \subseteq [\ell]}\{h'(J)\} - \xi > \left(2 - \frac{\ell}{k}\right) \cdot \varepsilon + \xi,$$

and the algorithm rejects. ■

3.3.2 Approximate submodular function minimization

In this section we use results from [LSW15] to obtain an approximate submodular minimization algorithm, as defined in Definition 3.3.2.[1] This is done in three steps: (1) We use the known fact that the problem of finding the minimum of a submodular function g can be reduced to finding the minimum of the Lovász extension for that function, denoted $\mathcal{L}_g$. (2) We then extend the results of [LSW15] (and specifically of Theorem 61) and provide a noisy separation oracle for $\mathcal{L}_g$ when only given approximate oracle access to the function g. (3) Finally, we apply Theorem 42 from [LSW15], which provides an algorithm that, when given access to a separation oracle for a function, returns an approximation to that function's minimum value. Note that in this section, we analyze the *time* complexity of our algorithms for submodular function minimization; we will later, in the following sections, switch back to query complexity when applying them to our testing problem, using running time as an upper bound for query complexity.

We start with the following definition of the Lovász extension of a submodular function.

Definition 3.3.4 (Lovász Extension)**.** Given a submodular function $g\colon 2^{[\ell]} \to \mathbb{R}$, the *Lovász extension* of g is a function $\mathcal{L}_g\colon [0,1]^\ell \to \mathbb{R}$ which is defined for all $x \in [0,1]^\ell$ by

$$\mathcal{L}_g(x) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\boldsymbol{t}\sim[0,1]}[g(\{\, i \,:\, x_i \geq \boldsymbol{t} \,\})] \ ,$$

where $\boldsymbol{t} \sim [0,1]$ denotes that $\boldsymbol{t}$ is drawn uniformly at random from $[0,1]$.

The following theorem is standard in combinatorial optimization (see e.g. [Bac13] and [GLS12, Sch02]) and provides useful properties of the Lovász extension.

Theorem 3.3.5. *The Lovász extension $\mathcal{L}_g$ of a submodular function $g\colon 2^{[\ell]} \to \mathbb{R}$ satisfies the following properties.*

1. *$\mathcal{L}_g$ is convex and $\min_{x\in[0,1]^\ell}\{\mathcal{L}_g(x)\} = \min_{S\subseteq[\ell]}\{g(S)\}$.*
2. *If $x_1 \geq \ldots \geq x_\ell$, then*
$$\mathcal{L}_g(x) = \sum_{i=1}^{\ell} \big(g([i]) - g([i-1])\big)x_i \ .$$

By the first item of Theorem 3.3.5, in order to approximate the minimum value of a submodular function g, it suffices to approximate the minimum of its Lovász extension. As discussed at the start of the section, this is done by providing a *separation oracle* for $\mathcal{L}_g$.

Definition 3.3.6 (Separation Oracle)**.** Let h be a convex function over $\mathbb{R}^\ell$ and let Ω be a convex set in $\mathbb{R}^\ell$. A *separation oracle for h* with respect to Ω is an algorithm that for an input $x \in \Omega$ and parameters $\eta, \gamma \geq 0$ satisfies the follows. It either asserts that $h(x) \leq \min_{y\in\Omega}\{h(y)\} + \eta$ or it outputs a halfspace $H \stackrel{\text{def}}{=} \{z : a^T z \leq a^T x + c\}$ such that

$$\{\, y \in \Omega \,:\, h(y) \leq h(x) \,\} \subset H \ ,$$

where $a \in [0,1]^\ell$, $a \neq 0$, and $c \leq \gamma\|a\|_2$.

[1]We remark that more efficient algorithms, such as the convex optimization algorithm (given noisy access) of [BLNR15] applied to the Lovász extension, might also apply. Such algorithms would yield a better polynomial for the query complexity of our problem; however, they typically require stronger requirements on the noise (e.g., subgaussian distribution for [BLNR15]), where our implementation is robust to adversarial noise. This explicit robustness to noise is the reason why we rely on the results of [LSW15], which – to the best of our knowledge – is the first algorithm to provide the robustness guarantees we need.

In Theorem 61 in [LSW15] it is shown how to define a separation oracle for a function g when given *exact* query access to g; we adapt the proof to the case where one is only granted access to an approximate oracle for g, and the resulting procedure has small failure probability.

Algorithm 5 Separation Oracle $(\mathcal{O}_g^{\pm}, \bar{x}, \eta, \gamma, \delta)$

```
1: Assume without loss of generality that $\bar{x}_1 \geq \bar{x}_2 \geq \ldots \geq \bar{x}_\ell$ (otherwise re-index the coordinates).
2: Let $\tau = \min\{\eta/4\ell, \gamma/2\ell\}$.
3: For each $i \in [\ell]$, let $\tilde{\boldsymbol{g}}([i])$ be the returned value from invoking $\mathcal{O}_g^{\pm}$ on the set $[i]$ with parameters $\frac{\tau^2}{2}$ and $\frac{\delta}{\ell}$.
4: Define $\tilde{\boldsymbol{a}} \in \mathbb{R}^\ell$ by $\tilde{\boldsymbol{a}}_i \overset{\text{def}}{=} \tilde{\boldsymbol{g}}([i]) - \tilde{\boldsymbol{g}}([i-1])$ for each $i \in [\ell]$.
5: Let $\tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) \overset{\text{def}}{=} \tilde{\boldsymbol{a}}^T \bar{x}$.
6: if for every $i \in [\ell]$, $|\tilde{\boldsymbol{a}}_i| < \tau$ then
7:     return $\bar{x}$ (which satisfies "$\mathcal{L}_g(\bar{x}) \leq \min_{y \in [0,1]^\ell}\{\mathcal{L}_g(y)\} + \eta$").
8: else
9:     return the halfspace $H = \{z : \tilde{\boldsymbol{a}}^T z \leq \tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) + 2\tau\ell\|\tilde{\boldsymbol{a}}\|_2\}$ .
10: end if
```

Lemma 3.3.7. *Let $g \colon 2^{[\ell]} \to \mathbb{R}$ be a convex function, and let $\Phi_g(\cdot,\cdot)$ denote the running time of the approximate oracle $\mathcal{O}_g^{\pm}$ for g. For every $x \in [0,1]^\ell$, $\eta, \gamma, \delta \in (0,1)$, with probability at least $1-\delta$, Algorithm 5 satisfies the guarantees of a separation oracle for $\mathcal{L}_g$ (with respect to $[0,1]^\ell$). The algorithm makes ℓ queries to $\mathcal{O}_g^{\pm}$ with parameters $\tau^2/2$ and δ/ℓ, where $\tau = \min\{\eta/4\ell, \gamma/2\ell\}$, and its running time is $\ell \cdot \Big(\Phi_g(\frac{\tau^2}{2}, \delta/\ell) + \log \ell\Big)$.*

In order to prove the above lemma we will use the following theorem from [LSW15].

Theorem 3.3.8 ([LSW15, Theorem 61], restated). *Let $g \colon 2^{[\ell]} \to \mathbb{R}$ be a submodular function. For every $x \in [0,1]^\ell$,*

$$\sum_{i=1}^{\ell} \big(g([i]) - g([i-1])\big) x_i \leq \mathcal{L}_g(x) \ .$$

Proof of Lemma 3.3.7: For every $i \in [\ell]$, let $a_i \overset{\text{def}}{=} g([i]) - g([i-1])$, and note that by a union bound over all $i \in [\ell]$, we have that $\max_{i \in [\ell]}\{|g([i]) - \tilde{\boldsymbol{g}}([i])|\} \leq \tau^2/2$, with probability at least $1-\delta$. We henceforth condition on this, and observe that this implies that, for any $y \in [0,1]^\ell$,

$$|\tilde{\boldsymbol{a}}^T y - a^T y| \leq 2\ell \cdot \frac{\tau^2}{2} = \ell\tau^2 \ . \tag{3.4}$$

We next consider two cases. Assume first that there exists an index $i \in [\ell]$ such that $|\tilde{\boldsymbol{a}}_i| \geq \tau$. That is, assume that the condition in Step 6 of the algorithm does not hold. Then we prove that for every $y \in [0,1]^\ell$ such that $\mathcal{L}_g(y) \leq \mathcal{L}_g(\bar{x})$ it holds that $y \in H$, where H is the halfspace defined in Step 9 of the algorithm.

By Theorem 3.3.8, we have that $\sum_{i=1}^{\ell} a_i \cdot y_i \leq \mathcal{L}_g(y)$ for every $y \in [0,1]^\ell$. Since $\mathcal{L}_g(y) \leq \mathcal{L}_g(\bar{x})$, we get that

$$\tilde{\boldsymbol{a}}^T y \leq a^T y + \ell\tau^2 \leq \mathcal{L}_g(y) + \ell\tau^2 \leq \mathcal{L}_g(\bar{x}) + \ell\tau^2 \ . \tag{3.5}$$

By Theorem 3.3.5, together with the assumption that the coordinates of $\bar{x}$ are sorted,

$$\mathcal{L}_g(\bar{x}) = \sum_{i=1}^{\ell} a_i \cdot \bar{x}_i \leq \sum_{i=1}^{\ell} \tilde{\boldsymbol{a}}_i \cdot \bar{x}_i + \ell\tau^2 = \tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) + \ell\tau^2. \tag{3.6}$$

Combining Equation (3.5) and Equation (3.6), and since there exists an i such that $|\tilde{\boldsymbol{a}}_i| \geq \tau$,

$$\tilde{\boldsymbol{a}}^T y \leq \tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) + 2\ell\tau^2 \leq \tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) + 2\ell\tau\|\tilde{\boldsymbol{a}}\|_2 \,.$$

This implies that y is in H and that for $c = 2\tau\ell\|\tilde{\boldsymbol{a}}\|_2$ and $\gamma \leq 2\tau\ell$, H fulfills the requirements stated in Definition 3.3.6.

Now consider the case that $|\tilde{\boldsymbol{a}}_i| \leq \tau$ for all $i \in [\ell]$. It follows that for any $y \in [0,1]^\ell$, $-\ell\tau \leq \tilde{\boldsymbol{a}}^T y \leq \ell\tau$. In particular, we have that $-\ell\tau \leq \tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) \leq \ell\tau$, which implies that for every $y \in [0,1]^\ell$,

$$\tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) - 2\ell\tau \;\leq\; -\ell\tau \;\leq\; \tilde{\boldsymbol{a}}^T y \,.$$

Therefore, for every $y \in [0,1]^\ell$ we get

$$\tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) - 3\ell\tau \;\leq\; \tilde{\boldsymbol{a}}^T y - \ell\tau \;\leq\; a^T y \;\leq\; \mathcal{L}_g(y) \,,$$

where the second inequality follows from Equation (3.4), and the last inequality follows from Theorem 3.3.8. Hence, if we let $x^* = \arg\min_x\{\mathcal{L}_g(x)\}$, we have that

$$\tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) \;\leq\; \mathcal{L}_g(x^*) + 3\ell\tau \,.$$

By Equation (3.6) we have that $\mathcal{L}_g(\bar{x}) \leq \tilde{\boldsymbol{\mathcal{L}}}_g(\bar{x}) + \ell\tau^2$. Hence,

$$\mathcal{L}_g(\bar{x}) \leq \mathcal{L}_g(x^*) + 3\ell\tau + \ell\tau^2 \leq \mathcal{L}_g(x^*) + 4\ell\tau \,,$$

and since by the setting of τ in Step 2 of the algorithm, $\tau \leq \eta/4\ell$, we get that $\bar{x}$ satisfies

$$\mathcal{L}_g(\bar{x}) \;\leq\; \min_{y\in[0,1]^\ell}\{\mathcal{L}_g(y)\} + \eta.$$

Therefore, with probability at least $1-\delta$ the algorithm satisfies the conditions of a separation oracle with parameters η and γ.

The algorithm performs ℓ queries to the approximate oracle for g with parameters $\tau^2/2$ and δ/ℓ, where $\tau = \min\{\eta/4\ell, \gamma/2\ell\}$. Hence, the running time of the algorithm is $\ell\cdot\Phi_g(\frac{\tau^2}{2}, \frac{\delta}{\ell}) + \ell\log\ell$, as it also sorts the coordinates of $\bar{x}$ (in order to re-index the coordinates). ■

We can now use the separation oracle for $\mathcal{L}_g$ and apply the following theorem to get an approximate minimum of $\mathcal{L}_g$, which is also an approximate minimum of g.

Theorem 3.3.9 ([LSW15, Theorem 42], restated)**.** *Let h be a convex function on $\mathbb{R}^\ell$ and let Ω be a convex set with constant min-width*[2] *that contains a minimizer of h. Suppose we have a separation oracle for h and that Ω is contained inside $B_\infty(R) \overset{\text{def}}{=} \{\, x : \|x\|_\infty \leq R \,\}$, where $R > 0$ is a constant. Then there is an algorithm, which for any $0 < \alpha < 1$ and $\eta > 0$ outputs $x \in \mathbb{R}^\ell$ such that*

$$h(x) - \min_{y\in\Omega}\{h(y)\} \leq \eta + \alpha \cdot \left(\max_{y\in\Omega}\{h(y)\} - \min_{y\in\Omega}\{h(y)\}\right) \,.$$

In expectation, the algorithm performs $O\left(\ell \cdot \log\left(\frac{\ell}{\alpha}\right)\right)$ calls to Algorithm 5, and has expected running time of

$$O\left(\ell \cdot \mathrm{SO}(\eta,\gamma)\log\left(\frac{\ell}{\alpha}\right) + \ell^3 \log^{O(1)}\left(\frac{\ell}{\alpha}\right)\right) \,,$$

where $\gamma = \Theta\left(\frac{\alpha}{\ell^{3/2}}\right)$ and $\mathrm{SO}(\eta,\gamma)$ denotes the running time of the separation oracle when invoked with parameters η and γ.

[2]For a compact set $K \subseteq \mathbb{R}^\ell$, the min-width is defined as $\min_{a\in\mathbb{R}^\ell:\,\|a\|_2=1}\max_{x,y\in K}\langle a, x-y\rangle$. [LSW15, Definition 41]. In particular, it is not hard to see that the set $K = [0,1]^\ell \subseteq B_\infty(1)$ has unit min-width.

Lemma 3.3.10. *Let $g\colon 2^{[\ell]} \to \mathbb{R}$ be a submodular function. There exists an algorithm that, when given access to $\mathcal{O}_g^{\pm}$, and for input parameters $\xi, \delta \in (0,1)$, returns with probability at least $9/10 - 2\delta$ a value $\boldsymbol{\nu} \in \mathbb{R}$ such that $\boldsymbol{\nu} \le \min_{S\subseteq[\ell]}\{g(S)\} + \xi$.*

The algorithm performs $O\left(\ell^2 \log\left(\frac{\ell M}{\xi}\right)\right)$ calls to $\mathcal{O}_g^{\pm}$ with parameters $\frac{\xi^2}{128\ell^5 M^2}$ and $\frac{\delta}{C\ell^2\log\left(\frac{\ell M}{\xi}\right)}$, where $M \overset{\text{def}}{=} \max\left\{2\max_{S\subseteq[\ell]}\{|g(S)|\}, \xi/2\right\}$ and $C > 0$ is an absolute constant. The running time of the algorithm is

$$O\left(\ell^2 \cdot \Phi_g\left(\frac{\xi^2}{128\ell^5 M^2}, \frac{\delta}{C\ell^2 \log\frac{\ell M}{\xi}}\right)\log\frac{\ell M}{\xi} + \ell^3 \log^{O(1)}\frac{\ell M}{\xi}\right) ,$$

where Φ_g is the running time of $\mathcal{O}_g^{\pm}$.

Proof: We refer to the algorithm from Theorem 3.3.9 as *the minimization algorithm* and apply it to $\mathcal{L}_g$, with Algorithm 5 as a separation oracle. Once the minimization algorithm returns a point $x \in [0,1]^\ell$, we return the value $\boldsymbol{\nu} = \mathcal{O}_{\mathcal{L}_g}^{\pm}(x, \xi/4, \delta)$.

Let $M' \overset{\text{def}}{=} 2\max_{S\subseteq[\ell]}\{|g(S)|\}$, and recall that $\mathcal{L}_g(x) = \mathbf{E}_{\boldsymbol{t}\sim[0,1]}\left[g(\{\, i : x_i \ge \boldsymbol{t} \,\})\right]$. Hence, $\max_{x\in[0,1]^\ell}\{\mathcal{L}_g(x)\} - \min_{x\in[0,1]^\ell}\{\mathcal{L}_g(x)\} \le M'$. Setting $\alpha = \xi/(2M)$ and $\eta = \xi/4$ ensures that $0 < \alpha < 1$ and that

$$\eta + \alpha \cdot \left(\max_{x\in[0,1]^\ell}\{\mathcal{L}_g(x)\} - \min_{x\in[0,1]^\ell}\{\mathcal{L}_g(x)\}\right) \le \eta + \alpha M' \le 3\xi/4 . \tag{3.7}$$

The minimization algorithm invokes the separation oracle $C_1 \cdot \ell\log(\ell/\alpha) = C_1 \cdot \ell\log(\ell M/\xi)$ times in expectation, for some constant C_1. We convert this to a worst-case bound as follows. If at some point the number of calls to the separation oracle exceeds $20C_1 \cdot \ell\log(\ell M/\xi)$, then we halt and return fail . Similarly, the algorithm runs in expected time

$$T \overset{\text{def}}{=} C_2 \cdot \left(\ell \cdot \mathrm{SO}(\eta,\gamma)\log\left(\frac{\ell}{\alpha}\right) + \ell^3 \log^{C_3}\left(\frac{\ell}{\alpha}\right)\right)$$

for some absolute constants $C_2, C_3 > 0$. If at some point the running time exceeds $20T$, then we also halt and return fail . By Markov's inequality, both events each happen with probability at most $1/20$, and therefore by a union bound our algorithm halts and outputs fail with probability at most $1/10$. Hence, every time the minimization algorithm calls the separation oracle with parameters η and γ we invoke Algorithm 5 with parameters η, γ and $\delta' = \frac{\delta}{10C_1\ell\log\left(\frac{\ell M}{\xi}\right)}$. Therefore, by Lemma 3.3.7, with probability at least $1 - 1/10 - \delta$ all the calls to Algorithm 5 satisfy the guarantee of a separation oracle for $\mathcal{L}_g$ with parameters η and γ. By Theorem 3.3.9 and Equation (3.7), with probability at least $9/10 - \delta$ the minimization algorithm returns a point x such that

$$\mathcal{L}_g(x) - \min_{y\in[0,1]^\ell}\{\mathcal{L}_g(y)\} \le \eta + \alpha \cdot \left(\max_{y\in[0,1]^\ell}\{\mathcal{L}_g(y)\} - \min_{y\in[0,1]^\ell}\{\mathcal{L}_g(y)\}\right) \le \frac{3\xi}{4} ,$$

and with probability at least $9/10 - 2\delta$ the value $\boldsymbol{\nu}$ satisfies

$$\boldsymbol{\nu} \le \min_{y\in[0,1]^\ell}\{\mathcal{L}_g(y)\} + \xi ,$$

as desired.

By the above settings and by Lemma 3.3.7 we get that $\tau = \frac{\xi}{8\ell^{5/2}M}$ so the running time of each invocation of the separation oracle (recall that each such invocation involves ℓ calls to $\mathcal{O}_g^{\pm}$) is

$$\ell \cdot \Phi_g\left(\frac{\tau^2}{2}, \frac{\delta'}{\ell}\right) + \ell \log \ell = \ell\left(\Phi_g\left(\frac{\xi^2}{128\ell^5 M^2}, \frac{\delta}{10C_1\ell^2 \log\frac{\ell M}{\xi}}\right) + \log \ell\right) .$$

Since the evaluation of $\boldsymbol{\nu}$ in the final step is negligible in the running time of the minimization algorithm, we get that the overall time complexity is

$$O\left(\ell^2 \cdot \Phi_g\left(\frac{\xi^2}{128\ell^5 M^2}, \frac{\delta}{10C_1\ell^2 \log\frac{\ell M}{\xi}}\right)\log\frac{\ell M}{\xi} + \ell^2 \log \ell \cdot \log\frac{\ell M}{\xi} + \ell^3 \log^{O(1)}\frac{\ell M}{\xi}\right) ,$$

which gives the stated bound, recalling that $\ell^2 \log \ell \cdot \log\frac{\ell M}{\xi} = O(\ell^2 \log^2\frac{\ell M}{\xi}) = O(\ell^3 \log^{O(1)}\frac{\ell M}{\xi})$. ■

Lemma 3.3.11. *There exists an algorithm that, when given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and a partition $\mathcal{I} = \{I_1, \dots, I_\ell\}$ of $[n]$ into ℓ parts, as well as input parameters $k \in \mathbb{N}, \varepsilon, \xi \in (0,1)$, satisfies the following. It has query complexity $\tilde{O}\left(\max\left(\frac{\ell^{12}}{\xi^4}, \frac{\ell^{16}\varepsilon^4}{k^4\xi^4}\right)\right)$, and distinguishes with probability at least $5/6$ between the following two cases:*

1. *There exists a set $S \subseteq [\ell]$ such that $|S| \geq \ell - k$ and $h(S) \leq \varepsilon$.*
2. *For every set S such that $|S| \geq \ell - 2(1+\frac{\xi}{\varepsilon})k$, $h(S) > 2\varepsilon + 2\xi$,*

where $h\colon 2^{[\ell]} \to \mathbb{R}$ is defined as $h(S) \overset{\text{def}}{=} \mathbf{Inf}_f(\phi_{\mathcal{I}}(S))$.
Moreover, the second item can be strengthened so that it holds for functions f that satisfy the following: (i) for every set S such that $|S| \geq \ell - k$, $h(S) > 2\varepsilon + 2\xi$ and (ii) for every set S such that $|S| \geq \ell - 2(1+\frac{\xi}{\varepsilon})k$, $h(S) > \varepsilon + 2\xi$.

Proof: We apply Lemma 3.3.10 to $h'\colon 2^{[\ell]} \to \mathbb{R}$, defined as in Algorithm 4 by $h'(S) \overset{\text{def}}{=} h(S) - \frac{\varepsilon}{k}|S|$, with ξ, $M \overset{\text{def}}{=} \max\left(2\max(2, \frac{\varepsilon\ell}{k}), \xi/2\right) = 4\max(1, \frac{\varepsilon\ell}{2k})$, and $\delta \overset{\text{def}}{=} \frac{1}{30}$. In order to do so, we need to simulate an approximate oracle for h' (as defined in Definition 3.3.1). Since $h(S) = \mathbf{Inf}_f(\phi_{\mathcal{I}}(S))$, in order to estimate $h'(S)$ within an additive approximation of τ' with probability at least $1-\delta'$, it is sufficient to estimate $\mathbf{Inf}_f(\phi_{\mathcal{I}}(S)) \in [0,1]$ within an additive approximation of τ' with probability at least $1-\delta'$ (indeed, the additional term $\frac{\varepsilon}{k}|S|$ can be computed exactly). By Chernoff bounds, this can be done with $\Phi_h(\tau', \delta') = O(\frac{1}{\tau'^2}\log\frac{1}{\delta'})$ queries to f.

This yields an approximate oracle $\mathcal{O}_h^{\pm}$, and therefore $\mathcal{O}_{h'}^{\pm}$ which can be provided to the algorithm of Theorem 3.3.3 (resulting in a success probability at least $9/10 - 2\delta = 5/6$). The resulting query complexity is

$$O\left(\ell^2 \cdot \Phi_h\left(\frac{\xi^2}{\ell^5 M^2}, \frac{1}{10C_1\ell^2 \log\frac{\ell M}{\xi}}\right)\log\frac{\ell M}{\xi} + \ell^2 \log \ell + \ell^3 \log^{O(1)}\frac{\ell M}{\xi}\right)$$

which, given the above expression for Φ_h, can be bounded as follows.

- If $\varepsilon < \frac{2k}{\ell}$, so that $M = 4$, this simplifies as
$$O\left(\frac{\ell^{12}}{\xi^4}\log^2\frac{\ell}{\xi}\right).$$
- If $\varepsilon \geq \frac{2k}{\ell}$, which implies that $M = \frac{2\varepsilon\ell}{k}$, this becomes
$$O\left(\frac{\ell^{16}\varepsilon^4}{k^4\xi^4}\log^2\frac{\ell}{\xi}\right).$$

Observing that the function h is indeed a non-negative submodular function (and that h' is also submodular since it is the sum of a submodular function and a modular function) allows us to conclude by Theorem 3.3.3. ■

In particular, setting $\xi = \varepsilon/(4k)$ we get the following:

Lemma 3.3.12. *There exists an algorithm that, given query access to a function* $f\colon \{-1,1\}^n \to \{-1,1\}$, *a fixed partition* $\mathcal{I}$ *of* $[n]$ *into* $\ell = O(k^2)$ *parts, and parameters* $k \geq 1$ *and* $\varepsilon \in (0,1)$, *satisfies the following. The query complexity of the algorithm is* $\tilde{O}\left(\frac{k^{28}}{\varepsilon^4} + k^{32}\right) = \operatorname{poly}(k, 1/\varepsilon)$, *and:*

1. *if* f $\frac{\varepsilon}{2}$*-approximates being a* k*-part junta with respect to* $\mathcal{I}$*, then the algorithm accepts with probability at least* $\frac{5}{6}$*;*
2. *if* f $\frac{5}{4}\varepsilon$*-violates being a* $2k$*-part junta with respect to* $\mathcal{I}$*, then the algorithm rejects with probability at least* $\frac{5}{6}$*.*

Moreover, the second item can be strengthened to "simultaneously $\left(1 + \frac{1}{4k}\right)\varepsilon$*-violates being a* k*-part junta and* $(1 + \frac{1}{2k})\frac{\varepsilon}{2}$*-violates being a* $2k$*-part junta."*

Proof: By applying Lemma 3.3.11 with $\xi = \varepsilon/(4k)$, we get an algorithm that distinguishes between (1) there exists a set $S \subseteq [\ell]$ such that $|S| \geq \ell - k$ and $h(S) \leq \varepsilon$; and (2) either (i) for every set S such that $|S| \geq \ell - k$, $h(S) > 2\left(1 + \frac{1}{4k}\right)\varepsilon$ or (ii) for every set S such that $|S| \geq \ell - (2k + \frac{1}{2})$, $h(S) > \varepsilon(1 + \frac{1}{2k})$. Since $|S|$ is always an integer, the condition $|S| \geq \ell - (2k + \frac{1}{2})$ is equivalent to $|S| \geq \ell - 2k$ in (2)(ii). This implies the guarantees of the lemma, by the correspondence with partition juntas (Definition 3.1.2) and using for simplicity that $1 + \frac{1}{4k} \leq \frac{5}{4}$ as $k \geq 1$. (The claimed query complexity is immediate from Lemma 3.3.11.) ■

Proof of Theorem 1.3.2: The proof follows immediately from Lemma 3.3.12, together with Proposition 3.1.5. With probability at least 5/6, a random partition of the variables into $\ell \stackrel{\text{def}}{=} 192k^2$ parts will have the right guarantees, reducing the problem to distinguishing between $\frac{\varepsilon}{2}$-approximating being a k-part junta vs. $\frac{5}{4}\varepsilon$-violating being a $2k$-part junta (with regard to this random partition). Overall, by a union bound, the result is therefore correct with probability at least 2/3. ■

Chapter 4

Hardness Results for Tolerant Junta Testing

In this chapter we consider the question of whether the tolerance requirement of a junta testing algorithm affects the query complexity. Prior to our work, it was reasonable to believe that the additional tolerance requirement does not result in a significant increase in the algorithm's query complexity. We show that it is not the case for non-adaptive algorithms. We restate the main result proved in this chapter.

Theorem 1.3.3. *For any* $\alpha < 1$*, there exist constants* $0 < \varepsilon_0 < \varepsilon_1 < 1$ *such that for any* $k = k(n) \leq \alpha n$*, any non-adaptive algorithm that distinguish between functions* $f : \{-1, 1\}^n \to \{-1, 1\}$ *that are* ε_0*-close to* k*-junta and functions* ε_1*-far from any* k*-junta must make* $\Omega(k^2/\text{polylog } k)$ *queries.*

Our lower bound combined with the fact that (standard) k-junta testing requires $\widetilde{\Theta}(k^{3/2})$ (see [Bla08, CST$^+$17]) provides a polynomial separation in the query complexity of non-adaptive tolerant testing and non-adaptive (standard) junta testing.

In order to prove our lower bound we will describe a new model of graph property testing, which we call the *rejection sampling model*. For $n \in \mathbb{N}$ and a subset $\mathcal{P}$ of graphs on the vertex set $[n]$, we say a graph G on vertex set $[n]$ has property $\mathcal{P}$ if $G \in \mathcal{P}$ and say G is ε-far from having property $\mathcal{P}$ if all graphs $H \in \mathcal{P}$ differ on at least εn^2 edges[1]. The problem of ε-testing $\mathcal{P}$ with *rejection sampling queries* is the following task:

> Given some $\varepsilon > 0$ and access to an unknown graph $G = ([n], E)$, output "accept" with probability at least $\frac{2}{3}$ if G has property $\mathcal{P}$, and output "reject" with probability at least $\frac{2}{3}$ if G is ε-far from having property $\mathcal{P}$. The access to G is given by the following oracle queries: given a query set $L \subseteq [n]$, the oracle samples an edge $(\boldsymbol{i}, \boldsymbol{j}) \sim E$ uniformly at random and returns $\{\boldsymbol{i}, \boldsymbol{j}\} \cap L$.

We measure the complexity of algorithms with rejection sampling queries by considering the sizes of the queries. The complexity of an algorithm making queries $L_1, \ldots, L_t \subset [n]$ is $\sum_{i=1}^{t} |L_i|$.

The rejection sampling model allows us to study testers which rely on random sampling of edges, while providing the flexibility of making lower-cost queries. This type of query access strikes a delicate balance between simplicity and generality: queries are constrained enough for us to show high lower bounds, and at the same time, the flexibility of making queries allows us to reduce the rejection sampling model to Boolean function testing problems. Specifically, we reduce to tolerant junta testing (and tolerant unateness testing; see Chapter 6.2).

[1]The distance definition can be modified accordingly when one considers bounded degree or sparse graphs.

Our main result in the rejection sampling model is regarding *non-adaptive* algorithms. These algorithms need to fix their queries in advance and are not allowed to depend on answers to previous queries (in the latter case we say that the algorithm is *adaptive*). We show a lower bound on the complexity of testing whether an unknown graph G is bipartite using non-adaptive queries.

Theorem 4.0.1. *There exists a constant $\varepsilon > 0$ such that any non-adaptive ε-tester for bipartiteness in the rejection sampling model has cost $\widetilde{\Omega}(n^2)$.*

More specifically, Theorem 4.0.1 follows from applying Yao's principle to the following lemma.

Lemma 4.0.2. *Let $\mathcal{G}_1$ be the uniform distribution over the union of two disjoint cliques of size $n/2$, and let $\mathcal{G}_2$ be the uniform distribution over complete bipartite graphs with each part of size $n/2$. Any deterministic non-adaptive algorithm that can distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$ with constant probability using rejection sampling queries must have complexity $\widetilde{\Omega}(n^2)$.*

In the next section we formally define the model and the distributions over graphs which will be used throughout this chapter (and Chapter 6.2).

4.1 The Rejection Sampling Model

We describe a new model of graph property testing, which we call the *rejection sampling model.* The rejection sampling model allows us to study testers which rely on random sampling of edges, while providing the flexibility of making lower-cost queries. This type of query access strikes a delicate balance between simplicity and generality: queries are constrained enough for us to show high lower bounds, and at the same time, the flexibility of making queries allows us to reduce the rejection sampling model to Boolean function testing problems. Specifically, we reduce to tolerant junta testing and tolerant unateness testing (see Chapter 6.2).

Definition 4.1.1. Consider two distributions, $\mathcal{G}_1$ and $\mathcal{G}_2$ supported on graphs with vertex set $[n]$. The problem of distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$ with a rejection sampling oracle aims to distinguish between the following two cases with a specific kind of query:

- Cases: We have an unknown graph $\mathbf{G} \sim \mathcal{G}_1$ or $\mathbf{G} \sim \mathcal{G}_2$.
- Rejection Sampling Oracle: Each query is a subset $L \subset [n]$; an oracle samples an edge $(\boldsymbol{j}_1, \boldsymbol{j}_2)$ from $\mathbf{G}$ uniformly at random, and the oracle returns $\boldsymbol{v} = \{\boldsymbol{j}_1, \boldsymbol{j}_2\} \cap L$. The complexity of a query L is given by $|L|$.

We say a non-adaptive algorithm Alg for this problem is a sequence of query sets $L_1, \dots, L_q \subset [n]$, as well as a function Alg: $([n] \cup ([n] \times [n]) \cup \{\emptyset\})^q \to \{\text{“}\mathcal{G}_1\text{”}, \text{“}\mathcal{G}_2\text{”}\}$. The algorithm sends each query to the oracle, and for each query L_i, the oracle responds $\boldsymbol{v}_i \in [n] \cup ([n] \times [n]) \cup \{\emptyset\}$, which is either a single element of $[n]$, an edge in $\mathbf{G}$, or $\emptyset$. The algorithm succeeds if:

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1,\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_q}}[\text{Alg}(\boldsymbol{v}_1, \dots, \boldsymbol{v}_q) \text{ outputs “}\mathcal{G}_1\text{”}] - \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_2,\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_q}}[\text{Alg}(\boldsymbol{v}_1, \dots, \boldsymbol{v}_q) \text{ outputs “}\mathcal{G}_1\text{”}] \geq \frac{1}{3}.$$

The complexity of Alg is measured by the sum of the complexity of the queries, so we let $\text{cost}(\text{Alg}) = \sum_{i=1}^{q} |L_i|$.

Definition 4.1.2 (The distributions $\mathcal{G}_1$ and $\mathcal{G}_2$)**.** Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two distributions supported on graphs with vertex set $[n]$ defined as follows. Let $\mathbf{A} \subset [n]$ be a uniform random subset of size $\frac{n}{2}$.

$$\mathcal{G}_1 = \left\{ K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}} : \mathbf{A} \subset [n] \text{ random subset size } \frac{n}{2} \right\}$$
$$\mathcal{G}_2 = \left\{ K_{\mathbf{A},\overline{\mathbf{A}}} : \mathbf{A} \subset [n] \text{ random subset size } \frac{n}{2} \right\},$$

where for a subset A, K_A is the complete graph on vertices in A and $K_{A,\overline{A}}$ is the complete bipartite graph whose sides are A and $\overline{A}$.

4.2 Reducing Tolerant Junta Testing From Rejection Sampling

In this section, we will prove that distinguishing the two distributions $\mathcal{G}_1$ and $\mathcal{G}_2$ using a rejection sampling oracle reduces to distinguishing two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ over Boolean functions, where $\mathcal{D}_{\text{yes}}$ is supported on functions that are close to k-juntas and $\mathcal{D}_{\text{no}}$ is supported on functions that are far from any k-junta with high probability.

4.2.1 High Level Overview

We start by providing some intuition. We define two distributions supported on Boolean functions, $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, so that functions in $\mathcal{D}_{\text{yes}}$ are ε_0-close to being k-juntas and functions in $\mathcal{D}_{\text{no}}$ are ε_1-far from being k-juntas (where ε_0 and ε_1 are appropriately defined constants and $k = \frac{3n}{4}$). We note that the proof for general k follows by a simple padding argument (see Lemma A.2.3).

As mentioned in the introduction, our distributions are based on the indexing function used in [CST+17]. We draw a uniform random subset $\mathbf{M} \subset [n]$ of size $n/2$ and our function $\mathbf{\Gamma} = \Gamma_{\mathbf{M}} \colon \{0,1\}^n \to [2^{n/2}]$ projects the points onto the variables in $\mathbf{M}$. Thus, it remains to define the sequence of functions $\mathbf{H} = (\boldsymbol{h}_i \colon \{0,1\}^n \to \{0,1\} : i \in [2^{n/2}])$.

We will sample a graph $\mathbf{G} \sim \mathcal{G}_1$ (in the case of $\mathcal{D}_{\text{yes}}$), and a graph $\mathbf{G} \sim \mathcal{G}_2$ (in the case of $\mathcal{D}_{\text{no}}$) supported on vertices in $\overline{\mathbf{M}}$. Each function $\boldsymbol{h}_i \colon \{0,1\}^n \to \{0,1\}$ is given by first sampling an edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ and letting $\boldsymbol{h}_i$ be a parity (or a negated parity) of the variables $x_{\boldsymbol{j}_1}$ and $x_{\boldsymbol{j}_2}$. Thus, a function $\boldsymbol{f}$ from $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$ will have all variables being relevant, however, we will see that functions in $\mathcal{D}_{\text{yes}}$ have a group of $\frac{n}{4}$ variables which can be eliminated efficiently[2].

We think of the sub-functions $\boldsymbol{h}_i$ defined with respect to edges from $\mathbf{G}$ as implementing a sort of *gadget*: the gadget defined with respect to an edge (j_1, j_2) will have the property that if $\boldsymbol{f}$ eliminates the variable j_1, it will be "encouraged" to eliminate variable j_2 as well. In fact, each time an edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ is used to define a sub-function $\boldsymbol{h}_i$, any k-junta $g \colon \{0,1\}^n \to \{0,1\}$ where variable $\boldsymbol{j}_1$ or $\boldsymbol{j}_2$ is irrelevant will have to change half of the corresponding part indexed by $\mathbf{\Gamma}$. Intuitively, a function $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$ (which originally depends on all n variables) wants to eliminate its dependence of $n-k$ variables in order to become a k-junta. When $\boldsymbol{f}$ picks a variable $j \in \overline{\mathbf{M}}$ to eliminate (since variables in $\mathbf{M}$ are too expensive), it must change points in parts where the edge sampled is incident on j. The key observation is that when $\boldsymbol{f}$ needs to eliminate multiple variables, if $\boldsymbol{f}$ picks the variables j_1 and j_2 to eliminate, whenever a part samples the edge (j_1, j_2), the function changes the points in one part and eliminates two variables. Thus, $\boldsymbol{f}$ eliminates two variables by changing the same number of points when there are edges between j_1 and j_2.

At a high level, the gadgets encourage the function $\boldsymbol{f}$ to remove the dependence of variables within a group of edges, i.e., the closest k-junta will correspond to a function g which eliminates

[2]We say that a variable is eliminated if we change the function to remove the dependence of the variable.

groups of variables with edges within each other and few outgoing edges. More specifically, if we wants to eliminate $\frac{n}{4}$ variables from $\boldsymbol{f}$, we must find a bisection of the graph $\mathbf{G}$ whose cut value is small; in the case of $\mathcal{G}_1$, one of the cliques will have cut value 0, whereas any bisection of a graph from $\mathcal{G}_2$ will have a high cut value, which makes functions in $\mathcal{D}_{\text{yes}}$ closer to $\frac{3n}{4}$-juntas than functions in $\mathcal{D}_{\text{no}}$.

The reduction from rejection sampling is straight-foward. We consider all queries which are indexed to the same part, and if two queries indexed to the same part differ on a variable j, then we the algorithm "explores" direction j. Each part $i \in [2^{n/2}]$ where some query falls in has a corresponding rejection sampling query L_i, which queries the variables explored by the Boolean function testing algorithm.

4.2.2 The Distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$

The goal of this subsection is to define the two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, supported over Boolean functions with n variables. Functions $f \in \mathcal{D}_{\text{yes}}$ will be *close* to being a k-junta (for $k = \frac{3n}{4}$) with high probability, and functions $f \sim \mathcal{D}_{\text{no}}$ will be *far* from any k-junta with high probability.

Distribution $\mathcal{D}_{\text{yes}}$ A function f from $\mathcal{D}_{\text{yes}}$ is generated from a tuple of three random variables, $(\mathbf{M}, \mathbf{A}, \mathbf{H})$, and we set $f = f_{\mathbf{M},\mathbf{A},\mathbf{H}}$. The tuple is drawn according to the following randomized procedure:

1. Sample a uniformly random subset $\mathbf{M} \subset [n]$ of size $m \stackrel{\text{def}}{=} \frac{n}{2}$. Let $N = 2^m$ and $\Gamma_{\mathbf{M}} : \{0,1\}^n \to [N]$ be the function that maps $x \in \{0,1\}^n$ to a number encoded by $x|_{\mathbf{M}} \in [N]$.

2. Sample $\mathbf{A} \subset \overline{\mathbf{M}}$ of size $\frac{n}{4}$ uniformly at random, and consider the graph $\mathbf{G}$ defined on vertices $[\overline{\mathbf{M}}]$ with $\mathbf{G} = K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}}$, i.e., $\mathbf{G}$ is a uniformly random graph drawn according to $\mathcal{G}_1$.

3. Define a sequence of N functions $\mathbf{H} = \{\boldsymbol{h}_i \colon \{0,1\}^n \to \{0,1\} : i \in [N]\}$ drawn from a distribution $\mathcal{E}(\mathbf{G})$. For each $i \in \{1, \dots, N/2\}$, we let $\boldsymbol{h}_i(x) = \bigoplus_{\ell \in \mathbf{M}} x_\ell$.

   For each $i \in \{N/2+1, \dots, N\}$, we will generate $\boldsymbol{h}_i$ independently by sampling an edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ uniformly at random, as well as a uniform random bit $\boldsymbol{r} \sim \{0,1\}$. We let
   $$\boldsymbol{h}_i(x) = x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} \oplus \boldsymbol{r}.$$

4. Using $\mathbf{M}, \mathbf{A}$ and $\mathbf{H}$, define $f_{\mathbf{M},\mathbf{A},\mathbf{H}} = \boldsymbol{h}_{\Gamma_{\mathbf{M}}(x)}(x)$ for each $x \in \{0,1\}^n$.

Distribution $\mathcal{D}_{\text{no}}$ A function f drawn from $\mathcal{D}_{\text{no}}$ is also generated by first drawing the tuple $(\mathbf{M}, \mathbf{A}, \mathbf{H})$ and setting $f = f_{\mathbf{M},\mathbf{A},\mathbf{H}}$. Both $\mathbf{M}$ and $\mathbf{A}$ are drawn using the same procedure; the only difference is that the graph $\mathbf{G} = K_{\mathbf{A},\overline{\mathbf{A}}}$, i.e., $\mathbf{G}$ is a uniformly random graph drawn according to $\mathcal{G}_2$. Then $\mathbf{H} \sim \mathcal{E}(\mathbf{G})$ is sampled from the modified graph $\mathbf{G}$.

We let
$$k \stackrel{\text{def}}{=} \frac{3n}{4} \qquad \varepsilon_0 \stackrel{\text{def}}{=} \frac{1}{8} \qquad \varepsilon_1 \stackrel{\text{def}}{=} \frac{3}{16}.$$

Consider a fixed subset $M \subset [n]$ which satisfies $|M| = \frac{n}{2}$, and a fixed subset $A \subset \overline{M}$ which satisfies $|A| = \frac{n}{4}$. Let G be a graph defined over vertices in $\overline{M}$, and for any subsets $S_1, S_2 \subset \overline{M}$, let
$$E_G(S_1, S_2) = \left|\{(j_1, j_2) \in G : j_1 \in S_1, j_2 \in S_2\}\right|,$$

be the number of edges between sets S_1 and S_2. Additionally, we let

$$\chi(G) = \min\left\{\frac{E_G(S,S) + E_G(S,\overline{S})}{E_G(\overline{M},\overline{M})} : S \subset \overline{M}, |S| \geq \frac{n}{4}\right\} \tag{4.1}$$

be the minimum fraction of edges adjacent to a set S of size at least $\frac{n}{4}$. The following lemma relates the distance of a function $\boldsymbol{f} = f_{M,A,\mathbf{H}}$ where $\mathbf{H} \sim \mathcal{E}(G)$ to being a k-junta to $\chi(G)$. We then apply this lemma to the graph in $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ to show that functions in $\mathcal{D}_{\text{yes}}$ are ε_0-close to being k-juntas, and functions in $\mathcal{D}_{\text{no}}$ are ε_1-far from being k-juntas.

Lemma 4.2.1. *Let G be any graph defined over vertices in A. If $\boldsymbol{f} = f_{M,A,\mathbf{H}}$, where $\mathbf{H} \sim \mathcal{E}(G)$, then for $k = 3n/4$*

$$\frac{1}{4}\cdot\chi(G) - o(1) \leq \text{dist}(\boldsymbol{f}, k\text{-Junta}) \leq \frac{1}{4}\cdot\chi(G) + o(1)$$

with probability at least $1 - o(1)$.

Proof: We first show that $\text{dist}(\boldsymbol{f}, k\text{-Junta}) \leq \frac{1}{4}\cdot\chi(G) + o(1)$. Let $S \subset \overline{M}$ with $|S| \geq \frac{n}{4}$ be the subset achieving the minimum in (4.1), and consider the indicator random variables $\mathbf{X}_i$ for $i \in \{N/2+1, \dots, N\}$ defined as:

$$\mathbf{X}_i = \begin{cases} 1 & \boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus r \text{ with } j_1 \in S \text{ or } j_2 \in S \\ 0 & \text{otherwise} \end{cases},$$

and note that the variables $\mathbf{X}_i$ are independent and equal 1 with probability $\chi(G)$. Consider the function $\boldsymbol{g}\colon \{0,1\}^n \to \{0,1\}$ is defined as:

$$\boldsymbol{g}(x) = \begin{cases} \boldsymbol{h}_{\Gamma_M(x)}(x) & \mathbf{X}_{\Gamma_M(x)} = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Note that the function $\boldsymbol{g}$ is a k-junta, since $\boldsymbol{g}$ only depends on variables in $[n] \setminus S$, and $|S| \geq \frac{n}{4}$. In addition, we have that:

$$\text{dist}(\boldsymbol{f}, k\text{-Junta}) \leq \text{dist}(\boldsymbol{f}, \boldsymbol{g}) = \frac{1}{2^n}\sum_{i=N/2+1}^{N} \frac{2^{n-m}}{2}\cdot\mathbf{X}_i = \frac{1}{2\cdot 2^m}\sum_{i=N/2+1}^{N}\mathbf{X}_i,$$

and by a Chernoff bound, we obtain the desired upper bound.

For the lower bound, let $T \subset [n]$ of size $\frac{n}{4}$. We divide the proof into two cases: 1) $M \cap T \neq \emptyset$, and 2) $M \cap T = \emptyset$.

We handle the first case first, and let $j \in M \cap T$.

