

TEL AVIV UNIVERSITY
The Raymond and Beverly Sackler Faculty of Exact Sciences
School of Computer Science

Property Testing of Graphs and Codes

Thesis submitted for the degree of “Doctor of Philosophy”

by

Tali Kaufman

Under the supervision of Professors Noga Alon and Michael Krivelevich

Submitted to the senate of Tel Aviv University
August 2005

Abstract

Property Testing is nowadays one of the central fields in Theoretical Computer Science. This concept played a major role in the proof of the celebrated PCP theorem. A typical task in Property Testing is the following. Given an object and a property, design an algorithm that decides whether the object possesses the property or it is *far* from having it. *Far* means that the object should be modified in a non-negligible manner so as to obtain the property. The algorithm should look only at a small (sub-linear or even constant) part of the given object, and its decision should be correct with high probability. The aim is to characterize properties, for which we can rule out quickly objects that are far from having the property. In this work we focus on the testability of two fundamental families of objects, Graphs and Codes.

Graph Testing: A great body of works deals with properties of graphs that can be efficiently tested. However, prior to the work described in this thesis, all known results apply to very dense graphs or very sparse graphs. Some properties are dramatically harder to test for sparse graphs while others are harder in the case of dense graphs. Our efforts were focused on understanding testability of *general* graphs (with general density). We describe our results, and discuss the phenomena that for some properties we obtain a smooth transition between the sparse and the dense case, while in others a threshold-like behavior is exhibited.

Code Testing: A code is a set of strings with typically large pairwise (Hamming) distance. Testing of a code is the ability to perform few queries into an input string and decide if the string at hand belongs to the code or it is far from every string in the code. The study of property testing was originally focused on testability of codes. Our results show that some large families of codes that were not known to be efficiently testable are indeed such. We describe our approaches for code testing and some connections to and implications for Coding Theory.

Acknowledgments

One of the greatest opportunities while working on this thesis, was the interaction with so many amazing people. The fact that you are all tremendously smart is maybe predictable, but getting to know you as persons made such a great difference and it is such a wonderful experience for me.

Noga, thank you so much for always being there for me. The interactions we had were so precious and meaningful for me. I very much hope that there are still more of them to come.

Michael, I can not start to tell how lucky I find myself, having you as my advisor. Having your door always open for me, having your smile always welcoming me, getting your smart advices, getting to see you thinking, the ability to learn a few things (only a few ...) from you, were all so wonderful. Thank you.

Dana, though not my official advisor, your impact on me can not be overestimated. It was such a great experience to get to know you so well, both professionally and personally. Dana, nothing would be the same without having you there. I know that now, and knew that before. I could never say enough of thank you for all that you have done for me.

Simon, always full of enthusiasm and belief that we are just next to achieving our goal. Always welcomes any phone call from me, or willing to meet me at every strange hour. Thank you for being what you are for me, and for the opportunity to learn so many things from you.

Oded, I can never forget the first time I have seen you in my thesis test. Well, so many things have changed since then. I remember seeing you and can't believe that it is you sitting there. Since then, I was lucky to have so many talks with you that are so influential on me. Thanks for your care and for your endless will to show me the right way. I so much appreciate it, and so much hope that it never ends.

Nir, having you with me, besides me, loving me, care for me, all those years, from before I even knew how a university looks like, is so wonderful. I have the *great* opportunity to have you supporting me (even teaching me in part...) since my first steps in the university. Nothing would be the same without you next to me. Nothing.

Ima and Aba, Though you haven't completed your PhD yet (to say the least), your influence on me can not be expressed by words. All I know and everything I am is yours. Thank you for your great love and endless care.

Tali Kaufman
August 2005

Contents

I	Introduction and Definitions	1
1	Introduction	2
1.1	Property Testing	2
1.2	Property Testing of Graphs.	3
1.2.1	Models for Testing Graph Properties	3
1.2.2	The Gap Phenomena in Graph Property Testing	3
1.3	Our Results for Graph Testing	4
1.3.1	A Model for Testing General Graphs.	4
1.3.2	Testing Bipartiteness	5
1.3.3	Testing k -Colorability	6
1.3.4	Testing H -Freeness	6
1.4	Property Testing of Codes	7
1.5	Our Results for Code Testing	8
1.5.1	Testing Low Degree Polynomials over General Fields	8
1.5.2	Testing Almost Orthogonal Linear Codes	10
1.6	Organization	11
1.7	Bibliographic Notes	12
2	Preliminaries	13
2.1	Graph Definitions	13
2.2	Polynomial Definitions	14
2.3	Code Definitions	16
II	Testing of Graphs	18
3	Testing Bipartiteness in General Graphs	19
3.1	Introduction	19
3.1.1	Our Results	19
3.1.2	Our Techniques	21
3.2	The Algorithm for the Almost-Regular Case	21
3.2.1	Gaining Intuition: The Rapidly-Mixing Case	24
3.2.2	The Markov Chain $M_{\ell_1}^{\ell_2}(H)$	25
3.2.3	Useful Vertices and Small Cuts	25
3.2.4	Sufficient Conditions for Good Partitions	26
3.2.5	Sufficient Conditions for Detecting Odd Cycles	27
3.2.6	Proof of Theorem 3.2.1	29

3.3	The Algorithm for the General Case	30
3.3.1	Defining G' and Proving the First Two Items in Theorem 3.3.1	32
3.3.2	Establishing Item 3 in Theorem 3.3.1	36
3.3.3	Establishing Item 4 in Theorem 3.3.1	38
3.4	A Lower Bound	40
4	Testing k-Colorability in General Graphs	42
4.1	Introduction	42
4.2	One-Sided Lower Bound	43
4.3	A Two-Sided Lower Bound	45
4.4	An Upper Bound for Almost Regular Graphs	46
4.5	An Upper Bound for General Graphs	49
5	Testing Subgraph-Freeness in General Graphs	54
5.1	Introduction	54
5.1.1	A lower bound and a sharp threshold	55
5.1.2	Upper bounds	55
5.1.3	Our techniques	56
5.2	A lower bound of $\Omega(\sqrt{n/d})$	57
5.3	A lower bound of $\Omega(\min\{d, n/d\})$	57
5.3.1	Definition of the lower-bound distribution	57
5.3.2	The lower bound	61
5.4	An improved lower bound for high degrees	63
5.4.1	A variant of Behrend graphs	63
5.4.2	The edge density of large sets in random Behrend graphs	64
5.4.3	The lower bound distribution $BG(n, d)$	66
5.4.4	Online generation of graphs according to $BG(n, d)$	66
5.4.5	Proof of Lemma 5.4.1	68
5.5	A Reduction from One-Sided Error Lower Bounds to Two-Sided Error Lower Bounds	69
5.6	Upper bounds	74
5.6.1	An upper bound of $O(\sqrt{nd}/\epsilon^{3/2})$ for general graphs	74
5.6.2	An improved upper bound for relatively dense general graphs	74
6	Sampling Edges Almost Uniformly from a Graph	76
6.1	Introduction	76
6.2	The Sampling Algorithm	76
7	Bounds on the Edge Density of Dense Random Cayley Graphs	80
7.1	Introduction	80
7.2	The Bounds	80
III	Testing of Codes	83
8	Testing Polynomials over General Fields	84
8.1	Introduction	84
8.2	The Characterization	86
8.3	The Test	92

8.4	A Lower Bound	100
9	A Characterization of Low-Weight Words that Span Generalized Reed Muller Codes	102
9.1	Introduction	102
9.2	The Characterization	102
9.2.1	Proof of the Theorem	103
10	Almost Orthogonal Linear Codes are Locally Testable	104
10.1	Introduction	104
10.1.1	Our Results for General Codes	105
10.1.2	Our Results for Dual-BCH Codes and the BLR Test	105
10.1.3	Our Techniques	106
10.2	The Approach Applied in This Work	106
10.3	A Tester for the Dual-BCH Code - First Try	108
10.3.1	Back to the BLR Test	111
10.4	Improved Tester for Dual-BCH Code	112
10.4.1	Useful Lemmas for the Improved Tester	113
10.4.2	The Improved Tester	115
10.4.3	Proof of Case 1 in Theorem 10.4.1	116
10.4.4	Proof of Case 2 in Theorem 10.4.1	118
10.4.5	Local Testability of Almost Orthogonal Codes	120
10.4.6	The $C_{BCH(t)}$ is Spanned by its Codewords of Weight at most $2t + 2$	120
10.5	Efficient Construction of a Random Short Codeword of $C_{BCH(t)}$	120
IV	Open Problems	123
11	Further Research	124
A	A formal definition of $M_{\ell_1}^{\ell_2}(H)$	130

Part I

Introduction and Definitions

Chapter 1

Introduction

1.1 Property Testing.

Property testing is the study of the following class of problems: Given the ability to perform *local* queries concerning a particular object (e.g. a function or a graph), the task is to determine whether the object has a predefined *global* property (e.g. linearity or bipartiteness), or is far from having the property. The task should be performed by inspecting only a small (possibly constant) part of the whole object, where a small probability of failure is allowed.

If the testing algorithm is required always to accept objects that possess the property then it is a *one-sided tester*, otherwise it is a *two-sided tester*. A property testing algorithm is *non-adaptive* if it decides about all its queries in advance, before getting any answers. Namely a query may not depend on answers to previous queries. If a query may depend on previous answers then the tester is noted to be *adaptive*.

The study of property testing was initiated by Rubinfeld and Sudan [66] who considered testing of functions for some algebraic properties. The systematic study of testing combinatorial properties and in particular graph properties was initiated by the celebrated paper of Goldreich, Goldwasser and Ron [37].

The *complexity* of a testing algorithm is measured by the number of queries it performs. The study of property testing seeks *sub-linear algorithms*. There are problems, such as k -colorability, whose decision version is NP-hard but they can be tested, in dense graphs, using a constant number of queries. Understanding which computational problems can be solved with sub-linear complexity is an important question in the theory of computing. Moreover, studying the testability of objects may shed new light on them. For example, in [51] the study of the testability of Generalized Reed Muller codes led to a new characterization of short words that span these codes. Property testing is strongly related to other fields in computer science such as Probabilistically Checkable Proofs (PCP), learning theory and approximation algorithms. Questions concerning testing are relevant to many fields such as graph theory, coding theory, formal languages and many more.

The concept of testing has also *practical implications*. Specifically, polynomial computation can become non-feasible when massive data sets are considered. Hence, the possibility of applying sub-linear algorithms with some error is appropriate in these cases.

This work focuses on the *gap phenomenon in property testing*. Specifically, it turns out that the complexity of testing objects is highly influenced by their underlying structure. For example, testing k -colorability of dense graphs can be done using a constant number of queries, while there exists no sub-linear algorithm for testing k -colorability of sparse graphs. In this work we try to understand how the testability of objects is affected by their underlying structure. We deal mainly with *property testing in graphs and codes*.

1.2 Property Testing of Graphs.

A great body of works, initiated by the seminal paper of Goldreich, Goldwasser and Ron [37], deals with properties of graphs that can be efficiently tested. A partial list of these works includes [37, 6, 3, 2, 10, 42, 26, 39, 38, 25]. However, prior to the work described in this thesis, all known results apply to very dense graphs or to very sparse graphs. Some properties are dramatically harder to test for sparse graphs, while others are harder in the case of dense graphs. Our efforts were focused on understanding testability of *general* graphs with arbitrary density. Specifically, we are interested in understanding the testing complexity as a function of the graph density.

There are two main models for graph testing that were considered by various researchers. The first model introduced in [37], fits dense graphs, while the second model introduced in [39] fits sparse bounded degree graphs. A third model, suggested in [64], is appropriate for testing sparse graphs whose degree is not bounded. We next describe the main models and show three fundamental graph properties that exhibit a gap between the dense and the sparse cases. We then introduce a new model for testing general graphs and study these properties in this new model. We observe a phenomena that for some properties we obtain a smooth transition between the sparse and the dense case, while in others a threshold-like behavior is exhibited.

In the following subsections, we consider graphs over n vertices and $m = nd/2$ edges, where d denotes the density of the graph, i.e. its average degree.

1.2.1 Models for Testing Graph Properties

The first model, introduced in [37], is the *adjacency-matrix* model. In this model the algorithm may perform queries of the form: “Is there an edge between vertices u and v in the graph?” That is, the algorithm may probe the adjacency matrix representing the graph. We refer to such queries as *vertex-pair* queries. The notion of distance is also linked to this representation: a graph is said to be ϵ -far from having property P if more than ϵn^2 edge modifications should be performed on the graph so that it obtains the property. In other words, ϵ measures the fraction of entries in the adjacency matrix of the graph that should be modified. This model is most suitable for *dense* graphs in which the number of edges is $\Theta(n^2)$. This model was studied in [37, 6, 3, 2, 10, 42, 26].

The second model, introduced in [39], is the (*bounded-degree*) *incidence-lists* model. In this model, the algorithm may perform queries of the form: “Who is the i 'th neighbor of 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 . We refer to these queries as *neighbor* queries. Here too the notion of distance is linked to the representation: A graph is said to be ϵ -far from having property 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 (among all dn entries), that should be modified. This model is most suitable for graphs with $m = \Theta(dn)$ edges; that is, whose maximum degree is of the same order as the average degree. In particular, this is true for *sparse* graphs that have *constant degree*. This model was studied in [39, 38, 25].

In [64] it was suggested to decouple the issues of representation and type of queries allowed from the definition of distance to having a property. Specifically, it was suggested to measure the distance simply with respect to the number of edges in the graph. Namely, a graph is said to be ϵ -far from having a property, if more than ϵm edge modifications should be performed so that it obtains the property.

1.2.2 The Gap Phenomena in Graph Property Testing

In the following we consider three fundamental properties of graphs that were studied quite extensively in the context of property testing. We observe that these properties exhibit a large gap between testing results

obtained for dense graphs and for constant-degree graphs.

Testing Bipartiteness. A graph is bipartite if it is possible to partition its vertices into two parts such that there are no edges with both endpoints in the same part. Bipartiteness can be tested in the dense model using $\text{poly}(1/\epsilon)$ queries [37, 6]. Thus, the complexity of this problem in the dense case is independent of the number of vertices n . The complexity of testing bipartiteness is significantly different when considering the bounded-degree model. In [39] a lower bound of $\Omega(\sqrt{n})$ was established for this model. An almost matching upper bound was shown in [38].

Testing k -Colorability. A graph is k -colorable if it is possible to partition its vertices into k parts such that there are no edges with both endpoints in the same part. k -Colorability can be tested in the dense model using $\text{poly}(k/\epsilon)$ queries [37, 6]. Thus, similarly to the previous case, the complexity of this problem in the dense case is independent of the number of vertices n . In [25] an $\Omega(n)$ lower bound is presented for testing this property in the sparse bounded-degree case.

Testing H -Freeness. A graph is H -free if it doesn't contain H as a subgraph. The authors of [3] showed that it is possible to test H -freeness in dense graphs using a number of queries that is *independent* of n , and is of tower-type behavior in $1/\epsilon$. Alon [2, 10] proved that a super-polynomial dependence on $1/\epsilon$ is necessary for testing subgraph-freeness of all non-bipartite subgraphs. When the subgraph is bipartite then $O(1/\epsilon)$ queries suffice [2]. In the other extreme, as was observed in [39], $O(1/\epsilon)$ queries suffice for testing sparse bounded-degree graphs.

We observe that there exists a gap between testing results obtained for dense graphs, and those obtained for constant-degree graphs. Bipartiteness and k -colorability are harder in the sparse case. While H -freeness is harder for the sparse case. Note that the first two properties are *global* properties of a graph, while the third is a *local* property of a graph.

1.3 Our Results for Graph Testing

In the following we introduce a new model for testing general graphs and study the above properties in this new model. We observe a phenomena that for bipartiteness and k -colorability properties we obtain a *smooth transition* between the sparse and the dense case, while in H -freeness a *threshold-like* behavior is exhibited.

1.3.1 A Model for Testing General Graphs.

We are interested in a model that may be useful for testing all types of graphs: dense, sparse, and graphs that lie in-between the two extremes. Recall that a model for testing graph properties is defined by the distance measure used and by the queries allowed. The model of [64] is suitable for all graphs in terms of the distance measure used, since distance is measured with respect to the actual number of edges m in the graph. Thus this notion of distance adapts itself to the density of the graph, and we shall use it in our work. The focus in [64] was on testing properties that are of interest in sparse (but not necessarily bounded-degree) graphs, and hence they allowed only neighbor queries. However, consider the case in which the graph is not sparse (but not necessarily dense). In this case vertex-pair queries may become helpful. Hence, we allow our algorithms to perform both neighbor queries and vertex-pair queries.

1.3.2 Testing Bipartiteness

The Main Result. As noted above, dense graphs can be tested for bipartiteness with constant complexity, while the complexity of testing bounded-degree graphs is $\tilde{\Theta}(\sqrt{n})$. In the result presented in Chapter 3 we bridge the gap described above. In particular, we study the problem of testing bipartiteness in the model that is suitable for all densities. We present a testing algorithm whose complexity is $\tilde{O}(\min(\sqrt{n}, n/d))$ and match it with an almost tight lower bound. Furthermore, the algorithm has a one-sided error, while the lower bound holds also for two-sided error algorithms. Our result presents a *smooth transition* between the known results in the sparse and in the dense case.

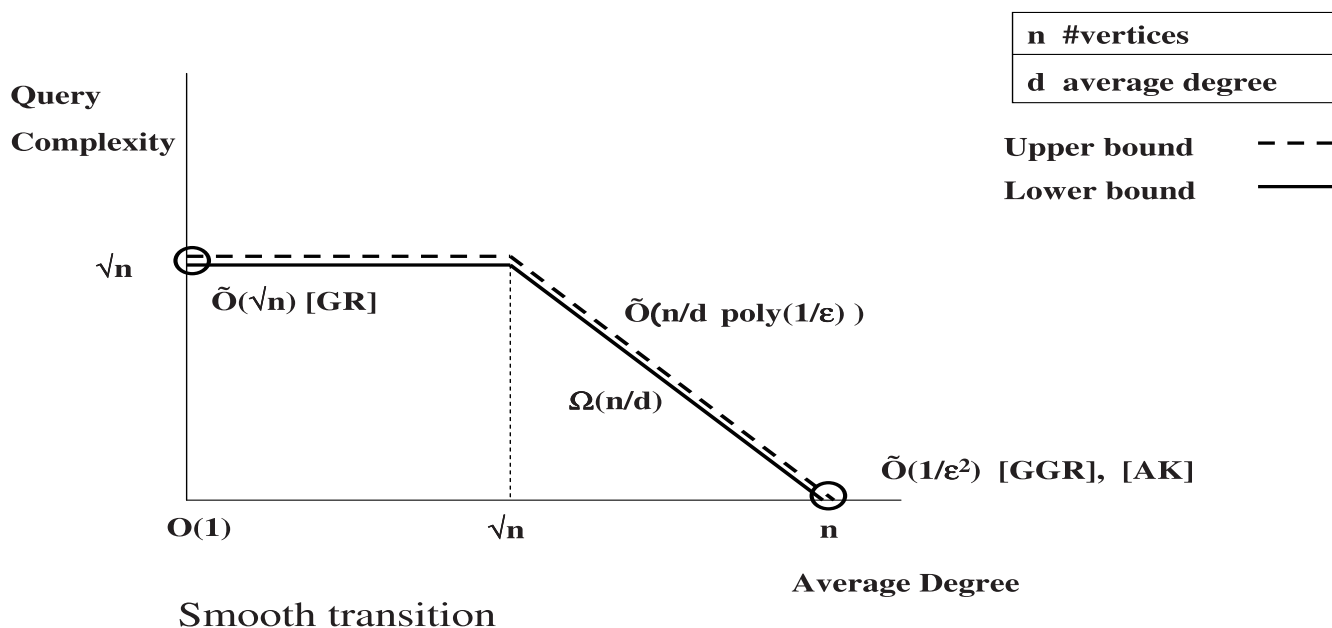


Figure 1.1: Testing bipartiteness - results for general graphs presented in Chapter 3.

Our Techniques. We first design a testing algorithm for the case of expanding regular graphs. We then show a reduction from regular graphs to expanding regular graphs. This reduction preserves the query complexity of the testing algorithm and is in the spirit of a similar one from [38]. The reduction uses that fact that for a regular non-expanding graph: one can remove a constant fraction of its edges, such that the remaining graph can be partitioned into small pieces where each is an expander. We further show a *deterministic reduction* from graphs with varying degrees to regular graphs. This reduction again preserves the query complexity of the testing algorithm. The latter reduction uses a construction of explicit expanders.

Sampling Edges Almost Uniformly in a Graph. As part of the analysis of the testing algorithm we are required to sample *edges* uniformly in a graph G , using $\tilde{O}(\min(\sqrt{n}, n/d))$ queries. This task is non-trivial when the number of edges is significantly smaller than $n^{1.5}$. We provide a sampling procedure that selects edges according to a distribution that approximates the desired uniform distribution on edges, and has the desired complexity. The approximation is such that for all but a small fraction of the m edges, the

probability of selecting an edge is $\Omega(1/m)$. This procedure may be of independent interest. There is a subsequent work [40] that uses similar ideas.

1.3.3 Testing k -Colorability

The Main Result. As was discussed above, dense graphs can be tested for k -colorability with constant complexity, while the complexity of testing bounded-degree graphs is $\Omega(n)$. In the result presented in Chapter 4 we consider testing k -colorability in general graphs. We present an algorithm whose complexity is $\tilde{O}(\min((n/d)^2, n/d \cdot \sqrt{n}, nd)) < \tilde{O}(n^{5/4})$ where d is the average degree in the graph. We also present a two-sided lower bound $\Omega(n/d)$. Thus, our upper bound is at most quadratic in the lower bound. Our result presents a *smooth transition* between the known results in the sparse and in the dense case.

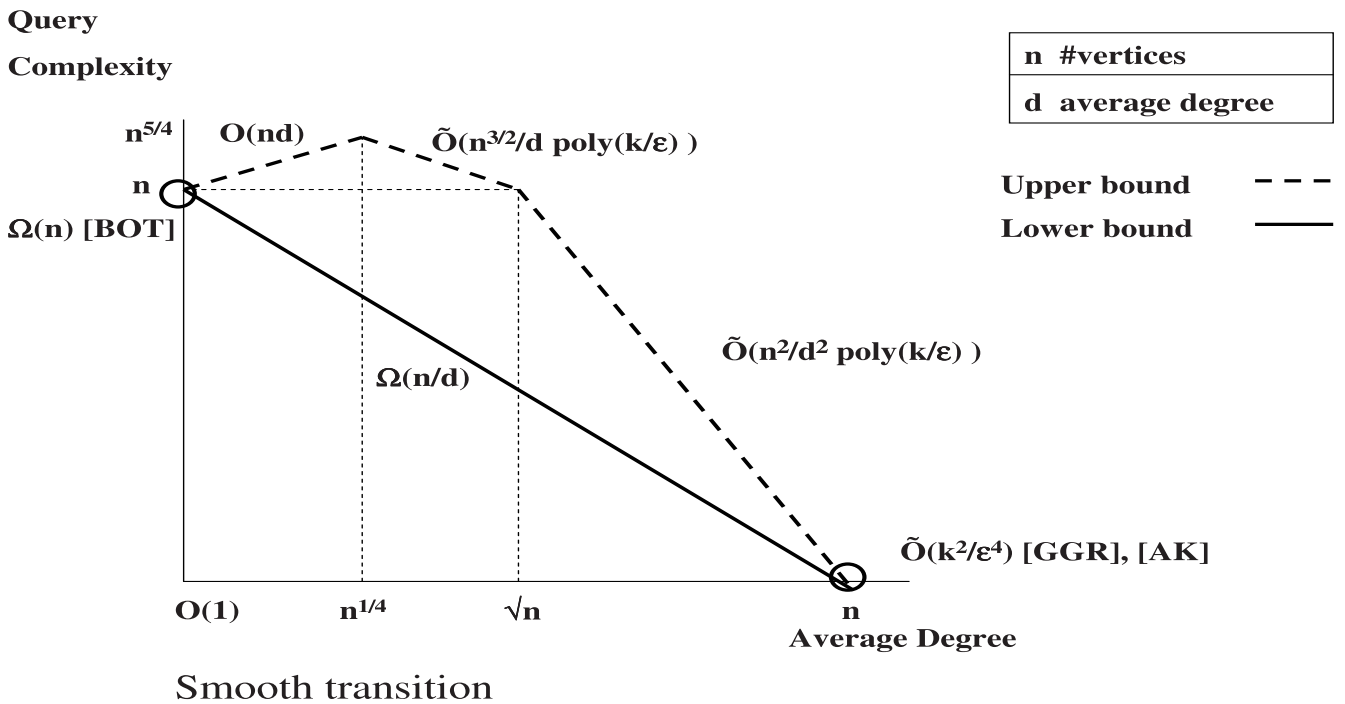


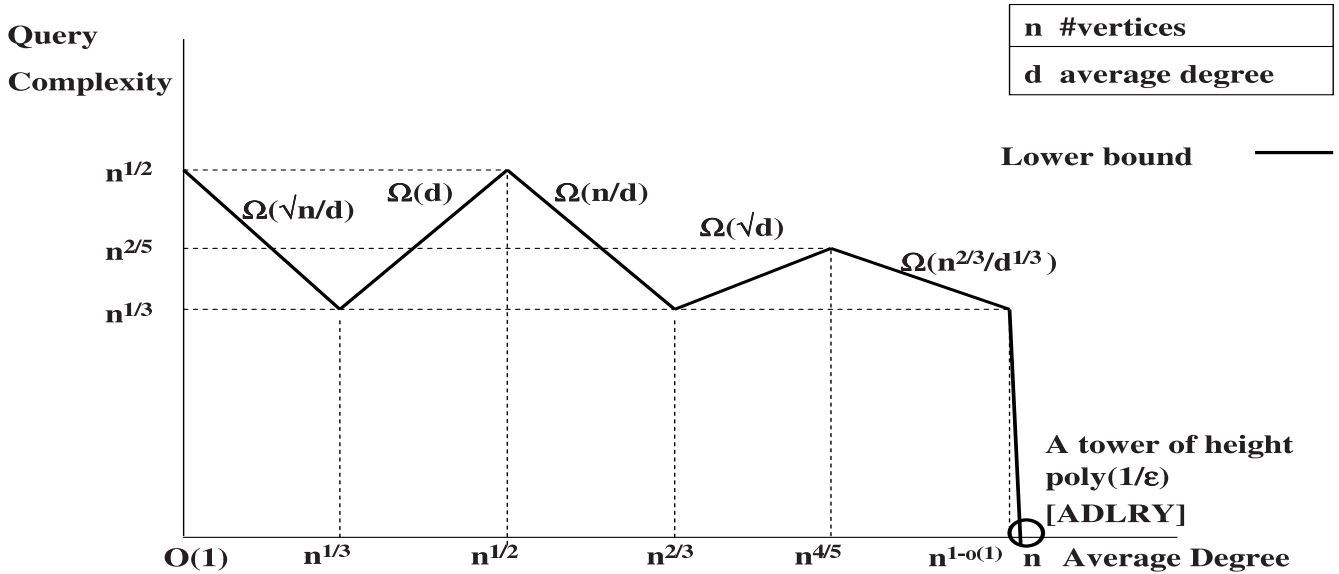
Figure 1.2: Testing k -colorability - results for general graphs presented in Chapter 4.

Our Techniques. We first design a testing algorithm for the case of regular graphs that is based on [6]. We further show a *probabilistic reduction* from regular graphs to graphs with varying degrees. This reduction preserves the query complexity of the testing algorithm. The reduction is quite general, it proved to be helpful in the case of testing bipartiteness, and it might be proven to be useful for some other graph properties.

1.3.4 Testing H -Freeness

The Main Result. As was mentioned above, testing H -freeness in dense graphs can be done using a number of queries that is *independent* of n , and is of tower-type behavior in $1/\epsilon$. Moreover, a super-polynomial dependence on $1/\epsilon$ is necessary for testing subgraph-freeness of all non-bipartite subgraphs.

Our main finding, presented in Chapter 5, is a lower bound of $\Omega(n^{1/3})$ on the number of queries required for testing H -freeness of every non-bipartite subgraph H . This bound holds for every $d < n^{1-o(1)}$. Since when $d = \Theta(n)$ the number of queries sufficient for testing does not depend on n , we observe an abrupt, *threshold-like* behavior of the complexity of testing around n . Additionally we provide sub-linear upper bounds for the problem.



Sharp transition – Threshold-like phenomena

Figure 1.3: Testing H -Freeness - results for general graphs presented in Chapter 5.

From 1-Sided to 2-Sided Lower Bound We provide a transformation from lower bounds for testing H -freeness using one-sided error algorithms to those for two-sided error algorithms; though the suggested transformation carries some technical restrictions, it is general enough to capture a variety of lower bounds of this sort.

The Edge Density of Random Cayley Graphs. We show that dense random Cayley graphs behave like quasi-random graphs in the sense that relatively large subsets of vertices have the “correct” edge density. The result for subsets of this size cannot be obtained from the known spectral techniques that only supply such estimates for much larger subsets.

1.4 Property Testing of Codes

Codes and Dual Codes. A *code* is a set of strings (*codewords*) of equal length over a finite alphabet. A code is *linear* if its codewords form a linear subspace. The *dual code* to a given linear code is obtained by taking all strings that are orthogonal to all the codewords of the considered code. The *weight* of a string is the number of its non-zero coordinates. A codeword is *short* if its weight is small (e.g. a constant independent

of the length). A *basis* of a linear code is a set of linearly independent codewords that span the code. A code has a *short basis* if it is spanned by short words. The *minimal distance* of a code is the weight of its lightest codeword.

Locally Testable Codes (LTC). A *tester* for a code is a randomized algorithm that is given a string w and a distance parameter $\epsilon > 0$. The tester is allowed to perform oracle queries about values of coordinates of w . It should accept if w is a codeword and reject with high probability if w is ϵ -far from every codeword. The notion ϵ -far indicates that at least ϵ -fraction of the coordinates of w should be changed as to obtain a codeword of the code. A code is *locally testable* if it has a tester that performs a number of queries that is a function of ϵ only, and is independent of the length of the code. The focus of this work is to study testability of linear codes. Typically, a tester for a property may have two-sided error and it can be adaptive. However, in [21] it was shown that for linear codes one-sided non-adaptive testers are as powerful as two-sided adaptive testers. Hence, throughout this work we consider such testers. Our work is focused on proving local testability of some well studied families of linear codes.

Related Research. Locally testable codes have been a subject of much research over the last years due to their close relation to *probabilistically checkable proofs* (PCP). The question of characterizing codes that are locally testable is highly complex. For surveys on the issue see [36, 69]. A great deal of attention was devoted to testing *polynomial codes*, that is, the codes whose codewords are evaluations of some (low degree) polynomials over a finite field. Hence, for polynomial codes the question of the testability of a code is equivalent to testing whether a given function is a low degree polynomial. In the latter case the testing algorithm is given a query access to a function f and a distance parameter $\epsilon > 0$. The tester should accept if f is a low-degree polynomial and reject with high probability if f differs from every such polynomial on more than an ϵ -fraction of the domain elements. Having in mind the equivalence between testing polynomial codes and testing low degree polynomials we refer to them interchangeably throughout this work.

Various families of polynomial codes differ in the size of the field, the maximum degree of the polynomials, and the number of variables. The study of testing polynomial codes was initiated by [24] who showed that the *Hadamard codes*, based on multivariate linear functions over a binary field are locally testable. The testability of the Hadamard codes was extensively studied in [15, 17, 18, 19, 24, 33, 68]. However, the Hadamard codes are a special case of a much larger family of codes that is known as the Dual-BCH codes. The question whether the general family is locally testable was open.

In works [12, 15, 14, 33, 34, 35, 66] the testability of *Reed-Solomon codes* and *Reed-Muller codes* was studied. These codes are based on univariate and multivariate low-degree polynomials over finite fields. The results obtained show that these codes are locally testable when the polynomials are over fields that are *larger than the degree-bound, d* . The question of testability of polynomial codes where the degree is greater than the field size remained open. In this work we completely answer the two open questions mentioned above.

A different series of works [20, 41], initiated by [41], attempts proving existence of locally testable codes possessing good parameters (e.g. constant rate and linearly growing minimum distance).

1.5 Our Results for Code Testing

1.5.1 Testing Low Degree Polynomials over General Fields

We consider the problem of testing, for a given finite field F and degree-bound d , whether a function $f : F^n \rightarrow F$ is a multivariate polynomial of total degree at most d over F . As noted before, multivariate low-degree polynomials over finite fields are known to be locally testable for $d + 2 \leq |F|$. The number

of queries required in this case is *polynomial* in d (and is independent of n) [34, 66]. However, the case that d is larger than the field size ($|F|$) remained open. In the following we solve this case and show that multivariate low-degree polynomials over finite fields are locally testable for *all* fields. Specifically, we show the following.

Testing Reed Muller Codes. In the first result presented in Chapter 8, we study the case $|F| = 2$ and $d \geq 2$. We show that such polynomials are locally testable. The number of queries both necessary and sufficient in this case is *exponential* in d . Specifically, the query complexity is $O(|F|^{\Theta(d)}) = O(2^{\Theta(d)})$. Hence, we encounter a very *large gap* in terms of the dependence on d between the query complexity when $|F| > d$ and the query complexity when $|F| = 2$. The “gap phenomenon” that we observe here, is not unique to testing polynomials. analogous gaps arise also in testing of graph properties, as we described in Section 1.2.2. As codewords of the Reed Muller code are evaluations of low degree polynomials over binary field ($|F| = 2$), our results imply that the Reed Muller code is locally testable.

Testing Generalized Reed Muller (GRM) Codes. In the second result presented in Chapter 8, we bridge the gap between the two cases mentioned above and show a *smooth transition* between them. In particular, we show that polynomials of degree at most d over a general finite field F where $2 < |F| < O(d)$ are locally testable. The number of queries both necessary and sufficient in this case is $O(|F|^{\Theta(d/|F|)})$. Observe that as the field size $|F|$ increases, the dependence on d decreases from being exponential to being polynomial. As codewords of the Generalized Reed Muller code are evaluations of low degree polynomials over general field, our results imply that the Generalized Reed Muller code is locally testable.

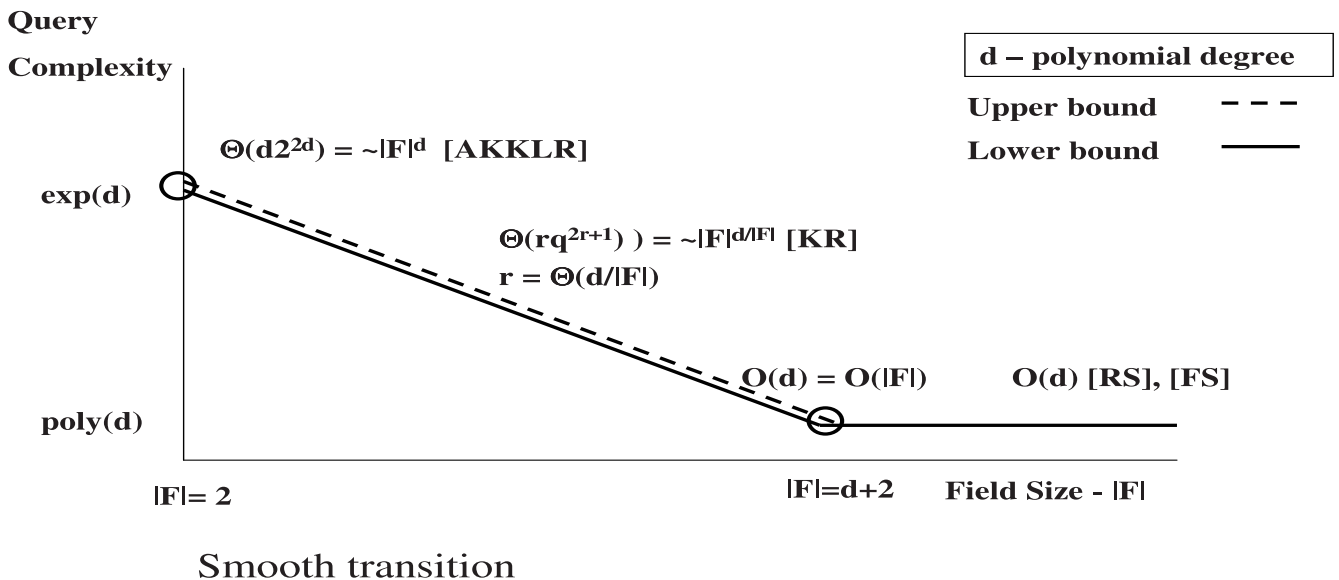


Figure 1.4: Testing GRM Codes - results presented in Chapter 8.

Characterization of Degree- d Polynomials over a General Field F . In the result presented in Chapter 8, we show that a function $f : F^n \rightarrow F$ is a polynomial of degree at most d , if and only if, its restriction to every affine subspace of dimension $\ell = \Theta(d/|F|)$ is such. Moreover, this characterization is tight. Namely, there exist polynomials of degree greater than d whose restrictions to affine subspaces of dimension less than ℓ are all degree- d polynomials. Some partial versions of this characterization were previously obtained by [24] for the case of linear functions, by [34, 66] for field size with cardinality greater than the degree, and by [31] for prime fields.

Short Basis for Generalized Reed Muller Codes. The characterization of low degree polynomials presented Chapter 8 implies that there exists a short basis to the Generalized Reed Muller Codes. Every codeword in such a basis has weight at most quadratic in the minimal weight of the code. The detailed description of a short basis obtained for the Generalized Reed Muller Codes, appears in Chapter 9. The existence of a short basis for this code was previously known only when the field size is prime [34], or is sufficiently large, in which case the minimum weight codewords span the code. Short basis for a code implies that there exists a total order of all codewords of the code, so that every two consecutive codewords differ only in few coordinates.

Self-Correction Approach. The testing results for the Generalized Reed Muller codes, were all obtained using the *self-correction approach*. The self-correction approach is very common in codes testing, and was used in e.g. [4, 24, 50, 66]. This approach can be described as follows. We start by showing a characterization for the code, which is a set of tests of constant cost, where a vector that passes them all must belong to the code. The local testability of the code is obtained as follows: We consider an input vector f . Clearly, a tester that invokes tests from the characterization set always accepts codewords. The crux of the proof is to show that few such tests are required to reject a vector f that is far from the code. Indeed, we show that if too many tests are required to reject f , then f can be **corrected** in few locations and become a codeword. Hence, f is close to the code. We stress here that this approach is totally different than the approach we develop in the next section, for code testing.

1.5.2 Testing Almost Orthogonal Linear Codes

As noted previously, the Hadamard codes which are based on multivariate linear functions over a binary field, are locally testable. The testability of the Hadamard codes was extensively studied. However, the Hadamard codes are a special case of a much larger family of codes that is known as the Dual-BCH codes. The question whether the general family is locally testable remained open. In the result described in Chapter 10, we solve this question and show that the Dual-BCH codes are locally testable. Our result is more general and applies to a larger family of codes that contain the Dual-BCH codes.

Sufficient Condition for Local Testability. In the result described in Chapter 10, we provide a sufficient condition for linear codes to be locally testable. Our condition is based on the weight distribution (spectrum) of the code and of its dual.

Regular Testability implies Short Basis for the Dual Code. We define a notion of *regular* locally testable codes, covering most known locally testable codes. Roughly speaking, a regular tester is a tester that performs several local tests where each local test involves a constant (independent of the error parameter (ϵ)) number of queries into the given word. For example, in the linearity test of Blum, Luby and Rubinfeld [24] every local test involves 3 queries into the given word. Note that apriori a tester for a code is not necessarily regular. We show that regular local testability of a linear code implies that the dual code is spanned by short

(low weight) words. The opposite statement does not hold, as shown by [21]. That is, there exist non-locally testable codes, whose dual is spanned by low weight codewords.

Testability of Almost Orthogonal Linear Codes. Codes of (large) length n and minimum distance $\frac{n}{2} - \Theta(\sqrt{n})$ have size which is at most polynomial in n . We call such codes *almost orthogonal*. We use our sufficient condition for testability to show that almost orthogonal codes are locally testable with a regular tester. Hence, their dual codes can be spanned by low weight codewords.

Testability of the Dual-BCH Codes and Short Basis for the BCH codes. Dual-BCH(n, t) codes are generalizations of the well studied Hadamard codes ($t = 1$ is Hadamard). They can be defined as binary trace images of evaluations of univariate polynomials of degree $2t$ over fields of size $n + 1$ and characteristic 2. The authors of [4] raised the question whether Dual-BCH(n, t) codes are locally testable for constant t . As these codes are known to be almost orthogonal, we solve this question and show that Dual-BCH(n, t) codes are locally testable using $O(t/\epsilon)$ queries. A lower bound on the number of queries for this problem is $\Omega(t + 1/\epsilon)$. Since we describe a regular tester for the Dual-BCH(n, t) codes, we conclude that the BCH(n, t) code is spanned by its almost lightest words, that is, by codewords of weight at most $2t + 2$, while the minimum weight is $2t + 1$.

The Spectra Approach. Consider a linear code C . For a given vector f the code $C \cup f$ is the code obtained by adding f to the subspace of C . A vector f is ϵ -far from C if it differs from every codeword of C on at least ϵ -fraction of its coordinates. Our sufficient condition for local testability of a code can be roughly stated as follows: A code C is locally testable if there exists a constant k , such that for every vector f that is ϵ -far from C the following holds. The number of k -weight codewords in the code dual to $C \cup f$ is significantly smaller than the number of k -weight codewords in the code dual to C .

The results described above were all obtained using our new sufficient condition for local testability. We used coding theory tools for analyzing the weight distributions of the relevant codes. We refer to this approach for proving local testability of codes as the *spectra approach*.

1.6 Organization

The thesis is organized as follows. It is divided into four parts.

Part 1 - Introduction and Definitions.

- Chapter 1 - Introduction.
- Chapter 2 - Preliminaries and relevant definitions being used throughout the thesis.

Part 2 - Property Testing of Graphs.

- Chapter 3 - deals with testing bipartiteness in general graphs.
- Chapter 4 - deals with testing k -colorability in general graphs.
- Chapter 5 - deals with testing subgraph-freeness in general graphs.
- Chapter 6 - provides a procedure that efficiently selects a random edge from a sparse graph.
- Chapter 7 - provides bounds on the edge density of dense random Cayley graphs.

Part 3 - Property Testing of Codes.

- Chapter 8 - deals with testing of Generalized Reed Muller codes.
- Chapter 9 - shows a characterization of low weight vectors that span a Generalized Reed Muller code.
- Chapter 10 - shows that every almost orthogonal linear code is locally testable.

Part 4 - Open Problems.

- Chapter 11 - deals with open problems arise from this thesis.

1.7 Bibliographic Notes

Chapter 3 is based on [46] that was co-authored with Michael Krivelevich and Dana Ron. Chapter 4 is based on [47] that was co-authored with Michael Krivelevich and Dana Ron. Chapter 5 is based on [5] that was co-authored with Noga Alon, Michael Krivelevich and Dana Ron. Chapter 6 is based on [46] that was co-authored with Michael Krivelevich and Dana Ron. Chapter 7 is based on [5] that was co-authored with Noga Alon, Michael Krivelevich and Dana Ron. Chapter 8 is based on [4, 50] that were co-authored with Noga Alon, Michael Krivelevich, Simon Litsyn and Dana Ron, and with Dana Ron. Chapter 9 is based on [50, 51] that were co-authored with Dana Ron. Chapter 10 is based on [48, 49] that were co-authored with Simon Litsyn.

Chapter 2

Preliminaries

2.1 Graph Definitions

Let $G = (V, E)$ be an undirected graph with n vertices labeled $1, \dots, n$, and let $m = m(G) = |E(G)|$ be the total number of edges in G . Unless stated otherwise, we assume that G contains no multiple edges. For each vertex $v \in V$ let $\Gamma(v)$ denote its set of neighbors, and let $\deg(v) = |\Gamma(v)|$ denote its degree. The edges incident to v are labeled from 1 to $\deg(v)$. We make no assumption on the order of the neighbors of a vertex. Note that each edge has two, possibly different, labels, one with respect to each of its end-points. We hence view edges as quadruples. That is, if there is an edge between v and u , and it is the i -th edge incident to v and the j -th edge incident to u , then this edge is denoted by (u, v, i, j) . When we want to distinguish between the quadruple (u, v, i, j) and the pair (u, v) then we refer to the latter as an *edge-pair*. We let $d_{\max} = d_{\max}(G)$ denote the maximum degree in the graph G and $d_{\text{avg}} = d_{\text{avg}}(G) = d$ denote the average degree in the graph (that is, $d = d_{\text{avg}}(G) = 2m(G)/n$). For a graph G and a subset of vertices $U \subseteq V$, we refer to the edges in the subgraph of G that is induced by U as the edges *spanned* by U in G .

Distance to Having a Property. Consider a fixed graph property \mathcal{P} . For a given graph G , let $e_{\mathcal{P}}(G)$ be the minimum number of edges that should be added to G or removed from G so that it obtains property \mathcal{P} . The distance of G to having property \mathcal{P} is defined as $e_{\mathcal{P}}(G)/m(G)$. In particular, we say that graph G is ϵ -far from having the property \mathcal{P} for a given distance parameter $0 \leq \epsilon < 1$, if $e_{\mathcal{P}}(G) > \epsilon \cdot m(G)$. Otherwise, it is ϵ -close to having property \mathcal{P} . In some cases we may define the distance to having a property with respect to an upper bound $m_{\max} \geq m(G)$ on the number of edges in the graph (that is, the distance to having property \mathcal{P} is defined as $e_{\mathcal{P}}(G)/m_{\max}$). For example, if the graph is dense, so that $m(G) = \Omega(n^2)$ then we set $m_{\max} = n^2$, and alternatively, if the graph has some bounded degree d , then we set $m_{\max} = d \cdot n$. (In the latter case we could set $m_{\max} = (d \cdot n)/2$, but for simplicity we set the slightly higher upper bound.) If $e_{\mathcal{P}}(G)/m_{\max} > \epsilon$ then we shall say that the graph is ϵ -far from having property \mathcal{P} with respect to m_{\max} .

Testing Algorithms. A testing algorithm for a graph property \mathcal{P} is required to accept with probability at least $2/3$ every graph that has property \mathcal{P} and to reject with probability at least $2/3$ every graph that is ϵ -far from having property \mathcal{P} , where ϵ is a given distance parameter. If the algorithm always accepts graphs that have the property then it is a *one-sided error* algorithm. The testing algorithm is given the number of vertices in the graph, the number of edges in the graph, or an upper bound on this number, and it is provided with *query access* to the graph. Specifically, we allow the algorithm the following types of queries.

- The first type of queries are *degree* queries. That is, for any vertex u of its choice, the algorithm can obtain $\deg(u)$.

- The second type of queries are *neighbor* queries. Namely, for every vertex u and index $1 \leq i \leq \deg(u)$, the algorithm may obtain the i -th neighbor of vertex u . If a vertex u has less than i neighbors then the answer is some special symbol.
- The third type of queries are *vertex-pair* queries. Namely, for any pair of vertices (u, v) , the algorithm can query whether there is an edge between u and v in G .

Note that degree queries can be easily implemented using neighbor queries with cost $O(\log d_{\max}) = O(\log n)$.

Bipartiteness Property. Let (V_1, V_2) be a partition of V . We say that an edge $(u, v) \in E$ is a *violating* edge with respect to (V_1, V_2) , if u and v belong to the same subset V_b , (for some $b \in \{1, 2\}$). A graph is *bipartite* if it has a two-way partition. That is, there exists a partition of its vertices with respect to which there are no violating edges. By definition, a graph is ϵ -far from being bipartite if for every partition of its vertices, the number of violating edges with respect to the partition is greater than $\epsilon \cdot nd$. Recall that a graph is bipartite if and only if it contains no odd cycles.

k -Colorability Property. Let (V_1, \dots, V_k) be a partition of V . We say that an edge $(u, v) \in E$ is a *violating* edge with respect to (V_1, \dots, V_k) , if u and v belong to the same subset V_b , (for some $b \in \{1, \dots, k\}$). A graph is *k -colorable* if it has a k -way partition. That is, there exists a partition of its vertices with respect to which there are no violating edges. By definition, a graph is ϵ -far from being k -colorable if for every partition of its vertices, the number of violating edges with respect to the partition is greater than $\epsilon \cdot nd$.

Triangle Freeness Property. A graph G is said to be *triangle-free* if for every three vertices u, v, w in G , at least one of the three vertex-pairs (u, v) , (v, w) , or (w, u) is not an edge in G . A graph G is ϵ -far from (being) *triangle-free* if it is necessary to remove more than $\epsilon \cdot nd$ edges from G in order to obtain a triangle-free graph.¹

2.2 Polynomial Definitions

Let F be a field of cardinality q and characteristic p (that is, $q = p^s$ where p is prime). Let $\omega \in F$ be a generator of the field F , so that $F = \{0, \omega^1 = \omega, \omega^2, \dots, \omega^{q-2}, \omega^{q-1} = 1\}$. We note that this order of the elements in F in which 1 is represented by ω^{q-1} rather than ω^0 will serve us better than the more standard order $F = \{0, \omega^0 = 1, \omega^1 = \omega, \omega^2, \dots, \omega^{q-2}\}$. In particular, using this order, for any $n \geq 1$ we consider a one-to-one mapping between $[q-1]^n$ and F^n (where $[q-1] \stackrel{\text{def}}{=} \{0, 1, \dots, q-1\}$). Specifically, for each $\beta \in [q-1]^n$, the point $x \in F^n$ that *corresponds* to β is defined as follows: $x_i = 0$ if $\beta_i = 0$ and otherwise $x_i = \omega^{\beta_i}$.