- Suppose j is the highest order bit of M, so that $\Gamma_M(x^{(j\to 0)}) \in \{1, \dots, N/2\}$ and $\Gamma_M(x^{(j\to 1)}) \in \{N/2+1, \dots, N\}$. For $y \in \{0,1\}^{M\setminus\{j\}}$ and $\alpha \in \{0,1\}$, let $X_{y,\alpha} = \{x \in \{0,1\}^n : x_{|M\setminus\{j\}} = y, x_j = \alpha\}$, $X_y = X_{y,0} \cup X_{y,1}$. For every $x \in X_y$,

$$\boldsymbol{f}(x) = \begin{cases} \bigoplus_{i\in M} x_i & x_j = 0 \\ x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} \oplus \boldsymbol{r} & x_j = 1 \end{cases},$$

  for some $\boldsymbol{j}_1, \boldsymbol{j}_2 \in \overline{M}$ and $\boldsymbol{r} \in \{0,1\}$. Thus, for at least half of all points in $x \in X_{y,0}$, $\boldsymbol{f}(x) \neq \boldsymbol{f}(x^{(j)})$. Therefore, for any function $g\colon \{0,1\}^n \to \{0,1\}$ which does not depend on j, for each $x \in X_{y,0}$ where $\boldsymbol{f}(x) \neq \boldsymbol{f}(x^{(j)})$, either $\boldsymbol{f}(x) \neq g(x)$, or $\boldsymbol{f}(x^{(j)}) \neq g(x^{(j)})$, thus,

$$\text{dist}(\boldsymbol{f}, g) \geq \frac{1}{2^n}\sum_{y\in\{0,1\}^{M\setminus\{j\}}} \frac{1}{2}\cdot|X_{y,0}| \geq \frac{1}{4}.$$

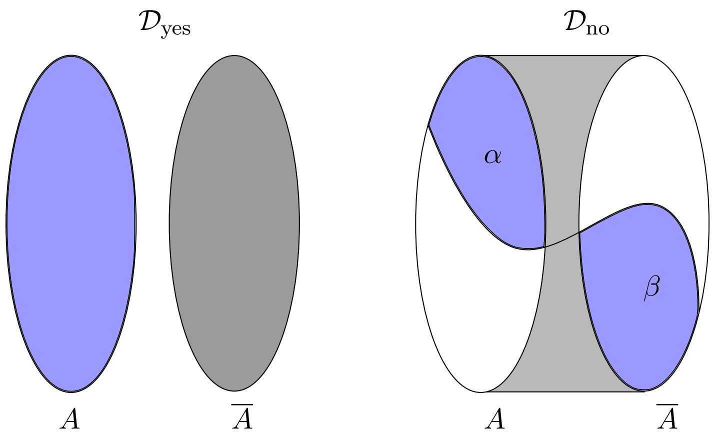


Figure 4.1: Example of graphs $\mathbf{G}$ from $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$. On the left, the graph $\mathbf{G}$ is the union of two cliques of size $\frac{n}{4}$, corresponding to $\mathcal{D}_{\text{yes}}$. We note that $\chi(G) = \frac{1}{2}$, since if we let $S = \mathbf{A}$ (pictured as the blue set), we see that S contains half of the edges. On the right, the graph $\mathbf{G}$ is the complete bipartite graph with side sizes $\frac{n}{4}$, corresponding to $\mathcal{D}_{\text{no}}$. We note that $\chi(G) = \frac{3}{4}$: consider any set $S \subset \overline{M}$ of size at least $\frac{n}{4}$ pictured in the blue region, and let $\alpha = |S \cap A|$ and $\beta = |S \cap \overline{A}|$, where $\alpha + \beta \geq \frac{n}{4}$, so $E(S, S) + E(S, \overline{S}) \geq (\frac{n}{4})^2 - \alpha\beta \geq (\frac{n}{4})^2(1 - \frac{1}{4})$.

- Suppose j is not the highest order bit of M. Then, if $\Gamma_M(x) \in \{1, \dots, N/2\}$, then $\Gamma_M(x^{(j)}) \in \{1, \dots, N/2\}$. We note that for each $y \in \{0,1\}^{M \setminus \{j\}}$ and $x \in X_{y,0}$ with $\Gamma_M(x) \in \{1, \dots, 2^{m-1}\}$, $\boldsymbol{f}(x) \neq \boldsymbol{f}(x^{(i)})$. Thus again, for any $g\colon \{0,1\}^n \to \{0,1\}$ which does not depend on j, $\text{dist}(\boldsymbol{f}, g) \geq \frac{1}{4}$, since half of all points $x \in \{0,1\}^n$ satisfy $\Gamma_M(x) \in \{1, \dots, N/2\}$.

Therefore, we may assume that $T \subset \overline{M}$. Again, consider the indicator random variables $\mathbf{X}_i$ for $i \in \{N/2+1, \dots, N\}$ given by

$$\mathbf{X}_i = \begin{cases} 1 & \text{if } \boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus r \text{ with } j_1 \in T \text{ or } j_2 \in T \\ 0 & \text{otherwise} \end{cases},$$

and by the definition of $\chi(G)$, we have that $\mathbf{X}_i = 1$ with probability at least $\chi(G)$. Suppose $x \in \{0,1\}^n$ with $\Gamma_M(x) = i$ and $\mathbf{X}_i = 1$ with $\boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus r$ with $j_1 \in T$, then $\boldsymbol{f}(x) \neq \boldsymbol{f}(x^{(j_1)})$, which means that any function $g\colon \{0,1\}^n \to \{0,1\}$ which does not depend on variables in T, either $g(x) \neq \boldsymbol{f}(x)$ or $g(x^{(j_1)}) \neq \boldsymbol{f}(x^{(j_1)})$, thus, for all such functions g,

$$\text{dist}(\boldsymbol{f}, g) \geq \frac{1}{4 \cdot 2^{m-1}} \sum_{i=N/2+1}^{N} \mathbf{X}_i \geq \frac{1}{4} \cdot \chi(G) - \frac{1}{n}$$

with probability $1 - \exp\left(-\Omega(\frac{N}{n^2})\right)$ by a Chernoff bound. Thus, we union bound over at most $2^{n/2}$ possible subsets $T \subset \overline{M}$ with $|T| \geq \frac{n}{4}$ to conclude that $\text{dist}(\boldsymbol{f}, k\text{-Junta}) \geq \frac{1}{4} \cdot \chi(G) - \frac{1}{n}$ with probability $1 - o(1)$. ■

Corollary 4.2.2. *We have that $\boldsymbol{f} \sim \mathcal{D}_{yes}$ has $\text{dist}(\boldsymbol{f}, k\text{-Junta}) \leq \varepsilon_0 + o(1)$ with probability $1 - o(1)$, and that $\boldsymbol{f} \sim \mathcal{D}_{no}$ has $\text{dist}(\boldsymbol{f}, k\text{-Junta}) \geq \varepsilon_1 - o(1)$ with probability $1 - o(1)$.*

Proof: For the upper bound in $\mathcal{D}_{\text{yes}}$, when $G = K_A \cup K_{\overline{A}}$, we have $\chi(G) \leq \frac{1}{2}$. For the lower bound in $\mathcal{D}_{\text{no}}$, when $G = K_{A,\overline{A}}$, $\chi(G) \geq \frac{3}{4}$ (see Figure 4.2.2). ■

4.2.3 Reducing from Rejection Sampling

In this subsection, we will prove that distinguishing the two distributions $\mathcal{G}_1$ and $\mathcal{G}_2$ using rejection sampling oracle reduces to distinguishing the two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$.

Lemma 4.2.3. *Suppose there exists a deterministic non-adaptive algorithm* Alg *making* $q = o(n^{50})$ *queries to Boolean functions* $f\colon \{0,1\}^{2n} \to \{0,1\}$*. Then, there exists a deterministic non-adaptive algorithm* Alg′ *making rejection sampling queries to an n-vertex graph such that:*

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\boldsymbol{f}\sim\mathcal{D}_{yes}}[\text{Alg}(\boldsymbol{f}) \text{ ``accepts''}] &= \mathop{\mathbf{Pr}}_{\mathbf{G}\sim\mathcal{G}_1}[\text{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_1\text{''}], \qquad \text{and} \\ \mathop{\mathbf{Pr}}_{\boldsymbol{f}\sim\mathcal{D}_{no}}[\text{Alg}(\boldsymbol{f}) \text{ ``accepts''}] &= \mathop{\mathbf{Pr}}_{\mathbf{G}\sim\mathcal{G}_2}[\text{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_1\text{''}].\end{aligned}$$

and has $\text{cost}(\text{Alg}') = O(q\log n)$ *with probability* $1 - o(1)$ *over the randomness in* Alg′*.*

Proof: Consider an algorithm Alg making q queries to a Boolean function $\boldsymbol{f} = f_{\mathbf{M},\mathbf{A},\mathbf{H}}\colon \{0,1\}^{2n} \to \{0,1\}$ (sampled from either $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$). First, note that $\mathbf{M}$ and $\mathbf{A}$ is distributed in the same way in $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$. Therefore, a rejection sampling algorithm may generate $\mathbf{M}$ and $\mathbf{A}$ and utilize its randomness from the rejection sampling oracle to simulate $\mathbf{H}$.

Specifically, given the queries $z_1, \dots, z_q \in \{0,1\}^{2n}$ of Alg, we will partition them into sets $\mathbf{Q}_1, \dots, \mathbf{Q}_t$, such that for all $z, z' \in \mathbf{Q}_i$, we have that $z|_{\mathbf{M}} = z'|_{\mathbf{M}}$. Given the above partition, we define our queries to the rejection sampling oracle $\mathbf{L}_1, \dots, \mathbf{L}_t \subset \overline{\mathbf{M}}$ such that for every $i \in [t]$ we let

$$\mathbf{L}_i \stackrel{\text{def}}{=} \{j \in \overline{\mathbf{M}} : \exists z, z' \in \mathbf{Q}_i, (z)_j \neq (z')_j\}\ .$$

Since $|\overline{\mathbf{M}}| = n$, we may associate each element of $\overline{\mathbf{M}}$ with an integer in $[n]$ and view the graphs in $\mathcal{G}_1$ and $\mathcal{G}_2$ as having vertex set $\overline{\mathbf{M}}$. In short, we let $\mathbf{L}_i$ is the set of indices with two queries in $\mathbf{Q}_i$ disagreeing in that index. Next, we claim that the cost of Alg′ is at most $O(q\log n)$ with probability $1 - o(1)$.

Consider the bad event which occurs if there exist two queries $z, z' \in \{0,1\}^{2n}$ such that $z|_{\mathbf{M}} = z'|_{\mathbf{M}}$ and $\|z - z'\|_1 > 100\log(2n)$. Note that for any two queries z, z' such that $\|z - z'\|_1 > 100\log(2n)$, the probability that $z|_{\mathbf{M}} = z'|_{\mathbf{M}}$ over the choice of $\mathbf{M}$ is at most $2^{-100\log(2n)} \ll \frac{1}{q^2}$, and thus we may use a union bound over all pairs of queries to get that the bad event occurs with probability $o(1)$. Therefore, we get that for any $i \in [t]$ and two queries $z, z' \in \mathbf{Q}_i$ we have that $\|z - z'\|_1 \le 100\log(2n)$ with probability $1 - o(1)$, which implies that the cost of Alg′ is $O(q\log n)$ with probability $1 - o(1)$.

Now, given the responses to the queries $\mathbf{L}_1, \dots, \mathbf{L}_t \subset [\overline{\mathbf{M}}]$, as well as the values of $\mathbf{M}, \mathbf{A}$, we will be able to simulate all the randomness in the construction of the two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$. More formally, Alg′ works in the following way.

1. Alg′ makes set queries $\mathbf{L}_1, \dots, \mathbf{L}_t$.

2. Once Alg′ receives the responses $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t \in \overline{\mathbf{M}} \cup \left(\overline{\mathbf{M}} \times \overline{\mathbf{M}}\right) \cup \{\emptyset\}$ from the oracle, it will generate a Boolean string $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q) \in \{0,1\}^q$ which is distributed exactly as $(f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z_1), \dots, f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z_q))$, where $f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{\text{yes}}$ if $\mathbf{G} \sim \mathcal{G}_1$ and $f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{\text{no}}$ if $\mathbf{G} \sim \mathcal{G}_2$.

3. Then if $\text{Alg}(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ outputs "accept", then Alg′ should output "$\mathcal{G}_1$", if $\text{Alg}(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ outputs "reject", then Alg′ should output "$\mathcal{G}_2$".

Next, we will describe how to generate $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q) \in \{0,1\}^q$. We start with setting some notations. For $i \in [t]$, we denote $\mathbf{Q}_i = \{z^i_1, \dots, z^i_{|\mathbf{Q}_i|}\}$ and $\boldsymbol{r}^i_1, \dots, \boldsymbol{r}^i_{|\mathbf{Q}_i|}$.

We aim to show that the random variables $(f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z^i_\ell) : \ell \in [|\mathbf{Q}_i|], i \in [t])$ when $f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{\text{yes}}$ is distributed exactly the same as $(\boldsymbol{r}^i_\ell : \ell \in [|\mathbf{Q}_i|], i \in [t])$ when $\mathbf{G} \sim \mathcal{G}_1$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ are sampled by the oracle (the complement case where $f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{\text{no}}$ and $\mathbf{G} \sim \mathcal{G}_2$ is similar).

We will proceed in t stages, each in stage $i \in [t]$, we will set the values of $\boldsymbol{r}^i_1, \dots, \boldsymbol{r}^i_{|\mathbf{Q}_i|}$ which will correspond to $f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z^i_1), \dots, f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z^i_{|\mathbf{Q}_i|})$.

If $\mathbf{Q}_i$ contains strings z such that $\Gamma_{\mathbf{M}}(z) \in \{1, \dots, 2^{n-1}\}$ then we let $\boldsymbol{r}^i_1, \dots, \boldsymbol{r}^i_{|\mathbf{Q}_i|}$ be given by $\boldsymbol{r}^i_\ell = \bigoplus_{j \in \mathbf{M}} (z^i_\ell)_j$ for $\ell \in [|\mathbf{Q}_i|]$. Otherwise $\Gamma_{\mathbf{M}}(z) \in \{2^{n-1}+1, \dots, 2^n\}$, the algorithm will use the response $\boldsymbol{v}_i$ to generate the values $\boldsymbol{r}^i_1, \dots, \boldsymbol{r}^i_{|\mathbf{Q}_i|}$: Alg$'$ samples a random bit $\boldsymbol{r}^i \sim \{0,1\}$ uniformly and generates $\boldsymbol{r}^i_1, \dots, \boldsymbol{r}^i_{|\mathbf{Q}_i|}$ according to three cases, corresponding to the three cases $\boldsymbol{v}_i$ can be in:

- If $\boldsymbol{v}_i = \emptyset$, then $\boldsymbol{r}^i_1 = \dots = \boldsymbol{r}^i_{|\mathbf{Q}_i|} = \boldsymbol{r}^i$.
- If $\boldsymbol{v}_i = \{j\} \subset \overline{\mathbf{M}}$, for each $\ell \in [|\mathbf{Q}_i|]$, $\boldsymbol{r}^i_\ell = \boldsymbol{r}^i$ if $(z^i_\ell)_j = 0$, and $\boldsymbol{r}^i_\ell = 1 - \boldsymbol{r}^i$ if $(z^i_\ell)_j = 1$.
- If $\boldsymbol{v}_i = \{j_1, j_2\} \subset \overline{\mathbf{M}}$, for each $\ell \in [|\mathbf{Q}_i|]$, $\boldsymbol{r}^i_\ell = \boldsymbol{r}^i$ if $(z^i_\ell)_{j_1} \oplus (z^i_\ell)_{j_2} = 0$, and $\boldsymbol{r}^i_\ell = 1 - \boldsymbol{r}^i$ if $(z^i_\ell)_{j_1} \oplus (z^i_\ell)_{j_2} = 1$.

We conclude with the following claim which is immediate from the definition of $\mathcal{D}_{\text{yes}}$, $\mathcal{D}_{\text{no}}$, $\mathcal{G}_1$ and $\mathcal{G}_2$, and the corresponding proof simply unravels the definitions of these distributions.

Claim 4.2.4. *If $\mathbf{G} \sim \mathcal{G}_1$, then $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ is distributed exactly as $(f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z_1), \dots, f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z_q))$ when $f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{yes}$, and if $\mathbf{G} \sim \mathcal{G}_2$, then $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ is distributed exactly as $(f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z_1), \dots, f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z_q))$ when $f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{no}$.*

Proof: We give the formal proof for $\mathcal{D}_{\text{yes}}$ and $\mathcal{G}_1$, as the case with $\mathcal{D}_{\text{no}}$ and $\mathcal{G}_2$ is the same argument. Recall from the definition of $\mathcal{D}_{\text{yes}}$, that $\mathbf{M}$ and $\mathbf{A}$ are uniform random sets of size n and $\frac{n}{2}$ respectively. Conditioned on $\mathbf{M}$ and $\mathbf{A}$, each sub-function $\boldsymbol{h}_i$ is picked independently. Thus, we have

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{f_{\mathbf{M},\mathbf{A},\mathbf{H}} \sim \mathcal{D}_{\text{yes}}} \left[\forall i \in [t], \forall \ell \in [|\mathbf{Q}_i|], f_{\mathbf{M},\mathbf{A},\mathbf{H}}(z^i_\ell) = y^i_\ell\right] \\
&\qquad = \binom{2n}{n}^{-1} \binom{n}{n/2}^{-1} \sum_{M \subset [2n]} \sum_{A \subset \overline{M}} \prod_{i=1}^{t} \mathop{\mathbf{Pr}}_{\boldsymbol{h}_i} \left[\forall \ell \in [|Q_i|], \boldsymbol{h}_i(z^i_\ell) = y^i_\ell \mid \mathbf{M} = M, \mathbf{A} = A\right].
\end{aligned}$$

We now turn to the graph problem. Recall from the definition of $\mathbf{G} \sim \mathcal{G}_1$, that conditioned on $\mathbf{M}$ and $\mathbf{A}$, the responses of the oracle, $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ are independent, and $\boldsymbol{r}^1, \dots, \boldsymbol{r}^t$ are independent. Thus, we may write:

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{M},\mathbf{A},\boldsymbol{v}_1,\dots,\boldsymbol{v}_t \\ \boldsymbol{r}^1,\dots,\boldsymbol{r}^t}} \left[\forall j \in [q], \forall \ell \in [|Q_i|], \boldsymbol{r}^i_\ell = y^i_\ell\right] = \binom{2n}{n}^{-1} \binom{n}{n/2}^{-1} \sum_{M} \sum_{A} \prod_{i=1}^{t} \mathop{\mathbf{Pr}}_{\boldsymbol{v}_i, \boldsymbol{r}^i} \left[\forall \ell \in [|Q_i|], \boldsymbol{r}^i_\ell = y^i_\ell\right].$$

Therefore, it suffices to show that for any $M \subset [2n]$ of size n, $A \subset \overline{M}$ of size $\frac{n}{2}$ and any $i \in [t]$, the random variable $(\boldsymbol{h}_i(z^i_\ell) : \ell \in [|Q_i|])$ with $\boldsymbol{h}_i$ from $\mathcal{D}_{\text{yes}}$ with sets M and A is distributed as $(\boldsymbol{r}^i_1, \dots, \boldsymbol{r}^i_{|Q_i|})$ with oracle response $\boldsymbol{v}_i$ and bit $\boldsymbol{r}^i$.

Let $(\boldsymbol{j}_1, \boldsymbol{j}_2)$ be a uniform random edge from $K_A \cup K_{\overline{A}}$, and we let $\boldsymbol{h}_i \colon \{0,1\}^{2n} \to \{0,1\}$ be given by:

$$\boldsymbol{h}_i(x) = \begin{cases} x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} & \text{with probability } \frac{1}{2} \\ \neg x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} & \text{with probability } \frac{1}{2} \end{cases}$$

Assume that $\boldsymbol{v}_i = L_i \cap \{\boldsymbol{j}_1, \boldsymbol{j}_2\} = \emptyset$. Then $\boldsymbol{h}_i(z_1^i) = \cdots = \boldsymbol{h}_i(z_{|Q_i|}^i)$ is given by a uniform random bit. Similarly, given these values of $\boldsymbol{v}_i = \emptyset$, $\boldsymbol{r}_1^i = \cdots = \boldsymbol{r}_{|Q_i|}^i$ is also given by a uniform random bit.

Now, assume that $L_i \cap \{\boldsymbol{j}_1, \boldsymbol{j}_2\} = \{\boldsymbol{j}\}$. Then, for any two queries $z, z' \in Q_i$ such that $(z)_{\boldsymbol{j}} \neq (z')_{\boldsymbol{j}}$ we must have that $\boldsymbol{h}_i(z) \neq \boldsymbol{h}_i(z')$, but after this condition is set, the value of any particular $\boldsymbol{h}_i(z)$ is a uniform random bit. Likewise, these constraints are set by the procedure generating $\boldsymbol{r}_1^i, \ldots, \boldsymbol{r}_{|Q_i|}^i$, and each $\boldsymbol{r}_\ell^i$ is a uniform random bit.

Finally, assume that $L_i \cap \{\boldsymbol{j}_1, \boldsymbol{j}_2\} = \{\boldsymbol{j}_1, \boldsymbol{j}_2\}$. Then, for any two queries $z, z' \in Q_i$ such that $(z)_{\boldsymbol{j}_1} \oplus (z)_{\boldsymbol{j}_2} \neq (z')_{\boldsymbol{j}_1} \oplus (z')_{\boldsymbol{j}_2}$ we have that $\boldsymbol{h}_i(z) \neq \boldsymbol{h}_i(z')$, and each value of $\boldsymbol{h}_i(z)$ is a uniform random bit. Finally, these constraints are also set forth in the definition of $\boldsymbol{r}_1^i, \ldots, \boldsymbol{r}_{|Q_i|}^i$. ■

■

Therefore, we conclude with the following corollary.

Corollary 4.2.5. *Suppose* Alg *is a deterministic non-adaptive algorithm which distinguishes* $\mathcal{D}_{yes}$ *and* $\mathcal{D}_{no}$ *supported on Boolean functions of* $2n$ *variables with query complexity* q*, then there exists a non-adaptive algorithm* Alg′ *for distinguishing between* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *supported on graphs with* n *vertices such that with probability* $1 - o(1)$ *over the randomness of* Alg′ *it holds that* $\mathrm{cost}(\mathrm{Alg}') = O(q \log n)$.

Proof: We have:

$$\Pr_{\mathbf{G}\sim\mathcal{G}_1}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_1\text{''}] - \Pr_{\mathbf{G}\sim\mathcal{G}_2}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_1\text{''}]$$
$$= \Pr_{f_{\mathbf{M},\mathbf{A},\mathbf{H}}\sim\mathcal{D}_{\mathrm{yes}}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}] - \Pr_{f_{\mathbf{M},\mathbf{A},\mathbf{H}}\sim\mathcal{D}_{\mathrm{no}}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}] \geq \frac{1}{3} - o(1).$$

We also have that with probability at least $1 - o(1)$, for each $i \in [t]$, if $Q_i = \{z_1^i, \ldots, z_{|Q_i|}^i\}$, then $|L_i| \leq \sum_{j=2}^{|Q_i|} \|z_1^i - z_j^i\|_1 \leq |Q_i| \cdot 100 \log(2n)$. Therefore, $\mathrm{cost}(\mathrm{Alg}') = \sum_{i=1}^t |L_i| = O(q \log n)$ with probability at least $1 - o(1)$. ■

4.3 A lower bound for distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$ with rejection samples

In this section, we derive a lower bound for distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$ with rejection samples.

Lemma 4.3.1. *Any deterministic non-adaptive algorithm* Alg *with* $\mathrm{cost}(\mathrm{Alg}) \leq \frac{n^2}{\log^6 n}$*, has:*

$$\Pr_{\mathbf{G}\sim\mathcal{G}_1}[\mathrm{Alg} \textit{ outputs ``}\mathcal{G}_1\textit{''}] \leq (1 + o(1)) \Pr_{\mathbf{G}\sim\mathcal{G}_2}[\mathrm{Alg} \textit{ outputs ``}\mathcal{G}_1\textit{''}] + o(1).$$

We assume Alg is a deterministic non-adaptive algorithm with $\mathrm{cost}(\mathrm{Alg}) \leq \frac{n^2}{\log^6 n}$. Alg makes queries $L_1, \ldots, L_t \subset [n]$ and the oracle returns $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t$, some of which are edges, some are lone vertices, and some are $\emptyset$. Let $\mathbf{G}_o \subset \mathbf{G}$ be the graph observed by the algorithm by considering all edges in $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t$. We let $|\mathbf{G}_o|$ be the number of edges.

Before going on to prove the lower bound, we use the following simplification. First, we assume that any algorithm Alg has all its queries $L_1, \ldots, L_t$ satisfying that either $|L_i| \leq \frac{n}{\log n}$, or $L_i = [n]$. Thus, it suffices to show for this restricted class of algorithms, the cost must be at least $\frac{n^2}{\log^5 n}$.

4.3.1 High Level Overview

In this subsection, we will give a high level overview of the proof of Lemma 4.3.1.

The idea is that we will argue outcome-by-outcome; i.e., we consider the possible ways the algorithm can act, which depends on the responses to the queries the algorithm gets. Consider some responses $v_1, \dots, v_t \in [n] \cup ([n] \times [n]) \cup \{\emptyset\}$, where each v_i may be either a lone vertex, an edge, or $\emptyset$. Suppose that upon observing this outcome, the algorithm outputs "$\mathcal{G}_1$". There will be two cases:

- The first case is when the probability of observing this outcome from $\mathcal{G}_2$ is not too much lower than the probability of observing this outcome from $\mathcal{G}_1$. In these outcomes, we will not get too much advantage in distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$.
- The other case is when the probability of observing this outcome from $\mathcal{G}_2$ is substantially lower than the probability of observing this outcome from $\mathcal{G}_1$. These cases do help us distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$; thus, we will want to show that collectively, the probability that we observe these outcomes from $\mathcal{G}_1$ is $o(1)$.

We will be able to characterize the outcomes which fall into the first case and the second case by considering a sequence of events. In particular we define five events which depend on $v_1, \dots, v_t$, as well as the random choice of $\mathbf{A}$. Consider the outcome $v_1, \dots, v_t$ which together form components $C_1, \dots, C_\alpha$. The events are the following[3]:

1. $\mathcal{E}_T$ (Observe small trees): this is the event where the values of $v_1, \dots, v_t$ form components $C_1, \dots, C_\alpha$ which are all trees of size at most $\log n$.
2. $\mathcal{E}_F$ (Observe few non-empty responses): this is the event where the values of $v_1, \dots, v_t$ have at most $\frac{n}{\log^4 n}$ non-$\emptyset$ responses. This event implies that the total number of vertices in the responses $v_1, \dots, v_t$ is at most $\frac{n}{\log^4 n}$.
3. $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ and $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ (Consistency condition of the components observed): these are the events where $\mathbf{A} \subset [n]$ partitions the components $C_1, \dots, C_\alpha$ in a manner consistent with $\mathcal{G}_1$ in $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ or $\mathcal{G}_2$ in $\boldsymbol{\mathcal{E}}_{C,\text{no}}$. See Definition 4.3.5 for a formal definition of this event. These events are random variables that depend only on $\mathbf{A}$. It will become clear that in order to observe the outcome $v_1, \dots, v_t$ in $\mathcal{G}_1$, event $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ must be triggered, and in $\mathcal{G}_2$, event $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ must be triggered. See Figure 4.2 for an illustration.
4. $\boldsymbol{\mathcal{E}}_O$ (Observe specific responses) : this event is over the randomness in $\mathbf{A}$, as well as the randomness in the responses of the oracle $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$. The event is triggered when the responses of the oracle are exactly those dictated by $v_1, \dots, v_t$; i.e., for all $i \in [t]$, $\boldsymbol{v}_i = v_i$.
5. $\boldsymbol{\mathcal{E}}_B$ (Balanced lone vertices condition) : this event is over the randomness in $\mathbf{A}$, as well as the responses $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$. The event occurs when a particular quantity which depends on $\mathbf{A}$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ is bounded by some predetermined value. See Definition 4.3.15 for a formal definition.

Having defined these events, the lower bound follows by the following three lemmas. The first lemma says that for any outcomes satisfying $\mathcal{E}_T$ and $\mathcal{E}_F$, the probability over $\mathbf{A}$ of being consistent in $\mathcal{G}_1$ cannot be much higher than in $\mathcal{G}_2$. The second lemma says that the outcomes satisfying the events described above do not help in distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$. The third lemma says that good outcomes occur with high probability over $\mathcal{G}_1$.

[3]We note that the first two event are not random and depends on the values $v_1, \dots, v_t$, and the rest are random variables depending on the partition $\mathbf{A}$ and the oracle responses $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$.

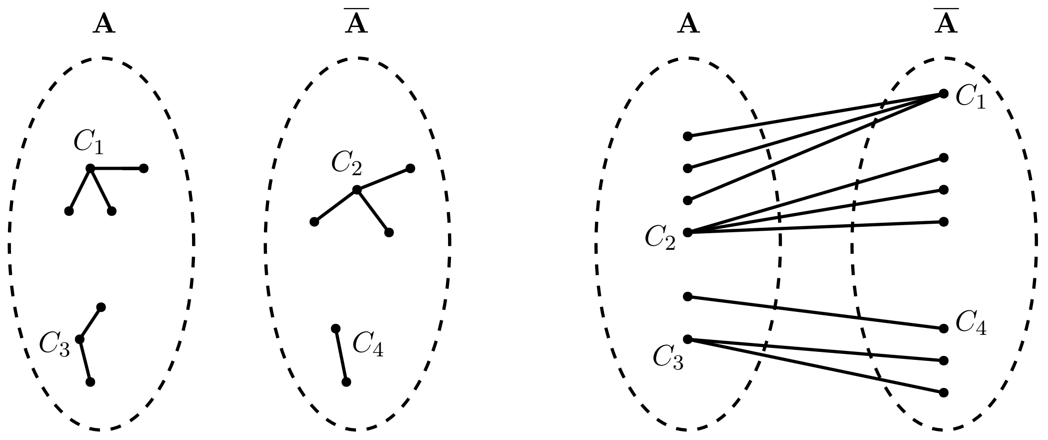


Figure 4.2: **A** consistently partition of the components C_1, C_2, C_3 and C_4 according to $\mathcal{G}_1$ (on the left) and $\mathcal{G}_2$ (on the right).

Lemma 4.3.2 (Consistency Lemma). *Consider a fixed $v_1, \dots, v_t \in [n] \cup ([n] \times [n]) \cup \{\emptyset\}$ forming components $C_1, \dots, C_\alpha$ where events $\mathcal{E}_T$ and $\mathcal{E}_F$ are satisfied. Then, we have:*

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,yes}] \le (1+o(1)) \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,no}].$$

Lemma 4.3.3 (Good Outcomes Lemma). *Consider a fixed $v_1, \dots, v_t \in [n] \cup ([n] \times [n]) \cup \{\emptyset\}$ forming components $C_1, \dots, C_\alpha$ where events $\mathcal{E}_T$ and $\mathcal{E}_F$ are satisfied. Then, we have:*

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_O \wedge \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,yes}] \le (1+o(1)) \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_O \mid \boldsymbol{\mathcal{E}}_{C,no}].$$

Lemma 4.3.4 (Bad Outcomes Lemma). *We have that:*

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\neg\boldsymbol{\mathcal{E}}_T \vee \neg\boldsymbol{\mathcal{E}}_F \vee \neg\boldsymbol{\mathcal{E}}_B] = o(1).$$

Assuming the above three lemmas, we may prove Lemma 4.3.1.

Proof: Let Λ be the set of outcomes of the algorithm which output "$\mathcal{G}_1$." Each outcome is a collection of responses $v_1, \dots, v_t$. We let

$$\Lambda_G = \{\ell \in \Lambda : \text{ responses } v_1, \dots, v_t \text{ satisfy } \mathcal{E}_T \wedge \mathcal{E}_F\},$$

and $\boldsymbol{\mathcal{E}}_{O,\ell}$ be the event that responses $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ result in outcome ℓ. We have:

$$\begin{aligned}
\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\text{Alg outputs "}\mathcal{G}_1\text{"}] &\le \sum_{\ell\in\Lambda_G} \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\ell \text{ is observed by Alg} \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}}] + \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\neg\boldsymbol{\mathcal{E}}_T \vee \neg\boldsymbol{\mathcal{E}}_F] \\
&\le \sum_{\ell\in\Lambda_G} \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{O,\ell} \wedge \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}}] + \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\neg\boldsymbol{\mathcal{E}}_T \vee \neg\boldsymbol{\mathcal{E}}_F \vee \neg\boldsymbol{\mathcal{E}}_B] \\
&\le (1+o(1)) \sum_{\ell\in\Lambda_G} \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{O,\ell} \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{no}}] + o(1) \\
&\le (1+o(1)) \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}} [\text{Alg outputs "}\mathcal{G}_1\text{"}] + o(1),
\end{aligned}$$

where we used Lemma 4.3.2, Lemma 4.3.3, and Lemma 4.3.4 from the second to third line. ■

4.3.1.1 Proof of the Consistency Lemma: Lemma 4.3.2

We now turn to proving Lemma 4.3.2. We first give the formal definitions of events $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ and $\boldsymbol{\mathcal{E}}_{C,\text{no}}$. Next, we set up some definitions necessary for the proof and give two claims which imply the lemma. For the remainder of the section, we consider fixing the responses $v_1, \dots, v_t \in [n] \cup ([n] \times [n]) \cup \{\emptyset\}$. We assume the responses form the components $C_1, \dots, C_\alpha$ which satisfy events $\mathcal{E}_T$ and $\mathcal{E}_F$. For each $i \in [\alpha]$, let u_i be the minimum vertex in C_i with respect to the natural ordering of $[n]$, and consider rooting the trees C_i at u_i, forming a layered tree with at most $\log n$ layers. Namely, u_i will be in the first layer, all its neighbors in C_i will be in the second layer, and so on. We let $C_i(\text{even})$ be the set of vertices in even layers, and $C_i(\text{odd})$ be the set of vertices in odd layers.

Definition 4.3.5. We let $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ be the event that $\mathbf{A} \subset [n]$ is consistent with the observations $v_1, \dots, v_t$ when $\mathbf{G} = K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}}$, and $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ be the event that $\mathbf{A} \subset [n]$ is consistent with the observations $v_1, \dots, v_t$ when $\mathbf{G} = K_{\mathbf{A},\overline{\mathbf{A}}}$. In other words,

- In $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$: for all $i \in [\alpha]$, either $C_i \subset \mathbf{A}$ or $C_i \subset \overline{\mathbf{A}}$.
- In $\boldsymbol{\mathcal{E}}_{C,\text{no}}$: for all $i \in [\alpha]$, either $C_i(\text{odd}) \subset \mathbf{A}$ and $C_i(\text{even}) \subset \overline{\mathbf{A}}$, or $C_i(\text{odd}) \subset \overline{\mathbf{A}}$ and $C_i(\text{even}) \subset \mathbf{A}$.

For each $i \in [\alpha]$, let $\mathbf{Y}_i$ be the indicator random variable for $u_i \in \mathbf{A}$. Let:

$$\mathbf{W}_{A,\text{yes}} = \sum_{i=1}^{\alpha} \mathbf{Y}_i \cdot |C_i| \qquad \mathbf{W}_{A,\text{no}} = \sum_{i=1}^{\alpha} \left(\mathbf{Y}_i \cdot |C_i(\text{odd})| + (1 - \mathbf{Y}_i) \cdot |C_i(\text{even})|\right) \qquad V = \sum_{i=1}^{\alpha} |C_i|.$$

Definition 4.3.6. We let $\boldsymbol{\mathcal{E}}_W$ be the event where:

$$\frac{V}{2} - \sqrt{V} \log n \le \mathbf{W}_{A,\text{no}} \le \frac{V}{2} + \sqrt{V} \log n.$$

Lemma 4.3.2 follows from the next two claims.

Claim 4.3.7. *For $v_1, \dots, v_t$ satisfying events $\mathcal{E}_T$ and $\mathcal{E}_F$, we have:*

$$\Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,yes} \wedge \boldsymbol{\mathcal{E}}_W] \le (1 + o(1)) \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_2 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,no}].$$

Claim 4.3.8. *For $v_1, \dots, v_t$ satisfying events $\mathcal{E}_T$ and $\mathcal{E}_F$, we have:*

$$\Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\neg \boldsymbol{\mathcal{E}}_W \mid \boldsymbol{\mathcal{E}}_{C,yes}] = o(1).$$

Given Claim 4.3.7 and Claim 4.3.8, we proceed to proving Lemma 4.3.2.
Proof of Lemma 4.3.2: We simply compute the respective probabilities.

$$\begin{aligned} \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}}] &= \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}} \wedge \boldsymbol{\mathcal{E}}_W] + \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\neg \boldsymbol{\mathcal{E}}_W \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}}] \\ &\le (1 + o(1)) \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_2 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{no}}] + o(1) \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}}], \end{aligned} \tag{4.2}$$

Where we applied both Claim 4.3.7 and Claim 4.3.8 in Line (4.2). Finally, this implies:

$$(1 - o(1)) \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_1 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{yes}}] \le (1 + o(1)) \Pr_{\substack{\mathbf{G} \sim \mathcal{G}_2 \\ \boldsymbol{v}_1, \dots, \boldsymbol{v}_t}} [\boldsymbol{\mathcal{E}}_{C,\text{no}}],$$

which finishes the proof. ■

We now proceed to proving Claim 4.3.7, followed by the proof of Claim 4.3.8.

Proof of Claim 4.3.7: Note that $V \leq \frac{n}{\log^4 n}$ since event $\mathcal{E}_F$ is satisfied. Let $y \in \{0,1\}^\alpha$ be an assignment of $u_1, \dots, u_\alpha$ to $\mathbf{A}$; more formally, for a fixed $y \in \{0,1\}^\alpha$, we let $\boldsymbol{\mathcal{E}}_y$ be the event that for each $i \in [\alpha]$, $u_i \in \mathbf{A}$ if $y_i = 1$, and $u_i \in \overline{\mathbf{A}}$ if $y_i = 0$. Additionally, let

$$Y_G = \{y \in \{0,1\}^\alpha : \text{ if } \mathbf{A} \text{ satisfies } \boldsymbol{\mathcal{E}}_y, \text{ then } \boldsymbol{\mathcal{E}}_W \text{ is satisfied}\}.$$

Then,

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{C,\text{yes}} \wedge \boldsymbol{\mathcal{E}}_W] = \sum_{y \in Y_G} \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{C,\text{yes}} \wedge \boldsymbol{\mathcal{E}}_y].$$

It suffices to show that for $y \in Y_G$:

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{C,\text{yes}} \wedge \boldsymbol{\mathcal{E}}_y] \leq (1+o(1)) \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{C,\text{no}} \wedge \boldsymbol{\mathcal{E}}_y].$$

Note that if $\mathbf{A}$ satisfies $\boldsymbol{\mathcal{E}}_y$ and $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ is satisfied, there is precisely one choice for assigning each vertex in $C_1, \dots, C_\alpha$ to $\mathbf{A}$ or $\overline{\mathbf{A}}$. Likewise, if $\mathbf{A}$ satisfied $\boldsymbol{\mathcal{E}}_y$ and $\boldsymbol{\mathcal{E}}_{C,\text{no}}$, there is precisely one choice for assigning each vertex in $C_1, \dots, C_\alpha$ to $\mathbf{A}$ or $\overline{\mathbf{A}}$. The remaining vertices may be placed in $\mathbf{A}$ or $\overline{\mathbf{A}}$ so the resulting set $\mathbf{A}$ contains half of all vertices, therefore, we have:

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{C,\text{yes}} \wedge \boldsymbol{\mathcal{E}}_y] \leq \frac{\binom{n-V}{\frac{n}{2}-\frac{V}{2}}}{\binom{n}{n/2}} \qquad \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{C,\text{no}} \wedge \boldsymbol{\mathcal{E}}_y] \geq \frac{\binom{n-V}{\frac{n}{2}-\frac{V}{2}-\sqrt{V}\log n}}{\binom{n}{n/2}}.$$

Taking the ratio, we have:

$$\begin{aligned}\frac{\mathbf{Pr}[\boldsymbol{\mathcal{E}}_{C,\text{yes}} \wedge \boldsymbol{\mathcal{E}}_y]}{\mathbf{Pr}[\boldsymbol{\mathcal{E}}_{C,\text{no}} \wedge \boldsymbol{\mathcal{E}}_y]} &\leq \frac{\binom{n-V}{\frac{n}{2}-\frac{V}{2}}}{\binom{n}{n/2}} \cdot \frac{\binom{n}{n/2}}{\binom{n-V}{\frac{n}{2}-\frac{V}{2}-\sqrt{V}\log n}} \leq \left(\frac{\frac{n}{2}-\frac{V}{2}+\sqrt{V}\log n}{\frac{n}{2}-\frac{V}{2}-\sqrt{V}\log n}\right)^{\sqrt{V}\log n} \\ &\leq \left(1+O\left(\frac{1}{\sqrt{n}\log n}\right)\right)^{\sqrt{n}/\log n} = 1+o(1).\end{aligned}$$

■

Proof of Claim 4.3.8: We let:

$$\mathbf{W}^{(O)}_{A,\text{no}} = \sum_{i=1}^{\alpha} \mathbf{Y}_i \cdot |C_i(\text{odd})| \qquad \text{and} \qquad \mathbf{W}^{(E)}_{A,\text{no}} = \sum_{i=1}^{\alpha} (1-\mathbf{Y}_i) \cdot |C_i(\text{even})|.$$

where $\mathbf{W}^{(O)}_{A,\text{no}} + \mathbf{W}^{(E)}_{A,\text{no}} = \mathbf{W}_{A,\text{no}}$ specifies the number of vertices in $\cup_{i\in[\alpha]} C_i$ assigned to $\mathbf{A}$. Conditioning on event $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$, $\mathbf{A}$ and $\overline{\mathbf{A}}$ can be interchanged, so

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\mathbf{Y}_i = 1 \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] = \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\mathbf{Y}_i = 0 \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] = \frac{1}{2}.$$

So,

$$\mathop{\mathbf{E}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\mathbf{W}^{O}_{A,\text{no}} \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] = \frac{1}{2}\sum_{i\in[\alpha]} |C_i(\text{odd})| \qquad \text{and} \qquad \mathop{\mathbf{E}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\mathbf{W}^{E}_{A,\text{no}} \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] = \frac{1}{2}\sum_{i\in[\alpha]} |C_i(\text{even})|.$$

Additionally, for any set of indices $I \subset [\alpha]$,

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\forall i \in I, \mathbf{Y}_i = 1 \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] \leq \frac{1}{2^{|I|}} \qquad \text{and} \qquad \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\forall i \in I, \mathbf{Y}_i = 0 \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] \leq \frac{1}{2^{|I|}},$$

which implies that the variables $\mathbf{Y}_i$, as well as the variables in $1 - \mathbf{Y}_i$ are negatively correlated. We may apply Chernoff bounds (for negatively correlated variables) to obtain deviation bounds for $\mathbf{W}^{(O)}_{A,\text{no}}$ and $\mathbf{W}^{(E)}_{A,\text{no}}$. Then, a union bound gives the desired result for $\mathbf{W}_{A,\text{no}}$. ■

4.3.1.2 Proof of the Bad Outcomes Lemma: Lemma 4.3.4

In this section, we give a proof of Lemma 4.3.4, which says that the probability over $\mathbf{G} \sim \mathcal{G}_1$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ of not satisfying events $\boldsymbol{\mathcal{E}}_T$, $\boldsymbol{\mathcal{E}}_F$, as well as $\boldsymbol{\mathcal{E}}_B$ is $o(1)$. In order to prove this, we will show that individually, the probability of not satisfying each event is $o(1)$ and conclude with a union bound.

4.3.1.3 $\boldsymbol{\mathcal{E}}_T$: components observed are small trees

The goal of this section is to show that with high probability, the algorithm only sees edges which form various components of small trees.

Definition 4.3.9. We let $\boldsymbol{\mathcal{E}}_T$ be the event that observed responses $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ generate components $\mathbf{C}_1, \dots, \mathbf{C}_\alpha$ which are all trees of size less than $\log n$.

Lemma 4.3.10. *We have that:*

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_T] \geq 1 - o(1).$$

We prove the above lemma by showing the following two claims.

Claim 4.3.11. *With probability $1 - o(1)$ over the draw of $\mathbf{G} \sim \mathcal{G}_1$ and the draw of $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$, $\mathbf{G}_o$ has no cycles.*

Proof: Recall that $L_1, \dots, L_t$ are the set queries made, and let $\boldsymbol{\mathcal{E}}_{\circ,\ell}$ be the event that $\mathbf{G}_o$ has a cycle of length ℓ. We have:

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{\circ,\ell}] &\leq \sum_{\substack{S\subset[t]\\ S=\{i_1,\dots,i_\ell\}}} \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{v}_{i_1}, \dots, \boldsymbol{v}_{i_\ell} \text{ form cycle}] \\ &\leq \sum_{\substack{S\subset[t]\\ S=\{i_1,\dots,i_\ell\}}} \sum_{\substack{U\subset[n]\\ U=\{u_1,\dots,u_\ell\}\\ u_j\in L_{i_j}\cap L_{i_{j+1}}}} \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\forall j \in [\ell], \boldsymbol{v}_{i_j} = (u_j, u_{j+1})], \qquad (4.3)\end{aligned}$$

where we think $j + 1 = 1$ when $j = \ell$. The above restriction of $u_j \in L_{i_j} \cap L_{i_{j+1}}$ is necessary if edges $\boldsymbol{v}_{i_j}$ and $\boldsymbol{v}_{i_{j+1}}$ will be the edges of the cycle incident on node u_j. Additionally, we may upper bound (4.3) by disregarding the effect of the partition $\mathbf{A}$ and $\overline{\mathbf{A}}$; in fact, the presence of $\mathbf{A}$ and $\overline{\mathbf{A}}$ make it harder to achieve a cycle, since if $u_j \in \mathbf{A}$ and $u_{j+1} \in \overline{\mathbf{A}}$, the probability of $\boldsymbol{v}_{i_j} = (u_j, u_{j+1})$ is 0. For any $S = \{i_1, \dots, i_\ell\}$, once we fix a set $U = \{u_1, \dots, u_\ell\}$ where $u_j \in L_{i_j} \cap L_{i_j+1}$,

$$\mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\forall j \in [\ell], \boldsymbol{v}_{i_j} = (u_j, u_{j+1})] \leq \left(\frac{1}{2\binom{n/2}{2}}\right)^\ell.$$

Thus, we have:

$$\begin{aligned}
\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{\circ,\ell}] &\le \sum_{\substack{S\subset[t]\\ S=\{i_1,\dots,i_\ell\}}} \left(\prod_{j=1}^{\ell} |L_{i_j}\cap L_{i_{j+1}}|\right)\left(\frac{1}{2\binom{\frac{n}{2}}{2}}\right)^{\ell}\\
&\le \left(\frac{1}{\Omega(n)}\right)^{2\ell} \sum_{\substack{S\subset[t]\\ S=\{i_1,\dots,i_\ell\}}} \prod_{j=1}^{\ell}|L_{i_j}|\\
&\le \left(\frac{1}{\Omega(n)}\right)^{2\ell}\left(\sum_{i=1}^{t}|L_i|\right)^{\ell}\left(\frac{1}{t}\right)^{\ell}\binom{t}{\ell} \le \left(O\left(\frac{1}{\log^5 n}\right)\right)^{\ell}.
\end{aligned}$$

where we used the fact that $\sum_S \prod_{j=1}^{\ell}|L_{i_j}|$ is the elementary symmetric polynomial of degree ℓ, and $\sum_{i=1}^{t}|L_i| \le \frac{n^2}{\log^5 n}$. Thus, we obtain:

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\mathbf{G}_o \text{ contains a cycle}] \le \sum_{\ell=1}^{t}\left(O\left(\frac{1}{\log^5 n}\right)\right)^{\ell} = o(1).$$

■

Claim 4.3.12. *With probability* $1-o(1)$ *over the draw of* $\mathbf{G}\sim\mathcal{G}_1$ *and the draw of* $\boldsymbol{v}_1,\dots,\boldsymbol{v}_t$*, we have* $\mathbf{G}_o$ *has all components of size at most* $\log n$*.*

Proof: This proof is very similar to the one above. Let $\boldsymbol{\mathcal{E}}_{T,\ell}$ be the event there exists a tree of ℓ edges. We note that there are at most $\exp(O(\ell))$ rooted trees of ℓ edges and $\ell+1$ vertices. We consider first picking a rooted tree, and we pick an arbitrary vertex to be the root of the tree. We then pick the ℓ edges of the tree to some responses, $\boldsymbol{v}_{i_1},\dots,\boldsymbol{v}_{i_\ell}$. We select the vertex on query of the edge going away from the root; this leaves the root, which we choose arbitrarily from $[n]$.

So we have:

$$\begin{aligned}
\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_{T,\ell}] &\le \exp(O(\ell)) \sum_{\substack{S\subset[t]\\ S=\{i_1,\dots,i_\ell\}}} \left(n\prod_{j=1}^{\ell}|L_{i_j}|\right)\left(\frac{1}{2\binom{n/2}{2}}\right)^{\ell}\\
&\le n\cdot\left(O\left(\frac{1}{\log^5 n}\right)\right)^{\ell} = \left(O\left(\frac{1}{\log^5 n}\right)\right)^{\ell},
\end{aligned}$$

when $\ell \ge \log n$. Thus, we sum over all $\ell \ge \log$ to get that there exists a tree of size $\log n$ or greater with probability $o(1)$. ■

4.3.1.4 $\boldsymbol{\mathcal{E}}_F$: few vertices are observed

The goal of this section is to show that the algorithm does not observe too many vertices from the responses $\boldsymbol{v}_1,\dots\boldsymbol{v}_t$ with high probability.

Definition 4.3.13. We let $\boldsymbol{\mathcal{E}}_F$ be the event that the responses $\boldsymbol{v}_1,\dots,\boldsymbol{v}_t$ contain at most $\frac{n}{\log^4 n}$ values which are not $\emptyset$.

Lemma 4.3.14. *We have:*

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_F] \geq 1 - o(1) \qquad \textit{and} \qquad \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\dots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_F] \geq 1 - o(1).$$

In other words, any rejection sampling algorithm with cost less than $\frac{n^2}{\log^6 n}$ will observe at most $\frac{n}{\log^4 n}$ non-$\emptyset$ responses in both $\mathcal{G}_1$ and $\mathcal{G}_2$ with high probability.

Proof: Simply note that for a query L_i, and any $G \in \mathcal{G}_1$, the probability of observing a response which is not $\emptyset$ is at most $\dfrac{|L_i| \cdot \frac{n}{2}}{2\binom{n/2}{2}} = O(|L_i|/n)$ (in the case of $\mathcal{G}_1$, and $\frac{|L_i|\cdot\frac{n}{2}}{n^2/4}$ in the case of $\mathcal{G}_2$). Therefore, the expected number of responses which are not $\emptyset$ is at most $O(n/\log^5 n)$, and via a Markov bound, we have the desired result. ■

4.3.1.5 $\boldsymbol{\mathcal{E}}_B$: vertices observed do not prefer any side too much

We now formally define the event $\boldsymbol{\mathcal{E}}_B$, and prove the event occurs with high probability over the draw of $\mathbf{G} \sim \mathcal{G}_1$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$.

Definition 4.3.15. Let $\mathbf{V}_L \subset [t]$ be the random variable corresponding to the set of indices of responses $\boldsymbol{v}_1, \dots, \boldsymbol{v}_t$ which correspond to observing lone vertices, and for $i \in \mathbf{V}_L$, we let $\boldsymbol{y}_i$ be the indicator random variable for $\boldsymbol{v}_i \in \mathbf{A}$. Let $\boldsymbol{\mathcal{E}}_B$ be the event where:

$$\mathbf{B} = \sum_{i\in\mathbf{V}_L} (-1)^{\boldsymbol{y}_i} \left(|L_i \cap \mathbf{A}| - |L_i \cap \overline{\mathbf{A}}|\right) = O\left(\frac{n}{\log n}\right).$$

We start by giving some intuition. Fix some query L_i such that $|L_i| \leq \frac{n}{\log n}$. By using Chernoff bound we have that $||L_i \cap \mathbf{A}| - |L_i \cap \overline{\mathbf{A}}|| = O(\sqrt{|L_i|}\log n)$ with high probability. Now assume that *every* query we make is skewed toward $\overline{\mathbf{A}}$. This bad event will create a gap in the probabilities to see a lone vertex between the two distributions, and the algorithm might use it in order to distinguish $\mathcal{G}_1$ and $\mathcal{G}_2$. Hence, we would like to claim that *collectively* the probability of observing such bad events is extremely small. More precise details follows.

Definition 4.3.16. Let $\boldsymbol{\mathcal{E}}_Q$ be the event that all queries $L_1, \dots, L_t$ satisfy:

$$\left||L_i \cap \mathbf{A}| - |L_i \cap \overline{\mathbf{A}}|\right| = O\left(\sqrt{|L_i|}\log n\right).$$

Claim 4.3.17. *We have:*

$$\Pr_{\mathbf{G}\sim\mathcal{G}_1}[\boldsymbol{\mathcal{E}}_Q] \geq 1 - o(1).$$

Proof: This simply follows from a union bound over $2t$ applications of the Chernoff bound for negatively correlated random variables. In particular, for all $k \in [n]$, let $\mathbf{Y}_k$ be the indicator random variable for $k \in \mathbf{A}$. Then we note that for each $i \in [t]$,

$$|L_i \cap \mathbf{A}| = \sum_{k\in L_i} \mathbf{Y}_k \qquad \text{and} \qquad |L_i \cap \overline{\mathbf{A}}| = \sum_{k\in L_i} (1 - \mathbf{Y}_k).$$

In a similar way to the proof of Claim 4.3.8, we note that all $\mathbf{Y}_k$ are negatively correlated, and all $(1 - \mathbf{Y}_i)$ are negatively correlated, thus, we have that with probability at least $1 - n^{-10}$,

$$|L_i \cap \mathbf{A}| \leq \frac{|L_i|}{2} + O(\sqrt{|L_i|}\log n) \qquad \text{and} \qquad |L_i \cap \overline{\mathbf{A}}| \leq \frac{|L_i|}{2} + O(\sqrt{|L_i|}\log n).$$

Thus, we may union bound over all $2t$ events, for the desired result. ■

Lemma 4.3.18. *We have that:*

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\neg\boldsymbol{\mathcal{E}}_B \wedge \boldsymbol{\mathcal{E}}_F] = o(1).$$

Proof: We first note that because of Claim 4.3.17, we have:

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\neg\boldsymbol{\mathcal{E}}_B \wedge \boldsymbol{\mathcal{E}}_F] = \sum_{\substack{A\subset[n]\\ \mathcal{E}_Q \text{ satisfied}}} \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\mathbf{A}=A] \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\neg\boldsymbol{\mathcal{E}}_B \wedge \boldsymbol{\mathcal{E}}_F \mid \mathbf{A}=A] + o(1).$$

So consider a fixed set $A \subset [n]$ which satisfies event $\mathcal{E}_Q$. Additionally, we have:

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\neg\boldsymbol{\mathcal{E}}_B \wedge \boldsymbol{\mathcal{E}}_F \mid \mathbf{A}=A] = \sum_{\substack{V_L\subset[t]\\ |V_L|\le\frac{n}{\log^4 n}}} \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\mathbf{V}_L = V_L \mid \mathbf{A}=A] \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\neg\boldsymbol{\mathcal{E}}_B \mid \mathbf{A}=A, \mathbf{V}_L = V_L]$$

Thus, it suffices to prove that for all $A \subset [n]$ which satisfy $\mathcal{E}_Q$ and $V_L \subset [t]$ of size at most $\frac{n}{\log^4 n}$, $\mathbf{Pr}[\neg\boldsymbol{\mathcal{E}}_B \mid \mathbf{A}=A, \mathbf{V}_L = V_L] = o(1)$. In fact, once we condition on $\mathbf{A}=A$ and $\mathbf{V}_L = V_L$, we have:

$$\mathbf{B} = \sum_{i\in V_L} (-1)^{\boldsymbol{y}_i}\left(|L_i \cap A| - |L_i \cap \overline{A}|\right),$$

which is a sum of independent random variables. Additionally, since $\boldsymbol{y}_i$ is the indicator random variable for $\boldsymbol{v}_i \in A$ conditioned on $\boldsymbol{v}_i$ being a lone vertex, we have each $\boldsymbol{y}_i$ is independent and is 1 with probability p_i, where:

$$p_i = \frac{|L_i\cap A|\left(\frac{n}{2} - |L_i \cap A|\right)}{|L_i|\cdot\frac{n}{2} - |L_i\cap A|^2 - |L_i\cap\overline{A}|^2} = \frac{1}{2} \pm O\left(\frac{\log n}{\sqrt{n}}\right).$$

Thus, we have:

$$\mathop{\mathbf{E}}_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\mathbf{B} \mid \mathbf{A}=A, \mathbf{V}_L = V_L] = |V_L|\cdot O(\log^2 n) = O\left(\frac{n}{\log^2 n}\right).$$

Additionally, each variable can contribute $O(\sqrt{|L_i|}\log n)$ to the sum, so via a standard Chernoff bound, noting the fact that $\sum_{i\in V_L}|L_i|\log^2 n \le \frac{n^2}{\log^3 n}$, we have that $\boldsymbol{\mathcal{E}}_B$ is satisfied with high probability. ■

4.3.1.6 Proof of the Good Outcomes Lemma: Lemma 4.3.3

We may divide $v_1, \ldots, v_t$ into three sets: 1) V_E contain the indices $i \in [t]$ whose responses v_i which are edges, 2) V_L contain the indices $i \in [t]$ whose responses v_i are vertices, and 3) V_T contain the indices $i \in [t]$ whose responses v_i are $\emptyset$. We let:

$$\Pr_{\substack{\mathbf{G}\sim\mathcal{G}_1\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_O \wedge \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] = \mathcal{Y} \qquad \Pr_{\substack{\mathbf{G}\sim\mathcal{G}_2\\ \boldsymbol{v}_1,\ldots,\boldsymbol{v}_t}}[\boldsymbol{\mathcal{E}}_O \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] = \mathcal{N}.$$

We note that for a fixed $\mathbf{A}$ the values of $\boldsymbol{v}_i$ are independent. Therefore, we may write:

$$\mathcal{Y} = \mathop{\mathbf{E}}_{\mathbf{A}}[\mathcal{Y}_E\cdot\mathcal{Y}_L\cdot\mathcal{Y}_T\cdot\boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] \qquad \mathcal{N} = \mathop{\mathbf{E}}_{\mathbf{A}}[\mathcal{N}_E\cdot\mathcal{N}_L\cdot\mathcal{N}_T \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}]$$

$$\mathcal{Y}_E = \prod_{i\in V_E}\Pr_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid Y(\mathbf{A})] \qquad \mathcal{N}_E = \prod_{i\in V_E}\Pr_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid N(\mathbf{A})]$$

$$\mathcal{Y}_L = \prod_{i\in V_L}\Pr_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid Y(\mathbf{A})] \qquad \mathcal{N}_L = \prod_{i\in V_L}\Pr_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid N(\mathbf{A})]$$

$$\mathcal{Y}_T = \prod_{i\in V_T}\Pr_{\boldsymbol{v}_i}[\boldsymbol{v}_i = \emptyset \mid Y(\mathbf{A})] \qquad \mathcal{N}_T = \prod_{i\in V_T}\Pr_{\boldsymbol{v}_i}[\boldsymbol{v}_i = \emptyset \mid N(\mathbf{A})]$$

where we slightly abused notation to let $\mathbf{Pr}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid Y(\mathbf{A})]$ denote the probability that the sampled response $\boldsymbol{v}_i$ is v_i conditioned on the graph $\mathbf{G}$ being from $\mathcal{G}_1$ with partition $\mathbf{A}$; i.e., $\mathbf{G} = K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}}$. Likewise, $\mathbf{Pr}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid N(\mathbf{A})]$ denotes the probability that the sampled response $\boldsymbol{v}_i$ is v_i conditioned on the graph $\mathbf{G}$ being from $\mathcal{G}_2$ with partition $\mathbf{A}$; i.e., $\mathbf{G} = K_{\mathbf{A},\overline{\mathbf{A}}}$. We now simply go through the three products in to show each is at most $1 + o(1)$. We shall prove the following claims:

Claim 4.3.19. *For any* $\mathbf{A}$ *for which* $\boldsymbol{\mathcal{E}}_{C,yes}$ *occurs, we have* $\mathcal{Y}_E \leq (1 + o(1))\mathcal{N}_E$.

Proof: Note that for any choice of A for which $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$ occurs, since the v_i's are specific edges:

$$\mathop{\mathbf{Pr}}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid Y(A)] = \frac{1}{2\binom{n/2}{2}}$$

and for any choice of A for which $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ occurs,

$$\mathop{\mathbf{Pr}}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid N(A)] = \frac{1}{(n/2)^2}.$$

Thus,

$$\frac{\mathbf{Pr}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid Y(A)]}{\mathbf{Pr}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid N(A)]} = \frac{n^2}{4} \cdot \frac{4}{n^2 - 2n} = 1 + O\left(\frac{1}{n}\right),$$

and since $|V_E| \leq \frac{n}{\log^4 n}$, we get that $\dfrac{\mathcal{Y}_E}{\mathcal{N}_E} = 1 + o(1)$. ■

Claim 4.3.20. *For any* $\mathbf{A}$ *for which* $\boldsymbol{\mathcal{E}}_{C,yes}$ *occurs, we have* $\mathcal{Y}_T \leq \mathcal{N}_T$.

Proof: Here, we have that for any set A which satisfies $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$, we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = \emptyset \mid Y(A)] = \frac{2\binom{n/2}{2} - |L_i|\frac{n}{2}}{2\binom{n/2}{2}} = 1 - \frac{2|L_i|}{n-2}$$

and similarly, for any set A which satisfies $\boldsymbol{\mathcal{E}}_{C,\text{no}}$, we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = \emptyset \mid N(A)] = \frac{(n/2)^2 - |L_i|\frac{n}{2} + |A \cap L_i||\overline{A} \cap L_i|}{(n/2)^2} \geq 1 - \frac{2|L_i|}{n}.$$

Which finishes the proof. ■

Thus, by Claims 4.3.19 and 4.3.20 we have:

$$\frac{\mathbf{E}_{\mathbf{A}}\left[\mathcal{Y}_E \cdot \mathcal{Y}_L \cdot \mathcal{Y}_T \cdot \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}\right]}{\mathbf{E}_{\mathbf{A}}\left[\mathcal{N}_E \cdot \mathcal{N}_L \cdot \mathcal{N}_T \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}\right]} \leq (1 + o(1))\frac{\mathbf{E}_{\mathbf{A}}[\mathcal{Y}_L \cdot \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}]}{\mathbf{E}_{\mathbf{A}}[\mathcal{N}_L \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}]}.$$

Therefore, it suffices to prove the following:

$$\frac{\mathbf{E}_{\mathbf{A}}[\mathcal{Y}_L \cdot \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}]}{\mathbf{E}_{\mathbf{A}}[\mathcal{N}_L \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}]} \leq 1 + o(1).$$

Suppose $\mathbf{A} \subset [n]$ satisfies $\boldsymbol{\mathcal{E}}_{C,\text{yes}}$, then if v_i is a vertex response at query L_i. We have:

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid Y(A)] &= \frac{2}{n-2}\left(1 - \frac{|L_i|}{n} + (-1)^{\mathbf{Y}_i}\left(\frac{|L_i \cap \mathbf{A}| - |L_i \cap \overline{\mathbf{A}}|}{n}\right)\right) \\ &= \frac{2}{n-2}\left(1 - \frac{|L_i|}{n}\right)(1 + \mathbf{Z}_i),\end{aligned}$$

where:

$$\mathbf{Z}_i = c_i(-1)^{\mathbf{Y}_i}\left(\frac{|L_i \cap \mathbf{A}| - |L_i \cap \overline{\mathbf{A}}|}{n}\right),$$

where $c_i = \dfrac{1}{1-|L_i|/n} \leq 1+o(1)$, since $|L_i| \ll \frac{n}{\log n}$, and $\mathbf{Y}_i$ is the indicator random variable for $v_i \in \mathbf{A}$. Thus, we may simplify:

$$\mathop{\mathbf{E}}_{\mathbf{A}}[\mathcal{Y}_L \cdot \boldsymbol{\mathcal{E}}_B \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}] = \left(\frac{2}{n-2}\right)^{|V_L|}\left(1-\frac{|L_i|}{n}\right)^{|V_L|} \mathop{\mathbf{E}}_{\mathbf{A}}\left[\boldsymbol{\mathcal{E}}_B \prod_{i \in V_L}(1+\mathbf{Z}_i) \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}\right].$$

Similarly, suppose $\mathbf{A} \subset [n]$ satisfies $\boldsymbol{\mathcal{E}}_{C,\text{no}}$, then if v_i is a vertex response at query L_i, we have:

$$\mathop{\mathbf{Pr}}_{\boldsymbol{v}_i}[\boldsymbol{v}_i = v_i \mid N(A)] = \frac{2}{n}\left(1-\frac{|L_i|}{n}\right)(1+\mathbf{S}_i),$$

where we let $\mathbf{S}_i$ be the random variable:

$$\mathbf{S}_i = c_i(-1)^{\mathbf{Y}_i}\left(\frac{|L_i \cap \overline{\mathbf{A}}| - |L_i \cap \mathbf{A}|}{n}\right),$$

Therefore, we have:

$$\mathop{\mathbf{E}}_{\mathbf{A}}[\mathcal{N}_L \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] = \left(\frac{2}{n}\right)^{|V_L|}\left(1-\frac{|L_i|}{n}\right)^{|V_L|} \mathop{\mathbf{E}}_{\mathbf{A}}\left[\prod_{i \in V_L}(1+\mathbf{S}_i) \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}\right].$$

We note that since $|V_L| \leq \frac{n}{\log^4 n}$, we finish off the proof with the following two claims.

Claim 4.3.21.

$$\mathop{\mathbf{E}}_{\mathbf{A}}\left[\boldsymbol{\mathcal{E}}_B \prod_{i \in V_L}(1+\mathbf{Z}_i) \mid \boldsymbol{\mathcal{E}}_{C,yes}\right] \leq 1+o(1).$$

Claim 4.3.22.

$$\mathop{\mathbf{E}}_{\mathbf{A}}\left[\prod_{i \in V_L}(1+\mathbf{S}_i) \mid \boldsymbol{\mathcal{E}}_{C,no}\right] \geq 1-o(1)$$

Proof of Claim 4.3.21:

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{A}}\left[\boldsymbol{\mathcal{E}}_B \prod_{i \in V_L}(1+\mathbf{Z}_i) \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}\right] &\leq \mathop{\mathbf{E}}_{\mathbf{A}}\left[\boldsymbol{\mathcal{E}}_B \cdot e^{\sum_{i \in V_L}\mathbf{Z}_i} \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}\right] \\ &\leq e^{\frac{1}{\log n}} = 1+o(1).\end{aligned}$$

Where the last inequality follows from the fact that $\boldsymbol{\mathcal{E}}_B$ occurs. ■

Proof of Claim 4.3.22: Recall that

$$\mathbf{S}_i = c_i(-1)^{\mathbf{Y}_i}\left(\frac{|L_i \cap \overline{\mathbf{A}}| - |L_i \cap \mathbf{A}|}{n}\right),$$

therefore, by Chernoff bound (for negative correlations) we have that with probability at least $1 - \frac{1}{n^{10}}$, $|\mathbf{S}_i| \leq O\left(\frac{\log n}{\sqrt{n}}\right)$. We let $\mathbf{S}'_i$ be the random variable which is equal to $\mathbf{S}_i$ when $|\mathbf{S}_i| \leq O(\frac{\log n}{\sqrt{n}})$ and $-2n$ otherwise. Via a very similar analysis to Claim A.1 from [CWX17a], we have:

$$\mathop{\mathbf{E}}_{\mathbf{A}}\left[\prod_{i \in V_L} (1 + \mathbf{S}_i) \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}\right] \geq (1 - o(1)) \left(1 + \sum_{i \in V_L} \mathop{\mathbf{E}}_{\mathbf{A}}[\mathbf{S}'_i \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}]\right).$$

We now evaluate each $\mathbf{E}_{\mathbf{A}}[\mathbf{S}'_i \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}]$ for $i \in V_L$ individually. We have:

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{A}}[\mathbf{S}'_i \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] &\geq \mathop{\mathbf{E}}_{\mathbf{A}}[\mathbf{S}_i \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] + (-2n - c_i) \mathop{\mathbf{Pr}}_{\mathbf{A}}\left[|\mathbf{S}_i| > O\left(\frac{\log n}{\sqrt{n}}\right) \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}\right] \\ &\geq \mathop{\mathbf{E}}_{\mathbf{A}}[\mathbf{S}_i \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] - O\left(\frac{1}{n^9}\right).\end{aligned}$$

Assume that v_i is in component C_j, and note that since $\mathbf{A}$ and $\overline{\mathbf{A}}$ are inter-changeable,

$$\mathop{\mathbf{Pr}}_{\mathbf{A}}[v_i \in \mathbf{A} \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] = \mathop{\mathbf{Pr}}_{\mathbf{A}}[v_i \in \overline{\mathbf{A}} \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}] = \frac{1}{2}.$$

Now we have that,

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{A}}[\mathbf{S}_i \mid \boldsymbol{\mathcal{E}}_{A,\text{no}}] &\geq \frac{c_i}{n} \sum_{k \in L_i \setminus C_j} \mathop{\mathbf{E}}_{\mathbf{A}}\left[(-1)^{\mathbf{Y}_i}(-1)^{\mathbf{Y}_k} \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}\right] - O\left(\frac{\log n}{n}\right) \\ &= \frac{c_i}{n} \sum_{k \in L_i \setminus C_j} \left(2 \mathop{\mathbf{Pr}}_{\mathbf{A}}[\mathbf{Y}_k = 1 \mid \mathbf{Y}_i = 1; \boldsymbol{\mathcal{E}}_{C,\text{no}}] - 1\right) - O\left(\frac{\log n}{n}\right),\end{aligned}$$

where we used the fact that $|C_i| \leq \log n$, as well as the fact that $\mathbf{A}$ and $\overline{\mathbf{A}}$ are interchangeable. Since $|V_L| \leq \frac{n}{\log^4 n}$ and $|L_i| \leq \frac{n}{\log n}$ for each $i \in V_L$ (otherwise, we would have observed an edge), it suffices to prove that $\mathbf{Pr}_{\mathbf{A}}[\mathbf{Y}_k = 1 \mid \mathbf{Y}_i = 1; \boldsymbol{\mathcal{E}}_{C,\text{no}}] \geq \frac{1}{2} - \frac{\log^4 n}{n}$. This is indeed true, since $\sum_{i=1}^{\alpha} |C_i| \leq \frac{n}{\log^4 n}$ and $|C_i| \leq \log n$ (see Lemma A.1.1). ■

Putting everything together, we have:

$$\frac{\mathbf{E}_{\mathbf{A}}[\mathcal{Y}_L \cdot \boldsymbol{\mathcal{E}}_B \cdot \boldsymbol{\mathcal{E}}_Q \mid \boldsymbol{\mathcal{E}}_{C,\text{yes}}]}{\mathbf{E}_{\mathbf{A}}[\mathcal{N}_L \mid \boldsymbol{\mathcal{E}}_{C,\text{no}}]} \leq \left(\frac{n}{n-2}\right)^{|V_L|} \frac{1 + o(1)}{1 - o(1)} \leq 1 + o(1).$$

Chapter 5

General Separation between Tolerant Testing and Intolerant Testing

The question of whether tolerant testing is strictly harder than standard testing was explicitly studied in the work of Fischer and Fortnow [FF06]. They showed that there exists a property $\mathcal{P} \subseteq \{0,1\}^n$ that admits a tester with constant query complexity (independent of n), but every tolerant tester for $\mathcal{P}$ has to query $\Omega(n/\log n)$ many bits. In this chapter we provide a stronger separation. We construct a family of properties testable using number of queries independent of n, but require much more than $\Omega(n/\log n)$ in order to tolerantly test. Specifically, the main result in this chapter is the following:

Theorem 5.0.1 (informal restatement of Theorem 5.5.1)**.** *For any constant integer $\ell \in \mathbb{N}$, there exist a property of boolean strings $\mathcal{P} \subseteq \{0,1\}^n$ and a constant $\varepsilon_1 \in (0,1)$ such that $\mathcal{P}$ is ε_0-testable for any $\varepsilon_0 > 0$ with a number of queries independent of n, but for any $\varepsilon_0 \in (0, \varepsilon_1)$, every $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{P}$ requires $\Omega(n/\mathrm{polylog}^{(\ell)} n)$ many queries.*

The main tool used toward proving their result was designing short *probabilistically checkable proof of proximity* (PCPP). We refer the reader to Chapter 2.3 for formal definitions.

5.1 Overview and Techniques

PCPPs were introduced independently by Ben-Sasson *et al.* [BGH+06] and Dinur and Reingold [DR06]. In the PCPP setting, a verifier is given oracle access to both an input x and a proof π. It should make a few (e.g., constant) number of queries to both oracles to ascertain whether $x \in \mathcal{L}$. Since the verifier can only read a few of the input bits, we only require that it rejects inputs that are *far* (in Hamming distance) from $\mathcal{L}$, no matter what proof π is provided. PCPPs are highly instrumental in the construction of standard PCPs. Indeed, using modern terminology, both the original algebraic construction of PCPs [ALM+98] (see also [BGH+06]) as well as Dinur's [Din07] combinatorial proof utilize PCPPs.

By combining the seminal works of Ben-Sasson and Sudan [BSS08] and Dinur [Din07], one can obtain PCPs and PCPPs with only poly-logarithmic (multiplicative) overhead. More specifically, the usual benchmark for PCPPs is with respect to the `CircuitEval` problem, in which the verifier is given explicit access to a circuit C and oracle access to both an input x and a proof π, and needs to verify that x is close to the set $\{x' : C(x') = 1\}$. The works of

[BSS08, Din07] yield a PCPP whose length is quasilinear in the size $|C|$ of the circuit C.[1]

Our main technical result is designing a property $\mathcal{P} \subseteq \{0,1\}^n$ such that any testing algorithm for $\mathcal{P}$ requires $\Omega(n)$ queries, while $\mathcal{P}$ admits a constant query PCPP with almost linear proof length. This construction will be used to derive our improved separation (Theorem 5.0.1).

Theorem 5.1.1 (informal restatement of Theorem 5.4.2). *For every constant integer $\ell \in \mathbb{N}$, there exists a property $\mathcal{P} \subseteq \{0,1\}^n$ such that any testing algorithm for $\mathcal{P}$ requires $\Omega(n)$ many queries, while $\mathcal{P}$ admits a (constant query)* PCPP *system with proof length $O(n \cdot \log^{(\ell)}(n))$.*

We remark that all such maximally hard properties cannot have constant-query PCPP proof-systems with a *sub-linear* length proof string (see Proposition 2.3.3), leaving only a small gap of $\log^{(\ell)}(n)$ on the proof length in 5.1.1.

We remark that Theorem 5.1.1 might be of interest in its own merit. Given the important connections both to constructions of efficient proof-systems, and to hardness of approximation, a central question in the area is whether this result can be improved: Do PCPPs with only a *constant* overhead exist? In a recent work, Ben Sasson *et al.* [BKK+16] construct PCPs with constant overhead, albeit with very large query complexity (as well as a non-uniform verification procedure).[2] To verify that $C(x) = 1$ the verifier needs to make $|C|^\delta$ queries, where $\delta > 0$ can be any fixed constant.

Given the lack of success (despite the significant interest) in constructing constant-query PCPPs with constant overhead, it may be the case that there exist languages that do not have such efficient PCPPs. A natural class of candidate languages for which such PCPPs may not exist are languages for which it is *maximally* hard to test whether $x \in \mathcal{L}$ or is far from such, *without* a PCPP proof. In other words, languages (or rather properties) that do not admit sub-linear query testers. Thus, one might investigate the following question:

> *Supposing that $\mathcal{L}$ requires $\Omega(n)$ queries for every (property) tester, must any constant-query* PCPP *for $\mathcal{L}$ have proof length $n \cdot (\log n)^{\Omega(1)}$?*

Theorem 5.1.1 answers the above question negatively, by constructing a property that is maximally hard for testing, while admitting a very short PCPP.

5.1.1 Techniques

Central to our construction are (univariate) polynomials over a finite field $\mathbb{F}$. A basic fact is that a random polynomial $p : \mathbb{F} \to \mathbb{F}$ of degree (say) $|\mathbb{F}|/2$, evaluated at any set of at most $|\mathbb{F}|/2$ points, looks exactly the same as a totally random function $f : \mathbb{F} \to \mathbb{F}$. This is despite the fact that a random function is very far (in Hamming distance) from the set of low degree polynomials. Indeed, this is the basic fact utilized by Shamir's secret sharing scheme [Sha79].

Thus, the property of being a low degree polynomial is a hard problem to decide for any tester, in the sense that such a tester must make $\Omega(|\mathbb{F}|)$ queries to the truth table of the function in order to decide. Given that, it seems natural to start with this property in order to prove 5.1.1. Here we run into two difficulties. First, the property of being a low degree polynomial is defined over a large alphabet, whereas we seek a property over boolean strings. Second, the best known PCPPs for this property have quasi-linear length [BSS08], which falls short of our goal.

[1] Note that a PCPP for `CircuitEval` can be easily used to construct a PCP for `CircuitSAT` with similar overhead (see [BGH+06, Proposition 2.4]).

[2] Although it is not stated in [BKK+16], we believe that their techniques can also yield PCPPs with similar parameters.

To cope with these difficulties, our approach is to use composition, or more accurately, an iterated construction. The main technical contribution of this paper lies in the mechanism enabling this iteration. More specifically, rather than having the property contain the explicit truth table of the low degree polynomial p, we would like to use a more redundant representation for encoding each value $p(\alpha)$. This encoding should have several properties:

- It must be the case that one needs to read (almost) the entire encoding to be able to decode $p(\alpha)$. This feature of the encoding, which we view as a secret-sharing type of property, lets us obtain a hard to test property over boolean strings.
- The encoding need not be efficient, and in fact it will be made long enough to eventually subsume the typical length of a PCPP proof-string for the low degree property, when calculated with respect to an *unencoded* input string.
- Last but not least, we need the value to be decodable using very few queries, when given access to an auxiliary PCP-like proof string. This would allow us to "propagate" the PCPP verification of the property across iterations.

In more detail, we would like to devise a (randomized) encoding of strings in $\{0,1\}^k$ by strings in $\{0,1\}^m$. The third requirement listed above can be interpreted as saying that given oracle access to $v \in \{0,1\}^m$ and explicit access to a value $w \in \{0,1\}^k$, it will be possible verify that v indeed encodes w using a PCPP-like scheme, i.e. by providing a proof that can be verified with a constant number of queries. We refer to this property as a *probabilistically checkable unveiling (*PCU*)*[3]. Note that in our setting a single value w may (and usually will) have more than one valid encoding.

Going back to the first requirement of the encoding, we demand that without a proof, one must query at least $\Theta(m)$ bits of v to obtain *any* information about the encoded w, or even discern that v is indeed a valid encoding of some value. Given this combination of requirements, we refer to the verification procedure as a *Probabilistically Checkable Unveiling of a Shared Secret (*PCUSS*)*.

Low degree polynomials can be used to obtain a PCUSS based on Shamir's secret sharing scheme. More specifically, to encode a k bit string w, we take a random polynomial whose values on a subset $H \subseteq \mathbb{F}$ are exactly equal to the bits of w. However, we provide the values of this polynomial only over the sub domain $\mathbb{F} \setminus H$. Then, the encoded value is represented by the (interpolated) values of g over H, which admit a PCU scheme. On the other hand, the "large independence" feature of polynomials makes the encoded value indiscernible without a supplied proof string, unless too many of the values of g over $\mathbb{F} \setminus H$ are read, thus allowing for a PCUSS.

This construction can now be improved via iteration. Rather than explicitly providing the values of the polynomial, they will be provided by a PCUSS scheme. Note that the PCUSS scheme that we now need is for strings of a (roughly) exponentially smaller size. The high level idea is to iterate this construction ℓ times to obtain the ℓ iterated log function in our theorems.

At the end of the recursion, i.e., for the smallest blocks at the bottom, we utilize a linear-code featuring both high distance and high dual distance, for a polynomial size PCUSS of the encoded value. This is the only "non-constructive" part in our construction, but since the relevant block size will eventually be less than $\log\log(n)$, the constructed property will still be uniform with polynomial calculation time (the exponential time in $\text{poly}(\log\log(n))$, needed to construct the linear-code matrix, becomes negligible).

Our PCUSS in particular provides a property that is hard to test (due to its shared secret feature), and yet has a near-linear PCPP through its unveiling, thereby establishing Theorem 5.1.1.

[3]In fact, we will use a stronger variant where the access to w is also restricted.

We utilize this property for separation results in a similar manner to [FF06] and [DRTV18], by considering a weighted version of a "PCPP with proof" property, where the proof part holds only a small portion of the total weight. The PCPP proof part enables a constant query test, whereas if the PCPP proof is deleted.

5.2 Code Ensembles

It will be necessary for us to think of a generalized definition of an encoding, in which each encoded value has multiple legal encodings.

Definition 5.2.1 (Code ensemble)**.** A *code ensemble* is a function $\mathcal{E} : \Sigma^k \to 2^{\Sigma^m}$. Namely, every $x \in \Sigma^k$ has a set of its valid encodings from Σ^m. We define the distance of the code ensemble as

$$\min_{x \neq x' \in \{0,1\}^k} \ \min_{(v,u) \in \mathcal{E}(x) \times \mathcal{E}(x')} \operatorname{dist}(v,u).$$

It is useful to think of a code ensemble $\mathcal{E} : \Sigma^k \to 2^{\Sigma^m}$ as a *randomized mapping*, that given $x \in \Sigma^k$, outputs a uniformly random element from the set of encodings $\mathcal{E}(x)$. Using the above we can define a *shared secret* property. In particular, we use a strong information theoretic definition of a shared secret, in which $o(m)$ bits do not give *any information at all* about the encoded value. Later on, we construct code ensembles with a shared secret property.

Definition 5.2.2 (Shared Secret)**.** For $m, k \in \mathbb{N}$ and a constant $\zeta > 0$, we say that a code ensemble $\mathcal{C} : \{0,1\}^k \to 2^{(\{0,1\}^m)}$ has a ζ*-shared secret* property it satisfies the following. For any $Q \subseteq [m]$ of size $|Q| \leq \zeta m$, any $w, w' \in \{0,1\}^k$ such that $w \neq w'$, and any $t \in \{0,1\}^{|Q|}$ it holds that

$$\Pr_{\boldsymbol{v} \sim \mathcal{C}(w)}[\boldsymbol{v}|_Q = t] \ = \Pr_{\boldsymbol{v}' \sim \mathcal{C}(w')}[\boldsymbol{v}'|_Q = t].$$

Namely, for any $w \neq w'$ and any $Q \subseteq [m]$ of size at most ζm, the distribution obtained by choosing a uniformly random member of $\mathcal{C}(w)$ and considering its restriction to Q, is identical to the distribution obtained by choosing a uniformly random member of $\mathcal{C}(w')$ and considering its restriction to Q.

5.2.1 A construction of a hard code ensemble

We describe a construction of a code ensemble for which a linear number of queries is necessary to verify membership or to decode the encoded value. This code will be our *base* code in the iterative construction. The existence of such a code ensemble is proved probabilistically, relying on the following simple lemma.

Lemma 5.2.3. *Fix constant $\alpha, \beta > 0$ where $\beta \log(e/\beta) < \alpha$. Let $s, t \in \mathbb{N}$ so that $s \leq (1-\alpha)t$. Then, with probability $1 - o(1)$, a sequence of s uniformly random vectors $\{v_1, \ldots, v_s\}$ from $\{0,1\}^t$ is linearly independent, and corresponds to a β-distance linear code.*

Proof: The proof follows from a straightforward counting argument. If we draw s uniformly random vectors $v_1, \ldots, v_s \in \{0,1\}^t$, then each non-trivial linear combination of them is in itself a uniformly random vector from $\{0,1\}^t$, and hence has weight less than β with probability at most

$$2^{-t} \cdot \binom{t}{\beta t} \leq 2^{-t} \left(\frac{et}{\beta t}\right)^{\beta t} = 2^{-t} \cdot 2^{\beta \log(e/\beta) t} = 2^{(\gamma - 1)t},$$

where we set $\gamma = \beta \log(e/\beta) < \alpha$.

By a union bound over all $2^s \leq 2^{(1-\alpha)t}$ possible combinations, the probability that there exists a linear combination with weight less than β is at most $2^{(\gamma-\alpha)t} = o(1)$. If this is not the case, then $v_1, \ldots, v_s$ are linearly independent, and moreover, $\{v_1, \ldots, v_s\}$ corresponds to a β-distance linear code (where we use the fact that the distance of a linear code is equal to the minimal Hamming weight of a non-zero codeword). ■

Our construction makes use of a sequence of vectors that correspond to a high-distance and high-dual distance code, as described below.

Definition 5.2.4 (Hard code ensemble $\mathcal{H}_k$)**.** Let $k \in \mathbb{N}$ and let $\{v_1, \ldots, v_{3k}\}$ be a sequence of vectors in $\{0,1\}^{4k}$ such that $\mathrm{Span}\{v_1, \ldots, v_{3k}\}$ is a 1/30-distance code, and that $\mathrm{Span}\{v_{k+1}, \ldots, v_{3k}\}$ is a 1/30-dual distance code. Let

$$A = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_{3k} \\ | & & | \end{bmatrix}.$$

We define the code ensemble $\mathcal{H}_k : \{0,1\}^k \to 2^{\{0,1\}^{4k}}$ as

$$\mathcal{H}_k(w) = \{Au \ : \ u \in \{0,1\}^{3k} \text{ where } \ u|_{\{1,\ldots,k\}} = w\},$$

where all operations are over GF(2).

The next lemma states that a collection of random vectors $\{v_1, \ldots, v_{3k}\}$ in $\{0,1\}^{4k}$ satisfies the basic requirements of a code ensemble $\mathcal{H}_k$ with high probability (that is, with probability tending to one as $k \to \infty$), and hence such a code ensemble exists.

Lemma 5.2.5. *A set $\{v_1, \ldots, v_{3k}\}$ of random vectors in $\{0,1\}^{4k}$ satisfies with high probability the following two conditions:* $\mathrm{Span}\{v_1, \ldots, v_{3k}\}$ *is a* 1/30*-distance code, and* $\mathrm{Span}\{v_{k+1}, \ldots, v_{3k}\}$ *is a* 1/10*-dual distance code. In particular, for all k large enough the code ensemble $\mathcal{H}_k$ exists.*

Proof: We apply Lemma 5.2.3 multiple times. First, picking $t = 4k$, $s = 3k$, $\alpha = 1/4$, and $\beta = 1/30$, we conclude that $v_1, \ldots, v_{3k}$ with high probability correspond to a 1/30-distance code.

To show that with high probability the code spanned by the last $2k$ vectors has high dual distance, we compare the following two processes, whose output is a linear subspace of $(\mathrm{GF}(2))^{4k}$, that we view as a code: (i) Choose $2k$ vectors and return their span. (ii) Choose $4k - 2k = 2k$ vectors and return the dual of their span. Conditioning on the chosen $2k$ vectors being linearly independent, the output distributions of these two processes are identical. Indeed, by a symmetry argument it is not hard to see that under the conditioning, the linear subspace generated by Process (i) is uniformly distributed among all rank-$2k$ subspaces V of $(\mathrm{GF}(2))^{4k}$. Now, since we can uniquely couple each such V with its dual $V^\perp$ (also a rank-$2k$ subspace) and since $V = (V^\perp)^\perp$, this means that the output distribution of Process (ii) is uniform as well.

However, it follows again from Lemma 5.2.3 (with $t = 4k$, $s = 2k$, $\alpha = 1/2$, and any $\beta > 0$ satisfying the conditions of the lemma) that the chosen $2k$ vectors are independent with high probability. This means that (without the conditioning) the output distributions of Process (i) and Process (ii) are $o(1)$-close in variation distance. Applying Lemma 5.2.3 with $t = 4k$, $s = 2k$, $\alpha = 1/2$, and $\beta = 1/10$ we get that the distance of the code generated by Process (i) is at least $\beta = 1/10$ with high probability. However, the latter distance equals by definition to the dual distance of the code generated by Process (ii). By the closeness of the distributions, we conclude that the dual distance of Process (i) is also at least 1/10 with high probability. ■

We next state a simple but important observation regarding membership verification.

Observation 5.2.6. *Once a matrix A with the desired properties is constructed (which may take* $\exp(k^2)$ *time if we use brute force), given $w \in \{0,1\}^k$, the membership of v in $\mathcal{H}_k(w)$ can be verified in* $\mathrm{poly}(k)$ *time (by solving a system of linear equations over* GF(2)*).*

5.3 PCUs and PCUSSs

Next, we define the notion of Probabilistically Checkable Unveiling (PCU). This notion is similar to PCPP, but here instead of requiring our input to satisfy a given property, we require our input to encode a value $w \in \{0,1\}^k$ (typically using a large distance code ensemble). We then require that given the encoded value w, it will be possible to prove in a PCPP-like fashion that the input is indeed a valid encoding of w.

Definition 5.3.1 (PCU)**.** Fix $m, t, k \in \mathbb{N}$, and let $\mathcal{C} : \{0,1\}^k \to 2^{\{0,1\}^m}$ be a code ensemble. We say that $\mathcal{C}$ has a $q(\varepsilon, \delta)$-query, length-t PCU if the following holds. There exists a verification algorithm V that takes as inputs $\varepsilon, \delta > 0$, $m \in \mathbb{N}$, and $w \in \{0,1\}^k$, makes at most $q(\varepsilon, \delta)$ queries to the strings $v \in \{0,1\}^m$ and $\pi \in \{0,1\}^t$, and satisfies the following:

1. If $v \in \mathcal{C}(w)$, then there exists a proof $\pi = \mathbf{Proof}_{\mathcal{C}}(v) \in \{0,1\}^t$ such that for every $\varepsilon, \delta > 0$, the verifier V accepts with probability 1.

2. If $\mathrm{dist}(v, \mathcal{C}(w)) > \varepsilon$, then for every alleged proof $\pi \in \{0,1\}^t$, the verifier V rejects v with probability greater than δ.

In order to facilitate the proof of the main theorem, we utilize a more stringent variant of the above PCU definition. Recall that Spiel is a constant rate and relative distance code, in which membership can be decided using quasi-linear size circuit; See Chapter 2.6 and Theorem 2.6.2 therein.

Instead of supplying $w \in \{0,1\}^k$ to the algorithm, we supply oracle access to a a string $\tau \in \{0,1\}^{100k}$ that is supposed to represent $\mathsf{Spiel}(w)$, along with the proof π, and the algorithm only makes $q(\varepsilon, \delta)$ queries to the proof string π, the original encoding v *and* the string τ. For cases where $v \in \mathcal{C}(w)$, we use $\mathbf{Value}(v)$ to denote $\mathsf{Spiel}(w)$.

Definition 5.3.2 (Spiel-PCU)**.** Fix $m, t, k \in \mathbb{N}$, and let $\mathcal{C} : \{0,1\}^k \to 2^{\{0,1\}^m}$ be a code ensemble. We say that $\mathcal{C}$ has a $q(\varepsilon, \delta)$-query, length-t Spiel-PCU if the following holds. There exists a verification algorithm V that takes as inputs $\varepsilon, \delta > 0$, $m \in \mathbb{N}$, makes at most $q(\varepsilon, \delta)$ queries to the strings $v \in \{0,1\}^m$, $\tau \in \{0,1\}^{100k}$ and $\pi \in \{0,1\}^t$, and satisfies the following:

1. If there exists $w \in \{0,1\}^k$ for which $v \in \mathcal{C}(w)$ and $\tau = \mathbf{Value}(v) = \mathsf{Spiel}(w)$, then there exists a proof $\pi = \mathbf{Proof}_{\mathcal{C}}(v) \in \{0,1\}^t$ such that for every $\varepsilon, \delta > 0$, the verifier V accepts with probability 1.

2. If for every $w \in \{0,1\}^k$ either $\mathrm{dist}(\tau, \mathsf{Spiel}(w)) > \varepsilon$ or $\mathrm{dist}(v, \mathcal{C}(w)) > \varepsilon$, then for every alleged proof $\pi \in \{0,1\}^t$, the verifier V rejects v with probability greater than δ.

Note that a code ensemble admitting a Spiel-PCU automatically admits a PCU. Indeed, given the string w, an oracle for $\mathsf{Spiel}(w)$ can be simulated.

The following lemma states the existence of Spiel-PCU for efficiently computable code ensembles, and will be used throughout this work. The proof follows from Lemma 2.3.2 together with a simple concatenation argument.

Lemma 5.3.3. *Let $k, m, t \in \mathbb{N}$ be such that $t \geq m$, and let $\mathcal{C} : \{0,1\}^k \to 2^{\{0,1\}^m}$ be a code ensemble. If given $w \in \{0,1\}^k$ and $v \in \{0,1\}^m$, it is possible to verify membership of v in $\mathcal{C}(w)$ using a circuit of size t, then there is a $q(\varepsilon, \delta)$-query, length-t'* Spiel-PCU *for $\mathcal{C}$ where $t' = t \cdot \mathrm{polylog}\, t$.*

Proof: Assume without the loss of generality that $m \geq |\mathsf{Spiel}(0^k)|$. Let $\xi = \left\lfloor \frac{m}{|\mathsf{Spiel}(0^k)|} \right\rfloor$ (note that $\xi \geq 1$), and define

$$\mathcal{C}_{eq} \stackrel{\text{def}}{=} \left\{ v \sqcup (\mathsf{Spiel}(w))^\xi \mid \exists w \in \{0,1\}^k \text{ for which } v \in \mathcal{C}(w) \right\},$$

where $(\mathsf{Spiel}(w))^\xi$ denotes the ξ-times concatenation of $\mathsf{Spiel}(w)$.

For any string u it is possible to check, using a quasilinear size circuit (see [Spi96]), that the substring that corresponds to the domain of $(\mathsf{Spiel}(w))^\xi$ is a ξ-times repetition of $\mathsf{Spiel}(w)$ for some w. After doing so, we decode w using a quasilinear size circuit (as in [Spi96]), and then, by the premise of the lemma, we can verify membership in $\mathcal{C}(w)$ using a circuit of size t. Therefore, membership in $\mathcal{C}_{eq}$ can be decided using a $O(t)$ size boolean circuit, and therefore by Lemma 2.3.2 admits a PCPP system whose proof length is quasilinear in t.