Consider the standard (and unique) representation of the function f as a polynomial over F with degree at most $q-1$ in each variable:

$$f(x) = \sum_{\alpha \in [q-1]^n} C_\alpha^f \cdot x^\alpha \quad \text{where} \quad x^\alpha = \prod_{i=1}^n x_i^{\alpha_i} \quad (2.1)$$

Here \vec{C}^f is the *coefficients vector* (indexed by points $\alpha \in [q-1]^n$). If we view the function f as a q^n -dimensional vector \vec{f} (where for $\beta \in [q-1]^n$, f_β is the evaluation of f on the point $x \in F^n$ that corresponds

¹Since the number of edges in the graph is $(nd)/2$, the standard definition of ϵ -far would be that more than $(\epsilon nd)/2$ edges should be removed so that the graph becomes triangle-free. In order to simplify the presentation we slightly modify the definition.

to β), then we can write Equation (2.1) in the following equivalent form:

$$\vec{f} = \mathcal{H}_n \cdot \vec{C}^f \quad (2.2)$$

Here \mathcal{H}_n is the $q^n \times q^n$ matrix whose entries are indexed by pairs $\beta, \alpha \in [q-1]^n$ where $\mathcal{H}_n(\beta, \alpha)$ is the evaluation of the term x^α at the point $x \in F^n$ that corresponds to β . By inverting \mathcal{H}_n (which is non-singular) we can represent the coefficients vector \vec{C}^f in terms of \vec{f} as follows:

$$\vec{C}^f = \mathcal{A}_n \cdot \vec{f} \quad (2.3)$$

Both \mathcal{H}_n and \mathcal{A}_n can be defined recursively using tensor products: $\mathcal{H}_n = \mathcal{H}_1 \otimes \mathcal{H}_{n-1}$ and $\mathcal{A}_n = \mathcal{A}_1 \otimes \mathcal{A}_{n-1}$ where

$$\mathcal{H}_1 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & \omega & \omega^2 & \dots & 1 \\ 1 & \omega^2 & \omega^4 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{q-2} & \omega^{2(q-2)} & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix} \quad \mathcal{A}_1 = (-1) \cdot \begin{pmatrix} -1 & 0 & 0 & \dots & 0 \\ 0 & \omega^{-1} & \omega^{-2} & \dots & 1 \\ 0 & \omega^{-2} & \omega^{-4} & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \omega^{-(q-2)} & \omega^{-2(q-2)} & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix} \quad (2.4)$$

For our purposes, the important thing that should be noted is that every coefficient C_α^f of the polynomial representation of f is some (easy to compute) linear combination of the values of f on different domain elements $x \in F^n$. In particular, for $\alpha = \langle q-1, \dots, q-1 \rangle$, $C_\alpha^f = (-1)^n \cdot \sum_{x \in F^n} f(x)$.

Definition 2.2.1 Let $\text{POLY}_{n,d}$ denote the class of all functions $f : F^n \rightarrow F$ that are polynomials of total degree at most d (where the degree in each variable is at most $q-1$). Namely, if we consider the representation of f as defined in Equation (2.1), then $C_\alpha^f = 0$ for every $\alpha \in [q-1]^n$ such that $\sum_{i=1}^n \alpha_i > d$.

Definition 2.2.2 For $m \geq 1$ and any choice of a point $y \in F^n$ and m linearly independent points $y_1, \dots, y_m \in F^n$, let $S(y_0, y_1, \dots, y_m)$ denote the affine subspace of dimension m that contains all points of the form $y_0 + \sum_{i=1}^m a_i y_i$, where $a_1, \dots, a_m \in F$. Note that y_0 has a different role from all other y_i 's.

Observe that in the special case that y_0 is a linear combination of y_1, \dots, y_m , then $S(y_0, y_1, \dots, y_m)$ is a linear subspace of dimension m .

Definition 2.2.3 For a function $f : F^n \rightarrow F$, a point $y_0 \in F^n$, and m linearly independent points $y_1, \dots, y_m \in F^n$, we denote by $f_{|(y_0, y_1, \dots, y_m)}$ the restriction of f to the affine subspace $S(y_0, y_1, \dots, y_m)$. Namely, $f_{|(y_0, y_1, \dots, y_m)} : F^m \rightarrow F$ is defined as follows: for every $v \in F^m$, $f_{|(y_0, y_1, \dots, y_m)}(v) = f(y_0 + \sum_{i=1}^m v_i y_i)$. With slight abuse of notation (and for sake of succinctness), we shall sometimes use the notation $f|_S$ instead, where $S = S(y_0, y_1, \dots, y_m)$ is the subspace spanned by the points. In case the set of points spanning the subspace S is not explicitly stated, then $f|_S$ is determined by some canonical choice of a basis.²

Definition 2.2.4 For any two functions $f, g : F^n \rightarrow F$, let $\text{dist}(f, g) = \Pr_{y \in F^n} [f(y) \neq g(y)]$ (where y is selected uniformly).

²Our interest lies in the *degree* of these functions (represented as polynomials). Since for any given subspace this degree is invariant with respect to the choice of the basis, the particular choice of the basis is only a matter of convenience.

Testing Algorithms. A testing algorithm is given query access to $f : F^n \rightarrow F$ and a parameter $\epsilon > 0$. If f is a polynomial of degree at most d then the testing algorithm must accept. On the other hand, if f differs from every such polynomial on more than an ϵ -fraction of the domain elements, then the test should reject with probability at least $2/3$.

2.3 Code Definitions

Codes and Dual Codes. A *code* is a set of strings (*codewords*) of equal length over a finite alphabet. The (Hamming) *distance* between pair of codewords is the number of coordinates in which they differ. The *length* of a code is the length of its codewords. The *minimum distance* of a code is the minimum of pairwise distances between its distinct codewords. A code is *linear* if its codewords form a linear subspace (here we assume that the alphabet is endowed with the structure of a finite field). The *dual code* to a given linear code is obtained by taking all strings that are orthogonal to the codewords of the considered code. The *weight* of a string is the number of its non-zero coordinates. The minimum distance of a linear code coincides with the minimum weight of a non-zero codeword. Informally, a codeword is *short* if its weight is small (e.g. a constant independent of the length). A *basis* of a linear code is a set of linearly independent codewords that span the code. A code has a short basis if it is spanned by short words. Hereby we assume that the length of considered codes is large enough to justify all approximations.

Testing Algorithms. A *tester* for a code is a randomized algorithm that is given a string x and a distance parameter $\epsilon > 0$. The tester is allowed to perform oracle queries about values of coordinates of x . It should accept if x is a codeword and reject with probability at least $2/3$. if x is ϵ -far from every codeword. The notion ϵ -far indicates that at least ϵ -fraction of the coordinates of x should be changed for obtaining a codeword of the code. A code is *locally testable* if it has a tester that performs a number of queries that is a function of ϵ only, and is independent of the length of the code.

Let x be a vector in F^n of weight $w(x)$. For a linear code $C \subseteq F^n$ and $v \in F^n$ such that $v \notin C$, the v -*coset* of C is $C + v \stackrel{\text{def}}{=} \{c + v | c \in C\}$. Note that $|C + v| = |C|$. Let $C \cup v \stackrel{\text{def}}{=} C \cup (C + v) = \{c | c \in C \vee c \in C + v\}$, $|C \cup v| = 2|C|$. The *covering radius* of C , $R(C)$ is the maximum distance from the code to a vector in the ambient space.

Claim 2.3.1 [Johnson Bound, see [58]] Consider a code C of length n and distance d . The number of words of weight x in C is at most $\frac{dn}{2x^2 - 2nx + dn}$

Let $C \subseteq F^n$ be a linear code. The *distance distribution* of C : $B^C = (B_0^C, B_1^C, \dots, B_n^C)$ is: $B_i^C = |\{c \in C | w(c) = i\}|$. The distance distribution B^{C^\perp} of the dual code C^\perp is called the *dual spectrum* of C .

The following claim shows that B^{C^\perp} is uniquely determined by B^C :

Claim 2.3.2 [MacWilliams transform [29]] For a linear binary code C of length n , $B_j^{C^\perp} = \frac{1}{|C|} \sum_{i=0}^n B_i^C P_j^n(i)$, where $P_j^n(i) = \sum_{\ell=0}^j (-1)^\ell \binom{i}{\ell} \binom{n-i}{j-\ell}$ is the Krawtchouk polynomial of degree j .

For a coset $C + v$: $B_j^{[C+v]^\perp} \stackrel{\text{def}}{=} \frac{1}{|C|} \sum_{i=0}^n B_i^{[C+v]} P_j^n(i)$.

For properties of the Krawtchouk polynomials see e.g. survey [55]. The MacWilliams transform can be stated also for non-binary linear codes. There, non-binary Krawtchouk polynomials are used. For simplicity of presentation we stick to the binary case. However, our results apply to general linear codes as well. Following is a useful simple bound on values of Krawtchouk polynomials.

Claim 2.3.3 $\forall k, P_k(i) \leq \frac{|n-2i|^k}{k!}$.

Dual-BCH Codes (see [58]): Let F_q be a finite field of size $q = n + 1 = 2^m$, having α as a primitive element. Let $F(x)$ be the set of polynomials of the form $f(x) = a_1x + a_3x^3 + \dots + a_{2t-1}x^{2t-1}$ with the coefficients from F_q . Clearly $|F(x)| = 2^{mt} = (n+1)^t$. Let $Tr(x)$, $Tr(x) = x + x^2 + x^{2^2} + x^{2^3} + \dots + x^{2^{m-1}}$, be the trace function linearly mapping elements from F_q to F_2 . Then the collection of vectors c , $c \in \{0, 1\}^n$, with components $c_i = Tr(f(\alpha^i))$, $i = 1, \dots, 2^m - 1$, and f running through $F(x)$, constitute the $C_{dBCH(t)}$ code.

For the reader familiar with the definition using the parity check matrix of $C_{BCH(t)}$, we note that both definitions are equivalent. The reason we preferred the above one is that it relates the Dual-BCH codes to the polynomial ones.

Claim 2.3.4 [Weil, Carlitz Uchiyama, see [58]]: For $c \in C_{dBCH(t)}$:

$$\frac{n}{2} - (t-1)\sqrt{n} \leq w(c) \leq \frac{n}{2} + (t-1)\sqrt{n}$$

BCH Codes (see [58]): $C_{BCH(t)}$ code is the code dual to $C_{dBCH(t)}$. Its minimum distance is $2t + 1$.

Claim 2.3.5 [54] In $C_{BCH(t)}$ the number of codewords of weight i , $B_i^{C_{BCH(t)}}$, is:

$$B_i^{C_{BCH(t)}} = \frac{\binom{n}{i}}{(n+1)^t} (1 + O(\frac{1}{n})) [= \frac{\binom{n}{i}}{|C_{dBCH(t)}|} (1 + O(\frac{1}{n}))]$$

Throughout this work we consider $C_{BCH(t)}$ and $C_{dBCH(t)}$ codes where t is constant and n is large enough.

Part II

Testing of Graphs

Chapter 3

Testing Bipartiteness in General Graphs

3.1 Introduction

One of the properties that has received quite a bit of attention in the context of property testing, is *bipartiteness*. Recall that a graph is bipartite if it is possible to partition its vertices into two parts such that there are no edges with both endpoints in the same part. This property was first studied in [37] where it was shown that bipartiteness can be tested in the adjacency-matrix model by a simple algorithm using $\tilde{O}(1/\epsilon^3)$ queries. This was improved in [6] to $\tilde{O}(1/\epsilon^2)$ queries. The best lower bound known in this model is $\Omega(1/\epsilon^{1.5})$, due to [26]. Thus the complexity of this problem in the adjacency-matrix model is independent of the number of vertices n and is polynomial in $1/\epsilon$. It is interesting to note that testing bipartiteness is considered implicitly already in [27]. The result in [27] can be used to obtain a testing algorithm in the adjacency-matrix model whose query complexity does not depend on the size of the graph, but whose dependence on ϵ is a tower of height polynomial in $1/\epsilon$.

The complexity of testing bipartiteness is significantly different when considering the bounded-degree incidence-lists model. In [39] a lower bound of $\Omega(\sqrt{n})$ was established in this model, for constant ϵ and constant d (where d is the degree bound). An almost matching upper bound of $\tilde{O}(\sqrt{n} \cdot \text{poly}(1/\epsilon))$ was shown in [38]. Thus, in the case of bipartiteness there is a large gap between the results that can be obtained for dense graphs and for constant-degree graphs. Here we venture into the land of graphs that are neither necessarily sparse, nor necessarily dense, and study the complexity of testing bipartiteness. Other graph properties exhibit similar (and sometimes even larger) gaps, and hence we believe that understanding the transformation from sparse to dense graphs is of general interest.

3.1.1 Our Results

In this chapter we bridge the gap between the two cases mentioned above and show a smooth transition between them. In particular, we present two complementary results for n -vertex graphs having m edges:

- We describe and analyze an algorithm for testing bipartiteness in general graphs whose query complexity and running time are $O(\min(\sqrt{n}, n^2/m) \cdot \text{poly}(\log n/\epsilon))$. The algorithm has a one-sided error (i.e., it always accepts bipartite graphs). Furthermore, whenever it rejects a graph it provides *evidence* that the graph is not bipartite in the form of an odd cycle¹ of length $\text{poly}(\log n/\epsilon)$.
- We present an almost matching lower bound of $\Omega(\min(\sqrt{n}, n^2/m))$ (for a constant ϵ). This bound holds for all testing algorithms (that is, for those that are allowed a two-sided error and are adaptive). Furthermore, the bound holds for regular graphs.

¹We use the term “odd cycle” as a shorthand for “odd-length cycle”.

As seen from the above expressions, as long as $m = O(n^{1.5})$, that is, the average degree is $O(\sqrt{n})$, the complexity of testing (in terms of the dependence on n) is $\tilde{O}(\sqrt{n})$. Once the number of edges goes above $n^{1.5}$, we start seeing a decrease in the query complexity, which in this case is at most $O((n^2/m) \cdot \text{poly}(\log n/\epsilon))$. In terms of our algorithm, this is exactly the point where our algorithm starts exploiting its access to vertex-pair queries. Our lower bound shows that this behavior of the query complexity is not only an artifact of our algorithm but is inherent in the problem.

Notes:

1. Observe that even if the graph is sparse then we obtain a new result that does not follow from [38]. Namely, we have an algorithm with complexity $\tilde{O}(\sqrt{n} \cdot \text{poly}(1/\epsilon))$ for sparse graphs with varying degrees.
2. We note that the algorithm does not actually require to be given the number of edges, m , in the graph, but can instead compute an estimate of this number. Such an estimate can be obtained without increasing the query complexity of the algorithm [32, 40], and we discuss this issue shortly in Section 3.3.
3. We assume that $m = \Omega(n)$. Since our distance measure is with respect to the number of edges in the graph, without such an assumption it would be impossible to distinguish between the following two (families of) graphs: a graph that consists of many isolated edges and a single very small subgraph that is far from bipartite (e.g., a clique), a graph that consists of many isolated vertices and a very small bipartite subgraph. We discuss this issue shortly in Section 3.3 as well.

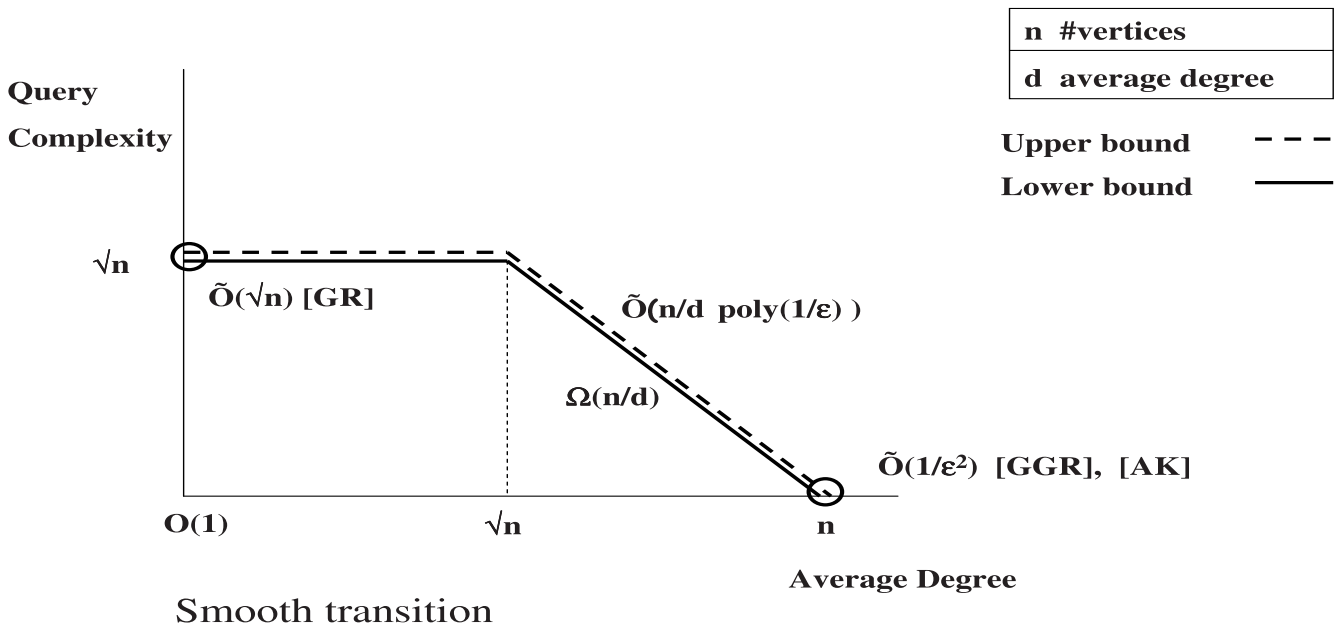


Figure 3.1: Testing bipartiteness - illustration of results for general graphs obtained in this chapter.

3.1.2 Our Techniques

We present our algorithm in two stages. First we describe an algorithm that works for almost-regular graphs, that is, graphs in which the maximum degree is of the same order as the average degree. The algorithm and its analysis closely follow the algorithm and analysis in [38]. Indeed, as long as the degree d of the graph is at most \sqrt{n} , we execute the algorithm described in [38]. The place where we depart from [38] is in the usage of vertex-pair queries once $d > \sqrt{n}$. We refer to our first algorithm as Test-Bipartite-Reg.

In the second stage we show how to reduce the problem of testing bipartiteness of general graphs to testing bipartiteness of almost-regular graphs. Namely, we show how, for every given graph G , it is possible to define a graph G' such that: (1) G' has roughly the same number of vertices and edges as G , and its maximum degree is of the same order as its average degree (which is roughly the same as the average degree in G); (2) If G is bipartite then so is G' , and if G is far from being bipartite then so is G' . We then show how to emulate the execution of the algorithm Test-Bipartite-Reg on G' given query access to G , so that we may accept G if it accepts G' , and reject G if it rejects G' .

In the course of this emulation we are confronted with the following interesting problem: We would like to sample vertices in G according to their degrees (which aids us in sampling vertices uniformly in G' , a basic operation that is required by Test-Bipartite-Reg). The former is equivalent to sampling *edges* uniformly in G . In order not to harm the performance of our testing algorithm, we are required to perform this task using $\tilde{O}(\min(\sqrt{n}, n^2/m))$ queries. If m is sufficiently large (once again, if $m \geq n^{1.5}$), this can be performed without increasing the complexity of our algorithm simply by sampling sufficiently many pairs of vertices in G . However, we do not know how to perform this task exactly (in an efficient manner) when the number of edges is significantly smaller than $n^{1.5}$. Nonetheless, we use a sampling procedure that selects edges according to a distribution that approximates the desired uniform distribution on edges. A detailed description of this procedure appears in Chapter 6.

We establish our lower bound by describing, for every pair n, d (n even, $d \geq 64$), two distributions over d -regular graphs. In one distribution all graphs are bipartite by construction. For the other distribution we prove that almost all graphs are far from being bipartite. We then show that every testing algorithm that can distinguish between a graph chosen randomly from the first distribution (which it should accept with probability at least $2/3$), and a graph chosen randomly from the second distribution (which it should reject with probability at least $2/3$), must perform $\Omega(\min(\sqrt{n}, n/d)) = \Omega(\min(\sqrt{n}, n^2/m))$ queries.

Our lower bound proof implies the necessity of both neighbor queries and vertex-pair queries in obtaining an upper bound whose dependence on n and m is $\tilde{O}(\min(\sqrt{n}, n^2/m))$. Specifically, if only neighbor queries are allowed then our analysis implies a lower bound of $\Omega(\sqrt{n})$ for every m , which is higher than $\tilde{O}(n^2/m)$ when $m = \omega(n^{1.5})$. If only vertex-pair queries are allowed then our analysis implies a lower bound of $\Omega(n^2/m)$, which is above the upper bound of $\tilde{O}(\sqrt{n})$ when $m = o(n^{1.5})$.

3.2 The Algorithm for the Almost-Regular Case

In this section we describe an algorithm that accepts every bipartite graph and that rejects with probability at least $2/3$ every graph that is ϵ -far from being bipartite with respect to an upper bound $m_{\max} = d_{\max}n$ on the number of edges. Namely, this algorithm rejects (with probability at least $2/3$) graphs for which the number of edges that need to be removed so that they become bipartite is greater than $\epsilon \cdot m_{\max} = \epsilon \cdot d_{\max}n$. The query complexity and running time of this algorithm are $O(\min(\sqrt{n}, n/d_{\max}) \cdot \text{poly}(\log n/\epsilon))$.

In the case where the graph is almost-regular, that is, the maximum degree of the graph d_{\max} is of the same order as the average degree, d_{avg} , then we essentially obtain a tester as desired (since in such a case $\epsilon d_{\max}n = O(\epsilon m)$). However, in general, d_{\max} may be much larger d_{avg} (for example, it is possible that

$d_{\max} = \Theta(n)$ while $d_{\text{avg}} = \Theta(1)$). To deal with the general case we show in the next section (Section 3.3) how to reduce the problem in the general case to the special case of $d_{\max} = O(d_{\text{avg}})$.

A High Level Description of the Algorithm

Throughout this section let $d = d_{\max}$. Our algorithm, whose pseudo-code appears in Figure 3.2, builds on the testing algorithm for bipartiteness described in [38]. The query complexity of that algorithm is $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$ and it works with respect to $m_{\max} = dn$ as well. In fact, as long as $d \leq \sqrt{n}$ our algorithm is the same as the algorithm in [38].

In particular, as in [38], our algorithm selects $\Theta(1/\epsilon)$ *starting vertices* and from each it performs several random walks (using neighbor queries), each walk of length $\text{poly}(\log n/\epsilon)$. The exact form of these random walks is described momentarily. If $d \leq \sqrt{n}$ then the number of these random walks from each starting vertex s is $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$, and the algorithm simply checks whether an odd cycle was detected in the course of these random walks. Specifically, the algorithm checks whether there exists some vertex v that is reached at the end of two different walks from s , where one walk corresponds to a path in the graph with even length, and one walk corresponds to a path with odd length. The existence of such a vertex v implies an odd cycle that contains s and v , and the algorithm rejects in such a case.

If $d > \sqrt{n}$ then there are two important modifications as compared to the case $d \leq \sqrt{n}$ (which, as noted above, follows [38]). These modifications reduce the number of queries performed as the degree increases.

1. The number of random walks performed from each starting vertex is reduced to $O(\sqrt{n/d} \cdot \text{poly}(\log n/\epsilon))$ (as compared to $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$ walks for the case $d \leq \sqrt{n}$).
2. The algorithm still considers the vertices reached at the end of these walks, but now it performs an additional step. It partitions these vertices into two subsets, denoted A_0 and A_1 , according to the parity of the lengths of the paths corresponding to the walks. The algorithm then performs vertex-pair queries on each pair of vertices that belong to the same subset. If any edge (u, v) is detected for $u, v \in A_0$ or $u, v \in A_1$, then the algorithm has evidence to an odd cycle that included s (the starting vertex), u and v , and it rejects. The total number of vertex-pair queries is $O((n/d) \cdot \text{poly}(\log n/\epsilon))$.

On a very intuitive level, if a graph is far from being bipartite then many edges (and vertices) belong to many odd cycles. The difference between the two cases described above is that if $d \leq \sqrt{n}$ then the algorithm tries to reach the same vertex twice, once via an even-length path and once via an odd-length path. To this end it performs about \sqrt{n} random walks (so as to “hit” the same vertex twice). In the case $d > \sqrt{n}$, the algorithm performs much fewer walks and so we cannot expect to reach the same vertex twice. However, since the edge-density is higher, we do expect to have an edge in the subgraph induced by the ending points of the walk. In particular, as our analysis shows, we expect to see such an edge between vertices that are reached via paths whose lengths have the same parity.

Random Walks and Paths in the Graph. The random walks performed are defined as follows: At each step, if the degree of the current vertex v is $d' \leq d$, then the walk *remains* at v with probability $1 - \frac{d'}{2d} \geq \frac{1}{2}$, and for each $u \in \Gamma(v)$, the walk *traverses* to u with probability $\frac{1}{2d}$. The important property of the random walk is that the stationary distribution it induces over the vertices is uniform.

To every walk (or, more generally, to any sequence of steps), there corresponds a *path* in the graph. The path is determined by those steps in which an edge is traversed (while ignoring all steps in which the walk stays at the same vertex). Such a path is not necessarily simple, but does not contain self loops. Note that when referring to the length of a walk, we mean the total number of steps taken, including steps in which the walk remains at the current vertex, while the length of the corresponding path does not include these steps.

Test-Bipartite-Reg(n, d_{\max}, ϵ)

- Repeat $T = \Theta(\frac{1}{\epsilon})$ times:
 1. Uniformly select a vertex s in V .
 2. If Odd-Cycle(s) returns found then output reject.
- In case no call to Odd-Cycle returned found then output accept.

Odd-Cycle(s)

1. If $d = d_{\max} \leq \sqrt{n}$ then let $K \stackrel{\text{def}}{=} \Theta\left(\frac{\sqrt{n} \cdot \log^{1/2}(n/\epsilon)}{\epsilon^3}\right)$ and $L \stackrel{\text{def}}{=} \Theta\left(\frac{\log^3(n/\epsilon)}{\epsilon^5}\right)$. Otherwise ($d > \sqrt{n}$), let $K \stackrel{\text{def}}{=} \Theta\left(\frac{\sqrt{n/d} \cdot \log^{1/2}(n/\epsilon)}{\epsilon^8}\right)$, and $L \stackrel{\text{def}}{=} \Theta\left(\frac{\log^6(n/\epsilon)}{\epsilon^8}\right)$.
2. Perform K random walks starting from s , each of length L .
3. Let A_0 (A_1) be the set of vertices that appear at the ends of the walks performed in the previous step whose paths are of even (odd) length.
4. If $d \leq \sqrt{n}$ then check whether $A_0 \cap A_1 \neq \emptyset$. If the intersection is non-empty then return found, otherwise return not-found.
5. Else ($d > \sqrt{n}$), perform vertex-pair queries between every pair of vertices $u, v \in A_0$ ($u, v \in A_1$). If an edge is detected then return found, otherwise return not-found.

Figure 3.2: Algorithm Test-Bipartite-Reg for testing bipartiteness with respect to the upper bound $m_{\max} = d_{\max} \cdot n$ on the number of edges, and the procedure Odd-Cycle for detecting odd cycles in the graph G .

Theorem 3.2.1 *The algorithm Test-Bipartite-Reg accepts every graph that is bipartite, and rejects with probability at least $2/3$ every graph that is ϵ -far from being bipartite with respect to $m_{\max} = d_{\max}n$. Furthermore, whenever the algorithm rejects a graph it outputs a certificate to the non-bipartiteness of the graph in form of an odd cycle of length $\text{poly}(\log n/\epsilon)$. The query complexity and running time of the algorithm are $O(\min(\sqrt{n}, n/d_{\max}) \cdot \text{poly}(\log n/\epsilon))$.*

Note that the algorithm can work when G contains self-loops and multiple-edges. The latter will be of importance in the next section.

As a direct corollary of Theorem 3.2.1 (using $m(G) = (nd_{\text{avg}}(G))/2$) we get:

Corollary 3.2.2 *For a given graph G , let $\gamma(G) \stackrel{\text{def}}{=} d_{\max}(G)/d_{\text{avg}}(G)$. Then Test-Bipartite-Reg($n, d_{\max}(G), \epsilon/(2\gamma(G))$) accepts every graph that is bipartite, and rejects with probability at least $2/3$ every graph that is ϵ -far from being bipartite (with respect to $m(G)$).*

The corollary below will become useful in the next section.

Corollary 3.2.3 *If G is ϵ -far from being bipartite with respect to $m_{\max} = d_{\max}n$, then $\Omega(\epsilon)$ -fraction of its vertices s are such that Odd-Cycle(s) returns found with probability at least $\frac{2}{3}$.*

The completeness part of Theorem 3.2.1 (i.e., showing that the algorithm accepts bipartite graphs) is straightforward. We focus on proving the soundness of the algorithm (i.e., that graphs that are ϵ -far from being bipartite are rejected with probability $\frac{2}{3}$). What we eventually show (in Subsection 3.2.6) is the contrapositive statement. Namely, that if the test accepts G with probability greater than $\frac{1}{3}$ then there exists an ϵ -good partition of G .

Our analysis follows the analysis presented in [38] quite closely. In particular, whenever possible we refer the reader to proofs given in [38]. Here we present what is necessary to establish the correctness of our algorithm and in particular those proofs in which we diverge from [38]. Since the algorithm for the case $d_{\max} \leq \sqrt{n}$ is fully analyzed in [38], from this point on we assume $d_{\max} > \sqrt{n}$ and analyze the algorithm for this case.

3.2.1 Gaining Intuition: The Rapidly-Mixing Case

To gain some intuition, consider first the following “ideal” case: From each starting vertex s in G , and for every $v \in V$, the probability that a random walk of length $L = \text{poly}((\log n)/\epsilon)$ ends at v is at least $\frac{1}{2n}$ and at most $\frac{2}{n}$ – i.e., approximately the probability assigned by the stationary distribution. (Note that this ideal case occurs when G is an expander). Let us fix a particular starting vertex s . For each vertex v , let p_v^0 be the probability that a random walk (of length L) starting from s , ends at v and corresponds to an even-length path. Define p_v^1 analogously for odd paths. Then, by our assumption on G , for every v , $p_v^0 + p_v^1 \geq \frac{1}{2n}$. We consider two cases regarding the sum $\sigma(G) \stackrel{\text{def}}{=} \sum_{\substack{v,u \in V \\ (v,u) \in E}} (p_v^0 p_u^0 + p_v^1 p_u^1)$.

In case $\sigma(G)$ is (relatively) “small”, we show that there exists a partition (V_0, V_1) of V that is ϵ -good, and so G is ϵ -close to being bipartite. Otherwise (i.e., when the sum is not “small”), we show that the rejection probability is bounded away from zero. This implies that in case G is accepted with probability at least $\frac{1}{3}$ then G is ϵ -close to being bipartite.

Consider first the case in which $\sigma(G) < c \cdot \frac{\epsilon d}{n}$ for some suitable constant $c < 1$. Let the partition (V_0, V_1) be defined as follows: $V_0 = \{v : p_v^0 \geq p_v^1\}$ and $V_1 = \{v : p_v^1 > p_v^0\}$. Consider a particular vertex $v \in V_0$. By definition of V_0 and our rapid-mixing assumption, $p_v^0 \geq \frac{1}{4n}$.

$$\begin{aligned}
\sigma(G) &= \sum_{\substack{v,u \in V \\ (v,u) \in E}} (p_v^0 p_u^0 + p_v^1 p_u^1) \\
&\geq \sum_{\substack{v,u \in V_0 \\ (v,u) \in E}} p_v^0 p_u^0 + \sum_{\substack{v,u \in V_1 \\ (v,u) \in E}} p_v^1 p_u^1 \\
&\geq \sum_{\substack{v,u \in V_0 \\ (v,u) \in E}} \frac{1}{16n^2} + \sum_{\substack{v,u \in V_1 \\ (v,u) \in E}} \frac{1}{16n^2} \\
&\geq \frac{1}{16n^2} \cdot (\text{The number of violating edges w.r.t. } (V_0, V_1)). \tag{3.1}
\end{aligned}$$

Thus, if there are more than ϵdn violating edges with respect to (V_0, V_1) , then $\sigma(G) > \frac{1}{16} \cdot \frac{\epsilon d}{n}$ which contradicts our case hypothesis concerning $\sigma(G)$ assuming $c \leq 1/16$.

We now turn to the second case, $\sigma(G) \geq c \cdot \frac{\epsilon d}{n}$. For every fixed pair $i, j \in \{1, \dots, K\}$, (recall that $K = \Theta(\sqrt{n/d} \cdot \text{poly}(\log n/\epsilon))$ is the number of walks taken from s), consider the 0/1 random variable $\eta_{i,j}$ that is 1 if and only if both the i -th and the j -th walks have path length with the same parity, and if the end-points of the paths are vertices u, v such that $(u, v) \in E$. Then for every pair i, j ,

$$\text{Exp}[\eta_{i,j}] = \sum_{u,v \in V, (u,v) \in E} (p_v^0 p_u^0 + p_v^1 p_u^1) = \sigma(G). \tag{3.2}$$

Since there are $K^2 = \Theta(n/d \cdot \text{poly}(\log n/\epsilon))$ such pairs i, j , the expected value of the sum over all $\eta_{i,j}$'s is greater than some constant $c' > c$. These random variables are not pairwise independent, nonetheless

we can obtain a constant bound on the probability that the sum is 0 using Chebyshev's inequality (cf., [11, Sec. 4.3]).

Unfortunately, we may not assume in general that for every (or even some) starting vertex, all (or even almost all) vertices are reached with probability $\Theta(1/n)$. However, roughly speaking, we are able to show that every graph can be partitioned into parts such that within each part we can perform an analysis that builds on the ideas presented above. Furthermore, the different parts are separated by small cuts so that if each part is close to being bipartite, then so is the whole graph. An important component in the analysis is the definition of the Markov Chain $M_{\ell_1}^{\ell_2}(H)$, and we turn to this definition in the next subsection.

3.2.2 The Markov Chain $M_{\ell_1}^{\ell_2}(H)$

Let H be an induced subgraph of G . For any given pair of lengths, ℓ_1 and ℓ_2 , we define a Markov Chain $M_{\ell_1}^{\ell_2}(H)$. Roughly speaking, $M_{\ell_1}^{\ell_2}(H)$ captures random walks of length at most $\ell_1 \cdot \ell_2$ in G that do not exit H for (sub)walks of length ℓ_2 or more. The states of the chain consist of the vertices of H and some additional auxiliary states. For vertices that do not have neighbors outside of H , the transition probabilities in $M_{\ell_1}^{\ell_2}(H)$ are exactly as in walks on G . However, for vertices v that have neighbors outside of H there are two modifications: (1) For each vertex u , the transition probability from v to u , denoted $q_{v,u}$, is the probability of a walk (in G) starting from v and ending at u after less than ℓ_2 steps (without passing through any other vertex in H). Thus, walks of length less than ℓ_2 out of H (and in particular the walk $v - u$ in case $(v, u) \in E$), are contracted into single transitions. Note that for every u and v in H we have $q_{u,v} = q_{v,u}$. (2) There is an auxiliary path of length ℓ_1 emitting from v . The transition probability from v to the first auxiliary vertex on the path equals the probability that a walk starting from v exits H and does not return in less than ℓ_2 steps. From the last vertex on the auxiliary path there are transitions to vertices in H with the corresponding conditional probabilities of reaching them after such a walk.

A more formal definition of $M_{\ell_1}^{\ell_2}(H)$ appears in the appendix, together with an illustration (see Figure A.1). The following definition and lemma will be instrumental in our analysis.

Definition 3.2.1 *We say that a vertex s is useful with respect to $M_{\ell_1}^{\ell_2}(H)$ if the probability that a walk in $M_{\ell_1}^{\ell_2}(H)$ starting from s enters an auxiliary path after at most ℓ_1 steps, is at most $\frac{2\ell_1}{\ell_2} \cdot \frac{n}{|H|}$.*

Lemma 3.2.1 *Let H be a subgraph of G , and let ℓ_1 and ℓ_2 be integers. Then at least half of the vertices s in H are useful with respect to $M_{\ell_1}^{\ell_2}(H)$.*

The proof of the lemma appears in [38].

3.2.3 Useful Vertices and Small Cuts

The following lemma can be viewed as presenting a ‘‘contrapositive statement’’ of the work of Mihail [62]. While Mihail showed that high expansion leads to fast convergence of random walks to the stationary distribution, the lemma below shows that too slow of a convergence implies small cuts that have certain additional properties. In particular, the vertices on one side of the cut can be reached with roughly the same, relatively high probability from some vertex s (where s need not necessarily be on the same side of the cut). In the special case where $H = G$ and G is rapidly mixing, the set S will be all of V , but in the general case it will be a subset of those vertices that are reached from s with probability that is not much smaller than that assigned by the stationary distribution (of $M_{\ell_1}^{\ell_2}(H)$).

For states x and y in $M_{\ell_1}^{\ell_2}(H)$ and an integer t , let $q_{x,y}(t)$ denote the probability that a random walk in $M_{\ell_1}^{\ell_2}(H)$ that starts at x , ends at y after t steps.

Lemma 3.2.2 *Let H be a subgraph of G with at least $\frac{\epsilon}{4}n$ vertices, and let $\ell_1 = \Theta\left(\left(\frac{\log(n/\epsilon)}{\epsilon}\right)^3\right)$, $\ell_2 = \Theta\left(\frac{\ell_1}{\epsilon^2}\right)$, and $F = O\left(\frac{1}{\epsilon}\right)$. Then for every vertex s that is useful with respect to $M_{\ell_1}^{\ell_2}(H)$, there exists a subset of vertices S in H , an integer t , $\ell_1/2 \leq t \leq \ell_1$, and a value $\beta = \Omega\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$, such that:*

1. *The number of edges between S and the rest of H is at most $\frac{\epsilon}{2} \cdot d \cdot |S|$.*
2. *For every $v \in S$,*

$$\sqrt{\frac{1}{|S|} \cdot \frac{\beta}{|H|}} \leq q_{s,v}(t) \leq F \cdot \sqrt{\frac{1}{|S|} \cdot \frac{\beta}{|H|}}.$$

The proof of the lemma appears in [38].

3.2.4 Sufficient Conditions for Good Partitions

In the next lemma we give sufficient conditions under which subsets of vertices can be partitioned without having many violating edges. For each $b \in \{0, 1\}$ let $q_{s,v}^b(t)$ denote the probability in $M_{\ell_1}^{\ell_2}(H)$ of a walk of length t starting from s , ending at v , and corresponding to a path whose length has parity b . What the lemma essentially requires is that for some fixed vertex s and subset of vertices S in H , there is a lower bound on the probability that each vertex in S is reached from s (in t steps), and there aren't too many vertices v in the subset such that both $q_{s,v}^0(t)$ and $q_{s,v}^1(t)$ are large (with respect to this lower bound).

Lemma 3.2.3 *Let H be a subgraph of G , s a vertex in H , S a subset of vertices in H and ℓ_1 and ℓ_2 integers. Assume that for some $\alpha > 0$, $t < \ell_1$, the following holds in $M_{\ell_1}^{\ell_2}(H)$:*

1. *For every $v \in S$, $q_{s,v}(t) \geq \alpha$;*
2. *$\sum_{v,u \in S, (v,u) \in E} (q_{s,v}^0(t)q_{s,u}^0(t) + q_{s,v}^1(t)q_{s,u}^1(t)) < \frac{\epsilon}{c} \cdot d \cdot |S| \cdot \alpha^2$ for some constant c .*

Let (S_0, S_1) be a partition of S , where $S_0 = \{v : q_{s,v}^0(t) \geq q_{s,v}^1(t)\}$, and $S_1 = \{v : q_{s,v}^1(t) > q_{s,v}^0(t)\}$. Then the number of violating edges in G with respect to (S_0, S_1) is at most $\frac{\epsilon}{c} \cdot d \cdot |S|$.

Proof: Consider a vertex v and let $v \in S_b$, for $b \in \{0, 1\}$. By definition of the partition (S_0, S_1) , $q_{s,v}^b(t) \geq \frac{1}{2}q_{s,v}(t) \geq \frac{\alpha}{2}$.

Assume, contrary to what is claimed in the lemma, that the number of violating edges with respect to (S_0, S_1) is more than $\frac{\epsilon}{c} \cdot d \cdot |S|$. Then

$$\begin{aligned} & \sum_{v,u \in S, (v,u) \in E} (q_{s,v}^0(t)q_{s,u}^0(t) + q_{s,v}^1(t)q_{s,u}^1(t)) \\ & \geq \sum_{v,u \in S, (v,u) \in E, u, v \in S_0} (q_{s,v}^0(t)q_{s,u}^0(t)) + \sum_{v,u \in S, (v,u) \in E, u, v \in S_1} (q_{s,v}^1(t)q_{s,u}^1(t)) \quad (3.3) \end{aligned}$$

$$\geq \sum_{v,u \in S, (v,u) \in E, u, v \in S_0} \frac{\alpha^2}{4} + \sum_{v,u \in S, (v,u) \in E, u, v \in S_1} \frac{\alpha^2}{4} \quad (3.4)$$

$$\geq \frac{\alpha^2}{4} \cdot \frac{\epsilon}{c} \cdot d \cdot |S|. \quad (3.5)$$

But this contradicts the second hypothesis of the lemma. ■

3.2.5 Sufficient Conditions for Detecting Odd Cycles

In the next lemma we describe sufficient conditions for “detecting” odd cycles when performing walks in $M_{\ell_1}^{\ell_2}(H)$ starting from some vertex s . What the lemma essentially requires is that there exists a subset S of vertices such that there are both lower and upper bounds on the probability that each vertex in S is reached from s (in $t < \ell_1$ steps), and there are many vertices v in S such that both $q_{s,v}^0(t)$ and $q_{s,v}^1(t)$ are large (with respect to the lower bound). As stated later in Corollary 3.2.4, these conditions are sufficient for detecting odd cycles when performing random walks in G of length $\ell_1 \cdot \ell_2$.

Lemma 3.2.4 *Let H be a subgraph of G , s a vertex in H , S a subset of vertices in H and ℓ_1 and ℓ_2 integers. Assume that for some $\alpha > 0$ and $F = \Theta(\frac{1}{\epsilon})$ and $t < \ell_1$, the following holds in $M_{\ell_1}^{\ell_2}(H)$:*

1. For every $v \in S$, $\alpha \leq q_{s,v}(t) \leq F \cdot \alpha$;
2. $\sum_{v,u \in S, (v,u) \in E} (q_{s,v}^0(t)q_{s,u}^0(t) + q_{s,v}^1(t)q_{s,u}^1(t)) \geq \frac{\epsilon}{c} \cdot d \cdot |S| \cdot \alpha^2$ for some constant c .

Suppose we perform $O\left(\frac{F^5}{\epsilon \cdot \alpha \sqrt{d} \sqrt{|S|}}\right)$ random walks of length t starting from s in $M_{\ell_1}^{\ell_2}(H)$. Let A_0 (A_1) be the set of vertices that appear at the end of the walks whose corresponding paths have even (odd) length, and let G_0 (G_1) be the subgraph induced by A_0 (A_1). Then with probability at least 0.99 (taken over the random walks), either G_0 contains an edge or G_1 contains an edge (i.e., the algorithm detects an odd cycle).

We note that when we apply Lemma 3.2.4, we set $\alpha = \text{poly}(\epsilon/(\log n))/\sqrt{|S| \cdot |H|}$, and $F = O(1/\epsilon)$, so that the number of random walks that should be performed is $O(\sqrt{n/d} \cdot \text{poly}((\log n)/\epsilon))$.

Proof: Let $\gamma \stackrel{\text{def}}{=} \sum_{v,u \in S, (v,u) \in E} (q_{s,v}^0(t)q_{s,u}^0(t) + q_{s,v}^1(t)q_{s,u}^1(t))$ so that by the second hypothesis of the lemma $\gamma \geq \frac{\epsilon}{c} \cdot d \cdot |S| \cdot \alpha^2$. Consider $m = O\left(\frac{F^5}{\epsilon \cdot \alpha \sqrt{d} \sqrt{|S|}}\right)$ random walks of length t starting from s . For $1 \leq i \neq j \leq m$, let $\eta_{i,j}$ be a 0/1 random variable that is 1 if and only if both the i -th and the j -th walk have path length with the same parity, and if the end-points of the paths are the vertices $u, v \in S$ such that $(u, v) \in E$.

Thus, we would like to bound the probability that $\sum_{i < j} \eta_{i,j} = 0$. The difficulty is that the $\eta_{i,j}$'s are not pairwise independent. Yet, since the sum of the covariances of the dependent $\eta_{i,j}$'s is quite small, Chebyshev's Inequality is still very useful (cf., [11, Sec. 4.3]). Details follow. For every $i \neq j$,

$$\text{Exp}[\eta_{i,j}] = \sum_{v,u \in S, (v,u) \in E} (q_{s,v}^0(t)q_{s,u}^0(t) + q_{s,v}^1(t)q_{s,u}^1(t)) = \gamma.$$

By Chebyshev's inequality,

$$\Pr \left[\sum_{i < j} \eta_{i,j} = 0 \right] \leq \frac{\text{Var} \left[\sum_{i < j} \eta_{i,j} \right]}{\left(\text{Exp} \left[\sum_{i < j} \eta_{i,j} \right] \right)^2} < \frac{\text{Var} \left[\sum_{i < j} \eta_{i,j} \right]}{\left(\binom{m}{2} \cdot \gamma \right)^2}. \quad (3.6)$$

We now bound $\text{Var}[\sum_{i < j} \eta_{i,j}]$. Since the $\eta_{i,j}$'s are not pairwise independent, some care is needed: Let $\bar{\eta}_{i,j} \stackrel{\text{def}}{=} \eta_{i,j} - \text{Exp}[\eta_{i,j}]$.

$$\text{Var} \left[\sum_{i < j} \eta_{i,j} \right] = \text{Exp} \left[\left(\sum_{i < j} \bar{\eta}_{i,j} \right)^2 \right]$$

$$\begin{aligned}
&= \sum_{i < j} \sum_{k < \ell} \text{Exp} [\bar{\eta}_{i,j} \cdot \bar{\eta}_{k,\ell}] \\
&= \sum_{i < j} \text{Exp} [\bar{\eta}_{i,j}^2] + 4 \sum_{i < j < k} \text{Exp} [\bar{\eta}_{i,j} \cdot \bar{\eta}_{j,k}] + 0 \\
&= \binom{m}{2} \cdot \text{Exp}[\bar{\eta}_{1,2}^2] + 4 \cdot \binom{m}{3} \cdot \text{Exp} [\bar{\eta}_{1,2} \cdot \bar{\eta}_{2,3}]. \tag{3.7}
\end{aligned}$$

The factor of 4 in the third equality is the number of possibilities that among the four elements i, j, k, ℓ (where $i < j$ and $k < \ell$) exactly two are equal (namely: $i = k < j < \ell$; $i < j = k < \ell$; $i < k < j = \ell$; and $k < i = \ell < j$). The 0 term is due to the fact that when i, j, k, ℓ are all distinct,

$$\begin{aligned}
\text{Exp} [\bar{\eta}_{i,j} \cdot \bar{\eta}_{k,\ell}] &= \text{Exp} [\eta_{i,j} \cdot \eta_{k,\ell}] - \gamma^2 \\
&= \sum_{i,j,k,\ell \in S, (i,j), (k,\ell) \in E} q_{s,i}^0(t) q_{s,j}^0(t) q_{s,k}^0(t) q_{s,\ell}^0(t) \\
&\quad + \sum_{i,j,k,\ell \in S, (i,j), (k,\ell) \in E} q_{s,i}^0(t) q_{s,j}^0(t) q_{s,k}^1(t) q_{s,\ell}^1(t) \\
&\quad + \sum_{i,j,k,\ell \in S, (i,j), (k,\ell) \in E} q_{s,i}^1(t) q_{s,j}^1(t) q_{s,k}^0(t) q_{s,\ell}^0(t) \\
&\quad + \sum_{i,j,k,\ell \in S, (i,j), (k,\ell) \in E} q_{s,i}^1(t) q_{s,j}^1(t) q_{s,k}^1(t) q_{s,\ell}^1(t) - \gamma^2 \\
&= \left(\sum_{(i,j) \in E(S)} q_{s,i}^0(t) q_{s,j}^0(t) + q_{s,i}^1(t) q_{s,j}^1(t) \right)^2 - \gamma^2 = \gamma^2 - \gamma^2 = 0. \tag{3.8}
\end{aligned}$$

We next bound each of the two terms in Equation (3.7).

$$\text{Exp}[\bar{\eta}_{1,2}^2] \leq \text{Exp}[\eta_{1,2}^2] = \text{Exp}[\eta_{1,2}] = \gamma. \tag{3.9}$$

Let v_i be a random variable that represents the vertex that the i -th walk ends at.

$$\begin{aligned}
\text{Exp}[\bar{\eta}_{1,2} \cdot \bar{\eta}_{2,3}] &\leq \text{Exp}[\eta_{1,2} \cdot \eta_{2,3}] \\
&\leq \sum_{v_1, v_2, v_3 \in S, (v_1, v_2), (v_2, v_3) \in E} q_{s,v_1}^0(t) q_{s,v_2}^0(t) q_{s,v_3}^0(t) + q_{s,v_1}^1(t) q_{s,v_2}^1(t) q_{s,v_3}^1(t) \\
&\leq (\text{number of pairs of edges in } S \text{ with a common vertex in } S) \cdot 2(\max_v \{q_{s,v}(t)\})^3 \\
&\leq 2 \cdot \min(|S|^2 d, |S| d^2) \cdot F^3 \cdot \alpha^3 \tag{3.10}
\end{aligned}$$

Since by the lemma's second hypothesis $\gamma \geq \frac{\epsilon}{c} \cdot d \cdot |S| \cdot \alpha^2$, we can replace α in Equation (3.10) with $\sqrt{\frac{c \cdot \gamma}{\epsilon \cdot d \cdot |S|}}$ and get

$$\text{Exp}[\bar{\eta}_{1,2} \cdot \bar{\eta}_{2,3}] \leq 2 \cdot \min(|S|^2 d, |S| d^2) \cdot F^3 \cdot \left(\frac{c \gamma}{\epsilon d |S|}\right)^{\frac{3}{2}} \tag{3.11}$$

Combining Equations (3.6)–(3.11) we get

$$\Pr \left[\sum_{i < j} \eta_{i,j} = 0 \right] = O \left(\frac{m^2 \cdot \gamma + m^3 \cdot \min(|S|^2 d, |S| d^2) \cdot F^3 \cdot \left(\frac{\gamma}{\epsilon d |S|}\right)^{\frac{3}{2}}}{m^4 \cdot \gamma^2} \right)$$

$$\begin{aligned}
&= O\left(\frac{1}{\gamma \cdot m^2} + \frac{F^3 \min(\sqrt{\frac{|S|}{d}}, \sqrt{\frac{d}{|S|}})}{m \cdot \epsilon^{\frac{3}{2}} \cdot \sqrt{\gamma}}\right) \\
&= O\left(\frac{\epsilon}{F^{10}} + \frac{1}{F^2 \cdot \epsilon}\right) = O(\epsilon)
\end{aligned} \tag{3.12}$$

As observed above, by the lemma's hypothesis concerning γ , it holds that $\alpha = O(\sqrt{\gamma/(\epsilon d|S|)})$. Since $m = \Omega\left(\frac{F^5}{\epsilon \cdot \alpha \cdot \sqrt{d} \sqrt{|S|}}\right)$, we have that $m = \Omega\left(F^5 \sqrt{\frac{1}{\epsilon \cdot \gamma}}\right)$, and the lemma follows. ■

Based on the construction of $M_{\ell_1}^{\ell_2}(H)$ we can map walks of length $\ell_1 \cdot \ell_2$ in G to walks of length ℓ_1 in $M_{\ell_1}^{\ell_2}(H)$, and obtain as a corollary to Lemma 3.2.4:

Corollary 3.2.4 *Let H be a subgraph of G and let S , s , ℓ_1 , ℓ_2 , t , α and F be as in Lemma 3.2.4. Suppose we perform $\Theta\left(\frac{F^5}{\epsilon \cdot \alpha \cdot \sqrt{d} \sqrt{|S|}}\right)$ random walks of length $\ell_1 \cdot \ell_2$ starting from s in G . Let A_0 (A_1) be the set of vertices that appear at the end of the walks whose corresponding paths have even (odd) length, and let G_0 (G_1) be the subgraph induced by A_0 (A_1). Then with probability at least 0.99, either G_0 contains an edge or G_1 contains an edge (i.e., the algorithm detects an odd cycle).*

The proof of the corollary is similar to that of an analogous corollary that appears in [38].

3.2.6 Proof of Theorem 3.2.1

Recall that we need to show that if the test accepts G with probability greater than $\frac{1}{3}$ then G is ϵ -close to being bipartite.

We say that a vertex s in G is *good* (for defining a partition) if the following holds. Suppose we take K random walks of length L in G starting from s . Then the probability that we reach two vertices u and v such that $(u, v) \in E$ and both u and v appear at the ends of walks whose corresponding paths have lengths with the same parity, is at most 0.1. If a vertex is not good then it is *bad*. Here K and L are set in the algorithm.

Since the test rejects G with probability less than $\frac{2}{3}$, and $T = \Theta(1/\epsilon)$, we know that, for an appropriate constant in the $\Theta(\cdot)$ notation above, the fraction of bad vertices in G is at most $\frac{\epsilon}{16}$. We now show that in such a case we can find a partition of the graph vertices that has at most ϵdn violating edges. We shall do so in steps, where in each step we partition a new set of vertices, denoted S , until we are left with at most $\frac{\epsilon}{4}n$ vertices. For each partitioned set S we show that: (1) there are few (at most $\frac{\epsilon}{4}d|S|$) violating edges with respect to the partition of S ; and (2) there are few (at most $\frac{\epsilon}{2}d|S|$) edges between S and the yet "unpartitioned" vertices R so that no matter how the vertices in R are partitioned, the number of violating edges between S and R is small.

At each step, let D be the set of vertices we have already partitioned, and let H be the subgraph induced by $V \setminus D$. Initially, $D = \emptyset$, and $H = G$. Let ℓ_1 and ℓ_2 be as required by Lemma 3.2.2, and let the length L of the walks we perform on G be $\ell_1 \cdot \ell_2$. Since $\ell_1 = O\left(\left(\frac{\log(n/\epsilon)}{\epsilon}\right)^3\right)$, and $\ell_2 = O\left(\frac{\ell_1}{\epsilon^2}\right)$, we get that $L = O\left(\frac{\log^6(n/\epsilon)}{\epsilon^8}\right)$. Let $M \stackrel{\text{def}}{=} M_{\ell_1}^{\ell_2}(H)$. While $|H| \geq \frac{\epsilon}{4}n$ we do the following. We select any vertex s in H that is both *good* and *useful* with respect to M (see Definition 3.2.1). By Lemma 3.2.1, at least half of the vertices in H are *useful*. Since $|H| \geq \frac{\epsilon}{4}n$ and the total number of *bad* vertices is $\frac{\epsilon}{16}n < \frac{\epsilon}{8}n$, there exist at least $\frac{\epsilon}{16}n$ vertices which are *good* and *useful*.

We next apply Lemma 3.2.2 to determine a set S , and an integer t , $\ell_1/2 \leq t \leq \ell_1$, with the properties stated in the lemma. In particular, the number of edges between S and the rest of H is at most $\frac{\epsilon}{2}d|S|$, and for every $v \in S$, $\sqrt{\frac{\beta}{|S| \cdot |H|}} \leq q_{s,v}(t) \leq F \cdot \sqrt{\frac{\beta}{|S| \cdot |H|}}$, where $F = O\left(\frac{1}{\epsilon}\right)$, and $\beta = \Omega\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$. We claim

that it must be the case that $\sum_{v,u \in V, (v,u) \in E} (p_v^0(t)p_u^0(t) + p_v^1(t)p_u^1(t)) \leq \frac{\epsilon \cdot \beta \cdot d}{|H|}$. This claim, (which we establish momentarily) implies that we can apply Lemma 3.2.3 (with $\alpha = \sqrt{\frac{\beta}{|S| \cdot |H|}}$) to show that S can be partitioned so that there are at most $\frac{\epsilon}{4}d|S|$ violating edges with respect to this partition. The claim holds since otherwise we could apply Corollary 3.2.4 and reach a contradiction. Specifically, by letting the number of walks performed from each starting vertex be

$$O\left(\frac{F^5}{\epsilon \cdot \alpha \cdot \sqrt{d} \cdot \sqrt{|S|}}\right) = O\left(\frac{\sqrt{|H|}}{\epsilon^6 \cdot \sqrt{d} \cdot \sqrt{\beta}}\right) = O\left(\frac{\log^{1/2}(n/\epsilon) \cdot \sqrt{n/d}}{\epsilon^8}\right) = K$$

(where F , α and β are as set above), we would obtain a contradiction to our assumption that s is good.

Thus, as long as $|H| \geq \frac{\epsilon}{4}n$, each set S contributed at most $\frac{\epsilon}{4} \cdot |S| \cdot d + \frac{\epsilon}{2} \cdot |S| \cdot d$ violating edges to the partition. Since these sets are disjoint, all these violating edges sum up to $\frac{3\epsilon}{4} \cdot d \cdot n$. The final H contributes at most $\frac{\epsilon}{4} \cdot n \cdot d$, and so G is ϵ -close to being bipartite.

Verifying that indeed $T = O(1/\epsilon)$, $K = \Theta(\sqrt{n/d} \cdot \text{poly}(\log n/\epsilon))$, and $L = \text{poly}((\log n)/\epsilon)$, and that the algorithm can be implemented using $O(K \cdot L + K^2) = O(n/d \cdot \text{poly}(\log n/\epsilon))$ queries, the theorem follows. (Recall that if $d < \sqrt{n}$ then we obtain the bound of $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$.)

3.3 The Algorithm for the General Case

In this section we build on the testing algorithm presented in the previous section and describe a one-sided error testing algorithm for bipartiteness that works with respect to the actual number of edges $m = m(G)$. Hence this algorithm is suitable for general graphs (for which d_{\max} may vary significantly from d_{avg}). The query complexity and running time of the algorithm are of the same order of magnitude as for Test-Bipartite-Reg, that is, $O(\min(\sqrt{n}, n^2/m) \cdot \text{poly}(\log n/\epsilon))$. We note that once the graph becomes very dense, that is $m = \Omega(n^2/\log^c n)$ (where c is approximately 4), it is preferable to use the adjacency-matrix model algorithm [37, 6] with distance parameter $\epsilon/(n^2/m)$.

A High Level Description of the Algorithm. The basic idea is to reduce the problem of testing with respect to the actual number of edges m to the problem of testing with respect to the upper bound $m_{\max} = d_{\max} \cdot n$. Specifically, for any graph G we show how to define a graph G' over $\Theta(n)$ vertices that has the following useful properties. First, the maximum degree in G' is roughly the same as the average degree, and furthermore, this degree is roughly the same as the average degree in G . In particular this implies that the two graphs have roughly the same number of edges. Second, G' approximately preserves the distance of G to bipartiteness. More precisely, if G is bipartite then so is G' , but if G is far from being bipartite with respect to $m(G)$, then G' is far from being bipartite with respect to $m_{\max} = d_{\max}(G')n'$. Thus G' can be viewed as a kind of “regularized-degree version” of G .

If we had direct access to G' , then by the above we would be done: by running the algorithm Test-Bipartite-Reg on G' we could decide whether G is bipartite or far from being bipartite. However, we only have access to G . Nonetheless, given query access to G we can efficiently “emulate” queries in G' . This would almost suffice for running Test-Bipartite-Reg on G' . One more issue is the uniform selection of starting vertices in G' , required by Test-Bipartite-Reg. As we shall see, selecting a vertex uniformly from G' is (roughly) equivalent to uniformly selecting an edge in G .

While we do not know how to efficiently select a vertex in G' uniformly, we describe a different selection procedure that suffices for our purposes. Specifically, the selection procedure is such that for all but a small fraction of the n' vertices in G' , the probability of selecting a vertex v is $\Omega(1/n')$. With slight abuse of termi-

nology we shall refer to this procedure as Sample-Vertices-Almost-Uniformly-in-G'.² By Corollary 3.2.3, this suffices for our purposes.

Multiple Edges and the Relation Between m and n . The analysis of the algorithm Test-Bipartite-Reg did not require any assumptions on the actual number of edges m in the graph, and it did not preclude the existence of multiple edges. Here we consider graphs that do not contain any multiple edges and we assume that the number of edges m in G is $\Omega(n)$. To justify the assumption on the number of edges, consider a graph that consists of a clique over $k = o(\sqrt{n})$ vertices, where all remaining vertices are isolated. This graph has $m = \Theta(k^2)$ edges, and is clearly far from being bipartite. However, in order to distinguish it from a graph that consists of a complete bipartite graph over $2k$ vertices (where all remaining vertices are isolated and is clearly bipartite), we need $\Omega(n/k) = \omega(\sqrt{N})$ queries. (Taking this to an extreme, if $k = \Theta(1)$ then we will need $\Omega(n)$ queries.) We note that we could replace this assumption by introducing to the complexity of the algorithm a dependence on n/m . This would however make the analysis more cumbersome, without much benefit.

Another alternative assumption would be that the algorithm has the ability to “ignore” isolated vertices (that is, vertices that have no incident edges and are hence immaterial to the question of bipartiteness), and sample uniformly from the *non-isolated* vertices. This would effectively imply that the algorithm is executed on a subgraph induced by the $n' \leq n$ non-isolated vertices, where within this subgraph, the number of edges $m' = m$, is at least $n'/2$.

For simplicity we assume from this point on that $m \geq n$.

We also note that we can actually deal with the case where there are multiple edges, but they do not constitute more than a constant fraction of the total number of edges.³ However, in order to deal with this case efficiently, we need to assume that there is a concise way to represent the sets of labels of multiple edges that are incident to each vertex. (In particular this holds if the labels of multiple edges incident to each vertex are consecutive). For simplicity we assume there are no multiple edges.

The main theorem of this subsection follows.

Theorem 3.3.1 *For every graph G having n vertices and $m \geq n$ edges, we can define a graph G' having n' vertices and m' edges for which the following holds:*

1. $n \leq n' \leq 4n$, $m \leq m' \leq 8m$, and $d_{\max}(G') \leq 2d_{\text{avg}}(G)$.
2. If G is bipartite then G' is bipartite, and if G is ϵ -far from being bipartite with respect to m , then G' is ϵ' -far from being bipartite with respect to $m_{\max}(G') = d_{\max}(G')n'$ for $\epsilon' = \Theta(\epsilon)$.
3. Given a starting vertices s in G' , it is possible to emulate random walks in G' starting from s , by performing queries to G . The amortized cost of each random walk step is $O(\log^2 n)$ (degree and neighbor) queries in G . By emulating these random walks it is possible to execute a slight variant of Odd-Cycle(s) in G' which we denote Odd-Cycle'(s). This variant is such that $\Pr[\text{Odd-Cycle}'(s)=\text{found}] \geq \Pr[\text{Odd-Cycle}(s)=\text{found}]$, where if Odd-Cycle'(s) returns found, then we can obtain an odd cycle of length $\text{poly}(\log n/\epsilon)$ in the original graph G .

²The reason we say that we abuse terminology is that the distribution on vertices in G' induced by this procedure may be very far from uniform according to any standard distance measure (e.g. statistical difference). However, it approximates the uniform distribution in the sense of assigning relatively large weight to every sufficiently large subset.

³If the number of multiple edges is more than a constant fraction then it is possible to obtain a lower bound on the number of queries that depends on the ratio between the number of multiple edges and the total number of edges. Specifically, consider a graph that contains a small clique with many multiple edges, which is far from bipartite, but cannot be distinguished from a bipartite graph that contains a small complete bipartite graph with many multiple edges.

4. There exists a procedure `Sample-Vertices-Almost-Uniformly-in-G'` that for any given parameter $0 < \delta \leq 1$, performs $\tilde{O}(\min(\sqrt{n/\delta}, n^2/m))$ queries in G and returns a vertex in G' such that the following holds: For all but at most $\delta n'$ of the vertices x in G' , the probability that x is selected by the procedure is $\Omega(1/n')$.

We note that for every graph G there is actually a family of graphs G' with the above properties (all defined over the same set of vertices). When we run algorithm `Test-Bipartite-Gen`, we construct one such (arbitrary) graph G' in the family as we go along. One difficulty that arises is that when the algorithm asks a neighbor query of the form: "Who is the i -th neighbor of v ", it gets a vertex name u as an answer. However, the algorithm lacks the information that v is (say) the j -th neighbor of u . This lack of knowledge makes the emulation of random walks and `Odd-Cycle(s)` in G' more complicated. Due to that, Item 3 of Theorem 3.3.1 is somewhat more involved.

As a corollary to Theorem 3.3.1 and Corollary 3.2.3 we obtain:

Corollary 3.3.2 *Algorithm `Test-Bipartite-Gen` (see Figure 3.3) accepts every graph G that is bipartite, and rejects with probability at least $2/3$ every graph G that is ϵ -far from being bipartite (with respect to $m(G)$). Furthermore, whenever the algorithm rejects a graph it outputs a certificate to the non-bipartiteness of the graph G in form of an odd cycle of length $\text{poly}(\log n/\epsilon)$.*

The query complexity and running time of the algorithm are $O(\min(\sqrt{n}, n^2/m) \cdot \text{poly}(\log n/\epsilon))$.

Test-Bipartite-Gen($n, d_{\text{avg}}, \epsilon$)

- Repeat $T = \Theta(\frac{1}{\epsilon})$ times:
 1. Set $\epsilon' = \epsilon/144$.
 2. Select a vertex s in G' by calling the procedure `Sample-Vertices-Almost-Uniformly-in-G'` with $\delta = \epsilon'/c$ (where c is a sufficiently large constant).
 3. Apply `Odd-Cycle'(s)`.
 4. If `Odd-Cycle'(s)` returns found then output reject.
- In case no call to `Odd-Cycle'` returned found then output accept.

Figure 3.3: Algorithm `Test-Bipartite-Gen` for testing bipartiteness with respect to the actual number of edges $m = m(G)$ in the graph G .

Note that d_{avg} , the average degree of the graph, is given as a parameter to the algorithm. Since the algorithm does not actually need the exact value of d_{avg} it can instead estimate it. Specifically, Feige [32] shows how to obtain an estimate that is within a factor of roughly 2 of the true value by performing at most $O(\sqrt{n/d_0})$ degree queries where d_0 is an a priori lower bound on d_{avg} . Inspired by our procedure `Sample-Edges-Almost-Uniformly-in-G'`, Goldreich and Ron [40] suggest an alternative procedure that improves on the quality of the estimate. More importantly in our context, they observe that both in the case of their procedure and in the case of Feige's procedure, it is possible to eliminate the need of the lower bound d_0 . That is, given Feige's algorithm, it is possible to obtain a constant factor estimate by performing $O(\sqrt{n/d_{\text{avg}}}) \leq O(\min(\sqrt{n}, n^2/m))$ queries (but without any knowledge about d_{avg}).

We now turn to proving Theorem 3.3.1.

3.3.1 Defining G' and Proving the First Two Items in Theorem 3.3.1

In all that follows, let $d = d_{\text{avg}}(G)$, and let $d' = d_{\text{max}}(G')$. We shall assume that d is a sufficiently large constant ($d \geq 1$). If $d_{\text{avg}}(G)$ is not sufficiently large then we still set d in the construction below to be

sufficiently large, and run the algorithm with ϵ set to $\epsilon/(d/d_{\text{avg}}(G))$.

The Idea. Recall that part of our goal is to have G' be a “regularized” version of G in the sense that in G' all vertices have degree at most $2d$ (while in G there may be vertices with degree much higher than the average degree d). To this end, every vertex of G with degree higher than d is represented in G' by a subset of vertices. Each such subset is partitioned into two equal parts: an *external* subset (consisting of *external* vertices), and an *internal* subset (consisting of *internal* vertices). The edges between external vertices in G' are determined by the edges of G . Namely, if (u, v) is an edge in G , then in G' there is an edge between one of the vertices in the external subset of u to one of the vertices in the external subset of v . In addition, for every vertex v (with degree greater than d) there is a bipartite subgraph between its internal and external vertices. All vertices in the subgraph have degree d , and the subgraph has good expansion properties.

The role of these subgraphs between external and internal vertices is to ensure that if G is far from being bipartite then so is G' . To gain some intuition observe that for every partition (V_1, V_2) of G , there exists a corresponding partition (V'_1, V'_2) of G' that has the same number of violating edges. Specifically, for every vertex $v \in V_1$ ($v \in V_2$) we put in V'_1 (V'_2) all external vertices that correspond to v , and we put in V'_2 (V'_1) all internal vertices that correspond to v . By this construction, there is a one-to-one mapping between the edges in G that are violating with respect to (V_1, V_2) and the edges in G' that are violating with respect to (V'_1, V'_2) (where all edges in the subgraphs between external and internal vertices are non-violating). Thus, if G is far from being bipartite, so that all partitions (V_1, V_2) of G have many violating edges, then this is also true of all partitions (V'_1, V'_2) as defined above. However, there are other partitions of G' that may split the external vertices that correspond to the same vertex in G into different parts and possibly have fewer violating edges. What we show is that such splits must introduce violating edges in the subgraphs between external and internal edges, where this is due to the expansion properties of the subgraphs. Roughly speaking, if a partition “avoids violations” between external vertices by splitting sets of external vertices, then it “pays” by introducing violations between external and internal vertices.

The Construction of G'

For each vertex v in G such that $\deg(v) \leq d$, we have a single vertex in G' . For each vertex v in G such that $\deg(v) > d$ the graph G' contains a subgraph, denoted $H(v)$. It is a bipartite graph over two subsets of vertices, one denoted $X(v)$, the *external* part, and one denoted $I(v)$, the *internal* part. Both parts consist of $\lceil \deg(v)/d \rceil$ vertices. Every vertex in $X(v)$ is assigned up to d specific neighbors of v according to some fixed, *but arbitrary* partition of the neighbors of v . As we shall see below, this assignment determines the edges between pairs of external vertices in G' that correspond to different vertices in G . We refer to the vertices in the two subsets by $\{X_i(v)\}_{i=1}^{\lceil \deg(v)/d \rceil}$ and $\{I_i(v)\}_{i=1}^{\lceil \deg(v)/d \rceil}$, respectively.

The edges in $H(v)$ are determined as follows. In case $\deg(v)/d < d$ then we have $\lfloor d^2/\deg(v) \rfloor$ -multiple edges between every internal vertex and every external vertex in $H(v)$. It follows that the degree of every vertex within $H(v)$ is

$$\frac{\lfloor d^2/\deg(v) \rfloor}{2} \cdot \lceil \deg(v)/d \rceil \leq \frac{d^2}{2 \cdot \deg(v)} \cdot \frac{\deg(v) + d}{d} \leq \frac{d^2}{2 \cdot \deg(v)} \cdot \frac{2\deg(v)}{d} = d$$

In case $\deg(v)/d \geq d$, denote $s = \lceil \deg(v)/d \rceil$ and let $H(v)$ be a bipartite expander where each of its sides has s vertices ($s \geq d$). Each vertex in $H(v)$ has degree d . All eigenvalues of the adjacency matrix of H , but the largest one and the smallest one (which are equal to d and $-d$, respectively), are at most $d/4$ in their absolute values. Explicit constructions of such expanders can be found, e.g., in [59, 57]. Furthermore, these constructions allow the determination of the i -th neighbor of any given vertex in constant time.

For sake of the presentation, when $\deg(v) \leq d$, so that v is represented by a single vertex, we let $H(v)$ be the subgraph that consists of this single vertex. This vertex is considered an external vertex, denoted $X_1(v)$, and it is assigned all neighbors of v .

We have described how vertices of G are transformed into vertices of G' (some of which are connected by edges). It remains to describe the relevant transformation to the edges of G . Consider an edge $(u, v) \in E(G)$ where v is the i -th neighbor of u and u is the j -th neighbor of v . Let $X_k(u)$ and $X_\ell(v)$ be the external vertices that are assigned the i -th neighbor of u , and the j -th neighbor of v , respectively. Then, there is an edge $(X_k(u), X_\ell(v))$ in G' . It directly follows that every vertex in G' has degree at most $2d$ and that $n' = |V(G')| \leq \sum_{v \in G} 2\lceil \deg(v)/d \rceil \leq 4n$, and $m' = m(G') \leq 4dn = 8m$.

For an illustration of the construction of G' , see Figure 3.4.

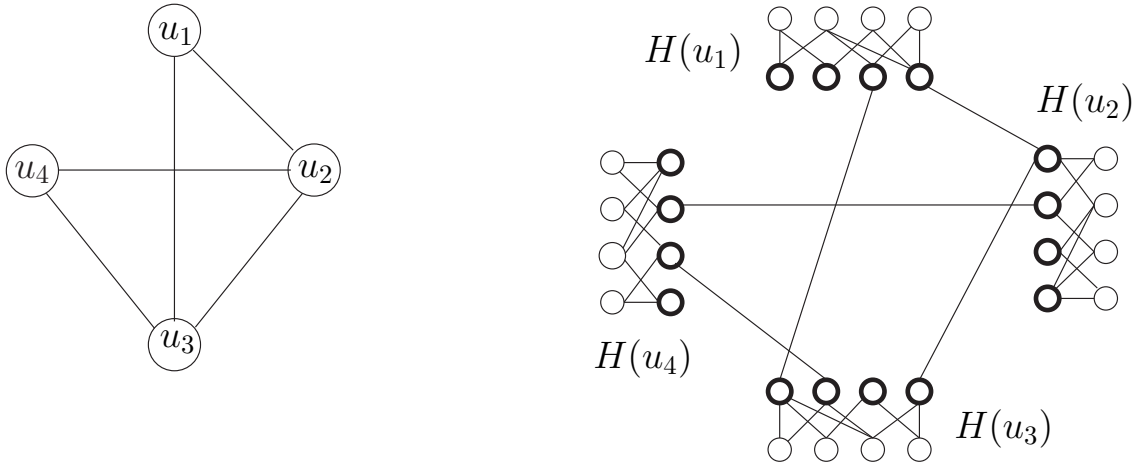


Figure 3.4: An illustration for the construction of G' . On the left are 4 vertices in G and their induced subgraph. On the right are the 4 corresponding subgraphs in G' and the edges between the external vertices in these subgraphs. (The external vertices are marked in bold, and there are additional edges that do not appear in the figure, between the external vertices in the figure and external vertices of other subgraphs.)

We have thus established the first item in Theorem 3.3.1, and we turn to the second item. From the construction of G' it is obvious that if G is bipartite, then so is G' . It remains to prove the following lemma.

Lemma 3.3.1 *If G is ϵ -far from being bipartite (with respect to $m = (dn)/2$) then G' is ϵ' -far from being bipartite with respect to $d'n'$, for $\epsilon' = \frac{\epsilon}{144}$.*

In order to prove Lemma 3.3.1, we first prove the following proposition concerning bipartite expander graphs. For any two (disjoint) subsets of vertices, A and B , we let $e(A, B)$ denote the number of edges with one end-point in A and another in B .

Proposition 3.3.3 *Let $G = (A \cup B, E)$ be a d -regular bipartite graph with sides A and B of size s . Assume that all eigenvalues of the adjacency matrix of G , but the largest one and the smallest one, are at most λ in their absolute values. Assume further that $\lambda \leq d/4$. Then for every two partitions $A = A_1 \cup A_2$, $B = B_1 \cup B_2$, satisfying $|A_1| \geq s/2$,*

$$e(A_1, B_1) + e(A_2, B_2) \geq \frac{d|A_2|}{8}.$$

Proof: It is well known that the larger is the “spectral gap” (i.e., the difference between d and λ), the closer the edge distribution in G approaches that of a truly random bipartite graph with sides of size s and edge probability d/s . Specifically, for every $A_0 \subseteq A$, $B_0 \subseteq B$ of sizes $|A_0| = a_0$, $|B_0| = b_0$,

$$\left| e(A_0, B_0) - \frac{da_0b_0}{s} \right| \leq \lambda\sqrt{a_0b_0} \quad (3.13)$$

(see, e.g., Chapter 9 of [11]).

Let $|A_1| = a_1$, $|A_2| = a_2 = s - a_1$, $|B_1| = b_1$, $|B_2| = b_2 = s - b_1$. It is given that $a_1 \geq s/2$. We may obviously assume that $b_1 \geq a_2/2$, as otherwise at least half of the edges incident to A_2 have their other endpoint outside B_1 , implying $e(A_2, B_2) \geq da_2/2$.

Applying the bound in Equation (3.13) twice we get:

$$e(A_1, B_1) = d|B_1| - e(A_2, B_1) \geq db_1 - \frac{da_2b_1}{s} - \lambda\sqrt{a_2b_1} = \frac{da_1b_1}{s} - \lambda\sqrt{a_2b_1}, \quad (3.14)$$

$$e(A_2, B_2) = d|A_2| - e(A_2, B_1) \geq da_2 - \frac{da_2b_1}{s} - \lambda\sqrt{a_2b_1} = \frac{da_2b_2}{s} - \lambda\sqrt{a_2b_1}. \quad (3.15)$$

Consider first the case $b_2 \leq s/2$. In this case it follows from Equation (3.14) that

$$\begin{aligned} e(A_1, B_1) &\geq \frac{da_1b_1}{s} - \lambda\sqrt{a_2b_1} \geq \frac{d(s/2)(s/2)}{s} - \lambda\sqrt{(s/2)s} \\ &= \frac{ds}{4} - \frac{\lambda s}{\sqrt{2}} \geq \frac{ds}{14} \geq \frac{da_2}{7}. \end{aligned}$$

We thus assume that $b_2 > s/2$. If $da_2b_2/s \geq 2\lambda\sqrt{a_2b_1}$, we obtain from Equation (3.15) that

$$e(A_2, B_2) \geq \frac{da_2b_2}{2s} \geq \frac{da_2(s/2)}{2s} = \frac{da_2}{4}.$$

Hence we may assume that $da_2b_2/s \leq 2\lambda\sqrt{a_2b_1}$. If $da_1b_1/s \geq 2\lambda\sqrt{a_2b_1}$, then it follows from Equation (3.14) that

$$e(A_1, B_1) \geq \frac{da_1b_1}{2s} \geq \frac{d(s/2)(a_2/2)}{2s} = \frac{da_2}{8},$$

as required. Hence we may assume that $da_1b_1/s \leq 2\lambda\sqrt{a_2b_1}$. It remains to check that the latter assumption together with $da_2b_2/s \leq 2\lambda\sqrt{a_2b_1}$ bring to a contradiction. Indeed, multiplying these inequalities we get: $d^2a_1a_2b_1b_2/s^2 \leq 4\lambda^2a_2b_1$, or $d^2a_1b_2 \leq 4\lambda^2s^2$. Recalling that $a_1 \geq s/2$, $b_2 > s/2$, it follows that $d < 4\lambda$ – a contradiction to our assumption on λ . ■

Proof of Lemma 3.3.1: We shall show that the number of edges that should be removed from G' so as to make it bipartite, is at most a constant factor smaller than the number of edges that should be removed from G so as to make G bipartite. Since the total number of edges in G and in G' is of the same order, this suffices to prove the lemma. To this end we prove the contrapositive statement. Specifically, suppose G' is ϵ' -close to being bipartite with respect to $m_{\max} = d'n'$. Namely, there exists a partition $P' = (V'_0, V'_1)$ of the vertices in G' with respect to which there are at most $\epsilon' \cdot d'n'$ violating edges in G' . We shall show how to construct, based on P' , a partition $P = (V_0, V_1)$ of the vertices in G with respect to which there are at most $\epsilon dn/2 = \epsilon m$ violating edges in G , thus proving the lemma.

Consider a particular vertex v in G , and the subset of external vertices $X(v)$ in G' that correspond to v . Let $X^0(v) = X(v) \cap V'_0$, and let $X^1(v) = X(v) \cap V'_1$. We refer to the larger subset as the *majority subset* of v , and to the smaller subset as the *minority subset* of v . We define $P = (V_0, V_1)$ by assigning each vertex v in G according to its majority subset. Namely, if $|X^0(v)| \geq |X^1(v)|$ then v is assigned to V_0 , otherwise it is assigned to V_1 . In what follows it will be convenient to refer to 0 and 1 as *colors*.

Also, when we refer to edges in G' as violating edges, we mean with respect to P' , and when we refer to edges in G as violating edges, we mean with respect to P . Note that the partition P is defined only according to the coloring of the external vertices in G' , ignoring the coloring of the internal vertices. Also recall that there is a one-to-one mapping between edges in G and edges in G' whose end-points are both external vertices.

Since each vertex v in G is assigned the color of its majority subset, the violating edges in G' between pairs of vertices that both belong to majority subsets, or between pairs of vertices that both belong to minority subsets, be violating edges in G . Similarly, non-violating edges in G' between vertices in majority subsets, or between vertices in minority subsets, become non-violating edges in G . It remains to deal with edges between minority and majority subsets in G' . These edges can be non-violating in G' , but may become violating in G .

We next show that the total number of vertices in G' that belong to minority subsets can be bounded as a function of the number of violating edges in G' . To this end we show that if there were many minority vertices, then there would be many violating edges in G' *between internal and external vertices*.

For each vertex v in G , consider the majority and minority subsets of (the external vertices of) v . Let the majority subset of $X(v)$ be $X^\alpha(v)$ and let the minority subset be $X^\beta(v)$ (where $\alpha, \beta \in \{0, 1\}$).

Claim 3.3.1.1 *For every vertex v in G , the number of violating edges in G' between vertices in $X(v)$ and vertices in $I(v)$ is at least $|X^\beta(v)| \cdot (d/8)$.*

Proof: Similarly to our notation for external vertices, for the internal vertices of v let $I^0(v) \stackrel{\text{def}}{=} I(v) \cap V'_0$ and $I^1(v) \stackrel{\text{def}}{=} I(v) \cap V'_1$. Consider first the case $\frac{\deg(v)}{d} < d$. By construction of G' , $|X(v)| = |I(v)| = \lceil \deg(v)/d \rceil$, and there are $\lfloor \frac{d^2/\deg(v)}{2} \rfloor$ multiple edges between every pair of vertices (x, y) such that $x \in X(v)$ and $y \in I(v)$. Hence the number of edges between $X(v)$ and $I(v)$ that are violating (with respect to P') is

$$\begin{aligned} & \left(|X^\alpha(v)||I^\alpha(v)| + |X^\beta(v)||I^\beta(v)| \right) \cdot \frac{\lfloor d^2/\deg(v) \rfloor}{2} \\ & \geq \left(|X^\beta(v)||I^\alpha(v)| + |X^\beta(v)||I^\beta(v)| \right) \cdot \frac{d^2}{4\deg(v)} = |X^\beta(v)| \cdot \frac{d}{4}. \end{aligned}$$

Next consider the more interesting case where $\frac{\deg(v)}{d} \geq d$. In this case Claim 3.3.1.1 directly follows from Proposition 3.3.3. \diamond (Proof of Claim 3.3.1.1.)

Thus we can conclude that if there are w external vertices that belong to minority subsets then they contribute at least $w \cdot d/8$ violating edges in G' . Since the number of violating edges in G' is at most $\epsilon' n' d' \leq 8\epsilon' nd$, we have that $w \leq 64\epsilon' n$. As noted previously, the total number of violating edges in G is upper bounded by the number of violating edges in G' plus the number of edges between minority and majority (external) subsets. By the above discussion and the fact that every external vertex has at most d neighbors that are external vertices, the number of violating edges in G is at most $8\epsilon' nd + 64\epsilon' nd = 72\epsilon' nd$. Since $\epsilon' = \epsilon/144$, and $m = (nd)/2$, the lemma follows. \blacksquare

We have completed proving the first two items of Theorem 3.3.1, and we now turn to the third item.

3.3.2 Establishing Item 3 in Theorem 3.3.1

Here we stress that if the neighbor and the vertex-pair queries would have returned more information, then the proof of the current item would be significantly simpler. In particular, suppose that a neighbor query (u, i) is answered with a pair (v, j) (instead of only v), which means that v is the i -th neighbor of u , and u is the j -th neighbor of v . Suppose also that when performing a vertex-pair query (u, v) , the algorithm is not only told whether (u, v) is an edge or not, but rather, in the former case it is also provided with pair

(i, j) , which means that v is the i -th neighbor of u , and u is the j -th neighbor of v . With this additional information, the structure of G' is implicitly given and, thus, the emulation of random walks in the execution of the procedure Odd-Cycle on G' , is straightforward. Here we need to work harder to overcome the lack of information.

Random-Walk Steps. We first shortly discuss the emulation of random walks. If the walk stays at the current vertex, then clearly there is no need for any emulation. Hence, we only need to consider the case in which we have to select a random neighbor. Recall that vertices in G' are either of the form $X_i(v)$ (the i -th external vertex corresponding to vertex v in G), or $I_i(v)$ (the i -th internal vertex), where $1 \leq i \leq \lceil \deg(v)/d \rceil$. Recall that we can obtain $\deg(v)$ for any v by a single degree query, and in particular use this to find the degree of vertices in G' . For simplicity of the presentation, we assume from this point on that for every vertex v in G , the degree of every vertex $I_i(v)$ in G' is d (instead of being at most d), and the degree of every $X_i(v)$ is $2d$ (instead of being at most $2d$).

Performing a random-walk step in G' from an internal vertex $I_i(v)$ can be easily done by using the explicit structure of the graph $H(v)$ (which is either a complete bipartite graph with multiple edges, or an explicitly constructible expander). In order to perform a random-walk step from an external vertex, $X_i(v)$, we first determine whether to take one of the d edges within the graph $H(v)$, or whether to take one of the d edges going from $X_i(v)$ to another external vertex. In the first case we then select an internal neighbor given the explicit structure of $H(v)$. It remains to deal with selecting an external neighbor. Note that in the special, but easy, case in which $\deg(v) \leq d$ and so $H(v)$ is a single vertex, there is only the latter option.

As noted just following the statement of Theorem 3.3.1, we actually construct G' as we go along. The important thing to note is that the definition of G' allows us to assign the vertices in $X(v)$ edges of v in an *arbitrary* manner (as long as each $X_i(v)$ is assigned (at most d) different edges). Hence, all we need to take care of is to be consistent with previous choices, and to ensure the correct distribution in the choice of the random walk step. To this end we may think of each external vertex as having d “ports”, labeled $1, \dots, d$, which are initially unassigned. As the algorithm proceeds, it puts a “link” between, say, the t 'th port of $X_i(v)$ and the ℓ 'th port of $X_j(u)$ (where $(v, u) \in E(G)$). When performing a random-walk step from $X_i(v)$ (to a vertex outside of $H(v)$), we uniformly select a port. Let us denote the index of the port selected by t . If port t of $X_i(v)$ is already linked to another port, then we simply take this link to the port (and vertex) at the other end. Otherwise, we first set the link, and then take it. In order to set the link properly, we need to uniformly select a neighbor u of v among the neighbors of v that were not yet assigned (to any port of one of the external vertices of v). After doing so, and selecting a neighbor u , we need to select a yet unassigned port of one of the external vertices of u . The above can be done, with the aid of sampling, at an amortized cost of $O(\log n)$ queries in G . Details follow.

For each external vertex $X_i(v)$, the algorithm keeps a vector of length d , $\Gamma_i(v)$. The k 'th entry of $\Gamma_i(v)$ contains the k 'th neighbor of $X_i(v)$, if it was determined, and is empty otherwise. Denote by $f_i(v)$ the number of free entries in the vector $\Gamma_i(v)$. Also denote by $A(v)$ the set of neighbors of v that were already assigned to external vertices of v , and by $NA(v)$ the vertices that were not assigned. When setting a link to a yet-unassigned port (filling in a new entry in $\Gamma_i(v)$), we distinguish between two cases.

Case 1: If less than half of the neighbors of v in G are in $A(v)$, the algorithm repeats at most $(\log^2 n)$ times the following procedure: It chooses uniformly at random a neighbor of v in G . If that neighbor belongs to $NA(v)$, then a desired neighbor is found. By repeating the above procedure $O(\log^2 n)$ times, the probability that the algorithm didn't find a desired neighbor is at most $o(1/n)$. In this case we say that the algorithm fails. Since the total number of queries that the algorithm performs is at most $o(n)$, the total failure probability of the algorithm is $o(1)$. Suppose that a desired neighbor of v , with the name u is found. In that case the algorithm should move to one of the external vertices of u . The selected vertex $X_k(u)$ is chosen with probability: $\frac{f_k(u)}{\sum_{1 \leq j \leq \lceil \deg(u)/d \rceil} f_j(u)}$. According to the chosen $X_j(u)$, the algorithm sets

$\Gamma_i(v)[t] \leftarrow X_j(u)$. In addition, the algorithm chooses uniformly at random one of the $f_j(u)$ free entries in the vector $\Gamma_j(u)$. Assume that the chosen index is t' , $1 \leq t' \leq d$, then, the algorithm sets $\Gamma_j(u)[t'] \leftarrow X_i(v)$.

Case 2: If more than half of the neighbors of v in G are in $A(v)$, the algorithm reads all the neighbors of v in G that belong to $NA(v)$, and attaches them arbitrarily to the unoccupied entries in $\Gamma_j(v)$, $1 \leq j \leq \lceil \deg(v)/d \rceil$. By doing this, the algorithm (at most) doubles the number of neighbor queries performed on vertex v of G . Now, suppose that in $\Gamma_i(v)[t]$ there is a name of a vertex of G , say u . In that case, the algorithm should move to one of the external vertices $X_k(u)$, $1 \leq k \leq \lceil \deg(u)/d \rceil$, and this is done as in the first case.

Modifying the procedure Odd-Cycle. The procedure 'Odd-Cycle' is the same as Odd-Cycle in terms of the performance of random walks, which are emulated as described above. The only modification is in the last stage, where the procedure performs vertex-pair queries. Let (x, y) be the pair of vertices queried in G' . We answer the query as follows. If $(x, y) = (X_i(v), I_j(v))$ for some vertex v in G , then we answer according to the explicit construction of the subgraph $H(v)$. If $(x, y) = (X_i(v), I_j(u))$ for $u \neq v$, then the answer is always negative. If $(x, y) = (X_i(u), X_j(v))$ then we query the pair (u, v) in G . If there is no edge between (u, v) in G , we answer that there is no edge between $X_i(u)$ and $X_j(v)$. Otherwise, we give a positive answer. While this answer may be inconsistent with the construction of G' (since it would correspond to having a complete bipartite subgraph in G' between the external vertices of u and the external vertices of v), it always provides evidence to an odd cycle in the input graph G . An explanation follows.

Consider two paths in G' , where both paths start at the same vertex $x \in H(s)$, end at a pair of external vertices $X_i(v)$ and $X_j(u)$, respectively, and whose lengths have the same parity (so that $X_i(v)$ and $X_j(u)$ both belong to the same A_b , $b \in \{0, 1\}$). By construction of G' , such a pair of paths in G' corresponds to a pair of paths in G , which start at s , end at v and u respectively, and have the same parity b as well. But if there is an edge in G between v and u , then there is an odd cycle in G .

3.3.3 Establishing Item 4 in Theorem 3.3.1

In this subsection we prove the last item in Theorem 3.3.1. Recall that we are interested in a procedure for selecting a vertex in G' so that there is a sufficiently high probability of hitting any fixed sufficiently large subset of vertices in G' . In particular, if G' is far from being bipartite then we are interested in hitting the subset of vertices s for which Odd-Cycle(s) returns **found** with probability at least $2/3$.

Let $V_\ell(G) = \{v \in V(G) : \deg(v) \leq d\}$ and let $V_h(G) = \{v \in V(G) : \deg(v) > d\}$, where ' ℓ ' stands for *low*, and ' h ' for *high*, and as before, $d = d_{\text{avg}}(G)$ denotes the average degree of vertices in G . We also define the corresponding sets in G' : $V_\ell(G') = \{x \in V(H(v)) : v \in V_\ell(G)\}$ and $V_h(G') = \{x \in V(H(v)) : v \in V_h(G)\}$. (Recall that $H(v)$ is the subgraph in G' that corresponds to v , and $V(H(v))$ is its set of vertices.) Since $V(G') = V_\ell(G') \cup V_h(G')$, it follows that selecting a vertex uniformly in $V(G')$ can be done by first deciding whether to pick a vertex from $V_\ell(G')$ or from $V_h(G')$ with probability proportional to the size of each set (relative to $n' = |V(G')|$), and then picking a vertex uniformly from the selected set.

Recall that for every $v \in V_\ell(G)$ we have $|V(H(v))| = 1$ while for every $v \in V_h(G)$, $|V(H(v))| = 2 \lceil \deg(v)/d \rceil$. Therefore, picking a vertex uniformly in $V_\ell(G')$ corresponds to picking a vertex uniformly in $V_\ell(G)$, while picking a vertex uniformly in $V_h(G')$ corresponds to picking a vertex in $V_h(G)$ with probability proportional to its degree.

Since we are not required to actually select every vertex in $V(G')$ with exactly equal probability, but rather we are required to be able to select all but $\delta n'$ of the vertices in $V(G')$ with probability at least $\Omega(1/n')$, we may perform the above steps in an approximate manner. In particular, by taking a sample of $\Theta(1/\delta^2)$ vertices in G and querying their degrees, we may obtain an estimate, denoted $\hat{\mu}(G')$, of $|V_\ell(G')|/n' = |V_\ell(G)|/n'$ such that if $|V_\ell(G')|/n' \geq \delta/2$ then $(1/8)|V_\ell(G')|/n' \leq \hat{\mu}(G') \leq 2(|V_\ell(G')|/n')$

(recall that $n \leq n' \leq 4n$). In order to uniformly select a vertex in $V_\ell(G)$ (so as to obtain a uniformly selected vertex in $V_\ell(G')$), we can simply take a sample of vertices from $V(G)$, query their degrees, and pick the first vertex in the sample that belongs to $V_\ell(G)$, if such exists. If $|V_\ell(G')|/n' \geq \delta/2$, so that $|V_\ell(G)|/n \geq \delta/2$, then a sample of size $O(1/\delta)$ suffices to ensure that with high constant probability, the sample will indeed contain a vertex in $V_\ell(G)$.

The only step that is more involved is that of selecting a vertex in $V_h(G)$ with probability proportional to its degree. Observe that selecting a vertex from all of $V(G)$ with probability proportional to its degree can be performed by uniformly selecting an *edge* in $E(G)$ and then selecting one of its two end-points with equal probability.

In the following we use a procedure presented in Chapter 6 that performs a certain approximation to the uniform selection of an edge in $E(G)$.

Sampling Vertices in G'

We now return to selecting vertices in G' . As discussed earlier, we can easily obtain an estimate of $|V_\ell(G')|/n' = |V_\ell(G)|/n'$, denoted $\hat{\mu} = \hat{\mu}(G')$, such that if $|V_\ell(G')|/n' \geq \delta/2$ then $(1/8)|V_\ell(G')|/n' \leq \hat{\mu}(G') \leq 2(|V_\ell(G')|/n')$, and hence we assume that we indeed have such an estimate.

Sample-Vertices-Almost-Uniformly-in- $G'(d, \delta, \hat{\mu})$

1. Flip a coin with bias $\hat{\mu}$.
2. If the outcome is “heads” then do (select a vertex in $V_\ell(G')$):
 - (a) Uniformly select $\Theta(1/\delta)$ vertices in G and query their degrees.
 - (b) If some vertex in the sample belongs to $V_\ell(G)$ then let v be the first such vertex and output the single vertex $x \in V(H(v))$. Otherwise, pick an arbitrary vertex v in the sample and output an arbitrary vertex $x \in V(H(v))$.
3. Else (the outcome is “tails”) do (select a vertex in $V_h(G')$):
 - (a) If $d > \sqrt{\delta n}$ then sample an edge $e \in E(G)$ by running the procedure Sample-Edges-Uniformly-in- G . In case the procedure fails, pick an arbitrary edge $e \in E(G)$.
 - (b) Else ($d \leq \sqrt{\delta n}$), sample an edge $e \in E(G)$ by running the procedure Sample-Edges-Almost-Uniformly-in- $G(\delta/2)$.
 - (c) Choose with equal probability one of the end-points v of the edge e .
 - (d) Choose uniformly at random one of the vertices x in $V(H(v))$.

Figure 3.5: A procedure for selecting a vertex in G' so that all but at most a δ -fraction of the vertices are selected with probability $\Omega(1/n')$.

Proof of Item (4) in Theorem 3.3.1. We now show that the procedure Sample-Vertices-Almost-Uniformly-in- G' (see Figure 3.5), is as required in Item (4) of Theorem 3.3.1. Consider first the vertices in $V_\ell(G')$. If $|V_\ell(G')|/n' \geq \delta/2$, then for each vertex $x \in V_\ell(G')$, the probability that we select x is $\hat{\mu}(G')$, times the probability that the sample contains a vertex in $V_\ell(G')$, times $1/|V_\ell(G')|$. Since $\hat{\gamma} = \Omega(|V_\ell(G')|/n')$ and the sample contains a vertex from $V_\ell(G)$ with constant probability, the probability that we select x is $\Omega(1/n')$ as required. If $|V_\ell(G')|/n' < \delta/2$ then the probability that we obtain any vertex in $V_\ell(G')$ may be very small, but we are allowed to have $\delta n'$ such vertices and we shall account for these at most $(\delta/2)n'$ vertices.

We now turn to the vertices in $V_h(G')$. For an edge $e \in E(G)$, let C_e denote the event that e is selected by Sample-Edges-Uniformly-in-G in case $d > \sqrt{\delta n}$, or by Sample-Edges-Almost-Uniformly-in-G in case $d \leq \sqrt{\delta n}$. For $v \in V(G)$ let C_v denote the event that v is selected in Step 3c of procedure Sample-Vertices-Almost-Uniformly-in-G'. For $x \in V(G')$ let C_x denote the event that x is selected in Step 3d of the procedure and let $v(x)$ be such that $x \in H(v(x))$. Recall that for every $v \in V_h(G)$, we have $|V(H(v))| = 2\lceil \deg(v)/d \rceil$. Therefore for every $x \in V_h(G')$,

$$\Pr[C_x] \geq \Pr[C_{v(x)}] \cdot \frac{1}{2\lceil \deg(v(x))/d \rceil} = \sum_{e=(u,v(x))} \frac{1}{2} \Pr[C_e] \cdot \frac{1}{2\lceil \deg(v(x))/d \rceil} \quad (3.16)$$

If $d > \sqrt{\delta n}$ then $\Pr[C_e]$ is only slightly smaller than $1/m$ (since there is a probability that the procedure Sample-Edges-Uniformly-in-G fails to output an edge), implying that $\Pr[C_x] = \Omega(1/n')$. If $d \leq \sqrt{\delta n}$, then $\Pr[C_e]$ is determined by the procedure Sample-Edges-Almost-Uniformly-in-G. Let U_0 be as defined in Theorem 6.2.1, and consider first the case where $v(x) \notin U_0$. Then for every edge e that is incident to $v(x)$, we have that $\Pr[C_e] \geq 1/(64m) = 1/(32dn)$. By Equation (3.16), for each such vertex x we have that

$$\Pr[C_x] \geq \frac{1}{2} \deg(v(x)) \cdot \frac{1}{32dn} \cdot \frac{1}{2\lceil \deg(v(x))/d \rceil} \geq \frac{1}{256n} \geq \frac{1}{256n'} \quad (3.17)$$

as required. Next consider the case that $v(x) \in U_0$ but $\deg(v(x)) \geq 2|U_0|$. In such a case for at least half of the edges e incident to $v(x)$ we have that $\Pr[C_e] \geq 1/(32dn)$, and we can deduce that $\Pr[C_x] \geq \frac{1}{512n'}$. It remains to show that the total number of vertices x such that $v(x) \in U_0$ and $\deg(v) \leq 2|U_0|$, is at most $(\delta/2)n'$. Using the fact that $|U_0| \leq \sqrt{(\delta/2)n}/2$ (recall that the procedure Sample-Edges-Almost-Uniformly-in-G is called with its input parameter set to $\delta/2$), and for every vertex $v \in V_h(G)$, $|V(H(v))| = 2\lceil \deg(v)/d \rceil \leq 2\deg(v)$, we get:

$$\sum_{v \in U_0: \deg(v) \leq 2|U_0|} |V(H(v))| \leq 4|U_0|^2 \leq (\delta/2)n'. \quad (3.18)$$

The lemma follows.

3.4 A Lower Bound

In this section we present a lower bound on the number of queries necessary for testing bipartiteness. Similarly to the lower bound presented in [39], this lower bound holds for testing algorithms that are allowed a two-sided error, and the graphs used for the lower bound construction are regular graphs. However, the lower bound of $\Omega(\sqrt{n})$ (for constant ϵ) established in [39], holds for graphs having constant degree (e.g., degree 3), and when the algorithm is allowed only neighbor queries. Our lower bound is more general in that it allows the algorithm to perform both neighbor queries and vertex-pair queries, and it is applicable to all degrees. Indeed, the two families of graphs that we define below in our lower bound construction, can be viewed as generalizing the two families presented in [39]. However, since we have to deal with any given degree d and not only with $d = 3$, and since we have to deal with both types of queries, the analysis itself does not follow as a straightforward generalization of the analysis in [39].

Theorem 3.4.1 *Every algorithm for testing bipartiteness with distance parameter $\epsilon \leq 2^{-4}$ must perform $\Omega(\min(\sqrt{n}, n^2/m))$ queries.*

The high-level structure of our proof is similar to other lower-bound proofs for testing, which can be traced back to [70]. We present two distributions over graphs, where all graphs generated by one distribution are bipartite (and hence should be accepted), while with very high probability a graph generated according

to the other distribution is far from bipartite. We then show that any algorithm with query complexity below the lower bound, cannot distinguish between the two distributions (and hence must have a large failure probability).