Given an input v to Spiel-PCU, let $v' = v \sqcup (\mathsf{Spiel}(w))^\xi$ and use the PCPP system for $\mathcal{C}_{eq}$, with detection radius $\varepsilon/3$ and soundness δ, where each query to v' is emulated by a corresponding query to v or $\mathsf{Spiel}(w)$. Note that if $v \in \mathcal{C}(w)$, then $v' \in \mathcal{C}_{eq}$, so the PCPP system for $\mathcal{C}_{eq}$ will accept with probability 1.

Next, suppose that $\text{dist}(v, \mathcal{C}(w)) > \varepsilon$, and observe that this implies that v' is at least $\varepsilon/3$-far from $\mathcal{C}_{eq}$. Thus, by the soundness property of the PCPP for $\mathcal{C}_{eq}$, the verifier rejects with probability at least δ, regardless of the contents of the alleged proof π it is supplied with. ■

Next we define Probabilistically Checkable Unveiling of a Shared Secret (PCUSS).

Definition 5.3.4. For $m, k, t \in \mathbb{N}$, we say that a function $\mathcal{C} \colon \{0,1\}^k \to 2^{(\{0,1\}^n)}$ has a $q(\varepsilon, \delta)$-query, length-t PCUSS, if $\mathcal{C}$ has a shared secret property, as well as $\mathcal{C}$ has a $q(\varepsilon, \delta)$-query, length-t PCU. Similarly, when $\mathcal{C}$ has a shared secret property (for constant ζ), as well as $\mathcal{C}$ has a $q(\varepsilon, \delta)$-query, length-t Spiel-PCU, we say that $\mathcal{C}$ has a $q(\varepsilon, \delta)$-query, length-t Spiel-PCUSS.

Note that $\mathcal{C}$ admitting a Spiel-PCUSS directly implies that it admits a PCUSS with similar parameters.

The following lemma establishes the existence of a Spiel-PCUSS for $\mathcal{H}_k$, where $\mathcal{H}_k$ is the code ensemble from Definition 5.2.4.

Lemma 5.3.5. *For any $k \in \mathbb{N}$, $\mathcal{H}_k$ has a $q(\varepsilon, \delta)$-query, length-t'* Spiel-PCUSS *where $t' = \text{poly}(k)$.*

Proof: By Observation 5.2.6, given w, membership in $\mathcal{H}_k(w)$ can be checked in $\text{poly}(k)$ time, which means that there exists a polynomial size circuit that decides membership in $\mathcal{H}_k(w)$. Combining the above with Lemma 5.3.3 implies a $q(\varepsilon, \delta)$-query, length-t' Spiel-PCU where $t' = \text{poly}(k)$. By Lemma 2.6.7, the large dual distance property of $\mathcal{H}_k$ implies its shared secret property for some constant ζ, which concludes the proof of the lemma. ■

5.4 PCUSS construction

In this section we give a construction of code ensembles that admit a PCUSS. First we show that our code ensemble has a PCU with a short proof. Specifically,

Lemma 5.4.1. *For any fixed $\ell \in \mathbb{N}$ and any $k \in \mathbb{N}$, there exists $n_0(\ell, k)$ and a code ensemble $\mathcal{E}^{(\ell)} : \{0,1\}^k \to 2^{(\{0,1\}^n)}$, such that for all $n > n_0(\ell, k)$, the code ensemble $\mathcal{E}^{(\ell)}$ has a $q(\varepsilon, \delta)$-query length-t* PCU, *for $t = O(n \cdot \text{polylog}^{(\ell)} n)$.*

Later, we prove that our code ensemble has a shared secret property, which implies that it has a PCUSS (which implies Theorem 5.1.1, as we shall show).

Theorem 5.4.2. *For any fixed $\ell \in \mathbb{N}$ and any $k \in \mathbb{N}$, there exists $n_0(\ell, k)$ and a code ensemble $\mathcal{E}^{(\ell)} : \{0,1\}^k \to 2^{(\{0,1\}^n)}$, such that for all $n > n_0(\ell, k)$, the code ensemble $\mathcal{E}^{(\ell)}$ has a $q(\varepsilon, \delta)$-query length-t* PCUSS, *for $t = O(n \cdot \text{polylog}^{(\ell)} n)$.*

5.4.1 The iterated construction

Our iterative construction uses polynomials over a binary finite field $\mathrm{GF}(2^t)$. In our proof we will need to be able to implement arithmetic operations over this field efficiently (i.e., in $\mathrm{poly}(t)$ time). This can be easily done given a suitable representation of the field: namely, a degree t irreducible polynomial over $\mathrm{GF}(2)$. It is unclear in general whether such a polynomial can be found in $\mathrm{poly}(t)$ time. Fortunately though, for $t = 2 \cdot 3^r$ where $r \in \mathbb{N}$, it is known that the polynomial $x^t + x^{t/2} + 1$ is irreducible over $\mathrm{GF}(2)$ (see, e.g., [Gol08, Appendix G]). We will therefore restrict our attention to fields of this form. At first glance this seems to give us a property that is defined only on a sparse set of input lengths. However, towards the end of this section, we briefly describe how to bypass this restriction.

We next formally define our iterated construction, starting with the "level-0" construction as a base case. The constants c, d in the definition will be explicitly given in the proof of Lemma 5.4.8. Additionally, for any $\ell \in \mathbb{N}$, we shall pick a large enough constant c_ℓ that satisfies several requirements for the "level-ℓ" iteration of the construction.

Definition 5.4.3 (Iterated coding ensemble)**.** For $k \in \mathbb{N}$ and $w \in \{0,1\}^k$, we define the *base code ensemble of* w (i.e., level-ℓ code ensemble of w for $\ell = 0$) as

$$\mathcal{E}_k^{(0)}(w) = \mathcal{H}_k(w).$$

Let $c, d \in \mathbb{N}$ be large enough global constants, fix $\ell > 0$, let c_ℓ be large enough, and let $\mathbb{F}$ be a finite field for which $|\mathbb{F}| \geq \max\{c_\ell, c \cdot k\}$.

We define the *level-ℓ code ensemble of* $w \in \{0,1\}^k$ *over* $\mathbb{F}$ as follows. Let $r \in \mathbb{N}$ be the smallest integer such that $(\log|\mathbb{F}|)^d \leq 2^{2\cdot 3^r}$, set $\mathbb{F}' = \mathrm{GF}\left(2^{2\cdot 3^r}\right)$ and $k' = \log|\mathbb{F}|$. Note that these satisfy the recursive requirements of a level-$(\ell-1)$ code ensemble provided that c_ℓ is large enough (specifically we require $(\log|\mathbb{F}|)^{d-1} > c$, so that $|\mathbb{F}'| \geq ck'$). Finally, let $H \subseteq \mathbb{F}$ be such that $|H| = k$, and define

$$\mathcal{E}_{\mathbb{F},k}^{(\ell)}(w) = \bigcup_{g \in \mathcal{C}_\mathbb{F}\colon\, g|_H = w} \ \bigsqcup_{\beta \in \mathbb{F}\setminus H} \mathcal{E}_{\mathbb{F}',k'}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle).$$

(Note that for $\ell = 1$ we just use $\mathcal{E}_{\mathbb{F},k}^{(1)}(w) = \bigcup_{g \in \mathcal{C}_\mathbb{F}\colon\, g|_H = w} \bigsqcup_{\beta \in \mathbb{F}\setminus H} \mathcal{E}_{k'}^{(0)}(\langle\!\langle g(\beta) \rangle\!\rangle)$).

That is, $v \in \mathcal{E}_{\mathbb{F},k}^{(\ell)}(w)$ if there exists a polynomial $g \in \mathcal{C}_\mathbb{F}$ such that $v = \bigsqcup_{\beta \in \mathbb{F}\setminus H} v_\beta$, where $v_\beta \in \mathcal{E}_{\mathbb{F}',k'}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle)$ for every $\beta \in \mathbb{F} \setminus H$ and $g|_H = w$ (where we identify the 0 and 1 elements of $\mathbb{F}$ with 0 and 1 bits respectively). When the context is clear, we sometimes omit the subscripts.

Our choice of the constants c, d, c_ℓ needs to satisfy the following conditions. The constant c is chosen such that H will not be an overly large portion of $\mathbb{F}$ (this requirement is used in Lemma 5.4.14). The constant d is needed to subsume the length of PCPP proof string which is part of the construction (this requirement is used in Lemma 5.4.8). Finally, the constant c_ℓ needs to be large enough to enable iteration (as explained in Definition 5.4.3 itself).

Let $\ell \geq 0$ be some fixed iteration. The following simple observation follows by a simple inductive argument using the definition of the level-ℓ coding ensemble, and in particular that $|\mathbb{F}'| = \mathrm{polylog}\,|\mathbb{F}|$.

Observation 5.4.4. *For $\ell > 0$, let $n = |\mathbb{F}|$ and $w \in \{0,1\}^k$. If $v \in \mathcal{E}^{(\ell)}(w)$, then $m_\mathbb{F}^{(\ell)} \stackrel{\mathrm{def}}{=} |v| = n \cdot \mathrm{poly}(\log n) \cdot \mathrm{poly}(\log\log n) \cdots \mathrm{poly}(\log^{(\ell)} n)$, where $\log^{(\ell)} n$ is the* log *function iterated ℓ times.*

When the field $\mathbb{F}$ is clear from context, we shall usually write $m^{(\ell)}$ as a shorthand for $m^{(\ell)}_{\mathbb{F}}$. The following lemma, proved in the next subsection, establishes the existence of short length Spiel-PCUs for our code ensembles.

Lemma 5.4.5. *For any $\ell \geq 0$, the code ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k}$ admits a $q(\varepsilon,\delta)$-query, length-t* Spiel-PCU *for $t = O(m^{(\ell)} \cdot \mathrm{polylog}^{(\ell)} m^{(\ell)})$*

5.4.2 Proof of Lemma 5.4.5

We start by defining the PCU proof string for a given $v \in \mathcal{E}^{(\ell)}_{\mathbb{F},k}(w)$ for some $w \in \{0,1\}^k$.

Definition 5.4.6 (The PCU Proof String)**.** For $\ell = 0$, let $v \in \mathcal{E}^{(0)}_{k}(w)$ and $\mathbf{Value}^{(0)}(v) = \mathsf{Spiel}(w)$. We define the proof string for v, $\mathbf{Proof}^{(0)}(v)$, as the one guaranteed by Lemma 5.3.5 (note that the length of $\mathbf{Proof}^{(0)}(v)$ is $\mathrm{poly}(k)$).

For $\ell > 0$, let $g \in \mathcal{C}_{\mathbb{F}}$ and $w \in \{0,1\}^k$ be such that $v \in \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathcal{E}^{(\ell-1)}(\langle\langle g(\beta)\rangle\rangle)$, $\mathbf{Value}^{(\ell)}(v) = \mathsf{Spiel}(w)$ and $g|_H = w$. In addition, set $S_v \overset{\text{def}}{=} \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathbf{Value}^{(\ell-1)}(v_\beta) = \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathsf{Spiel}(g(\beta))$. The proof string for $v \in \mathcal{E}^{(\ell)}_{\mathbb{F},k}$ is defined as follows.

$$\mathbf{Proof}^{(\ell)}(v) = S_v \sqcup \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathbf{Proof}^{(\ell-1)}(v_\beta) \sqcup \mathbf{Proof}_{\mathcal{L}}\left(S_v\right)$$

where the code ensemble $\mathcal{L} : \{0,1\}^k \to 2^{\{0,1\}^{O(|\mathbb{F}|\cdot\log|\mathbb{F}|)}}$ is defined as follows. Given $w \in \{0,1\}^k$, $S \in \mathcal{L}(w)$ if and only if there exists a polynomial $g \in \mathcal{C}_{\mathbb{F}}$ such that the following conditions are satisfied.

1. $g|_H = w$.
2. $S = \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathsf{Spiel}(g(\beta))$.

The following lemma establishes the existence of a Spiel-PCU for $\mathcal{L}$.

Lemma 5.4.7. *$\mathcal{L}$ has a $q(\varepsilon,\delta)$-query length-t* Spiel-PCU *for $t = O(|\mathbb{F}| \cdot \mathrm{polylog}\,|\mathbb{F}|)$.*

Proof: By Theorem 2.6.2, there exists a quasilinear size circuit that decodes $\mathsf{Spiel}(\alpha)$. Using such a circuit, we can decode $g(\beta)$ from S for every $\beta \in \mathbb{F}$. Then, using all the values $g(\beta)$ and w (where the i-th bit of w correspond to the value of the i-th element in H according to the ordering), we use Theorem 2.6.4 to interpolate the values and achieve a representation of a polynomial $g : \mathbb{F} \to \mathbb{F}$. If $g \in \mathcal{C}_{\mathbb{F}}$ we accept S and otherwise we reject. Since deciding if $S \in \mathcal{L}(w)$ has a quasilinear size circuit, by Lemma 5.3.3, there is a quasilinear length Spiel-PCU for $\mathcal{L}$. ■

Having defined $\mathbf{Proof}^{(\ell)}$, we first provide an upper bound on the bit length of the prescribed proof string. For $\ell > 0$, let $z^{(\ell)}_{\mathbb{F},k}$ denote the bit length of the proof for membership in $\mathcal{E}^{(\ell)}$ as defined in Definition 5.4.6, where for $\ell = 0$ we replace the (nonexistent) field $\mathbb{F}$ with $|w|$.

The following lemma, establishing the proof string's length, relies on our choice of the constant d in Definition 5.4.3. In particular, d needs to be large enough to subsume the size of $\mathbf{Proof}_{\mathcal{L}}(\cdot)$

Lemma 5.4.8. *For any $\ell \geq 0$, we have that $z^{(\ell)}_{\mathbb{F},k} = O\big(m^{(\ell)} \cdot \mathrm{polylog}^{(\ell)} m^{(\ell)}\big)$.*

Proof: The proof follows by induction on ℓ. The base case ($\ell = 0$) follows directly from the definition of $\mathcal{P}^{(0)}$ by our convention that $\log^{(0)}|w| = |w|$.

Consider $\ell > 0$, and note that since the size of S_v is $O(|\mathbb{F}|\log|\mathbb{F}|)$, the size of $\mathbf{Proof}_{\mathcal{L}}(S_v)$ is $O(|\mathbb{F}| \cdot \text{polylog}\,|\mathbb{F}|)$. By combining the above with the definition of the proof string we have

$$z^{(\ell)}_{\mathbb{F},k} \leq |\mathbb{F}| \cdot \text{polylog}\,|\mathbb{F}| + |\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}',k'}.$$

Now, assume that $z^{(\ell-1)}_{\mathbb{F}',k'} = O(m^{(\ell-1)} \cdot \text{polylog}^{(\ell-1)}|\mathbb{F}'|)$. Note that since the global constant d was chosen so that $|\mathbb{F}| \cdot |\mathbb{F}'| \geq |\mathbf{Proof}_{\mathcal{L}}(S_v)|$, we have that $|\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}',k'} \geq |\mathbb{F}| \cdot |\mathbb{F}'| \geq |\mathbf{Proof}_{\mathcal{L}}(S_v)|$. Therefore,

$$m^{(\ell)} = \Theta(|\mathbb{F}| \cdot m^{(\ell-1)}) = \Omega(|\mathbb{F}| \cdot |\mathbb{F}'|) = \Omega(|\mathbb{F}| \cdot \text{polylog}\,|\mathbb{F}|),$$

so that $|\mathbb{F}| \cdot \text{polylog}\,|\mathbb{F}| = O(m^{(\ell)})$, and

$$z^{(\ell)}_{\mathbb{F},k} = O(|\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}'}).$$

In addition, by the fact that $m_\ell = \Theta(|\mathbb{F}| \cdot m^{(\ell-1)})$ and the induction hypothesis we obtain

$$|\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}',k'} = O(|\mathbb{F}| \cdot m^{(\ell-1)} \cdot \text{polylog}^{(\ell-1)}|\mathbb{F}'|) = O(m^{(\ell)} \cdot \text{polylog}^{(\ell)}|\mathbb{F}|) = O(m^{(\ell)} \cdot \text{polylog}^{(\ell)} m_\ell).$$

So overall, we get that $z^{(\ell)}_{\mathbb{F},k} = O(m^{(\ell)} \cdot \text{polylog}^{(\ell)} m^{(\ell)})$ as required.

■

Next, for an alleged proof $\pi = \mathbf{Proof}^{(\ell)}(v)$, we use the notation $\pi|_{\text{Dom}(X)}$ to denote the restriction of π to the bits that correspond to X in π as defined in Definition 5.4.6. For example, $\pi|_{\text{Dom}(\mathbf{Value}^{(\ell-1)}(v_\beta))}$ refers to the bits that represent $\mathbf{Value}^{(\ell-1)}(v_\beta)$.

We introduce the verifier procedure for $\mathcal{E}^{(\ell)}_{\mathbb{F},k}$ (see Figure 5.1), and prove its completeness and soundness.

Lemma 5.4.9. *If there exist $w \in \{0,1\}^k$ for which $v \in \mathcal{E}^{(\ell)}_{\mathbb{F},k}(w)$, then* `Verifier-Procedure`$_{\mathcal{E}^{(\ell)}}$ *accepts v with probability 1 when supplied with oracle access to the corresponding $\mathbf{Proof}^{(\ell)}(v)$ and $\tau = \mathbf{Value}^{(\ell)}(v) = \mathsf{Spiel}(w)$.*

Proof: The proof follows by induction on ℓ. The base case follows directly from Lemma 5.3.5. Hence, the verifier for $\mathcal{E}^{(0)}$ supplied with $\mathbf{Proof}^{(0)}(v)$ as the proof oracle and $\mathbf{Value}^{(\ell)}(v)$ as the value oracle, will accept v with probability 1.

Assume that `Verifier-Procedure`$_{\mathcal{E}^{(\ell-1)}}$ accepts with probability 1 any valid encoding v' when supplied with the corresponding oracles for $\mathbf{Value}^{(\ell-1)}(v')$ and $\mathbf{Proof}^{(\ell-1)}(v')$. Let $v \in \mathcal{E}^{(\ell)}$ and write $v = \bigsqcup_{\beta \in \mathbb{F}\setminus H} v_\beta$, where there exist $w \in \{0,1\}^k$ and $g \in \mathcal{C}_{\mathbb{F}}$ such that for all $\beta \in \mathbb{F} \setminus H$, $v_\beta \in \mathcal{E}^{(\ell-1)}(g(\beta))$, where $g|_H = w$ and $\tau = \mathbf{Value}^{(\ell)}(v) = \mathsf{Spiel}(w)$. Then, by the definition of the language $\mathcal{L}$ and the first two components of $\mathbf{Proof}^{(\ell)}(v)$, Step (2a) of `Verifier-Procedure`$_{\mathcal{E}^{(\ell)}}$ will always accept. In addition, for every $\beta \in \mathbb{F} \setminus H$, we have that $v_\beta \in \mathcal{E}^{(\ell-1)}$, and therefore by the induction hypothesis, Step (2b) of `Verifier-Procedure`$_{\mathcal{E}^{(\ell)}}$ will accept the corresponding unveiling for any picked $\beta \in \mathbb{F} \setminus H$. ■

Lemma 5.4.10. *If for every $w \in \{0,1\}^k$ either $\text{dist}(\tau, \mathsf{Spiel}(w)) > \varepsilon$ or $\text{dist}(v, \mathcal{E}^{(\ell)}(w)) > \varepsilon$ (or both), then with probability greater than δ,* `Verifier-Procedure`$_{\mathcal{E}^{(\ell)}}$ *will reject v regardless of the contents of the supplied proof string.*

Proof: Let $\tau \in \{0,1\}^{100k}$ be an alleged value for v, and $\pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},k}}$ be an alleged proof string for v. We proceed by induction on ℓ. For $\ell = 0$ we use the PCU verifier for $\mathcal{E}^{(0)}$ with error

```
Verifier-Procedure$_{\mathcal{E}^{(\ell)}}$

Input: Parameters $\varepsilon, \delta \in (0,1)$, an input $v \in \{0,1\}^{m^{(\ell)}}$, an alleged value $\tau \in \{0,1\}^{100k}$ of
$v$, and an alleged proof $\pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},k}}$ for $v$.

  1. If $\ell = 0$, use the PCU for $\mathcal{E}^{(0)}$ with parameters $\varepsilon$ and $\delta$.
  2. If $\ell > 0$:
     (a) Use the PCU verifier for $\mathcal{L}$ with error $\varepsilon/300$ and soundness $\delta$, to verify the
         unveiling of $\pi|_{\mathrm{Dom}(S_v)}$, using $\tau$ as the value oracle and $\pi|_{\mathrm{Dom}(\mathbf{Proof}_{\mathcal{L}}(S_v))}$ as the
         proof oracle.
     (b) For $6/\varepsilon$ many times:
         i. Pick $\boldsymbol{\beta} \in \mathbb{F} \setminus H$ uniformly at random.
         ii. Use the PCU verifier procedure for $\mathcal{E}^{(\ell-1)}$ with parameters $\varepsilon/3$ and $2\delta$, to
             verify the unveiling of $v_{\boldsymbol{\beta}}$, using $\pi|_{\mathrm{Dom}(\mathbf{Value}^{(\ell-1)}(v_{\boldsymbol{\beta}}))}$ as the value oracle and
             $\pi|_{\mathrm{Dom}(\mathbf{Proof}^{(\ell-1)}(v_{\boldsymbol{\beta}}))}$ as the proof oracle.

If any of the stages rejected then Reject, and otherwise Accept.
```

Figure 5.1: Description of `Verifier-Procedure`$_{\mathcal{E}^{(\ell)}}$.

ε and soundness δ to check that v is a member of the code ensemble $\mathcal{E}^{(0)}$ and τ is its value. If the PCU verifier for $\mathcal{E}^{(0)}$ rejects with probability at most δ, then there exist $w \in \{0,1\}^k$ such that $\mathrm{dist}(v, \mathcal{E}^{(0)}(w)) \le \varepsilon$ and $\mathrm{dist}(\tau, \mathsf{Spiel}(w)) \le \varepsilon$, and the base case is complete.

Next assume that the lemma holds for $\ell - 1$. If the PCU verifier for $\mathcal{L}$ in Step (2a) rejects with probability at most δ, then there exist a function $g \in \mathcal{C}_{\mathbb{F}}$ and $w \in \{0,1\}^k$ for which $g|_H = w$ so that

$$\mathrm{dist}(\pi|_{\mathrm{Dom}(S_v)}, \mathsf{Spiel}(g|_{\mathbb{F}\setminus H})) \le \varepsilon/300 \qquad \text{and} \qquad \mathrm{dist}(\tau, \mathsf{Spiel}(w)) \le \varepsilon/300.$$

In particular, the leftmost inequality means that for at most $\frac{\varepsilon}{3}|\mathbb{F} \setminus H|$ of the elements $\beta \in \mathbb{F} \setminus H$, it holds that

$$\mathrm{dist}(\pi|_{\mathrm{Dom}(\mathbf{Value}^{(\ell-1)}(v_\beta))}, \mathsf{Spiel}(g(\beta)) > 1/100.$$

We refer to elements $\beta \in \mathbb{F} \setminus H$ satisfying the above inequality as *bad* elements, and to the rest as *good* elements. Let G denote the set of good elements.

Next, we show that if the loop that uses the PCU verifier for $\mathcal{E}^{(\ell-1)}$ in Step (2b) rejects with probability at most δ, then for at most an $\varepsilon/3$ fraction of the good $\beta \in \mathbb{F} \setminus H$, it holds that

$$\mathrm{dist}\left(v_\beta, \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle)\right) > \varepsilon/3.$$

Assume that there are more than $\frac{\varepsilon}{3} \cdot |G|$ good elements such that $\mathrm{dist}\left(v_\beta, \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle)\right) > \varepsilon/3$. Then, by our induction hypothesis, each of them will be rejected by the PCU verifier for $\mathcal{E}^{(\ell-1)}$ with probability more than 2δ. In addition, with probability at least $1/2$ we sample at least one such good β, and then during this iteration the verifier in Step (2b(ii)) rejects with conditional probability more than 2δ, and hence the verifier will reject with overall probability more than

δ. Summing everything up, when the input is rejected with probability at most δ,

$$\operatorname{dist}\left(v, \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle)\right) \leq \varepsilon/3 + (1-\varepsilon/3) \cdot \varepsilon/3 + (1-\varepsilon/3)^2 \cdot \varepsilon/3 \leq \varepsilon,$$

where the three summands are respectively the contribution to the distance of the bad elements, the good elements with v_β being far from any level $\ell - 1$ encoding of $\langle\!\langle g(\beta) \rangle\!\rangle$, and all the other elements. ■

The proof of Lemma 5.4.5 follows directly by combining Lemma 5.4.8, Lemma 5.4.9 and Lemma 5.4.10.

The following corollary follows directly from Lemma 5.4.5 and the definition of Spiel-PCU (Definition 5.3.2), and implies Lemma 5.4.1.

Corollary 5.4.11. *Let $\mathbb{F}$ be a finite field and $k \in \mathbb{N}$ which satisfy the requirements in Definition 5.4.3. Then, for every $\ell \geq 0$ the coding ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k} : \{0,1\}^k \to 2^{\left(\{0,1\}^{m^{(\ell)}}\right)}$ has a $q(\varepsilon, \delta)$-query, length-t* Spiel-PCU *for $t = O(m^{(\ell)} \text{polylog}^{(\ell)} m^{(\ell)})$.*

5.4.3 The Lower Bound

We turn to prove the linear query lower bound for the testability of our property. We start by defining distributions over strings of length $m^{(\ell)}$.

Distribution $\mathcal{D}^{(\ell)}_{\textbf{yes}}(w)$: Given $w \in \{0,1\}^k$, we define the distribution $\mathcal{D}^{(\ell)}_{\text{yes}}(w)$ to be the uniform distribution over elements in $\mathcal{E}^{(\ell)}(w)$.

Distribution $\mathcal{D}^{(\ell)}_{\textbf{no}}$: An element v from $\mathcal{D}^{(\ell)}_{\text{no}}$ is drawn by the following process. For $\ell = 0$, $\boldsymbol{v}$ is a uniformly random string in $\{0,1\}^{4k}$. For $\ell > 0$, we pick a uniformly random function $\boldsymbol{\lambda} : \mathbb{F} \setminus H \to \mathbb{F}$, and let v be a uniformly random element of $\bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle \boldsymbol{\lambda}(\beta) \rangle\!\rangle)$

Lemma 5.4.12. *For any $\ell \geq 0$, every $w \in \{0,1\}^k$ and $q = o(m^{(\ell)}/10^\ell)$, any algorithm making at most q queries cannot distinguish (with constant probability) between $\boldsymbol{v} \sim \mathcal{D}^{(\ell)}_{yes}(w)$ and $\boldsymbol{u}$ which is drawn according to any of the following distributions:*

1. *$\mathcal{D}^{(\ell)}_{yes}(w')$ for any $w' \neq w$.*
2. *$\mathcal{D}^{(\ell)}_{no}$.*

Note that Item (1) in the above follows immediately from Item (2). Additionally, the first item implies the shared secret property of the code ensemble $\mathcal{E}^{(\ell)}$. Furthermore, we remark that that above lemma implies a more stringent version of PCUSS. In addition to the shared secret property, Item (2) implies that the ensemble $\mathcal{E}^{(\ell)}$ is indistinguishable from strings that are mostly far from any encoding (i.e., drawn from $\mathcal{D}^{(\ell)}_{\text{no}}$).

The proof of Lemma 5.4.12 follows by induction over ℓ. Before we continue, we introduce some useful lemmas that will be used in the proof.

Lemma 5.4.13. *For any $\ell \geq 0$ and $w, w' \in \{0,1\}^k$ for which $w \neq w'$ it holds that*

$$\min_{(v,v') \in \mathcal{E}^{(\ell)}(w) \times \mathcal{E}^{(\ell)}(w')} \operatorname{dist}(v, v') = \Theta\left(1/4^{\ell+1}\right)$$

Proof: The proof follows by induction over ℓ. The base case for $\ell = 0$ follows directly by the fact that the code from Definition 5.2.4 has high distance, and in particular $\text{dist}(\mathcal{E}^{(0)}(w), \mathcal{E}^{(0)}(w')) > 1/10$. Assume that the lemma holds for $\ell - 1$. Namely, for $w, w' \in \{0,1\}^{k'}$ for which $w \neq w'$ it holds that

$$\min_{(v,v') \in \mathcal{E}^{(\ell-1)}(w) \times \mathcal{E}^{(\ell-1)}(w')} \text{dist}(v, v') = \Theta\left(1/4^{\ell}\right).$$

Let $\tilde{w}, \tilde{w}' \in \{0,1\}^k$ be such that $\tilde{w}' \neq \tilde{w}$. Then we can write $(\tilde{v}, \tilde{v}') \in \mathcal{E}^{(\ell)}(\tilde{w}) \times \mathcal{E}^{(\ell)}(\tilde{w}')$ as

$$\tilde{v} = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle) \qquad \text{and} \qquad \tilde{v}' = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g'(\beta) \rangle\!\rangle),$$

for some $g, g' \in \mathcal{C}_{\mathbb{F}}$ such that $g|_H = \tilde{w}$ and $g'|_H = \tilde{w}'$. By the fact that g and g' are degree $|\mathbb{F}|/2$ polynomials (which are not identical), we have that g and g' disagree on at least $|\mathbb{F} \setminus H|/4$ of the elements $\beta \in \mathbb{F} \setminus H$. By applying the induction hypothesis on the minimum distance between $\mathcal{E}^{(\ell)}(\langle\!\langle g(\beta) \rangle\!\rangle)$ and $\mathcal{E}^{(\ell)}(\langle\!\langle g'(\beta) \rangle\!\rangle)$, for all β such that $g(\beta) \neq g'(\beta)$, we have that

$$\min_{(\tilde{v},\tilde{v}') \in \mathcal{E}^{(\ell)}(\tilde{w}) \times \mathcal{E}^{(\ell)}(\tilde{w}')} \text{dist}(\tilde{v}, \tilde{v}') > \frac{1}{4} \cdot \Theta\left(\frac{1}{4^{\ell}}\right) = \Theta\left(1/4^{\ell+1}\right).$$

■

Lemma 5.4.14. *For any $\ell \geq 0$, with probability at least $1 - o(1)$, a string v drawn from $\mathcal{D}_{no}^{(\ell)}$ satisfies $\text{dist}(v, \mathcal{E}^{(\ell)}(w)) = \Theta\left(1/4^{\ell+1}\right)$ for all $w \in \{0,1\}^k$.*

Proof: The proof follows by induction over ℓ. For $\ell = 0$, fix some $w \in \{0,1\}^k$. Consider the size of a ball of relative radius $1/40$ around some $v \in \mathcal{E}^{(0)}(w)$ in the space of all strings $\{0,1\}^{4k}$. The number of strings contained in this ball is at most

$$\binom{4k}{k/10} \leq (40e)^{k/10} = 2^{k/10 \cdot \log(40e)}.$$

Thus, the size of the set of strings which are at relative distance $1/40$ from any legal encoding of some word $w \in \{0,1\}^k$ is at most

$$2^{3k} \cdot 2^{k/10 \cdot \log(40e)} = o(2^{4k}).$$

This implies that with probability at least $1 - o(1)$, a random string from $\{0,1\}^{4k}$ is $1/40$-far from $\mathcal{E}^{(0)}(w)$ for any $w \in \{0,1\}^k$.

For any $\ell > 0$, consider v' sampled according to $\mathcal{D}_{\text{no}}^{(\ell)}$. Then, v' can be written as

$$v' = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle \boldsymbol{\lambda}(\beta) \rangle\!\rangle),$$

where $\boldsymbol{\lambda} : \mathbb{F} \setminus H \to \mathbb{F}$ is a uniformly random function. On the other hand, each member $\tilde{v}$ of $\mathcal{P}^{(\ell)}$ can be written as

$$\tilde{v} = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle),$$

for some $g \in \mathcal{C}_{\mathbb{F}}$ such that $g|_H = w$ for some $w \in \{0,1\}^k$. Note that by Lemma 5.4.13, whenever $\boldsymbol{\lambda}(\beta) \neq g(\beta)$, we have that the minimum distance between any $\tilde{v} \in \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle)$ and $v' \in \mathcal{E}^{(\ell-1)}(\langle\!\langle \boldsymbol{\lambda}(\beta) \rangle\!\rangle)$ is at least $\Theta(1/4^{\ell})$. In addition, by Lemma 2.6.5, we have that that with probability at least $1 - o(1)$, a uniformly random function $\boldsymbol{\lambda} : \mathbb{F} \to \mathbb{F}$ is $1/3$-far from any $g \in \mathcal{C}_{\mathbb{F}}$. By the restrictions on k in Definition 5.4.3, which implies that $|H| \leq |F|/c$, we can ensure (by the choice of c) that with probability at least $1 - o(1)$, that a uniformly random $\boldsymbol{\lambda} : \mathbb{F} \setminus H \to \mathbb{F}$ is at least 1/4-far from the restriction $g|_{\mathbb{F}\setminus H}$. This implies that for at least $|\mathbb{F} \setminus H|/4$ of the elements $\beta \in \mathbb{F} \setminus H$, we have that $\boldsymbol{\lambda}(\beta) \neq g(\beta)$. Therefore, we have that $\mathrm{dist}(v', \mathcal{E}^{(\ell)}(w)) = \frac{1}{4} \cdot \Theta\left(\frac{1}{4^\ell}\right) = \Theta\left(1/4^{\ell+1}\right)$ for all $w \in \{0,1\}^k$, and the proof is complete. ■

Lemma 5.4.15. *Fix any $\ell > 0$, and suppose that for any $w' \in \{0,1\}^{k'}$, and for any set Q' of at most $q^{(\ell-1)}_{\mathbb{F}',k'}$ queries (where $\mathbb{F}'$ and k' are picked according to the recursive definition of the level ℓ-encoding, and for $q^{(0)}$ we substitute k' for the nonexistent $\mathbb{F}'$) the restricted distributions $\mathcal{D}^{(\ell-1)}_{yes}(w')|_{Q'}$ and $\mathcal{D}^{(\ell-1)}_{no}|_{Q'}$ are identical. Then, for any $w \in \{0,1\}^k$, and any set Q of at most $\frac{|\mathbb{F}\setminus H|}{10} \cdot q^{(\ell-1)}_{\mathbb{F}',k'}$ queries, the restricted distributions $\mathcal{D}^{(\ell)}_{yes}(w)|_Q$ and $\mathcal{D}^{(\ell)}_{no}|_Q$ are identical.*

Proof: Let $Q \subset [m^{(\ell)}]$ be the set of queries, and fix a canonical ordering over the elements in $\mathbb{F} \setminus H$. Let $\boldsymbol{v}$ be an element drawn according to distribution $\mathcal{D}^{(\ell)}_{\text{yes}}(w)$, and let $\boldsymbol{v}'$ be an element drawn according to distribution $\mathcal{D}^{(\ell)}_{\text{no}}$. The sampling process from $\mathcal{D}^{(\ell)}_{\text{yes}}(w)$ can be thought of as first drawing a uniformly random function $\boldsymbol{g} \in \mathcal{C}_\mathbb{F}$ such that $\boldsymbol{g}|_H = w$, and for every $\beta \in \mathbb{F} \setminus H$, letting $\boldsymbol{v}_\beta$ be a uniformly random element in $\mathcal{E}^{(\ell-1)}(\langle\!\langle \boldsymbol{g}(\beta) \rangle\!\rangle)$.
For each $\beta \in \mathbb{F} \setminus H$ we set $Q_\beta = Q \cap \mathrm{Dom}(v_\beta)$, and define the set of *big clusters*

$$I = \left\{ \beta \in \mathbb{F} \setminus H \ : \ |Q_\beta| \geq q^{(\ell-1)}_{\mathbb{F}',k'} \right\}.$$

Note that since $|Q| \leq |\mathbb{F} \setminus H| \cdot q^{(\ell-1)}_{\mathbb{F}',k'}/10$, we have that $|I| \leq |\mathbb{F} \setminus H|/10$.

By the fact that $\boldsymbol{g}$ is a uniformly random polynomial of degree $|\mathbb{F}|/2 > |I|$, we have that $\boldsymbol{g}|_I$ is distributed exactly as $\boldsymbol{\lambda}|_I$ (both are a sequence of $|I|$ independent uniformly random values), which implies that $\boldsymbol{v}|_{\bigcup_{j\in I} Q_j}$ is distributed exactly as $\boldsymbol{v}'|_{\bigcup_{j\in I} Q_j}$.

Next, let $\mathbb{F} \setminus (I \cup H) = \{i_1, \ldots, i_{|\mathbb{F}\setminus(I\cup H)|}\}$ be a subset ordered according to the canonical ordering over $\mathbb{F}$. We proceed by showing that $\boldsymbol{v}|_{\bigcup_{j\in I\cup\{i_1,\ldots,i_t\}} Q_j}$ is distributed identically to $\boldsymbol{v}'|_{\bigcup_{j\in I\cup\{i_1,\ldots,i_t\}} Q_j}$ by induction over t.

The base case ($t = 0$) corresponds to the restriction over $\bigcup_{j\in I} Q_j$, which was already proven above. For the induction step, let $T = \{i_1, \ldots, i_{t-1}\} \subseteq \mathbb{F} \setminus (I \cup H)$ be an ordered subset that agrees with the canonical ordering on $\mathbb{F}$, and let $i_t \in \mathbb{F} \setminus (H \cup T \cup I)$ be the successor of i_{t-1} according to the ordering. We now prove that for each $x \in \{0,1\}^{m^{(\ell)}}$ for which $\boldsymbol{v}|_{\bigcup_{j\in I\cup T} Q_j}$ has a positive probability of being equal to $x|_{\bigcup_{j\in I\cup T} Q_j}$, conditioned on the above event taking place (and its respective event for v'), $\boldsymbol{v}|_{Q_{i_t}}$ is distributed exactly as $\boldsymbol{v}'|_{Q_{i_t}}$.

Observe that conditioned on the above event, $\boldsymbol{v}|_{Q_{i_t}}$ is distributed exactly as a uniformly random element in $\mathcal{E}^{(\ell-1)}(\boldsymbol{\rho})$ for some $\boldsymbol{\rho} \in \{0,1\}^{k'}$ (which follows some arbitrary distribution, possibly depending on $x|_{\bigcup_{j\in I\cup T} Q_j}$), while $\boldsymbol{v}'|_{Q_{i_t}}$ is distributed exactly as a uniformly random element in $\mathcal{E}^{(\ell-1)}(\boldsymbol{y})$ for a uniformly random $\boldsymbol{y} \in \{0,1\}^{k'}$. By the fact that $|Q_{i_t}| \leq q^{(\ell-1)}_{\mathbb{F}',k'}/10$, we can apply the induction hypothesis and conclude that $\boldsymbol{v}|_{Q_{i_t}}$ is distributed exactly as $\boldsymbol{v}'|_{Q_{i_t}}$, because by our hypothesis both are distributed identically to the corresponding restriction of $\mathcal{D}^{(\ell-1)}_{\text{no}}$, regardless of the values picked for $\boldsymbol{\rho}$ and $\boldsymbol{y}$. This completes the induction step for t. The lemma follows by setting $t = |\mathbb{F} \setminus H \cup I|$. ■

Lemma 5.4.16. *For any $\ell \geq 0$, $w \in \{0,1\}^k$ and any set of queries $Q \subset [m^{(\ell)}]$ such that $|Q| = O\left(\frac{m^{(\ell)}}{10^\ell}\right)$, the restricted distributions $\mathcal{D}^{(\ell)}_{yes}(w)|_Q$ and $\mathcal{D}^{(\ell)}_{no}|_Q$ are identically distributed.*

Proof: By induction on ℓ. For $\ell = 0$ and any $w \in \{0,1\}^k$, by the fact that our base encoding $\mathcal{E}^{(0)}(w)$ is a high dual distance code, we can select (say) $q^{(0)} = k/c$ (for some constant $c > 0$), making the assertion of the lemma trivial.

Assume that for any $w' \in \{0,1\}^{k'}$, and any set of queries Q' of size at most $O(m^{(\ell-1)}/10^{\ell-1})$ the conditional distributions $\mathcal{D}^{(\ell-1)}_{\text{yes}}(w')|_{Q'}$ and $\mathcal{D}^{(\ell-1)}_{\text{no}}|_{Q'}$ are identically distributed. Then, by Lemma 5.4.15, we have that for any $w \in \{0,1\}^k$ and any set of queries Q of size at most

$$O\left(\frac{|\mathbb{F} \setminus H|}{10^\ell} \cdot m^{(\ell-1)}\right),$$

the restricted distributions $\mathcal{D}^{(\ell)}_{\text{yes}}(w)|_Q$ and $\mathcal{D}^{(\ell)}_{\text{no}}|_Q$ are identically distributed. Note that by definition of the level ℓ-encoding, $m^{(\ell)} = |\mathbb{F} \setminus H| \cdot m^{(\ell-1)}$, which implies the conclusion of the lemma. ■

Proof of Lemma 5.4.12: Lemma 5.4.12 follows directly by combining Lemma 2.1.4, and Lemma 5.4.16. ■

Combining Lemma 5.4.12 with the definition of Spiel-PCU (Definition 5.3.2) establishes that we have constructed a Spiel-PCUSS, which implies Theorem 5.4.2.

Corollary 5.4.17. *Let $\mathbb{F}$ be a finite field and $k \in \mathbb{N}$ which satisfy the requirements in Definition 5.4.3. Then, for every $\ell \geq 0$, the coding ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k} : \{0,1\}^k \to 2^{\left(\{0,1\}^{m^{(\ell)}}\right)}$ has $q(\varepsilon,\delta)$-query length-t Spiel-PCUSS for $t = O(m^{(\ell)}\text{polylog}^{(\ell)} m^{(\ell)})$.*

5.4.4 Handling arbitrary input lengths

As mentioned in the beginning of this section, our construction of code ensembles relies on the fact that operations over a finite field GF(2^t) can be computed efficiently. In order to do so we need to have an irreducible polynomial of degree t over GF(2), so that we have a representation GF(2^t). Given such a polynomial, operations over the field can be implemented in polylogarithmic time in the size of the field. By [Gol08] (Appendix G), we know that for $t = 2 \cdot 3^r$ where $r \in \mathbb{N}$, we do have such a representation. However, the setting of t restricts the sizes of the fields that we can work with, which will limit our input size length.

We show here how to extend our construction to a set of sizes that is "log-dense". For a global constant c', our set of possible input sizes includes a member of $[m', c'm']$ for every m'. Moving from this set to the set of all possible input sizes now becomes a matter of straightforward padding.

For any $n \in \mathbb{N}$, let r be the smallest integer such that $n < 2^{2\cdot 3^r}$ and let $\mathbb{F} = \text{GF}(2^{2\cdot 3^r})$. We make our change only at the level-ℓ construction. First, we use $4d$ instead of d in the calculation of the size of $\mathbb{F}'$. Then, instead of using $\mathbb{F} \setminus H$ as the domain for our input, we use $E \setminus H$, for any arbitrary set $E \subseteq \mathbb{F}$ of size $n \geq \max\{4k, |\mathbb{F}|^{1/4}, c_\ell\}$ that contains H. Then, for the level-ℓ, instead of considering polynomials of degree $|\mathbb{F}|/2$, we consider polynomials of degree $|E|/2$. The rest of the construction follows the same lines as the one defined above. This way, all of our operations can be implemented in polylogarithmic time in $|E|$.

5.5 Separation of testing models

In this section we use Theorem 5.4.2 to prove a separation between the standard testing model, and the tolerant testing model (in Chapter 6.3 we will show a similar separation between the standard testing model and the erasure resilient model). Specifically, we prove the following.

Theorem 5.5.1 (Restatement of Theorem 5.0.1)**.** *For every constant $\ell \in \mathbb{N}$, there exist a property $\mathcal{Q}^{(\ell)}$ and $\varepsilon_1 = \varepsilon_1(\ell) \in (0,1)$ such that the following hold.*

1. *For every $\varepsilon \in (0,1)$, the property $\mathcal{Q}^{(\ell)}$ can be ε-tested using a number of queries depending only on ε (and ℓ).*

2. *For every $\varepsilon_0 \in (0, \varepsilon_1)$, any $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ needs to make $\Omega(N/10^{\ell} \cdot \mathrm{polylog}^{(\ell)} N)$ many queries on inputs of length N.*

In order to prove the separation we use the code ensemble $\mathcal{E}_{\mathbb{F},k}^{(\ell)}$ where k is set to 0. Namely, we consider $\mathcal{E}_{\mathbb{F},0}(\emptyset)$. Note that in this case, the code ensemble becomes a property (i.e. a subset of the set of all possible strings).

Next, we define the property that exhibits the separation between the standard testing model and the tolerant testing model

Definition 5.5.2. Fix a finite field $\mathbb{F}$ and a constant integer $\ell \in \mathbb{N}$ and let $\varepsilon(\ell) = \Theta(1/4^{\ell})$. Let $n \overset{\text{def}}{=} m_{\mathbb{F}}^{(\ell)}$, $z_{\mathbb{F},0}^{(\ell)} \leq n \cdot \mathrm{polylog}^{(\ell)} n$ denote the length of the proof for the PCUSS from Theorem 5.4.2, and let $N = (\log^{(\ell)} n + 1) \cdot z_{\mathbb{F},0}^{(\ell)}$. Let $\mathcal{Q}^{(\ell)} \subseteq \{0,1\}^N$ be defined as follows. A string $x \in \{0,1\}^N$ satisfies $\mathcal{Q}^{(\ell)}$ if the following hold.

1. The first $z_{\mathbb{F},0}^{(\ell)} \cdot \log^{(\ell)} n$ bits of x consist of $s = \frac{z_{\mathbb{F},0}^{(\ell)} \cdot \log^{(\ell)} n}{n}$ copies of $y \in \mathcal{E}_{\mathbb{F},0}^{(\ell)}$.

2. The remaining $z_{\mathbb{F},0}^{(\ell)}$ bits of x consist of a proof string $\pi \in \{0,1\}^{z_{\mathbb{F},0}^{(\ell)}}$, for which the `Verifier-Procedure` $_{\mathcal{E}_{\mathbb{F},0}^{(\ell)}}$ in Figure 5.1 accepts y given oracle access to y and π.

We first show that $\mathcal{Q}^{(\ell)}$ can be tested using a constant number of queries in the standard testing model.

`Testing Algorithm for` $\mathcal{Q}^{(\ell)}$

Input: Parameter $\varepsilon \in (0,1)$, an oracle access to $x \in \{0,1\}^N$.

```
1. Set $s \overset{\text{def}}{=} \frac{z_{\mathbb{F},0}^{(\ell)} \cdot \log^{(\ell)} n}{n}$.
2. Repeat $4/\varepsilon$ times:
    (a) Sample $\boldsymbol{j} \in [n]$ and $\boldsymbol{i} \in [s] \setminus \{1\}$ uniformly at random.
    (b) If $x_{\boldsymbol{j}} \neq x_{(\boldsymbol{i}-1) \cdot n + \boldsymbol{j}}$, then **Reject**.
3. Let $v = (x_1, \ldots, x_n)$, $\pi = (x_{z_{\mathbb{F},0}^{(\ell)} \cdot \log^{(\ell)} n + 1}, \ldots, x_{(\log^{(\ell)} n + 1) z_{\mathbb{F},0}^{(\ell)}})$ and $\tau$ be the empty string.
4. Run the PCU verifier for $\mathcal{E}_{\mathbb{F},0}^{(\ell)}$ with parameters $\varepsilon/3$ and $\delta = 2/3$ on $v$, using $\pi$ as the alleged proof for $v$, and $\tau$ as the alleged value for $v$.
5. If the PCU verifier rejects, then **Reject**; otherwise **Accept**.
```

Figure 5.2: Description of `Testing Algorithm for` $\mathcal{Q}^{(\ell)}$.