Specifically, both distributions, denoted $\mathcal{G}(n, d)$, and $\mathcal{G}(n/2, n/2, d)$, are over d -regular graphs having n vertices, where we assume for simplicity that n is even. A graph generated according to $\mathcal{G}(n, d)$ is obtained by selecting, uniformly and independently, d perfect matchings between the n vertices. A graph generated according to $\mathcal{G}(n/2, n/2, d)$ is obtained by first randomly partitioning the n vertices into two equal parts, and then selecting, uniformly and independently, d perfect matchings between the two parts. By definition, all graphs in the support of $\mathcal{G}(n/2, n/2, d)$ are bipartite, and we prove that graphs generated according to $\mathcal{G}(n, d)$ are ϵ -far from being bipartite with high probability, for $\epsilon \leq 1/16$ and $d \geq 64$

We then show that the following two claims hold when a graph is generated either according to $\mathcal{G}(n, d)$ or according to $\mathcal{G}(n/2, n/2, d)$: (1) Any algorithm that asks $o(n^2/m) = o(n/d)$ queries, will not detect an edge by any vertex-pair query with very high probability. (2) Any algorithm that asks $o(\sqrt{n})$ queries will not receive as an answer to any neighbor query, a vertex it has already observed in a previous query (with very high probability as well). From this we can conclude that any algorithm that asks $o(\min(\sqrt{n}, n^2/m))$ queries cannot distinguish between the two distributions, as desired. In the lower bound proof we show the necessity of both *neighbor queries* and *vertex-pair queries*. Specifically, by using only one type of queries the lower bound increases. A detailed description of the lower bound is quite technical and appears in [46].

Chapter 4

Testing k -Colorability in General Graphs

4.1 Introduction

A graph is k -colorable if it is possible to partition its vertices into k parts such that there are no edges with both endpoints in the same part. k -Colorability can be tested in the dense model using $\text{poly}(k/\epsilon)$ queries [37, 6]. Thus, the complexity of this problem in the dense case is independent of the number of vertices n . In [25] an $\Omega(n)$ lower bound is presented for testing this property in the sparse bounded-degree case.

The Main Result. Dense graphs can be tested for k -colorability with constant complexity, while the complexity of testing bounded-degree graphs is $\Omega(n)$. Here, we consider testing k -colorability in general graphs. We present an algorithm whose complexity is $\tilde{O}(\min((n/d)^2, n/d \cdot \sqrt{n}, nd)) < \tilde{O}(n^{\frac{5}{4}})$ where d is the average degree in the graph. We also present a two-sided lower bound of $\Omega(n/d)$ for this problem. Thus, our upper bound is at most quadratic in the lower bound. Our results present a *smooth transition* between the known results in the sparse and in the dense case.

k -colorability is a generalization of the bipartiteness property. Indeed, if a graph is k -colorable for $k = 2$, then it is bipartite. By adapting the techniques presented in Chapter 3, we could get a lower bound of $\tilde{O}(\min(\sqrt{n}, n/d))$ for testing k -colorability. However, the result of this chapter suggests a stronger lower bound. A one-sided tester for k -colorability that rejects a graph, must provide an evidence that the input graph is not k -colorable. For $k = 2$ such an evidence is an odd cycle. However, for general k , such evidence may have much more complicated structure. This, could be the intuition explaining the reason for the difficulty in designing an efficient tester for k -colorability. Indeed, the result we provide here is not optimal in the sense that there exists a gap between the obtained lower and upper bounds.

Our Techniques. We first design a testing algorithm for the case of regular graphs that is based on [6]. We further show a *probabilistic reduction* from regular graphs to graphs with varying degrees. This reduction preserves the query complexity of the testing algorithm. The reduction is quite general, it proved to be helpful in the case of testing bipartiteness (see [46]), and it might be proven to be useful for some other graph properties.

Some Useful Definitions. Given a graph G' over n' vertices and m' edges, a d -blowup of G' , is a graph G obtained in the following manner. Every vertex of G' is transformed into d vertices in G . Thus, the number of vertices in G is $n = n'd$. Every edge of G' is transformed into d^2 edges in G that form a complete bipartite graph between the sets of the corresponding vertices. Thus, the number of edges in G is $m'd^2$.

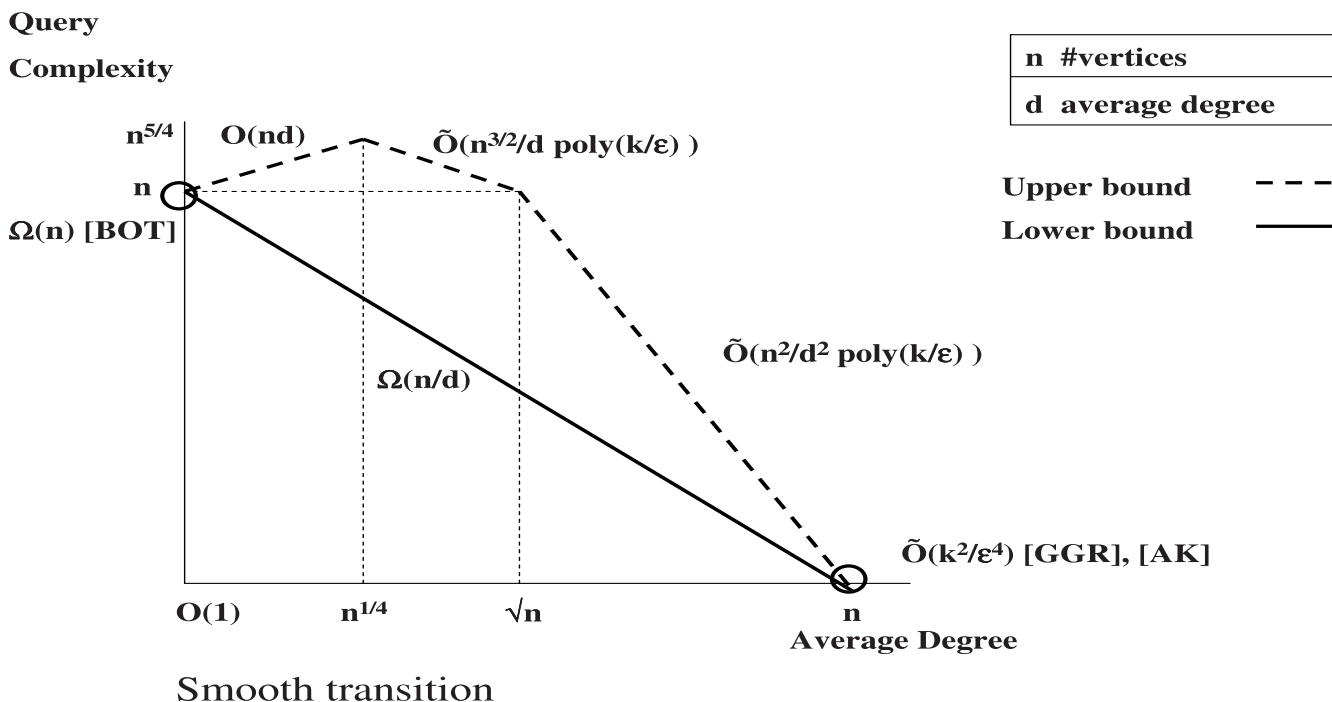


Figure 4.1: Testing k -colorability - illustration of results for general graphs obtained in this chapter.

Consider a graph G' with n' vertices $v_1, \dots, v_{n'}$. The d -blowup of G' is G and its set of vertices is labeled $v_{1,1}, \dots, v_{1,d}, \dots, v_{n',1}, \dots, v_{n',d}$.

We denote by $P_{n,s}$ the set of all subsets of $\{1, \dots, n\} = [n]$ that contain at most s elements. Given a graph G and a set of its vertices v_{i_1}, \dots, v_{i_s} , we denote by $G_{v_{i_1}, \dots, v_{i_s}}$, the subgraph of G induced by these vertices.

4.2 One-Sided Lower Bound

In this section we prove an $\Omega(\frac{n}{d})$ one-sided lower bound for testing k -colorability in general graphs, over n vertices with average degree (density) d . We first prove a one-sided error lower bound, which follows from a combinatorial analysis, and later show a two-sided error lower bound.

Theorem 4.2.1 *Every one-sided error algorithm for testing k -colorability in graphs with average degree (density) d over n vertices, requires $\Omega(\frac{n}{d})$ queries.*

The following lemma will be of use:

Lemma 4.2.1 *There exists a graph G on n vertices with maximum degree bounded by a constant, that is $\Theta(1)$ -far from being k -colorable. Yet, there exist a constant $\alpha > 0$, such that the induced subgraph on every set of αn vertices of G is k -colorable.*

Proof: We use a probabilistic argument for showing the existence of a graph G as required by the lemma. The distribution of graphs noted as $G(n, p)$, is a distribution of graphs over n vertices, such that there is an

edge between a pair of vertices with probability p , and there is no-edge with probability $1 - p$. The edge events are independent. Consider a graph G selected uniformly from the distribution $G(n, p = 8k^3/n)$. We next show that such G obeys the lemma conditions with positive probability and deduce that such G exists.

Using known properties of graphs created by the distribution $G(n, p = 8k^3/n)$ (see [11]), with probability $1 - o(1)$, G contains at least $4k^3n$ edges (half of the expected number of edges). Moreover, there are at most a constant number of vertices with degree $\text{poly}(\log n)$, and there are no vertices with higher degrees.

Next, we show that G is far from being k -colorable with probability $1 - o(1)$. Consider a k -partition $P = V_1, \dots, V_k$ of the vertices of G . Clearly one of the parts has size at least n/k . Assume without loss of generality that $|V_1| \geq n/k$. Let X_{V_1} be a random variable indicating the number of edges in V_1 .

As X_{V_1} is a sum of $|V_1|^2 \geq n^2/k^2$ independent Bernoulli random variables, each with expectation at least $8k^3/n$, it follows from standard bounds on the tails of binomial random variables (see, e.g., [11]) that

$$\Pr[X_{V_1} < 4kn] < e^{-kn} \ll k^{-n}.$$

By taking the union bound over all possible partitions P of the vertices of G we get that for every partition $P = V_1, \dots, V_k$ of the vertices of G , the number of edges in G that are within a partition is at least $4kn$ with probability $1 - o(1)$.

Next, we show that with probability $1 - o(1)$, every subset of vertices of G of size αn is k -colorable, for $\alpha < \frac{1}{k^2 e^4}$.

Note that a subset of vertices $S \subset V(G)$ of minimal size that is not k -colorable must have minimal degree at least k . Hence, such S must contain at least $k|S|/2$ edges.

The probability that there exists a set $S \subset V(G)$ of size $s \leq \alpha n - 1$ that contains at least $ks/2$ edges can be bounded from above by:

$$\sum_{s=4}^{\alpha n - 1} \binom{n}{s} \binom{\binom{s}{2}}{\binom{ks}{2}} (8k^3/n)^{ks/2}$$

By standard bounds, the sum can be bounded by its last term, that is,

$$\begin{aligned} \sum_{s=4}^{\alpha n - 1} \binom{n}{s} \binom{\binom{s}{2}}{\binom{ks}{2}} (8k^3/n)^{ks/2} &\leq \left(\frac{ne}{\alpha n}\right)^{\alpha n} \cdot \left(\frac{e^{k/2} \alpha n^{k/2}}{k^{k/2}}\right)^{\alpha n} \left(\frac{8^{k/2} k^{3k/2}}{n^{k/2}}\right)^{\alpha n} = \\ &\left(\frac{e}{\alpha} \cdot \frac{e^{k/2} \alpha^{k/2} n^{k/2}}{k^{k/2}} \cdot \frac{8^{k/2} k^{3k/2}}{n^{k/2}}\right)^{\alpha n} < (e^2 k \alpha^{k/2} k^k)^{\alpha n} = [e^2 \alpha^{1/2} k]^{\alpha n k} = o(1) \end{aligned} \quad (4.1)$$

From all the above a graph G selected uniformly from $G(n, p = 8k^3/n)$ has with probability $1 - o(1)$ the following properties:

- G contains a linear number of edges.
- G is $\Theta(1)$ -far from being k -colorable
- Every subset of αn of the vertices of G span a k -colorable graph.
- The total number of edges incident to vertices of G with degree greater than a constant is $o(n)$.

Hence, we can remove edges incident to vertices of G with degree greater than a constant, and obtain a graph that answers all requirements of the lemma. As a graph required by the lemma exists with positive probability, the lemma follows. ■

Proof of Theorem 4.2.1: A one-sided tester for k -colorability can reject a graph G' only if it finds a subgraph of G' which is not k -colorable. Consider a graph G' over n' vertices that is promised to exist by

Lemma 4.2.1. Such graph contains a linear number of edges, and its maximal degree is a constant (G' is a sparse graph). Moreover, it is $\Theta(1)$ -far from being k -colorable, yet every set of linear number of vertices is k -colorable. Clearly, a one-sided tester for k -colorability must reject G' . However, it needs to provide a witness for the non k -colorability of G' . By the properties of G' , such a witness must contain $\Theta(n')$ vertices, and hence we obtain a lower bound of $\Omega(n')$ for testing k -colorability in sparse graphs, with average degree $d = \Theta(1)$.

In the following we deduce that there exists a one-sided lower bound for testing k -colorability in graphs with density d (i.e. average degree d) which is $\Omega(n/d)$. Consider a graph G over $n = dn'$ vertices obtained by a d -blowup of G' . Following the properties of G' and the definition of the blowup, G is $\Theta(1)$ -far from being k -colorable, yet every subset of $\alpha n' = \alpha n/d$ vertices of G spans a k -colorable graph. Clearly, a one-sided tester for k -colorability must reject G , however, since the smallest evidence for the non k -colorability of G , contains $\Theta(n/d)$ vertices, a $\Omega(n/d)$ lower bound for testing k -colorability by a one-sided tester is obtained. ■

4.3 A Two-Sided Lower Bound

In this section we prove an $\Omega(\frac{n}{d})$ two-sided lower bound for testing k -colorability in general graphs, over n vertices with average degree (density) d . Our proof is based on the two-sided lower bound for testing k -colorability in constant degree graphs obtained in [25].

Theorem 4.3.1 *Every two-sided error algorithm for testing k -colorability in graphs with average degree (density) d over n vertices, requires $\Omega(\frac{n}{d})$ queries.*

Proof: The authors of [25] show that there exist two distributions of d -regular graphs (for constant d) over n' vertices with the following properties. The first distribution, denoted as $D'_{3\text{col}}$ contains graphs that are all 3-colorable. The second distribution, denoted as $D'_{3\text{col-far}}$ contains graphs that are all $\Theta(1)$ -far from being 3-colorable with high probability. Moreover, $|D'_{3\text{col-far}}| = |D'_{3\text{col}}|$ and the distributions are statistically indistinguishable by a tester that has observed $o(n')$ vertices of the input graph.

Equivalently, one can say that for every graph on $s' = o(n')$ vertices $G'_{s'}$, the the probability that $G'_{s'}$ appears as an induced subgraph of a graph created by the distribution $D'_{3\text{col-far}}$, is equal to the probability that $G'_{s'}$ appears as an induced subgraph of a graph created by the distribution $D'_{3\text{col}}$ (that is, $\Pr_{D'_{3\text{col-far}}}(G'_{s'}) = \Pr_{D'_{3\text{col}}}(G'_{s'})$).

The fact that for every $G'_{s'}$ on $s' = o(n')$ vertices, $\Pr_{D'_{3\text{col-far}}}(G'_{s'}) = \Pr_{D'_{3\text{col}}}(G'_{s'})$ is equivalent to the fact that there exists a bijection function $f' : D'_{3\text{col}} \times P_{n',s'} \rightarrow D'_{3\text{col-far}} \times P_{n',s'}$ that is defined as follows. For a pair $G' \in D'_{3\text{col}}$ and a set $v_{i_1}, \dots, v_{i_{s'}}$ of s' of its vertices, consider the induced graph $G_{v_{i_1}, \dots, v_{i_{s'}}}$. The function f' returns a graph $G'^* \in D'_{3\text{col-far}}$, and a set of its vertices $u_{i_1}, \dots, u_{i_{s'}}$, s.t. $G'_{v_{i_1}, \dots, v_{i_{s'}}} = G'^*_{u_{i_1}, \dots, u_{i_{s'}}}$.

In the following we use the distributions $D'_{3\text{col}}, D'_{3\text{col-far}}$ to construct two distributions of d -regular graphs over $n = n'd$ vertices where $\Theta(1) \leq d \leq \Theta(n)$, such that one distribution contains 3-colorable graphs and the other contains graphs that are all $\Theta(1)$ -far from being 3-colorable with high probability, and such that the two distributions are identical for a tester that have seen $o(n/d)$ vertices of the input graph.

The proof of the last claim goes as follows. The two distributions denoted $D_{3\text{col}}, D_{3\text{col-far}}$, are created by a d -blowup of every graph in the original distributions $D'_{3\text{col}}$ and $D'_{3\text{col-far}}$. Note that we get graphs on $n'd = n$ vertices, and $n'd^2/2 = nd/2$ edges that are d -regular. The vertices of the graphs in the new distributions are labeled by $v_{1,1}, \dots, v_{n',d}$ such that for every vertex we keep track of its original name before the blowup. Clearly from the properties of the original distributions and from the nature of a blowup, graphs

in $D_{3\text{col}}$ are all 3-colorable, and graphs in $D_{3\text{col-far}}$ are all $\Theta(1)$ -far from being 3-colorable with high probability.

It remains to show that the two distributions are indistinguishable unless $s = o(n/d) = o(n')$ vertices of the input graph were seen. Using similar arguments to those mentioned above, it is sufficient to show that there exists a bijection function $f : D_{3\text{col}} \times P_{n,s} \rightarrow D_{3\text{col-far}} \times P_{n,s}$ that is defined as follows. For a pair $G \in D_{3\text{col}}$ and a set $v_{i_1, j_1}, \dots, v_{i_s, j_s}$ of s of its vertices, consider the induced graph $G_{v_{i_1, j_1}, \dots, v_{i_s, j_s}}$. The function f returns a graph $G^* \in D_{3\text{col-far}}$, and a set of its vertices $u_{i_1, j_1}, \dots, u_{i_s, j_s}$, s.t. $G_{v_{i_1, j_1}, \dots, v_{i_s, j_s}} = G_{u_{i_1, j_1}, \dots, u_{i_s, j_s}}^*$.

In what follows, we show that a required bijection function f exists. Consider a graph $G \in D_{3\text{col}}$, whose original graph (before the blowup) is $G' \in D_{3\text{col}}'$. Consider a set of $s = o(n') = o(n/d)$ of the vertices of G , $v_{i_1, j_1}, \dots, v_{i_s, j_s}$. Now $f(G, v_{i_1, j_1}, \dots, v_{i_s, j_s})$ is determined as follows. Consider the (possibly multi) set v_{i_1}, \dots, v_{i_s} . Remove repetitions from the last set and obtain the set $v_{k_1}, \dots, v_{k_{s'}}$, $s' \leq s$. Now consider the value obtained by $f'(G', v_{k_1}, \dots, v_{k_{s'}}) = u_{k_1^*}, \dots, u_{k_{s'}^*}$. With a slight abuse of notation we could use the notion $f_{G', v_{k_1}, \dots, v_{k_{s'}}}(k_i) = k_i^*$.

The value of $f(G, v_{i_1, j_1}, \dots, v_{i_s, j_s})$ is defined, now, as follows:

$$f(G, v_{i_1, j_1}, \dots, v_{i_s, j_s}) = u_{f_{G', v_{k_1}, \dots, v_{k_{s'}}}(i_1), j_1}, \dots, u_{f_{G', v_{k_1}, \dots, v_{k_{s'}}}(i_s), j_s}$$

Clearly from the definition of the blowup and from the bijectivity of f' it follows that f is bijective and hence the theorem follows.

Since the two original distributions presented in [25] could be adjusted to show two-sided lower bound for testing k -colorability for constant $k > 3$, we can apply the same arguments and get our two-sided lower bound for testing k -colorability for $k \geq 3$. ■

4.4 An Upper Bound for Almost Regular Graphs

A graph G over n vertices with average degree d is *almost regular* if the maximum degree in the graph is $\Theta(d)$. In this section we use arguments similar to the ones appear in [6] to obtain an $O(\min((\frac{n}{d})^2, n))$ upper bound for testing k -colorability in almost regular graphs with general densities.

Theorem 4.4.1 *There exists a one-sided error algorithm that uses $O(\min((\frac{n}{d})^2, n) \cdot \text{poly}(k/\epsilon))$ queries, for testing k -colorability in almost regular graphs over n vertices with degree d .*

We start with a high level discussion of the strategy used in the analysis. Let G be a graph on n vertices that is ϵ -far from being k -colorable. In the following we assume that G is regular, and in the end we explain that the result is obtained also for almost regular graphs. Suppose we are given a subset $S \subset V(G)$ and its k -partition $\phi : S \rightarrow [k]$, our aim is to find with high probability inside the next several random vertices a witness to the fact that ϕ can not be extended to a proper coloring of the sample. If a k -coloring $c : V(G) \rightarrow [k]$ of G is to coincide with ϕ on S , then for every vertex $v \in V \setminus S$, the colors of neighbors of v in S under ϕ are forbidden for v in c . The rest of the colors are still feasible for v . It could be that v has no feasible colors left at all. Such a vertex will be called *colorless* with respect to S and ϕ . If the number of colorless vertices is large, then there is a decent chance that between the next few random sampled vertices, there will be one such colorless vertex v^* . Clearly, adding v^* to S provides the desired witness for non-extendibility of ϕ .

If the set of colorless vertices is small, then one can show that, as G is ϵ -far from being k -colorable, there is a relatively large subset W of vertices (called restricting) such that adding any vertex $v \in W$ to S and coloring it by any feasible color with regard to ϕ excludes this color from the lists of feasible colors of at least $\Omega(\epsilon d)$ neighbors of v . If such v is caught in the next few vertices of the random sample, then adding

v to S and coloring it by any of its feasible colors reduces substantially the total length of the lists of feasible colors for the vertices of V , thus helping to approach the case when there are many colorless vertices.

The above described process can be represented by a tree in which every node corresponds to a colorless or a restricting vertex v and each edge corresponds to a feasible color for v . As the degree of such a node can be as large as k , the size of the tree grows quickly as we choose new sampled vertices. We therefore will need the probability of success (the probability of catching a colorless/restricting vertex) along several consecutive steps to be exponentially close to 1.

Next we present the formal description of the above argument. First we introduce some notation. We denote the set $\{1, \dots, k\}$ by $[k]$. Suppose $G = (V, E)$ is a graph on n vertices. Given a subset $S \subset V$ and its k -partition $\phi : S \rightarrow [k]$, for every $v \in V - S$ let

$$L_\phi(v) = [k] \setminus \{1 \leq i \leq k : \exists u \in S \cap \Gamma(v), \phi(u) = i\}$$

If $S = \emptyset$, we let $L_\phi(v) = [k]$ for every $v \in V$. If a k -coloring $c : V(G) \rightarrow [k]$ of G coincides with ϕ on S , then for every $v \in V - S$ the color of v in c belongs to $L_\phi(v)$. Hence, $L_\phi(v)$ is called the *list of feasible colors* for v . A vertex $v \in V - S$ is called *colorless* if $L_\phi(v) = \emptyset$. We denote by U the set of all colorless vertices under (S, ϕ) . For every vertex $v \in V - (S \cup U)$ define

$$\delta_\phi(v) = \min_{i \in L_\phi(v)} |\{u \in \Gamma(v) - (S \cup U) : i \in L_\phi(u)\}|.$$

Thus, coloring v by one of the colors from $L_\phi(v)$ and then adding it to S results in deleting this color and thus shortening the lists of feasible colors of at least $\delta_\phi(v)$ neighbors of v outside S .

Lemma 4.4.1 *For every set $S \subset V$ and every k -partition ϕ of S , the graph G is at most $d|S \cup U| + \sum_{v \in V - (S \cup U)} \delta_\phi(v)$ edges far from being k -colorable.*

Proof: For every $v \in S$, color v according to $\phi(v)$. For every $v \in U$, color v in an arbitrary color from $[k]$. For every $v \in V - (S \cup U)$, color v in color $i \in L_\phi(v)$ for which $\delta_\phi(v) = |\{u \in \Gamma(v) - (S \cup U) : i \in L_\phi(u)\}|$.

Let us estimate the number of monochromatic edges under this coloring. The number of monochromatic edges incident with $S \cup U$ is at most $d|S \cup U|$. Every vertex $v \in V - (S \cup U)$ has exactly $\delta_\phi(v)$ neighbors $u \in V - (S \cup U)$, whose color list $L_\phi(v)$ contains the color chosen for v . Therefore, v will have at most $\delta_\phi(v)$ neighbors in $V - (S \cup U)$ colored in the same color. Hence, the total number of monochromatic edges is as claimed. ■

Corollary 4.4.2 *If G is a graph on n vertices that is ϵ -far from being k -colorable, then for any pair (S, ϕ) , where $S \subset V(G)$, $\phi : S \rightarrow [k]$, one has:*

$$\sum_{v \in V - (S \cup U)} \delta_\phi(v) > \epsilon nd - d(|S| + |U|),$$

where U is the set of colorless vertices for the pair (S, ϕ) .

Given a pair (S, ϕ) , a vertex $v \in V - (S \cup U)$ is called *restricting* if $\delta_\phi(v) \geq \epsilon d/2$. We denote by W the set of all restricting vertices.

Lemma 4.4.2 *If G is a graph on n vertices that is ϵ -far from being k -colorable, then for every pair (S, ϕ) , where $S \subset V(G)$, $\phi : S \rightarrow [k]$, one has:*

$$|U \cup W| > \epsilon n/2 - |S|.$$

Proof: By Corollary 4.4.2,

$$\epsilon nd - d(|S| + |U|) < \sum_{v \in V - (SUU)} \delta_\phi(v) \leq d|W| + \sum_{v \in V - (SUUW)} \delta_\phi(v) < d|W| + \frac{n \cdot \epsilon d}{2}.$$

This implies $|S| + |U| + |W| \geq \epsilon n/2$. As U and W are disjoint, the result follows. ■

Let now G be a graph on n vertices that is ϵ -far from being k -colorable. While choosing random vertices r_1, \dots, r_s we construct an auxiliary k -ary tree T . To distinguish between the vertices of G and those of T we call the latter *nodes*. Each node of T is labeled either by a vertex of G or by the special symbol $\#$, whose meaning is explained soon. If a node t of T is labeled by $\#$, then t is called a *terminal node*. The edges of T are labeled by integers from $[k]$.

Let t be a node of T . Consider the path from the root of T to t , not including t itself. The labels of the nodes along this path form a subset $S(t)$ of $V(G)$. The labels of the edges along the path define a k -partition $\phi(t)$ of $S(t)$ in a natural way: the label of the edge following a node t' in the path determines the color of its label $v(t')$. The labeling of the nodes and edges of T will have the following property: if t is labeled by v and v has a neighbor in $s(t)$ whose color in $\phi(t)$ is i , then the son of v along the edge labeled by i is labeled by $\#$. This label indicates the fact that in this case color i is infeasible for v , given $(S(t), \phi(t))$. At each step of the construction of T we will maintain the following: all leaves of T are either unlabeled or are labeled by $\#$. Also, only leaves of T can be labeled by $\#$. We start the construction of T from an unlabeled single node, the root of T .

Suppose that $j - 1$ vertices of T have already been chosen, and we are about to choose vertex r_j . Consider a leaf t of T . If t is labeled by $\#$, we do nothing for this leaf. (That is the reason such a t is called a terminal node; nothing will ever grow out of this node.) Assume now that t is unlabeled. Define the pair $(S(t), \phi(t))$ as described above. Now, for the pair $(S(t), \phi(t))$, we define the set $U(t)$ of colorless vertices and the set $W(t)$ of restricting vertices as described above. Round j is called *successful* for the node t if the random vertex r_j satisfies $r_j \in U(t) \cup W(t)$. If round j is indeed successful for t , then we label t by r_j , create k sons of t and label the corresponding edges by $1, \dots, k$. Now, if color i is infeasible for r_j , given $(S(t), \phi(t))$, we label the son of t along the edge with label i by $\#$, otherwise we leave this son unlabeled. Note that if $r_j \in U(t)$, then none of the colors from $[k]$ is feasible for r_j , and thus all the sons of t will be labeled by $\#$. This completes the description of the process of constructing T . Next, we state some properties of T .

Lemma 4.4.3 *The depth of T is bounded from above by $\frac{2kn}{\epsilon d}$.*

Proof: Let t^* be a leaf of T . Notice that if the label of a node t of T belongs to $U(t)$, then all sons of t in T are labeled by $\#$ and are terminal nodes. Therefore, all nodes on the path from the root of T to t^* , but possibly the node immediately preceding t^* , have their labels in the corresponding sets $W(t)$. Since each vertex in $W(t)$ is restricting with respect to $(S(t), \phi(t))$, coloring v in any feasible color decreases the total size of the lists of feasible colors for all vertices of G by at least $\epsilon d/2$. Therefore, each time when on the path from the root of T to t^* we leave a node t , whose label belongs to $W(t)$, the total length of the list of feasible colors shrinks by at least $\epsilon d/2$. As initially all k colors are feasible for all vertices, we start with lists of feasible colors of total length nk . Thus, we can not make more than $nk/(\epsilon d/2) = \frac{2nk}{\epsilon d}$ steps down from the root of T to t^* . This implies that the depth of T is at most $\frac{2nk}{\epsilon d}$. ■

Lemma 4.4.4 *If a leaf t^* of T is labeled by $\#$, then $\phi(t^*)$ is not a proper k -coloring of $S(t^*)$.*

Proof: By the definition of the labeling procedure. ■

Lemma 4.4.5 *If after round j all leaves of the tree T are terminal nodes, then the subgraph $G[\{r_1, \dots, r_j\}]$ is not k -colorable.*

Proof: Notice first that the labels of all nodes of T are either $\#$ or vertices from $\{r_1, \dots, r_j\}$. Let $c : \{r_1, \dots, r_j\} \rightarrow [k]$ be a k -partition of $\{r_1, \dots, r_j\}$. In order to show that c creates some monochromatic edges in the induced subgraph of G on $\{r_1, \dots, r_j\}$, we start with the root t^0 of T and traverse T guided by c as follows: while at node t of T , labeled by $v(t) \in \{r_1, \dots, r_j\}$, we move from t to its son along the edge of T labeled by $c(v(t))$. Once we reach a terminal node t^* of T , we have then $S(t^*) \subseteq \{r_1, \dots, r_j\}$ and $\phi(t^*)$ coincides with c on $S(t^*)$. As t^* is a terminal node, it follows from lemma 4.4.4 that c is not a proper k -coloring of $S(t^*)$. ■

Lemma 4.4.6 *If G is a graph on n vertices that is ϵ -far from being k -colorable, then after $\Theta(k \ln k \epsilon^{-2} \frac{n}{d})$ rounds, with probability at least $1/2$ all leaves of T are terminal nodes.*

Proof: As every non-leaf node of T has k sons and by lemma 4.4.3, T has depth at most $2kn/\epsilon d$, it can be embedded naturally in a k -ary tree $T_{k, \frac{2kn}{\epsilon d}}$ of depth $\frac{2kn}{\epsilon d}$. Moreover, this embedding can be prefixed even before exposing the sampled vertices. Note that the number of vertices of $T_{k, \frac{2kn}{\epsilon d}}$ is bounded by $k^{\frac{2kn}{\epsilon d} + 1}$. Recall that during the construction of the random sample R and the tree T , a successful round for a leaf t of T results in creating k sons of T . Fix some node t of $T_{k, \frac{2kn}{\epsilon d}}$. If after $36k \ln k \epsilon^{-2} \frac{n}{d}$ rounds t is a leaf of T , then the total number of successful rounds for the path from the root of T to t , is equal to the depth of t . As $S(t) \subseteq R$ and thus $|S(t)| = O(n/d)$, by lemma 4.4.2, each round has probability of success at least $\epsilon/3$. Therefore, the probability that t is non-terminal leaf of T after $36k \ln k \epsilon^{-2} \frac{n}{d}$ steps can be bounded from above by the probability that the Binomial random variable $B(36k \ln k \epsilon^{-2} \frac{n}{d}, \epsilon/3)$ is less than $\frac{2kn}{\epsilon d}$. The latter probability is at most:

$$e^{-\frac{(\frac{12k \ln kn}{\epsilon d} - \frac{2kn}{\epsilon d})^2}{24k \ln kn}} < e^{-\frac{(9k \ln kn)^2}{24k \ln kn}} = e^{-\frac{27k \ln kn}{8\epsilon d}} < k^{-\frac{3kn}{\epsilon d}}.$$

Thus, by a union bound we conclude that the probability that some node of $T_{k, \frac{2kn}{\epsilon d}}$, is a leaf of T , non labeled by $\#$, is at most $|V(T_{k, \frac{2kn}{\epsilon d}})| k^{-\frac{3kn}{\epsilon d}} < 1/2$ ■

Proof of Theorem 4.4.1: From lemmas 4.4.5 and 4.4.6 it is clear that a sample of vertices R of size $\Theta(k \ln k \epsilon^{-2} \frac{n}{d})$ spans a non- k -colorable subgraph of G with positive constant probability. Note that if G is almost regular, rather than being regular, this may affect the size of R only by a constant factor, and hence the size of the required R stays affectively the same. We can either ask all possible pair queries induced by the sampled set, or ask all possible d neighbors from every vertex in the set. Hence we obtain an algorithm with query complexity $O(\min((\frac{n}{d})^2, n) \cdot \text{poly}(k/\epsilon))$. ■

4.5 An Upper Bound for General Graphs

In this section we use arguments similar to the ones that appear in Chapter 3 to obtain an $O(\min((\frac{n}{d})^2, \frac{n \cdot \sqrt{n}}{d}, nd))$ upper bound for testing k -colorability in general graphs over n vertices with average degree d .

Theorem 4.5.1 *There exists a one-sided error algorithm that uses $O(\min((\frac{n}{d})^2, \frac{n \cdot \sqrt{n}}{d}, nd) \cdot \text{poly}(k/\epsilon))$ queries, for testing k -colorability in general graphs over n vertices with average degree d .*

We start by proving few lemmas that show the following: for a given general graph G with average degree d that is ϵ -far from k -colorability, we can construct a graph G' that is a "regular" version of G and it preserves the distance to k -colorability of G . Moreover, we can perform every query into G' by performing a constant number of queries into G . Lastly, we can select a random vertex of G' with a cost of $\tilde{O}(\min(\sqrt{n/\delta}, n/d))$ queries into G , for any given parameter $0 < \delta \leq 1$.

Next we show that there exists a random graph G' that preserves the number of edges and vertices of G , and the distance to k -colorability of G , and such that G' is close to be regular.

Lemma 4.5.1 *For every graph G having n vertices and $nd/2$ edges, we can construct randomly a graph G' having $n' = \Theta(n)$ vertices and the same number of edges with maximum degree $d' < 2d$. Moreover, If G is k -colorable then G' is k -colorable, and if G is ϵ -far from being k -colorable, then G' is ϵ' -far from being k -colorable for $\epsilon' = \Theta(\epsilon)$. All properties of G' apply w.h.p.*

Proof: The Construction: every vertex of G is transformed into $\lceil \deg(v)/d \rceil$ vertices. Denote by $X(v)$ the vertices in G' related to a vertex $v \in V(G)$. The vertices in $X(v)$ are denoted by $X_i(v), 1 \leq i \leq \lceil \deg(v)/d \rceil$. Thus, $n' = |V(G')| \leq \sum_{v \in G} \lceil \frac{\deg(v)}{d} \rceil \leq 2n$. The edges of G' are determined as follows: an edge $(u, v) \in E(G)$ chooses independently uniformly at random a vertex from $X(v)$ and a vertex from $X(u)$. In G' there will be an edge between these two randomly chosen vertices. Clearly, $|E(G')| = |E(G)| = (nd)/2$. The required properties of G' follows from the following set of claims.

Claim 4.5.2 *For $d = \Omega(\log n)$, the maximum degree d' of G' constructed above is $2d$ with probability $1 - o(1)$.*

Proof: Let $W_v^{i,j}$ be an indicator random variable which is 1, if the j -th induced edge of $v \in V(G)$ ($1 \leq j \leq \deg(v)$), chooses vertex $X_i(v) \in V(G')$. Let $W_v^i \stackrel{\text{def}}{=} \sum_{1 \leq j \leq \deg(v)} W_v^{i,j}$. W_v^i is a sum of j independent indicator variables. $\text{Exp}[W_v^i] = \deg(v) \cdot 1/(\deg(v)/d) = d$. Let X_v^i be an indicator variable which is 1, if $W_v^i > 2\text{Exp}[W_v^i] = 2d$, and 0 otherwise.

Using standard bounds on tails of sums of bounded random variables (see, e.g., [11]), for a specific $v \in V(G)$ and $1 \leq i \leq \lceil \deg(v)/d \rceil$, it follows that $\Pr[X_v^i = 1] < e^{-cd}$. Using a union bound over all X_v^i 's $v \in V(G)$ and $1 \leq i \leq \lceil \deg(v)/d \rceil$, we get that the probability that there exists a vertex v and an index i for which $X_v^i = 1$ is at most $n' \cdot e^{-cd} = o(1)$, thus with probability $1 - o(1)$, the maximum degree of G' constructed above is $2d$. ■

Claim 4.5.3 *For a graph G with $d > 16k^4/\epsilon$ the following holds: If G is ϵ -far from being k -colorable (with respect to number of edges being: $(dn)/2$), then with probability $1 - o(1)$, G' is ϵ' -far from being k -colorable with respect to $d'n'$, for $\epsilon' = \frac{\epsilon}{16k^3}$.*

Proof: Consider a fixed k -partition $P' = (V'_1, \dots, V'_k)$ of the vertices in G' . The partition P' induces a partition of $X(v)$ for every v . Let us denote by $X^\alpha(v)$ the majority subset of $X(v)$ induced by P' . Consider a partition $P = (V_1, \dots, V_k)$ of the vertices of G induced by P' in the following way. For $v \in V(G)$, if $X^\alpha(v) \subset V'_i$ for $1 \leq i \leq k$, then $v \in V_i$. Since G is ϵ -far from being k -colorable at least one of the subsets V_1, \dots, V_k contains $\frac{1}{2k}\epsilon nd$ edges. W.l.o.g. assume that $|E(V_1)| \geq \frac{1}{2k}\epsilon nd$. Let H' be a subgraph of G' , defined as follows. The vertices of H' are:

$$\bigcup_{v \in V_1} X^\alpha(v)$$

The edges of H' are the edges of G' , induced by $V(H')$. Thus, $V(H') \subset V'_1$ and $E(H') \subset E(V'_1)$.

We next show the following for $c > 1$:

$$\Pr[|E(H')| \leq (1/4k^3)\epsilon nd] < k^{-c \cdot n} \quad (4.2)$$

Once we establish Equation 4.2, by taking a union bound over all possible partitions P' of the vertices of G' we get that for every partition $P' = V'_1, \dots, V'_k$ of the vertices of G' , the number of violating edges in G' is at least $(1/4k^3)\epsilon nd$ with probability $1 - o(1)$. Recall that $n' \leq 2n$ and $d' \leq 2d$ with probability $1 - o(1)$.

Thus, for every partition of the vertices of G' , the number of violating edges is at least $\epsilon/16k^3 \cdot n' \cdot d'$ with probability $1 - o(1)$, as required.

Proof of 4.2: Consider an edge $e = (u, v) \in E(G)$, and let $\phi(e) = (X_i(v), X_j(u))$ be its corresponding edge in $E(G')$. For $e \in E(V_1)$, $\Pr[\phi(e) \in E(H')] \geq 1/k \cdot 1/k = 1/k^2$. Thus,

$$\text{Exp}[|E(H')|] \geq 1/k^2 |E(V_1)|.$$

As $|E(H')|$ is a sum of $|E(V_1)|$ independent Bernoulli random variables, each with expectation at least $1/k^2$, it follows from standard bounds on the tails of binomial random variables (see, e.g., [11]) that

$$\Pr[X_{|E(V_1)|} < (1/4k^3)\epsilon nd] < e^{-\epsilon nd/16k^3}$$

Thus, for $d \geq 16k^4/\epsilon$ we have

$$\Pr[|E(H')| \leq (1/4k^3)\epsilon nd] < k^{-cn}$$

for $c > 1$, and the claim is proved. ■ ■

In the following we show that we can perform queries into G' by accessing G , where every query can be performed using a constant number of queries into G .

Lemma 4.5.4 *Each of the possible queries into G' (degree, neighbor, vertex-pair queries) can be implemented using a constant number of queries into G .*

Proof: We consider each of the possible queries into G' .

Degree Queries. Here we assume that in G' the maximum degree is $2d$. It was proved before to be true with probability $1 - o(1)$. Consider the following procedure **Assign-Edges**(v) where v is a vertex of G : Find $\deg(v)$ by one query on G ; for each edge-index $r \in \{1, \dots, \deg(v)\}$, choose one of the vertices $X_i(v) \in X(v)$ ($1 \leq i \leq \lceil \deg(v)/d \rceil$), uniformly at random. Denote by $Q_i(v)$ a vector of $2d$ cells. This vector contains the indices of edges that are assigned to $X_i(v)$. Finally, choose an empty cell t in the vector $Q_i(v)$, and set $Q_i(v)[t] := r$. This procedure corresponds to the random construction of G' .

Neighbor Queries. For each vertex $X_i(v)$, the algorithm keeps a vector (of length $2d$), $W_i(v)$ that contains an ordered list of all the vertices of G' s.t. the algorithm is committed to the existence of an edge between them and $X_i(v)$. That is, if $W_i(v)[t] = X_j(u)$, it means that there is an edge $(X_i(v), X_j(u))$ in G' . In this case $W_j(u)[t'] = X_i(v)$ for some $t', 1 \leq t' \leq 2d$. Note that the procedure **Assign-Edges**(v) assigns each of the edges of v independently to one of the vertices $X_i(v)$, $1 \leq i \leq \lceil \deg(v)/d \rceil$.

Suppose that the algorithm is located in $X_i(v)$ and it wishes to perform a neighbor query. Choose uniformly at random, a number k , $1 \leq k \leq 2d$. Take an empty vector $Q_i(v)$ of length $2d$ and copy into it the vector $W_i(v)$. By that the algorithm keeps its commitment about the edges which have already been exposed. Then, apply the **Assign-Edges**(v) procedure only for edges of v , that do not appear in one of the $W_i(v)$, $1 \leq i \leq \lceil \deg(v)/d \rceil$ (i.e. to edges which were not exposed yet by the algorithm). If $Q_i(v)[k]$ is empty then the walk remains at $X_i(v)$. If $Q_i(v)[k]$ contains a name of a vertex in G' , the walk moves to that vertex, otherwise, $Q_i(v)[k]$ contains an index of a neighbor of v in G . By performing a single neighbor query on G , the algorithm determines the name of that neighbor (say) u in G . Then the walk needs to move to one of the vertices $X_j(u)$, $1 \leq j \leq \deg(u)/d$. It chooses one of the vertices $X_j(u)$ uniformly at random. For the chosen j set $W_i(v)[k] := X_j(u)$. Now choose a free cell k' , $1 \leq k' \leq 2d$, in $W_j(v)$, and set $W_j(v)[k'] := X_i(v)$. The neighbor query return $X_j(u)$.

Vertex-Pair Queries. In order to answer a vertex-pair query $(X_i(v), X_j(u))$ in G' , perform a vertex pair query (u, v) in G . If there is no edge between (u, v) in G , answer that there is no edge between $X_i(u)$ and $X_j(v)$, otherwise pick at random one of the copies (say k) of u in G' , one of the copies (say q) of v in G' . Now if $i = k$ and $j = q$ answer "yes", otherwise answer "no". ■

In the following we show that we can select a vertex (almost) uniformly at random from G' with a cost of $\tilde{O}(\min(\sqrt{n/\delta}, n/d))$ queries into G , for any given parameter $0 < \delta \leq 1$.

Lemma 4.5.5 *There exists a procedure Sample-Vertices-Almost-Uniformly-in- G' that for any given parameter $0 < \delta \leq 1$, performs $\tilde{O}(\min(\sqrt{n/\delta}, n/d))$ queries in G and returns a vertex in G' such that the following holds: For all but at most $\delta n'$ of the vertices x in G' , the probability that x is selected by the procedure is $\Omega(1/n')$.*

Proof: Hereby we describe the required procedure:

Sample-Vertices-Almost-Uniformly-in- $G'(d, \delta)$

1. If $d > \sqrt{\delta n}$ then sample an edge $e \in E(G)$ by running the procedure Sample-Edges-Uniformly-in- G (see Chapter 6). In case the procedure fails, pick an arbitrary edge e in $E(G)$.
2. Else ($d \leq \sqrt{\delta n}$), sample an edge $e \in E(G)$ by running the procedure Sample-Edges-Almost-Uniformly-in- $G(\delta/2)$ (see Chapter 6).
3. Choose with equal probability one of the end-points v of the edge e .
4. Choose uniformly at random one of the vertices x in $X(v)$.

Figure 4.2: A procedure for selecting a vertex in G' so that all but at most a δ -fraction of the vertices are selected with probability $\Omega(1/n')$.

We now show that the procedure Sample-Vertices-Almost-Uniformly-in- G' (see Figure 4.2) is as required by the lemma.

For an edge $e \in E(G)$, let C_e denote the event that e is selected by Sample-Edges-Uniformly-in- G in case $d > \sqrt{\delta n}$, or by Sample-Edges-Almost-Uniformly-in- G in case $d \leq \sqrt{\delta n}$. For $v \in V(G)$ let C_v denote the event that v is selected in Step 3 of procedure Sample-Vertices-Almost-Uniformly-in- G' . For $x \in V(G')$ let C_x denote the event that x is selected in Step 4 of the procedure and let $v(x)$ be such that $x \in X(v(x))$. Recall that $|X(v)| = \lceil \deg(v)/d \rceil$. Therefore for every $x \in V(G')$,

$$\Pr[C_x] \geq \Pr[C_{v(x)}] \cdot \frac{1}{\lceil \deg(v(x))/d \rceil} = \sum_{e=(u,v(x))} \frac{1}{2} \Pr[C_e] \cdot \frac{1}{\lceil \deg(v(x))/d \rceil}. \quad (4.3)$$

If $d > \sqrt{\delta n}$ then $\Pr[C_e]$ is only slightly smaller than $1/m$ (since there is a probability that the procedure Sample-Edges-Uniformly-in- G fails to output an edge), implying that $\Pr[C_x] = \Omega(1/n')$. If $d \leq \sqrt{\delta n}$, then $\Pr[C_e]$ is determined by the procedure Sample-Edges-Almost-Uniformly-in- G . Let U_0 be as defined in Theorem 6.2.1, and consider first the case where $v(x) \notin U_0$. Then for every edge e that is incident to $v(x)$, we have that $\Pr[C_e] \geq 1/(64m) = 1/(32dn)$. By Equation (4.3), for each such vertex x we have that

$$\Pr[C_x] \geq \frac{1}{2} \deg(v(x)) \cdot \frac{1}{32dn} \cdot \frac{1}{\lceil \deg(v(x))/d \rceil} \geq \frac{1}{128n} \geq \frac{1}{128n'}. \quad (4.4)$$

as required. Next consider the case that $v(x) \in U_0$ but $\deg(v(x)) \geq 2|U_0|$. In such a case for at least half of the edges e incident to $v(x)$ we have that $\Pr[C_e] \geq 1/(32dn)$, and we can deduce that $\Pr[C_x] \geq \frac{1}{256n'}$.

It remains to show that the total number of vertices x such that $v(x) \in U_0$ and $\deg(v) \leq 2|U_0|$, is at most $(\delta/2)n'$. Using the fact that $|U_0| \leq \sqrt{(\delta/2)n}/2$ (recall that the procedure `Sample-Edges-Almost-Uniformly-in-G` is called with its input parameter set to $\delta/2$), and for every vertex $v \in V(G)$, $|X(v)| = \lceil \deg(v)/d \rceil \leq \deg(v)$, we get:

$$\sum_{v \in U_0: \deg(v) \leq 2|U_0|} |X(v)| \leq 2|U_0|^2 \leq (\delta/2)n'. \quad (4.5)$$

Thus, the lemma follows. ■

Proof of Theorem 4.5.1: First note that nd is a trivial upper bound obtained by picking all possible vertices and asking all possible neighbor queries from each of them. Given a general graph G to be tested for k -colorability, we apply the algorithm from Theorem 4.4.1, on G' , which is the "regularized" version of G (described in Lemma 4.5.1). Recall that the algorithm from Theorem 4.4.1 selects $\Theta(k \ln k \epsilon^{-2} \frac{n}{d})$ vertices from the input graph (first stage) and then asks all vertex pair queries between them or all neighbor queries that are relevant to them (second stage). By Lemma 4.5.5, the cost of the first stage of the algorithm into G' is $O(\min((\frac{n}{d})^2, \frac{n \cdot \sqrt{n}}{d}) \cdot \text{poly}(k/\epsilon))$. By Lemma 4.5.4, the cost of the second stage of the algorithm into G' is $O(\min((\frac{n}{d})^2, \frac{n \cdot d}{d}) \cdot \text{poly}(k/\epsilon))$. Thus, the total cost of the upper bound for the general case is $O(\min((\frac{n}{d})^2, \frac{n \cdot \sqrt{n}}{d}, nd) \cdot \text{poly}(k/\epsilon))$. ■

Chapter 5

Testing Subgraph-Freeness in General Graphs

5.1 Introduction

In this chapter we consider the problem of testing subgraph-freeness, and in particular triangle-freeness, in general graphs. Let n denote the number of vertices in the graph, let d denote the average degree, and let d_{\max} denote the maximum degree. Given a distance parameter $\epsilon > 0$, we would like to design an algorithm that distinguishes with high probability between the case that the graph contains no triangles and the case in which more than $\epsilon \cdot nd$ edges should be removed so that no triangles remain. To this end we allow the algorithm query access to the graph. In particular, for any vertex of its choice, the algorithm may ask for the degree of the vertex, it may ask for any neighbor of the vertex, and it may ask whether there is an edge between any two vertices.

Subgraph-freeness, and more specifically, triangle-freeness, is one of the most basic problems studied in property testing. The interest in this problem is both due to the fact that triangle-freeness is a fundamental and simple graph property, and it is due to the relation between triangle-freeness and the study of dense sets of integers with no three-term arithmetic progression.

Dense graphs. Most of the focus in previous works was on testing triangle-freeness in dense graphs, that is, when $d = \Theta(n)$. The authors of [3] showed that it is possible to test triangle-freeness in dense graphs using a number of queries that is *independent* of n , and is of tower-type behavior in $1/\epsilon$. Alon [2] proved that a super-polynomial dependence on $1/\epsilon$ is necessary for testing subgraph-freeness of all non-bipartite subgraphs. When the subgraph is bipartite then $O(1/\epsilon)$ queries suffice [2]. It is also observed in [2] that the problem of testing triangle-freeness is intimately related to the famous (and very hard) problem of the existence of dense sets of integers without a three-term arithmetic progression. Alon's lower bound, which was proved for one-sided error algorithms, was extended in [10] to two-sided error algorithms. Other related results include [9].