Lemma 5.5.3. *The property $\mathcal{Q}^{(\ell)}$ has a tester with query complexity depending only on ε.*

Proof: We show that the algorithm described in Figure 5.2 is a testing algorithm for $\mathcal{Q}^{(\ell)}$. We assume that n is large enough so that $\log^{(\ell)} n > 6/\varepsilon$.

Assume that $x \in \mathcal{Q}^{(\ell)}$. Then, there exists a string $y \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$, such that $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \log^{(\ell)} n} = (y)^s$ (where $(y)^s$ denotes the concatenation of s copies of y), and $x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n+1}, \ldots, x_{(\log^{(\ell)}+1) z^{(\ell)}_{\mathbb{F},0}} = \pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},0}}$, where π is a proof that makes the PCU verifier for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ accept when given oracle access to y and π. Therefore, the algorithm in Figure 5.2 accepts x.

Next, assume that x is ε-far from $\mathcal{Q}^{(\ell)}$, and let $y' = x_1, \ldots, x_n$. Note that if $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}$ is $\varepsilon/2$-far from being $(z')^s$, then the loop in Step 2 rejects x with probability at least 2/3, and we are done. If $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}$ is $\varepsilon/2$-close to $(y')^s$, then y' must be $\varepsilon/3$-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$. To see this, assume toward a contradiction that y' is $\varepsilon/3$-close to $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$. Then, by modifying at most $\frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{2}$ bits, we can make $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}$ equal to $(y')^s$. Since, by our assumption y' is $\varepsilon/3$-close to $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, we can further modify the string $(y')^s$ to $(\tilde{y})^s$, where $\tilde{y} \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$, by changing at most $\frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{3}$ bits. Finally, by changing at most $z^{(\ell)}_{\mathbb{F},0}$ bits from π, we can get a proof string $\tilde{\pi}$ which will make the PCPP verifier accept $\tilde{y}$. By our assumption that $6/\varepsilon < \log^{(\ell)} n$, the total number of changes to the input string x is at most

$$\frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{2} + \frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{3} + z^{(\ell)}_{\mathbb{F},0} \leq \varepsilon \cdot (\log^{(\ell)} n + 1) \cdot z^{(\ell)}_{\mathbb{F},0} = \varepsilon N,$$

which is a contradiction to the fact that x is ε-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$.

Finally, having proved that y' is $\varepsilon/3$-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, the PCU verifier for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ (when called with parameters $\varepsilon/3$ and $\delta = 2/3$) rejects with probability at least 2/3. ■

Lemma 5.5.4. *For every constant $\ell \in \mathbb{N}$, there exists $\varepsilon_1 \stackrel{\text{def}}{=} \Theta(1/4^\ell)$ such that for every $\varepsilon_0 < \varepsilon_1$, any $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ needs to make at least $\Omega\left(\frac{N}{10^\ell \cdot \text{polylog}^{(\ell)} N}\right)$ many queries.*

Proof: Fix some constant $\ell \in \mathbb{N}$. The proof follows by a reduction from $2\varepsilon_1$-testing of $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$. Given oracle access to a string $y \in \{0,1\}^n$ which we would like to $2\varepsilon_1$-test for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, we construct an input string $x \in \{0,1\}^N$ where $N = (\log^{(\ell)} n + 1) \cdot z^{(\ell)}_{\mathbb{F},0}$ as follows.

$$x \stackrel{\text{def}}{=} (y)^{\frac{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{n}} \sqcup (0)^{z^{(\ell)}_{\mathbb{F},0}}.$$

That is, we concatenate $z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n/n$ copies of y, and set the last $z^{(\ell)}_{\mathbb{F},0}$ bits to 0. Note that a single query to the new input string x can be simulated using at most one query to the string y.

If $y \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$, then for large enough n we have that x is ε_0-close to $\mathcal{Q}^{(\ell)}$, since the last $z^{(\ell)}_{\mathbb{F},0}$ bits that are set to 0 are less than an ε_0-fraction of the input length.

On the other hand, if $\text{dist}(x, \mathcal{E}^{(\ell)}_{\mathbb{F},0}) > 2\varepsilon_1$, since each copy of y in x is $2\varepsilon_1$-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, then x is $\frac{2\varepsilon_1 \cdot \log^{(\ell)} n}{\log^{(\ell)} n+1}$-far from $\mathcal{Q}^{(\ell)}$ (note that $\frac{\log^{(\ell)} n}{\log^{(\ell)} n+1} > 1/2$). Therefore, an $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ would imply an $2\varepsilon_1$-tester for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ with the same query complexity. By Lemma 5.4.12, since for some $\varepsilon_1 = \Theta(1/4^\ell)$, every $2\varepsilon_1$-tester for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ requires $\Omega(n/10^\ell)$ queries on inputs of length n, any $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ requires to make $\Omega\left(\frac{N}{10^\ell \cdot \text{polylog}^{(\ell)} N}\right)$ many queries. ■

Proof of Theorem 5.5.1: The proof follows by combining Lemma 5.5.3 and Lemma 5.5.4. ■

Chapter 6

Applications

In this chapter we will use the techniques developed throughout this thesis to obtain additional applications. We consider the following problems.

Tolerant isomorphism testing. In the problem on isomorphism testing, we are given access to two unknown Boolean functions $f, g : \{-1,1\}^n \to \{-1,1\}$ and a parameter $\varepsilon \in (0,1)$, and wish to distinguish between the case that f is equal to g up to some permutation of the input variables, or the distance between f and g is at least ε for every relabeling of the variables in g. We show how to obtain a tolerant testing algorithm for the above problem whose query complexity is parameterized by $k^* = k(f, g, \gamma)$ – the smallest k such that either f or g is γ-close to a k-junta.

Theorem 1.3.5 (Tolerant isomorphism testing)**.** *There exists an algorithm that, given query access to two functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$ and parameter $\varepsilon \in (0,1)$, satisfies the following, for some absolute constant $C \geq 1$.*

- *If f and g are $\frac{\varepsilon}{C}$-close to isomorphic, then the algorithm accepts with high constant probability.*
- *If f and g are ε-far from isomorphic, then the algorithm rejects with high constant probability.*

The query complexity of the algorithm is $O\big(2^{\frac{k^*}{2}}/\varepsilon\big)$ with high-probability (and $O\big(2^{\frac{n}{2}}/\varepsilon\ \big)$ in the worst case), where $k^* = k^*(f, g, \frac{\varepsilon}{C})$.

Lower bounds for tolerant unateness testing. Recall that a function $f : \{-1,1\}^n \to \{-1,1\}$ is unate if it is either non-increasing or non-decreasing in every variable. Namely, there exists a string $r \in \{0,1\}^n$ such that the function $f(x \oplus r)$ is monotone with respect to the bit-wise partial order on $\{0,1\}^n$. We will show that the lower bound for testing bipartiteness using rejection sampling oracle (Theorem 4.0.1) implies lower bounds for tolerant unateness testing for both the adaptive and non-adaptive settings. Specifically,

Theorem 1.3.6. *There exist constants $0 < \varepsilon_0 < \varepsilon_1 < 1$ such that any (possibly adaptive) algorithm that distinguishes between functions ε_0-close to unate and functions ε_1-far from unate, must make $\Omega(n/\mathrm{polylog}\, n)$ queries.*

Theorem 1.3.7. *There exist constant $0 < \varepsilon_0 < \varepsilon_1 < 1$ such that any non-adaptive algorithm that distinguishes between functions ε_0-close to unate and functions ε_1-far from unate, must make $\Omega(n^{3/2}/\mathrm{polylog}\, n)$ queries.*

Separating property testing from erasure-resilient testing. Finally, we will use the short PCPP developed in Chapter 5 (Theorem 5.1.1) to obtain a sharper separation between the erasure-resilient testing model and the standard testing model.

Theorem 1.3.8 (informal restatement of Theorem 6.3.2)**.** *For any constant integer $\ell \in \mathbb{N}$, there exist a property of boolean strings $\mathcal{P} \subseteq \{0,1\}^n$ and a constant $\varepsilon_1 \in (0,1)$ such that $\mathcal{P}$ is ε-testable for any $\varepsilon > 0$ with number of queries independent of n, but for any $\alpha = \Omega(1/\log^{(\ell)} n)$ and $\varepsilon \in (0, \varepsilon_1)$ such that $\varepsilon < 1 - \alpha$, any α-erasure-resilient ε-tester is required to query $\Omega(n/\mathrm{polylog}^{(\ell)} n)$ many bits.*

6.1 "Instance-adaptive" tolerant isomorphism testing

In this section, we show how the machinery developed in section 3.2, and more precisely the algorithm from Theorem 1.3.1, can be leveraged to obtain *instance-adaptive tolerant isomorphism testing* between two unknown Boolean functions f.

The structure of our tolerant isomorphism testing algorithm is quite intuitive, and consists of two phases. In the first phase, we run a linear search on k, repeatedly invoking our tolerant junta tester to discover the smallest value k satisfying $\min(\mathrm{dist}(f, \mathcal{J}_k), \mathrm{dist}(g, \mathcal{J}_k)) \leq \varepsilon/C$. We note that a similar approach using a tester whose tolerance is only $\mathrm{poly}(\varepsilon/k)$ might return a much larger value of k, since as k increases, the allowed tolerance decreases. In the second phase, we use this value of k to tolerantly test isomorphism between f and g. This phase, however, is not as straightforward as it seems: indeed, to achieve the desired query complexity, we would like to test isomorphism – for which we have known algorithms – between f_k and g_k, that is, the k-juntas closest to f and g respectively.

Yet here we face two issues: (i) we do not have query access to f_k and g_k; (ii) even in the completeness case f_k and g_k *need not actually be isomorphic.* Indeed, f and g are only promised to be close to k-juntas, and close to isomorphic. Hence, the corresponding juntas are only guaranteed to be *close* to isomorphic.

Addressing item (ii) relies on adapting the algorithm of [ABC$^+$13], along with a careful and technical analysis of the distribution of the points it queries. (This analysis is also the key to providing the tolerance guarantees of our isomorphism tester.) We address item (i) as follows. Our algorithm builds on the ideas of Chakraborty et al. [CGM11], namely on their notion of a "noisy sampler". A noisy sampler is given query access to a function that is promised to be close to some k-junta and provides (almost) uniformly distributed samples labeled (approximately) according to this k-junta. While the [CGM11] noisy sampler works for functions that are $\mathrm{poly}(\varepsilon/k)$-close to $\mathcal{J}_k$, we need a noisy sampler that works for functions that are only $\frac{\varepsilon}{C}$-close to $\mathcal{J}_k$. To this end, we replace the weakly tolerant testing algorithm of [Bla09] used in the noisy sampler of [CGM11] with our tolerant testing algorithm. The query complexity of the resulting noisy sampler is indeed much higher than that of [CGM11]. However, this does not increase the overall query complexity of our tolerant isomorphism testing algorithm, as stated in Theorem 1.3.5.

We begin with some notation: Let $\mathcal{S}_n$ denote the set of permutations of $[n]$. For $f, g \colon \{-1,1\}^n \to \{-1,1\}$, we denote by $\mathrm{distiso}(f,g)$ the distance between f and the closest isomorphism of g, that is $\mathrm{distiso}(f,g) \stackrel{\text{def}}{=} \min_{\pi \in \mathcal{S}_n} \mathrm{dist}(f, g \circ \pi)$. Given oracle access $\mathcal{O}_f, \mathcal{O}_g$ to two unknown Boolean functions $f, g \colon \{-1,1\}^n \to \{-1,1\}$ and a parameter $\varepsilon \in (0,1)$, isomorphism testing then amounts to distinguishing between (i) $\mathrm{distiso}(f,g) = 0$; and (ii) $\mathrm{distiso}(f,g) > \varepsilon$.[1]
Our result will be parameterized in terms of the *junta degree* of the unknown functions f and g, formally defined below:

[1]Phrased differently, this is testing the property $\mathcal{P} = \left\{ (f, f \circ \pi) : f \in 2^{2^{[n]}}, \pi \in \mathcal{S}_n \right\} \subseteq 2^{2^{[n]}} \times 2^{2^{[n]}}$.

Definition 6.1.1 (Junta degree)**.** Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function, and $\gamma \in [0,1]$ a parameter. We define the *γ-junta degree of f* as the smallest integer k such that f is γ-close to being a k-junta, that is

$$k^*(f,\gamma) \overset{\text{def}}{=} \min\{\, k \in [n] : \operatorname{dist}(f,\mathcal{J}_k) \leq \gamma \,\}\,.$$

Finally, we extend this definition to two functions f, g by setting $k^*(f,g,\gamma) = \min(k^*(f,\gamma), k^*(g,\gamma))$.

With this terminology in hand, we can restate Theorem 1.3.5:

Theorem 6.1.2 (Theorem 1.3.5, rephrased)**.** *There exist absolute constants $c \in (0,1)$, $\varepsilon_0 \in (0,1)$ and a tolerant testing algorithm for isomorphism of two unknown functions f and g with the following guarantees. On inputs $\varepsilon \in (0,\varepsilon_0]$, $\delta \in (0,1]$, and query access to functions $f,g\colon \{-1,1\}^n \to \{-1,1\}$:*

- *if* $\operatorname{distiso}(f,g) \leq c\varepsilon$, *then it outputs* accept *with probability at least* $1-\delta$*;*
- *if* $\operatorname{distiso}(f,g) > \varepsilon$, *then it outputs* reject *with probability at least* $1-\delta$*.*

The query complexity of the algorithm satisfies the following, where $k^* = k^*(f,g,\frac{\rho c\varepsilon}{16})$ is the $\frac{\rho c\varepsilon}{16}$-junta degree of f and g:

- *it is $\tilde{O}\big(2^{\frac{k^*}{2}}\frac{1}{\varepsilon}\log\frac{1}{\delta}\big)$ with probability at least $1-\delta$;*
- *it is always at most $\tilde{O}\big(2^{\frac{n}{2}}\frac{1}{\varepsilon}\log\frac{1}{\delta}\big)$.*

Moreover, one can take $c = \frac{1}{1750}$, and $\varepsilon_0 \overset{\text{def}}{=} \frac{16}{15}(5-2\sqrt{6}) \simeq 0.108$.

6.1.1 Proof of Theorem 6.1.2

As described in the opening of this section, our algorithm first performs a linear search on k, invoking at each step the tolerant tester of section 3.2 with parameter ε', to obtain (with high probability) a value k^* such that $k^*(f,g,\varepsilon') \leq k^* \leq k^*(f,g,\frac{\rho\varepsilon'}{16})$. In the second stage, it calls a "noisy sampler" to obtain uniformly random labeled samples from the "cores" of the k^*-juntas closest to f and g (both notions are defined formally in subsubsection 6.1.1.2), and robustly tests isomorphism between them. We accordingly divide this section in two, proving respectively these two statements:

Lemma 6.1.3. *There exists an algorithm (Algorithm 6) with the following guarantees. On inputs $\varepsilon', \delta \in (0,1)$ and query access to $f,g\colon \{-1,1\}^n \to \{-1,1\}$, it returns a value $0 \leq \boldsymbol{k} \leq n$, such that:*

- *with probability at least $1-\delta$, we have that:*
  (i) $k^*(f,g,\varepsilon') \leq \boldsymbol{k} \leq k^*(f,g,\frac{\rho\varepsilon'}{16})$;
  (ii) the algorithm performs $O\Big(2^{\frac{\boldsymbol{k}}{2}+o(\boldsymbol{k})}\cdot\frac{1}{\varepsilon}\log\frac{1}{\delta}\Big)$ queries;
- *the algorithm performs at most $O\Big(2^{\frac{n}{2}+o(n)}\cdot\frac{1}{\varepsilon}\log\frac{1}{\delta}\Big)$ queries.*

Proposition 6.1.4. *There exists an algorithm (Algorithm 7) with query complexity $\tilde{O}\Big(\frac{2^{k/2}}{\varepsilon}\Big)$ for testing of isomorphism of two unknown functions f and g, under the premise that f is close to $\mathcal{J}_k$. More precisely, there exist absolute constants $c > 0$ and $\varepsilon_0 \in (0,1]$ such that, on inputs $k \in \mathbb{N}$, $\varepsilon \in (0,\varepsilon_0]$ and query access to functions $f,g\colon \{-1,1\}^n \to \{-1,1\}$, the algorithm has the following guarantees. Conditioned on $\operatorname{dist}(f,\mathcal{J}_k) \leq c\varepsilon$, it holds that:*

- *if* $\mathrm{distiso}(f,g) \leq c\varepsilon$*, then it outputs* accept *with probability at least* $8/15$*;*
- *if* $\mathrm{distiso}(f,g) > \varepsilon$*, then it outputs* reject *with probability at least* $8/15$*.*

Moreover, one can take $c = \frac{1}{1750}$*, and* $\varepsilon_0 \overset{\text{def}}{=} \frac{16}{15}(5-2\sqrt{6}) \simeq 0.108$*.*

Theorem 6.1.2 follows by the combination of Lemma 6.1.3 and Proposition 6.1.4.

Proof of Theorem 6.1.2: Let $\rho \overset{\text{def}}{=} 1 - \frac{1}{\sqrt{2}}$, and $\varepsilon' = c\varepsilon$. The algorithm proceeds as follows: it first invokes Algorithm 6 with inputs $f, g, \varepsilon', \delta/2$, and gets by Lemma 6.1.3, a value $1 \leq \boldsymbol{k}^* \leq n$ such that $k^*(f,g,\varepsilon') \leq \boldsymbol{k}^* \leq k^*(f,g,\frac{\rho\varepsilon'}{16})$ with probability at least $1-\frac{\delta}{2}$. In particular, conditioning on this we are guaranteed that either f or g is ε'-close to some $\boldsymbol{k}^*$-junta (i.e., by our choice of c, one of the functions is $c\varepsilon$-close to $\mathcal{J}_{\boldsymbol{k}^*}$). It then calls Algorithm 7 with inputs $f, g, \boldsymbol{k}^*, \varepsilon$ independently $O(\log\frac{1}{\delta})$ times (for probability amplification from $8/15$ to $1-\frac{\delta}{2}$), and accepts if and only if the majority of these executions returned accept . The correctness of the algorithm follows from Proposition 6.1.4 and the bound on the query complexity follows from the bounds in Lemma 6.1.3 and Proposition 6.1.4. ∎

6.1.1.1 Linear search: finding k^*.

Let $\mathcal{T}$ denote the algorithm of Theorem 1.3.1, with probability of success amplified by standard techniques to $1-\delta$ for any $\delta \in (0,1]$ (at the price of a factor $O\big(\log\frac{1}{\delta}\big)$ in its query complexity); and write $q_{\mathcal{T}}(k,\varepsilon,\rho,\delta) = O\Big(\frac{k\log k}{\varepsilon\rho(1-\rho)^k}\log\frac{1}{\delta}\Big)$ for its query complexity. Algorithm 6, given next, performs the linear search for k^*: we then analyze its correctness and query complexity.

Algorithm 6 Junta Degree Finder($\mathcal{O}_f, \mathcal{O}_g, \varepsilon', \rho, \delta$)

```
1: Set $\rho \leftarrow 1 - \frac{1}{\sqrt{2}}$ and let $\mathcal{T}$ be the algorithm of Theorem 1.3.1.
2: for $k = 0$ to $n$ do
3:     Call $\mathcal{T}$on $f$ with parameters $k$, $\varepsilon'$, $\rho$, and $3\delta/(2\pi^2(k+1)^2)$.
4:     Call $\mathcal{T}$on $g$ with parameters $k$, $\varepsilon'$, $\rho$, and $3\delta/(2\pi^2(k+1)^2)$.
5:     if either call to $\mathcal{T}$returned accept then return $k$.
6:     end if
7: end for
8: return $n$
```

Proof of Lemma 6.1.3: By a union bound, all executions of $\mathcal{T}$ will be correct with probability at least $1-2\sum_{j=1}^{\infty}\frac{3\delta}{2\pi^2 j^2} = 1-\frac{\delta}{2}$. Conditioning on this, the tester will accept for some k between $k^*(f,g,\varepsilon')$ and $k^*(f,g,\rho\varepsilon'/16)$. This is true since as long as we invoke $\mathcal{T}$ with values k such that f and g are ε'-far from $\mathcal{J}_k$, both invocations of $\mathcal{T}$ will reject. Therefore, once we accept, we have that either f or g is at least ε'-close to $\mathcal{J}_k$. Hence, $k \geq k^*(f,g,\varepsilon')$. Also, $\mathcal{T}$ is guaranteed to accept on some k' whenever invoked on a function that is $\rho\varepsilon'/16$-close to $\mathcal{J}_{k'}$. By definition, $k^*(f,g,\rho\varepsilon'/16)$ is such a k' for either f or g; hence, $k \leq k^*(f,g,\rho\varepsilon'/16)$.

In the case that all the executions of $\mathcal{T}$ returned correctly, the query complexity is

$$q(\varepsilon,f,g) = \sum_{k=0}^{k^*(f,g,\frac{\rho\varepsilon'}{16})} 2q_{\mathcal{T}}\left(k,\varepsilon',\rho,\frac{3\delta}{2\pi^2(k+1)^2}\right).$$

By the expression of $q_{\mathcal{T}}$, we get that $q(\varepsilon,f,g)$ is upper bounded by

$$q(\varepsilon,f,g) \leq \frac{O(1)}{\varepsilon\rho}\sum_{k=1}^{k^\star}\frac{k\log k\log\frac{k}{\delta}}{(1-\rho)^k} \leq \frac{O(1)}{\varepsilon}(k^\star\log k^\star)^2 2^{k^\star\log\frac{1}{1-\rho}}\log\frac{1}{\delta}$$

where $k^\star \stackrel{\text{def}}{=} k^*(f, g, \frac{\rho\varepsilon'}{16})$. In particular, from the choice of ρ, we get $q(\varepsilon, f, g) \leq O\Big(2^{\frac{k^\star}{2}+o(k^\star)}\frac{1}{\varepsilon}\log\frac{1}{\delta}\Big)$.

(If not all the executions of the tester are successful, in the worst case the algorithm considers all possible values of k, before finally returning n. In this case, the query complexity is similarly bounded by $O\Big(2^{\frac{n}{2}+o(n)}\frac{1}{\varepsilon}\log\frac{1}{\delta}\Big)$.)

$\blacksquare$

6.1.1.2 Noisy samplers and core juntas.

For a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ we denote by $f_k : \{-1,1\}^n \to \{-1,1\}$ the k-junta closest to f. That is, the function $h \in \mathcal{J}_k$ such that $\operatorname{dist}(f,h) = \operatorname{dist}(f,\mathcal{J}_k)$ (if this function is not unique, then we define f_k to be the first according to lexicographic order). Moreover, following Chakraborty et al. [CGM11], for a k-junta $h \in \mathcal{J}_k$ (where we assume without loss of generality that h depends on exactly k variables) we define the *core* of h, as follows. The core of h, denoted $\mathsf{core}_h\colon \{-1,1\}^k \to \{-1,1\}$, is the restriction of h to its relevant variables (where these variables are numbered according to the natural order); so that for some $i_1 \leq \cdots \leq i_k \in [n]$ we have

$$h(x) = \mathsf{core}_h(x_{i_1}, \dots, x_{i_k})$$

for every $x \in \{-1,1\}^n$.

Definition 6.1.5 ([CGM11, Definition 1]). Let $g\colon \{-1,1\}^k \to \{-1,1\}$ be a function and let $\eta, \mu \in [0,1)$. An (η,μ)-*noisy sampler for g* is a probabilistic algorithm that on each execution outputs a pair $(\boldsymbol{x}, \boldsymbol{a}) \in \{-1,1\}^k \times \{-1,1\}$ such that

(i) For all $y \in \{-1,1\}^k$, $\mathbf{Pr}[\boldsymbol{x} = y] \in \Big[\frac{1-\mu}{2^k}, \frac{1+\mu}{2^k}\Big]$;

(ii) $\mathbf{Pr}[\boldsymbol{a} = g(\boldsymbol{x})] \geq 1-\eta$;

(iii) the pairs output on different executions are mutually independent.

An η-*noisy sampler* is an $(\eta,0)$-noisy sampler, i.e., one that on each execution selects a uniformly random $\boldsymbol{x} \in \{-1,1\}^k$.

Chakraborty et al. [CGM11] show how to build an efficient $O(\varepsilon)$-noisy sampler for core_{f_k}, which is guaranteed to apply as long as $\operatorname{dist}(f,\mathcal{J}_k) = O\big(\varepsilon^6/k^{10}\big)$. In more detail, they first run a modified version of the junta tester from [Bla09], which, whenever it accepts, also returns some preprocessing information that enables one to build such a noisy sampler. Moreover, they show that this tester will indeed accept any function that is $O\big(\varepsilon^6/k^{10}\big)$-close to $\mathcal{J}_k$ (in addition to rejecting those ε-far from it), giving the above guarantee. Using instead (a small modification of) our tolerant tester from section 3.2, we are able to extend their techniques to obtain the following – less efficient, but more robust – noisy sampler.

Proposition 6.1.6 (Noisy sampler for close-to-junta functions). *There are algorithms $\mathcal{A}_P, \mathcal{A}_S$ (respectively preprocessor and sampler), which both require oracle access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$, and satisfy the following properties.*

- *The preprocessor $\mathcal{A}_P$ takes $\varepsilon' \in (0,1]$, $\rho \in (0,1)$, $k \in \mathbb{N}$ as inputs, makes $O\bigg(\frac{k\log\frac{k}{\varepsilon'}}{\varepsilon'\rho(1-\rho)^k}\bigg)$ queries to f, and either returns* fail *or a state $\boldsymbol{\sigma} \in \{0,1\}^{\operatorname{poly}(n)}$. The sampler $\mathcal{A}_S$ takes as input such a state $\boldsymbol{\sigma} \in \{0,1\}^{\operatorname{poly}(n)}$, makes a single query to f, and outputs a pair $(\boldsymbol{x}, \boldsymbol{a}) \in \{-1,1\}^k \times \{-1,1\}$. We say that a state $\boldsymbol{\sigma}$ is* γ-good *if for some permutation $\pi \in \mathcal{S}_k$, $\mathcal{A}_S(\boldsymbol{\sigma})$ is a γ-noisy sampler for $\mathsf{core}_{f_k} \circ \pi$.*

- *$\mathcal{A}_P(\varepsilon', \rho, k)$ fulfills the following conditions:*
   (i) If $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{16}\varepsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns a state $\boldsymbol{\sigma}$ that is $3\varepsilon'$-good.
   (ii) If $\operatorname{dist}(f, \mathcal{J}_k) > \varepsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns fail .
   (iii) If $\operatorname{dist}(f, \mathcal{J}_k) \leq \varepsilon'$, then with probability at least $4/5$, $\mathcal{A}_p$ either returns fail or returns a state $\boldsymbol{\sigma}$ that is $3\varepsilon'$-good.

The proof of Proposition 6.1.6 is deferred to Appendix 6.1.2; indeed, it is almost identical to the proof of Proposition 4.16 in [CGM11], with small adaptations required to comply with the use of the tolerant tester from section 3.2 instead of the tester from [Bla09].

We note that the main difference between the guarantees of our noisy sampler and those of the noisy sampler in [CGM11, Lemma 2] lies in the set of functions for which the noisy sampler is required to return a good state. In our case, this set consists of functions that are *somewhat* close to k-juntas. In comparison, the construction from [CGM11] is more query-efficient (only $\tilde{O}(k/\varepsilon)$ queries to f in the preprocessing stage), but only guarantees the output of a noisy sampler for functions f that are $O\big(\varepsilon^6/k^{10}\big)$-close to $\mathcal{J}_k$.

With these primitives in hand, we are almost ready to prove the main proposition of this subsection, Proposition 6.1.4. To state the algorithm (Algorithm 7) and proceed with its analysis, we will require the following definition:

Definition 6.1.7 (Number of violating pairs V_π)**.** Given two sets $Q_1, Q_2 \subseteq \{-1,1\}^k \times \{-1,1\}$ and a permutation $\pi \in \mathcal{S}_k$ we say that pairs $(x, a_1) \in Q_1$ and $(y, a_2) \in Q_2$ are *violating with respect to* π, if $y = \pi(x)$ and $a_1 \neq a_2$. We denote the number of violating pairs with respect to π by V_π.

Algorithm 7 Tolerant isomorphism testing to an unknown f such that $\operatorname{dist} f \mathcal{J}_k \leq c\varepsilon$ $(\mathcal{O}_f, \mathcal{O}_g, \varepsilon, k)$

```
1: Let $\mathcal{A}_P, \mathcal{A}_S$ be as in Proposition 6.1.6, $\rho \leftarrow 1 - \frac{1}{\sqrt{2}}$, $\varepsilon' \leftarrow \frac{\varepsilon}{16}$, $\alpha \leftarrow 4c\varepsilon$.
2: $s \leftarrow C\frac{2^{k/2}}{\varepsilon}\sqrt{k \ln k}$, $t \leftarrow (3\alpha + 9\varepsilon')\frac{s^2}{2^k}$.        ▷ $C > 1$ is an absolute constant.
3: Run the preprocessor $\mathcal{A}_P$ on $f$ and $g$ with parameters $\varepsilon'$, $\rho$, $k$.
4: if either invocation of $\mathcal{A}_P$ returned fail then
5:     return reject .
6: end if
7: Using the $3\varepsilon'$-noisy sampler $\mathcal{A}_S$ (called with the states returned on Step 3), construct "core"
   sets $\mathbf{Q}_f, \mathbf{Q}_g \subseteq \{-1,1\}^k \times \{-1,1\}$ each of size $s \leftarrow C\frac{2^{k/2}}{\varepsilon}\sqrt{k \ln k}$.
8: if there exist $\pi \in \mathcal{S}_k$ such that $\mathbf{V}_\pi \leq t$ then
9:     return accept .
10: end if
11: return reject .
```

Proof of Proposition 6.1.4: The query complexity is the sum of the query complexities from Steps 3 and 7, i.e.,

$$O\left(\frac{k \log \frac{k}{\varepsilon}}{\varepsilon\rho(1-\rho)^k}\right) + 2s \cdot 1 = O\left(\frac{2^{k/2}}{\varepsilon} k \log \frac{k}{\varepsilon} + \frac{2^{k/2}}{\varepsilon}\sqrt{k \ln k}\right) = O\left(\frac{2^{k/2}}{\varepsilon} k \log \frac{k}{\varepsilon}\right).$$

Completeness. Assume that g is $c\varepsilon$-close to isomorphic to f, which itself is $c\varepsilon$-close to being a k-junta. Therefore, by the triangle inequality and by our choice of $c \leq \frac{\rho}{512}$, $\operatorname{dist}(g, \mathcal{J}_k) \leq 2c\varepsilon \leq \rho\varepsilon'/16$ as well, so that with probability at least $3/5$ the algorithm does not output reject on Step 5 (we thereafter analyze this case). Moreover, by the triangle inequality there exists a permutation $\pi \in \mathcal{S}_n$ such that $\operatorname{dist}(f_k, g_k \circ \pi) \leq 2c\varepsilon + 2c\varepsilon = 4c\varepsilon \overset{\text{def}}{=} \alpha$. In particular, this implies that there exists a permutation $\pi^* \in \mathcal{S}_k$ such that $\operatorname{dist}(\mathsf{core}_{f_k}, \mathsf{core}_{g_k} \circ \pi^*) \leq \alpha$. Let $T^* \subseteq \{-1,1\}^k$ be the disagreement set between core_{f_k} and $\mathsf{core}_{g_k} \circ \pi^*$: by the above $|T^*| \leq \alpha 2^k$.

Let $\mathbf{Q}_f^s, \mathbf{Q}_g^s \subseteq \{-1,1\}^k$ denote the sets resulting from taking the first element in each pair in $\mathbf{Q}_f$ and $\mathbf{Q}_g$ respectively. The size of the intersection $\mathbf{Z} \overset{\text{def}}{=} \left|\mathbf{Q}_f^s \cap T^*\right|$ is distributed as a Binomial random variable, namely $\mathbf{Z} \sim \operatorname{Bin}\left(s, \frac{|T^*|}{2^k}\right)$, and conditioned on $\mathbf{Z}$ we have $\mathbf{Z}^* \overset{\text{def}}{=} \left|Q_f^s \cap \mathbf{Q}_g^s \cap T^*\right| \sim \operatorname{Bin}\left(s, \frac{Z}{2^k}\right)$. In particular, we get

$$\mathbf{E}[\mathbf{Z}] = \frac{s\,|T^*|}{2^k}, \quad \mathbf{E}[\mathbf{Z}^*|Z] = \mathbf{E}\left[\left|Q_f^s \cap \mathbf{Q}_g^s \cap T^*\right| \mid \left|Q_f^s \cap T^*\right|\right] = \frac{sZ}{2^k}\ .$$

Let $\mathcal{A}_S^f$ denote the noisy sampler algorithm when invoked for f, and for every $x \in \mathbf{Q}_f^s$ let $\mathcal{A}_S^f(x)$ denote the label given to x by $\mathcal{A}_S^f$. Since $\mathcal{A}_S^f$ is a $3\varepsilon'$-noisy sampler for core_{f_k}, $\mathbf{Pr}[\mathcal{A}_S^f(\boldsymbol{x}) \neq \mathsf{core}_{f_k}(\boldsymbol{x})] \leq 3\varepsilon'$. An analogous statement holds for g. We let $\mathbf{N} \overset{\text{def}}{=} \left|\{x \in \mathbf{Q}_f^s \cap \mathbf{Q}_g^s : \mathcal{A}_S^f(x) \neq \mathsf{core}_{f_k}(x) \text{ or } \mathcal{A}_S^g(x\right.$ be the number of common samples incorrectly labelled by either noisy sampler, and observe that $\mathbf{N}$ is dominated by a Binomial random variable $\tilde{\mathbf{N}} \sim \operatorname{Bin}\left(\left|Q_f^s \cap Q_g^s\right|, 6\varepsilon'\right)$.
With this in hand, we can bound $\mathbf{Pr}[\mathbf{V}_{\pi^*} > t]$ as follows (recall that $t = 3\alpha + 9\varepsilon'$):

$$\begin{aligned}
\mathbf{Pr}\left[\mathbf{V}_{\pi^*} > (3\alpha + 9\varepsilon')\frac{s^2}{2^k}\right] &\leq \mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T^*\right| > 3\alpha\frac{s^2}{2^k}\right] + \mathbf{Pr}\left[\mathbf{N} > 9\varepsilon'\frac{s^2}{2^k}\right] \\
&\leq \mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T^*\right| > 3\alpha\frac{s^2}{2^k}\right] + \mathbf{Pr}\left[\tilde{\mathbf{N}} > 9\varepsilon'\frac{s^2}{2^k}\right].
\end{aligned}$$

Recall that $\mathbf{Z}^* = \left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T^*\right|$. Since $\mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T^*\right| > 3\alpha\frac{s^2}{2^k}\right]$ is maximized when $|T^*|$ is maximal, we assume without loss of generality that $|T^*| = \alpha 2^k$. We will handle each term separately.

$$\begin{aligned}
\mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\cdot\alpha\frac{s^2}{2^k}\right] &= \mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}}\right] \\
&= \mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \,\middle|\, \mathbf{Z} > \frac{5}{4}\frac{s\,|T^*|}{2^k}\right]\cdot\mathbf{Pr}\left[\mathbf{Z} > \frac{5}{4}\frac{s\,|T^*|}{2^k}\right] \\
&\quad + \mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \,\middle|\, \mathbf{Z} \leq \frac{5}{4}\frac{s\,|T^*|}{2^k}\right]\cdot\mathbf{Pr}\left[\mathbf{Z} \leq \frac{5}{4}\frac{s\,|T^*|}{2^k}\right] \\
&\leq \mathbf{Pr}\left[\mathbf{Z} > \frac{5}{4}\frac{s\,|T^*|}{2^k}\right] + \mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \,\middle|\, \mathbf{Z} \leq \frac{5}{4}\frac{s\,|T^*|}{2^k}\right].
\end{aligned}$$

We again bound the two terms separately. By the assumption that $|T^*| = \alpha 2^k$ and by the choice of s,

$$\mathbf{Pr}\left[\mathbf{Z} > \frac{5}{4}\frac{s\,|T^*|}{2^k}\right] = \mathbf{Pr}\left[\mathbf{Z} > \frac{5}{4}\mathbf{E}[\mathbf{Z}]\right] < \exp\left(-\frac{1}{3}\cdot\left(\frac{1}{4}\right)^2\cdot\frac{s\,|T^*|}{2^k}\right) < \frac{1}{30}.$$

As for the second term, since $\mathbf{E}[\mathbf{Z}^*] = \frac{s\mathbf{Z}}{2^k}$ and by the assumption on T^* and the setting of s,

$$\begin{aligned}\mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; \mathbf{Z} < \frac{5}{4}\frac{s\,|T^*|}{2^k}\right] &\le \mathbf{Pr}\left[\mathbf{Z}^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; \mathbf{Z} = \frac{5}{4}\frac{s\,|T^*|}{2^k}\right]\\ &= \mathbf{Pr}\left[\mathbf{Z}^* > \frac{6}{5}\mathbf{E}[\mathbf{Z}^*] \;\middle|\; \mathbf{Z} = \frac{5}{4}\frac{s\,|T^*|}{2^k}\right]\\ &< \exp\left(-\frac{1}{3}\cdot\left(\frac{1}{5}\right)^2\cdot\frac{s}{2^k}\frac{s\,|T^*|}{2^k}\right) < \frac{1}{30}\end{aligned}$$

for a sufficiently large constant C in the definition of s.
As for the last term of the initial expression, since $\mathbf{E}[\tilde{\mathbf{N}}] = 6\varepsilon'\left|Q_f^s \cap Q_g^s\right|$ we have,

$$\begin{aligned}\mathbf{Pr}\left[\tilde{\mathbf{N}} > 9\varepsilon'\frac{s^2}{2^k}\right] &\le \mathbf{Pr}\left[\tilde{\mathbf{N}} > 9\varepsilon'\frac{s^2}{2^k} \;\middle|\; \left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| \le \frac{5}{4}\frac{s^2}{2^k}\right]\cdot\mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| \le \frac{5}{4}\frac{s^2}{2^k}\right]\\ &\quad + \mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| > \frac{5}{4}\frac{s^2}{2^k}\right]\\ &\le \mathbf{Pr}\left[\tilde{\mathbf{N}} > 9\varepsilon'\frac{s^2}{2^k} \;\middle|\; \left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| = \frac{5}{4}\frac{s^2}{2^k}\right] + \mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| > \frac{5}{4}\frac{s^2}{2^k}\right]\\ &\le \mathbf{Pr}\left[\tilde{\mathbf{N}} > \frac{6}{5}\mathbf{E}[\tilde{\mathbf{N}}] \;\middle|\; \left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| = \frac{5}{4}\frac{s^2}{2^k}\right] + \mathbf{Pr}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s\right| > \frac{5}{4}\frac{s^2}{2^k}\right]\\ &< \exp\left(-\frac{1}{3}\cdot\left(\frac{1}{6}\right)^2\cdot\frac{16\varepsilon'\cdot 5s^2}{4\cdot 2^k}\right) + \exp\left(-\frac{1}{3}\cdot\left(\frac{1}{4}\right)^2\cdot\frac{s^2}{2^k}\right) \le \frac{1}{15}.\end{aligned}$$

(Actually $o(1)$.)

The algorithm will therefore reject with probability at most $\frac{2}{5} + \frac{1}{15} + \frac{1}{15} = \frac{7}{15}$.

Soundness. Assume that $\operatorname{dist}(f, \mathcal{J}_k) \le c\varepsilon$, and that g is ε-far from being isomorphic to f. Then one of the following must hold:

1. $\operatorname{dist}(g, \mathcal{J}_k) > \varepsilon'$.
2. for all $\pi \in \mathcal{S}_k$, $\operatorname{dist}(\mathsf{core}_{f_k}, \mathsf{core}_{g_k} \circ\pi) > \varepsilon - (\varepsilon' + c\varepsilon) > \varepsilon - 2\varepsilon'$.

If the first case holds, then the function will be rejected in Step 3 with probability at least $\frac{4}{5}$, and so the algorithm will reject as desired. We can therefore focus on the second case.

If the second case holds, either the tester rejects in Step 5 (and we are done) or it outputs a state which will be used to get the $3\varepsilon'$-noisy sampler. Fix any $\pi \in \mathcal{S}_k$. Since $\operatorname{dist}\mathsf{core}_{f_k}\mathsf{core}_{g_k} \circ\pi > (\varepsilon - 2\varepsilon')$, there are $m \overset{\text{def}}{=} m(\pi) \ge (\varepsilon - 2\varepsilon')2^k$ inputs $x \in \{-1,1\}^k$ such that $\mathsf{core}_{f_k}(x) \neq \mathsf{core}_{g_k} \circ\pi(x)$. Let $T = T(\pi) \subseteq \{-1,1\}^k$ denote the set of all such inputs (so that $|T| = m$).

We can make a similar argument as for the completeness case: we have that $\left|\mathbf{Q}_f^s \cap T\right|$ is a random variable with Binomial distribution (of parameters s, and $\frac{|T|}{2^k}$). Conditioned on $\left|\mathbf{Q}_f^s \cap T\right|$, we also have $\left|Q_f^s \cap \mathbf{Q}_g^s \cap T\right| \sim \operatorname{Bin}\left(s, \frac{|Q_f^s \cap T|}{2^k}\right)$, so that

$$\begin{aligned}\mathbf{E}[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T\right|] &= \mathbf{E}\left[\mathbf{E}\left[\left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T\right| \;\middle|\; \left|\mathbf{Q}_f^s \cap T\right|\right]\right]\\ &= \mathbf{E}\left[\frac{s\left|\mathbf{Q}_f^s \cap T\right|}{2^k}\right] = \frac{s^2\,|T|}{2^{2k}} \ge (\varepsilon - 2\varepsilon')\frac{s^2}{2^k} = 14\varepsilon'\frac{s^2}{2^k}.\end{aligned}$$

(Recall that our threshold was set to $t = (3\alpha + 9\varepsilon')\frac{s^2}{2^k} \leq 12\varepsilon'\frac{s^2}{2^k}$.) Moreover, each element $x \in Q_f^s \cap Q_g^s \cap T$ will contribute to V_π with probability at least $(1-3\varepsilon')^2 > \frac{24}{25}$ (since this is a lower bound on the probability that both $\mathcal{A}_S^f(x) = \mathsf{core}_{f_k}(x)$ and $\mathcal{A}_S^g(x) = \mathsf{core}_{g_k}(x)$, and as $\varepsilon' \leq \frac{\varepsilon_0}{16}$). As before, we can therefore write, letting $\mathbf{Z} \overset{\text{def}}{=} \left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T\right|$, and taking $|T|$ to be minimal so that $|T| = (\varepsilon - 2\varepsilon')2^k$,

$$\begin{aligned}\mathbf{Pr}[\,\mathbf{V}_\pi > t\,] &\geq \mathbf{Pr}\left[\,\mathbf{V}_\pi > t \;\middle|\; \mathbf{Z} \geq \frac{13}{12}t\,\right]\mathbf{Pr}\left[\,\mathbf{Z} \geq \frac{13}{12}t\,\right]\\ &\geq (1 - e^{-\frac{1}{3}\left(\frac{1}{26}\right)^2\cdot\frac{24}{25}\cdot\frac{13t}{12}})\,\mathbf{Pr}\left[\,\mathbf{Z} \geq \frac{13}{12}t\,\right] = \left(1 - e^{-\frac{t}{2000}}\right)\mathbf{Pr}\left[\,\mathbf{Z} \geq \frac{13}{12}t\,\right]\end{aligned}$$

(Chernoff bound)

so that it is sufficient to lower bound $\mathbf{Pr}[\,\mathbf{Z} \geq \frac{13}{12}t\,]$. To do so, we will bound the probability of the two following events:

E1: $\mathbf{Y} \overset{\text{def}}{=} \left|\mathbf{Q}_f^s \cap T\right| < \frac{99}{100}\frac{s|T|}{2^k}$

E2: $\mathbf{Z} = \left|\mathbf{Q}_f^s \cap \mathbf{Q}_g^s \cap T\right| < \frac{99}{100}\frac{s}{2^k}\left|\mathbf{Q}_f^s \cap T\right|$, conditioning on $\left|\mathbf{Q}_f^s \cap T\right| \geq \frac{99}{100}\frac{s|T|}{2^k}$.