Bounded-Degree graphs. In the other extreme, as was observed in [39], when $d_{\max} = O(1)$ then $O(1/\epsilon)$ queries suffice for testing triangle-freeness. More generally, $O(d^\tau/\epsilon)$ queries suffice for testing H -freeness in graphs with maximum degree $O(d)$, where τ is the diameter of H .

General graphs. In this work we study the complexity of testing triangle-freeness of graphs that lie between the two extremes. Namely, we would like to understand the dependence of the query complexity on the average degree d , and we do not want to necessarily assume that $d_{\max} = O(d)$. In the latter aspect we follow the work [64] on testing the diameter of sparse, but unbounded-degree, graphs, and in both aspects we follow the work presented in Chapter 3, on testing bipartiteness of general graphs.

Our contributions. The main contributions of this chapter, on a qualitative level, are as follows:

- We discover a threshold-type behavior in testing H -freeness, for every non-bipartite fixed graph H : whenever $d = O(n^{1-o(1)})$, the number of queries that are necessary to test H -freeness is $\Omega(n^{1/3})$, while, as discussed above, for $d = \Theta(n)$ the query complexity is a function of ϵ only. This is in sharp contrast with the results of Chapters 3, 4, where a smooth behavior of the complexity of testing bipartiteness and k -colorability as a function of d were described;
- We provide a transformation from lower bounds for testing H -freeness using one-sided error algorithms to those for two-sided error algorithms; though the suggested transformation carries some technical restrictions, it is general enough to capture a variety of lower bounds of this sort;
- We give quantitative lower and upper bounds for testing triangle-freeness in general graphs;
- We show that the edge distribution in random Cayley graphs is close to that of truly random graphs of the same edge density. This is proven by direct combinatorial and probabilistic arguments, without relying on the eigenvalue machinery, which is incapable of proving such results for subsets that are too small. Although we need this result for property testing purposes, we feel it is of enough independent interest to be stated here.

5.1.1 A lower bound and a sharp threshold

Our main result is:

Theorem 5.1.1 *There is a lower bound of $\Omega(n^{1/3})$ for testing triangle-freeness in general graphs. The lower bound holds for algorithms that are allowed two-sided error, and for every d that is upper bounded by $n^{1-o(1)}$. For some values of d the lower bound reaches $\Omega(n^{1/2})$.*

We note that the expression for $o(1)$ in the theorem is the function $\frac{\log \log \log n + 4}{\log \log n}$. Theorem 5.1.1 is actually the union of three lower bounds (whose one-sided error versions are stated in Lemmas 5.2.1, 5.3.1 and 5.4.1), which are applied to different values of d . The exact expression for the lower bound is

$$\Omega \left(\max \left\{ \sqrt{n/d}, \min\{d, n/d\}, \min \left\{ \sqrt{d}, n^{2/3}/d^{1/3} \right\} \cdot n^{-o(1)} \right\} \right) \quad (5.1)$$

For a schematic illustration see Figure 5.1.

Recall that when $d = \Theta(n)$ then testing can be performed using a number of queries that is independent of n [3]. Thus we observe a sharp transition between our lower bound of $\Omega(n^{1/3})$ that holds until $d = n^{1-o(1)}$, and the upper bound at $d = \Theta(n)$, which does not depend on n . The exact behavior of the complexity of testing triangle-freeness when $n^{1-o(1)} \leq d \leq n$ remains open.

Using techniques that were previously applied in [2] it is possible to extend Theorem 5.1.1 to testing subgraph-freeness of other non-bipartite subgraphs.

Theorem 5.1.2 *There is a lower bound of $\Omega(n^{1/3})$ for testing subgraph-freeness of non-bipartite subgraphs in general graphs. The lower bound holds for algorithms that are allowed two-sided error, and for every d that is upper bounded by $n^{1-o(1)}$. For some values of d the lower bound reaches $\Omega(n^{1/2})$.*

We wish to note that the difference between the complexity of testing bipartite and non-bipartite graphs is caused by the difference in the behavior of their Turán numbers – they are subquadratic for the former and quadratic for the latter. A more detailed discussion can be found in [2].

5.1.2 Upper bounds

We show that for every graph density, there exists an algorithm for testing triangle-freeness whose query complexity is sublinear in n . Furthermore, the upper bound is always at most quadratic in the corresponding lower bound.

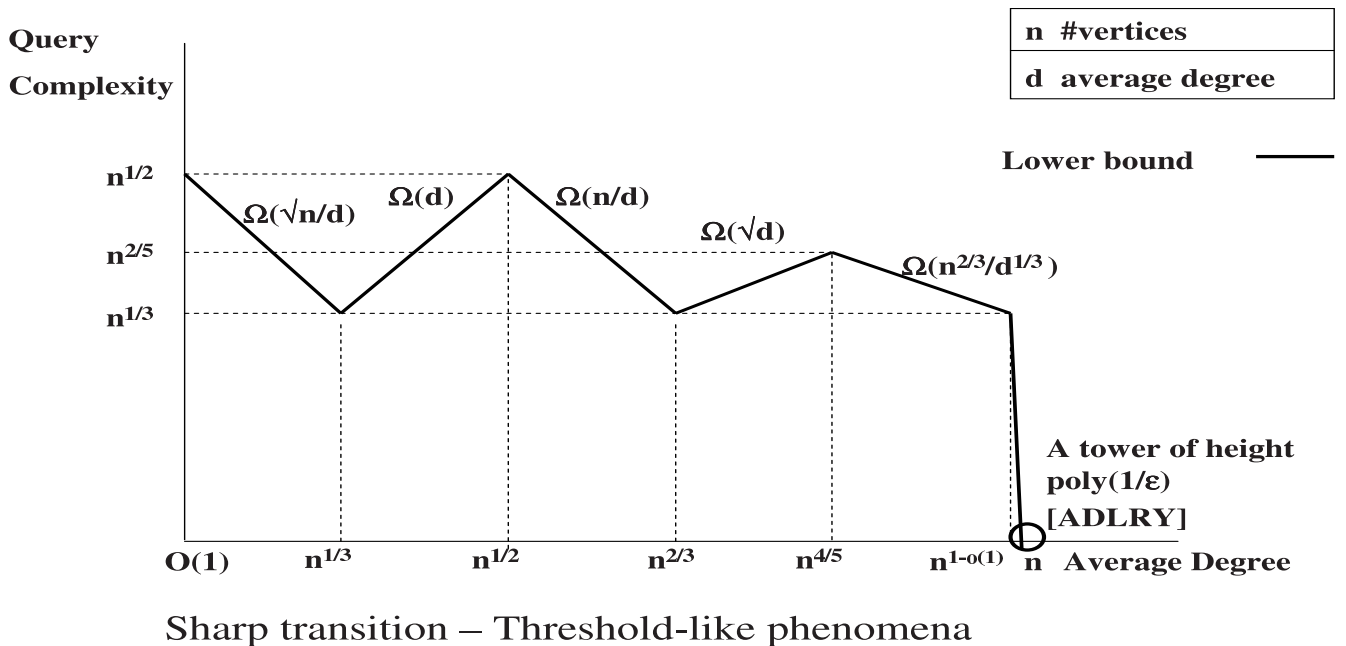


Figure 5.1: Testing H -Freeness - illustration of results for general graphs obtained in this chapter. Notice that the lower bound lies entirely above the horizontal line at height $n^{1/3}$.

Theorem 5.1.3 *There is an upper bound of $O(n^{6/7})$ for testing triangle-freeness in general graphs for every value of d . The upper bound can go down to $O(n^{1/2})$ for some values of d . In all cases the upper bound is at most quadratic in the lower bound that holds for that density. If $d_{\max} = O(d)$ then the upper bound is $O(n^{4/5})$ for all values of d .*

The exact expression for our upper bound is $O\left(\min\left\{\sqrt{nd}/\epsilon^{3/2}, (n^{4/3}/d^{2/3})/\epsilon^2\right\}\right)$, where in the case that $d_{\max} = O(d)$, the first term is replaced by d/ϵ .

Tight results

There are two cases in which our lower and upper bounds are tight. The first case is graphs in which $d_{\max} = O(d)$ and $d \leq \sqrt{n}$. For this case the complexity is $\Theta(d)$ (for constant ϵ). The second case is general sparse graphs, that is, graphs for which $d = \Theta(1)$. For these graphs the complexity is $\Theta(\sqrt{n})$.

5.1.3 Our techniques

Behrend Graphs and Cayley graphs. In the proof of our third lower bound (Lemma 5.4.1), we build on graphs that are known as Behrend graphs, which were previously used in the context of testing triangle-freeness in [2]. Here we prove that random Behrend graphs have a certain property that we can exploit in order to obtain our lower bound. Behrend graphs are variants of the well studied Cayley graphs, and our proof concerning properties of random Behrend graphs extends to Cayley graphs. See Chapter 7 for relevant results for Cayley graphs.

A reduction from one-sided error lower bounds to two-sided error lower bounds. We obtain our two main lower bounds by first establishing lower bounds that hold for one-sided error algorithms. We then prove a transformation from one-sided error lower bounds to two-sided error lower bounds that hold under certain assumptions, and apply it to obtain our two-sided error lower bounds. This transformation may be of use in future lower bound proofs for subgraph freeness. We note that in [10] a transformation was given in the case of dense graphs, but it is not applicable in general.

5.2 A lower bound of $\Omega\left(\sqrt{n/d}\right)$

In this section we establish our first, and simplest lower bound.

Lemma 5.2.1 *Every algorithm for testing triangle-freeness must perform $\Omega(\sqrt{n/d})$ queries. This lower bound holds for two-sided error algorithms as well.*

Proof: Consider the following two families of graphs over n vertices and with average degree d . Each family is determined by a single graph, and consists of all possible $n!$ labelings of the vertices of the graph. Hence it suffices to describe the two graphs (one per family). In one graph there is a clique of size \sqrt{nd} , and in the other graph there is a complete bipartite graph between two sets of vertices, each of size \sqrt{nd} . In addition, in both graphs the remaining set of vertices spans a d -regular triangle-free graph. The second graph is clearly triangle-free, and it is not hard to verify that the first graph is $(1/c)$ -far from being triangle-free for some constant c . However, in order to distinguish between the two graphs (or more precisely, in order to distinguish between graphs that are selected uniformly from each of the two families), the algorithm must obtain a vertex in the clique / complete bipartite subgraph. To this end the algorithm must perform $\Omega(n/\sqrt{nd}) = \Omega(\sqrt{n/d})$ queries. ■

5.3 A lower bound of $\Omega(\min\{d, n/d\})$

Our next lower bound improves on the lower bound in Section 5.2 when $d > n^{1/3}$.

Lemma 5.3.1 *Every one-sided error algorithm for testing triangle-freeness must perform $\Omega(\min\{d, n/d\})$ queries. This lower bound holds even when $d_{\max} = O(d)$.*

The lower bound in Lemma 5.3.1 is extended to two-sided error algorithms in Section 5.5. We start by considering the case that $d = c \cdot \sqrt{n}$ for a particular constant $c < 1$. We later discuss how to deal with the case that $d > c \cdot \sqrt{n}$ and with the case that $d < c \cdot \sqrt{n}$. We prove the lemma by describing a distribution on graphs such that the following holds: On one hand almost all of its support is on graphs that are far from triangle free. On the other hand, every algorithm that uses neighbor and/or vertex-pair queries, must perform $\Omega(\min\{d, n/d\})$ queries before it views a triangle (with sufficiently high constant probability). Since we currently focus on testing algorithms that have one-sided error, this implies a lower bound on the query complexity of such algorithms.

5.3.1 Definition of the lower-bound distribution

The graphs we consider below are d -regular, but may have multiple edges. At the end of this section we discuss how to remove the multiple edges. Let D_{Δ} be a distribution over graphs with n vertices and degree $d = \frac{2}{3}\sqrt{n/3}$ that is defined as follows. A graph is generated by first selecting a random 3-way partition of the vertices into equal-size subsets of size $n' = n/3$ denoted V_1, V_2, V_3 . Next, between each pair of subsets, $d' = d/2 = \sqrt{n'}/3$ random perfect matching are selected. In all that follows we assume that n' is sufficiently large ($n' > 100$ suffices).

Lemma 5.3.2 *With probability at least $1 - O(1/n')$, a graph chosen uniformly according to the distribution D_Δ is $\Omega(1)$ -far from being triangle free.*

Proof: We shall show that with high probability over the choice of a graph according to D_Δ , there are at least $\alpha \cdot nd$ edge-disjoint triangles in the graph, for some constant α .

Consider first a fixed choice of a pair of vertices v_i, v_j . The probability that there is an edge between v_i and v_j is $1 - (1 - \frac{1}{n'})^{d'}$ which is at most $\frac{d'}{n'}$ and at least $\frac{d'}{n'} - \binom{d'}{2} \cdot \frac{1}{(n')^2} \geq \frac{d'}{n'} - \frac{1}{18n'}$. Next consider any choice of three vertices $v_{1,i} \in V_1, v_{2,j} \in V_2$ and $v_{3,k} \in V_3$. Let $\alpha_{i,j,k}$ denote the 0/1 random variable that is 1 if and only if there is a triangle between the three vertices. Since the choice of the matchings between V_1 and V_2 is independent of the choice of the matchings between V_2 and V_3 and the choice of the matching between V_1 and V_3 ,

$$\Pr[\alpha_{i,j,k} = 1] \leq \left(\frac{d'}{n'}\right)^3 \quad (5.2)$$

and

$$\Pr[\alpha_{i,j,k} = 1] \geq \left(\frac{d' - 1/18}{n'}\right)^3 = \left(\frac{d'}{n'}\right)^3 \cdot \left(1 - \frac{1}{18d'}\right)^3 \quad (5.3)$$

Let $\beta_{i,j,k}$ denote a 0/1 random variable that is 1 if there is *no* triangle of the form (v'_i, v_j, v_k) , where $v'_i \neq v_i$, or (v_i, v'_j, v_k) where $v'_j \neq v_j$, or (v_i, v_j, v'_k) where $v'_k \neq v_k$, and is 0 if there is such a triangle. Then

$$\Pr[\alpha_{i,j,k} = 1 \text{ and } \beta_{i,j,k} = 0] \leq \left(\frac{d'}{n'}\right)^3 \cdot 3 \cdot (n' - 1) \cdot \left(\frac{d' - 1}{n'}\right)^2 \leq \left(\frac{d'}{n'}\right)^3 \cdot \frac{3(d' - 1)^2}{n'} \quad (5.4)$$

It follows that

$$\begin{aligned} & \Pr[\alpha_{i,j,k} = 1 \text{ and } \beta_{i,j,k} = 1] \\ &= \Pr[\alpha_{i,j,k} = 1] - \Pr[\alpha_{i,j,k} = 1 \text{ and } \beta_{i,j,k} = 0] \end{aligned} \quad (5.5)$$

$$\geq \left(\frac{d' - 1/18}{n'}\right)^3 - \left(\frac{d'}{n'}\right)^3 \cdot \frac{3(d' - 1)^2}{(n')^2} \quad (5.6)$$

$$= \left(\frac{d'}{n'}\right)^3 \cdot \left(\left(1 - \frac{1}{18d'}\right)^3 - 3 \cdot \left(1 - \frac{1}{d'}\right)^2 \cdot \frac{(d')^2}{(n')^2} \right) \quad (5.7)$$

$$\geq \frac{1}{2} \cdot \left(\frac{d'}{n'}\right)^3 = \frac{1}{54} \cdot (n')^{-3/2} \quad (5.8)$$

where we have used our assumption that n' and $d' = \sqrt{n'}/3$ are sufficiently large. Hence, the expected number of triples (v_1, v_2, v_3) that constitute a triangle that does not share an edge with any other triangle is at least

$$(n')^3 \cdot \frac{1}{54} \cdot (n')^{-3/2} = \frac{1}{54} \cdot (n')^{3/2} = \frac{1}{54} \cdot n \cdot d \quad (5.9)$$

Next, we shall use Chebychev's inequality to show that with sufficiently high probability, the number of such triples is not much smaller than this expected value.

Let the vertices in each set V_ℓ , $\ell \in \{1, 2, 3\}$, be denoted $v_{\ell,1}, \dots, v_{\ell,n'}$. For each triple $v_{1,i}, v_{2,j}, v_{3,k}$ let $\eta_{i,j,k}$ be the 0/1 random variable that is 1 if and only if $(v_{1,i}, v_{2,j}, v_{3,k})$ constitute a triangle, and there is no other triangle that shares any of its edges. Using our previous notation we have that $\Pr[\eta_{i,j,k} = 1] = \Pr[\alpha_{i,j,k} = 1 \text{ and } \beta_{i,j,k} = 1]$. Clearly, $\Pr[\eta_{i,j,k} = 1] \leq \Pr[\alpha_{i,j,k} = 1]$, which we have shown is upper

bounded by $\left(\frac{d'}{n'}\right)^3$. By Chebychev's inequality,

$$\Pr \left[\sum_{i,j,k} \eta_{i,j,k} < \frac{1}{2} \text{Exp} \left[\sum_{i,j,k} \eta_{i,j,k} \right] \right] \leq \frac{4 \text{Var} \left[\sum_{i,j,k} \eta_{i,j,k} \right]}{\left(\text{Exp} \left[\sum_{i,j,k} \eta_{i,j,k} \right] \right)^2} < \frac{c' \text{Var} \left[\sum_{i,j,k} \eta_{i,j,k} \right]}{(n')^3} \quad (5.10)$$

for $c' = 4 \cdot 54^2$. We would like to bound the variance of $\sum_{i,j,k} \eta_{i,j,k}$:

$$\text{Var} \left[\sum_{i,j,k} \eta_{i,j,k} \right] = \text{Exp} \left[\left(\sum_{i,j,k} \eta_{i,j,k} \right)^2 \right] - \left(\text{Exp} \left[\sum_{i,j,k} \eta_{i,j,k} \right] \right)^2 \quad (5.11)$$

Now,

$$\begin{aligned} \text{Exp} \left[\left(\sum_{i,j,k} \eta_{i,j,k} \right)^2 \right] &= \sum_{i,j,k} \sum_{i',j',k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}] \\ &= \sum_{i \neq i'} \sum_{j \neq j'} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}] \\ &\quad + 3 \cdot \sum_i \sum_{j \neq j'} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i,j',k'}] \\ &\quad + 3 \cdot \sum_{i,j} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i,j,k'}] \\ &\quad + \sum_{i,j,k} \text{Exp}[\eta_{i,j,k}^2] \end{aligned} \quad (5.12)$$

while

$$\left(\text{Exp} \left[\sum_{i,j,k} \eta_{i,j,k} \right] \right)^2 = \sum_{i,j,k} \sum_{i',j',k'} \text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}] \quad (5.14)$$

$$> \sum_{i \neq i'} \sum_{j \neq j'} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}] \quad (5.15)$$

Therefore,

$$\begin{aligned} \text{Var} \left[\sum_{i,j,k} \eta_{i,j,k} \right] &\leq \sum_{i \neq i'} \sum_{j \neq j'} \sum_{k \neq k'} \left(\text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}] - \text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}] \right) \\ &\quad + 3 \cdot \sum_i \sum_{j \neq j'} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i,j',k'}] + 3 \cdot \sum_{i,j} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i,j,k'}] + \sum_{i,j,k} \text{Exp}[\eta_{i,j,k}^2] \end{aligned} \quad (5.16)$$

First observe that by definition of η , for every i, j and $k \neq k'$, since if both $(v_{1,i}, v_{2,j}, v_{3,k})$ and $(v_{1,i}, v_{2,j}, v_{3,k'})$ are triangles then they share an edge, we get that

$$\eta_{i,j,k} \cdot \eta_{i,j,k'} = 0 \quad (5.17)$$

Next observe that

$$\sum_{i,j,k} \text{Exp}[\eta_{i,j,k}^2] = \sum_{i,j,k} \text{Exp}[\eta_{i,j,k}] \leq (n')^3 \cdot (d'/n')^3 = (d')^3 \quad (5.18)$$

As for the sum over triangles that share a (single) vertex,

$$\text{Exp}[\eta_{i,j,k} \cdot \eta_{i,j',k'}] \leq \text{Exp}[\alpha_{i,j,k} \cdot \alpha_{i',j',k'}] \leq \left(\frac{d'}{n'} \cdot \frac{d'-1}{n'}\right)^2 \cdot \frac{d'}{n'} \cdot \left(\frac{d'-1}{n'} + \frac{1}{n'-1}\right) \leq 2 \cdot \left(\frac{d'}{n'}\right)^6 \quad (5.19)$$

and so

$$\sum_i \sum_{j \neq j'} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i,j',k'}] \leq (n')^5 \cdot 2 \cdot \frac{(d')^6}{(n')^6} = 2 \frac{(d')^6}{n'} \quad (5.20)$$

To complete the proof we bound the difference between $\text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}]$ and $\text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}]$ (for $i \neq i', j \neq j',$ and $k \neq k'$). By definition,

$$\begin{aligned} & \text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}] - \text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}] \\ &= \Pr[\eta_{i,j,k} = 1] \cdot (\Pr[\eta_{i',j',k'} = 1 | \eta_{i,j,k} = 1] - \Pr[\eta_{i',j',k'} = 1]) \end{aligned} \quad (5.21)$$

It can be verified that

$$\Pr[\eta_{i',j',k'} = 1 | \eta_{i,j,k} = 1] = \left(1 + O\left(\frac{1}{n'}\right)\right) \cdot \Pr[\eta_{i',j',k'} = 1] \quad (5.22)$$

and since $\Pr[\eta_{i,j,k} = 1] = \Pr[\eta_{i',j',k'} = 1] \leq \left(\frac{d'}{n'}\right)^3$, we get that

$$\text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}] - \text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}] = O\left(\frac{(d')^6}{(n')^7}\right) \quad (5.23)$$

Hence,

$$\sum_{i \neq i'} \sum_{j \neq j'} \sum_{k \neq k'} \text{Exp}[\eta_{i,j,k} \cdot \eta_{i',j',k'}] - \text{Exp}[\eta_{i,j,k}] \cdot \text{Exp}[\eta_{i',j',k'}] = O\left((n')^6 \cdot \frac{(d')^6}{(n')^7}\right) = O\left(\frac{(d')^6}{n'}\right) \quad (5.24)$$

By combining Equation (5.10) with Equations (5.16)–(5.24) we get that

$$\Pr\left[\sum_{i,j,k} \eta_{i,j,k} < \frac{1}{2} \text{Exp}\left[\sum_{i,j,k} \eta_{i,j,k}\right]\right] = \frac{O((d')^6/n') + O((d')^3)}{(n')^3} = o(1). \quad (5.25)$$

■

The case $d > \frac{2}{3\sqrt{3}}\sqrt{n}$. The distribution in this case is the same as described at the start of this section: the graph vertices are partitioned into three equal parts $V_1, V_2,$ and $V_3,$ and between every pair V_i and V_j we put $d/2$ random perfect matchings. We would like to show that with high probability the resulting graph contains at least $\frac{1}{c} \cdot nd$ edge-disjoint triangles in the graph, for some constant c . To this end we think of the matchings as being selected in $k = \frac{3\sqrt{3}}{2} \cdot \frac{d}{\sqrt{n}}$ rounds, where in each round $d' = \frac{1}{3\sqrt{3}} \cdot \sqrt{n}$ matching are selected between every pair $V_i, V_j, i \neq j$. For each round we can apply Lemma 5.3.2 and get that with high probability we have at least $\frac{1}{c} \cdot n \cdot d'$ edge-disjoint triangles. Observe that the triangles created in the different rounds are edge-disjoint. By applying a union bound we get that with probability at least $1 - O(k/n)$ we obtain $\frac{1}{c} \cdot n \cdot d$ edge-disjoint triangles.

The case $d < \frac{2}{3\sqrt{3}}\sqrt{n}$. In this case we first partition the vertices into k parts V^1, \dots, V^k where $|V^i| = \frac{27}{4}d^2$. We then apply the construction described at the start of this section to each V^i . For small k (i.e., $k < \log n$) we can apply a union bound on the different V^i 's, and once k is sufficiently large we can use the fact that the different subgraphs are constructed independently. In either case we get that with high probability, for at least a half of the V^i 's there are $\Omega(|V^i| \cdot d)$ edge disjoint triangles within the subgraph induced by V^i .

5.3.2 The lower bound

Let A be any one-sided error algorithm for testing triangle freeness, where A is allowed to perform both neighbor queries and vertex-pair queries. If A views a triangle in the tested graph then clearly it can reject the graph. However, since A is a one-sided error algorithm, if it terminates before viewing a triangle, then it must accept. Suppose we run A on a graph chosen according to D_Δ , with some (sufficiently small) constant ϵ . Since we have shown that with high probability such a graph is ϵ -far from being triangle free, the probability that A terminates before viewing a triangle must be small. Hence it remains to prove the following lemma.

Lemma 5.3.3 *Any algorithm whose goal is to detect, with high constant probability, a triangle in a graph selected according to D_Δ , must ask $\Omega(\min\{d, n/d\})$ queries.*

We note that the proof actually establishes the stronger statement: $\Omega(\min\{d, n/d\})$ queries are required to detect a cycle of any length.

Proof: We first present the argument for $d \geq \frac{2}{3\sqrt{3}}\sqrt{n}$ and later discuss the modifications required for $d < \frac{2}{3\sqrt{3}}\sqrt{n}$.

It will be convenient to view graphs in the support of D_Δ as being represented by “matchings over tables”. Namely, there are 6 tables: $T_{1,2}, T_{1,3}, T_{2,1}, T_{2,3}, T_{3,1}, T_{3,2}$, two for each set of vertices V_b , $b \in \{1, 2, 3\}$. Each table $T_{b,b'}$ is of size $(n/3) \times (d/2)$: there is a row for each vertex v in V_b , and each entry in v 's row corresponds to one of the $d/2$ edges that are incident to v and to vertices in $V_{b'}$. An edge between $u \in V_b$ and $v \in V_{b'}$ is represented by a pair of entries $(T_{b,b'}[u][i], T_{b',b}[v][j])$. Hence the $d/2$ perfect matchings between V_b and $V_{b'}$ correspond to a single perfect matching between the entries of the two tables $T_{b,b'}$ and $T_{b',b}$.

Let ALG be an algorithm that performs $Q = Q(n, d)$ queries and whose goal is to detect a triangle with probability at least $9/10$. The probability is taken over the choice of the graph G in the support of D_Δ and the coin flips of the algorithm. Namely ALG is a (possibly probabilistic) mapping from *query-answer histories* $\langle (q_1, t_1), \dots, (q_t, a_t) \rangle$, to q_{t+1} for every $t < Q$. A vertex-pair query is of the form $q_t = (u, v)$ where $u \in V_b$ and $v \in V_{b'}$ for some $b \neq b'$, $b, b' \in \{1, 2, 3\}$. The answer is either $a_t = (1, i, j)$, which denotes that there is an edge between u and v and it corresponds to the pair of entries $(T_{b,b'}[u][i], T_{b',b}[v][j])$, or $a_t = 0$, which denotes that there is no edge between u and v . A neighbor query is of the form $q_t = (u, b', i)$ where $u \in V_b$, $b \neq b'$. The answer is of the form $a_t = (v, j)$ where $v \in V_{b'}$ denoting that there is an edge between u and v and it corresponds to the pair of entries $(T_{b,b'}[u][i], T_{b',b}[v][j])$. We note that the mapping from query-answer histories to queries needs to be defined only on histories that are consistent with some graph in the support of D_Δ .

In what follows we define a process that answers the queries of the algorithm while generating a graph according to D_Δ . At any time t , the queries of the algorithms and the answers it is provided with determine the *knowledge graph* $G^t = (V^t, E^t, \bar{E}^t)$, where V^t are the vertices, E^t are the edges and \bar{E}^t are the non-edges. Namely, V^t consists of all vertices that appeared in queries of the algorithm or in answers to neighbor queries. Similarly E^t consists of all pairs u, v such that either (u, v) was a vertex-pair query that was answered positively, or v was an answer to a neighbor query involving u , and \bar{E}^t consists of all pairs u, v

such that either (u, v) was a vertex-pair query that was answered negatively. For every edge $(u, v) \in E^t$ the knowledge graph will include the indices of the entries in the tables by which u and v are connected (that is, (i, j) such that $T_{b,b'}[u][i]$ is matched to $T_{b',b}[v][j]$). Given a vertex-pair query $q_t = (u, v)$ where $u \in V_b$ and $v \in V_{b'}$, the process computes the probability, conditioned on the current knowledge graph, that (u, v) is an edge. Namely, it considers all graphs in the support of D_Δ that are consistent with G^{t-1} and answers positively with probability that is proportional to the number of these graphs in which there is an edge between u and v . Let $u_{t-1,b,b'}$ denote the number of unmatched entries in $T_{b,b'}$ at time t (before the t 'th query). Note that this number equals the number of unmatched entries in $T_{b',b}$ at the same time.

It is not hard to verify that the conditional probability that there is an edge between u and v is upper bounded by: $d/2$ (the maximum number of unmatched entries in u 's row in $T_{b,b'}$), times $d/2$ (the maximum number of unmatched entries in v 's row in $T_{b',b}$), divided by $u_{t-1,b,b'}$. Since $u_{t-1,b,b'} \geq (n/3) \cdot (d/2) - (t-1)$ and $t-1 < Q = o(n/d)$ the probability is $o(d/n)$. It follows that the probability that the algorithm gets a positive answer for *any* of the at most $Q = o(n/d)$ vertex-pair queries is $o(1)$. But if the algorithm always gets negative answers for its vertex-pair queries it clearly cannot close a triangle with such a query.

Given a neighbor query (u, b', i) where $u \in V_b$, the process gives an answer (v, j) where $v \in V_{b'}$ according to the conditional probability that the entries $T_{b,b'}[u, i]$ and $T_{b',b}[v][j]$ are matched (where again, the conditioning is on the current knowledge graph G^{t-1}). Here we would like to upper bound the probability that v already belongs to the knowledge graph. Observe that the knowledge graph G^{t-1} contains at most $2t$ vertices and at most t edges and non-edges. Hence the conditional probability that $T_{b,b'}[u, i]$ is matched to $T_{b',b}[v, j]$ for some $v \in V_{b'}^{t-1} = V^{t-1} \cap V_{b'}$ and $j \in \{1, \dots, d/2\}$, is upper bounded by the following expression: $(d/2) \times (2t)$ (the maximum number of unmatched entries $T_{b',b}[v, j]$ for $v \in V_{b'}^{t-1}$) divided by the total number of unmatched entries in $T_{b',b}$ that do not belong to rows of vertices $u \in V_b$ such that $(u, v) \in \bar{E}^{t-1}$. By our assumption on Q (where $t-1 < Q$), the numerator in the above expression is $o(\min\{n, d^2\})$ and the denominator is at least $(n/3) \cdot (d/2) - (t-1) \cdot (d/2) = \Omega(n \cdot d)$. Hence, the probability that such an event occurs in a given particular time t is $o(\min\{(1/d), (d/n)\})$, and the probability that it occurs at any $t \leq Q$ is $o(1)$. But if the algorithm never gets a vertex in the knowledge graph as an answer to a neighbor query, it clearly cannot close a triangle with such a query.

Finally we address the case that $d < \frac{2}{3\sqrt{3}}\sqrt{n}$. Recall that in this case we first partition the vertices into k parts V^1, \dots, V^k where $|V^i| = n' = \frac{27}{4}d^2$. We then apply the construction to each V^i . By the argument described above, here we can get a lower bound of the form $\Omega(\min\{d, n'/d\}) = \Omega(d)$. But for the current setting of d we also have that $\min\{d, n'/d\} = d$, and so the lower bound holds in this case as well. ■

A remark about multiple edges. The lower bound proof stated above is valid for graphs that may contain multiple edges. In the distribution D_Δ the probability of a multiple edge between a pair of vertices is $O(\frac{d^2}{n^2})$. Thus, graphs created according to this distribution contain multiple edges with probability close to 1. However, with probability $1 - o(1)$ there are $O(d^2)$ multiple edges in the graphs created according to this distribution. We have shown in Lemma 5.3.2 that graphs created according to the distribution D_Δ are $\Omega(1)$ -far from being triangle free with probability $1 - o(1)$. Hence we can deduce that by removing multiple edges from a graph G constructed according to the distribution D_Δ , the resulting graph is $\Omega(1)$ -far from being triangle free with probability $1 - o(1)$. In addition, an algorithm ALG that interacts with the process detects a multiple edge with probability $o(1)$ due to the following reason: ALG doesn't detect any edges by vertex-pair queries with probability $1 - o(1)$. The probability of detecting a multiple edge in a neighbor query is at most the probability that such a query is answered by a vertex in the knowledge graph. This probability was shown in Lemma 5.3.2 to be $o(1)$. Thus, at the end of the interaction with ALG, the process can delete all the multiple edges from the resulting graph, so that the resulting graph contains no multiple edges. The graph remains $\Omega(1)$ -far from being triangle-free after the deletion, and its average degree is $d - o(1)$. As a conclusion we get that our lower bound is valid also for graphs with no multiple edges.

5.4 An improved lower bound for high degrees

In this section we establish the following lemma, which improves on our previous lower bound of $\min\{d, n/d\}$ when the degree of the graph is at least $n^{2/3+o(1)}$.

Lemma 5.4.1 *Every one-sided error testing algorithm for triangle-freeness must perform $\Omega\left(\min\left\{\sqrt{d}, \frac{n^{2/3}}{d^{1/3}}\right\} \cdot n^{-g(n)}\right)$ queries, where $g(n) = \frac{\log \log \log n + 4}{\log \log n}$. This lower bound holds even for d -regular graphs.*

In order to prove the lemma, here too we define a distribution over graphs that are far from being triangle free. We then prove a lower bound on the number of queries that are required in order to detect a triangle with probability bounded away from zero in a graph that is generated according to the distribution. As we shall see, it will actually be convenient to consider graphs over $3n$ vertices and degree $2d$.

5.4.1 A variant of Behrend graphs

Our lower bound distribution builds on graphs that are variants of what are known as *Behrend Graphs*. These graphs are defined by sets of integers that include no three-term arithmetic progression (abbreviated as 3AP). Namely, these are sets $X \subset \{1, \dots, n\}$ such that for every three elements $x_1, x_2, x_3 \in X$, if $x_2 - x_1 = x_3 - x_2$ (i.e., $x_1 + x_3 = 2x_2$), then necessarily $x_1 = x_2 = x_3$. Below we describe a construction of such sets that are large (relative to n), and later explain how such sets determine Behrend graphs. Our construction of X uses similar ideas to those used in known constructions [16, 67] and gives a slightly weaker result. However, our alternative construction appears to be somewhat simpler, and the size of the resulting set suffices for our purposes.

Lemma 5.4.2 *For every sufficiently large n there exists a set $X \subset \{1, \dots, n\}$, $|X| \geq n^{1 - \frac{\log \log \log n + 4}{\log \log n}}$, such that X contains no three-term arithmetic progression.*

Proof: Let $b = \log n$ and $k = \log n / \log b - 2$. Since $\log n / \log b = \log n / \log \log n$ we have that $k < b/2$ for every $n \geq 8$. We arbitrarily select a subset of k different numbers $\{x_1, \dots, x_k\} \subset \{0, \dots, b/2 - 1\}$ and define $X = \left\{ \sum_{i=1}^k x_{\pi(i)} b^i : \pi \text{ is a permutation of } \{1, \dots, k\} \right\}$. By the definition of X we have that $|X| = k!$. By using $z! > (z/e)^z$, we get that

$$|X| = k! = (\log n / \log \log n)! > n^{1 - \frac{\log \log \log n + 4}{\log \log n}} \quad (5.26)$$

Consider any three elements $u, v, w \in X$ such that $u + v = 2w$. By definition of X , these elements are of the form $u = \sum_{i=1}^k x_{\pi_u(i)} b^i$, $v = \sum_{i=1}^k x_{\pi_v(i)} b^i$ and $w = \sum_{i=1}^k x_{\pi_w(i)} b^i \in X$, where π_u, π_v, π_w are permutations over $\{1, \dots, k\}$. Since $x_i < b/2$ for every $1 \leq i \leq k$, it must be the case that for each i , $x_{\pi_u(i)} + x_{\pi_v(i)} = 2x_{\pi_w(i)}$. This implies that for every i : $x_{\pi_u(i)}^2 + x_{\pi_v(i)}^2 \geq 2x_{\pi_w(i)}^2$ where the inequality is strict unless $x_{\pi_u(i)} = x_{\pi_v(i)} = x_{\pi_w(i)}$. If we sum over all i 's and there is at least one index i for which the inequality is strict we get that $\sum_{i=1}^k x_{\pi_u(i)}^2 + \sum_{i=1}^k x_{\pi_v(i)}^2 > \sum_{i=1}^k 2x_{\pi_w(i)}^2$ which is a contradiction since we took permutations of the same numbers. Thus, we get that $u = v = w$. ■

Remark. In fact, the set constructed above is also 3AP-free when all calculations are performed modulo n . We will use this observation below.

Behrend graphs. Given a set $X \subset \{1, \dots, n\}$ with no three-term arithmetic progression we define the Behrend graph BG_X as follows. It has $3n$ vertices that are partitioned into three equal parts: V_1, V_2 , and V_3 . For each $i \in \{1, 2, 3\}$ we associate with each vertex in V_i a different integer in $\{0, \dots, n-1\}$. The edges of the graph are defined as follows:

- The edges between V_1 and V_2 : For every $x \in X$ and $j \in \{0, \dots, n-1\}$ there is an edge between $j \in V_1$ and $(j+x) \bmod n \in V_2$;
- The edges between V_2 and V_3 : For every $x \in X$ and $j \in \{0, \dots, n-1\}$ there is an edge between $(j+x) \bmod n \in V_2$ and $(j+2x) \bmod n \in V_3$;
- The edges between V_1 and V_3 : For every $x \in X$ and $j \in \{0, \dots, n-1\}$ there is an edge between $j \in V_1$ and $(j+2x) \bmod n \in V_3$.

We shall say that an edge between $j \in V_1$ and $j' \in V_2$ or between $j \in V_2$ and $j' \in V_3$ is *labeled* by x , if $j' = (j+x) \bmod n$, and we shall say that an edge between $j \in V_1$ and $j' \in V_3$ is *labeled* by x , if $j' = (j+2x) \bmod n$.

The graph BG_X is $2|X|$ -regular and it contains $3|X|n$ edges. For every $j \in \{0, \dots, n-1\}$ and $x \in X$, the graph contains a triangle $(j, (j+x) \bmod n, (j+2x) \bmod n)$ where $j \in V_1, (j+x) \bmod n \in V_2$ and $(j+2x) \bmod n \in V_3$. There are $n \cdot |X|$ such edge-disjoint triangles and every edge is part of one such triangle. Moreover, it is not hard to verify (based on the assumption that X is 3AP-free) that there are no other triangles in the graph.

5.4.2 The edge density of large sets in random Behrend graphs

In this subsection we prove the following lemma, which is central to the proof of Lemma 5.4.1. We shall use the following notation: For a subset $Y \subseteq X$ and a subset of vertices C in BG_Y , we let $e_Y(C)$ denote the number of edges spanned by C in BG_Y .

Lemma 5.4.3 *Let $0 < \beta < \frac{1}{2}$ and $0 < \alpha \leq 1$ be such that $\alpha - 2\beta > \frac{1}{\log \log n}$, and let $X \subset \{1, \dots, n\}$, $|X| \geq n^\beta$. Consider the random Behrend graph BG_Y obtained by choosing a random subset $Y \subseteq X$, $|Y| = d = \frac{|X|}{n^\beta}$. With high probability over the choice of Y , for every subset C of vertices in BG_Y where $|C| = n^\alpha$, we have $e_Y(C) \leq \frac{90}{\alpha-2\beta} \frac{n^{2\alpha}}{n^\beta}$ edges.*

The lemma states that for sufficiently large subsets C (i.e., for $|C| = n^\alpha$, where $\alpha - 2\beta$ is a constant), the number of edges $e_Y(C)$ is close to its expected value. Note that the smaller we choose β (i.e., the larger we choose Y), the smaller can α be. That is, the lemma can be applied to sets with smaller size.

Before proving the lemma we introduce some notation and prove two claims. For a subset $W \subseteq V_1 \cup V_2$, $|W| = s$, consider the subgraph of BG_X induced by W . Let

$$\Delta(W) = \{(j_2 - j_1) \bmod n : j_1 \in W_1, j_2 \in W_2, (j_2 - j_1) \bmod n \in X\} \quad (5.27)$$

denote the set of *differences* in W . That is, it is the set of labels of the edges between W_1 and W_2 in BG_X . Obviously, $|\Delta(W)| \leq s^2$. For every difference $x \in \Delta(W)$, we define the *multiplicity* of x in W as the number of edges in BG_X between vertices in W_1 and vertices in W_2 that are labeled by x .

Let $k = \frac{5}{\alpha-2\beta}$. For β and α that satisfy the condition of the lemma ($\alpha - 2\beta > \frac{1}{\log \log n}$) we have that $k \leq 5 \log \log n$. We shall say that W is *good* if no difference in $\Delta(W)$ has multiplicity higher than k in W .

Claim 5.4.4 *With high probability over the choice of $Y \subseteq X$, for every good W such that $|W| = s \geq n^\beta \log n$, we have that $e_Y(W) \leq \frac{2ks^2}{n^\beta}$.*

Claim 5.4.4 is easily established by using known bounds on the tail of the Hypergeometric distribution (see, e.g., [44, Page 29]). We also need the following claim.

Claim 5.4.5 Let C be a subset of $V_1 \cup V_2$, such that $|C| = n^\alpha$. Suppose we uniformly and independently select $W \subset C$, $|W| = n^\beta \log n$. Then the probability that W is not good is at most $\frac{1}{n^\beta}$.

Proof: Note that by the definition of Behrend graphs, the edges between vertices in C that are labeled by a specific difference, form a matching. When we choose a random subset $W \subset C$, the probability that there exists a single difference of C (i.e. an element of $\Delta(C)$) that has multiplicity at least $k+1$ in W is bounded by $|C|^2 |C|^{k+1} \binom{|C|-(2k+2)}{|W|-(2k+2)} \binom{|C|}{|W|}^{-1}$. Now,

$$\frac{|C|^2 |C|^{k+1} \binom{|C|-(2k+2)}{|W|-(2k+2)}}{\binom{|C|}{|W|}} \leq |C|^2 \cdot |C|^{k+1} \cdot \left(\frac{|W|}{|C|} \right)^{2k+2} \leq n^{2\alpha} \left(\frac{\log^2 n}{n^{\alpha-2\beta}} \right)^{\frac{5}{\alpha-2\beta}} \leq \frac{1}{n^2} \quad (5.28)$$

The last expression is upper bounded by $\frac{1}{n^\beta}$ as required. ■

Proof of Lemma 5.4.3: Consider a set C of vertices of BG_Y such that $|C| = n^\alpha$. Let $C_i = C \cap V_i$ for $1 \leq i \leq 3$, and let $C = C_1 \cup C_2$. We will show that almost surely the number of edges between C_1 and C_2 is at most $\frac{30}{\alpha-2\beta} \frac{n^{2\alpha}}{n^\beta}$. The argument for the number of edges between C_2 and C_3 and between C_1 and C_3 is analogous, and hence the lemma follows. We shall prove the claim for every C_1, C_2 such that $|C_1 \cup C_2| = n^\alpha$. Clearly this implies that it holds for every C_1, C_2 , s.t. $|C_1 \cup C_2| \leq n^\alpha$.

By Claim 5.4.4, with high probability over the choice of Y the following holds. For every good W , $W \subset C$, such that $|W| = s \geq n^\beta \log n$, the number of edges spanned by W in BG_Y is at most $2ks^2 n^{-\beta}$. Assume from now on that the selected Y has this property. We shall use Claim 5.4.5 to derive an upper bound on the number of edges in BG_Y that are spanned by the vertices of C .

By our assumption on Y , if W is good and $|W| = s \geq n^\beta \log n$ then $e_Y(W) \leq 2ks^2 n^{-\beta}$. Clearly, $e_Y(W) \leq s^2$. If we uniformly and independently select $W \subset C$, such that $|W| = s$ then $\text{Exp}[e_Y(W)] \geq \frac{1}{2} \cdot e_Y(C) \cdot \frac{s^2}{n^{2\alpha}}$. We stress that the expectation is taken only over the choice of W and not over the choice of Y . Now,

$$\begin{aligned} \text{Exp}[e_Y(W)] &= \text{Exp}[e_Y(W) \mid W \text{ is good}] \cdot \Pr[W \text{ is good}] + \text{Exp}[e_Y(W) \mid W \text{ is not good}] \cdot \Pr[W \text{ is not good}] \\ &\leq 2ks^2 \cdot n^{-\beta} + s^2 \cdot n^{-\beta} = (2k+1)s^2 \cdot n^{-\beta} \end{aligned} \quad (5.29)$$

It follows that $e_Y(C) \leq (2k+1) \cdot 2|C|^2 \cdot n^{-\beta} \leq 5k|C|^2 \cdot n^{-\beta}$. Since $k = \frac{5}{\alpha-2\beta}$, the lemma follows. ■

As a corollary of Lemma 5.4.3 we get:

Corollary 5.4.1 Let $0 < \beta < \frac{1}{2}$ and $X \subset \{1, \dots, n\}$ where $|X| \geq n^{1-g(n)}$ for $g(n) = \frac{\log \log \log n + 4}{\log \log n}$. Consider the random Behrend graph BG_Y obtained by choosing a random subset $Y \subseteq X$, $|Y| = d = \frac{|X|}{n^\beta}$. With high probability over the choice of Y , for every subset C of vertices in BG_Y such that $|C| \leq \min \left\{ \sqrt{d}, \frac{n^{2/3}}{d^{1/3}} \right\} \cdot n^{-g(n)}$, the following bound applies: $|C| \cdot e_Y(C) \leq n^{1-g(n)}$.

Proof: Note that $d = n^{1-g(n)-\beta}$, and so we need to bound the number of edges in sets C s.t.

$$|C| \leq \min \left\{ n^{\frac{1-g(n)-\beta}{2}}, n^{\frac{1+g(n)+\beta}{3}} \right\} \cdot n^{-g(n)} \leq n^{2/5-g(n)}, \quad (5.30)$$

where $n^{2/5-g(n)}$ is obtained for $\beta = 1/5 - g(n)$.

Consider first the case that $\beta \leq \frac{1}{5} - g(n)$. In this case we need to bound the number of edges in sets C , s.t. $|C| \leq \frac{n^{2/3-g(n)}}{d^{1/3}} = n^{\frac{1-2g(n)+\beta}{3}}$.

Let $\alpha = \frac{1-2g(n)+\beta}{3}$ and observe that $\alpha - 2\beta \geq g(n) > \frac{1}{\log \log n}$. Hence we can apply Lemma 5.4.3 and get that with high probability over the choice of Y , for every subset C such that $|C| = n^\alpha$, we have that $e_Y(C) \leq \frac{90}{\alpha-2\beta} \cdot \frac{n^{2\alpha}}{n^\beta}$. Clearly this upper bound holds also for every subset C such that $|C| \leq n^\alpha$. Hence,

$$|C| \cdot e_Y(C) \leq \frac{90}{\alpha-2\beta} \cdot \frac{n^{3\alpha}}{n^\beta} \leq \frac{90}{g(n)} \cdot n^{1-2g(n)} \leq n^{1-g(n)} \quad (5.31)$$

Consider now the case that $\beta > \frac{1}{5} - g(n)$. Note that we need to bound the number of edges in sets C such that $|C| \leq n^{2/5-g(n)}$. We have shown that the bound applies for the case that $\beta = \frac{1}{5} - g(n)$, and $|C| = n^{2/5-g(n)}$. Hence, for $\beta > \frac{1}{5} - g(n)$, the sets C for which $|C| \leq n^{2/5-g(n)}$ are less dense and therefore the previous bound applies. ■

5.4.3 The lower bound distribution $BG(n, d)$

Let $X \subset [n]$ be a set with no three-term arithmetic progression, as constructed in Subsection 5.4.1, such that $|X| = n^{1-g(n)}$ (where $g(n) = \frac{\log \log \log n + 4}{\log \log n}$). Consider the Behrend graph, denoted BG_X , whose set of generators is X . Recall that BG_X , which is a graph over $3n$ vertices, contains $|X| \cdot n$ edge-disjoint triangles: every edge belongs to exactly one triangle, and every triangle corresponds to some $x \in X$.

For each subset $Y \subset X$, such that $|Y| = d$ we consider the subgraph of BG_X that contains all its vertices but only the edges labeled by differences $y \in Y$. This (sub-)graph contains $n \cdot |Y| = nd$ edge-disjoint triangles and is hence ϵ -far from being triangle free for any $0 < \epsilon < 1/3$. Next we apply a permutation π on the names of the vertices. More precisely, π consists of 3 permutations, π^b , $b \in \{1, 2, 3\}$, each over $\{0, \dots, n-1\}$. If we denote each vertex v in BG_X by a pair (b, i) where $b \in \{1, 2, 3\}$ is the index of the subset that the vertex belongs to and $i \in \{0, \dots, n-1\}$, then $\pi(v) = \pi(b, i) = (b, \pi^b(i))$. We denote the resulting graph by $BG_{Y, \pi}$.

A graph is generated according to the distribution $BG(n, d)$ by uniformly selecting Y and π and outputting the resulting graph $BG_{Y, \pi}$. We also assume that the edges incident to a vertex v are ordered randomly in the incidence list of v . For the sake of simplicity, we do not include these random labelings in the notation.

5.4.4 Online generation of graphs according to $BG(n, d)$

Recall that we would like to show that any algorithm that is given query access to a graph generated according to $BG(n, d)$ must perform $\Omega\left(\min\left\{\sqrt{d}, \frac{n^{2/3}}{d^{1/3}}\right\} \cdot n^{-g(n)}\right)$ queries in order to detect a triangle with sufficiently high constant probability. In order to prove this it will be convenient to define an online process, denoted P , that answers the algorithm's queries while generating a graph according to $BG(n, d)$. The process P will actually provide the algorithm with more information than required by neighbor and vertex-pair queries. Namely, whenever the algorithm asks a query involving a vertex $v \in \{(1, 0), \dots, (3, n-1)\}$ (in either type of query), the process will provide it with $\pi^{-1}(v)$. This will also be the case when the process answers a neighbor query (u, i) with a vertex v . Thus the algorithm is provided with the "identity" in BG_X of the vertices it has observed, and can also derive the labels $y \in Y$ of the edges it has observed.

Clearly, a lower bound on the number of queries of an algorithm that is provided with the additional information described above constitutes a lower bound on an algorithm that is not provided with this additional information. It will actually be convenient to consider the following three types of queries:

1. Vertex queries: for any choice of a vertex $v \in \{(1, 0), \dots, (3, n-1)\}$ the algorithm is provided with $\pi^{-1}(v)$;
2. Random neighbor queries: for any vertex v the algorithm has already observed, it may ask for a new random neighbor u of v (together with $\pi^{-1}(u)$);

3. Difference queries: for any $x \in X$ the algorithm can ask whether $x \in Y$.

Note that a vertex-pair query can be performed by asking at most two vertex queries and one difference query, and a neighbor query can be performed by asking a vertex query and a random neighbor query (recall that the edges adjacent to a vertex are labeled randomly in $BG_{Y,\pi}$). It follows that any lower bound on an algorithm that performs these types of queries implies a lower bound that is at most a factor of 3 smaller on an algorithm that performs vertex-pair and neighbor queries.

Let ALG be an algorithm that performs $Q = Q(n, d)$ queries of the above three types and whose goal is to detect a triangle with probability at least $9/10$. The probability is taken over the choice of the graph $BG_{Y,\pi}$ and the coin flips of the algorithm. Namely ALG is a (possibly probabilistic) mapping from *query-answer histories* $\langle (q_1, t_1), \dots, (q_t, a_t) \rangle$, to q_{t+1} for every $t < Q$. The mapping needs to be defined only on histories that are consistent with some graph $BG_{Y,\pi}$.

As described above, a vertex query $q_t = (VQ, v_t)$ for $v_t \in \{(1, 0), \dots, (3, n-1)\}$ that has not yet been observed is answered by $\pi^{-1}(v_t)$. A random neighbor query $q_t = (NQ, v_t, b')$ for $v_t = (b, i)$ that has been observed is answered by a new random neighbor $u_t = (b', j)$ of v_t together with $\pi^{-1}(u_t)$. A difference query $q_t = (DQ, x)$ for $x \in X$ is answered by ‘1’ (yes) or ‘0’ (no), indicating whether $x \in Y$ or not.

Any query-answer history of length t can be used to define the *knowledge base* $K_t = (V_t, Y_t, \bar{Y}_t, \pi_t)$ at time t , where $V_t \subset \{(1, 0), \dots, (3, n-1)\}$, $Y_t, \bar{Y}_t \subset X$ and $\pi_t : V_t \rightarrow \{(1, 0), \dots, (3, n-1)\}$ (where π_t is one-to-one). Specifically, V_t consists of all vertices (b, j) such that $(b, j) = \pi^{-1}(v)$ for some v that appeared either in one of the first t queries of ALG or in one of the first t answers. The set Y_t consists of all $x \in X$ such that for some $t' \leq t$ there was either a query $q_{t'} = (DQ, x)$ that was answered by ‘1’, or a query $q_{t'} = (NQ, v_t, b')$ that was answered with u_t where the edge between $\pi^{-1}(v_t)$ and $\pi^{-1}(u_t)$ is labeled by x . The set \bar{Y}_t consists of all $x \in X$ such that for some $t' \leq t$ there was a query $q_{t'} = (DQ, x)$ that was answered by ‘0’. Finally, for every $(b, j) \in V_t$, $\pi_t(b, j) = v$ where v is such that $\pi^{-1}(v) = (b, j)$.

Observe that V_t together with Y_t determine a subgraph of BG_X : the vertices of the subgraph are the vertices of V_t , and the edges are all pairs (u, v) , $u, v \in V_t$ such that there is an edge between u and v in BG_X and this edge is labeled by some $x \in Y_t$. For each $b \in \{1, 2, 3\}$ we let $V_{t,b} = \{(b, j) : (b, j) \in V_t\}$.

Definition of the process P

Let $R = R(n, d)$ denote the set of all graphs $BG_{Y,\pi}$ in the support of $BG(n, d)$. For $b \in \{1, 2, 3\}$ and $i, j \in \{0, \dots, n-1\}$ let $R_{b,i,j} \subset R$ denote the subset of graphs $BG_{Y,\pi} \in R$ such that $\pi(b, j) = (b, i)$. For $x \in X$ let $R_x \subset R$ denote the subset of graphs $BG_{Y,\pi} \in R$ such that $x \in Y$, and let $R_{\neg x} \subset R$ denote the subset of graphs $BG_{Y,\pi} \in R$ such that $x \notin Y$.

The process P answers ALG’s queries as follows, where we assume without loss of generality that ALG does not ask any query q_t whose answer can be deduced from the knowledge base K_{t-1} . In particular, for every vertex query $q_t = (VQ, v)$ we have that $v \notin V_{t-1}$, and for every difference query $q_t = (DQ, x)$ we have that $x \notin Y_{t-1} \cup \bar{Y}_{t-1}$. The process P initializes $R_0 = R$, and in general, for any $t \geq 0$, we have that R_t consists of all graphs in R that are consistent with the first t queries.

- To answer a vertex query $q_t = (VQ, v)$, where $v = (b, i)$, the process uniformly selects $(b, j) \in \{(b, 0), \dots, (b, n-1)\} \setminus V_{t-1,b}$ and sets $R_t = R_{b,i,j} \cap R_{t-1}$. Note that (b, j) is selected with probability $\frac{|R_{b,i,j} \cap R_{t-1}|}{|R_{t-1}|} = \frac{|R_t|}{|R_{t-1}|}$.
- Given a difference query (DQ, x) , with probability $\frac{d-|Y_{t-1}|}{|X|-|Y_{t-1}|-|\bar{Y}_{t-1}|}$ the process answers ‘1’ and with probability $1 - \frac{d-|Y_{t-1}|}{|X|-|Y_{t-1}|-|\bar{Y}_{t-1}|} = \frac{|X|-d-|\bar{Y}_{t-1}|}{|X|-|Y_{t-1}|-|\bar{Y}_{t-1}|}$ it answers ‘0’. In the former case it sets $R_t = R_x \cap R_{t-1}$ and in the latter case it sets $R_t = R_{\neg x} \cap R_{t-1}$. Note that here the answer is ‘1’ with probability $\frac{|R_x \cap R_{t-1}|}{|R_{t-1}|} = \frac{|R_t|}{|R_{t-1}|}$ and is ‘0’ with probability $1 - \frac{|R_x \cap R_{t-1}|}{|R_{t-1}|} = \frac{|R_{\neg x} \cap R_{t-1}|}{|R_{t-1}|} = \frac{|R_t|}{|R_{t-1}|}$.

- To answer a random neighbor query $q_t = (NQ, v, b')$, the process performs two steps. First it selects $x \in X$ in the following manner. With probability $\frac{|Y_{t-1}|}{d}$ it selects x uniformly from Y_{t-1} , and with probability $1 - \frac{|Y_{t-1}|}{d}$ it selects x uniformly from $X \setminus (Y_{t-1} \cup \bar{Y}_{t-1})$. In either case the choice of x (together with b') determines the value of $j \in \{0, \dots, n-1\}$ such that there is an edge labeled x in BG_X between $\pi^{-1}(v)$ and (b', j) . If $(b', j) \in V_{t-1}$ then $u = \pi_t(b', j) = \pi(b', j)$ is known and $R_t = R_x \cap R_{t-1}$. Otherwise $u = (b', i)$ is selected uniformly in $\{(b', 0), \dots, (b', n-1)\} \setminus V_{t-1, b'}$ and $R_t = R_x \cap R_{b', i, j} \cap R_{t-1}$.

Once P completes answering the Q queries of ALG it uniformly selects a graph in R_Q . The next lemma is easily derived from the definition of the process.

Lemma 5.4.6 *For every algorithm ALG, the process P , when interacting with ALG, uniformly generates graphs in $BG(n, d)$*

Proof: Consider a specific graph G in $R_0 = BG(n, d)$. The probability that G is generated by P is

$$\begin{aligned} & \Pr[G \in R_1] \cdot \Pr[G \in R_2 | G \in R_1] \cdots \Pr[G \in R_Q | G \in R_{Q-1}] \cdot \frac{1}{|R_Q|} \\ &= \frac{R_1}{R_0} \cdot \frac{R_2}{R_1} \cdots \frac{R_Q}{R_{Q-1}} \cdot \frac{1}{|R_Q|} = \frac{1}{R_0} \end{aligned} \quad (5.32)$$

and the lemma follows. ■

5.4.5 Proof of Lemma 5.4.1

Consider any algorithm ALG that interacts with the process P . We shall show that if ALG asks $Q = o\left(\min\left\{\sqrt{d}, \frac{n^{2/3}}{d^{1/3}}\right\} \cdot n^{-g(n)}\right)$ queries, then the probability that it reveals a triangle is $o(1)$. By Corollary 5.4.1, with high probability over the choice of Y , $|Y| = d$, for every subset U of vertices such that $|U| = o\left(\min\left\{\sqrt{d}, \frac{n^{2/3}}{d^{1/3}}\right\} \cdot n^{-g(n)}\right)$, we have $|U| \cdot e(U) = o(n^{1-g(n)})$, where $e(U)$ denote the number of edges induced on the vertices of U .

In what follows we assume that the graph generated by P in fact has this property, where we take into account the probability of $o(1)$ that this is not the case. Let E_t be the set of edges between vertices in V_t that are known to the algorithm at time t . That is, E_t consists of all edges in BG_X whose labels are in Y_t and are between vertices in V_t . Therefore, for every $t = o\left(\min\left\{\sqrt{d}, \frac{n^{2/3}}{d^{1/3}}\right\} \cdot n^{-g(n)}\right)$ we have that $t|E_t| = o(n^{1-g(n)})$.

Recall that every edge $(i, j) \in E_t$ participates in exactly one triangle. There are two ways by which ALG can close a triangle in its t 'th query. If the query is either a vertex query or a random neighbor query, the algorithm must receive as an answer one of the $|E_{t-1}|$ vertices that close triangles with (the known) edges between vertices in V_{t-1} . If the query is a difference query (DQ, x) (where $x \notin Y_{t-1} \cup \bar{Y}_{t-1}$), then there must be three vertices $(1, i), (2, j), (3, k) \in V_{t-1}$ that form a triangle in BG_X whose edges are labeled by x and the answer to the query is '1' (i.e., $x \in Y$). For sake of simplicity we assume that whenever the algorithm obtains a vertex that closes a triangle in BG_X , then it is also told whether this triangle is in $BG_{Y, \pi}$ or not (i.e., it gets a difference query "for free"). We now turn to bounding the probability of each of the above events by which a triangle is closed.

We first observe that since $|V_t| \leq t = o(n)$, whenever ALG asks a vertex query (VQ, v) where $v = (b, i)$, the answer, (b, j) , is uniformly distributed in a subset of size $\Theta(n)$. Since $|Y_t| + |\bar{Y}_t| \leq t$, whenever ALG asks a random neighbor query $q_t = (NQ, v, b')$, with probability at least $1 - \frac{t}{d}$ we have that $(b', j) = \pi^{-1}(u)$ is uniformly distributed in a subset of size $|X| - t$. Since $|X| = \Omega(n^{1-g(n)})$, and $t = o(\sqrt{d})$, with

probability $1 - o(1)$, for every neighbor query performed, the answer to the neighbor query is uniformly distributed in a subset of size $\Omega(n^{1-g(n)})$. Let us assume from this point on that this is in fact the case (where we take into account that $o(1)$ probability that this is not the case).

Given the above, the probability that ALG closes a triangle in the t 'th query when one of the edges of the triangle is already in E_t is $|E_t|/\Omega(n^{1-g(n)})$. It remains to bound the probability that ALG closes a triangle by obtaining a vertex (b, j) that closes a triangle in the subgraph of BG_X that is induced by V_{t-1} , and that the difference $x \notin Y_{t-1} \cup \bar{Y}_{t-1}$ corresponding to the triangle is determined to be in Y . Since the number of edges in the subgraph of BG_X that is induced by V_{t-1} is upper bounded by $|E_t|$, the probability of the above event is at most $\frac{|E_t|}{\Omega(n^{1-g(n)})} \cdot \frac{d-t}{\Omega(n^{1-g(n)})}$. Since $t = o\left(\min\{\sqrt{d}, n^{2/3}/d^{1/3}\} \cdot n^{-g(n)}\right)$, we know that $t|E_t| = o(n^{1-g(n)})$ and $t^3 \cdot d = o(n^{2(1-g(n))})$. Hence, the probability that one of the above events occurs for any $t \leq Q$ is $o(1)$, as required. ■

5.5 A Reduction from One-Sided Error Lower Bounds to Two-Sided Error Lower Bounds

In this section we establish that under certain conditions, a one-sided error lower bound for triangle-freeness can be transformed into a two-sided error lower bound. Since these conditions hold for our one-sided error lower bounds, we obtain two-sided error lower bounds.

Theorem 5.5.1 *Let D_Δ be a distribution over graphs with n vertices and average degree d , and let $q(n, d)$ be a function of these parameters. Assume the following holds:*

1. *With probability $1 - o(1)$ a graph selected according to D_Δ is ϵ -far from being triangle-free for some constant ϵ .*
2. *One of the following two conditions holds:*
 - *In all graphs in the support of D_Δ , the triangles are edge-disjoint, and for any algorithm A , the probability that A reveals a triangle in a graph selected according to D_Δ using $o(q(n, d))$ queries is less than $2/3$.*
 - *For any algorithm A , the probability that A reveals a cycle (of any length) in a graph selected according to D_Δ using $o(q(n, d))$ queries is less than $2/3$.*

Then any two-sided error algorithm for testing triangle-freeness that has success probability at least $9/10$ must perform $\Omega(q(n/2, d))$ queries.

Since the distributions that are defined for our one-sided error lower bounds, which are stated in Lemmas 5.3.1 and 5.4.1, are as required by Theorem 5.5.1, we get the following corollary.

Corollary 5.5.2 *Any algorithm for testing triangle-freeness must perform*

$$\Omega\left(\max\left\{\min\{d, n/d\}, \min\left\{\sqrt{d}, n^{2/3}/d^{1/3}\right\} \cdot n^{-g(n)}\right\}\right)$$

queries. This lower bound holds even if the algorithm is allowed two-sided error and $d_{\max} = O(d)$.

Here we prove Theorem 5.5.1 in two parts – Theorem 5.5.3 and Theorem 5.5.4 (depending on the conditions on the lower-bound distribution D_Δ). The proofs use similar ideas, but the first one is simpler and hence we present them separately. For the sake of simplicity, and since it suffices for our purposes, in the first theorem we slightly strengthen the first condition on D_Δ .

Theorem 5.5.3 *Let D_Δ be a distribution over graphs with n vertices and average degree d , and let $q(n, d)$ be a function of these parameters.*

- *Every graph in the support of D_Δ is ϵ -far from being triangle-free for some constant ϵ .*
- *For every algorithm A , the probability that A reveals a triangle in a graph generated according to D_Δ using $o(q(n, d))$ queries is less than $2/3$.*
- *In all graphs in the support of D_Δ , the triangles are edge-disjoint.*

Then any two-sided error algorithm for testing triangle freeness that has success probability at least $5/6$ must perform $\Omega(q(n/2, d))$ queries.

Proof: Given the distribution D_Δ we define two distributions over graphs that have $n' = 2n$ vertices and average degree d . One distribution, denoted D'_Δ , generates graphs that are ϵ -far from being triangle free, and the other distribution, denoted $D_{\bar{\Delta}}$ generates graphs that are triangle free. Assume, contrary to what is claimed in the theorem that there exists a two-sided error algorithm A' for testing triangle freeness that performs $o(q(n'/2, d))$ queries and has success probability at least $5/6$. Then, in particular, using $o(q(n'/2, d)) = o(q(n/d))$ queries, A' should be able to distinguish with sufficiently high probability between graphs generated by D'_Δ and graphs generated by $D_{\bar{\Delta}}$. We shall show that we can then use A' to obtain an algorithm A that performs $o(q(n, d))$ queries and with probability at least $2/3$ reveals a triangle in a random graph generated according to D_Δ .

Defining the two distributions. In both distributions, a graph G' over $n' = 2n$ vertices is generated by first selecting a graph G from D_Δ . Every vertex v in G is replaced by two vertices, v_0 and v_1 . Every edge (u, v) in G is replaced by two edges: either the two edges (u_0, v_0) and (u_1, v_1) (so that they are “in parallel”) or the two edges (u_0, v_1) and (u_1, v_0) (so that they are “crossing”). If v is the j 'th neighbor of u and u is the ℓ 'th neighbor of v , then in both cases we maintain the ordering on neighbors. Namely, in the case of parallel edges we have that v_0 is the j 'th neighbor of u_0 and v_1 is the j 'th neighbor of u_1 , u_0 is the ℓ 'th neighbor of v_0 and u_1 is the ℓ 'th neighbor of v_1 (an analogous correspondence holds for crossing edges). The difference between the distributions is in the choice (distribution on the choice) between the above two options.

Recall that the triangles in G are edge-disjoint. Hence, for each triangle in G , the edges between the corresponding vertices in G' can be determined independently from the edges that belong to other triangles. Consider a particular triangle (u, v, w) in G . There are $2^3 = 8$ ways to select the edges between the vertices $u_0, u_1, v_0, v_1, w_0, w_1$ (depending on whether we select parallel or crossing edges). In 4 of these ways we get 2 edge-disjoint triangles (e.g., (u_0, v_0, w_0) and (u_1, v_1, w_1)), and in 4 of these ways we get a single cycle of length 6 (e.g. $(u_0, v_0, w_0, u_1, v_1, w_1)$). The graph generated by D'_Δ simply selects one of the former 4 ways uniformly, and the graph generated by $D_{\bar{\Delta}}$ selects one of the latter 4 ways uniformly.

The basic, but important observation is that for both distributions the following holds: If we consider any edge that belongs to a particular triangle in G , then the probability that the corresponding pair of edges in G' are parallel is equal to the probability that they are crossing. Moreover, this remains true if we condition on any other (single) edge in the triangle being transformed to either parallel or crossing edges. Independence breaks down only when we consider all 3 edges in a triangle. We shall refer to this observation as the *Independence Observation*.

Using a two-sided error algorithm to find triangles. Let A' be a two-sided error algorithm for testing triangle freeness that performs $o(q(n'/2, d))$ queries when testing graphs over $n' = 2n$ vertices and has success probability at least $5/6$. We next show how to use it in order to detect triangles in a graph G over n vertices that is generated randomly according to D_Δ . The idea is that by performing queries to G and flipping some coins, we shall actually be emulating the execution of A' on graphs generated by either D'_Δ or $D_{\bar{\Delta}}$. Since A' is supposed to test graphs over $n' = 2n$ vertices, we denote the vertices in the queries it performs by $\{v_{1,0}, v_{1,1}, \dots, v_{n,0}, v_{n,1}\}$.

Thus let G be a graph generated according to D_{Δ} . Algorithm A (whose goal is to detect a triangle in G) runs A' as a subroutine and answers its queries by performing queries to G and transforming the answers to the queries in an appropriate manner described below. In this process A maintains a *knowledge graph*, denoted \hat{G} , which contains all the edges it has observed in G as well as the “non-edges” (i.e., pairs (u, v) that do not have an edge between them). In addition, A records all the answers it has already given to A' .

Whenever A' performs a degree query for a vertex $v_{i,b}$ ($b \in \{0, 1\}$), algorithm A queries the degree of v_i and returns it as an answer. Whenever A' performs a vertex-pair query $(v_{i,b}, v_{j,b'})$ ($b, b' \in \{0, 1\}$), if (u, v) is an edge or a non-edge in the knowledge graph \hat{G} then the answer to A' is determined. If this is not the case then A performs the vertex-pair query (v_i, v_j) . If the answer is that there is no edge between the two vertices, then the answer given to A' is “no” as well. If the answer is that there is an edge, then there are two cases. If this edge closes a triangle with two other edges in \hat{G} then A terminates successfully. Otherwise, with probability $1/2$ A answers that there is an edge between $v_{i,b}$ and $v_{j,b'}$ and with probability $1/2$ it answers that there is no such edge. In addition, in the former case A' is provided with the information concerning which neighbor is $v_{j,b'}$ of $v_{i,b}$.

Whenever A' performs a neighbor query $(v_{i,b}, \ell)$ (that does not correspond to an edge already in \hat{G}), algorithm A performs the neighbor query (v_i, ℓ) . Let the answer be (v_j, t) . Namely, there is an edge between v_i and v_j , where v_j is the ℓ 'th neighbor of v_i , and v_i is the t 'th neighbor of v_j . Here too, if a triangle in G is detected then A terminates successfully. Otherwise it answers the query of A' in an analogous manner to the way a vertex-pair query is answered. If A' terminates before A has found a triangle, then A terminates unsuccessfully.

Completing the proof. Since A always terminates when or before it finds a triangle, by the Independence Observation, the distribution on the answers it gives to the queries of A' is exactly the same the one we would get if the queries of A' were answered by a graph that is selected either according to D'_{Δ} or according to $D_{\bar{\Delta}}$. We claim that this implies that the probability that A' terminates before A finds a triangle (thus causes A to terminate unsuccessfully) is less than $1/3$. Here the probability is taken over the choice of G , the coin flips of A and the possible coin flips of A' .

Assume, contrary to the claim, that the probability that A' terminates before A finds a triangle is at least $1/3$. Consider the distribution over graphs that results from selecting with probability $1/2$ a graph G' according to D'_{Δ} , and with probability $1/2$ a graph G' according to $D_{\bar{\Delta}}$. By our counter assumption (and the Independence Observation) the probability that A' terminates before it sees three edges of the form (v_{i,b_1}, v_{j,b_2}) , (v_{j,b_3}, v_{k,b_4}) and (v_{k,b_5}, v_{i,b_6}) (where $b_1, \dots, b_6 \in \{0, 1\}$) is greater than $1/3$. In such a case, the distribution on the answers to the queries of A' (and hence on its queries conditioned on these answers) is the same if the graph G' is selected according to D'_{Δ} or according to $D_{\bar{\Delta}}$. Therefore, the probability that A' terminates with an incorrect output, is greater than $1/6$. But this contradicts our assumption on A' .

Since the number of queries performed by A before it terminates is upper bounded by the number of queries performed by A' , the theorem follows. ■

We next turn to the alternative condition in Theorem 5.5.1.

Theorem 5.5.4 *Let D_{Δ} be a distribution graphs with n vertices and average degree d and let $q(n, d)$ be a function over these parameters. Assume the following holds:*

- *With probability $1 - o(1)$ a graph selected according to D_{Δ} is ϵ -far from being triangle-free for some constant ϵ .*
- *For every algorithm A , the probability that A reveals a cycle (of any length) in a graph generated according to D_{Δ} using $o(q(n, d))$ queries is less than $2/3$.*

Then any two-sided error algorithm for testing triangle freeness that has success probability at least $9/10$ must perform $\Omega(q(n/2, d))$ queries.

Before proving the theorem we introduce some notation and prove a few lemmas. Some of the notions introduced are based on ideas used in the proof of Theorem 5.5.3. For any graph G over n vertices, consider the family, denoted \mathcal{H}_G , of graphs over $2n$ vertices that is defined as follows. For every vertex v in G there are two vertices, v_0 and v_1 in each $G' \in \mathcal{H}_G$. For each edge (u, v) in G , every graph $G' \in \mathcal{H}_G$ has two edges: either (u_0, v_0) and (u_1, v_1) (“parallel edges”) or (u_0, v_1) and (u_1, v_0) (“crossing edges”). The family \mathcal{H}_G consists of all possible $2^{|E(G)|}$ graphs that obey the above constraints.

Lemma 5.5.1 *Let G be a graph over n vertices and average degree d that is ϵ -far from being triangle free. With probability $1 - \exp(-\Omega(\epsilon nd))$, a graph selected uniformly at random from \mathcal{H}_G is at least $\epsilon/4$ -far from being triangle free.*

Proof: Since G is ϵ -far from being triangle free, it contains at least ϵnd edge-disjoint triangles. For each such triangle, the probability that it is transformed into two triangles in the random graph in \mathcal{H}_G is $1/2$. By a multiplicative Chernoff bound, the probability that less than $(\epsilon nd)/4$ of the disjoint triangles in G are transformed into a pair of triangles is $\exp(-\Omega(\epsilon nd))$. ■

Lemma 5.5.2 *Let G be a graph over n vertices that has C connected components. There exists a subset $\mathcal{H}_{G, \bar{\Delta}}$ of \mathcal{H}_G that has size 2^{n-C} and such that every graph $G' \in \mathcal{H}_{G, \bar{\Delta}}$ is triangle-free.*

Proof: We assume without loss of generality that G is connected and show that $|\mathcal{H}_{G, \bar{\Delta}}| = 2^{n-1}$. (If G is not connected then the lemma follows by applying the argument to each connected component.) Furthermore, we prove the claim for the case of a clique over n vertices. Since every graph is a subgraph of the clique, the lemma follows.

Consider a fixed ordering of the vertices v_0, v_1, \dots, v_{n-1} . Let $\mathcal{H}_{K_n, \bar{\Delta}, 1}$ consist of two subgraphs: one contains the edges $(v_{1,0}, v_{0,0})$ and $(v_{1,1}, v_{0,1})$, and the other contains the edges $(v_{1,0}, v_{0,1})$ and $(v_{1,1}, v_{0,0})$. In general, $\mathcal{H}_{K_n, \bar{\Delta}, j}$ will consist of subgraphs over $\{v_{0,0}, v_{0,1}, \dots, v_{j,0}, v_{j,1}\}$. We shall prove by induction on j that all subgraphs in $\mathcal{H}_{K_n, \bar{\Delta}, j-1}$ are triangle-free and that $|\mathcal{H}_{K_n, \bar{\Delta}, j}| = 2 \cdot |\mathcal{H}_{K_n, \bar{\Delta}, j-1}|$.

The base case, $j = 1$, is easily established. Assume the claim holds for $j-1 \geq 1$, we prove it for j . Consider a fixed subgraph $H \in \mathcal{H}_{K_n, \bar{\Delta}, j-1}$. Recall that H is over the vertices $\{v_{0,0}, v_{0,1}, \dots, v_{j-1,0}, v_{j-1,1}\}$. We show how it is possible to extend H to two different subgraphs over $\{v_{0,0}, v_{0,1}, \dots, v_{j,0}, v_{j,1}\}$, such that both subgraphs are triangle-free. In one subgraph, denoted H_p (where p stands for “parallel”), first we put an edge between $v_{j,0}$ and $v_{0,0}$, and an edge between $v_{j,1}$ and $v_{0,1}$. In the other subgraph, denoted H_c (where c stands for “crossing”), we put an edge between $v_{j,0}$ and $v_{0,1}$, and an edge between $v_{j,1}$ and $v_{0,0}$. Next, we consider every other pair of vertices $v_{r,0}$ and $v_{r,1}$ in H , $r > 0$, and put the two edges that connect $v_{j,0}$ and $v_{j,1}$ to $v_{r,0}$ and $v_{r,1}$. We do so in the unique way so that there is no triangle between the 6 vertices $v_{j,0}, v_{j,1}, v_{r,0}, v_{r,1}, v_{0,0}$ and $v_{0,1}$. We claim that both resulting graphs are triangle-free.

Assume, contrary to the claim, that there is a triangle in one of the resulting graphs. Since by the induction hypothesis there are no triangles in H , this triangle must contain $v_{j,b}$ for $b \in \{0, 1\}$. Let $v_{r,b'}$ and $v_{s,b''}$ be the two other vertices in this triangle, where $r < s$ (and where $r \neq 0$ by construction). In order to simplify the notation, assume that $b = b' = b'' = 0$. It will be easy to verify that the argument holds for any other setting of b, b', b'' . By definition of H , there is either an edge between $v_{r,0}$ and $v_{0,0}$ or an edge between $v_{r,0}$ and $v_{0,1}$. Assume (once again without loss of generality) that the edge is between $v_{r,0}$ and $v_{0,0}$. Since H is triangle-free, it contains the edge $(v_{s,0}, v_{r,1})$.

But now we reach a contradiction to the existence of the triangle $(v_{j,0}, v_{r,0}, v_{s,0})$: If we are in H_p then $v_{j,0}$ is connected to $v_{r,0}$ and by construction we wouldn't have the edge $(v_{j,0}, v_{r,0})$, and if we are in H_c then $v_{j,0}$ is connected to $v_{r,1}$ and by construction we wouldn't have the edge $(v_{j,0}, v_{s,0})$.

We have thus established that all graphs in $\mathcal{H}_{K_n, \bar{\Delta}} = \mathcal{H}_{K_n, \bar{\Delta}, n-1} \subset \mathcal{H}_{K_n}$ are triangle free, and that $|\mathcal{H}_{K_n, \bar{\Delta}}| = 2^{n-1}$, as required. ■

In what follows, for a fixed choice of G and $G' \in \mathcal{H}_G$, and for any edge (u, v) in G , we let $\sigma_{G'}(u, v) = p$ if G' contains the edges (u_0, v_0) and (u_1, v_1) and we let $\sigma_{G'}(u, v) = c$ if G' contains the edges (u_0, v_1) and (u_1, v_0) .

Lemma 5.5.3 *Let G be a graph and let $T = \{(u_1, v_1), \dots, (u_t, v_t)\}$ be a subset of edges in G that contains no cycles. Consider the distribution over $\sigma_{G'}(u_1, v_1), \dots, \sigma_{G'}(u_t, v_t)$ when G' is selected uniformly in $\mathcal{H}_{G, \bar{\Delta}}$. Then this distribution is uniform over $\{p, c\}^{|T|}$.*

Proof: Recall that in the proof of Lemma 5.5.2, the graphs in $\mathcal{H}_{K_n, \bar{\Delta}}$ (and hence in $\mathcal{H}_{G, \bar{\Delta}}$) were constructed by considering a fixed ordering v_0, \dots, v_{n-1} of the vertices. Starting from the two possible subgraphs over $\{v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}\}$, at each step every subgraph in $\mathcal{H}_{G, \bar{\Delta}, j-1}$ was extended to two subgraphs in $\mathcal{H}_{G, \bar{\Delta}, j}$. This was done by selecting one of the two possible ways to connect $v_{j,0}$ and $v_{j,1}$ to $v_{0,0}$ and $v_{0,1}$, and then adding the remaining edges in unique way that does not create triangles.

Now consider any choice of a “starting” vertex v_{i_0} and any sequence of vertex pairs $(v_{i_1}, u_{i_1}), \dots, (v_{i_{n-1}}, u_{i_{n-1}})$ such that $u_{i_j} \neq v_{i_j}$ and $u_{i_j} \in \{v_{i_0}, v_{i_1}, \dots, v_{i_{j-1}}\}$, for every $1 \leq j \leq n-1$. (In particular we have that $u_{i_1} = v_{i_0}$.) Suppose we apply an analogous process to the one described in Lemma 5.5.2. Namely, at each step j of the process we can either add the edges $(v_{i_j,0}, u_{i_j,0})$ and $(v_{i_j,1}, u_{i_j,1})$ or the edges $(v_{i_j,0}, u_{i_j,1})$ and $(v_{i_j,1}, u_{i_j,0})$, and this determines all other possible edges between $v_{i_j,0}$ and $v_{i_j,1}$ and $v_{i_0,0}, v_{i_0,1}, \dots, v_{i_{j-1},0}, v_{i_{j-1},1}$. Clearly the argument given in Lemma 5.5.2 remains valid, and we obtain 2^{n-1} triangle-free graphs in \mathcal{H}_{K_n} (and hence in \mathcal{H}_G).

We claim that all possible orderings give exactly the same set of triangle-free graphs. Assume contrary to the claim that there exists a triangle-free graph G' which is obtained according to one ordering, but is not obtained according to another ordering. Let the latter ordering be $v_{i_0}, (v_{i_1}, u_{i_1}), \dots, (v_{i_{n-1}}, u_{i_{n-1}})$. Consider the first step j such that in the process of adding the vertices $v_{j,0}$ and $v_{j,1}$ we do not obtain the subgraph of G' that is induced by $v_{i_0,0}, v_{i_0,1}, \dots, v_{i_j,0}, v_{i_j,1}$. Let H be the subgraph of G' that is induced by $v_{i_0,0}, v_{i_0,1}, \dots, v_{i_{j-1},0}, v_{i_{j-1},1}$, and let H_p and H_c be the subgraphs that result from connecting $v_{i_j,0}$ and $v_{i_j,1}$ to $u_{i_j,0}$ and $u_{i_j,1}$ in parallel and crossing, respectively.

Suppose that $\sigma_{G'}(v_{i_j}, u_j) = p$ (the case that $\sigma_{G'}(v_{i_j}, u_j) = c$ is treated similarly). By the counter assumption, there exists some vertex v_{i_r} such that $\sigma_{G'}(v_{i_j}, v_{i_r}) \neq \sigma_{H_p}(v_{i_j}, v_{i_r})$. But by construction of H_p , and since H is a subgraph of both H_p and G' , this implies that G' contains a triangle, and we reach a contradiction.

Having established that all possible orderings result in the same set of triangle-free graphs, observe that the process for constructing the set of triangle-free graphs can be transformed to selecting one of the graphs uniformly: At each step we randomly select how to connect $v_{i_j,0}$ and $v_{i_j,1}$ to $u_{i_j,0}$ and $u_{i_j,1}$, which determines all other edges between $v_{i_j,0}$ and $v_{i_j,1}$ and $v_{i_0,0}, v_{i_0,1}, \dots, v_{i_{j-1},0}, v_{i_{j-1},1}$. Now consider an ordering that corresponds to a BFS on T (where all vertices that are not incident to edges in T are added after the BFS). Then we see that for each edge $(u, v) \in T$, independently from the previous edges in T that are traversed by the BFS, (u, v) is transformed into a parallel or crossing pair of edges in G' with equal probability. ■

Proof of Theorem 5.5.4: Similarly to the proof of Theorem 5.5.3, given the distribution D_{Δ} we define two distributions over graphs that have $n' = 2n$ vertices. One distribution, The distribution D'_{Δ} is defined by selecting a graph G according to D_{Δ} and then uniformly selecting a graph G' in \mathcal{H}_G . The distribution $D_{\bar{\Delta}}$ is defined by selecting a graph G according to D_{Δ} and then uniformly selecting a graph G' in $\mathcal{H}_{G, \bar{\Delta}}$. By the premise of the theorem and Lemma 5.5.1, D'_{Δ} generates graphs that with high probability are $(\epsilon/4)$ -far from being triangle free. By Lemma 5.5.2, $D_{\bar{\Delta}}$ generates graphs that are triangle-free.

Given Lemma 5.5.3, we can complete the proof of Theorem 5.5.4 in a manner that is very similar to the proof of Theorem 5.5.4. ■

5.6 Upper bounds

5.6.1 An upper bound of $O(\sqrt{nd}/\epsilon^{3/2})$ for general graphs

Lemma 5.6.1 *It is possible to test triangle-freeness by performing $O(\sqrt{nd}/\epsilon^{3/2})$ queries. If $d_{\max} = O(d)$ then $O(d/\epsilon)$ queries suffice.*

Proof: Let G be a graph with average degree d over n vertices that is ϵ -far from being triangle-free. By definition, G must contain at least ϵnd edges that belong to triangles. If $d_{\max} = O(d)$ then by uniformly selecting $\Theta(1/\epsilon)$ vertices and for each uniformly selecting an incident edge, with high probability we obtain an edge that belongs to a triangle. Conditioned on this event, if we now perform all $O(d)$ neighbor queries to the end-points of each selected edge, we reveal a triangle.

If the maximum degree of the graph differs significantly from its average degree, then the above argument cannot be applied: First, the suggested edge selection process might not select with sufficiently high probability an edge that belongs to a triangle. Second, even if we obtain such an edge, its end-points might have a very high degree. To address these issues, we first introduce some notation.

We say that a vertex has *high degree* if its degree is more than $c\sqrt{nd}$ (where we set c momentarily). We shall say that an edge is *covered* by these high degree vertices, if *both* its end-points have high degree. But the high-degree vertices can cover at most $((1/c)\sqrt{nd})^2 = (1/c^2)nd$ edges. Hence, among the edges that belong to triangles, there are at least $(\epsilon - (1/c^2))nd$ edges that have at least one end-point with degree at most $c\sqrt{nd}$. If we set $c = \sqrt{2/\epsilon}$ then we have at least $(\epsilon/2)nd$ such edges.

In order to obtain one of these edges, we would like to sample edges uniformly in G . In fact, it suffices to sample edges “almost uniformly” as defined in Chapter 6. In Chapter 6 an algorithm is described that uses $\tilde{O}(\sqrt{n/\delta})$ queries to a graph G and for which the following holds: For all but at most $\delta/4$ -fraction of the edges of G the probability that the edge is selected is at least $\frac{1}{32nd}$. We refer to this algorithm as “Edge-Select”. By definition of the algorithm, if we set $\delta = \epsilon$, we get that there are at least $(\epsilon/4)nd$ edges that can be returned by “Edge-Select” such that these edges belong to triangles and have at least one end-point with degree at most $\sqrt{2/\epsilon}\sqrt{nd}$. It follows that at a cost of $O(\sqrt{n}/\epsilon^{3/2})$ queries we obtain such an edge with a high constant probability. Thus the algorithm for detecting a triangle runs “Edge-Select” $\Theta(1/\epsilon)$ times. For each selected edge, if it has one end-point with degree less than $\sqrt{2/\epsilon} \cdot \sqrt{nd}$ then it asks all neighbor queries for that vertex, and for each of them it asks all pair queries with the other end point. (If both end-points have high degree then the algorithm does nothing). ■

5.6.2 An improved upper bound for relatively dense general graphs

Lemma 5.6.2 *It is possible to test triangle-freeness by performing $O\left(\max\left\{\frac{n^{4/3}}{\epsilon^{2/3}d^{2/3}}, \frac{d_{\max}^2}{\epsilon^2 d^2}\right\}\right)$ queries.*

Corollary 5.6.1 *It is possible to test triangle-freeness of graphs with average degree $d = \Omega(\sqrt{n})$ by performing $O\left(\frac{n^{4/3}}{d^{2/3}\epsilon^2}\right)$ queries.*

Proof: Let G be a graph over n vertices with average degree d and maximum degree d_{\max} that is ϵ -far from being triangle-free. We shall show that if we take a uniform sample of $\Theta\left(\max\left\{\frac{n^{2/3}}{\epsilon^{1/3}d^{1/3}}, \frac{d_{\max}}{\epsilon d}\right\}\right)$ vertices of G , and ask vertex-pair queries between all pairs in the sample, then a triangle is detected with probability at least $2/3$.

Since G is ϵ -far from being triangle-free, it must contain at least ϵnd triples of vertices that form a triangle. This lower bound on the number of triangles implies that the expected number of triangles in a set of s uniformly selected vertices is at least $s^3 \cdot \frac{\epsilon nd}{n^3}$. It follows that for $s \geq n^{2/3}/(\epsilon d)^{1/3}$, the expected number of triangles spanned by the sample is at least 1. This unfortunately does not imply in general that a uniform

sample of $s = \Omega(n^{2/3}/(\epsilon d)^{1/3})$ vertices spans a triangle with probability at least $2/3$. Rather, the size of the sample should depend on the ratio between d_{\max} and d .

Let $s = c \cdot \max\left(\frac{n^{2/3}}{(\epsilon d)^{1/3}}, \frac{d_{\max}}{\epsilon d}\right)$, where $c > 0$ is a sufficiently large constant. Since G is ϵ -far from being a triangle-free, it easily follows that G must contain a family T of $(\epsilon nd)/3$ pairwise edge-disjoint triangles. Fix such a family, and for every $v \in V(G)$, let $d_T(v)$ be the number of triangles in T containing v ; obviously, $d_T(v) \leq d(v)/2 \leq d_{\max}/2$. We sample a set S of s vertices of G uniformly at random. Let X be the random variable counting the number of triangles of T spanned by S . Due to the Chebyshev inequality, it is enough to prove that $\text{Exp}[X]$ is at least a large constant, and the ratio $\text{Var}[X]/\text{Exp}^2[X]$ is at most a small enough constant. We will estimate both quantities.

Observe that each triangle of T falls into S with probability $(1 + o(1))s^3/n^3$. It follows that

$$\text{Exp}[X] = (1 + o(1)) \frac{s^3}{n^3} |T| = \Theta\left(\frac{\epsilon ds^3}{n^2}\right). \quad (5.33)$$

Thus, taking c large enough, we get: $\text{Exp}[X]$ is large enough, too. Also,

$$\text{Var}[X] \leq \sum_{\substack{t, t' \in T \\ t \cap t' \neq \emptyset}} \Pr[t, t' \subset S] = \sum_{v \in V(G)} \binom{d_T(v)}{2} \frac{(1 + o(1))s^5}{n^5} \quad (5.34)$$

(the latter estimate is due to the fact that, since T is pairwise edge-disjoint, for any $t, t' \in T$ with $t \cap t' \neq \emptyset$, the union $t \cup t'$ contains exactly five vertices). Recall that $d_T(v) \leq d_{\max}$. Due to convexity, we get:

$$\text{Var}[X] = O\left(\frac{\epsilon nd}{d_{\max}} \cdot d_{\max}^2\right) \frac{s^5}{n^5}. \quad (5.35)$$

Using the assumption $s = \Omega\left(\frac{d_{\max}}{\epsilon d}\right)$, we derive: $\text{Var}[X]/\text{Exp}^2[X]$ is small enough, as required. ■

Chapter 6

Sampling Edges Almost Uniformly from a Graph

6.1 Introduction

In this chapter we describe and analyze a procedure that for an input graph G , computes a certain approximation to the uniform selection of an edge in $E(G)$. The input graph G is over n vertices with average degree d . Our goal is to use $\tilde{O}(\min(\sqrt{n/\delta}, n/d))$ queries to G , for a given $\delta > 0$. We needed such a procedure for the Bipartite tester, however we find it interesting by itself and there is a subsequent paper [40] that uses ideas in the spirit of those presented here.

6.2 The Sampling Algorithm

We consider two cases: $d > \sqrt{\delta n}$ and $d \leq \sqrt{\delta n}$. The first case is easy since if G contains sufficiently many edges then we simply sample $\Theta(n/d) = \Theta(n^2/m)$ pairs of vertices in order to obtain an edge.

Sample-Edges-Uniformly-in-G

Repeat $\Theta(n/d)$ times:

- Select two vertices in G uniformly and at random.
- Check if there is an edge between these two vertices (by performing a single vertex-pair query). If the answer is positive then output this edge (and exit the repeat loop).

Figure 6.1: A procedure for sampling edges uniformly in G .

The straightforward procedure for this case is given in Figure 6.1. Clearly every edge in G has equal probability of being selected by the procedure, and the query complexity of this procedure is $\Theta(n/d)$, which for $d > \sqrt{\delta n}$ is $\Theta(\min(\sqrt{n/\delta}, n/d))$ as required. (To be more precise, there is some probability that this procedure fails to output an edge. However, the probability that this occurs can be made sufficiently small so as to have a negligible effect on the success probability of our algorithm.)

In the second case, where G contains fewer edges ($d \leq \sqrt{\delta n}$), we do not have an algorithm that selects an edge uniformly from G (using relatively few queries). However, we can show the following:

Theorem 6.2.1 *There exists a procedure Sample-Edges-almost-Uniformly-in-G that uses $\tilde{O}(\sqrt{n/\delta})$ degree and neighbor queries in G and for which the following holds: For all but $(\delta/4)m$ of the edges e in G , the probability that the procedure outputs e is at least $1/(64m)$. Furthermore, there exists a subset $U_0 \subset V(G)$,*

$|U_0| \leq (\delta n/2)$, such that for all edges $e = (u, v)$ that are output with probability less than $1/(64m)$, we have $u, v \in U_0$.

Sample-Edges-Almost-Uniformly-in-G(δ)

1. Let $t = 2\sqrt{n/\delta} \cdot \log m$. Uniformly select a subset of vertices $S \subset V(G)$, where $|S| = t$.
2. Partition the sampled vertices into subsets according to their degree: $S_i = \{v \in S : \deg(v) \in (2^{i-1}, 2^i]\}$.
3. Choose an index i , $1 \leq i \leq \log m$ with probability $\frac{|S_i|2^i}{\sum_i |S_i|2^i}$.
4. Uniformly select a vertex $v \in S_i$.
5. Uniformly select an edge incident to v .

Figure 6.2: A procedure for selecting an edge in G so that all but at most a $(\delta/4)$ -fraction of the edges are selected with probability $\Omega(1/m)$.

The procedure referred to in Theorem 6.2.1 is described in Figure 6.2. Before proving Theorem 6.2.1 we provide some intuition concerning this procedure. We define the following $\log m$ “buckets”: for $1 \leq i \leq \log m$,

$$B_i = \{v \in V(G) : \deg(v) \in (2^{i-1}, 2^i]\} . \quad (6.1)$$

Thus, in each bucket, all vertices have approximately the same degree. Suppose we had a way to pick an index i with probability proportional to $|B_i| \cdot 2^i$, which is approximately the same as picking i with probability proportional to the number of edges incident to vertices in B_i . Further assume that for each i we could uniformly select a vertex in B_i . Then we could select an edge almost uniformly by selected an index i as described above, uniformly selecting a vertex $v \in B_i$, and then uniformly selected an edge incident to v .

The procedure Sample-Edges-Almost-Uniformly-in-G can be viewed as approximating this “ideal” procedure. Assume first that all buckets are relatively large (i.e., $|B_i| = \Omega(\sqrt{n})$ for every $1 \leq i \leq \log m$). Then, by taking a uniformly selected sample of $\Theta(\sqrt{n})$ vertices in G , we can obtain a good estimate of $|B_i|$ (and by that, of $|B_i| \cdot 2^i$), for every i , and we can uniformly select a vertex $v \in B_i$ for any given i . Unfortunately, if some buckets are small, then we might not sample from them at all.

To illustrate the seeming difficulty with such a case, suppose the graph is a star, so that there is one vertex, denoted v^* with degree $n - 1$, and all other vertices have degree 1. In terms of our buckets we have $|B_0| = n - 1$ so that $|B_0| \cdot 2^0 = n - 1$, and $|B_{\log n}| = 1$, so that $|B_{\log n}| \cdot 2^{\log n} = n$. The “ideal” procedure would select each of the two buckets roughly with equal probability, and if it selects the bucket $B_{\log n}$ then it picks v^* . In other words, it picks v^* with probability roughly $1/2$. But if we take a sample of $\Theta(\sqrt{n})$ vertices then the probability v^* falls in the sample is extremely small. However, this turns out to be almost immaterial to the analysis since we are interested in the end result of the distribution on *edges*. In the case of the star graph, every edge incident to v^* is also incident to one of the degree-1 vertices, and hence will be selected with equal probability.

In general, as we shall see in detail in the proof of Theorem 6.2.1, we can lower bound the probability of selecting each edge that has both end-points in sufficiently large buckets. On the other hand, we can upper bound the total number of edges that have both end-points is small buckets. Details follow.

Proof of Theorem 6.2.1: Let the subsets (“buckets”) B_i be as defined in Equation (6.1). Note that in the procedure Sample-Edges-Almost-Uniformly-in-G (Figure 6.2), the subsamples S_i are simply $S_i = S \cap B_i$. Next we define a set of indices I_0 , that includes indices of all buckets B_i that have “few” elements. More

precisely,

$$I_0 = \left\{ i : 1 \leq i \leq \log m \text{ and } |B_i| \leq \frac{\sqrt{\delta n}}{2 \log m} \right\}. \quad (6.2)$$

Let U_0 be the set of vertices that belong to buckets with “few” elements: $U_0 = \bigcup_{i \in I_0} B_i$.

Consider any fixed vertex $v \notin U_0$, and let $i(v)$ be the index of the bucket that v belongs to, that is, $v \in B_{i(v)}$. We denote by C_v the event that v is selected in the Step 4 of the sampling algorithm. Then, for a vector $(s_1, \dots, s_{\log m})$ we can estimate:

$$\Pr[C_v : |S_1| = s_1, \dots, |S_{\log m}| = s_{\log m}] = \frac{s_{i(v)}}{|B_{i(v)}|} \cdot \frac{s_{i(v)} 2^{i(v)}}{\sum_i s_i 2^i} \cdot \frac{1}{s_{i(v)}} \quad (6.3)$$

(first require that v falls in $S_{i(v)}$, then choose the bucket $S_{i(v)}$, and then choose v inside $S_{i(v)}$). Thus the above conditional probability is:

$$\frac{s_{i(v)}}{|B_{i(v)}|} \cdot \frac{2^{i(v)}}{\sum_i s_i 2^i} \geq \frac{s_{i(v)} \deg(v)}{|B_{i(v)}| \sum_i s_i 2^i}. \quad (6.4)$$

The random variable $s_{i(v)}$ is hypergeometrically distributed with parameters n , $|B_{i(v)}|$ and t . It thus has mean $t|B_{i(v)}|/n$, and using known bounds on the tails of the hypergeometric distribution and our assumption on v ($v \notin U_0$ and therefore $B_{i(v)}$ is large), we can get:

$$\Pr \left[\frac{s_{i(v)}}{|B_{i(v)}|} \geq \frac{t}{2n} \right] \geq \frac{3}{4}.$$

Consider the sum $\sum_{i=1}^t s_i 2^i$. We have:

$$\begin{aligned} \text{Exp} \left[\sum_i |S_i| 2^i \right] &= \sum_i \frac{|B_i|}{n} \cdot t \cdot 2^i \\ &= \frac{t}{n} \cdot \sum_i |B_i| 2^i \\ &\leq \frac{t}{n} \cdot \sum_i \sum_{v \in B_i} (2 \deg(v)) \leq \frac{4 \cdot t \cdot m}{n}. \end{aligned}$$

By Markov’s inequality, the probability that $\sum_i |S_i| 2^i > \frac{16tm}{n}$ is less than $1/4$. It thus follows that

$$\Pr \left[\left(\frac{s_{i(v)}}{|B_{i(v)}|} \geq \frac{t}{2n} \right) \& \left(\sum_i s_i 2^i \leq \frac{16tm}{n} \right) \right] \geq \frac{1}{2}. \quad (6.5)$$

Consider only such vectors (s_1, \dots, s_t) . From Equation (6.4) we obtain:

$$\Pr[C_v] \geq \frac{1}{2} \cdot \frac{t}{2n} \cdot \frac{\deg(v)}{\frac{16tm}{n}} = \frac{\deg(v)}{64m}. \quad (6.6)$$

Finally, for an edge $e \in E(G)$, let us denote by C_e the event that e is selected in the fourth step of the procedure Sample-Edges-Almost-Uniformly-in-G. Let $E(U_0)$ denote the set of edges between pairs of vertices in U_0 , that is, $E(U_0) = \{(u, v) \in E(G) : u, v \in U_0\}$. By definition of U_0 , $|U_0| \leq \sqrt{\delta n}/2$, and hence $|E(U_0)| \leq (\delta n)/4$. Therefore, for all but at most $(\delta n)/4 \leq \delta m/4$ of the edges in $E(G)$, at least one

of their end-points is *not* in U_0 . (Note that here we have used the assumption that $m \geq n$.) For each such edge (u, v) where $u \notin U_0$ (or $v \notin U_0$), we have

$$\Pr[C_e] \geq \Pr[C_u] \cdot \frac{1}{\deg(u)} \geq \frac{\deg(u)}{64m} \cdot \frac{1}{\deg(u)} = \frac{1}{64m}. \quad (6.7)$$

■

Chapter 7

Bounds on the Edge Density of Dense Random Cayley Graphs

7.1 Introduction

In this chapter we consider Random Cayley graphs. We show that for dense random Cayley graphs the edge density in relatively large induced subgraphs is close to the edge density of the whole graph. It was previously shown [8] that random Cayley graphs are expanders and hence have the property that the density of every induced subgraph on sufficiently many vertices is very close to the density of the graph. However, the known techniques for proving this property are based on estimating the second eigenvalue of the graph's adjacency matrix, and do not supply any informative bounds for sets of vertices that are much smaller than the number of vertices divided by the square root of the degree. For more details on estimating the edge density of a graph using the spectral technique, the reader should refer to [11]. Our techniques are somewhat reminiscent of those of [7, 43].