This will be sufficient for us to conclude, as by our choice of $t = (3\alpha + 9\varepsilon')\frac{s^2}{2^k}$, the setting $|T| = (\varepsilon - 2\varepsilon')2^k = \frac{7}{8}\varepsilon 2^k$, and since $\alpha \leq \varepsilon'$, we have

$$\frac{13}{12}\cdot t \leq \frac{13}{12}\cdot 12\varepsilon'\cdot\frac{s^2}{2^k} = \frac{13}{16}\cdot\varepsilon\cdot\frac{s^2}{2^k} \leq \left(\frac{99}{100}\right)^2\frac{s^2|T|}{2^{2k}}\,.$$

Therefore, by a Chernoff bound

$$\begin{aligned}\mathbf{Pr}\left[\,\mathbf{Z} < \frac{13}{12}t\,\right] &\leq \mathbf{Pr}\left[\,\mathbf{Z} < \left(\frac{99}{100}\right)^2\frac{s^2\,|T|}{2^{2k}}\,\right]\\ &\leq \mathbf{Pr}\left[\,\mathbf{Y} < \frac{99}{100}\frac{s\,|T|}{2^k}\,\right] + \mathbf{Pr}\left[\,\mathbf{Z} < \frac{99}{100}\frac{s}{2^k}Y \;\middle|\; \mathbf{Y} \geq \frac{99}{100}\frac{s\,|T|}{2^k}\,\right]\mathbf{Pr}\left[\,\mathbf{Y} \geq \frac{99}{100}\frac{s\,|T|}{2^k}\,\right]\\ &\leq \mathbf{Pr}\left[\,\mathbf{Y} < \frac{99}{100}\frac{s\,|T|}{2^k}\,\right] + \mathbf{Pr}\left[\,\mathbf{Z} < \frac{99}{100}\frac{s}{2^k}Y \;\middle|\; \mathbf{Y} \geq \frac{99}{100}\frac{s\,|T|}{2^k}\,\right]\\ &< \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{100}\right)^2\cdot\frac{s\,|T|}{2^k}\right) + \mathbf{Pr}\left[\,\mathbf{Z} < \frac{99}{100}\frac{sY}{2^k} \;\middle|\; \mathbf{Y} = \frac{99}{100}\frac{s\,|T|}{2^k}\,\right]\\ &\leq \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{100}\right)^2\cdot\frac{s(\varepsilon-2\varepsilon')\cdot 2^k}{2^k}\right) + \exp\left(-\frac{1}{2}\cdot\left(\frac{2}{100}\right)^2\cdot\frac{s^2(\varepsilon-2\varepsilon')\cdot 2^k}{2^{2k}}\right)\\ &< \exp(-\tau C^2 k\ln k),\end{aligned}$$

by the choice $s = C\frac{2^{k/2}}{\varepsilon}\sqrt{k\ln k}$, and for some constant $\tau \in (0,1)$. Hence setting C to a sufficiently large constant, the foregoing analysis implies that $\mathbf{Pr}[\,\mathbf{V}_\pi \leq t\,] \leq e^{-\frac{t}{2000}} + e^{-\tau C^2 k\ln k} = e^{-\frac{12c+3/4}{2000}\varepsilon s^2/2^k} + e^{-\tau C^2 k\ln k} \leq \frac{7}{15k^k}$. A union bound over all $k! < k^k$ permutations $\pi \in \mathcal{S}_k$ finally yields $\mathbf{Pr}[\,\exists\pi,\, \mathbf{V}_\pi \leq t\,] \leq \frac{7}{15}$ as claimed. ■

6.1.2 Construction of a noisy sampler

In this subsection we provide the proof of Proposition 6.1.6, restated below:

Proposition 6.1.8 (Noisy sampler for close-to-junta functions)**.** *There are algorithms $\mathcal{A}_P, \mathcal{A}_S$ (respectively preprocessor and sampler), which both require oracle access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$, and satisfy the following properties.*

- *The preprocessor $\mathcal{A}_P$ takes $\varepsilon' \in (0,1]$, $\rho \in (0,1)$, $k \in \mathbb{N}$ as inputs, makes $O\left(\frac{k \log \frac{k}{\varepsilon'}}{\varepsilon' \rho (1-\rho)^k}\right)$ queries to f, and either returns* fail *or a state $\boldsymbol{\sigma} \in \{0,1\}^{\mathrm{poly}(n)}$. The sampler $\mathcal{A}_S$ takes as input such a state $\boldsymbol{\sigma} \in \{0,1\}^{\mathrm{poly}(n)}$, makes a single query to f, and outputs a pair $(\boldsymbol{x}, \boldsymbol{a}) \in \{-1,1\}^k \times \{-1,1\}$. We say that a state $\boldsymbol{\sigma}$ is γ-*good *if for some permutation $\pi \in \mathcal{S}_k$, $\mathcal{A}_S(\boldsymbol{\sigma})$ is a γ-noisy sampler for $\mathsf{core}_{f_k} \circ \pi$.*
- *$\mathcal{A}_P(\varepsilon', \rho, k)$ fulfills the following conditions:*
  - *(i) If $\mathrm{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{16}\varepsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns a state $\boldsymbol{\sigma}$ that is $3\varepsilon'$-good.*
  - *(ii) If $\mathrm{dist}(f, \mathcal{J}_k) > \varepsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns* fail *.*
  - *(iii) If $\mathrm{dist}(f, \mathcal{J}_k) \leq \varepsilon'$, then with probability at least $4/5$, $\mathcal{A}_p$ either returns* fail *or returns a state $\boldsymbol{\sigma}$ that is $3\varepsilon'$-good.*

We will very closely follow the argument from the full version of [CGM11] (Proposition 4.16),[2] adapting the corresponding parts in order to obtain our result. For completeness, we tried to make this appendix below self-contained, reproducing almost verbatim several parts of the proof from [CGM11].[3]

Proof of Proposition 6.1.6: In order to use our result from section 3.2 *in lieu* of the junta tester from [Bla09], we first need to make a small modification to our algorithm. Specifically, in its first step our tester will now pick a random partition $\boldsymbol{\mathcal{I}}$ of $[n]$ in $\ell \stackrel{\text{def}}{=} \frac{Ck^2}{\varepsilon}$ parts instead of $24k^2$ (for some (small) absolute constant $C > 1$). It is easy to check that both Lemma 3.1.3 and Lemma 3.1.4 still hold (e.g., from the proof of [Bla12, Lemma 5.4]), now with probability at least 19/20. Moreover, our modified tolerant tester offers the same soundness and completeness guarantees as Theorem 1.3.1, at the price of a query complexity $O\left(\frac{k \log(k/\varepsilon)}{\varepsilon \rho (1-\rho)^k}\right)$ (instead of $O\left(\frac{k \log k}{\varepsilon \rho (1-\rho)^k}\right)$). Moreover, in Step 4 of Algorithm 3, i.e. when the algorithm found a suitable set $J \subseteq [\ell]$ (of size $\ell - k$) as a witness for accepting, we make the algorithm return $\boldsymbol{\mathcal{I}}$ and the set $\mathcal{J} \stackrel{\text{def}}{=} \{\mathbf{I}_j\}_{j \in \bar{J}}$ along with the verdict accept .

We will also require the definitions of the distribution induced by a partition $\mathcal{I}$ and a subset $\mathcal{J} \subseteq \mathcal{I}$, and of such a couple $(\mathcal{I}, \mathcal{J})$ being *good* for a function:

Definition 6.1.9 ([CGM11, Definition 4.6])**.** For any partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ of $[n]$, and subset of parts $\mathcal{J} \subseteq \mathcal{I}$, we define a pair of distributions:

The distribution $\mathcal{D}_{\mathcal{I}}$ on $\{-1,1\}^n$. An element $\boldsymbol{y} \sim \mathcal{D}_{\mathcal{I}}$ is sampled by

1. picking $\boldsymbol{z} \in \{-1,1\}^\ell$ uniformly at random among all $\binom{\ell}{\ell/2}$ strings of weight $\frac{\ell}{2}$;
2. setting $\boldsymbol{y}_i = \boldsymbol{z}_j$ for all $j \in [\ell]$ and $i \in I_j$.

The distribution $\mathcal{D}_{\mathcal{J}}$ on $\{-1,1\}^{|\mathcal{J}|}$. An element $\boldsymbol{x} \sim \mathcal{D}_{\mathcal{J}}$ is sampled by

1. picking $\boldsymbol{y} \sim \mathcal{D}_{\mathcal{I}}$;

[2]The full version can be found at `http://www.cs.technion.ac.il/~ariem/eseja.pdf`.

[3]The reader may notice that Chakraborty et al. rely on a definition of set-influence that differs from ours by a factor 2; we propagated the changes through the argument.

2. outputting $\mathsf{extract}_{(\mathcal{I},\mathcal{J})}(y)$, where $\boldsymbol{x} = \mathsf{extract}_{(\mathcal{I},\mathcal{J})}(\boldsymbol{y})$ is defined as follows. For all $j \in [\ell]$ such that $I_j \in \mathcal{J}$:
   - if $I_j \neq \emptyset$, set $\boldsymbol{x}_j = \boldsymbol{y}_i$ (where $i \in I_j$);
   - if $I_j = \emptyset$, set $\boldsymbol{x}_j$ to be a uniformly random bit.

Lemma 6.1.10 ([CGM11, Lemma 4.7]). *$\mathcal{D}_\mathcal{I}$ and $\mathcal{D}_\mathcal{J}$ as above satisfy the following.*

- *For all $a \in \{-1,1\}^n$, $\mathbf{Pr}_{\mathcal{I},\boldsymbol{y}\sim\mathcal{D}_\mathcal{I}}[\,\boldsymbol{y} = a\,] = \frac{1}{2^n}$.*
- *Assume $\ell > 4\,|\mathcal{J}|^2$. For every $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$, the total variation distance between $\mathcal{D}_\mathcal{J}$ and the uniform distribution on $\{-1,1\}^{|\mathcal{J}|}$ is bounded by $2\,|\mathcal{J}|^2/\ell$. Moreover, the ℓ_∞ distance between the two distributions is at most $4\,|\mathcal{J}|^2/(\ell 2^{|\mathcal{J}|})$.*

Definition 6.1.11 ([CGM11, Definition 4.8]). Given $(\mathcal{I},\mathcal{J})$ as above and oracle access to $f\colon \{-1,1\}^n \to \{-1,1\}$, we define a probabilistic algorithm $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ that on each execution produces a pair $\langle \boldsymbol{x}, \boldsymbol{a}\rangle \in \{-1,1\}^{|\mathcal{J}|} \times \{-1,1\}$ as follows: first it picks a random $\boldsymbol{y} \sim \mathcal{D}_\mathcal{I}$, then it queries f on $\boldsymbol{y}$, computes $\boldsymbol{x} = \mathsf{extract}_{(\mathcal{I},\mathcal{J})}(y)$ and outputs the pair $\langle \boldsymbol{x}, f(\boldsymbol{y})\rangle$.

Definition 6.1.12 ([CGM11, Definition 4.9]). Given $\alpha > 0$, a function $f\colon \{-1,1\}^n \to \{-1,1\}$, a partition $\mathcal{I} = \{I_1,\dots,I_\ell\}$ of $[n]$ and a subset $\mathcal{J} \subseteq \mathcal{I}$ of k parts, we call the pair $(\mathcal{I},\mathcal{J})$ *α-good* (with respect to f) if there exists a k-junta $h \in \mathcal{J}_k$ such that the following conditions are satisfied:

1. Conditions on h:
   (a) Every relevant variable of h is also a relevant variable of f_k;
   (b) $\operatorname{dist}(h, f_k) \leq \alpha$.
2. Conditions on $\mathcal{I}$:
   (a) For all $j \in [\ell]$, I_j contains at most one variable of core_{f_k};
   (b) $\mathbf{Pr}_{\boldsymbol{y}\sim\mathcal{D}_\mathcal{I}}[\,f(\boldsymbol{y}) \neq f_k(\boldsymbol{y})\,] \leq 10 \cdot \operatorname{dist}(f, f_k)$.
3. Condition on $\mathcal{J}$: the set $S \overset{\text{def}}{=} \bigcup_{I\in\mathcal{J}} I$ contains all relevant variables of h.

Lemma 6.1.13 ([CGM11, Lemma 4.10]). *Let $\alpha, f, \mathcal{I}, \mathcal{J}$ be as in the preceding definition. If the pair $(\mathcal{I},\mathcal{J})$ is α-good (with respect to f), then $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ (as per Definition 6.1.11) is an (η,μ)-noisy sampler for some permutation of core_{f_k}, with $\eta \leq 2\alpha + \frac{4k^2}{\ell} + 10\cdot\operatorname{dist}(f,f_k)$ and $\mu \leq \frac{4k^2}{\ell}$.*

The last piece we shall need is the ability to convert an (η,μ)-noisy sampler to a $(\eta',0)$-noisy sampler – that is, one whose samples are exactly uniformly distributed.

Lemma 6.1.14 ([CGM11, Lemma 4.4]). *Let $\tilde{g}$ be an (η,μ)-noisy sampler for $g\colon \{-1,1\}^k \to \{-1,1\}$, that on each execution picks x according to some fixed (and fully known) distribution $\mathcal{D}$. Then $\tilde{g}$ can be turned into an $(\eta+\mu)$-noisy sampler $\tilde{g}_{\mathrm{unif}}$ for g.*

With this in hand, we are ready to prove the main lemma:

Lemma 6.1.15 (Analogue of [CGM11, Proposition 4.16]). *The tester from Theorem 1.3.1, modified as above, has the following guarantees. It has query complexity $O\Big(\frac{k\log(k/\varepsilon)}{\varepsilon\rho(1-\rho)^k}\Big)$ and outputs, in case of acceptance, a partition $\mathcal{I}$ of $[n]$ in $\ell \overset{\text{def}}{=} O\big(k^2/\varepsilon\big)$ parts along with a subset $\mathcal{J} \subseteq \mathcal{I}$ of k parts such that for any f the following conditions hold:*

- *if* $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{16}\varepsilon$*, the algorithm accepts with probability at least* $9/10$*;*
- *if* $\operatorname{dist}(f, \mathcal{J}_k) > \varepsilon$*, the algorithm rejects with probability at least* $9/10$*;*
- *for any* f*, with probability at least* $4/5$ *either the algorithm rejects, or it outputs* $\mathcal{J}$ *such that the pair* $(\mathcal{I}, \mathcal{J})$ *is* $\frac{1}{2}(1 + \frac{3}{8}\rho)\varepsilon$*-good (as per Definition 6.1.12).*

In particular, if $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{16}\varepsilon$*, then with probability at least* $4/5$ *the algorithm outputs a set* $\mathcal{J}$ *such that* $(\mathcal{I}, \mathcal{J})$ *is* $\frac{1}{2}(1 + \frac{3}{8}\rho)\varepsilon$*-good.*

Proof of Lemma 6.1.15: The first two items follow from the analysis of the tester (Theorem 1.3.1) and the foregoing discussion; we thus turn to establishing the third item.

Called with parameters k, ρ, ε, our algorithm, with probability at least 19/20, either rejects or outputs a partition $\mathcal{I}$ of $[n]$ into $\ell = O(k^2)$ parts and set $\mathcal{J} \subseteq \mathcal{I}$ satisfying $\mathbf{Inf}_f(\overline{\phi(\mathcal{J})}) \leq \varepsilon$. Let $R \subseteq [n]$ (with $|R| \leq k$) denote the set of relevant variables of f_k, and $V \supseteq R$ (with $|V| = k$) the set of relevant variables of core_{f_k}. Assume that $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho\varepsilon}{16}$.[4] We then have:

- by the above, with probability at least 19/20 the algorithm outputs a set $\mathcal{J} \subseteq \mathcal{I}$ which satisfies
$$\mathbf{Inf}_f(\overline{\phi(\mathcal{J})}) \leq \varepsilon;$$
- since $\ell \gg k^2$, with probability at least 19/20 all elements of V fall in different parts of the partition $\mathcal{I}$;
- by Lemma 6.1.10 and by Markov's inequality, with probability at least 9/10 the partition $\mathcal{I}$ satisfies $\mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{D}_\mathcal{I}}[\, f(\boldsymbol{y}) \neq f_k(\boldsymbol{y})\,] \leq 10 \cdot \operatorname{dist}(f, f_k)$.

So by a union bound, 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 $(1 + \frac{3}{2}\rho)\varepsilon$-good. Let $U \stackrel{\text{def}}{=} R \cap \left(\bigcup_{I \in \mathcal{J}} I\right)$ (informally, U is the subset of the relevant variables of f_k that were successfully "discovered" by the tester). Since $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho\varepsilon}{16}$, we have $\mathbf{Inf}_f(\bar{V}) \leq 4\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho\varepsilon}{4}$. By the sub-additivity and monotonicity of influence we get

$$\mathbf{Inf}_f(\bar{U}) \leq \mathbf{Inf}_f(\bar{V}) + \mathbf{Inf}_f(V \setminus U) \leq \mathbf{Inf}_f(\bar{V}) + \mathbf{Inf}_f(\overline{\phi(\mathcal{J})}) \leq \frac{\rho\varepsilon}{4} + \varepsilon.$$

where the second inequality follows from $V \setminus U \subseteq \overline{\phi(\mathcal{J})}$. This means (see e.g. [Bla12, Lemma 2.21]) that there is a k-junta h on U such that $\operatorname{dist}(f, h) \leq \frac{1}{2}(\frac{\rho\varepsilon}{4} + \varepsilon)$, and by the triangle inequality $\operatorname{dist}(f_k, h) \leq \frac{1}{2}(\frac{\rho\varepsilon}{4} + \varepsilon) + \frac{\rho\varepsilon}{16} = \frac{1}{2}(1 + \frac{3}{8}\rho)\varepsilon$. Based on this h, we can verify that the pair $(\mathcal{I}, \mathcal{J})$ is $\frac{1}{2}(1 + \frac{3}{8}\rho)\varepsilon$-good by going over the conditions in Definition 6.1.12. ■

Concluding the proof of Proposition 6.1.6. We conclude as in Section 4.6 of [CGM11], and start by describing how $\mathcal{A}_P$ and $\mathcal{A}_S$ operate. The preprocessor $\mathcal{A}_P$ starts by calling the tester $\mathcal{T}$ of Lemma 6.1.15. Then, in case $\mathcal{T}$ accepted, $\mathcal{A}_P$ encodes in the state σ the partition $\mathcal{I}$ and the subset $\mathcal{J} \subseteq \mathcal{I}$ output by $\mathcal{T}$ (see Lemma 6.1.15), along with the values of k and ε. The sampler $\mathcal{A}_S$, given σ, obtains a pair $\langle x, a\rangle \in \{-1,1\}^k \times \{-1,1\}$ by executing $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ (from Definition 6.1.11) once. Now we show how Proposition 6.1.6 follows from Lemma 6.1.15. The first two items are immediate. As for the third item, notice that we only have to analyze the case where $\operatorname{dist}(f, f_k) \leq \frac{\rho\varepsilon}{16}$ and $\mathcal{T}$ accepted; all other cases are taken care of by the first two items. By the third item in Lemma 6.1.15, with probability at least 4/5 the pair $(\mathcal{I}, \mathcal{J})$ is

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

$\frac{1}{2}(1+\frac{3}{8}\rho)\varepsilon$-good. If so, by Lemma 6.1.13, $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ is an (η,μ)-noisy sampler for some permutation of core_{f_k}, where

$$\eta \le 2\cdot\frac{1}{2}(1+\frac{3}{8}\rho)\varepsilon + \frac{4k^2}{\ell} + 10\cdot \mathrm{dist}(f,\mathcal{J}_k) \le (1+\frac{3}{8}\rho)\varepsilon + \frac{10\rho\varepsilon}{16} + \frac{4k^2}{\ell} = (1+\rho)\varepsilon + \frac{4k^2}{\ell}$$

and $\mu \le \frac{4k^2}{\ell}$. This in turn implies by Lemma 6.1.14 an η'-noisy sampler, for

$$\eta' = \eta+\mu \le (1+\rho)\varepsilon + \frac{8k^2}{\ell} \le (2+\rho)\varepsilon \le 3\varepsilon$$

as claimed. (Where we used that $\frac{8k^2}{\ell} \le \varepsilon$ by our choice of ℓ.) ■

6.2 Lower bound for non-adaptive tolerant unateness testing

In this section, we show an additional application to our lower bound techniques from Chapter 4. In particular, we show how to reduce distinguishing distributions $\mathcal{G}_1$ and $\mathcal{G}_2$ (as defined in Section 4.2.2) to distinguishing between Boolean functions which are close to unate and Boolean functions which are far from unate. We start with a high level overview of the constructions and reduction, and then proceed to give formal definitions and the reductions for adaptive and non-adaptive tolerant testing.

6.2.1 High Level Overview

Similarly to Section 4.2, we define two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ supported on Boolean functions, so that functions in $\mathcal{D}_{\text{yes}}$ are ε_0-close to being unate, and functions in $\mathcal{D}_{\text{no}}$ are ε_1-far from being unate (where ε_0 and ε_1 are appropriately defined constants).

We will use a randomized indexing function $\mathbf{\Gamma}\colon \{0,1\}^n \to [N]$ based on the Talagrand-style constructions from [BB16, CWX17a] to partition $\{0,1\}^n$ in a unate fashion. Again, we will then use a graph $\mathbf{G} \sim \mathcal{G}_1$ or $\mathcal{G}_2$ to define the sequence of sub-function $\mathbf{H} = (\boldsymbol{h}_i\colon \{0,1\}^n \to \{0,1\} : i \in [N])$. The sub-functions $\boldsymbol{h}_i$ will be given by a parity (or negated parity) of three variables: two variables will correspond to the end points of an edge sampled $(\boldsymbol{j}_1,\boldsymbol{j}_2) \sim \mathcal{G}$, the third variable will be one of two pre-specified variables, which we call m_1 and m_2. Consider for simplicity the case when $\boldsymbol{h}_i(x) = x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} \oplus x_{m_1}$, and assume that we require that variable m_1 is non-decreasing.

Similarly to Section 4.2, the functions $\boldsymbol{h}_i$ are thought of as gadgets. We will have that if $\boldsymbol{h}_i$ is defined with respect to an edge (j_1,j_2) and m_1, then the function $\boldsymbol{f}$ will be "encouraged" to make variables j_1 and j_2 have opposite directions, i.e., either j_1 is non-increasing and j_2 is non-decreasing, or j_1 is non-decreasing and j_2 is non-increasing. In order to see why the three variable parity implements this gadget, we turn our attention to Figure 6.2.1 and Figure 6.2.1.

Intuitively, the function $\boldsymbol{f}$ needs to change some of its inputs to be unate, and it must choose whether the variables j_1 and j_2 will be monotone (non-decreasing) or anti-monotone (non-increasing). Suppose $\boldsymbol{f}$ decides that the variable j_1 should be monotone and j_2 be anti-monotone, and m_1 will always be monotone (since it will be too expensive to make it anti-monotone). Then, when $\boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus x_{m_1}$, $\boldsymbol{h}_i$ will have some *violating edges*, i.e., edges in direction j_1 which are decreasing, or edges in direction j_2 which are increasing, or edges in direction m_1 which are decreasing (see Figure 6.2.1, where these violating edges are marked in red). In this case, there exists a way that $\boldsymbol{f}$ may change $\frac{1}{4}$-th fraction of the points and remove all violating edges (again, this procedure is shown in Figure 6.2.1).

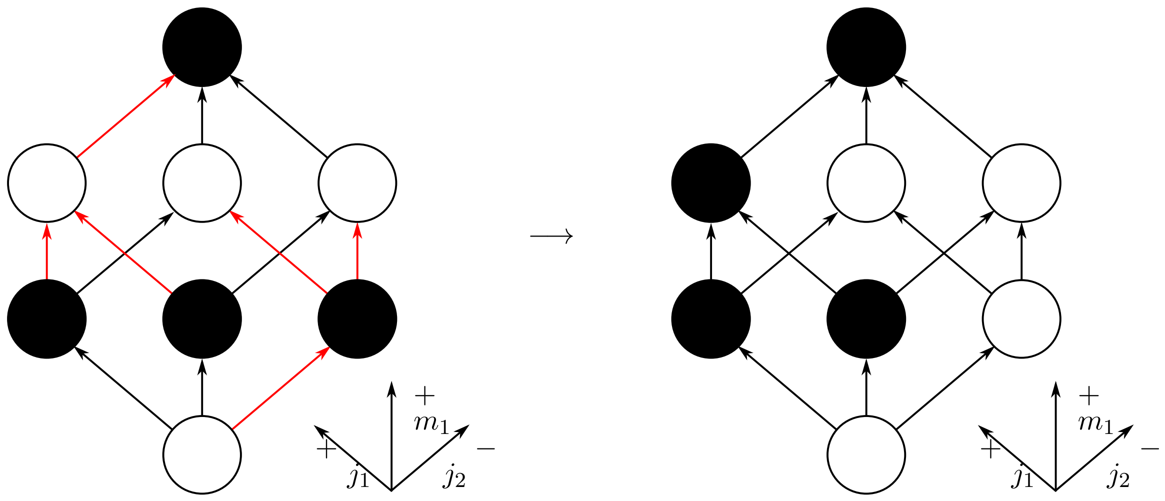


Figure 6.1: Example of a function $\boldsymbol{h}_i\colon \{0,1\}^n \to \{0,1\}$ with $\boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus x_{m_1}$ with variable j_1 (which ought to be monotone), j_2 (which ought to be anti-monotone), and m_1 (which is always monotone). The image on the left-hand side represents $\boldsymbol{h}_i$, and the red edges correspond to violating edges for variables j_1, j_2 and m_1. In other words, the red edges correspond to anti-monotone edges in variables j_1, monotone edges in variables j_2, and anti-monotone edges in direction m_1. On the right-hand side, we show how such a function can being "fixed" into a function $\boldsymbol{h}'_i\colon \{0,1\}^n \to \{0,1\}$ by changing $\frac{1}{4}$-fraction of the points.

In contrast, suppose that $\boldsymbol{f}$ decides that the variables j_1 and j_2 both should be monotone. Then, when $\boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus x_{m_1}$, the violating edges (shown in Figure 6.2.1) form vertex-disjoint cycles of length 6 in $\{0,1\}^n$, thus, the function $\boldsymbol{f}$ will have to change $\frac{3}{8}$-th fraction of the points in order to remove all violating edges. In other words, when there is an edge (j_1, j_2) sampled in $\boldsymbol{h}_i$, the function $\boldsymbol{f}$ is "encouraged" to make j_1 and j_2 have opposite directions, and "discouraged" to make j_1 and j_2 have the same direction. The other cases are presented in Figures 6.2.2, 6.2.2, and 6.2.2.

In order for $\boldsymbol{f}$ to become unate, it must first choose whether each variable will be monotone or anti-monotone. $\boldsymbol{f}$ will choose all variables in $\mathbf{M}$ to be monotone, the variable m_1 to be monotone, and m_2 to be anti-monotone, but will have to make a choice for each variable in $\overline{\mathbf{M}}$, corresponding to each vertex of the graph $\mathbf{G}$. As discussed above, for each edge (j_1, j_2) in the graph, $\boldsymbol{f}$ is encouraged to make these orientations opposite from each other, so $\boldsymbol{f}$ will want to look for the maximum cut on the graph, whose value will be different in $\mathcal{G}_1$ and $\mathcal{G}_2$.

Similarly to the case in Section 4.2, the reduction will follow by defining the rejection sampling queries L_i corresponding to variables explored in sub-function $\boldsymbol{h}_i$. The unate indexing functions $\mathbf{\Gamma}$ are not as strong as the indexing functions from the Section 4.2, so for each query in the Boolean function testing algorithm, our reduction will lose some cost in the rejection sampling algorithm. In particular, the adaptive reduction loses n cost for each Boolean function query, since adaptive algorithms can efficiently explore variables with a binary search; this gives the $\widetilde{\Omega}(n)$ lower bound for tolerant unateness testing. The non-adaptive reduction loses $O(\sqrt{n}\log n)$ cost for each Boolean function query since queries falling in the same part may be $\Omega(\sqrt{n})$ away from each other (the same scenario occurs in the non-adaptive monotonicity lower bound of [CWX17a]). The non-adaptive reduction is more complicated than the adaptive reduction since it is not exactly a black-box reduction (we require a lemma from Section 4.3). This gives the $\widetilde{\Omega}(n^{3/2})$ lower bound for non-adaptive tolerant unateness testing.

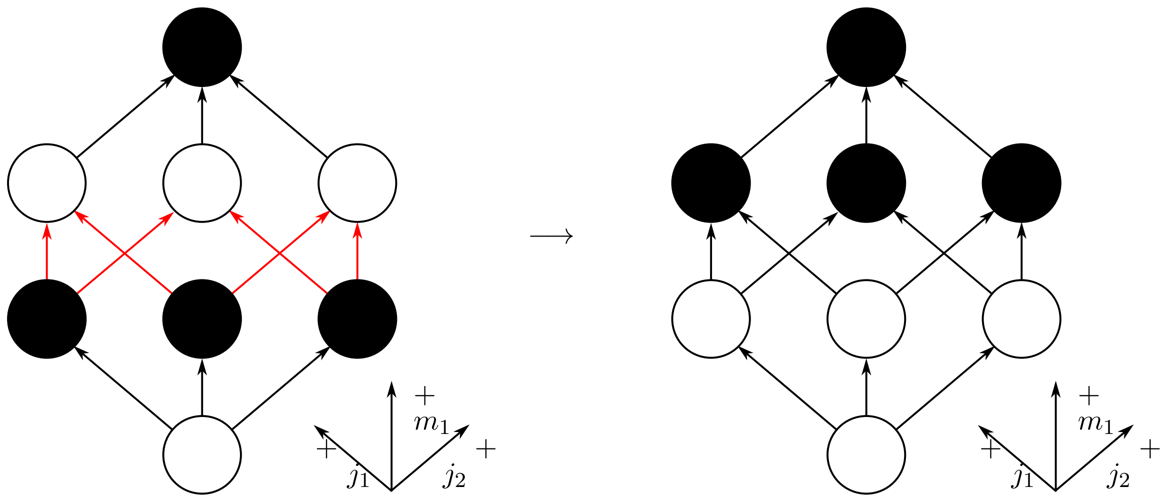


Figure 6.2: Example of a function $\boldsymbol{h}_i \colon \{0,1\}^n \to \{0,1\}$ with $\boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus x_{m_1}$ with variables j_1 and j_2 (which ought to be monotone), and m_1 (which ought to be monotone). On the left side, we indicate the violating edges with red arrows, and note that the functions in the left and right differ by $\frac{3}{8}$-fraction of the points. We also note that any function $\boldsymbol{h}'_i \colon \{0,1\}^n \to \{0,1\}$ which has j_1, j_2 and m_1 monotone must differ from $\boldsymbol{h}_i$ on at least $\frac{3}{8}$-fraction of the points because the violating edges of $\boldsymbol{h}_i$ form a cycle of length 6.

6.2.2 The Distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$

We now turn to describing a pair of distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ supported on Boolean functions $f \colon \{0,1\}^n \to \{0,1\}$. These distributions will have the property that for some constants ε_0 and ε_1 with $0 < \varepsilon_0 < \varepsilon_1$,

$$\mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}} [\text{dist}(\boldsymbol{f}, \text{Unate}) \le \varepsilon_0] = 1 - o(1) \qquad \text{and} \qquad \mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{\text{no}}} [\text{dist}(\boldsymbol{f}, \text{Unate}) \ge \varepsilon_1] = 1 - o(1).$$

We first define a function $\boldsymbol{f} \sim \mathcal{D}_{\text{no}}$, where we fix the parameter:

$$N = 2^{\sqrt{n}}.$$

1. Sample some set $\mathbf{M} \subset [n]$ of size $|\mathbf{M}| = \frac{n}{2}$ uniformly at random and let $\boldsymbol{m}_1, \boldsymbol{m}_2 \sim \mathbf{M}$ be two distinct indices.
2. We let $\mathbf{T} \sim \mathcal{E}(\mathbf{M} \setminus \{\boldsymbol{m}_1, \boldsymbol{m}_2\})$ (which we describe next). $\mathbf{T}$ is a sequence of terms $(\mathbf{T}_i : i \in [N])$ which is used to defined a multiplexer map $\Gamma_{\mathbf{T}} \colon \{0,1\}^n \to [N] \cup \{0^*, 1^*\}$.
3. We sample $\mathbf{A} \subset \overline{\mathbf{M}}$ of size $|\mathbf{A}| = \frac{n}{2}$ and define a graph as:
$$\mathbf{G} = K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}}.$$
4. We now define the distribution over sub-functions $\mathbf{H} = (\boldsymbol{h}_i : i \in [N]) \sim \mathcal{H}(\boldsymbol{m}_1, \boldsymbol{m}_2, \mathbf{G})$. For each function $\boldsymbol{h}_i \colon \{0,1\}^n \to \{0,1\}$, we generate $\boldsymbol{h}_i$ independently:
   - When $i \le 3N/4$, we sample $\boldsymbol{j} \sim \{\boldsymbol{m}_1, \boldsymbol{m}_2\}$ and we let:
   $$\boldsymbol{h}_i(x) = \begin{cases} x_{\boldsymbol{j}} & \boldsymbol{j} = \boldsymbol{m}_1 \\ \neg x_{\boldsymbol{j}} & \boldsymbol{j} = \boldsymbol{m}_2 \end{cases}.$$
   - Otherwise, if $i > 3N/4$, we sample an edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ and an index $\boldsymbol{j}_3 \sim \{\boldsymbol{m}_1, \boldsymbol{m}_2\}$ we let:
   $$\boldsymbol{h}_i(x) = \begin{cases} x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} \oplus x_{\boldsymbol{j}_3} & \boldsymbol{j}_3 = \boldsymbol{m}_1 \\ \neg x_{\boldsymbol{j}_1} \oplus x_{\boldsymbol{j}_2} \oplus x_{\boldsymbol{j}_3} & \boldsymbol{j}_3 = \boldsymbol{m}_2 \end{cases}.$$

The function $\boldsymbol{f}\colon \{0,1\}^n \to \{0,1\}$ is given by $\boldsymbol{f}(x) = f_{\mathbf{T},\mathbf{A},\mathbf{H}}(x)$ where:

$$f_{\mathbf{T},\mathbf{A},\mathbf{H}}(x) = \begin{cases} 1 & |x_{|\mathbf{M}}| > \frac{n}{4} + \sqrt{n} \\ 0 & |x_{|\mathbf{M}}| < \frac{n}{4} - \sqrt{n} \\ 1 & \Gamma_{\mathbf{T}}(x) = 1^* \\ 0 & \Gamma_{\mathbf{T}}(x) = 0^* \\ \boldsymbol{h}_{\Gamma_{\mathbf{T}}(x)}(x) & \text{otherwise} \end{cases} . \tag{6.1}$$

We now turn to define the distribution $\mathcal{E}(M)$ supported on terms $\mathbf{T}$, as well as the multiplexer map $\Gamma_{\mathbf{T}}\colon \{0,1\}^n \to [N]$. As mentioned above, $\mathbf{T} \sim \mathcal{E}(M)$ will be a sequence of N terms $(\mathbf{T}_i : i \in [N])$, where each $\mathbf{T}_i$ is given by a DNF term:

$$\mathbf{T}_i(x) = \bigwedge_{j \in \mathbf{T}_i} x_j,$$

where the set $\mathbf{T}_i \subset M$ is a uniformly random $\sqrt{n}$-element subset. Given the sequence of terms $\mathbf{T}$, we let:

$$\Gamma_{\mathbf{T}}(x) = \begin{cases} 0^* & \forall i \in [N], \mathbf{T}_i(x) = 0 \\ 1^* & \exists i_1 \neq i_2 \in [N], \mathbf{T}_{i_1}(x) = \mathbf{T}_{i_2}(x) = 1 \\ i & \mathbf{T}_i(x) = 1 \text{ for a unique } i \in [N] \end{cases} .$$

It remains to define the distribution $\mathcal{D}_{\text{yes}}$ supported on Boolean functions. The function $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ will be defined almost exactly the same. We still have $\boldsymbol{f} = f_{\mathbf{T},\mathbf{A},\mathbf{H}}$ as defined above, however, the graph $\mathbf{G}$ will be different. In particular, we will let:

$$\mathbf{G} = K_{\mathbf{A},\overline{\mathbf{A}}}.$$

Fix any set $M \subset [n]$ of size $\frac{n}{2}$ and let $m_1, m_2 \in M$ be two distinct indices and $M' = M \setminus \{m_1, m_2\}$. For any $\mathbf{T} \sim \mathcal{E}(M')$, let $\mathbf{X} \subset \{0,1\}^n$ be the subset of points indexed to some subfunction $\boldsymbol{h}_i$:

$$\mathbf{X} \overset{\text{def}}{=} \left\{ x \in \{0,1\}^n : |x_{|M}| \in [n/4 - \sqrt{n}, n/4 + \sqrt{n}] \text{ and } \Gamma_T(x) \in [N] \right\},$$

and define $\gamma \in (0,1)$ be the parameter:

$$\gamma \overset{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{T} \sim \mathcal{E}(M')} \left[\frac{|\mathbf{X}|}{2^n} \right].$$

Claim 6.2.1. *With probability at least* $1 - \exp\left(-\Omega(N/n^2)\right)$ *over the draw* $\mathbf{T} \sim \mathcal{E}(M)$ *the set* $\mathbf{X}$ *has size* $|\mathbf{X}| = 2^n \gamma(1 \pm \frac{1}{n})$*, where* $\gamma = \Omega(1)$.

Proof: Note that:

$$\mathop{\mathbf{E}}_{\mathbf{T} \sim \mathcal{E}(M)} [|\mathbf{X}|] = \sum_{\substack{x \in \{0,1\}^n : \\ n/4 - \sqrt{n} \leq |x_{|M}| \leq n/4 + \sqrt{n}}} \mathop{\mathbf{Pr}}_{\mathbf{T} \sim \mathcal{E}(M)} [x \in \mathbf{X}].$$

Fix $x \in \{0,1\}^n$ such that $n/4 - \sqrt{n} \leq |x_{|M}| \leq n/4 + \sqrt{n}$. We can view the probability on the right hand side as a sequence of N disjoint events. Every event $j \in [N]$ correspond to the case where x satisfies the unique term $\mathbf{T}_j$. The probability of each such event is:

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\mathbf{T} \sim \mathcal{E}(M)} [\Gamma_{\mathbf{T}}(x) = i] &\geq \left(\frac{1}{(n/2 - 2)^{\sqrt{n}}} \prod_{k=0}^{\sqrt{n}-1} (|x_{|M}| - k - 2) \right) \cdot \left(1 - \left(\frac{|x_{|M}|}{n/2 - 2} \right)^{\sqrt{n}} \right)^{N-1} \\ &\geq \left(\frac{n/4 - 2\sqrt{n}}{n/2} \right)^{\sqrt{n}} \cdot \left(1 - \left(\frac{n/4 + \sqrt{n}}{n/2 - 2} \right)^{\sqrt{n}} \right)^{N-1} = \Omega(1/N). \end{aligned}$$

Therefore, the probability that $x \in \mathbf{X}$ is at least $\Omega(1)$. Summing up all the x with $|x_{|M}| \approx \frac{n}{4} \pm \sqrt{n}$ gives $\mathbf{E}_{\mathbf{T}\sim\mathcal{E}(M)}[|\mathbf{X}|] = \Omega(2^n)$, so $\gamma = \Omega(1)$. In order to show that the random variable $|\mathbf{X}|$ is concentrated around the mean, let Ω be the space of all possible $\sqrt{n}$-sized terms with variables in $M \setminus \{m_1, m_2\}$, and let $c\colon \Omega^N \to \mathbb{Z}^{\geq 0}$ be the function on the independent terms which computes the size of $\mathbf{X}$:

$$c(\mathbf{T}_1, \dots, \mathbf{T}_N) = |\mathbf{X}|.$$

For every $j \in [N]$ and $T_1, \dots, T_N, T'_j \in \Omega$

$$\left|c(T_1, \dots, T'_j, \dots, T_N) - c(T_1, \dots, T_j, \dots, T_N)\right| \leq \frac{2^n}{N},$$

so by McDiarmid's inequality:

$$\Pr_{\mathbf{T}\sim\mathcal{E}(M)}\left[||\mathbf{X}| - \gamma 2^n| \geq 2^n/n\right] \leq \exp\left(-\frac{\Omega(2^{2n}/n^2)}{\sum_{i=1}^{N} 2^{2n}/N^2}\right) = \exp\left(-\Omega(N/n^2)\right).$$

■

In addition, let $X_i \subset X$ be the subset of points $x \in X$ with $\Gamma_T(x) = i$, and note that the subsets $X_1, \dots, X_N$ partition X, where each $|X_i| \leq 2^{n-\sqrt{n}}$. With a similar argument as Claim 6.2.1, we conclude that with probability $1 - o(1)$ over the draw of $\mathbf{T} \sim \mathcal{E}(M)$, we have:

$$\sum_{i=1}^{3N/4} |\mathbf{X}_i| = 2^n \cdot \frac{3\gamma}{4}\left(1 \pm \frac{1}{n}\right) \qquad \text{and} \qquad \sum_{i=3N/4+1}^{N} |\mathbf{X}_i| = 2^n \cdot \frac{\gamma}{4}\left(1 \pm \frac{1}{n}\right). \tag{6.2}$$

Thus, we only consider functions $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ (or $\sim \mathcal{D}_{\text{no}}$) where the sets M, and T satisfy (6.2).

We consider any set $A \subset \overline{M}$ of size $\frac{n}{4}$. Now, consider any graph G defined over vertices in $\overline{M}$, and we let:

$$\chi(G) = \min\left\{\frac{E_G(S,S) + E_G(\overline{S},\overline{S})}{E_G(\overline{M},\overline{M})} : S \subset \overline{M}\right\}.$$

In other words, we note that $\chi(G)$ is one minus the fractional value of the maximum cut, and the value of $\chi(G)$ is minimized for the set S achieving the maximum cut of G. The following lemma relates the distance to unateness of a function $\boldsymbol{f} = f_{T,A,\mathbf{H}}$ with $\mathbf{H} \sim \mathcal{H}(m_1, m_2, G)$, where G is an underlying graph defined on vertices in $\overline{M}$.

Lemma 6.2.2. *Let G be any graph defined over vertices in $\overline{M}$. If $\boldsymbol{f} = f_{T,A,\mathbf{H}}$ where $\mathbf{H} \sim \mathcal{H}(m_1, m_2, G)$, then*

$$\frac{\gamma}{16}\left(1 + \frac{1}{2}\cdot\chi(G)\right) - o(1) \leq \text{dist}(\boldsymbol{f}, \text{Unate}) \leq \frac{\gamma}{16}\left(1 + \frac{1}{2}\cdot\chi(G)\right) + o(1).$$

with probability $1 - o(1)$.

Proof: We first show that $\text{dist}(\boldsymbol{f}, \text{Unate}) \leq \frac{\gamma}{16}\left(1 + \frac{1}{2}\cdot\chi(G)\right) + o(1)$ with high probability. Consider the set $S \subset \overline{M}$ which achieves the minimum of $\chi(G)$, i.e.,

$$\chi(G) = \frac{E(S,S) + E(\overline{S},\overline{S})}{E(\overline{M},\overline{M})},$$

and let $\boldsymbol{g}\colon \{0,1\}^n \to \{0,1\}$ be the unate function which makes variables in M monotone, m_1 monotone, m_2 anti-monotone, S monotone, and $\overline{M} \setminus S$ anti-monotone. We defined $\boldsymbol{g}$ as follows:

$$\boldsymbol{g}(x) = \begin{cases} 1 & |x_{|M}| > \frac{n}{4} + \sqrt{n} \\ 0 & |x_{|M}| < \frac{n}{4} - \sqrt{n} \\ 1 & \Gamma_T(x) = 1^* \\ 0 & \Gamma_T(x) = 0^* \\ \boldsymbol{h}'_{\Gamma_T(x)}(x) & \text{otherwise} \end{cases},$$

where we define $\boldsymbol{h}'_i\colon \{0,1\}^n \to \{0,1\}$ as a Boolean function which depends on $\boldsymbol{h}_i$. In particular, if $i \le 3N/4$, we let $\boldsymbol{h}'_i = \boldsymbol{h}_i$. Otherwise, suppose $\boldsymbol{h}_i$ is defined with respect to (j_1, j_2, j_3). There are two cases:

- (Directions of j_1 and j_2 disagree) If $j_1 \in S$ and $j_2 \notin S$, or $j_1 \notin S$ and $j_2 \in S$, then we let $\boldsymbol{h}'_i$ be the function on variables x_{j_1}, x_{j_2} and x_{j_3} with $\mathrm{dist}(\boldsymbol{h}_i, \boldsymbol{h}'_i) = \frac{1}{4}$ (see Figure 6.2.1 for an example with $j_3 = m_1$ which needs to be monotone, $j_1 \in S$ and $j_2 \in \overline{S}$; Figure 6.2.2 and Figure 6.2.2 give the symmetric constructions when j_1 and j_2 are flipped, and when variable m_2 is used instead of m_1, respectively).

- (Directions of j_1 and j_2 agree) If $j_1 \in S$ and $j_2 \in S$, or $j_1 \notin S$ and $j_2 \notin S$, then we let $\boldsymbol{h}'_i$ be the function on variables x_{j_1}, x_{j_2} and x_{j_3} with $\mathrm{dist}(\boldsymbol{h}_i, \boldsymbol{h}'_i) = \frac{3}{8}$ (see Figure 6.2.1 for an example with $j_3 = m_1$ which needs to be monotone, $j_1 \in S$ and $j_2 \in S$; Figure 6.2.2 gives the violating edges of the symmetric examples when variable m_2 is used, and either both j_1 and j_2 are monotone, or both anti-monotone).

Therefore, we define the indicator random variable $\mathbf{C}_i$ for each $i \in \{3N/4+1, \dots, N\}$ by

$$\mathbf{C}_i = \begin{cases} 1 & (\boldsymbol{j}_1, \boldsymbol{j}_2) \text{ from } \boldsymbol{h}_i \text{ is not cut by } S \\ 0 & \text{otherwise} \end{cases},$$

and we note that all $\mathbf{C}_i$ are independent and $\mathbf{Pr}_{\mathbf{H}}[\mathbf{C}_i] = \chi(G)$. By the two cases displayed above, we have that:

$$\mathrm{dist}(\boldsymbol{f}, \boldsymbol{g}) = \frac{1}{2^n} \sum_{i=3N/4+1}^{N} |X_i| \left(\frac{1}{4} + \mathbf{C}_i \cdot \frac{1}{8} \right) \le \frac{\gamma}{16} \left(1 + \frac{1}{2} \cdot \chi(G) \right) + o(1/n),$$

with probability at least $1 - \exp\left(-\Omega(N/n^2)\right)$ over the draw of all $\mathbf{C}_i$.