7.2 The Bounds

In the following we consider random Cayley graphs. Using methods that were introduced in Subsection 5.4.2, we get bounds on the edge density of random Cayley graphs with large sets of generators. We start with a definition of Cayley graphs. For simplicity we consider only Cayley graphs over Abelian groups, but all arguments apply to the non-Abelian case as well.

Let H be a finite Abelian group. A set $X \subseteq H$ is *symmetric* if $X = -X$, where $-X = \{-x : x \in X\}$. The *Cayley graph* over H with respect to a symmetric set X , denoted CG_X , has H as its vertex set and distinct vertices $a, b \in H$ are connected by an edge if and only if $a - b$ (hence also $b - a$) is in X . We shall say in such a case that the edge *corresponds* to $x = a - b$. All operations involving vertices are performed in H . A *difference* of a set $T \subseteq H$ is an element $a - b$ such that $a, b \in T$. We let $\Delta(T)$ denote the set $\{a - b : a, b \in T\}$ of differences of T . By definition $\Delta(T)$ contains at most $|T|(|T| - 1)$ nonzero elements. The *multiplicity* in T of a difference $x \in \Delta(T)$ is $|\{(a, b) : a, b \in T \text{ and } a - b = x\}|$. Clearly, the differences that correspond to edges that are incident to a specific vertex are all distinct. It follows that the multiplicity in T of any specific difference in $\Delta(T)$ is at most $|T|$. Moreover, if we consider the complete graph over H , then the edges that correspond to a specific difference form a set of cycles that cover H .

In what follows, we show that for dense random Cayley graphs the edge density in relatively large induced subgraphs is close to the edge density of the whole graph. It was previously shown [8] that random Cayley graphs are expanders and hence have the property that the density of every induced subgraph on sufficiently many vertices is very close to the density of the graph. However, the known techniques for

proving this property are based on estimating the second eigenvalue of the graph's adjacency matrix, and do not supply any informative bounds for sets of vertices that are much smaller than the number of vertices divided by the square root of the degree.

We shall use the following notation: For $X, C \subseteq H$, we let $e_X(C)$ denote the number of edges in the subgraph of CG_X that is induced by C .

We now describe the result of this section. For simplicity we consider only Cayley graphs over Abelian groups, but all arguments apply to the non-Abelian case as well.

Theorem 7.2.1 *Let $0 < \beta < \frac{1}{2}$ and $0 < \alpha \leq 1$ satisfy $\alpha - 2\beta > \frac{1}{\log \log n}$, and let H be an Abelian group where $|H| = n$. Let $X \subset H$, $X = -X$, be determined as follows: Every pair $x, -x$ is chosen independently with probability $p(n) = \frac{1}{n^\beta}$ to be in X . With high probability over the choice of X , for every subset C of n^α vertices in CG_X , we have that $e_X(C) \leq \frac{90}{\alpha - 2\beta} \frac{n^{2\alpha}}{n^\beta}$.*

Similarly to Lemma 5.4.3, Theorem 7.2.1 shows that for sufficiently large subsets C (i.e., for $|C| = n^\alpha$, where $\alpha - 2\beta$ is a constant), the number of edges spanned by C in CG_X is close to its expected value. Here too, the smaller we choose β (i.e., the larger we choose X), the smaller can α be. That is, the theorem can be applied to sets with smaller size. The proof of Theorem 7.2.1 is similar to the proof of Lemma 5.4.3. The difference lies in the proof of Claim 7.2.2, which is analogous to Claim 5.4.4. Here too we let $k = \frac{5}{\alpha - 2\beta}$ and note that for β and α as required in the theorem, $k \leq 5 \log \log n$. We shall say that W is *good* if no difference in $\Delta(W)$ has multiplicity higher than k in W . Theorem 7.2.1 follows from the next two claims using the same arguments that were applied in showing that Lemma 5.4.3 follows from Claims 5.4.4 and 5.4.5.

Claim 7.2.1 *With high probability over the choice of X , for every good W such that $|W| = s \geq n^\beta \log n$, we have that $e_X(W) \leq \frac{2ks^2}{n^\beta}$.*

Proof: Consider a fixed choice of a good W such that $|W| = s = n^\beta \log n$. By definition, $|\Delta(W)| \leq |W|^2 = s^2$. Since W is good, for every $x \in \Delta(W)$, if $x \in X$, then the number of edges labeled by x in the subgraph of CG_X induced by W is at most k . Since each $x \in \Delta(W)$ is chosen to be in X independently with probability $n^{-\beta}$, the expected size of $\Delta(W) \cap X$ is $|\Delta(W)| \cdot n^{-\beta} \leq s^2 \cdot n^{-\beta}$. By a multiplicative Chernoff bound, the probability that $|\Delta(W) \cap X| > 2 \cdot s^2 \cdot n^{-\beta}$ is upper bounded by $\exp(-s^2 \cdot n^{-\beta})$. The claim follows by taking a union bound over all choices of W of size s . ■

Claim 7.2.2 *Let $C \subset H$ satisfy $|C| = n^\alpha$. Suppose we uniformly and independently select $W \subset C$, $|W| = n^\beta \log n$. Then the probability that W is not good is at most $\frac{1}{n^\beta}$.*

Proof: Consider any fixed difference $x \in \Delta(C)$, where there are at most $|C|^2$ such differences. Recall that there are at most $|C|$ edges in the subgraph of CG_H induced by C that correspond to x . We denote this set of edges by $E_x(C)$. Since $E_x(H)$ is a union of disjoint cycles, $E_x(C)$ is a union of disjoint cycles and paths. Therefore, every choice of $k + 1$ edges in $E_x(C)$ is a union of r_1 (sub)paths and r_2 cycles where $1 \leq r_1 \leq k + 1$ and $r_2 \leq (k + 1 - r_1)/3$ (the second inequality follows from the fact that the length of a cycle is at least 3). Note that when $r_1 = k + 1$ then the $k + 1$ edges constitute a matching, as was the case in the proof of Claim 5.4.5. In this case the number of vertices incident to them is $2(k + 1)$.

More generally, the number of incident vertices is $k + 1 + r_1$. For each pair (r_1, r_2) , the number of choices of $k + 1$ edges that constitute r_1 paths and r_2 cycles is at most $|C|^{r_1} (k + 1)^{r_1} \cdot |C|^{r_2}$. Namely, to determine each of the r_1 paths, we select a starting vertex (out of $|C|$ vertices) and a length (between 1 and $k + 1$). To determine each of the r_2 cycles, we select a vertex (that belongs to the cycle). Once the edges are selected, the number of choices of the remaining $|W| - (k + 1 + r_1)$ vertices in W is $\binom{|C| - (k + 1 + r_1)}{|W| - (k + 1 + r_1)}$.

Therefore, for each fixed choice of r_1 and r_2 , the probability that W spans $k + 1$ edges in $E_x(C)$ that constitute r_1 paths and r_2 cycles is at most

$$\frac{|C|^{r_1} (k+1)^{r_1} |C|^{r_2} \binom{|C|-(k+1+r_1)}{|W|-(k+1+r_1)}}{\binom{|C|}{|W|}} \leq |C|^{r_1} (k+1)^{r_1} |C|^{r_2} \cdot \left(\frac{|W|}{|C|} \right)^{k+1+r_1} \quad (7.1)$$

$$= (k+1)^{r_1} \cdot \frac{|W|^{k+1+r_1}}{|C|^{k+1+r_2}} \quad (7.2)$$

$$< (k+1)^{r_1} \frac{|W|^{k+1+r_1}}{|C|^{(2/3)(k+1)+(1/3)r_1}} \quad (7.3)$$

The expression in Equation (7.3) is maximized when the ratio between $|W|$ and $|C|$ is maximized (i.e., β is maximized with respect to α) and r_1 is maximized (i.e., $r_1 = k + 1$). In this case we get an upper bound of $(k+1)^{k+1} (\log n)^{2(k+1)} n^{(2\beta-\alpha)(k+1)}$. Substituting $k = \frac{5}{\alpha-2\beta}$, where $k \leq 5 \log \log n$, summing over all r_1, r_2 and taking a union bound over the at most $|C|^2 = n^{2\alpha} \leq n^2$ choices of $x \in \Delta(C)$, the claim follows. ■

It is worth noting that the technique here can be applied with other parameters as well. In particular, it can be shown, for example, that with high probability, in random Cayley graphs over groups of size n in which the number of generators is $(1 + o(1)) \frac{n}{2}$, every set X of at least some poly $\log(n)$ vertices spans $(1 + o(1)) \frac{1}{2} \binom{|X|}{2}$ edges. We omit the details.

Part III

Testing of Codes

Chapter 8

Testing Polynomials over General Fields

8.1 Introduction

In this chapter we consider the problem of testing, for a given finite field F and degree-bound d , whether a function $f : F^n \rightarrow F$ is a multivariate polynomial of total degree at most d over F . Specifically, the testing algorithm is given query access to f and a parameter $\epsilon > 0$. If f is a polynomial of degree at most d then the testing algorithm must accept. On the other hand, if f differs from every such polynomial on more than an ϵ -fraction of the domain elements, then the test should reject with probability at least $2/3$.

The problem of testing multivariate low-degree polynomials over finite fields has been studied extensively, mainly due to its applications to Probabilistically Checkable Proofs systems (PCPs). This is true both for the special case of linear functions (degree-1 polynomials) [24, 15, 33, 18, 19, 17, 68] and for the more general case of degree- d polynomials [15, 14, 35, 33, 66, 34, 12]. However, all these results apply only to testing polynomials over fields that are *larger than the degree-bound*, d . In particular, when the field size $|F|$ is at least $c \cdot d$, for some sufficiently large constant c , then a number of queries that is linear in d is sufficient [65, 34], and when $d + 2 \leq |F| < c \cdot d$ then the dependence on d is known to be polynomial [34, 66]. In [4] we studied the same property for the case $|F| = 2$ and for $d \geq 2$. Namely, we considered the case in which the degree-bound may be (much) larger than the field size, but this result holds only for $F = \text{GF}(2)$. The number of queries both necessary and sufficient in this case is *exponential* in d . Hence we encounter a very large gap in terms of the dependence on d between the query complexity when $|F| > d$ and the query complexity when $|F| = 2$.

Our Main Result. In the following we bridge the gap between the two cases mentioned above and show a smooth transition between them. In particular, we prove the following theorem.

Theorem 8.1.1 *There exists a testing algorithm for polynomials of degree at most d over finite fields of cardinality q where $2 \leq q = O(d)$. The algorithm performs $O(\ell \cdot q^{2\ell+1} + 1/\epsilon)$ queries, where for prime q , $\ell = \left\lceil \frac{d+1}{q-1} \right\rceil$, and more generally, when q is a power of a prime p then $\ell = \left\lceil \frac{d+1}{q-q/p} \right\rceil$.*

Observe that as the field size q increases, the dependence on d decreases from being exponential, to being polynomial. We note that this query complexity (when $q = O(d)$) is almost tight: for prime fields (and constant ϵ) $\Omega(q^{\ell-1})$ queries are necessary, and for non-prime fields $\Omega(q^{\lceil \ell/2 \rceil - 1})$ queries are necessary. As we discuss in more detail subsequently, the “gap phenomenon” that we observe, is not unique to testing polynomials: analogous gaps arise in other property testing problems. We do not provide an explicit proof for the case of $F = \text{GF}(2)$ that we obtained in [4], since this case is covered by the result for general fields that we obtained in [50].

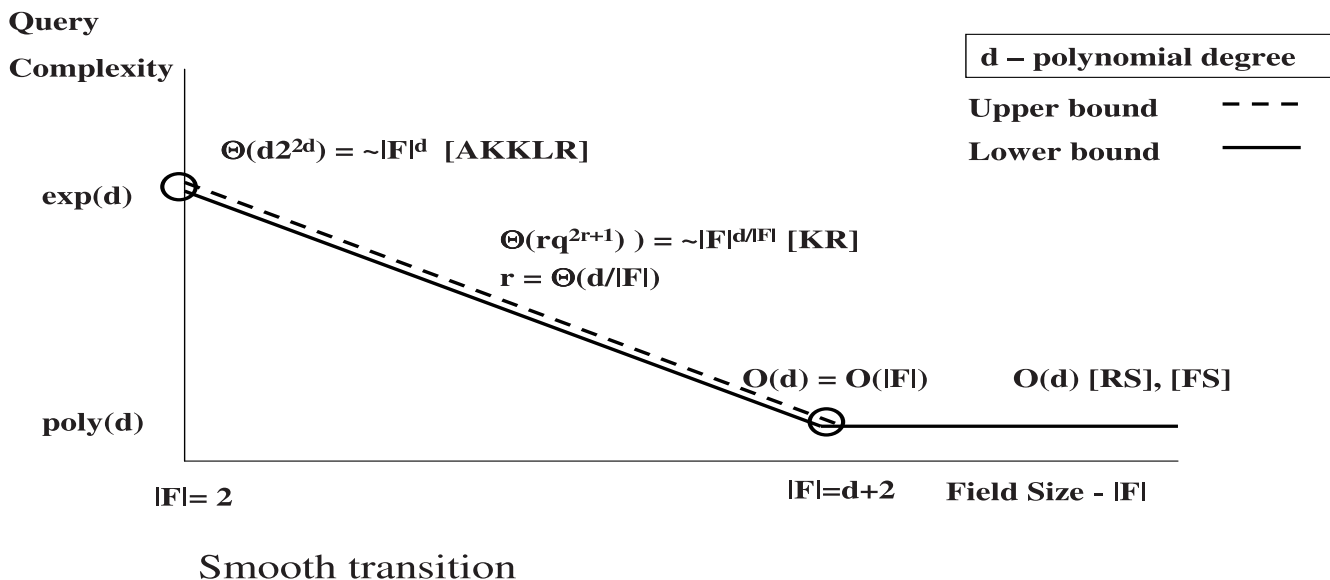


Figure 8.1: Testing low degree polynomials - illustration of results obtained in this chapter.

Characterization of Degree- d Polynomials over $\text{GF}(q)$. One of the building blocks of our analysis is a characterization of (total) degree- d multivariate polynomials over finite fields. In particular, we prove the following theorem.

Theorem 8.1.2 *Let $F = \text{GF}(q)$ where $q = p^s$ and p is prime. Let d be an integer, and let $f : F^n \rightarrow F$. Then f is a polynomial of degree at most d , if and only if its restriction to every affine subspace of dimension $\ell = \left\lceil \frac{d+1}{q-q/p} \right\rceil$ is a polynomial of degree at most d .*

Theorem 8.1.2 generalizes the characterization result of Friedl and Sudan [34] that refers to the case $q - q/p \geq d + 1$ (that is, $\ell = 1$). We also note that this value, ℓ , of the dimension of the considered subspaces, is tight. Namely, there exist polynomials of degree greater than d whose restrictions to affine subspaces of dimension less than ℓ are all degree- d polynomials.

A Unifying Approach to Testing Low-Degree Polynomials. The testing algorithm presented in this work utilizes the characterization in Theorem 8.1.2 (which is shown to be *robust* in the sense defined in [66]). Specifically, the algorithm selects random affine subspaces (of dimension ℓ as defined in the theorem), and checks that the restriction of the input function f to each of the selected subspaces is indeed a polynomial of degree at most d . Such a check is implemented by verifying that various linear combinations of the values of f on the subspace sum to 0. Observe that when the size of the field F is sufficiently larger than the degree bound d then $\ell = 1$. That is, when the field is sufficiently large, then the algorithm checks whether the univariate polynomials that correspond to restrictions of the function f to random *lines* in F^n all have degree at most d . But this is essentially the original low-degree test of Rubinfeld and Sudan [66].¹

¹The reason we say “essentially” is that when $|F| > d$ then it is not necessary to query f on all points on a selected line, but rather it suffices to interpolate using $d + 1$ points and check that the resulting degree- d polynomial agrees with a random point on

On the other hand, when $q = 2$ then the test in our work presented in [4], works by uniformly selecting $\frac{d+1}{q-1} = d + 1$ points in $\text{GF}(2)^n$ and verifying that the sum of the values of f taken over all sums of subsets of these points is 0. This too can be shown to amount to checking whether the restriction of f to the $(d + 1)$ -dimensional subspace spanned by the selected points is a polynomial of degree at most d . Thus our test suggests a uniform view of all these tests for low-degree polynomials.

Relation to Coding. The *Generalized Reed-Muller (GRM)* code of rank d over $\text{GF}(q)$, which we denote by $\mathcal{GRM}_q(d, n)$, consists of all words of length q^n that correspond to the evaluations of degree- d polynomials over $\text{GF}(q)^n$. (When $q = 2$ then the code is simply referred to as *Reed-Muller (RM)*.) Hence, an equivalent view of our main result, from a coding-theory perspective, is that GRM codes are locally testable. See Chapter 9 for detailed results for GRM codes.

The paper of Jutla et al. [45]. Independently from our work, Jutla, Patthak, Rudra, and Zuckerman [45] studied the problem of testing low-degree polynomials and described a testing algorithm that has the same query complexity as our algorithm. However, their algorithm works only for prime fields.

8.2 The Characterization

In this section we prove Theorem 8.1.2, which was stated in the introduction, and that provides a characterization of polynomials of total degree at most d over finite fields.

We also show that Theorem 8.1.2 is tight. Namely,

Theorem 8.2.1 *For any given d and $q = p^s$, let ℓ be as defined in Theorem 8.1.2. Then there exists a function $f : F^n \rightarrow F$ such that for every affine subspace S of F^n having dimension less than ℓ , the function f restricted to S is a degree- d polynomial, but f is not a degree- d polynomial.*

Proof: Let $f = x_1^{(p-1)p^{s-1}} \cdot x_2^{(p-1)p^{s-1}} \cdots x_\ell^{(p-1)p^{s-1}}$, so that the degree of f is $\ell \cdot (p - 1)p^{s-1}$, which is at least $d + 1$. On the other hand, consider any choice of ℓ points $y_0, y_1, \dots, y_{\ell-1}$ (where $y_1, \dots, y_{\ell-1}$ are linearly independent). Then

$$\begin{aligned} f|_{(y_0, y_1, \dots, y_{\ell-1})}(z_1, \dots, z_{\ell-1}) &= \prod_{j=1}^{\ell} \left(y_{0,j} + \sum_{i=1}^{\ell-1} z_i \cdot y_{i,j} \right)^{(p-1)p^{s-1}} \\ &= \prod_{j=1}^{\ell} \left(y_{0,j}^{p^{s-1}} + \sum_{i=1}^{\ell-1} \left(y_{i,j}^{p^{s-1}} \cdot z_i^{p^{s-1}} \right) \right)^{p-1} \end{aligned} \quad (8.1)$$

(where we have used the fact that $(a + b)^p = a^p + b^p$ in a field F with characteristic p). If we group together all terms that correspond to the same monomial in the z_i 's, then we get a sum of terms of the following form:

$$C \cdot \prod_{i=1}^{\ell-1} z_i^{p^{s-1} \cdot A_i} \quad (8.2)$$

where $\sum_{i=1}^{\ell-1} A_i \leq \ell(p - 1)$ and C is a coefficient that is a function of the $y_{i,j}$'s. Recall that $z_i^q = z_i$ for every $z_i \in F$. We claim that this implies that the total degree of each of the monomials in Equation (8.2) is at most $(\ell - 1) \cdot (p - 1)p^{s-1}$.

the line.

To verify this, consider any choice of $A = A_1, \dots, A_{\ell-1}$ such that $\sum_{i=1}^{\ell-1} A_i \leq \ell(p-1)$, and let D_A denote the degree of the corresponding monomial. That is

$$D_A = \sum_{i=1}^{\ell-1} ((p^{s-1} \cdot A_i) \bmod (q-1)) \quad (8.3)$$

Note that each summand in the above sum is computed modulo $q-1$, but the sum is then taken over the integers. Let us fix any choice of $A = A_1, \dots, A_{\ell-1}$ and show that $D_A \leq (\ell-1)(p-1)p^{s-1}$. Assume, without loss of generality that $A_1 = \max_{1 \leq i \leq \ell-1} \{A_i\}$. If $A_1 \leq p-1$ then clearly $D_A \leq (\ell-1)(p-1)p^{s-1}$. Otherwise, $A_1 = (p-1) + a$ where $0 < a \leq (\ell-1)(p-1)$ and $\sum_{i=2}^{\ell-1} A_i \leq (\ell-1)(p-1) - a$. Now

$$D_A \leq (p^{s-1} \cdot A_1) \bmod (q-1) + \sum_{i=2}^{\ell-1} p^{s-1} \cdot A_i \quad (8.4)$$

$$\leq (p^{s-1} \cdot (p-1+a)) \bmod (q-1) + p^{s-1} \cdot ((\ell-1)(p-1) - a) \quad (8.5)$$

$$= (\ell-1)(p-1)p^{s-1} - ap^{s-1} + ((p-1+a)p^{s-1}) \bmod (q-1) \quad (8.6)$$

$$< (\ell-1)(p-1)(p-1)p^{s-1} \quad (8.7)$$

We have thus established that the monomial with the highest degree has degree at most $(\ell-1) \cdot (p-1)p^{s-1}$. But now,

$$(\ell-1) \cdot (p-1)p^{s-1} = \left(\left\lceil \frac{d+1}{(p-1)p^{s-1}} \right\rceil - 1 \right) \cdot (p-1)p^{s-1} < d+1 \quad (8.8)$$

and since $(\ell-1) \cdot (p-1)p^{s-1}$ is an integer, we are done. ■

Proof of Theorem 8.1.2

As noted in the introduction, our proof of Theorem 8.1.2 generalizes the proof of the special case of $\ell = 1$, which is presented in [34]. We prove Theorem 8.1.2 by induction. Namely, we prove that for every $m \geq \ell$ and every affine subspace S of F^n having dimension m , if for every affine subspace S' of S that has dimension ℓ , we have $f_{|S'} \in \text{POLY}_{\ell,d}$, then $f_{|S} \in \text{POLY}_{m,d}$. Theorem 8.1.2 follows by setting $m = n$. The base case, $m = \ell$ clearly holds, and so we turn to the induction step.

Assume the induction claim holds for every $m \geq \ell$, we prove it for $m+1$. Namely, we take any $(m+1)$ -dimensional subspace T of F^n and consider the function $h = f_{|T} : F^{m+1} \rightarrow F$. We then use the induction hypothesis by which for every affine subspaces S of F^{m+1} having dimension m (which is isomorphic to an affine subspace of T having dimension m), the restriction of $h = f_{|T}$ to S is a degree- d polynomial.

Let the coefficients of the polynomial representation of h be denoted by $\{C_{\alpha}^h\}_{\alpha \in [q-1]^{m+1}}$. Our goal is to show that for each $\alpha \in [q-1]^{m+1}$ such that $\sum_{i=1}^{m+1} \alpha_i > d$, we have that $C_{\alpha}^h = 0$. Let us fix any such α , denote it by α^* and prove that $C_{\alpha^*}^h = 0$. We break the proof into three cases.

Case 1: *There exists a subset $R \subset \{1, \dots, m+1\}$, where $|R| = m$, such that $\sum_{i \in R} \alpha_i^* > d$.*

Assume, without loss of generality (since the variables of h , and the corresponding α_i^* 's can be re-ordered), that $R = \{1, \dots, m\}$, and assume, contrary to the claim, that $C_{\alpha^*}^h \neq 0$. We will show that this implies that there exist $y_0, y_1, \dots, y_m \in F^{m+1}$ and $\gamma = \gamma_1, \dots, \gamma_m$ where $\gamma_i \in [q-1]$, and $\sum_{i=1}^m \gamma_i > d$, such that for the affine subspace of $S = S(y_0, y_2, \dots, y_m)$ of dimension m we have $C_{\gamma}^{h|S} \neq 0$, contradicting the induction hypothesis.

Specifically, let $y_i = e_i$ for $i = 1, \dots, m$, where e_i is the i 'th unit vector, and for each $b \in F$, let $y_0(b) = b \cdot e_{m+1}$. Let $g_b = h_{|(y_0(b), y_1, \dots, y_m)}$ (so that $g_b : F^m \rightarrow F$). Then for each choice of $b \in F$ we have

$$\begin{aligned} g_b(z_1, \dots, z_m) &= h(z_1, \dots, z_m, b) \\ &= \sum_{\alpha \in [q-1]^{m+1}} C_\alpha^h \cdot \prod_{i=1}^m z_i^{\alpha_i} \cdot b^{\alpha_{m+1}} \end{aligned} \quad (8.9)$$

Consider the coefficient of the term $z_1^{\alpha_1^*} \cdot z_2^{\alpha_2^*} \cdots z_m^{\alpha_m^*}$ in g_b , that is, $C_{\alpha_1^*, \dots, \alpha_m^*}^{g_b}$ (where recall that α^* satisfies $\sum_{i=1}^{m+1} \alpha_i^* > d$, as well as the premise of this case). This coefficient has the following form:

$$C_{\alpha_1^*, \dots, \alpha_m^*}^{g_b} = \sum_{j=0}^{q-1} C_{\alpha_1^*, \dots, \alpha_m^*, j}^h \cdot b^j \quad (8.10)$$

Namely, it is the evaluation, at b , of the univariate polynomial: $\sum_{j=0}^{q-1} C_{\alpha_1^*, \dots, \alpha_m^*, j}^h \cdot x^j$. Note that for $j = \alpha_{m+1}^*$, the coefficient of x^j in this polynomial is $C_{\alpha^*}^h$, which is non-zero by our counter-assumption. Hence, this polynomial is a non-zero polynomial of degree at most $q - 1$ over F . This implies that for at least one value of b , this polynomial attains a non-zero value. But this means that for some choice of b , $C_{\alpha_1^*, \dots, \alpha_m^*}^{g_b} \neq 0$. Since $\sum_{i=1}^m \alpha_i^* > d$, we have reached a contradiction, and hence completed the proof for this case.

Case 2: *There exist a pair of indices $i, j \in \{1, \dots, m+1\}$ such that $\alpha_i^*, \alpha_j^* \neq 0$ and $\alpha_i^* + \alpha_j^* < q$.*

Assume, without loss of generality, that $i = m$ and $j = m+1$. Here too we assume, contrary to the claim, that $C_{\alpha^*}^h \neq 0$, and reach a contradiction to the induction hypothesis.

Let y_0 be the all-0 vector, let $y_i = e_i$ for $i = 1, \dots, m-1$, and for each $b \in F$, let $y_m(b) = \langle 0, \dots, 0, 1, b \rangle$ (recall that $y_0, \dots, y_m \in F^{m+1}$). Here too we denote $g_b = h_{|(y_0, \dots, y_m(b))}$. Then for each choice of $b \in F$ we have

$$\begin{aligned} g_b(z_1, \dots, z_m) &= h(z_1, \dots, z_m, b \cdot z_m) \\ &= \sum_{\alpha \in [q-1]^{m+1}} C_\alpha^h \cdot \prod_{i=1}^{m-1} z_i^{\alpha_i} \cdot z_m^{\alpha_m} \cdot (z_m \cdot b)^{\alpha_{m+1}} \\ &= \sum_{\alpha \in [q-1]^{m+1}} C_\alpha^h \cdot \prod_{i=1}^{m-1} z_i^{\alpha_i} \cdot z_m^{\alpha_m + \alpha_{m+1}} \cdot b^{\alpha_{m+1}} \end{aligned} \quad (8.11)$$

Consider the coefficient of the term $z_1^{\alpha_1^*} \cdots z_{m-1}^{\alpha_{m-1}^*} \cdot z_m^{\alpha_m^* + \alpha_{m+1}^*}$ in g_b (recall that $\alpha_m^* + \alpha_{m+1}^* < q$). This coefficient has the following form:

$$C_{\alpha_1^*, \dots, \alpha_{m-1}^*, \alpha_m^* + \alpha_{m+1}^*}^{g_b} = \sum_{\substack{j, k \in [q-1]^2 \\ j+k = \alpha_m^* + \alpha_{m+1}^*}} C_{\alpha_1^*, \dots, \alpha_{m-1}^*, j, k}^h \cdot b^k \quad (8.12)$$

That is, it is the evaluation, at b , of the univariate polynomial:

$$\sum_{k=0}^{q-1} C_{\alpha_1^*, \dots, \alpha_{m-1}^*, \alpha_m^* + \alpha_{m+1}^* - k, k}^h \cdot x^k \quad (8.13)$$

Note that for $k = \alpha_{m+1}^*$, the coefficient of x^k in this polynomial is $C_{\alpha^*}^h$, which is non-zero by our counter-assumption. Hence, this polynomial is a non-zero polynomial of degree at most $q - 1$ over F , which

implies that for at least one value of b it attains a non-zero value. But this means that for some choice of b , $C_{\alpha_1^*, \dots, \alpha_{m-1}^*, \alpha_m^* + \alpha_{m+1}^*}^{g_b} \neq 0$ and the proof of this case follows.

We observe that if $\ell = \left\lceil \frac{2(d+1)}{q} \right\rceil$ then either Case 1 or Case 2 must hold. This implies that Theorem 8.1.2 is established for q that is a power of 2 (since in this case $q - q/p = q/2$). It also follows that for any value of q , a variant of Theorem 8.1.2, which takes ℓ to be at most a factor of 2 larger than that stated in the theorem, is established as well. In order to get the tighter result, which holds for $\ell = \left\lceil \frac{d+1}{q-(q/p)} \right\rceil$ and any q , we need to analyze the third and final case.

Case 3 (neither Case 1 nor Case 2 hold): For every subset $R \subset \{1, \dots, m+1\}$, $|R| = m$, we have that $\sum_{i \in R} \alpha_i^* \leq d$, and for every pair of indices $i, j \in \{1, \dots, m+1\}$ we have that $\alpha_i^* + \alpha_j^* \geq q$.

Our proof of this case is similar in its general structure to the proofs of Cases 1 and 2, but is somewhat more involved since we take into account a larger set of m -dimensional affine subspaces. In what follows, when we write $t \cdot a$, where $a \in F$ and t is an integer, we mean the sum $\underbrace{a + a + \dots + a}_t$ in the field F . For every choice of a_1, \dots, a_m, b each in F , let $y_0 = b \cdot e_{m+1}$, and for $i = 1, \dots, m$, let $y_i = a_i \cdot e_i + e_{m+1}$. Consider the function $g_{a_1, \dots, a_m, b} = h_{|(y_0(b), y_1(a_1), \dots, y_m(a_m))} : F^m \rightarrow F$. By definition:

$$\begin{aligned} & g_{a_1, \dots, a_m, b}(z_1, \dots, z_m) \\ &= h\left(a_1 \cdot z_1, \dots, a_m \cdot z_m, \sum_{i=1}^m z_i + b\right) \\ &= \sum_{\alpha \in [q-1]^{m+1}} C_\alpha^h \cdot \prod_{i=1}^m (a_i \cdot z_i)^{\alpha_i} \cdot \left(\sum_{i=1}^m z_i + b\right)^{\alpha_{m+1}} \\ &= \sum_{\alpha \in [q-1]^{m+1}} \sum_{\substack{\delta \in [q-1]^m \\ \sum_{i=1}^m \delta_i \leq \alpha_{m+1}}} \binom{\alpha_{m+1}}{\delta_1, \dots, \delta_m} \cdot C_\alpha^h \cdot \prod_{i=1}^m a_i^{\alpha_i} \cdot b^{\alpha_{m+1} - \sum \delta_i} \cdot \prod_{i=1}^m z_i^{\alpha_i + \delta_i} \end{aligned} \quad (8.14)$$

Roughly speaking, for each $\alpha \in [q-1]^{m+1}$, the exponent α_{m+1} “gets distributed” among the different z_i ’s ($i = 1, \dots, m$), and b . Note that if $\alpha_i + \delta_i = q$ then $z_i^{\alpha_i + \delta_i} = z_i$, and more generally, if $\alpha_i + \delta_i \geq q$ then $z_i^{\alpha_i + \delta_i} = z_i^{(\alpha_i + \delta_i) \bmod (q-1)}$. In what follows we use the shorthand $(j)_q$ to denote $(j \bmod (q-1))$.

For any choice of $\gamma = \gamma_1, \dots, \gamma_m$, $\gamma_i \in [q-1]$, we consider the coefficient of the term $\prod_{i=1}^m z_i^{\gamma_i}$ in the representation of $g_{a_1, \dots, a_m, b}(z_1, \dots, z_m)$ as a polynomial of degree at most $q-1$ in each variable (that is, $C_\gamma^{g_{a_1, \dots, a_m, b}}$). It follows from Equation (8.14) that this coefficient is the evaluation of the following *multivariate* polynomial,

$$\begin{aligned} & H_\gamma(x_1, \dots, x_{m+1}) \\ &= \sum_{\substack{\alpha \in [q-1]^{m+1} \\ \alpha_{m+1} \geq \sum_{i=1}^m (\gamma_i - \alpha_i)_q}} \binom{\alpha_{m+1}}{(\gamma_1 - \alpha_1)_q, \dots, (\gamma_m - \alpha_m)_q} \cdot C_\alpha^h \cdot \prod_{i=1}^m x_i^{\alpha_i} \cdot x_{m+1}^{\alpha_{m+1} - \sum_{i=1}^m (\gamma_i - \alpha_i)_q} \end{aligned} \quad (8.15)$$

at the point $x_1 = a_1, \dots, x_m = a_m, x_{m+1} = b$.

As in Cases 1 and 2, we would like to show that under the assumption that $C_{\alpha^*}^h \neq 0$ for α^* such that $\sum_{i=1}^{m+1} \alpha_i^* > d$, we can get the following. There exist a_1, \dots, a_m and b in F and $\gamma_1, \dots, \gamma_m \in [q-1]$ such that $\sum_{i=1}^m \gamma_i > d$ and the coefficient of the term $\prod_{i=1}^m z_i^{\gamma_i}$ in the representation of $g_{a_1, \dots, a_m, b}$ as a polynomial of degree at most $q-1$ (in each variable) is non-zero. Since we want to exploit the existence of

$\alpha^* \in [q-1]^{m+1}$ as stated above, we shall consider $\gamma_1, \dots, \gamma_m$ of the form $\gamma_i = \alpha_i^* + \delta_i$ where $\delta_1, \dots, \delta_m$ ($\delta_i \in [q-1]$) obey the following conditions:

- C1. $\sum_{i=1}^m \delta_i \leq \alpha_{m+1}^*$;
- C2. $\alpha_i^* + \delta_i (= \gamma_i) \leq q-1$ for every $i, 1 \leq i \leq m$;
- C3. $\sum_{i=1}^m (\alpha_i^* + \delta_i) > d$ (that is, $\sum_{i=1}^m \gamma_i > d$).
- C4. $\binom{\alpha_{m+1}^*}{\delta_1, \dots, \delta_m}$ is not divisible by p . (If q is prime, that is, $q = p$, then this condition follows from condition C1, but this is not true in general.)

Suppose we have a setting of the δ_i 's that satisfies conditions C1–C4 (where we later show how to obtain such a setting). Let $\gamma = \gamma_1, \dots, \gamma_m$ be such that $\gamma_i = \alpha_i^* + \delta_i$. By condition C3, $\sum_{i=1}^m \gamma_i > d$, and by condition C2 we have that $\gamma_i \leq q-1$ and $\delta_i = (\gamma_i - \alpha_i^*)_q$. We claim that H_γ (which is defined in Equation (8.15)) includes at least one non-zero coefficient. To verify this first note that by condition C1 (and since $\delta_i = (\gamma_i - \alpha_i^*)_q$), the sum in Equation (8.15) includes $\alpha = \alpha^*$. By our counter assumption, $C_{\alpha^*}^h \neq 0$. Combining this with condition C4 we get that $\binom{\alpha_{m+1}^*}{\delta_1, \dots, \delta_m} \cdot C_{\alpha^*}^h$ is a non-zero coefficient of the term $\prod_{i=1}^m x_i^{\alpha_i^*} \cdot x_{m+1}^{\alpha_{m+1}^* - \sum_{i=1}^m \delta_i}$ in the polynomial H_γ , so that H_γ is a non-zero polynomial. That is, there exists a choice of a_1, \dots, a_m and b on which the value of H_γ is non-zero. But by definition of H_γ this means that $C_\gamma^{g_{a_1, \dots, a_m, b}} \neq 0$ for γ that satisfies $\sum_{i=1}^m \gamma_i > d$, in contradiction to the induction hypothesis.

It remains to show how we find a setting of the δ_i 's that satisfies conditions C1–C4.

Subcase 1: q is prime. Consider first the case that q is prime. That is, $q = p$. In this case $m \geq \ell = \left\lceil \frac{d+1}{q-1} \right\rceil$. Let $\delta_1 = q-1 - \alpha_1^*$, and recall that, by the premise of this case, for every i, j we have $\alpha_i^* + \alpha_j^* \geq q$, so that necessarily $\delta_1 < \alpha_{m+1}^*$. Next let $\delta_2 = \min\{q-1 - \alpha_2^*, \alpha_{m+1}^* - \delta_1\}$, and in general, $\delta_i = \min\{q-1 - \alpha_i^*, \alpha_{m+1}^* - \sum_{j<i} \delta_j\}$. Conditions C1 and C2 directly follow from the definition of the δ_i 's, and condition C4 is implied by C1 (since q is prime). It remains to verify that condition C3 holds. If there exists an index i such that $\delta_i = \alpha_{m+1}^* - \sum_{j<i} \delta_j$ then

$$\sum_{i=1}^m (\alpha_i^* + \delta_i) = \sum_{i=1}^{m+1} \alpha_i^* > d \quad (8.16)$$

Otherwise, $\delta_i = q-1 - \alpha_i^*$ for every $1 \leq i \leq m$ and so

$$\sum_{i=1}^m (\alpha_i^* + \delta_i) = m \cdot (q-1) \geq \ell \cdot (q-1) \geq d+1 \quad (8.17)$$

Subcase 2: q is not prime. When q is not a prime number, so that $q = p^s$ for $s > 1$, then the setting of the δ_i 's is a bit more involved because condition C4 does not follow from any of the other conditions, and we have to attend to is separately. Since

$$\binom{\alpha_{m+1}^*}{\delta_1, \dots, \delta_m} = \binom{\alpha_{m+1}^*}{\delta_1} \cdot \binom{\alpha_{m+1}^* - \delta_1}{\delta_2} \cdots \binom{\alpha_{m+1}^* - \sum_{j<i} \delta_j}{\delta_i} \cdots \binom{\alpha_{m+1}^* - \sum_{j<m} \delta_j}{\delta_m} \quad (8.18)$$

it suffices to show that each term in the above product is not divisible by p . Let us hence rewrite condition C4:

- C4. For every $1 \leq i \leq m$, $\binom{\alpha_{m+1}^* - \sum_{j<i} \delta_j}{\delta_i}$ is not divisible by p .

We shall use the following notation: For each α_i^* , let $k_{i,j}$ for $j \in [p-1]$ be such that $\alpha_i^* = \sum_{j=0}^{s-1} k_{i,j} p^j$. That is, in the representation of α_i^* in basis p , $k_{i,j}$ is the coefficient of p^j . Let us assume without loss of generality that α_{m+1}^* is the smallest α_i^* and that $\alpha_1^* \leq \alpha_2^* \leq \dots \leq \alpha_m^*$ (we may re-order the variables to get that). We know that $\alpha_1^* < (p-1)p^{s-1}$ (or else $\sum_{i=1}^m \alpha_i^* \geq (p-1)p^{s-1} \cdot \ell \geq d+1$). Since, by the premise of this case, $\alpha_i^* + \alpha_1^* \geq q = p^s$ for every $i > 1$, necessarily $\alpha_i^* > p^{s-1}$ for $i > 1$. Similarly, $\alpha_1^* > p^{s-1}$ (since $\alpha_1^* + \alpha_{m+1}^* \geq q$). Hence for each α_i^* we have that $1 \leq k_{i,s-1} \leq p-1$.

For $1 \leq i \leq m$, let $t_i \stackrel{\text{def}}{=} p-1 - k_{i,s-1}$ (for technical purposes $t_0 \stackrel{\text{def}}{=} 0$). That is, t_i indicates the maximum number that can be added to the coefficient $k_{i,s-1}$ of p^{s-1} . Recall that $\alpha_1^*, \dots, \alpha_m^*$ are in non-decreasing order, and so the t_i 's are in non-increasing order. In particular, if $t_i = 0$ then $t_{i'} = 0$ for every $i' > i > 0$. We shall use the following technical claim that was given in [34], and whose proof is provided here for completeness.

Claim 8.2.1 *Let $q = p^s$ for a prime number p and an integer s , and let r and t be integers that satisfy $0 < r \leq t \leq q-1$. If $r = kp^{s-1}$ for some integer k then $\binom{t}{r}$ is not divisible by p .*

Proof: For any positive integer j , the largest power of p that divides $j!$ is

$$\lfloor j/p \rfloor + \lfloor j/p^2 \rfloor + \lfloor j/p^3 \rfloor + \dots$$

But for $r = kp^{s-1}$, the identity $\lfloor n/p^i \rfloor = \lfloor r/p^i \rfloor + \lfloor (n-r)/p^i \rfloor$ holds. Thus the largest power of p that divides $n!$ is

$$\sum_{i=1}^{\infty} \lfloor n/p^i \rfloor = \sum_{i=1}^{\infty} (\lfloor r/p^i \rfloor + \lfloor (n-r)/p^i \rfloor).$$

Therefore $n!$ and $r!(n-r)!$ are divisible by exactly the same power of p . ■

The setting of the δ_i 's. We now show how to set the δ_i 's so that conditions C1–C4 hold. We start with an informal description of how to “distribute” the weight of α_{m+1}^* between the δ_i 's. Consider the representation of α_{m+1}^* in basis p as described above. The highest coefficient in the representation is $k_{m+1,s-1}$. That is

$$\alpha_{m+1}^* = \sum_{j=0}^{s-1} k_{m+1,j} p^j = k_{m+1,s-1} p^{s-1} + \sum_{j=0}^{s-2} k_{m+1,j} p^j. \quad (8.19)$$

We view α_{m+1}^* as having $k_{m+1,s-1}$ “units of p^{s-1} ” to distributed, and some “left-over”. Note that this left-over, $\sum_{j=0}^{s-2} k_{m+1,j} p^j$, is strictly smaller than p^{s-1} .

Starting from δ_1 , each δ_i in its turn will be assigned the maximum possible integer multiple of p^{s-1} . Namely, as long as the number of remaining “ p^{s-1} units” is more than the number of such units that can still be added to α_i^* (i.e., t_i) and $t_i > 0$, then δ_i is assigned t_i units of p^{s-1} . For these i 's we get that $k_{i,s-1} p^{s-1} + \delta_i = (p-1)p^{s-1}$. If we reach an index i such that $t_i = 0$, then we stop distributing what is left of α_{m+1}^* . Alternatively, if we reach an index i for which the number of p^{s-1} units that can be added to it (i.e., t_i) is bigger than the amount of p^{s-1} units that remain to be distributed, then we assign δ_i the rest of the weight of α_{m+1}^* that was not distributed yet.

We now turn to a more formal definition. We initialize i to 1 and do the following:

1. While $i \leq m$ and $0 < t_i \leq k_{m+1,s-1} - \sum_{j < i} t_j$:

Set $\delta_i = t_i p^{s-1}$, and increase i by 1.

2. If $i \leq m$:

- (a) If $t_i = 0$ then for every i' , $i \leq i' \leq m$, set $\delta_{i'} = 0$. (Recall that the t_i 's are non-increasing so that if $t_i = 0$ then for every $i' > i$ we also have $t_{i'} = 0$.)

- (b) Else ($t_i > k_{m+1,s-1} - \sum_{j<i} t_j$), set $\delta_i = \alpha_{m+1}^* - \sum_{j<i} \delta_j$, and for every $i' > i$ set $\delta_{i'} = 0$.
Note that

$$\delta_i = \left(k_{m+1,s-1} - \sum_{j<i} t_j \right) p^{s-1} + \sum_{j=0}^{s-2} k_{m+1,j} p^j < t_i p^{s-1}$$

We next verify that conditions C1–C4 hold for this setting of the δ_i 's. Condition C1 ($\sum_{i=1}^m \delta_i \leq \alpha_{m+1}^*$) directly follows from the above process. By the definition of the δ_i 's, for every $1 \leq i \leq m$, $\delta_i \leq t_i \cdot p^{s-1}$. Since $t_i = p - 1 - k_{i,s-1}$, we get that

$$\alpha_i^* + \delta_i \leq (p-1) \cdot p^{s-1} + \sum_{\ell=2}^s k_{i,s-\ell} p^{s-\ell} < q \quad (8.20)$$

and so condition C2 holds.

We next verify that condition C3 holds, that is, $\sum_{i=1}^m (\alpha_i^* + \delta_i) > d$. Let i_0 be the index reached at the end of Step 1 in the process. Observe that for every $i < i_0$, $\delta_i = t_i \cdot p^{s-1}$. By definition of t_i this implies that $\alpha_i^* + \delta_i \geq (p-1)p^{s-1}$. If $i_0 > m$ then, since $m \geq \ell = \left\lceil \frac{d+1}{(p-1)p^{s-1}} \right\rceil$, we get that $\sum_{i=1}^m (\alpha_i^* + \delta_i) \geq d+1$, as required. If $i_0 \leq m$, then there are two cases. In case $t_{i_0} = 0$ (so that for every $i' \geq i_0$ we have $t_{i'} = 0$), then $\alpha_i^* \geq (p-1)p^{s-1}$ for every $i \geq i_0$, so again we get that $\sum_{i=1}^m (\alpha_i^* + \delta_i) \geq d+1$. In case $t_{i_0} > k_{m+1,s-1} - \sum_{j<i_0} t_j$, then we set $\delta_{i_0} = \alpha_{m+1}^* - \sum_{j<i_0} \delta_j$, so that

$$\sum_{i=1}^m (\alpha_i^* + \delta_i) = \sum_{i=1}^{m+1} \alpha_i^* > d. \quad (8.21)$$

Finally, we verify that condition C4 holds. That is, for every $1 \leq i \leq m$, $\binom{\alpha_{m+1}^* - \sum_{j<i} \delta_j}{\delta_i}$ is not divisible by p . Let i_0 be as defined above in our verification of condition C3. Since for every $i < i_0$ we have that $\delta_i = t_i p^{s-1}$, by Claim 8.2.1, $\binom{\alpha_{m+1}^* - \sum_{j<i} \delta_j}{\delta_i}$ is not divisible by p . If $i_0 > m$ then we are done. Otherwise there are two cases. In the first case $\delta_i = 0$ for every $i \geq i_0$, so clearly the condition holds. In the second case, $\delta_{i_0} = \alpha_{m+1}^* - \sum_{j<i_0} \delta_j$ and $\delta_i = 0$ for every $i > i_0$. Thus condition C4 holds in this case too. We have thus completed the proof of Case 3 (Subcase 2), and hence of Theorem 8.1.2. ■

8.3 The Test

In this section we present and analyze our testing algorithm for degree- d polynomials over fields of cardinality $q = O(d)$.

Algorithm 1 Testing Algorithm for Degree- d Polynomials

1. Let $\ell = \ell(q, d) = \left\lceil \frac{d+1}{q-d/p} \right\rceil$ and repeat the following $t = \Theta\left(\ell \cdot q^{\ell+1} + \frac{1}{\epsilon \cdot q^\ell}\right)$ times:

(a) Uniformly and independently select ℓ linearly independent points $y_1, \dots, y_\ell \in F^n$, and a point $y_0 \in F^n$.

(b) If $f|_{(y_0, y_1, \dots, y_\ell)} \notin \text{POLY}_{\ell, d}$ then output reject.

2. If no step caused rejection then output accept.

Recall that checking whether $f|_S \notin \text{POLY}_{\ell, d}$ (where $S = S(y_0, y_1, \dots, y_\ell)$) can be done by querying f on all points in the subspace S and verifying that all linear constraints corresponding to the coefficients

$C_\alpha^{f|_S}$ such that $\sum_{i=1}^{\ell} \alpha_i > d$, hold. Hence the total number of queries performed by the algorithm is $O(t \cdot q^\ell) = O(\ell \cdot q^{2\ell+1} + \frac{1}{\epsilon})$ (where the q^ℓ term is due to the number of points in each affine subspace.)

As noted in the introduction, when q is sufficiently larger than d so that $\ell = 1$ (the subspaces are lines), then it is not necessary to query f on all points on the line, but rather $d + 2$ points suffice. We note that these checks involving $d + 2$ points on a line can be interpreted as selecting minimum-weight words from the dual GRM code and checking that they are orthogonal to the word defined by f . Our test can be modified so that instead of checking a set of constraints (several small-weight words in the dual code) in each step, it also selects one random constraint (one small-weight word in the dual code) in each step.² However, this will not reduce the query complexity in our case.

Given Algorithm 1, Theorem 8.1.1, which was stated in the introduction, follows from the next lemma.