For the lower bound, consider any function $g\colon \{0,1\}^n \to \{0,1\}$ which is unate. Suppose variable x_{m_1} is anti-monotone in g, then let $\mathbf{C}_i$ for $i \in [3N/4]$ be the indicator random variable

$$\mathbf{C}_i = \begin{cases} 1 & \boldsymbol{h}_i(x) = x_{m_1} \\ 0 & \boldsymbol{h}_i(x) = \neg x_{m_2} \end{cases}.$$

We note that if $\mathbf{C}_i = 1$, then $\boldsymbol{f}$ and g differ on at least $|X_i|/2$ from X_i. Thus, we have $\mathrm{dist}(\boldsymbol{f}, g) \ge \frac{3\gamma}{8}\left(1 - \frac{1}{n}\right) - o(1)$ with high probability over the draw of $\mathbf{C}_i$. Likewise, we may say that if x_{m_2} is monotone, then $\mathrm{dist}(\boldsymbol{f}, g) \ge \frac{3\gamma}{8}\left(1 - \frac{1}{n}\right) - o(1)$. Thus, we may consider functions $g\colon \{0,1\}^n \to \{0,1\}$ with x_{m_1} being monotone and x_{m_2} being anti-monotone. In this case, consider a set $S \subset \overline{M}$, then if g is any unate function with variables in S being monotone and variables in $\overline{M} \setminus S$ being anti-monotone, then we note that for each $i \in \{3N/4+1, \dots, N\}$, if $\boldsymbol{h}_i$ sampled an edge (j_1, j_2) which is cut by S, then X_i must differ on $\frac{1}{4}$th of the points in X_i (see Figure 6.2.1 for an example of the violating edges if j_1 and j_2 are oriented in opposite directions). On the other hand, if (j_1, j_2) is not cut by S, then X_i must differ on $\frac{3}{8}$ths of the points in X_i (see Figure 6.2.1 to see how the violating edges require $\frac{3}{8}$ths of the points being different). Thus, if we let the indicator random variable $\mathbf{C}_i$ be

$$\mathbf{C}_i = \begin{cases} 1 & (\boldsymbol{j}_1, \boldsymbol{j}_2) \text{ from } \boldsymbol{h}_i \text{ is not cut by } S \\ 0 & \text{otherwise} \end{cases},$$

we may write:

$$\mathrm{dist}(\boldsymbol{f}, g) \ge \frac{1}{2^n} \sum_{i=3N/4+1}^{N} |X_i| \left(\frac{1}{4} + \frac{1}{8} \cdot \mathbf{C}_i \right) \ge \frac{\gamma}{16} \left(1 + \frac{1}{2} \cdot \chi(G) \right) + O(1/n),$$

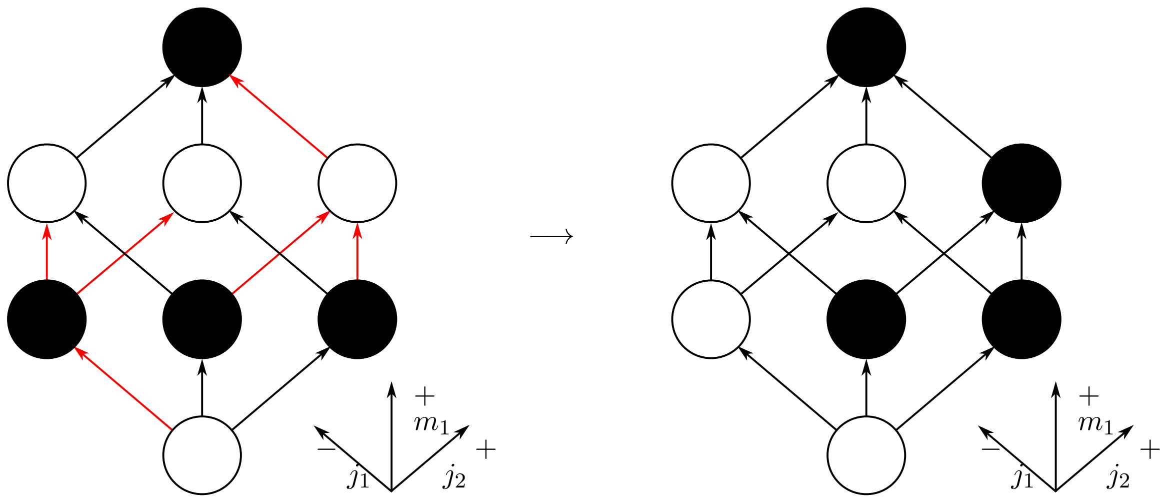


Figure 6.3: Similarly to Figure 6.2.1, this is an example of a function $\boldsymbol{h}_i\colon \{0,1\}^n \to \{0,1\}$ with $\boldsymbol{h}_i(x) = x_{j_1} \oplus x_{j_2} \oplus x_{m_1}$ variables j_1 (which ought to be anti-monotone), j_2 (which ought to be monotone), and m_1 (which is always monotone) being “fixed” into a function $\boldsymbol{h}'_i\colon \{0,1\}^n \to \{0,1\}$ defined on the right-hand side.

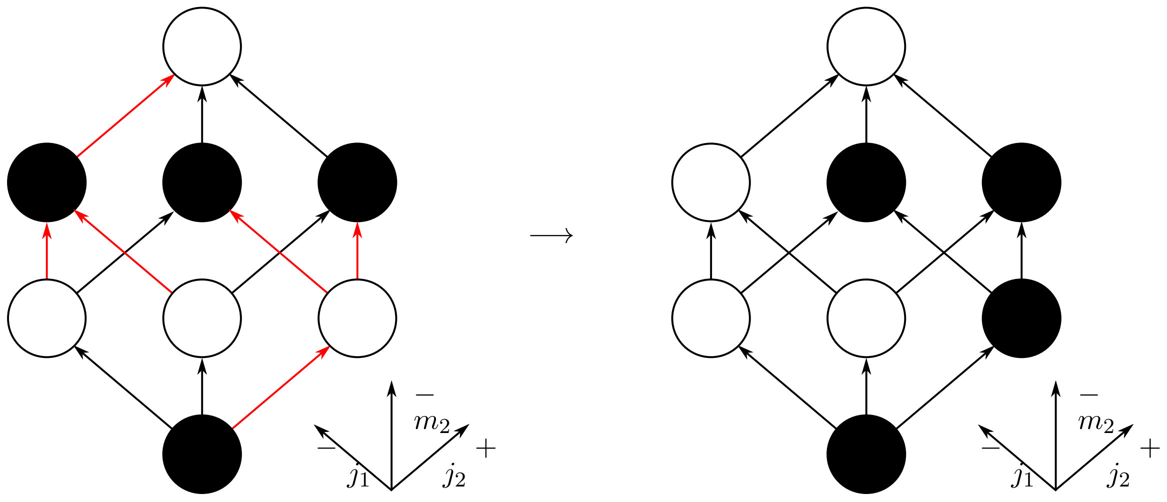
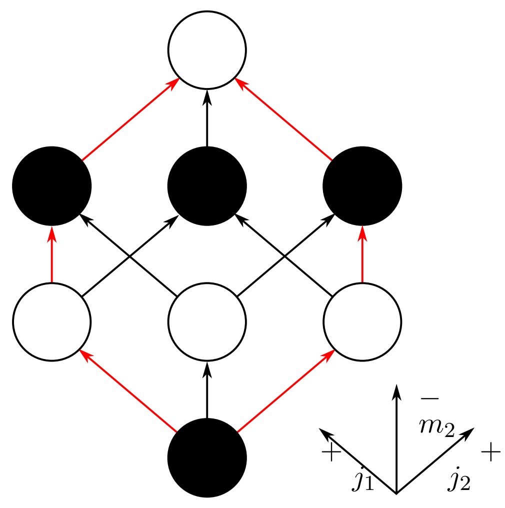


Figure 6.4: Similarly to Figure 6.2.1, this is an example of a function $\boldsymbol{h}_i\colon \{0,1\}^n \to \{0,1\}$ with $\boldsymbol{h}_i(x) = \neg x_{j_1} \oplus x_{j_2} \oplus x_{m_2}$ variables j_1 (which ought to be anti-monotone), j_2 (which ought to be monotone), and m_2 (which is always anti-monotone) being “fixed” into a function $\boldsymbol{h}'_i\colon \{0,1\}^n \to \{0,1\}$ defined on the right-hand side.



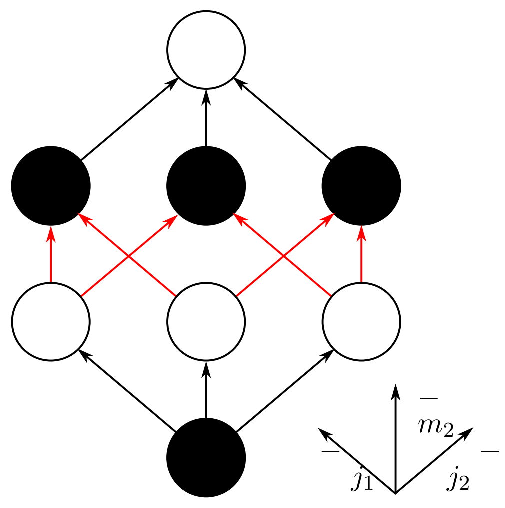


Figure 6.5: Examples of functions $\boldsymbol{h}_i\colon \{0,1\}^n \to \{0,1\}$ with orientations on the variables and violating edges. On the left-hand side, $\boldsymbol{h}_i(x) = \neg x_{j_1} \oplus x_{j_2} \oplus x_{m_2}$ with variables j_1 and j_2 (which ought to be monotone), and m_2 (which is always anti-monotone). On the right-hand side, $\boldsymbol{h}_i(x) = \neg x_{j_1} \oplus x_{j_2} \oplus x_{m_2}$ with variables j_1 and j_2 (which ought to be anti-monotone), and m_2 (which is always anti-monotone). We note that the violating edges form a cycle of length 6, so any unate function whose orientations on j_1 and j_2 are as indicated (both monotone on the left-hand side, and both anti-monotone on the right-hand side) must disagree on a $\frac{3}{8}$-fraction of the points.

with probability $1 - \exp\left(-\Omega(N/n^2)\right)$ over the draw of $\mathbf{C}_i$, since $\mathbf{Pr}[\mathbf{C}_i = 1] \geq \chi(G)$. Thus, we may union bound over all $2^{n/2}$ subsets $S \subset \overline{M}$ to conclude the claim. ■

We consider the constants

$$\varepsilon_0 = \frac{\gamma}{16} \qquad \text{and} \qquad \varepsilon_1 = \frac{5\gamma}{64}.$$

Corollary 6.2.3. *We have that $\boldsymbol{f} \sim \mathcal{D}_{yes}$ has $\mathrm{dist}(\boldsymbol{f}, \mathrm{Unate}) \leq \varepsilon_0 + o(1)$ with high probability, and $\boldsymbol{f} \sim \mathcal{D}_{no}$ has $\mathrm{dist}(\boldsymbol{f}, \mathrm{Unate}) \geq \varepsilon_1 - o(1)$ with high probability.*

Proof: We simply note that when $\mathbf{G} = K_{\mathbf{A},\overline{\mathbf{A}}}$ (as is the case in $\mathcal{D}_{\text{yes}}$), we have $\chi(\mathbf{G}) = 0$, and when $\mathbf{G} = K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}}$, we have $\chi(\mathbf{G}) \to \frac{1}{2}$ as $n \to \infty$. ■

6.2.3 Reducing from Rejection Sampling

The goal of this section is to prove the following two lemmas.

Lemma 6.2.4. *Suppose there exists a deterministic algorithm* Alg *making q queries to Boolean functions $f\colon \{0,1\}^{2n} \to \{0,1\}$. Then, there exists a deterministic non-adaptive algorithm* Alg′ *making rejection sampling queries to an n-vertex graph with* $\mathrm{cost}(\mathrm{Alg}') = qn$ *such that:*

$$\mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{yes}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}] = \mathop{\mathbf{Pr}}_{\mathbf{G} \sim \mathcal{G}_2}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_2\text{''}], \qquad \text{and}$$
$$\mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{no}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}] = \mathop{\mathbf{Pr}}_{\mathbf{G} \sim \mathcal{G}_1}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_2\text{''}].$$

Lemma 6.2.5. *Suppose there exists a deterministic non-adaptive algorithm* Alg *making q queries to Boolean functions $f\colon \{0,1\}^{2n} \to \{0,1\}$ where $q \leq \frac{n^{3/2}}{\log^8 n}$. Then, there exists a deterministic non-adaptive algorithm* Alg′ *making rejection sampling queries to an n-vertex graph* *such that:*

$$\mathop{\mathbf{Pr}}_{\boldsymbol{f}\sim\mathcal{D}_{yes}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}] \approx \mathop{\mathbf{Pr}}_{\mathbf{G}\sim\mathcal{G}_2}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_2\text{''}] \pm o(1), \qquad \text{and}$$
$$\mathop{\mathbf{Pr}}_{\boldsymbol{f}\sim\mathcal{D}_{no}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}] \approx \mathop{\mathbf{Pr}}_{\mathbf{G}\sim\mathcal{G}_1}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_2\text{''}] \pm o(1).$$

and has $\mathrm{cost}(\mathrm{Alg}') \leq q\sqrt{n}\log n$ *with probability* $1 - o(1)$ *over the randomness in* Alg'.

Combining Lemma 6.2.4 with Theorem 4.0.1, we conclude Theorem 1.3.6, and combining Lemma 6.2.5 with Theorem 4.0.1, we conclude Theorem 1.3.7.

6.2.4 Proof of Lemma 6.2.4

Consider an algorithm Alg making q queries to a Boolean function which receives access to a Boolean function $\boldsymbol{f} = f_{\mathbf{T},\mathbf{A},\mathbf{H}} : \{0,1\}^{2n} \to \{0,1\}$ (sampled from either $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$).

Since the values of $\mathbf{M}, \boldsymbol{m}_1, \boldsymbol{m}_2$ and $\mathbf{T}$ are distributed in the same way in $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, a rejection sampling algorithm may generate $\mathbf{M}, \boldsymbol{m}_1, \boldsymbol{m}_2$ and $\mathbf{T}$, and utilize the randomness from rejection sampling to output values of $\mathbf{H}$. In particular, for each query in Alg, we will query the set $[n]$ in the rejection sampling algorithm. Then, given the edges sampled, as well as the values of $\mathbf{M}$, $\boldsymbol{m}_1$, $\boldsymbol{m}_2$ and $\mathbf{T}$, we will be able to simulate all the randomness in the construction of $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$. We give a formal description of a rejection sampling algorithm Alg$'$ which assumes access to an algorithm Alg testing Boolean functions.

1. We first sample $\mathbf{M} \subset [2n]$ of size n, and let $\boldsymbol{m}_1, \boldsymbol{m}_2 \sim \mathbf{M}$ be two distinct indices. Sample $\mathbf{T} \sim \mathcal{E}(\mathbf{M} \setminus \{\boldsymbol{m}_1, \boldsymbol{m}_2\})$. We may now view the hidden graph $\mathbf{G}$ (from rejection sampling) as a graph on vertex set $\overline{\mathbf{M}}$.

2. For each $t \in [q]$, perform the query $L_t = \overline{\mathbf{M}}$, which returns $(j_1^{(t)}, j_2^{(t)}) \in \mathbf{G}$, we sample $\boldsymbol{j}_3^{(t)} \sim \{\boldsymbol{m}_1, \boldsymbol{m}_2\}$ and $\boldsymbol{j}^{(t)} \sim \{\boldsymbol{m}_1, \boldsymbol{m}_2\}$. Intuitively, the values of $(j_1^{(t)}, j_2^{(t)}, \boldsymbol{j}_3^{(t)})$ will generate the t-th accessed subfunction $\boldsymbol{h}_i$ with $\Gamma_{\mathbf{T}}(x) > 3N/4$, and $\boldsymbol{j}^{(t)}$ will generate the t-th accessed subfunction $\boldsymbol{h}_i$ with $\Gamma_{\mathbf{T}}(x) \leq 3N/4$.

3. We simulate Alg by maintaining two q-tuples $p_1, p_2 \in (\{0\} \cup [N])^q$, which is initially $p_1 = p_2 = (0, 0, \dots 0)$ which will record the indices of the subfunctions accessed. We proceed as follows, where we assume that Alg makes the query $z \in \{0,1\}^{2n}$:

    - Suppose $|z_{|\mathbf{M}}| > \frac{n}{2} + \sqrt{2n}$, $|z_{|\mathbf{M}}| < \frac{n}{2} - \sqrt{2n}$, $\Gamma_{\mathbf{T}}(z) = 1^*$, or $\Gamma_{\mathbf{T}}(z) = 0^*$, report to Alg the appropriate value of $\boldsymbol{f}(x)$.
    - Otherwise, consider $\Gamma_{\mathbf{T}}(z) = i \in [N]$.
        - Suppose $i \leq \frac{3N}{4}$ and $(p_1)_t = i$ (if $(p_1)_t \neq i$ for all t, then find the first $t \in [q]$ with $(p_1)_t = 0$ and write $(p_1)_t = i$). In this case, report $z_{\boldsymbol{j}^{(t)}}$ if $\boldsymbol{j}^{(t)} = \boldsymbol{m}_1$ and $\neg z_{\boldsymbol{j}^{(t)}}$ if $\boldsymbol{j}^{(t)} = \boldsymbol{m}_2$.
        - If $i > \frac{3N}{4}$ and $(p_2)_t = i$ (again, if $(p_2)_t \neq i$ for all t, then find the first $t \in [q]$ with $(p_2)_t = 0$ and write $(p_2)_t = i$). In this case, we report $\neg z_{j_1^{(t)}} \oplus z_{j_2^{(t)}} \oplus z_{\boldsymbol{j}_3^{(t)}}$ if $\boldsymbol{j}_3^{(t)} = \boldsymbol{m}_1$ and $z_{j_1^{(t)}} \oplus z_{j_2^{(t)}} \oplus z_{\boldsymbol{j}_3^{(t)}}$ if $\boldsymbol{j}_3^{(t)} = \boldsymbol{m}_2$.

4. If Alg outputs "accept", then Alg$'$ outputs "$\mathcal{G}_2$", if Alg outputs "reject", then Alg$'$ outputs "$\mathcal{G}_1$".

Clearly, $\mathrm{cost}(\mathrm{Alg}') = qn$. In addition, we may view $\mathrm{Alg}'(\mathbf{G})$ as generating the necessary randomness for answering queries $\boldsymbol{f}(x)$ on the go, where $\mathbf{G}$ will determine whether $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{yes}}$ or $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{no}}$. When $\mathbf{G} = K_{\mathbf{A},\overline{\mathbf{A}}}$ (in the case $\mathbf{G} \sim \mathcal{G}_2$, the resulting function $\boldsymbol{f}$ is distributed as a function drawn from $\mathcal{D}_{\mathrm{yes}}$; when $\mathbf{G} = K_{\mathbf{A}} \cup K_{\overline{\mathbf{A}}}$ (in the case $\mathbf{G} \sim \mathcal{G}_1$), the resulting function $\boldsymbol{f}$ is distributed as a function drawn from $\mathcal{D}_{\mathrm{no}}$. Therefore, by the principle of deferred decisions, we have that $\mathrm{Alg}'(\mathbf{G})$ perfectly simulates queries to a Boolean function $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{yes}}$ (if $\mathbf{G} \sim \mathcal{G}_2$) or $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{no}}$ (if $\mathbf{G} \sim \mathcal{G}_1$). We conclude that

$$\underset{\mathbf{G}\sim\mathcal{G}_1}{\mathbf{Pr}}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_2\text{''}] = \underset{\boldsymbol{f}\sim\mathcal{D}_{\mathrm{yes}}}{\mathbf{Pr}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}], \qquad \text{and}$$
$$\underset{\mathbf{G}\sim\mathcal{G}_2}{\mathbf{Pr}}[\mathrm{Alg}'(\mathbf{G}) \text{ outputs ``}\mathcal{G}_2\text{''}] = \underset{\boldsymbol{f}\sim\mathcal{D}_{\mathrm{no}}}{\mathbf{Pr}}[\mathrm{Alg}(\boldsymbol{f}) \text{ ``accepts''}].$$

Remark. A close inspection of the proof of Lemma 6.2.4 reveals that the rejection sampling algorithm distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$ always makes queries $L_i = [n]$. This makes the lower bound simpler, as we can focus on proving lower bounds against algorithms which receive random edge samples.

6.2.5 Proof of Lemma 6.2.5

Similarly to the proof of Lemma 6.2.4, we will proceed by generating the necessary randomness to generate the functions $\boldsymbol{f}$ from $\mathcal{D}_{\mathrm{yes}}$ or from $\mathcal{D}_{\mathrm{no}}$. However, unlike Lemma 6.2.4, this will not be a black box reduction, since we will not be able to simulate $\boldsymbol{f}$ exactly.

Consider a deterministic non-adaptive algorithm Alg which makes queries to a Boolean function $\boldsymbol{f} \colon \{0,1\}^{2n} \to \{0,1\}$ sampled from $\mathcal{D}_{\mathrm{yes}}$ or $\mathcal{D}_{\mathrm{no}}$ and outputs "accept" if Alg believes $\boldsymbol{f}$ was sampled from $\mathcal{D}_{\mathrm{yes}}$, and outputs "reject" if Alg believes $\boldsymbol{f}$ was sampled from $\mathcal{D}_{\mathrm{no}}$. Since Alg is non-adaptive and deterministic, all queries are determined, so consider the queries $z_1, \dots, z_q \in \{0,1\}^{2n}$, and let $\mathrm{Alg} \colon \{0,1\}^q \to \{\text{“accept”}, \text{“reject”}\}$ be a function.

We will now define a non-adaptive algorithm Alg' which makes rejection sampling queries to an unknown graph $\mathbf{G}$ on n vertices sampled from $\mathcal{G}_1$ or from $\mathcal{G}_2$. The algorithm Alg' proceeds as follows:

1. Using some randomness and answers from rejection sampling queries to an unknown graph $\mathbf{G}$, we generate a sequence of r bits $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ satisfying the following two conditions (we give the procedure to generate these random bits after)[5]:
   - If $\mathbf{G} \sim \mathcal{G}_1$, then $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ will be roughly distributed as $(\boldsymbol{f}(z_1), \dots, \boldsymbol{f}(z_q))$ where $\boldsymbol{f}$ is a Boolean function $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{no}}$.
   - If $\mathbf{G} \sim \mathcal{G}_2$, then $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ will be roughly distributed as a $(\boldsymbol{f}(z_1), \dots, \boldsymbol{f}(z_q))$ where $\boldsymbol{f}$ is a Boolean function $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{yes}}$.
2. Finally, if $\mathrm{Alg}(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ outputs "accept", then Alg' outputs "$\mathcal{G}_2$", and if $\mathrm{Alg}(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ outputs "reject", then Alg' outputs "$\mathcal{G}_1$".

In order to formalize the notion of "roughly distributed as" from above, let $\mathcal{V}_{\mathrm{yes}}$ and $\mathcal{V}_{\mathrm{no}}$ be the distributions supported on $\{0,1\}^q$ given by:

$$\begin{aligned} \boldsymbol{r} \sim \mathcal{V}_{\mathrm{yes}} \qquad &\text{where} \qquad \forall i \in [q], \boldsymbol{r}_i = \boldsymbol{f}(z_i), \text{ and } \boldsymbol{f} \sim \mathcal{D}_{\mathrm{yes}}. \\ \boldsymbol{r} \sim \mathcal{V}_{\mathrm{no}} \qquad &\text{where} \qquad \forall i \in [q], \boldsymbol{r}_i = \boldsymbol{f}(z_i), \text{ and } \boldsymbol{f} \sim \mathcal{D}_{\mathrm{no}}. \end{aligned}$$

[5]With a slight abuse of notation, we let $\mathrm{Alg}'(\mathbf{G})$ correspond to to the output $(\boldsymbol{r}_1, \dots, \boldsymbol{r}_q)$ that Alg' produces with rejection sampling access to graph $\mathbf{G}$.

Now, given the algorithm Alg$'$, we let $\mathcal{U}_{\text{yes}}, \mathcal{U}_{\text{no}}$ be the distributions supported in $\{0,1\}^q$ given by:

$$\begin{array}{lll} \boldsymbol{r} \sim \mathcal{U}_{\text{yes}} & \text{where} & \text{Alg}'(\mathbf{G}) \text{ outputs } (\boldsymbol{r}_1, \dots, \boldsymbol{r}_q) \text{ when } \mathbf{G} \sim \mathcal{G}_2 \\ \boldsymbol{r} \sim \mathcal{U}_{\text{no}} & \text{where} & \text{Alg}'(\mathbf{G}) \text{ outputs } (\boldsymbol{r}_1, \dots, \boldsymbol{r}_q) \text{ when } \mathbf{G} \sim \mathcal{G}_1 \end{array}$$

The following lemma is a simple consequence will allow us to conclude Lemma 6.2.5.

Lemma 6.2.6. *Suppose $\mathcal{V}_{yes}, \mathcal{V}_{no}, \mathcal{U}_{yes}$ and $\mathcal{U}_{no}$ satisfy:*

$$d_{TV}(\mathcal{V}_{yes}, \mathcal{U}_{yes}) = o(1) \qquad \textit{and} \qquad d_{TV}(\mathcal{V}_{no}, \mathcal{U}_{no}) = o(1).$$

Then, we have that:

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\mathbf{G} \sim \mathcal{G}_1}[\text{Alg}'(\mathbf{G}) \textit{ outputs "}\mathcal{G}_1\textit{"}] &\approx \mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{no}}[\text{Alg}(\boldsymbol{f}) \textit{ "rejects"}] \pm o(1). \\ \mathop{\mathbf{Pr}}_{\mathbf{G} \sim \mathcal{G}_2}[\text{Alg}'(\mathbf{G}) \textit{ outputs "}\mathcal{G}_2\textit{"}] &\approx \mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{yes}}[\text{Alg}(\boldsymbol{f}) \textit{ "accepts"}] \pm o(1). \end{aligned}$$

Proof: We show the first inequality in the conclusion, as the argument is the same for the second inequality. Consider the set $R = \{r \in \{0,1\}^q : \text{Alg}(r) = \text{"reject"}\}$. Then, we have:

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\boldsymbol{f} \sim \mathcal{D}_{\text{no}}}[\text{Alg}(\boldsymbol{f}) \text{ "rejects"}] &= \mathop{\mathbf{Pr}}_{\boldsymbol{r} \sim \mathcal{V}_{\text{no}}}[\boldsymbol{r} \in R] \\ &\approx \mathop{\mathbf{Pr}}_{\boldsymbol{r} \sim \mathcal{U}_{\text{no}}}[\boldsymbol{r} \in R] \pm o(1) \\ &\approx \mathop{\mathbf{Pr}}_{\mathbf{G} \sim \mathcal{G}_1}[\text{Alg}'(\mathbf{G}) \text{ outputs "accept"}] \pm o(1). \end{aligned}$$

■

Given Lemma 6.2.6, it remains to describe the randomized procedure Alg$'$ which given rejection sampling access to an unknown n-vertex graph $\mathbf{G}$ from $\mathcal{G}_1$ or $\mathcal{G}_2$ outputs a bit-string of length q such that:

$$d_{TV}(\mathcal{V}_{\text{yes}}, \mathcal{U}_{\text{yes}}) = o(1) \qquad \text{and} \qquad d_{TV}(\mathcal{V}_{\text{no}}, \mathcal{U}_{\text{no}}) = o(1).$$

The procedure will work as follows:

1. First, sample a random subset $\mathbf{M} \subset [2n]$ of size n, and let $\boldsymbol{m}_1, \boldsymbol{m}_2 \sim \mathbf{M}$ be two distinct random indices, and let $\mathbf{T} \sim \mathcal{E}(\mathbf{M} \setminus \{\boldsymbol{m}_1, \boldsymbol{m}_2\})$. This defines an indexing function[6] $\Gamma_{\mathbf{T}} \colon \{0,1\}^{2n} \to [N]$. We may view the unknown graph $\mathbf{G}$ as being defined over vertices in $\overline{\mathbf{M}}$ [7].

2. We now consider partitioning the queries $z_1, \dots, z_q \in \{0,1\}^{2n}$ into at most $t+4$ sets (where we will have $t \le q$) $\mathbf{Q}_{\mathbf{M}}^{(+)}, \mathbf{Q}_{\mathbf{M}}^{(-)}, \mathbf{Q}_*^{(0)}, \mathbf{Q}_*(1)$ and $\mathbf{Q}_{\ell_1}, \dots, \mathbf{Q}_{\ell_t}$ non-empty sets where $\ell_1, \dots, \ell_t \subset [N]$:

$$\begin{aligned} \mathbf{Q}_{\mathbf{M}}^{(-)} &= \left\{ z_i : |(z_i)_{|\mathbf{M}}| < \frac{n}{2} - \sqrt{2n} \right\}, \\ \mathbf{Q}_{\mathbf{M}}^{(+)} &= \left\{ z_i : |(z_i)_{|\mathbf{M}}| > \frac{n}{2} + \sqrt{2n} \right\}, \\ \mathbf{Q}_*^{(0)} &= \left\{ z_i : \Gamma_{\mathbf{T}}(z_i) = 0^* \wedge z_i \notin \mathbf{Q}_{\mathbf{M}}^{(-)} \cup \mathbf{Q}_{\mathbf{M}}^{(+)} \right\}, \\ \mathbf{Q}_*^{(1)} &= \left\{ z_i : \Gamma_{\mathbf{T}}(z_i) = 1^* \wedge z_i \notin \mathbf{Q}_{\mathbf{M}}^{(-)} \cup \mathbf{Q}_{\mathbf{M}}^{(+)} \right\}, \\ \mathbf{Q}_\ell &= \left\{ z_i : \Gamma_{\mathbf{T}}(z_i) = \ell \wedge z_i \notin \mathbf{Q}_{\mathbf{M}}^{(-)} \cup \mathbf{Q}_{\mathbf{M}}^{(+)} \right\}. \end{aligned}$$

[6]Note that now, $N = 2^{\sqrt{2n}}$ since we are considering Boolean functions with $2n$ variables.

[7]We may assume this by picking an arbitrary mapping of the indices in $\overline{\mathbf{M}}$ to $[n]$.

3. If $z_i \in \mathbf{Q}_M^{(-)}$, we let $\boldsymbol{r}_i = 0$, if $z_i \in \mathbf{Q}_M^{(+)}$, we let $\boldsymbol{r}_i = 1$. If $z_i \in \mathbf{Q}_*^{(0)}$, we let $\boldsymbol{r}_i = 0$, and if $z_i \in \mathbf{Q}_*^{(1)}$, we let $\boldsymbol{r}_i = 1$. We may thus only consider the queries in $\mathbf{Q}_{\ell_1}, \ldots, \mathbf{Q}_{\ell_t}$, and for simplicity in the notation, we re-index the queries to let:
$$\mathbf{Q}_{\ell_i} = \left\{ z_1^{(i)}, z_2^{(i)}, \ldots, z_{|\mathbf{Q}_{\ell_i}|}^{(i)} \right\}$$
for each $i \in [t]$, and the corresponding bits $\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)}, \ldots, \boldsymbol{r}_{|\mathbf{Q}_{\ell_i}|}^{(i)}$.

4. We thus consider each $i \in [t]$ and independently set the values of $\boldsymbol{r}_1^{(i)}, \ldots, \boldsymbol{r}_{|\mathbf{Q}_{\ell_i}|}^{(i)}$ as follows:
   (a) If $\ell_i \leq 3N/4$, sample some $\boldsymbol{j} \sim \{\boldsymbol{m}_1, \boldsymbol{m}_2\}$, and for every $\alpha \in [|\mathbf{Q}_{\ell_i}|]$, let:
$$\boldsymbol{r}_\alpha^{(i)} = \begin{cases} (z_\alpha^{(i)})_{\boldsymbol{j}} & \boldsymbol{j} = \boldsymbol{m}_1 \\ \neg (z_\alpha^{(i)})_{\boldsymbol{j}} & \boldsymbol{j} = \boldsymbol{m}_2 \end{cases}.$$
   (b) Otherwise, if $\ell_i > 3N/4$, consider the following sets
$$\mathbf{L}_i = \left\{ k \in \overline{\mathbf{M}} : \exists \alpha, \beta \in [|\mathbf{Q}_{\ell_i}|], (z_\alpha^{(i)})_k \neq (z_\beta^{(i)})_k \right\},$$
   and,
$$\overline{\mathbf{L}}_i^{(0)} = \left\{ k \in \overline{\mathbf{M}} \setminus \mathbf{L}_i : z \in \mathbf{Q}_{\ell_i}, z_k = 0 \right\} \qquad \overline{\mathbf{L}}_i^{(1)} = \left\{ k \in \overline{\mathbf{M}} \setminus \mathbf{L}_i : z \in \mathbf{Q}_{\ell_i}, z_k = 1 \right\}.$$
   We make the query $\mathbf{L}_i$ if $|\mathbf{L}_i| \leq \frac{n}{\log n}$ and $\overline{\mathbf{M}}$ otherwise to the rejection sampling oracle and obtain a response $\boldsymbol{v} \in (\overline{\mathbf{M}} \times \overline{\mathbf{M}}) \cup \overline{\mathbf{M}} \cup \{\emptyset\}$. In addition, sample $\boldsymbol{j}_3 \sim \{\boldsymbol{m}_1, \boldsymbol{m}_2\}$. We now consider three cases:
      i. If $\boldsymbol{v} = (\boldsymbol{j}_1, \boldsymbol{j}_2) \in \overline{\mathbf{M}} \times \overline{\mathbf{M}}$ is an edge, then for each $\alpha \in [|\mathbf{Q}_{\ell_i}|]$, we let:
$$\boldsymbol{r}_\alpha^{(i)} = \begin{cases} (z_\alpha^{(i)})_{\boldsymbol{j}_1} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_2} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_3} & \boldsymbol{j}_3 = \boldsymbol{m}_1 \\ \neg (z_\alpha^{(i)})_{\boldsymbol{j}_1} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_2} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_3} & \boldsymbol{j}_3 = \boldsymbol{m}_2 \end{cases}.$$
      ii. If $\boldsymbol{v} = \boldsymbol{j}_2 \in \overline{\mathbf{M}}$ is a lone vertex, then let $w = \neg (z_1^{(i)})_{\boldsymbol{j}_2}$ and $p_v(\overline{\mathbf{L}}_i^{(w)}) = \frac{|\overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{L}}_i|}$, we sample $\boldsymbol{b} \sim \mathrm{Ber}(p_v(\overline{\mathbf{L}}_i^{(w)}))$ and for each $\alpha \in [|\mathbf{Q}_{\ell_i}|]$, we let:
$$\boldsymbol{r}_\alpha^{(i)} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_2} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_3} = \begin{cases} \boldsymbol{b} \oplus (z_1^{(i)})_{\boldsymbol{j}_2} & \boldsymbol{j}_3 = \boldsymbol{m}_1 \\ \neg \boldsymbol{b} \oplus (z_1^{(i)})_{\boldsymbol{j}_2} & \boldsymbol{j}_3 = \boldsymbol{m}_2 \end{cases}.$$
      iii. Lastly, if $\boldsymbol{v} = \emptyset$ is the empty set, then let $p_\emptyset(\overline{\mathbf{L}}_i) = \frac{2|\overline{\mathbf{L}}_i^{(0)}||\overline{\mathbf{L}}_i^{(1)}|}{|\overline{\mathbf{L}}_i|^2}$ and sample $\boldsymbol{b} \sim \mathrm{Ber}(p(\overline{\mathbf{L}}_i))$ and for each $\alpha \in [|\mathbf{Q}_{\ell_i}|]$, we let:
$$\boldsymbol{r}_\alpha^{(i)} \oplus (z_\alpha^{(i)})_{\boldsymbol{j}_3} = \begin{cases} \boldsymbol{b} & \boldsymbol{j}_3 = \boldsymbol{m}_1 \\ \neg \boldsymbol{b} & \boldsymbol{j}_3 = \boldsymbol{m}_2 \end{cases}.$$

Remark. The procedure described above does not exactly simulate queries to a $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$ (in the case of $\mathbf{G} \sim \mathcal{G}_2$ or $\mathbf{G} \sim \mathcal{G}_1$, respectively) as in the reductions of Lemma 6.2.4 and Lemma 4.2.3). Let us briefly explain why this happens by giving an illuminating example. Consider a one-query algorithm which makes query $z \in \{0,1\}^n$ and suppose $|z_{\mathbf{M}}| \approx \frac{n}{2} \pm \sqrt{2n}$

and $\Gamma_{\mathbf{T}}(z) = i > \frac{3N}{4}$ with $z_{m_1} = 0$ and $z_{m_2} = 1$. Then, the value $\boldsymbol{f}(z) = \boldsymbol{h}_i(z)$ will be 0 if $z_{\boldsymbol{j}_1} = z_{\boldsymbol{j}_2}$, and 1 if $z_{\boldsymbol{j}_1} \neq z_{\boldsymbol{j}_2}$, where $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ is the edge sampled for subfunction $\boldsymbol{h}_i$.

We note that this probability is slightly different for $\mathbf{G} \sim \mathcal{G}_1$ and $\mathbf{G} \sim \mathcal{G}_2$ and depends on how $\mathbf{A}$ partitions the 0-variables and 1-variables of z. Despite this difference, Alg$'$ always observes $\emptyset$ from the rejection sampling oracle, so the output bit $\boldsymbol{r} \in \{0,1\}$ which Alg$'$ produces will not simulate $\boldsymbol{f}(z)$ exactly. The bulk of the argument shows that Alg$'$ can sample a random bit whose distribution is close to $\boldsymbol{f}(z)$ in total variation distance, so that Alg cannot exploit the fact that the simulation is not exact.

We first note the following lemma.

Lemma 6.2.7. *With probability $1 - o(1)$ over the draw of $\mathbf{M} \subset [n]$, $\boldsymbol{m}_1, \boldsymbol{m}_2$ and $\mathbf{T} \sim \mathcal{E}(\mathbf{M} \setminus \{\boldsymbol{m}_1, \boldsymbol{m}_2\})$, we have that for all $i \in [t]$,*

$$|\mathbf{L}_i| \leq |\mathbf{Q}_{\ell_i}| \cdot 90\sqrt{n}\log n.$$

Proof: We will prove this by showing that for any two $z, z' \in \mathbf{Q}_{\ell_i}$, $\|z - z'\|_1 \leq 90\sqrt{n}\log n$ with probability $1 - \frac{1}{n^{10}}$, so that we may union bound over all possible pairs. More specifically, consider two queries $z, z' \in \{0,1\}^{2n}$ which differ by more than $90\sqrt{n}\log n$ indices. Note that the distribution of the random variable $\|(z-z')_{|\mathbf{M}}\|_1 \sim \mathrm{HG}(2n, |z-z'|, n)$. Then using Theorem 2.4.3 we have that with probability at least $1 - \frac{1}{n^{10}}$ over the draw of $\mathbf{M}$, $\|z_{|\mathbf{M}} - z'_{|\mathbf{M}}\|_1 \geq 30\sqrt{n}\log n$.

Next, if $|z_{|\mathbf{M}}| \approx \frac{n}{2} \pm \sqrt{2n}$ and $|z'_{|\mathbf{M}}| \approx \frac{n}{2} \pm \sqrt{2n}$ (if either of these conditions do not hold, then we know the strings are not in $\mathbf{Q}_{\ell_i}$ for any i), then there exists a set $\mathbf{P} \subset \mathbf{M}$ with $|\mathbf{P}| = 15\sqrt{n}\log n$ such that for all $k \in \mathbf{P}$, $z_k = 1$ and $z'_k = 0$. Thus, we have that:

$$\Pr_{\mathbf{T}}[\exists i \in [t], z, z' \in \mathbf{Q}_{\ell_i}] \leq \Pr_{\mathbf{T}}[z' \in \mathbf{Q}_{\ell_i} \mid z \in \mathbf{Q}_{\ell_i}] \leq \Pr_{\mathbf{T}_{\ell_i}}[\mathbf{T}_{\ell_i} \cap \mathbf{P} = \emptyset] \leq \left(1 - \frac{15\log n}{\sqrt{n}}\right)^{\sqrt{n}} \ll \frac{1}{n^{10}}.$$

So we may union bound over all pairs of queries to conclude that if $z, z' \in \mathbf{Q}_{\ell_i}$, then $\|z - z'\|_1 \leq 90\sqrt{n}\log n$ with high probability, which gives the desired claim. ∎

Thus, given Lemma 6.2.7 as well as the fact that we query $[n]$ when $|\mathbf{L}_i| \geq \frac{n}{\log n}$, we conclude that if Alg makes q queries, then Alg$'$ has complexity at most $q \cdot O(\sqrt{n}\log^2 n)$ in the rejection sampling model.

Lemma 6.2.8. *If $q \leq \frac{n^{3/2}}{\log^8 n}$, then with probability $1 - o(1)$ over the draw of $\mathbf{M} \subset [n], \boldsymbol{m}_1, \boldsymbol{m}_2$, $\mathbf{T}$, and $\mathbf{A} \subset \overline{\mathbf{M}}$, we have that for every $i \in [t]$ where $|\mathbf{L}_i| \leq \frac{n}{\log n}$, the sets $|\overline{\mathbf{L}}_i^{(0)}|, |\overline{\mathbf{L}}_i^{(1)}|$ satisfy the following*

$$|\overline{\mathbf{L}}_i^{(0)}|, |\overline{\mathbf{L}}_i^{(1)}| = \Omega(n)\ ,$$

$$\left|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(0)}\right| \approx \frac{\left|\overline{\mathbf{L}}_i^{(0)}\right|}{2} \pm \sqrt{n}\log n \qquad \textit{and} \qquad \left|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(1)}\right| \approx \frac{\left|\overline{\mathbf{L}}_i^{(1)}\right|}{2} \pm \sqrt{n}\log n.$$

Proof: We first claim that with probability $1 - o(1)$ over the choice of $\mathbf{M}$, all the queries $z \in \{0,1\}^{2n}$ that are mapped to some $\mathbf{Q}_{\ell_i}$ are such that $|z| \approx n \pm 50\sqrt{2n}\log n$. Assume $z \in \{0,1\}^{2n}$ is such that $|z| > n + 50\sqrt{2n}\log n$, and consider the random variable $|z|_{\mathbf{M}}|$. Note that the distribution of $|z|_{\mathbf{M}}|$ is hyper-geometric with parameters $(2n, |z|, n)$. By using Theorem 2.4.3 on the tail bounds for hyper-geometric random variable, we get that for any $t > 0$

$$\Pr_{\mathbf{M}}\left[|z|_{\mathbf{M}}| < \left(\frac{|z|}{2n} - t\right)n\right] \leq e^{-2t^2 n}\ .$$

By choosing $t = \frac{50\log n}{\sqrt{2n}} - \frac{\sqrt{2}}{\sqrt{n}}$, and considering the complement event, we have that

$$\Pr_{\mathbf{M}}\left[|z|_{\mathbf{M}}| \geq \frac{|z|}{2} - \frac{50\sqrt{n}\log n}{\sqrt{2}} + \sqrt{2n}\right] \geq 1 - \frac{1}{n^{50}} \ .$$

Combining this with the fact that $|z| > n + 50\sqrt{2n}\log n$, we get that the probability that $|z|_{\mathbf{M}}| > n/2 + \sqrt{2n}$ is at least $1 - 1/n^{50}$.

Similarly, we get that when $|z| < n - 50\sqrt{2n}\log n$, we have that with probability $1 - 1/n^{50}$ over the choice of $\mathbf{M}$, $|z|_{\mathbf{M}}| < n/2 - \sqrt{2n}$. By using a union bound on the number of queries we get that with probability $1 - o(1)$ over the choice of $\mathbf{M}$, all the queries $z \in \{0,1\}^{2n}$ that are mapped to some $\mathbf{Q}_{\ell_i}$ are such that $|z| \approx n \pm 50\sqrt{2n}\log n$.

We henceforth condition on such $\mathbf{M} = M$. Consider any $T \sim \mathcal{E}(M)$ and all the indices $i \in [t]$ such that $|L_i| \leq \frac{n}{\log n}$. By definition, if $z \in \{0,1\}^{2n}$ is mapped to some Q_{ℓ_i}, then $|z|_M| \approx n/2 \pm \sqrt{2n}$, which implies that $|z|_{\overline{M}}| \approx n/2 \pm 49\sqrt{2n}\log n$. Therefore, by the fact that all queries in Q_{ℓ_i} must agree on all of the coordinates in $\overline{L}_i$, we can conclude that $|\overline{L}_i^{(0)}|$ and $|\overline{L}_i^{(1)}|$ are $\Omega(n)$.

Next, consider the random variable $|\mathbf{A} \cap \overline{L}_i^{(1)}|$, and note that its distribution is hyper-geometric with parameters $(n, |\overline{L}_i^{(1)}|, n/2)$. By using tail bounds for hyper-geometric random variable, we get that with probability at least $1 - o(1)$ over the choice of $\mathbf{A}$

$$|\mathbf{A} \cap \overline{L}_i^{(1)}| \approx \frac{|\overline{L}_i^{(1)}|}{2} \pm \sqrt{n}\log n \ .$$

Using the same argument, we also get that with probability $1 - o(1)$ over the choice of $\mathbf{A}$ we have that

$$|\mathbf{A} \cap \overline{L}_i^{(0)}| \approx \frac{|\overline{L}_i^{(0)}|}{2} \pm \sqrt{n}\log n \ .$$

By applying a union bound over all indices $i \in [t]$ the lemma follows. ■

Lemma 6.2.9. *If* $\mathrm{cost}(\mathrm{Alg}') \leq \frac{n^2}{\log^6 n}$ *which occurs with high probability over* $\mathbf{M}$*, with probability* $1 - o(1)$ *over the draw of* $\boldsymbol{v}$ *in Step 4(b), there are at most* $\frac{n}{\log^4 n}$ *responses* $\boldsymbol{v} \in \overline{\mathbf{M}}$ *which are lone vertices of case (ii).*

As discussed earlier, the proof of the above lemma is given in the lower bound for distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$ in Section 4.3 (Lemma 4.3.14). We assume its correctness for the rest of this section.

We note that since $\mathbf{M}, \boldsymbol{m}_1, \boldsymbol{m}_2$ and $\mathbf{T}$ are distributed in the same way in $\boldsymbol{f} \sim \mathcal{D}_{\mathrm{yes}}$ and in Step 1 of Alg, we may consider the distribution $\mathcal{V}_{\mathrm{yes}}(M, m_1, m_2, T)$ denoting $\mathcal{V}_{\mathrm{yes}}$ conditioned on $\mathbf{M} = M, \boldsymbol{m}_1 = m_1, \boldsymbol{m}_2 = m_2$ and $\mathbf{T} = T$, and we analogously define $\mathcal{U}_{\mathrm{yes}}(M, m_1, m_2, T)$, $\mathcal{V}_{\mathrm{no}}(M, m_1, m_2, T)$ and $\mathcal{U}_{\mathrm{no}}(M, m_1, m_2, T)$. In addition, we may denote the event $\boldsymbol{\mathcal{E}}_A$ to denote the event that the hidden subset $\mathbf{A}$ sampled in $\boldsymbol{f}$ or in the graph $\mathbf{G}$ satisfies the conditions of Lemma 6.2.8, and the event $\boldsymbol{\mathcal{E}}_V$ to be the event that there are at most $\frac{n}{\log^4 n}$ responses which are lone vertices from Lemma 6.2.9. We thus consider a fixed set M, m_1, m_2, and T satisfying the following conditions of Lemma 6.2.7 and consider the distribution $\mathcal{V}'_{\mathrm{yes}}$ to be the distribution given by sampling $\boldsymbol{r} \sim \mathcal{V}_{\mathrm{yes}}(M, m_1, m_2, T)$ conditioned on events $\boldsymbol{\mathcal{E}}_A$ and $\boldsymbol{\mathcal{E}}_V$. We analogously define $\mathcal{V}'_{\mathrm{no}}$, $\mathcal{U}'_{\mathrm{yes}}$ and $\mathcal{U}'_{\mathrm{no}}$. We note it suffices to show $d_{TV}(\mathcal{V}'_{\mathrm{yes}}, \mathcal{U}'_{\mathrm{yes}}) = o(1)$ and $d_{TV}(\mathcal{V}'_{\mathrm{no}}, \mathcal{U}'_{\mathrm{no}}) = o(1)$.

We now note that conditioned on M, m_1, m_2 and T, the sets $\mathbf{Q}_{\mathbf{M}}^{(+)}, \mathbf{Q}_{\mathbf{M}}^{(-)}, \mathbf{Q}_*^{(1)}$ and $\mathbf{Q}_*^{(0)}$, as well as all $\mathbf{Q}_{\ell_1}, \ldots, \mathbf{Q}_{\ell_t}$ are no longer random. Furthermore, when $z \in \mathbf{Q}_{\mathbf{M}}^{(+)} \cup \mathbf{Q}_{\mathbf{M}}^{(-)} \cup \mathbf{Q}_*^{(1)} \cup$

$\mathbf{Q}_*^{(0)}$ the values of $f_{T,\mathbf{A},\mathbf{H}}(z)$ from $\mathcal{D}_{\text{yes}}$ (and from $\mathcal{D}_{\text{no}}$) are fixed to their corresponding values according to (6.1), which match their settings in $\mathcal{U}'_{\text{yes}}$ and $\mathcal{U}'_{\text{no}}$. Likewise, when $z \in \mathbf{Q}_{\ell_i}$ with $\ell_i \leq \frac{3N}{4}$, $f_{T,\mathbf{A},\mathbf{H}}(z)$ is determined by a dictator or anti-dictator in $\{m_1, m_2\}$; by the principle of deferred decisions, the values of $f_{T,\mathbf{A},\mathbf{H}}(z)$ can be simulated exactly. Therefore, it remains to consider the values of $\boldsymbol{r}_\alpha^{(i)}$ corresponding to $f_{T,\mathbf{A},\mathbf{H}}(z_\alpha^{(i)})$ for each $i \in [t]$, where $\ell_i > \frac{3N}{4}$, so for simplicity, assume that every $\ell_i > \frac{3N}{4}$.

Consider a function $v\colon [t] \to \{$"edge", "lone vertex", "empty set"$\}$ which indicates whether the response of the ith rejection sampling query sampled in Step 4(b) falls into case (i) (when $\boldsymbol{v}_i$ is an edge), or case (ii) (when $\boldsymbol{v}_i$ is a lone vertex), or case (iii) (when $\boldsymbol{v}_i$ is $\emptyset$). In other words,

$$v(i) = \begin{cases} \text{"edge"} & \boldsymbol{v}_i \in \overline{\mathbf{M}} \times \overline{\mathbf{M}} \\ \text{"lone vertex"} & \boldsymbol{v}_i \in \overline{\mathbf{M}} \\ \text{"empty set"} & \boldsymbol{v}_i = \emptyset \end{cases}$$

We thus consider one fixed function $v\colon [t] \to \{$"edge", "lone vertex", "empty set"$\}$ and condition on the fact that v specifies the three cases of Step 4(b) (in the case of $\mathcal{U}_{\text{yes}}$ and $\mathcal{U}_{\text{no}}$) and whether the edge sampled $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ in the fourth step of generating $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ for $\boldsymbol{h}_{\ell_i}$ either intersects $\mathbf{L}_i$ fully (in the case of an edge), or partially (in the case of a lone vertex), or it does not intersect at all (in the case of the empty set). Thus, again, we may consider the distributions conditioned on the edges sampled are specified correctly by v.

The following three lemmas give the distribution of $\boldsymbol{r}_1^{(i)} \sim \mathcal{V}'_{\text{yes}}$ and $\boldsymbol{r}_1^{(i)} \sim \mathcal{V}'_{\text{no}}$ in the cases when $\boldsymbol{v}_i$ is an edge, or a lone vertex, or the empty set. We note that the three lemmas indicate how to generate the bits $\boldsymbol{r}_\alpha^{(i)}$ in Step 4(b) of Alg′.

Lemma 6.2.10. *For every $i \in [t]$ with $v(i) =$ "edge", we have that every $\alpha \in [|\mathbf{Q}_{\ell_i}|]$ has $\boldsymbol{r}_\alpha^{(i)}$ generated from* Alg′ *is distributed exactly as $\boldsymbol{f}(z_\alpha^{(i)})$.*

Proof: This simply follows from the principle of deferred decisions, since Alg′ generates all the necessary randomness to simulate a query to a function $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ or $\boldsymbol{f} \sim \mathcal{D}_{\text{no}}$ which indexes to the sub-function $\boldsymbol{h}_{\ell_i}$. ■

Lemma 6.2.11. *For every $i \in [t]$ with $v(i) =$ "empty set", there exists $|\gamma_{yes}|, |\gamma_{no}| = O(\frac{\log^2 n}{n})$ such that for $\boldsymbol{r} \sim \mathcal{V}'_{yes}$ satisfies*

$$\boldsymbol{r}_1^{(i)} \oplus (z_1^{(i)})_{\boldsymbol{j}_3} \sim \begin{cases} \text{Ber}\left(p_\emptyset(\overline{\mathbf{L}}_i) + \gamma_{yes}\right) & \boldsymbol{j}_3 = m_1 \\ \text{Ber}(1 - p_\emptyset(\overline{\mathbf{L}}_i) - \gamma_{yes}) & \boldsymbol{j}_3 = m_2 \end{cases},$$

and $\boldsymbol{r} \sim \mathcal{V}'_{no}$ satisfies

$$\boldsymbol{r}_1^{(i)} \oplus (z_1^{(i)})_{\boldsymbol{j}_3} \sim \begin{cases} \text{Ber}\left(p_\emptyset(\overline{\mathbf{L}}_i) + \gamma_{no}\right) & \boldsymbol{j}_3 = m_1 \\ \text{Ber}\left(1 - p_\emptyset(\overline{\mathbf{L}}_i) - \gamma_{no}\right) & \boldsymbol{j}_3 = m_2 \end{cases}.$$

Proof: We recall that $\boldsymbol{h}_{\ell_i}$ is determined by $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ and $\boldsymbol{j}_3 \sim \{m_1, m_2\}$ in the fourth step of generating $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$. Consider the case when $\boldsymbol{j}_3 = m_1$, and the case when $(z_1^{(i)})_{m_1} = 0$ (since the case $(z_1^{(i)})_{m_1} = 1$ is symmetric, except we flip the answer).

Recall that we condition on the fact that the edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ satisfies $\mathbf{L}_i \cap \{\boldsymbol{j}_1, \boldsymbol{j}_2\} = \emptyset$, as well as the conclusions from Lemma 6.2.8, so we may write:

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{\boldsymbol{r} \sim \mathcal{V}_{\text{yes}}}\left[\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{"empty set"}\right] \\
&= \mathop{\mathbf{Pr}}_{\substack{\mathbf{G} \sim \mathcal{D}_{\text{no}} \\ (\boldsymbol{j}_1, \boldsymbol{j}_2)}}\left[\left(\boldsymbol{j}_1 \in \mathbf{A} \cap \overline{\mathbf{L}}_i^{(0)} \wedge \boldsymbol{j}_2 \in \overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(1)}\right) \vee \left(\boldsymbol{j}_1 \in \mathbf{A} \cap \overline{\mathbf{L}}_i^{(1)} \wedge \boldsymbol{j}_2 \in \overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(0)}\right) \mid v(i) = \text{"empty set"}\right], \\
&= \frac{1}{|\mathbf{A} \cap \overline{\mathbf{L}}_i| \cdot |\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i|} \cdot \left(|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(0)}| \cdot |\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(1)}| + |\mathbf{A} \cap \overline{\mathbf{L}}_i^{(1)}| \cdot |\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(0)}|\right) \qquad (6.3)
\end{aligned}$$

since the value of $\boldsymbol{f}(z_1^{(i)})$ in the case of $\boldsymbol{j}_3 = m_1$ will be a parity of the end points, so this parity will be 1 when the values of the variables $\boldsymbol{j}_1$ and $\boldsymbol{j}_2$ under $z_1^{(i)}$ disagree. In order to see this, we recall that $\mathbf{G}$ is the complete bipartite graph (in the case when $\boldsymbol{r} \sim \mathcal{V}_{\text{yes}}$) with sides $\mathbf{A}$ and $\overline{\mathbf{A}}$, so the edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \in \mathbf{A} \times \overline{\mathbf{A}}$ must have $(z_1^{(i)})_{\boldsymbol{j}_1} \neq (z_1^{(i)})_{\boldsymbol{j}_2}$, and $\boldsymbol{j}_1, \boldsymbol{j}_2 \in \overline{\mathbf{L}}_i$.

Since $v(i)$ = "empty set", we note that $|\mathbf{L}_i| \leq \frac{n}{\log n}$, so $|\overline{\mathbf{L}}_i| = \Omega(n)$. In addition, by Lemma 6.2.8, let:

$$|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(0)}| = \frac{|\overline{\mathbf{L}}_i^{(0)}|}{2} + \xi_0 \qquad \text{and} \qquad |\mathbf{A} \cap \overline{\mathbf{L}}_i^{(1)}| = \frac{|\overline{\mathbf{L}}_i^{(1)}|}{2} + \xi_1, \tag{6.4}$$

where $|\xi_0|, |\xi_1| \leq \sqrt{n}\log n$, which in turn, implies:

$$|\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(0)}| = \frac{|\overline{\mathbf{L}}_i^{(0)}|}{2} - \xi_0 \qquad \text{and} \qquad |\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(1)}| = \frac{|\overline{\mathbf{L}}_i^{(1)}|}{2} - \xi_1. \tag{6.5}$$

Therefore, combining (6.3) with (6.4) and (6.5),

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{\boldsymbol{r}\sim\mathcal{V}_{\text{yes}}}\left[\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{“empty set”}\right] \\
&= \frac{1}{\left(\frac{|\overline{\mathbf{L}}_i|}{2} + \xi_0 + \xi_1\right)\left(\frac{|\overline{\mathbf{L}}_i|}{2} - \xi_0 - \xi_1\right)}\left(\left(\frac{|\overline{\mathbf{L}}_i^{(0)}|}{2} + \xi_0\right)\left(\frac{|\overline{\mathbf{L}}_i^{(1)}|}{2} - \xi_1\right) + \left(\frac{|\overline{\mathbf{L}}_i^{(1)}|}{2} + \xi_1\right)\left(\frac{|\overline{\mathbf{L}}_i^{(0)}|}{2} - \xi_0\right)\right) \\
&= \frac{2|\overline{\mathbf{L}}_i^{(0)}| \cdot |\overline{\mathbf{L}}_i^{(1)}| - 8\xi_0\xi_1}{|\overline{\mathbf{L}}_i|^2 - 4\xi_0^2 - 4\xi_1^2 - 8\xi_0\xi_1} = \frac{2|\overline{\mathbf{L}}_i^{(0)}| \cdot |\overline{\mathbf{L}}_i^{(1)}|}{|\overline{\mathbf{L}}_i|^2} + \gamma_{\text{yes}},
\end{aligned}$$

where $|\gamma_{\text{yes}}| \leq O(\frac{\log^2 n}{n})$, since $|\overline{\mathbf{L}}_i|, |\overline{\mathbf{L}}_i^{(0)}|, |\overline{\mathbf{L}}_i^{(1)}| = \Omega(n)$.

The case when $\boldsymbol{r} \sim \mathcal{V}_{\text{no}}$ is analogous, except that now the underlying graph is the union of two cliques at $\mathbf{A}$ and $\overline{\mathbf{A}}$, so:

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{\boldsymbol{r}\sim\mathcal{V}_{\text{no}}}\left[\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{“empty set”}\right] \\
&= \mathop{\mathbf{Pr}}_{\substack{\mathbf{G}\sim\mathcal{G}_1 \\ (\boldsymbol{j}_1,\boldsymbol{j}_2)}}\left[\left(\boldsymbol{j}_1 \in \mathbf{A} \cap \overline{\mathbf{L}}_i^{(0)} \wedge \boldsymbol{j}_2 \in \mathbf{A} \cap \overline{\mathbf{L}}_i^{(1)}\right) \vee \left(\boldsymbol{j}_1 \in \overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(0)} \wedge \boldsymbol{j}_2 \in \overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(1)}\right) \mid v(i) = \text{“empty set”}\right], \\
&= \frac{1}{\binom{|\mathbf{A}\cap\overline{\mathbf{L}}_i|}{2} + \binom{|\overline{\mathbf{A}}\cap\overline{\mathbf{L}}_i|}{2}} \cdot \left(|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(0)}| \cdot |\mathbf{A} \cap \overline{\mathbf{L}}_i^{(1)}| + |\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(0)}| \cdot |\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(1)}|\right) \\
&= \frac{1}{\binom{\frac{|\overline{\mathbf{L}}_i|}{2}+\xi_0+\xi_1}{2} + \binom{\frac{|\overline{\mathbf{L}}_i|}{2}-\xi_0-\xi_1}{2}}\left(\left(\frac{|\overline{\mathbf{L}}_i^{(0)}|}{2} + \xi_0\right)\left(\frac{|\overline{\mathbf{L}}_i^{(1)}|}{2} + \xi_1\right) + \left(\frac{|\overline{\mathbf{L}}_i^{(0)}|}{2} - \xi_0\right)\left(\frac{|\overline{\mathbf{L}}_i^{(1)}|}{2} - \xi_1\right)\right) \\
&= \frac{2|\overline{\mathbf{L}}_i^{(0)}| \cdot |\overline{\mathbf{L}}_i^{(1)}|}{|\overline{\mathbf{L}}_i|^2} + \gamma_{\text{no}},
\end{aligned}$$

were again, $|\gamma_{\text{no}}| \leq O(\frac{\log^2 n}{n})$. ■

Lemma 6.2.12. *For every $i \in [t]$ with $v(i)$ = "lone vertex", let $\boldsymbol{j}_2 \in \overline{M}$ be the lone vertex observed and let $w = \neg(z_1^{(i)})_{\boldsymbol{j}_2}$. There exists $|\gamma'_{yes}|, |\gamma'_{no}| \leq O(\frac{\log n}{\sqrt{n}})$ such that for $\boldsymbol{r} \sim \mathcal{V}'_{yes}$ satisfies*

$$\boldsymbol{r}_1^{(i)} \oplus (z_1^{(i)})_{\boldsymbol{j}_3} \sim \begin{cases} \mathrm{Ber}(p_v(\overline{\mathbf{L}}_i^{(w)}) + \gamma'_{yes}) & \boldsymbol{j}_3 = m_1 \\ \mathrm{Ber}(1 - p_v(\overline{\mathbf{L}}_i^{(w)}) - \gamma'_{yes}) & \boldsymbol{j}_3 = m_2 \end{cases},$$

and $\boldsymbol{r} \sim \mathcal{V}'_{no}$ satisfies

$$\boldsymbol{r}_1^{(i)} \oplus (z_1^{(i)})_{\boldsymbol{j}_3} \sim \begin{cases} \mathrm{Ber}(p_v(\overline{\mathbf{L}}_i^{(w)}) + \gamma'_{no}) & \boldsymbol{j}_3 = m_1 \\ \mathrm{Ber}(1 - p_v(\overline{\mathbf{L}}_i^{(w)}) - \gamma'_{no}) & \boldsymbol{j}_3 = m_2 \end{cases}.$$

Proof: We follow a similar strategy to Lemma 6.2.11, where we know that we sample an edge $(\boldsymbol{j}_1, \boldsymbol{j}_2) \sim \mathbf{G}$ whose value of $\boldsymbol{j}_2 \in \mathbf{L}_i$, and $\boldsymbol{j}_1 \notin \mathbf{L}_i$. Consider for simplicity the case when $\mathbf{G}$ is a complete bipartite graph with sides $\mathbf{A}$ and $\overline{\mathbf{A}}$, and $\boldsymbol{j}_3 = m_1$ and $(z_1^{(i)})_{m_1} = 0$.

Similarly to (6.3), we have that in order for $\boldsymbol{r}_1^{(i)} = 1$, we must have $(z_1^{(i)})_{\boldsymbol{j}_1} \neq (z_1^{(i)})_{\boldsymbol{j}_2}$. Suppose that $\boldsymbol{j}_2 \in \mathbf{A}$ and $w = \neg(z_1^{(i)})_{\boldsymbol{j}_2}$, then in order for $\boldsymbol{r}_1^{(i)} = 1$, $\boldsymbol{j}_1$ must have been sampled from $\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(w)}$. Using Lemma 6.2.8, we have that there exists $|\xi_0|, |\xi_1| \leq \sqrt{n} \log n$ so:

$$\Pr_{\boldsymbol{r} \sim \mathcal{V}'_{\text{yes}}} [\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{“lone vertex”}] = \frac{|\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{A}} \cap \overline{\mathbf{L}}|} = \frac{|\overline{\mathbf{L}}_i^{(w)}|/2 - \xi_w}{|\overline{\mathbf{L}}_i|/2 - \xi_0 - \xi_1} \approx \frac{|\overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{L}}_i|} \pm O(\tfrac{\log n}{\sqrt{n}}),$$

where we used the fact that $|\mathbf{L}_i|, |\mathbf{L}_i^{(w)}| = \Omega(n)$. If $\boldsymbol{j}_2 \in \overline{\mathbf{A}}$, then

$$\Pr_{\boldsymbol{r} \sim \mathcal{V}'_{\text{yes}}} [\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{“lone vertex”}] = \frac{|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(w)}|}{|\mathbf{A} \cap \overline{\mathbf{L}}|} = \frac{|\overline{\mathbf{L}}_i^{(w)}|/2 + \xi_w}{|\overline{\mathbf{L}}_i|/2 + \xi_0 + \xi_1} \approx \frac{|\overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{L}}_i|} \pm O(\tfrac{\log n}{\sqrt{n}}).$$

In both cases, we have that $\boldsymbol{r}_1^{(i)} \sim \mathrm{Ber}(p_v(\overline{\mathbf{L}}_i^{(w)}) \pm O(\frac{\log n}{\sqrt{n}}))$, and when we have $(z_1^{(i)})_{m_1} = 1$, we simply flip the answer. Likewise, when $\boldsymbol{j}_3 = m_2$, we flip the answer once more.

In the case of $\mathbf{G}$ being the union of two cliques at $\mathbf{A}$ and $\overline{\mathbf{A}}$, when $\boldsymbol{j}_3 = m_1$ and $(z_1^{(i)})_{m_1} = 0$, we have that when $\boldsymbol{j}_2 \in \mathbf{A}$,

$$\Pr_{\boldsymbol{r} \sim \mathcal{V}'_{\text{no}}} [\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{“lone vertex”}] = \frac{|\mathbf{A} \cap \overline{\mathbf{L}}_i^{(w)}|}{|\mathbf{A} \cap \overline{\mathbf{L}}|} = \frac{|\overline{\mathbf{L}}_i^{(w)}|/2 + \xi_w}{|\overline{\mathbf{L}}_i|/2 + \xi_0 + \xi_1} \approx \frac{|\overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{L}}_i|} \pm O(\tfrac{\log n}{\sqrt{n}}),$$

and when $\boldsymbol{j}_2 \in \overline{\mathbf{A}}$,

$$\Pr_{\boldsymbol{r} \sim \mathcal{V}'_{\text{no}}} [\boldsymbol{r}_1^{(i)} = 1 \mid v(i) = \text{“lone vertex”}] = \frac{|\overline{\mathbf{A}} \cap \overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{A}} \cap \overline{\mathbf{L}}|} = \frac{|\overline{\mathbf{L}}_i^{(w)}|/2 - \xi_w}{|\overline{\mathbf{L}}_i|/2 - \xi_0 - \xi_1} \approx \frac{|\overline{\mathbf{L}}_i^{(w)}|}{|\overline{\mathbf{L}}_i|} \pm O(\tfrac{\log n}{\sqrt{n}}),$$

so we obtain the analogous conclusion. ■

We note that after defining $\boldsymbol{r}_1^{(i)}$ in the cases with $v(i) =$ “empty set”, we have that all values $\boldsymbol{r}_\alpha^{(i)}$ are determined by flipping the answer when $(z_\alpha^{(i)})_{\boldsymbol{j}_3} \neq (z_1^{(i)})_{\boldsymbol{j}_3}$. Likewise, after defining $\boldsymbol{r}_1^{(i)}$ in the cases with $v(i) =$ “lone vertex”, we have that all values $\boldsymbol{r}_\alpha^{(i)}$ are determined by flipping the answer when $(z_\alpha^{(i)})_{\boldsymbol{j}_3} \neq (z_1^{(i)})_{\boldsymbol{j}_3}$ and when $(z_\alpha^{(i)})_{\boldsymbol{j}_2} \neq (z_1^{(i)})_{\boldsymbol{j}_2}$.

Finally, consider the indices $i \in [t]$ of responses $\boldsymbol{r}_\alpha^{(i)}$ with $v(i) =$ “empty set”, and call these E. We have that for all $i \in E$, $\mathcal{U}'_{\text{yes}}$ and $\mathcal{U}'_{\text{no}}$ outputs bits which equal 1 with probability τ_i where $\tau_i = \Omega(1)$, and $\mathcal{V}'_{\text{yes}}$ and $\mathcal{V}'_{\text{no}}$ outputs bits which equal 1 with probability $\tau_i \pm O(\frac{\log^2 n}{n})$. Since these groups are independent and there at at most $q \ll n^{1.5}$ groups, we have that the bits $(\boldsymbol{r}_1^{(i)})_{i \in E} \sim \mathcal{U}'_{\text{yes}}$ (and also $\mathcal{U}'_{\text{no}}$) satisfy:

$$(\boldsymbol{r}_1^{(i)})_{i \in E} \sim \prod_{i \in E} \mathrm{Ber}(\tau_i),$$

and for each $i \in E$, there exists $\gamma_{i,\text{yes}}$ and $\gamma_{i,\text{no}}$ with $|\gamma_{i,\text{yes}}|, |\gamma_{i,\text{no}}| = O(\frac{\log^2 n}{n})$ such that $(\boldsymbol{r}_1^{(i)})_{i\in E} \sim \mathcal{V}'_{\text{yes}}$ satisfies

$$(\boldsymbol{r}_1^{(i)})_{i\in E} \sim \prod_{i\in E} \text{Ber}(\tau_i + \gamma_{i,\text{yes}}),$$

and if $(\boldsymbol{r}_1^{(i)})_{i\in E} \sim \mathcal{V}'_{\text{no}}$ satisfies

$$(\boldsymbol{r}_1^{(i)})_{i\in E} \sim \prod_{i\in E} \text{Ber}(\tau_i + \gamma_{i,\text{no}}).$$

Thus, by [Roo01], we have that the distance in total variation between these two distributions is at most $o(1)$.

Similarly, we consider the indices $i \in [t]$ with $v(i) =$ "lone vertex", and call these V. By Lemma 6.2.9, we have that $|V| \le \frac{n}{\log^4 n}$ with probability $1 - o(1)$ if the cost of the rejection sampling algorithm is less than $\frac{n^2}{\log^6 n}$. So similarly to the case with the groups in E, these can only incur at most $o(1)$ in distance in total variation.

6.3 Separating Erasure-Resilient testing from property testing

Similarly to the tolerant testing scenario, PCPPs were also used in [DRTV18] to show that there exists a property of boolean strings of length n that has a tester with query complexity independent of n, but for any constant $\alpha > 0$, every α-erasure-resilient tester is required to query $\Omega(n^c)$ many bits for some $c > 0$, thereby establishing a separation between the models. Later, in [RRV19] PCPP constructions were used to provide a separation between the erasure-resilient testing model and the tolerant testing model.

We start with some terminology. A string $x \in \{0,1,\perp\}^n$ is α-erased if x_i is equal to $\perp$ on at most αn coordinates. A string $x' \in \{0,1\}^n$ that differs from x only on coordinates $i \in [n]$ for which $x_i = \perp$ is called a *completion* of x. The (pseudo-)distance $\text{dist}(x, \mathcal{P})$ of an α-erased string x from a property $\mathcal{P}$ is the minimum, over every completion x' of x, of the relative Hamming distance of x' from $\mathcal{P}$. Note that for a string with no erasures, this is simply the Hamming distance of x from $\mathcal{P}$. As before, x is ε-far from $\mathcal{P}$ if $\text{dist}(x, \mathcal{P}) > \varepsilon$, and ε-close otherwise.

Definition 6.3.1 (Erasure-resilient tester)**.** Let $\alpha \in [0,1)$ and $\varepsilon \in (0,1)$ be parameters satisfying $\alpha + \varepsilon < 1$. An q-query α-erasure-resilient ε-tester T for $\mathcal{P}$ is a probabilistic algorithm making q queries to an α-erased string $x \in \{0,1,\perp\}^n$, that outputs a binary verdict satisfying the following two conditions.

1. If $\text{dist}(x, \mathcal{P}) = 0$ (i.e., if there exists a completion x' of x, such that $x' \in \mathcal{P}$), then T accepts x with probability at least $2/3$.
2. If $\text{dist}(x, \mathcal{P}) > \varepsilon$ (i.e., if every completion of x' of x is ε-far from $\mathcal{P}$), then T rejects x with probability at least $2/3$.

Our main result in this section is the following.

Theorem 6.3.2. *For every constant $\ell \in \mathbb{N}$, there exist a property $\mathcal{Q}^{(\ell)}$ and $\varepsilon_1 = \varepsilon_1(\ell) \in (0,1)$ such that the following hold.*

1. *For every $\varepsilon \in (0,1)$, the property $\mathcal{Q}^{(\ell)}$ can be ε-tested using a number of queries depending only on ε (and ℓ).*

2. *For every $\varepsilon \in (0, \varepsilon_1)$ and any $\alpha = \Omega(1/\log^{(\ell)} N)$ satisfying $\varepsilon + \alpha < 1$, any α-erasure resilient ε-tester for $\mathcal{Q}^{(\ell)}$ needs to make $\Omega(N/10^{\ell} \cdot \mathrm{polylog}^{(\ell)} N)$ many queries on inputs of length N.*

Proof of Theorem 6.3.2: The proof of Theorem 6.3.2 is almost identical to the proof of Theorem 5.5.1. The only difference is that we replace Lemma 5.5.4 with a counterpart for erasure resilient testing, where instead of setting the last $z^{(\ell)}_{\mathbb{F},0}$ bits of x to $(0)^{z^{(\ell)}_{\mathbb{F},0}}$, we use $(\perp)^{z^{(\ell)}_{\mathbb{F},0}}$, noting that the relative size of this part of the input is $1/(s+1) = \Theta(1/\log^{(\ell)}(N))$. ■

Bibliography

[AB10] Noga Alon and Eric Blais. Testing Boolean function isomorphism. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 6302 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2010.

[ABC$^+$13] Noga Alon, Eric Blais, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. *SIAM Journal on Computing*, 42(2):459–493, 2013.

[ABEF17] Noga Alon, Omri Ben-Eliezer, and Eldar Fischer. Testing hereditary properties of ordered graphs and matrices. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 848–858, 2017.

[ACCL07] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007.

[AFKS00] Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000.

[AFNS09] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009.

[AKK$^+$05] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing reed-muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005.

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

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[AS08] Noga Alon and Asaf Shapira. Every monotone graph property is testable. *SIAM Journal on Computing*, 38(2):505–522, 2008.

[Bac13] Francis R. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013.

[Bar75] Zsolt Baranyai. On the factorization of the complete uniform hypergraph. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday)*, volume 1, pages 91–108, 1975.

[BB16] Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC '16)*, pages 1021–1032, 2016.

[BCE$^+$19] Eric Blais, Clément L Canonne, Talya Eden, Amit Levi, and Dana Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. *ACM Transactions on Computation Theory*, 11(4):1–33, 2019.

[BCGSM12] Jop Briët, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

[BCL$^+$06] Christian Borgs, Jennifer Chayes, László Lovász, Vera T Sós, Balázs Szegedy, and Katalin Vesztergombi. Graph limits and parameter testing. In *Proceedings of the 38th ACM Symposium on the Theory of Computing (STOC)*, pages 261–270, 2006.

[BCP$^+$17] Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and C. Seshadhri. Optimal unateness testers for real-values functions: Adaptivity helps. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, 2017.

[BEFLY18] Omri Ben-Eliezer, Eldar Fischer, Amit Levi, and Yuichi Yoshida. Ordered graph limits and their applications. *arXiv preprint arXiv:1811.02023*, 2018.

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

[BFLR20] Omri Ben-Eliezer, Eldar Fischer, Amit Levi, and Ron D. Rothblum. Hard properties with (very) short pcpps and their applications. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 9:1–9:27, 2020.

[BFLS91] László Babai, Lance Fortnow, Leonid Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC*, pages 21–31, 1991.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust pcps of proximity, shorter pcps, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[BKK$^+$16] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant rate PCPs for circuit-SAT with sublinear query complexity. *Journal of the ACM*, 63(4):32:1–32:57, 2016.

[BL97] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[Bla08] Eric Blais. Improved bounds for testing juntas. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 317–330. Springer, 2008.

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the 41st ACM Symposium on the Theory of Computing (STOC '09)*, pages 151–158, 2009.

[Bla12] Eric Blais. *Testing properties of Boolean functions.* PhD thesis, CMU, 2012.

[BLNR15] Alexandre Belloni, Tengyuan Liang, Hariharan Narayanan, and Alexander Rakhlin. Escaping the local minima via simulated annealing: Optimization of approximately convex functions. In *Proceedings of the 28th Annual Conference on Learning Theory (COLT '15)*, pages 240–265, 2015.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the 22nd ACM Symposium on the Theory of Computing (STOC '90)*, pages 73–83. ACM, 1990.

[Blu94] Avrim Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In *AAAI Fall Symposium on 'Relevance*, volume 5, page 1, 1994.

[BMR16] Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant testers of image properties. pages 90:1–90:14, 2016.

[BR02] Omer Barkol and Yuval Rabani. Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. *Journal of Computer and System Sciences*, 64(4):873–896, 2002.

[Bra15] Madeline V. Brandt. Intersecting hypergraphs and decompositions of complete uniform hypergraphs. B.A. Thesis, Reed College, 2015.

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008.

[BSS10] Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. *Advances in mathematics*, 223(6):2200–2218, 2010.

[CC16] Deeparnab Chakrabarty and Seshadhri Comandur. An o(n) monotonicity tester for boolean functions over the hypercube. *SIAM Journal on Computing*, 45(2):461–472, 2016.

[CFGM12] Sourav Chakraborty, Eldar Fischer, David García-Soriano, and Arie Matsliah. Junto-symmetric functions, hypergraph isomorphism and crunching. In *Proceedings of the 27th Conference on Computational Complexity (CCC '12)*, pages 148–158. IEEE, 2012.

[CG04] Hana Chockler and Dan Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, pages 301–305, 2004.

[CG17] Clément L. Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *arXiv preprint arXiv:1702.05678*, 2017.

[CGM11] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP '11)*, volume 6755 of *Lecture Notes in Computer Science*, pages 545–556. Springer, 2011.

[CGR13] Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 411–424. Springer, 2013.

[CGR+14] Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 45(2):139–184, 2014.

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

[CS13] Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Proceedings of the 45th ACM Symposium on the Theory of Computing (STOC '13)*, pages 419–428, 2013.

[CS14] Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10(17):453–464, 2014.

[CSS09] Artur Czumaj, Asaf Shapira, and Christian Sohler. Testing hereditary properties of nonexpanding bounded-degree graphs. *SIAM Journal on Computing*, 38(6):2499–2510, 2009.

[CST+17] Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the query complexity of non-adaptive junta testing. In *Proceedings of the 32nd Conference on Computational Complexity (CCC '17)*, 2017.

[CSTW] Xi Chen, Rocco A. Servedio, Li-Yan Tan, and Erik Waingarten. Adaptivity is exponential powerful for testing monotonicity of halfspaces. Available at: `http://www.cs.columbia.edu/~eaw/CSTW.pdf`.

[CWX17a] Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC '17)*, 2017.

[CWX17b] Xi Chen, Erik Waingarten, and Jinyu Xie. Boolean unateness testing with $\widetilde{O}(n^{3/4})$ adaptive queries. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2017.

[DGL+99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonocity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, 1999.

[Din07] Irit Dinur. The pcp theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[DLM+07] Ilias Diakonikolas, Homin K Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A Servedio, and Andrew Wan. Testing for concise representations. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pages 549–558. IEEE, 2007.

[DMN19] Anindya De, Elchanan Mossel, and Joe Neeman. Junta correlation is testable. 2019. To appear.

[Doe11] Benjamin Doerr. Analyzing randomized search heuristics: Tools from probability theory. *Theory of randomized search heuristics*, 1:1–20, 2011.

[DR06] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

[DRTV18] Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin M. Varma. Erasure-resilient property testing. *SIAM Journal on Computing*, 47(2):295–329, 2018.

[EKK$^+$00] Funda Ergün, Sampath Kannan, S. Ravi Kumar, Ronitt Rubinfeld, and Mahesh Vishwanthan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000.

[FF06] Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory of Computing*, 2(9):173–183, 2006.

[FGL$^+$91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete. In *Thirty-Second Annual Symposium of Foundations of Computer Science (FOCS*, pages 2–12. IEEE, 1991.

[FKR$^+$04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

[FLN$^+$02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '02)*, pages 474–483, 2002.

[FN07] Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37(2):482–501, 2007.

[FNS04] Eldar Fischer, Ilan Newman, and Jiří Sgall. Functions that have read-twice constant width branching programs are not necessarily testable. *Random Structures & Algorithms*, 24(2):175–193, 2004.

[FR10] Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms*, 6(3):52, 2010.

[Fri98] Ehud Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–35, 1998.

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Proceedings Third Israel Symposium on the Theory of Computing and Systems*, pages 190–198. IEEE, 1995.

[GGLR98] Oded Goldreich, Shafi Goldwasser, Eric Lehman, and Dana Ron. Testing monotonicity. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS '98)*, pages 426–435, 1998.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[GLR$^+$91] Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Thirty-Second Annual ACM Symposium on the Theory of Computing (STOC)*, volume 91, pages 32–42, 1991.

[GLS12] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[GM07] Oded Goldreich and Or Meir. A small gap in the gap amplification of assignment testers, 2007. In ECCC, 2007, TR05-46, Comment 3.

[Gol08] Oded Goldreich. *Computational complexity - A conceptual perspective*. Cambridge University Press, 2008.

[Gol17] Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.

[GOS$^+$11] Parikshit Gopalan, Ryan O'Donnell, Rocco A Servedio, Amir Shpilka, and Karl Wimmer. Testing fourier dimensionality and sparsity. *SIAM Journal on Computing*, 40(4):1075–1100, 2011.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[GR05] Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 306–317. Springer, 2005.

[GR11] Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer, 2011.

[HKNO09] Avinatan Hassidim, Jonathan A Kelner, Huy N Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '09)*, pages 22–31. IEEE, 2009.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[Hor72] Ellis Horowitz. A fast method for interpolation using preconditioning. *Information Processing Letters*, 1(4):157–163, 1972.

[JL10] Z. Q. John Lu. The elements of statistical learning: data mining, inference, and prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(3):693–694, 2010.

[JPRZ09] Charanjit S Jutla, Anindya C Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. *Random Structures & Algorithms*, 35(2):163–193, 2009.

[KNOW14] Pravesh Kothari, Amir Nayyeri, Ryan O'Donnell, and Chenggang Wu. Testing surface area. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 1204–1214. Society for Industrial and Applied Mathematics, 2014.

[KNR02] Yoshiharu Kohayakawa, Brendan Nagle, and Vojtěch Rödl. Efficient testing of hypergraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 1017–1028. Springer, 2002.

[KR98] Michael Kearns and Dana Ron. Testing problems with sub-learning sample complexity. In *Proceedings of the 11th Annual Conference on Learning Theory (COLT '98)*, pages 268–279. ACM, 1998.

[KR06] Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM Journal on Computing*, 36(3):779–802, 2006.

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

[KS09] Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 601–614. Springer, 2009.

[KSS94] Michael J Kearns, Robert E Schapire, and Linda M Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[KSS18] Akash Kumar, C Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A n^ 1/2+ o (1)-query one-sided tester for minor closed properties on bounded degree graphs. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 509–520. IEEE, 2018.

[Lev15] Amit Levi. On symmetric structures in graphs and applications in property testing. *M.Sc thesis, The Zandman-Slaner Graduate School of Engineering, Tel Aviv University*, 2015.

[LR15] Reut Levi and Dana Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Transactions on Algorithms*, 11(3):24, 2015.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15)*, pages 1049–1065. IEEE, 2015.

[LW19] Amit Levi and Erik Waingarten. Lower bounds for tolerant junta and unateness testing via rejection sampling of graphs. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 52:1–52:20, 2019.

[MORS10] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A Servedio. Testing halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010.

[MOS03] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning juntas. In *Proceedings of the 35th ACM Symposium on the Theory of Computing (STOC '03)*, pages 206–212. ACM, 2003.

[MR09] Sharon Marko and Dana Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Transactions on Algorithms*, 5(2):22, 2009.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

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

[O'D14] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[PR02] Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Structures & Algorithms*, 20(2):165–183, 2002.

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

[PRW19] Ramesh Krishnan S Pallavooh, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of boolean functions. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA '20)*, pages 1995–2009. SIAM, 2019.

[Roo01] Bero Roos. Binomial approximation to the poisson binomial distribution: The krawtchouk expansion. *Theory of Probability & Its Applications*, 45(2):258–272, 2001.

[RRV19] Sofya Raskhodnikova, Noga Ron-Zewi, and Nithin M. Varma. Erasures vs. errors in local decoding and property testing. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 63:1–63:21, 2019.

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

[Sağ18] Mert Sağlam. Near log-convexity of measured heat in (discrete) time and consequences. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 967–978. IEEE, 2018.

[Sch02] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

[SF11] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Spi96] Daniel A Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[STW15] Rocco A Servedio, Li-Yang Tan, and John Wright. Adaptivity helps for testing juntas. In *Proceedings of the 30th Conference on Computational Complexity (CCC '15)*, pages 264–279, 2015.

[Val15] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *Journal of the ACM*, 62(2):13, 2015.

Appendix A

A.1 A Useful Claim

Consider any set of trees $C_1, \dots, C_\alpha \subset [n]$ with roots $u_1, \dots, u_\alpha$ satisfying the following conditions:

- Each $|C_i| \leq \log n$ for $i \in [\alpha]$,
- We have $\sum_{i=1}^{\alpha} |C_i| \leq \frac{n}{\log^4 n}$.

Recall that $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ is the event that the components $C_1, \dots, C_\alpha$ is consistent with the partition $\mathbf{A} \subset [n]$. More formally, for each $i \in [\alpha]$, we consider layering the tree C_i with root u_i. We let $|C_i(\text{odd})|$ be the odd layers and $|C_i(\text{even})|$ be the even layers. Then, we have event $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ is satisfied if for each $i \in [\alpha]$, either $C_i(\text{odd}) \subset \mathbf{A}$ and $C_i(\text{even}) \subset \overline{\mathbf{A}}$ or $C_i(\text{even}) \subset \mathbf{A}$ and $C_i(\text{odd}) \subset \overline{\mathbf{A}}$.

The following lemma is the last necessary step of Claim 4.3.22.

Lemma A.1.1. *Then, for any two indices j, k, which do not lie in the same component, we have:*

$$\Pr_{\mathbf{A}}[k \in \mathbf{A} \mid j \in \mathbf{A}, \boldsymbol{\mathcal{E}}_{C,no}] \geq \frac{1}{2} - \frac{\log^4 n}{n}.$$

Proof: The proof is very straight-forward, we simply count the number of possible partitions $\mathbf{A}$ for which $j \in \mathbf{A}$ and are consistent with $C_1, \dots, C_\alpha$ and divide by the total number of such partitions. For simplicity, assume that j lies in $C_1(\text{odd})$ and k lies in $C_2(\text{odd})$; the other cases, when $j \in C_1(\text{even})$ or $k \in C_2(\text{even})$ follow from very similar arguments.

We let X be the number of partitions $A \subset [n]$ of size $\frac{n}{2}$ which trigger event $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ and have $C_1(\text{odd}) \subset A$ and $C_2(\text{odd}) \subset A$. In order to count these, we first choose which roots $u_3, \dots, u_\alpha$ will be included in A, and then we pick from the remaining vertices to include in A. For a subset $S \subset \{3, \dots, \alpha\}$, we define the quantities:

- $Q = \sum_{i=3}^{\alpha} |C_i|$ is the total vertices assigned from components.
- $S_A = \sum_{i \in S} |C_i(\text{odd})| + \sum_{i \in [\alpha] \setminus S} |C_i(\text{even})|$ is the total vertices assigned from components to A if we included the roots of components in S in A.
- $S_{\overline{A}} = Q - S_A$.

Note that for all subsets $S \subset \{3, \dots, \alpha\}$, we have $S_A \leq \frac{n}{\log^4 n}$.

Then we have:

$$X = \sum_{\ell=0}^{\alpha-2} \sum_{\substack{S \subset [3;\alpha] \\ |S|=\ell}} \binom{n - Q - |C_1| - |C_2|}{\frac{n}{2} - S_A - |C_1(\text{odd})| - |C_2(\text{odd})|}.$$

Let Y be the number of partitions $A \subset [n]$ of size $\frac{n}{2}$ which trigger event $\boldsymbol{\mathcal{E}}_{C,\text{no}}$ and have $C_1(\text{odd}) \subset A$ and $C_2(\text{even}) \subset A$. Similarly, we have:

$$Y = \sum_{\ell=0}^{\alpha-2} \sum_{\substack{S \subset [3;\alpha] \\ |S|=\ell}} \binom{n - Q - |C_1| - |C_2|}{\frac{n}{2} - S_A - |C_1(\text{odd})| - |C_2(\text{even})|}.$$

For a particular fixed $S \subset [3;\alpha]$ of size ℓ, we consider the ratio of the summand in X and in Y:

$$\begin{aligned}
\frac{\binom{n - Q - |C_1| - |C_2|}{\frac{n}{2} - S_A - |C_1(\text{odd})| - |C_2(\text{odd})|}}{\binom{n - Q - |C_1| - |C_2|}{\frac{n}{2} - S_A - |C_1(\text{odd})| - |C_2(\text{even})|}} &= \frac{\left(\frac{n}{2} - S_A - |C_1(\text{odd})| - |C_2(\text{even})|\right)!}{\left(\frac{n}{2} - S_A - |C_1(\text{odd})| - |C_2(\text{odd})|\right)!} \\
&\quad \times \frac{\left(\frac{n}{2} - S_{\overline{A}} - |C_1(\text{even})| - |C_2(\text{odd})|\right)!}{\left(\frac{n}{2} - S_{\overline{A}} - |C_1(\text{even})| - |C_2(\text{even})|\right)!} \\
&= \left(1 \pm O\left(\frac{\log n}{n}\right)\right)^{\log n} \left(1 \pm O\left(\frac{\log n}{n}\right)\right)^{\log n} \\
&= 1 \pm O\left(\frac{\log^2 n}{n}\right),
\end{aligned}$$

where we used the fact that $|C_2(\text{even})|, |C_2(\text{odd})| \le \log n$, and $\frac{n}{2} - S_A - |C_1(\text{odd})| = \Omega(n)$ and $\frac{n}{2} - S_{\overline{A}} - |C_1(\text{odd})| = \Omega(n)$. Thus, we have:

$$\frac{X}{Y} = 1 \pm O\left(\frac{\log^2 n}{n}\right),$$

and since:

$$\Pr_{\mathbf{A}}[k \in \mathbf{A} \mid j \in \mathbf{A}, \boldsymbol{\mathcal{E}}_{C,\text{no}}] = \frac{X}{X+Y},$$

we get the desired claim. ■

A.2 Reducing to the case $k = cn$ for constant $c < 1$

Claim A.2.1. *For $\varepsilon < \frac{1}{2}$, let $f\colon \{0,1\}^n \to \{0,1\}$ have* $\text{dist}(f, \mathcal{J}_k) = \varepsilon < \frac{1}{2}$. *Then, $g\colon \{0,1\}^n \times \{0,1\} \to \{0,1\}$ given by $g(x,y) = f(x) \oplus y$ has* $\text{dist}(g, \mathcal{J}_{k+1}) = \varepsilon$.

Proof: For the upper bound, suppose $h\colon \{0,1\}^n \to \{0,1\}$ had $\text{dist}(f,h) = \varepsilon$. Then, we have that $h'\colon \{0,1\}^n \times \{0,1\} \to \{0,1\}$ given by $h'(x,y) = h(x) \oplus y$ has $\text{dist}(h', g) = \varepsilon$. Thus, we have $\text{dist}(g, \mathcal{J}_{k+1}) \le \text{dist}(f, \mathcal{J}_k)$.

For the lower bound, suppose for the sake of contradiction that $h'\colon \{0,1\}^n \times \{0,1\} \to \{0,1\}$ is a $(k+1)$-junta with $\text{dist}(g,h') = \text{dist}(g, \mathcal{J}_{k+1}) < \text{dist}(f, \mathcal{J}_k)$. We note that since $\varepsilon < \frac{1}{2}$, the last variable must be influential in h'. Then, consider the functions $h_0, h_1\colon \{0,1\}^n \to \{0,1\}$ given by $h_0(x) = h'(x,0)$ and $h_1(x) = h(x,1)$. Since y is influential in h', h_0 and h_1 are both k-juntas, and therefore

$$\text{dist}(h', g) = \frac{\text{dist}(h_0, f) + \text{dist}(h_1, \neg f)}{2} \ge \text{dist}(f, \mathcal{J}_k),$$

which is a contradiction. ■

Claim A.2.2. *Let $f\colon \{0,1\}^n \to \{0,1\}$ have $\mathrm{dist}(f, \mathcal{J}_k) = \varepsilon$. Then $g\colon \{0,1\}^n \times \{0,1\} \to \{0,1\}$ given by $g(x,y) = f(x)$ has $\mathrm{dist}(g, \mathcal{J}_k) = \varepsilon$.*

Proof: For the upper bound, we have that if $h\colon \{0,1\}^n \to \{0,1\}$ has $\mathrm{dist}(f,h) = \varepsilon$, then if $h'\colon \{0,1\}^n \times \{0,1\} \to \{0,1\}$ is given by $h(x,y) = h(x)$, then $\mathrm{dist}(h', g) = \varepsilon$. Thus, we have $\mathrm{dist}(g, \mathcal{J}_{k+1}) \leq \mathrm{dist}(f, \mathcal{J}_k)$.

For the lower bound, suppose for the sake of contradiction that $h'\colon \{0,1\}^n \times \{0,1\} \to \{0,1\}$ is a k-junta with $\mathrm{dist}(g,h') = \mathrm{dist}(g,\mathcal{J}_k) < \mathrm{dist}(f, \mathcal{J}_k)$. Then, similarly to above, the functions $h_0, h_1\colon \{0,1\}^n \to \{0,1\}$ given by $h_0(x) = h'(x,0)$ and $h_1(x) = h'(x,1)$ are k-juntas with

$$\mathrm{dist}(g, \mathcal{J}_k) = \mathrm{dist}(g, h') = \frac{\mathrm{dist}(f, h_0) + \mathrm{dist}(f, h_1)}{2} \geq \varepsilon,$$

which is a contradiction. ■

Lemma A.2.3. *Fix a constant $c < 1$. For $0 < \varepsilon_0 < \varepsilon_1 < \frac{1}{2}$, let B be a $(\varepsilon_0, \varepsilon_1)$-tolerant k-junta tester for $n(k)$ variable functions making $q(k)$ queries, where $k \leq \alpha n(k)$. Then, there exists a $(\varepsilon_0, \varepsilon_1)$-tolerant cn-junta tester making $q(O(n))$ queries.*

Proof: We give an algorithm which on input $f\colon \{0,1\}^n \to \{0,1\}$, determines whether f is ε_0-close from being a cn-junta or is ε_1-far from being a cn-junta. The algorithm works as follows: on input $f\colon \{0,1\}^n \to \{0,1\}$, we let $g\colon \{0,1\}^n \times \{0,1\}^{n'} \to \{0,1\}$ be given by:

$$g(x,y) = f(x) \oplus \bigoplus_{j=1}^{n'} y_j,$$

where $n' = \max\{\frac{(c-1+\alpha)n}{1-\alpha}, 0\}$. Note that if we let $m = n + n'$ (the number of variables in g), by Claim A.2.1, if f is ε_0-close from being a cn-junta, then g is ε_0-close to being an αm-junta, and if f is ε_1-far from being a cn-junta, then g is ε_1-far from being an αm-junta. Finally, we run the tester B with $k = \alpha m$ on f, where we add $m - n(k)$ dummy variables.

The query complexity is given by $q(O(n))$, since $k = O(n)$ when $\alpha < 1$ is a constant. ■