Lemma 8.3.1 *If $f \in \text{POLY}_{n,d}$ then Algorithm 1 accepts with probability 1, and if $\text{dist}(f, \text{POLY}_{n,d}) > \epsilon$ then Algorithm 1 rejects with probability at least $2/3$.*

Lemma 8.3.1 shall be proved using the “self-correcting approach”, which has been applied in the analysis of many previous low-degree tests. Namely, given the function f we define another function g based on certain “majority votes” of f . We then show that if f passes the test with sufficiently high probability, then g is close to f and g is a polynomial of degree at most d . Bounding the distance between f and g follows easily from the definition of g , and hence the analysis is focused on showing that g is a polynomial of degree at most d . The analysis can be viewed as generalizing both the analysis in [66] (where the subspaces considered by the test are lines) and the analysis in [4] (where the subspaces are larger but the field is $GF(2)$), and the analysis relies on the fact that the field is $GF(2)$.

We start by introducing several notations.

Definition 8.3.1 *Let*

$$\eta = \eta(f, d) \stackrel{\text{def}}{=} \Pr_{y_0, y_1, \dots, y_\ell} \left[f|_{(y_0, y_1, \dots, y_\ell)} \notin \text{POLY}_{\ell, d} \right] \quad (8.22)$$

where the probability is taken over y_0, y_1, \dots, y_ℓ such that y_1, \dots, y_ℓ are linearly independent.

By definition of Algorithm 1, η is the probability that a single step of the algorithm causes f to be rejected. That is, it is the probability that the restriction of f to a random affine subspace of dimension ℓ is not a polynomial of degree at most d .

Definition 8.3.2 *For each $\alpha \in [q - 1]^\ell$, $y \in F^n$, and linearly independent points $y_1, \dots, y_\ell \in F^n$, let $C_\alpha^f(y_0, y_1, \dots, y_\ell)$ denote the coefficient C_α of the polynomial representation of $f|_{(y_0, y_1, \dots, y_\ell)}$. We shall use the notation $B^f(y_0, y_1, \dots, y_\ell)$ as a shorthand for the coefficient $C_{(q-1, \dots, q-1)}^f(y_0, y_1, \dots, y_\ell)$. That is, $B^f(y_0, y_1, \dots, y_\ell)$ denotes the coefficient of the highest-degree monomial $x_1^{q-1} \cdot x_2^{q-1} \cdots x_\ell^{q-1}$ in the polynomial representation of $f|_{(y_0, y_1, \dots, y_\ell)}$. Recall that this coefficient equals $(-1)^\ell$ times the sum of the values of f taken over all points in the subspace.*

We denote by $V^f(y; y_1, \dots, y_\ell)$ the value that $f(y)$ “should have” so that $B^f(y, y_1, \dots, y_\ell) = 0$. That is,

$$V^f(y; y_1, \dots, y_\ell) = - \sum_{\substack{b_1, \dots, b_\ell \in F \\ \exists i \text{ s.t. } b_i \neq 0}} f\left(y + \sum_{i=1}^{\ell} b_i \cdot y_i\right). \quad (8.23)$$

We refer to $V^f(y; y_1, \dots, y_\ell)$ as the **vote** of (y_1, \dots, y_ℓ) on the value assigned to y .

For succinctness of the notation, we shall remove f from the last two notations (i.e., $B(\cdot) = B^f(\cdot)$ and $V(\cdot) = V^f(\cdot)$).

²In case q is prime then, as shown in [45], it suffices to consider a single constraint per affine subspace.

Note that for η as in Definition 8.3.1,

$$\eta \geq \Pr_{y, y_1, \dots, y_\ell} [V(y; y_1, \dots, y_\ell) \neq f(y)] \quad (8.24)$$

where the probability is taken over y_1, \dots, y_ℓ that are linearly independent. This is true since the test checks that all coefficients $C_\alpha^f(y, y_1, \dots, y_\ell)$ for which $\sum_{i=1}^\ell \alpha_i > d$ are 0.

In our analysis, we shall sometimes have to address the case that y_1, \dots, y_ℓ are linearly dependent and we shall use the notation $V(y; y_1, \dots, y_\ell)$ (as defined in Equation (8.23)), in this case as well. This is despite the fact that it no longer has the same meaning of a “vote” on the value of $f(y)$ (or at least not an “objective vote”). We show:

Lemma 8.3.2 *For every $y \in F^n$ and for $y_1, \dots, y_\ell \in F^n$ that are linearly dependent, $V(y; y_1, \dots, y_\ell) = f(y)$.*

Proof: Since y_1, \dots, y_ℓ are linearly dependent, we can write y_ℓ as a linear combination of the other points. That is, $y_\ell = \sum_{i=1}^{\ell-1} a_i y_i$, where $a_1, \dots, a_{\ell-1} \in F$. By definition of $V(\cdot)$,

$$V(y; y_1, \dots, y_\ell) = - \sum_{b_1, \dots, b_\ell \in F} f\left(y + \sum_{i=1}^\ell b_i \cdot y_i\right) + f(y) \quad (8.25)$$

Since $y_\ell = \sum_{i=1}^{\ell-1} a_i y_i$,

$$\sum_{b_1, \dots, b_\ell \in F} f\left(y + \sum_{i=1}^\ell b_i \cdot y_i\right) = \sum_{b_1, \dots, b_{\ell-1} \in F} f\left(y + \sum_{i=1}^{\ell-1} b_i \cdot y_i + b_\ell \sum_{i=1}^{\ell-1} a_i y_i\right) \quad (8.26)$$

$$= \sum_{b_\ell \in F} \sum_{b_1, \dots, b_{\ell-1} \in F} f\left(y + \sum_{i=1}^{\ell-1} (b_i + b_\ell \cdot a_i) y_i\right) \quad (8.27)$$

$$= |F| \cdot \sum_{c_1, \dots, c_{\ell-1} \in F} f\left(y + \sum_{i=1}^{\ell-1} c_i y_i\right) \quad (8.28)$$

$$= 0 \quad (8.29)$$

In the above sequence of equalities, Equation (8.28) follows from the fact that for each $b_\ell \in F$, and for every choice of $c_1, \dots, c_{\ell-1} \in F$, there exists a choice of $b_1, \dots, b_{\ell-1} \in F$ such that $c_i = b_i + b_\ell \cdot a_i$ (i.e., $b_i = c_i - b_\ell \cdot a_i$). ■

We are now ready to define the *self corrected* version of f , denoted g .

Definition 8.3.3 *Let g be a plurality function that is defined as follows. For each $y \in F^n$,*

$$g(y) = \operatorname{argmax}_{a \in F} \left\{ \Pr_{y_1, \dots, y_\ell \in F^n} [V(y; y_1, \dots, y_\ell) = a] \right\} \quad (8.30)$$

The next lemma readily follows from the definition of g .

Lemma 8.3.3 *For any function f and for η and g as defined in Equations (8.22) and (8.30) respectively, $\operatorname{dist}(f, g) \leq 2\eta$.*

Proof: First observe that if the test selects points $y_0, y_1, \dots, y_\ell \in F^n$ such that $f(y_0) \neq V(y_0; y_1, \dots, y_\ell)$, then this means that $B(y_0; y_1, \dots, y_\ell) \neq 0$ (where $B(\cdot)$ is as defined in Definition 8.3.2), which causes the test to reject. Recall that the test selects y_1, \dots, y_ℓ that are linearly independent. If y_1, \dots, y_ℓ are linearly dependent then by Lemma 8.3.2, $V(y_0; y_1, \dots, y_\ell) = f(y_0)$. Let $U \subseteq F^n$ consist of all (“bad”) points $y \in F^n$ such that $\Pr_{y_1, \dots, y_\ell \in F^n} [f(y) \neq V(y; y_1, \dots, y_\ell)] > 1/2$. By definition of η (and Lemma 8.3.2) we know that $|U|/q^n < 2\eta$. But for every $x \in F^n \setminus U$, by definition of g we have that $f(x) = g(x)$, and the lemma follows. ■

In the next series of lemmas we prove that if η is sufficiently small then g is a polynomial of total degree at most d . In the first, and central lemma, we show that for every y , the value of $g(y)$, which by Definition 8.3.3 is the “plurality vote” of $V(y; y_1, \dots, y_\ell)$, taken over all y_1, \dots, y_ℓ , equals the vote of a large fraction of the ℓ -tuples y_1, \dots, y_ℓ (assuming η is sufficiently small).

Lemma 8.3.4 For any fixed $y \in F^n$, let

$$\gamma(y) \stackrel{\text{def}}{=} \Pr_{y_1, \dots, y_\ell} [V(y; y_1, \dots, y_\ell) = g(y)] \quad (8.31)$$

Then $\gamma(y) \geq 1 - 2q\ell\eta$.

In order to prove Lemma 8.3.4 it will actually be more convenient to work with another measure of “correctness” (or “consistency”) of a point y .

Lemma 8.3.5 For any fixed $y \in F^n$, let

$$\delta(y) = \Pr_{y_1, \dots, y_\ell, z_1, \dots, z_\ell} [V(y; y_1, \dots, y_\ell) = V(y; z_1, \dots, z_\ell)] \quad (8.32)$$

and let $\gamma(y)$ be as defined in Equation (8.31). Then $\gamma(y) \geq \delta(y)$.

Proof: Let $\beta_a(y) = \Pr_{y_1, \dots, y_\ell} [V(y; y_1, \dots, y_\ell) = a]$ (so that in particular, $\sum_{a \in F} \beta_a(y) = 1$). By definition of $\gamma(y)$ we have that $\gamma(y) = \max_a \beta_a(y)$, and by definition of $\delta(y)$ we have that, $\delta(y) = \sum_{a \in F} (\beta_a(y))^2$. By convexity, $\max_a \beta_a(y) \geq \sum_{a \in F} (\beta_a(y))^2$, and the claim follows. ■

An Auxiliary “Voting Graph”. In order to show that $\delta(y)$ is large (and hence $\gamma(y)$ is large), it will be useful to consider the following auxiliary graph. The definition of this graph was inspired by the way Shpilka and Wigderson used Cayley graphs in their work [68] and can also be viewed as formalizing and generalizing part of the analysis in [4]. Each vertex in this graph is labeled by a subset (multiset) of ℓ points, $\{y_1, \dots, y_\ell\}$, $y_i \in F^n$. The neighbors of $\{y_1, \dots, y_\ell\}$ are of the form $\{y_2, \dots, y_{\ell+1}\}$. Each vertex corresponds to ℓ points that can “vote” on the value of $f(y)$ for any given y and hence we refer to it as the *voting graph*.

For a fixed point $y \in F^n$, we say that an edge between $\{y_1, \dots, y_\ell\}$ and $\{y_2, \dots, y_{\ell+1}\}$ is *good with respect to y* if $V(y; y_1, \dots, y_\ell) = V(y; y_2, \dots, y_{\ell+1})$.

Recall that for linearly independent y_1, \dots, y_ℓ , $B(y_0, y_1, \dots, y_\ell)$ denotes the coefficient $C_{\langle q-1, \dots, q-1 \rangle}(y_0, y_1, \dots, y_\ell)$, of the restriction of f to the ℓ -dimensional affine subspace determined

by y_0, y_1, \dots, y_ℓ . That is, $B(y_0, y_1, \dots, y_\ell) = (-1)^\ell \cdot \sum_{b_1, \dots, b_\ell \in F} f\left(y + \sum_{i=1}^{\ell} y_i b_i\right)$.

Lemma 8.3.6 For any choice of $y, y_1, \dots, y_{\ell+1} \in F^n$ such that $y_1, \dots, y_{\ell+1}$ are linearly independent,

$$\begin{aligned} & V(y; y_1, \dots, y_\ell) - V(y; y_2, \dots, y_{\ell+1}) \\ &= (-1)^\ell \left(\sum_{a \in F, a \neq 0} B(y + a \cdot y_{\ell+1}, y_1, \dots, y_\ell) - \sum_{a \in F, a \neq 0} B(y + a \cdot y_1, y_2, \dots, y_{\ell+1}) \right) \end{aligned}$$

Proof: By definition of $V(y; \cdot)$ we have:

$$\begin{aligned} & V(y; y_1, \dots, y_\ell) - V(y; y_2, \dots, y_{\ell+1}) \\ &= - \sum_{\substack{b_1, \dots, b_\ell \in F \\ b_1 \neq 0}} f\left(y + \sum_{i=1}^{\ell} b_i \cdot y_i\right) + \sum_{\substack{b_2, \dots, b_{\ell+1} \in F \\ b_{\ell+1} \neq 0}} f\left(y + \sum_{i=2}^{\ell+1} b_i \cdot y_i\right) \end{aligned} \quad (8.33)$$

$$\begin{aligned} &= - \sum_{\substack{b_1, \dots, b_\ell \in F \\ b_1 \neq 0}} f\left(y + \sum_{i=1}^{\ell} b_i \cdot y_i\right) - \sum_{\substack{b_1, \dots, b_{\ell+1} \in F \\ b_1, b_{\ell+1} \neq 0}} f\left(y + \sum_{i=1}^{\ell+1} b_i \cdot y_i\right) \\ &+ \sum_{\substack{b_2, \dots, b_{\ell+1} \in F \\ b_{\ell+1} \neq 0}} f\left(y + \sum_{i=2}^{\ell+1} b_i \cdot y_i\right) + \sum_{\substack{b_1, \dots, b_{\ell+1} \in F \\ b_1, b_{\ell+1} \neq 0}} f\left(y + \sum_{i=1}^{\ell+1} b_i \cdot y_i\right) \end{aligned} \quad (8.34)$$

$$= - \sum_{\substack{b_1, \dots, b_{\ell+1} \in F \\ b_1 \neq 0}} f\left(y + b_1 \cdot y_1 + \sum_{i=2}^{\ell+1} b_i \cdot y_i\right) + \sum_{\substack{b_1, \dots, b_{\ell+1} \in F \\ b_{\ell+1} \neq 0}} f\left(y + b_{\ell+1} \cdot y_{\ell+1} + \sum_{i=1}^{\ell} b_i \cdot y_i\right) \quad (8.35)$$

$$= (-1)^\ell \left(\sum_{a \in F, a \neq 0} B(y + a \cdot y_{\ell+1}, y_1, \dots, y_\ell) - \sum_{a \in F, a \neq 0} B(y + a \cdot y_1, y_2, \dots, y_{\ell+1}) \right) \quad (8.36)$$

■

Proof of Lemma 8.3.4: Given Lemma 8.3.5, it suffices to show that for every $y \in F^n$, $\delta(y) \geq 1 - 2q\ell\eta$. For any (random) choice of y_1, \dots, y_ℓ and z_1, \dots, z_ℓ , and for each $0 \leq i \leq \ell$ let $v_i = \{y_1, \dots, y_i, z_{i+1}, \dots, z_\ell\}$, where we view v_i as a vertex in the voting graph. In particular, $v_\ell = \{y_1, \dots, y_\ell\}$ and $v_0 = \{z_1, \dots, z_\ell\}$. Since $y_1, \dots, y_\ell, z_1, \dots, z_\ell$ are selected uniformly and random, each v_i is a random variable. Consider the path v_ℓ, \dots, v_0 between v_ℓ and v_0 . In what follows we shall use the shorthand $V(y; v_i)$ for the vote $V(y; y_1, \dots, y_i, z_{i+1}, \dots, z_\ell)$. Recall that an edge (v_i, v_{i-1}) is good if $V(y; v_i) = V(y; v_{i-1})$.

We next show that the probability (taken over the choice of $y_1, \dots, y_\ell, z_1, \dots, z_\ell$) that an edge (v_i, v_{i-1}) on the path is not good is at most $2q\eta$. By taking a union bound it follows that the probability that all the edges on the path are good is at least $1 - 2q\ell\eta$. That is, with probability at least $1 - 2q\ell\eta$, $V(y; y_1, \dots, y_\ell) = V(y; y_1, \dots, y_{\ell-1}, z_\ell) = \dots = V(y; z_1, \dots, z_\ell)$, and the lemma follows.

Consider any edge (v_i, v_{i-1}) . We say that $V(y; v_i)$ is an *independent vote* for y if $y_1, \dots, y_i, z_{i+1}, \dots, z_\ell$ are linearly independent points, otherwise we say that $V(y; v_i)$ is a *dependent vote* for y . If both votes for y are dependent then by Lemma 8.3.2, $V(y; v_i) = f(y)$ and $V(y; v_{i-1}) = f(y)$ so that $V(y; v_i) = V(y; v_{i-1})$ and the edge is good. If one of the votes is a dependent vote and the other is an independent vote then the probability that the edge is not good is the probability that an independent vote for y differs from $f(y)$, which is η .

We next show that if both $V(y; v_i)$ and $V(y; v_{i-1})$ are independent votes for y then the edge (v_i, v_{i-1}) is not good with probability at most $2q\eta$. Indeed, by Lemma 8.3.6 such an edge is good if

$$\sum_{a \in F, a \neq 0} B(y + a \cdot y_i, y_1, \dots, y_{i-1}, z_i, \dots, z_\ell) - \sum_{a \in F, a \neq 0} B(y + a \cdot z_i, y_1, \dots, y_i, z_{i+1}, \dots, z_\ell) = 0 \quad (8.37)$$

Since $y_1, \dots, y_i, z_{i+1}, \dots, z_\ell$ and $y_1, \dots, y_{i-1}, z_i, \dots, z_\ell$ are two sets of linearly independent vectors selected uniformly at random, each of the $B(\cdot)$'s in the above summation is non-zero with probability at most η . Hence, by applying a union bound, the probability that (v_i, v_{i-1}) is not good is at most $2q\eta$ as claimed.

■

We next show that if η is sufficiently small then $g \in \text{POLY}_{n,d}$. This is obtained by showing that the restriction of g to every affine subspace of dimension ℓ results in a polynomial of degree at most d . By applying Theorem 8.1.2 we conclude that in such a case g is indeed in $\text{POLY}_{n,d}$. In order to show that the restriction of g to every affine subspace of dimension ℓ is a polynomial of degree at most d , we generalize the proof technique applied in [4] for the case of $F = GF(2)$. Roughly speaking, we show that the high degree coefficients in the polynomial representation of the restriction of g to any *fixed* subspace, can be expressed as linear combinations of these coefficients in the restriction of f to *random* subspaces.

Lemma 8.3.7 *If $\eta < \frac{1}{2(\ell+1)q^{\ell+1}}$ then $g \in \text{POLY}_{n,d}$.*

Proof: Consider any fixed set of points $y_0, y_1, \dots, y_\ell \in F^n$ such that y_1, \dots, y_ℓ are linearly independent. We shall show that $g_{|(y_0, y_1, \dots, y_\ell)} \in \text{POLY}_{\ell,d}$. Lemma 8.3.7 follows by applying Theorem 8.1.2. We start by describing the high-level idea of the proof. By using a probabilistic argument we shall show that there exists a choice of a subset of elements, denoted $\{z_{i,j}\}$, for which the following conditions hold. First, every point w in the subspace $S(y_0, y_1, \dots, y_\ell)$ equals the vote on w of a set, T_w , of ℓ points that are linear combinations of the $z_{i,j}$'s. Next, for each of these sets of points T_w , the restriction of f to the affine subspace defined by w and T_w , is a polynomial of degree at most d . That is, all high degree coefficients in each of these restrictions are 0. We then show that each high degree coefficient in the restriction of g to the subspace $S(y_0, y_1, \dots, y_\ell)$ is a linear combinations of the high degree coefficients in the abovementioned restrictions of f , and is hence 0. A formal proof follows.

Each point in the affine subspace $S(y_0, y_1, \dots, y_\ell)$ is of the form $y_0 + \sum_{i=1}^{\ell} b_i y_i$, where $b_i \in F$. Now consider $(\ell + 1) \cdot \ell$ elements in F^n , denoted $\{z_{i,j}\}_{i=0, \dots, \ell}^{j=1, \dots, \ell}$. Suppose that for every choice of $b_1, \dots, b_\ell \in F$,

$$g\left(y_0 + \sum_{i=1}^{\ell} b_i \cdot y_i\right) = V\left(y_0 + \sum_{i=1}^{\ell} b_i \cdot y_i; z_{0,1} + \sum_{i=1}^{\ell} b_i \cdot z_{i,1}, \dots, z_{0,\ell} + \sum_{i=1}^{\ell} b_i \cdot z_{i,\ell}\right) \quad (8.38)$$

If we select the elements $\{z_{i,j}\}$ uniformly and at random, then by Lemma 8.3.4, this event occurs with probability at least $1 - 2q\ell\eta \cdot q^\ell$. We assume from this point on that Equation (8.38) holds for every choice of $b_1, \dots, b_\ell \in F$.

In order to show that $g_{|(y_0, y_1, \dots, y_\ell)} \in \text{POLY}_{\ell,d}$ we need to show that for every $\alpha \in [q - 1]^\ell$ such that $\sum_{i=1}^{\ell} \alpha_i > d$, we have that $C_\alpha^g(y_0, y_1, \dots, y_\ell) = 0$. Let us fix any such α , and let R_α denote the row vector $\mathcal{A}_\ell(\alpha, \cdot)$. Recall that the coordinates of R_α are indexed by strings $\beta \in [q - 1]^\ell$ (where we denote the corresponding coordinate by $R_\alpha^\beta \in F$). In what follows we use the notation: $ex(\beta_i) = \omega^{\beta_i}$ if $\beta_i \neq 0$, and $ex(\beta_i) = 0$ if $\beta_i = 0$. Consider the structure of \mathcal{A}_ℓ presented in Section 2. We need to show that

$$\sum_{\beta \in [q-1]^\ell} R_\alpha^\beta \cdot g\left(y_0 + \sum_{i=1}^{\ell} ex(\beta_i) \cdot y_i\right) = 0 \quad (8.39)$$

For any fixed $\beta \in [q - 1]^\ell$, by our assumption that Equation (8.38) holds, and by definition of $V(\cdot)$ we have:

$$\begin{aligned} & g\left(y_0 + \sum_{i=1}^{\ell} ex(\beta_i) \cdot y_i\right) \\ &= - \sum_{\substack{\gamma \in [q-1]^\ell \\ \gamma \neq (0,0,\dots,0)}} f\left(y_0 + \sum_{i=1}^{\ell} ex(\beta_i) \cdot y_i + \sum_{j=1}^{\ell} ex(\gamma_j) \cdot \left(z_{0,j} + \sum_{i=1}^{\ell} ex(\beta_i) \cdot z_{i,j}\right)\right) \end{aligned}$$

$$= - \sum_{\substack{\gamma \in [q-1]^\ell \\ \gamma \neq (0,0,\dots,0)}} f \left(y_0 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{0,j} + \sum_{i=1}^{\ell} \text{ex}(\beta_i) \cdot \left(y_i + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{i,j} \right) \right) \quad (8.40)$$

This implies (by switching the order of summations) that

$$\begin{aligned} & \sum_{\beta \in [q-1]^\ell} R_\alpha^\beta \cdot g \left(y_0 + \sum_{i=1}^{\ell} \text{ex}(\beta_i) \cdot y_i \right) \\ &= - \sum_{\substack{\gamma \in [q-1]^\ell \\ \gamma \neq (0,0,\dots,0)}} \sum_{\beta \in [q-1]^\ell} R_\alpha^\beta \cdot f \left(y_0 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{0,j} + \sum_{i=1}^{\ell} \text{ex}(\beta_i) \cdot \left(y_i + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{i,j} \right) \right) \end{aligned} \quad (8.41)$$

But for any given choice of $\gamma = \gamma_1, \dots, \gamma_\ell$,

$$\begin{aligned} & \sum_{\beta \in [q-1]^\ell} R_\alpha^\beta \cdot f \left(y_0 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{0,j} + \sum_{i=1}^{\ell} \text{ex}(\beta_i) \cdot \left(y_i + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{i,j} \right) \right) \\ &= C_\alpha^f \left(y_0 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{0,j}, y_1 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{1,j}, \dots, y_\ell + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{\ell,j} \right) \end{aligned} \quad (8.42)$$

Consider the event that for every choice of $\gamma_1, \dots, \gamma_\ell$ that are not all 0, $y_1 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{1,j}, \dots, y_\ell + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{\ell,j}$ are linearly independent. This event occurs with probability at least $1 - q^\ell \cdot q^{\ell-n}$. In what follows we shall assume that this is indeed the case. Since $\gamma_1, \dots, \gamma_\ell$ are not all 0, then we know that for each setting of $\gamma_1, \dots, \gamma_\ell$, with probability at most η over the choice of the $z_{i,j}$'s, for every α such that $\sum_{i=1}^{\ell} \alpha_i > d$,

$$C_\alpha^f \left(y_0 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{0,j}, y_1 + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{1,j}, \dots, y_\ell + \sum_{j=1}^{\ell} \text{ex}(\gamma_j) \cdot z_{\ell,j} \right) \neq 0 \quad (8.43)$$

By taking a union bound over all $\gamma_1, \dots, \gamma_\ell$, adding the probability that we have at least one linearly dependent combination, and adding the probability that Equation (8.38) does not hold for some $\beta \in [q-1]^\ell$, we get that with probability at least

$$1 - q^\ell \eta - 2q\ell\eta q^\ell - q^{2\ell-n} \quad (8.44)$$

there exist $z_{i,j}$'s that satisfy all required constraints. Note that our algorithm performs $\Theta(\ell q^{2\ell+1} + 1/\epsilon)$ queries, so that we may assume that $\ell q^{2\ell+1} < q^n$ (or else the algorithm would simply query all q^n points). Therefore, the expression in Equation (8.44) is greater than 0 and the lemma follows. ■

By combining Lemmas 8.3.3 and 8.3.7 we obtain that if f is $\Omega\left(\frac{1}{\ell q^\ell}\right)$ -far from $\text{POLY}_{n,d}$, then $\eta = \Omega\left(\frac{1}{\ell q^\ell}\right)$, and so the algorithm rejects f with sufficiently high constant probability.

The next lemma, which will help us deal with the case in which η is small, is a variant of a very similar lemma that was proved in [4].

Lemma 8.3.8 *Let $\zeta \stackrel{\text{def}}{=} \frac{1 - q^\ell \cdot \text{dist}(f,g)}{1 + q^\ell \cdot \text{dist}(f,g)} \cdot q^\ell \cdot \text{dist}(f,g)$. If we uniformly and independently select $y_0, y_1, \dots, y_\ell \in F^n$ where y_1, \dots, y_ℓ are linearly independent, then the probability that for exactly one choice of $b_1, \dots, b_\ell \in F$, we have that $f\left(y_0 + \sum_{i=1}^{\ell} b_i \cdot y_i\right) \neq g\left(y_0 + \sum_{i=1}^{\ell} b_i \cdot y_i\right)$, is at least ζ .*

Proof: For each $\beta = \beta_1, \dots, \beta_\ell, \beta_i \in [q-1]$ let X_β be the indicator random variable whose value is 1 if and only if $f\left(y_0 + \sum_{i=1}^\ell ex(\beta_i)y_i\right) \neq g\left(y_0 + \sum_{i=1}^\ell ex(\beta_i)y_i\right)$. Thus $\Pr[X_\beta = 1] = \text{dist}(f, g)$ for every β . It is not difficult to verify that the random variables X_β are pairwise independent. This is true since for any two distinct β^1, β^2 , the points $\left(y_0 + \sum_{i=1}^\ell ex(\beta_i^1)y_i\right)$ and $\left(y_0 + \sum_{i=1}^\ell ex(\beta_i^2)y_i\right)$ attain each pair of distinct values in F^n with equal probability. It follows that the random variable $X = \sum_\beta X_\beta$, which counts the number of points $v = \left(y_0 + \sum_{i=1}^\ell ex(\beta_i)y_i\right)$ in which $f(v) \neq g(v)$, has expectation $E[X] = q^\ell \cdot \text{dist}(f, g)$ and variance $\text{Var}[X] = q^\ell \cdot \text{dist}(f, g) \cdot (1 - \text{dist}(f, g)) \leq E[X]$. Our objective is to lower bound the probability that $X = 1$. We need the well known fact that for a random variable X that attains nonnegative, integer values,

$$\Pr[X > 0] \geq \frac{(E[X])^2}{E[X^2]}. \quad (8.45)$$

Indeed, if X attains the value i with probability ν_i for $i > 0$, then, by Cauchy-Schwartz,

$$(E[X])^2 = \left(\sum_{i>0} i\nu_i\right)^2 = \left(\sum_{i>0} i\sqrt{\nu_i}\sqrt{\nu_i}\right)^2 \leq \left(\sum_{i>0} i^2\nu_i\right) \cdot \left(\sum_{i>0} \nu_i\right) = E[X^2] \cdot \Pr[X > 0]. \quad (8.46)$$

In our case, this implies

$$\Pr[X > 0] \geq \frac{(E[X])^2}{E[X^2]} \geq \frac{(E[X])^2}{E[X] + (E[X])^2} = \frac{E[X]}{1 + E[X]}. \quad (8.47)$$

Therefore

$$E[X] \geq \Pr[X = 1] + \left(\frac{E[X]}{1 + E[X]} - \Pr[X = 1]\right) \cdot 2 = \frac{2E[X]}{1 + E[X]} - \Pr[X = 1], \quad (8.48)$$

implying that

$$\Pr[X = 1] \geq \frac{E[X] - (E[X])^2}{1 + E[X]}. \quad (8.49)$$

Substituting the value of $E[X]$, the desired result follows. \blacksquare

We are now ready to wrap-up the proof of Lemma 8.3.1 (and hence Theorem 8.1.1).

Proof of Lemma 8.3.1: As we have noted previously, if f is in $\text{POLY}_{n,d}$, then by Theorem 8.1.2 the tester accepts (with probability 1). We next show that if f is ϵ -far from $\text{POLY}_{n,d}$, then the tester rejects with probability at least $\frac{2}{3}$.

Suppose that $\text{dist}(f, \text{POLY}_{n,d}) > \epsilon$. We shall show that $\eta \geq \min\left\{\frac{1}{2}q^\ell\epsilon, \frac{1}{2(\ell+1)q^{\ell+1}}\right\}$. Since η is the probability that a single iteration of the algorithm causes f to be rejected, and the algorithm performs $\Theta(1/\eta)$ iterations, the theorem follows. If $\eta \geq \frac{1}{2(\ell+1)q^{\ell+1}}$ then we are done. Hence, assume that $\eta < \frac{1}{2(\ell+1)q^{\ell+1}}$. We shall show that in such a case $\eta \geq \frac{1}{2} \cdot q^\ell \cdot \text{dist}(f, g) > \frac{1}{2} \cdot q^\ell \cdot \epsilon$. To verify this, first note that by Lemma 8.3.7 we have that $g \in \text{POLY}_{n,d}$ (since $\eta < \frac{1}{2(\ell+1)q^{\ell+1}}$). Next observe that if f and g disagree on exactly one point in a subspace S of dimension ℓ , then $f|_S \notin \text{POLY}_{\ell,d}$. It follows from Lemma 8.3.8 and the definition of η that $\eta \geq \zeta$ (where ζ is as defined in Lemma 8.3.8). In particular, since by Lemma 8.3.3 $\text{dist}(f, g) \leq 2\eta \leq \frac{1}{(\ell+1)q^{\ell+1}}$ where $\ell \geq 1$ and $q \geq 2$, we get that

$$\eta \geq \frac{1 - q^\ell \cdot \text{dist}(f, g)}{1 + q^\ell \cdot \text{dist}(f, g)} \cdot q^\ell \cdot \text{dist}(f, g) \quad (8.50)$$

$$\geq \frac{1 - \frac{1}{q^{(\ell+1)}}}{1 + \frac{1}{q^{(\ell+1)}}} \cdot q^\ell \cdot \text{dist}(f, g) \quad (8.51)$$

$$\geq \frac{1 - 1/4}{1 + 1/4} \cdot q^\ell \cdot \text{dist}(f, g) \quad (8.52)$$

$$> \frac{1}{2} \cdot q^\ell \cdot \text{dist}(f, g) \quad (8.53)$$

as claimed. ■

8.4 A Lower Bound

Theorem 8.4.1 *Every algorithm for testing $\text{POLY}_{n,d}$ with distance parameter ϵ must perform $\Omega(\max\{\frac{1}{\epsilon}, q^{\ell-1}\})$ queries when q is prime, and $\Omega(\max\{\frac{1}{\epsilon}, q^{\lceil \ell/2 \rceil - 1}\})$ queries otherwise.*

In order to establish Theorem 8.4.1, we consider the relation between polynomials and codes. Specifically, recall that the family $\text{POLY}_{n,d}$ over a field $F = \text{GF}(q) = \text{GF}(p^s)$, corresponds to the Generalized Reed-Muller (GRM) code $\mathcal{GRM}_q(d, n)$. Namely, each codeword (having length q^n) is determined by the evaluation of a polynomial in $\text{POLY}_{n,d}$ on all points in the domain F^n . The minimum distance, $\Delta(\mathcal{GRM}_q(d, n))$, of the code is the following (cf. [31]): If $d = r(q-1) + t$, where $0 \leq t < q-1$, and r is an integer, then $\Delta(\mathcal{GRM}_q(d, n)) = (q-t)q^{n-r-1}$. The dual code of $\Delta(\mathcal{GRM}_q(d, n))$ is the GRM code $\mathcal{GRM}_q(n(q-1) - (d+1), n)$, so that it has distance $\Omega(q^{\lfloor \frac{d+1}{q-1} \rfloor - 1})$. Let us denote the distance of the dual code by $\overline{\Delta}(\mathcal{GRM}_q(d, n))$, and let $\ell = \lfloor \frac{d+1}{q-1} \rfloor$ be as in our previous notation. Hence, if q is prime then $\overline{\Delta}(\mathcal{GRM}_q(d, n)) = \Omega(q^{\ell-1})$, and for non-prime q we can say that $\overline{\Delta}(\mathcal{GRM}_q(d, n)) = \Omega(q^{\lceil \ell/2 \rceil - 1})$.

Theorem 8.4.1 follows by applying the theorem below, which is a straightforward generalization of a similar theorem proved in [4] for binary codes.

Theorem 8.4.2 *Let \mathcal{F} be any family of functions $f : F^n \rightarrow F$ that corresponds to a linear code \mathcal{C} . Let $\Delta(\mathcal{C})$ denote the minimum distance of the code \mathcal{C} and let $\overline{\Delta}(\mathcal{C})$ denote the minimum distance of the dual code of \mathcal{C} .*

Every testing algorithm for the family \mathcal{F} must perform $\Omega(\overline{\Delta}(\mathcal{C}))$ queries, and if the distance parameter ϵ is at most $\Delta(\mathcal{C})/(2q^n)$, then $\Omega(1/\epsilon)$ is also a lower bound for the necessary number of queries.

Theorem 8.4.2 *Let \mathcal{F} be any family of functions $f : F^n \rightarrow F$ that corresponds to a linear code \mathcal{C} . Let $\Delta(\mathcal{C})$ denote the minimum distance of the code \mathcal{C} and let $\overline{\Delta}(\mathcal{C})$ denote the minimum distance of the dual code of \mathcal{C} .*

Every testing algorithm for the family \mathcal{F} must perform $\Omega(\overline{\Delta}(\mathcal{C}))$ queries, and if the distance parameter ϵ is at most $\Delta(\mathcal{C})/(2q^n)$, then $\Omega(1/\epsilon)$ is also a lower bound for the necessary number of queries.

Proof: We start by showing that $\Omega(\overline{\Delta}(\mathcal{C}))$ queries are necessary. A well known fact from coding theory (see [58, Chap. 1, Thm. 10]) states the following: for every linear code \mathcal{C} whose dual code has distance $\overline{\Delta}(\mathcal{C})$, if we examine a sub-word having length Δ' , where $\Delta' < \overline{\Delta}(\mathcal{C})$, of a uniformly selected codeword in \mathcal{C} , then the resulting sub-word is uniformly distributed in $F^{\Delta'}$. Hence it is not possible to distinguish between a random codeword in \mathcal{C} and a random word in $F^{\Delta'}$ (which with high probability is far from any codeword) using less than $\overline{\Delta}$ queries.

We now turn to the case $\epsilon < \Delta/2q^n$. To prove the lower bound here, we apply, as usual, the Yao principle by defining two distributions, one of positive instances, and the other of negative ones, and then showing that in order to distinguish between those distributions any algorithm must perform $\Omega(1/\epsilon)$ queries.

The positive distribution has all its mass at the zero vector $\bar{0} = (0, \dots, 0)$. To define the negative distribution, partition the set of all coordinates randomly into $t = 1/\epsilon$ nearly equal parts I_1, \dots, I_t and give weight $1/t$ to each of the characteristic vectors w_i of I_i , $i = 1, \dots, t$. (Observe that indeed $\bar{0} \in \mathcal{C}$ due to linearity, and $\text{dist}(w_i, \mathcal{C}) = \epsilon$ due to the assumption on the minimum distance of \mathcal{C}). Finally, a random instance is generated by first choosing one of the distributions with probability $1/2$, and then generating a vector according to the chosen distribution.

Consider the two distributions that were defined. Let A be a deterministic testing algorithm with query complexity s (where s is a function of ϵ). We need to show that if A gives an incorrect answer with probability at most $1/3$, it must be that $s > 1/(3\epsilon)$. If A is incorrect on $\bar{0}$ (that is, it does not accept it), then it is already incorrect with probability at least $1/2$. Otherwise A should accept the input if all the s queried bits are 0. Therefore it accepts as well at least $t - s$ (where $t = 1/\epsilon$ is as defined above) of the inputs w_i . This shows that A gives an incorrect answer with probability at least $(t - s)/2t$. For this to be smaller than $1/3$ it must be the case that $s > 1/(3\epsilon)$. ■

Chapter 9

A Characterization of Low-Weight Words that Span Generalized Reed Muller Codes

9.1 Introduction

The generalized Reed Muller code $\mathcal{R}_{F_q}(\rho, m)$ consists of all words of length q^m that correspond to the evaluations of m -variate polynomials of total degree at most ρ over F_q^m . We denote by p the characteristic of the field F_q , so that p is prime, and $q = p^t$ for an integer $t \geq 1$.

In the work of Delsarte, Goethals and MacWilliams [30] the minimum weight words of these codes are characterized (see also [13, Thm. 5.25]). These words have weight $(q - s)q^{m-r-1}$ where $\rho = r(q - 1) + s$, and each is a sum of multiples of incidence vectors of $(q - s)$ parallel affine subspaces (flats) of dimensions $(m - r - 1)$ that are contained in an $(m - r)$ -dimensional affine subspace. Delsarte et. al. also show that these words span the code in the case that q is prime.

Ding and Key [31] consider general fields, and ask under what conditions do the minimum weight words span the code. They show that the minimum weight words span the code if *and only if* one of the following conditions holds: (1) $m = 1$; (2) q is prime; (3) $\rho < p$; (4) $\rho > (m - 1)(q - 1) + p^{t-1} - 2$. We note that the fourth condition was also shown to be sufficient by Friedl and Sudan [34], in the context of their work on testing low-degree polynomials.

9.2 The Characterization

In the following we show that for all generalized Reed-Muller codes (and in particular for those codes that are *not* spanned by their minimum-weight words), there exists a subset of words that span the code whose weight is at most quadratic in the weight of the minimum-weight words. More precisely:

Theorem 9.2.1 *Let $\mathcal{R}_{F_q}(\rho, m)$ denote the q -ary generalized Reed Muller code of order ρ and length q^m , where $q = p^t$, p is prime, $t \geq 1$, and $q \leq \rho \leq m(q - 1)$. Then there is a set of words of weight at most $q^{\left\lceil \frac{m(q-1)-\rho}{q-q/p} \right\rceil}$ that span $\mathcal{R}_{F_q}(\rho, m)$.*

In particular, when q is prime (so that $q = p$) and ρ is divisible by $q - 1$, then we obtain the minimum-weight words of the code. If q is not prime, then the weight of the words spanning the code is roughly the minimum weight of words in $\mathcal{R}_{F_q}(\rho, m)$ taken to the power of $p/(p - 1)$.

Our analysis works by proving a characterization of (total) degree- ρ multivariate polynomials over the field F_q . This characterization takes the form of linear constraints on the evaluation of the polynomials taken over points that belong to affine subspaces of low dimension. Since the generalized Reed-Muller

code of order ρ , $\mathcal{R}_{F_q}(\rho, m)$, is determined by polynomials of degree ρ over F_q^m , these linear constraints corresponds to low-weight words that are orthogonal to the words of the code. The characterization implies that these words span the dual code. Because the dual code of $\mathcal{R}_{F_q}(\rho, m)$ is the generalized Reed-Muller code $\mathcal{R}_{F_q}(m(q-1) - (\rho+1), q)$, we can obtain Theorem 9.2.1.

Theorem 9.2.1 follows from the following theorem presented in Chapter 8.

Theorem 8.1.2 *Let $F = \text{GF}(q)$ where $q = p^s$ and p is prime. Let d be an integer, and let $f : F^n \rightarrow F$. Then f is a polynomial of degree at most d , if and only if its restriction to every affine subspace of dimension $\ell = \left\lceil \frac{d+1}{q-q/p} \right\rceil$ is a polynomial of degree at most d .*

9.2.1 Proof of the Theorem

As stated in the introduction, our characterization of low-weight words that span generalized Reed-Muller codes, referred to in Theorem 9.2.1, is derived from Theorem 8.1.2. We next show how Theorem 9.2.1 follows from Theorem 8.1.2.

Recall that Theorem 8.1.2 says that a function $f : F^m \rightarrow F$ is a polynomial of degree at most ρ if and only if its restriction to every affine subspace of dimension $\ell = \left\lceil \frac{\rho+1}{q-q/p} \right\rceil$ is a polynomial of degree at most ρ . Theorem 8.1.2 can be restated equivalently as follows: $f \in \text{POLY}_{\rho, m}$ if and only if there exists a subset of vectors $\mathcal{W} \subset F^{q^m}$, such that for each $\vec{w} \in \mathcal{W}$ we have:

1. $\vec{w} \cdot \vec{f} = 0$;
2. The non-zero coordinates of \vec{w} are all indexed by points that belong to some affine subspace of dimension ℓ in F^m .

Since $f \in \text{POLY}_{\rho, m}$ if and only if $\vec{f} \in \mathcal{R}_{F_q}(\rho, m)$, the first item implies that the subset of vectors \mathcal{W} spans the code dual to $\mathcal{R}_{F_q}(\rho, m)$, which is $\mathcal{R}_{F_q}(m(q-1) - (\rho+1), m)$. The second item implies that the weight of every $\vec{w} \in \mathcal{W}$ is at most $q^\ell = q^{\left\lceil \frac{\rho+1}{q-q/p} \right\rceil}$. Theorem 9.2.1 follows by a simple substitution of variables.

Chapter 10

Almost Orthogonal Linear Codes are Locally Testable

10.1 Introduction

In this work we study the testability of some families of linear codes.

Locally Testable Codes (LTC) and Regular Testers. A *tester* for a code is a randomized algorithm that is given a string w and a distance parameter $\epsilon > 0$. The tester is allowed to perform oracle queries about values of coordinates of w . It should accept if w is a codeword and reject with high probability if w is ϵ -far from every codeword. The notion ϵ -far indicates that at least ϵ -fraction of the coordinates of w should be changed for obtaining a codeword of the code. A code is *locally testable* if it has a tester that performs a number of queries that is a function of ϵ only, and is independent of the length of the code.

A *local test* for w is a selection of a codeword w' from the dual code, and verification that w' is orthogonal to w . Such verification requires several queries into w . The number of queries is equal to the weight of w' . If w' is not orthogonal to w then this implies that w is not a codeword. In such case a *violation* is detected. This leads to the following definition of a regular tester. Roughly speaking, a regular tester is a tester that performs several local tests into w , such that each local test involve k queries into w , where k is independent of ϵ .

Definition 10.1.1 A (k, δ) -**regular tester** for a code is a tester that, given w and ϵ , selects $O(\frac{1}{\delta})$ random codewords of weight k from the dual code ($\delta = \delta(k, \epsilon) > 0$). Then it performs the corresponding local tests. If a violation occurs the tester rejects, otherwise it accepts. The probability that a single local test rejects is denoted $Rej(\epsilon)$.

Note that most of the known testers for linear codes are regular.

Related Research. Locally testable codes have been a subject of much research over the last years due to their close relation to *probabilistically checkable proofs* (PCP). The question of characterizing codes that are locally testable is highly complex. For surveys on the issue see [36, 69]. A great deal of attention was devoted to testing polynomial codes, that is, the codes whose codewords are evaluations of some polynomials over a finite field. Various families of polynomial codes differ in the size of the field, the maximum degree of the polynomials, and the number of variables. The study of testing polynomial codes was initiated by [24] who considered the *Hadamard codes*, based on multivariate linear functions over a binary field. Other relevant works include [15, 17, 18, 19, 24, 33] and references therein. In [9, 15, 14, 33, 34, 66] it

was shown that *Reed-Solomon codes* and *Reed-Muller codes* are locally testable. These codes are based on univariate and multivariate low-degree polynomials over *large* finite fields. Recently [4, 45, 50] proved that Reed-Muller codes over general fields, including small and even binary fields are locally testable. Most testers for polynomial codes use the *self-correction approach*. A different series of works [20, 41], initiated by [41], attempt proving existence of locally testable codes possessing good parameters (e.g. constant rate and linearly growing minimum distance).

10.1.1 Our Results for General Codes

A Sufficient Condition for Local Testability of Linear Codes. In this work we present a sufficient condition for local testability of linear codes. Let an ϵ -far coset of a code be the code obtained by adding to every codeword a vector that is ϵ -far from the code. The condition is somewhat combinatorial. It is based on counting the number of fixed weight codewords in the dual code to the union of the code and its ϵ -far coset. If this number is substantially smaller than the number of the fixed length codewords in the dual of the code itself, we claim that the code is locally testable. Hence, the approach we present here is different from the *self-correction approach* (presented in e.g. [4, 24, 50, 66]). Note that there exist linear codes that can not be locally tested, as shown by Ben-Sasson *et al.* [21].

Regular Local Testability Implies Short Basis for the Dual Code. We show that local testability of a code by a regular tester implies that the dual code is spanned by short codewords. Note that the opposite does not hold. That is, there exist codes [21] whose dual is spanned by short (length 3) words, which are not locally testable. Note that in the works on testing codes that used the self-correction approach, there was a need, prior to the construction of a tester, to prove an existence of a short basis of the dual code (noted as “characterization” in works using the self-correction approach). Here we avoid this complication, since the local testability of the primary code implies the existence of a short basis for the dual code.

Our Main Result: Almost Orthogonal Linear Codes are Locally Testable. We say that a code C of length n is *almost orthogonal* if the minimum distance of C is $\frac{n}{2} - \Theta(\sqrt{n})$. It follows from upper bounds on the size of codes as a function of the minimum distance (see e.g. [56]) that the size of such codes is at most polynomial in n . We use our sufficient condition for testability to show that the almost orthogonal linear codes are locally testable. Moreover, we show that their dual codes are spanned by words of fixed weights. Specifically we show the following:

Theorem 10.1.1 *For a linear code C , if the distance of C is at least $\frac{n}{2} - \sqrt{tn}$ then C is locally testable using $O(t/\epsilon)$ queries. Moreover, C^\perp , the dual to C , is spanned by its words of weight at most $t + 2$.*

10.1.2 Our Results for Dual-BCH Codes and the BLR Test

Dual-BCH Codes. Dual-BCH(n, t) codes (denoted $C_{dBCH(t)}$) are generalizations of the well studied Hadamard codes ($t = 1$ is Hadamard). They can be defined as binary trace images of evaluations of univariate polynomials of degree $2t$ over the finite fields of size $(n + 1)$ and characteristic 2. The motivation for the current work stems from an open problem raised in [4]. Alon *et al.* asked whether $C_{dBCH(t)}$ are locally testable for constant t . Notice that $C_{dBCH(t)}$ are generalizations of codes defined by linear functions (Hadamard codes). They have been used extensively for derandomization and construction of epsilon-biased sets (see e.g. [1, 63]). Recently Khot [52] used them to obtain better inapproximability results for SVP. The $C_{dBCH(t)}$ codes for constant t are known (by the A. Weil-Carlitz-Uchiyama theorem) to be almost orthogonal. Hence, we conclude that these codes are locally testable. Their dual codes are the conventional BCH(n, t) codes (denoted $C_{BCH(t)}$).

Results for Dual-BCH and BCH Codes.

- $C_{dBCH(t)}$ are locally testable. They can be tested using $O(t/\epsilon)$ queries by a regular tester such that $Rej(\epsilon) \geq \epsilon$. The lower bound for this problem is $\Omega(\frac{1}{\epsilon} + t)$. The regular tester uses codewords of $C_{BCH(t)}$ of weight $2t + 3$. The lower bound follows from the minimum distance of $C_{BCH(t)}$.
- $C_{BCH(t)}$ is spanned by its almost shortest words, that is, by words of weight at most $2t + 2$, while the shortest are of weight $2t + 1$. Here, we do believe that the code is spanned by its shortest words.
- Extended $C_{BCH(t)}$ code, $C_{eBCH(t)}$, obtained by appending all-coordinates parity-check to each codeword, is spanned by its shortest words of weight $2t + 2$.
- There is an explicit procedure that generates a random codeword of constant weight from $C_{BCH(t)}$ with complexity $O(poly(t \log n))$.

Note that our results implies local testability of Goppa codes [23] and trace subcodes of algebraic-geometric codes.

Back to the BLR Test. The Hadamard test of Blum, Luby, and Rubinfeld [24] is given a binary vector v of length $2^m - 1$. The test selects uniformly at random $x, y \in \{0, 1\}^m$ and verifies that $v(x) + v(y) = v(x + y)$. Note that $x, y, x + y$ describe the non-zero coordinates of a random 3-weight codeword from the code dual to the Hadamard code (Hamming code). In various papers [17, 18, 24, 53] the following question was addressed: What is the lower bound on $Rej(\epsilon)$ of the BLR test [24]. $Rej(\epsilon)$ is important since it is related to the hardness of approximation of some NP-hard problems, see [17] for a relevant discussion. The best known bounds are described in [17, 53], showing that $Rej(\epsilon) \geq \epsilon - O(\frac{1}{n})$, for every ϵ . The result of [17] is based on Fourier transform, the one of [53] is based on discrete Fourier transform and a use of Krawtchouk polynomials. In this work we re-prove the bound $Rej(\epsilon) \geq \epsilon - O(\frac{1}{n})$. Our proof is somewhat simpler than the previous ones, and uses our general techniques.

10.1.3 Our Techniques

We use tools from coding theory to bound the *spectra* (weight distributions) of the code and its dual. We do that by using a linear programming approach based on the *MacWilliams transform*. This transform relates the weight distribution of a code to the one of its dual. We prove a *generalization of the Johnson Bound* which provides us with a tool to obtain an upper bound on the distribution of the number of codewords in a Hamming sphere and its moments. An essential point in our analysis is use of the *Karamata inequality* allowing extending of inequalities for the first moments of the distribution to higher ones. An orthonormal matrix is norm preserving. This is expressed in the Parseval identity, and is useful in our analysis of the Hadamard tester. For the case of general almost orthogonal codes, we prove a *generalized Parseval inequality*, namely, we show that when the vectors are almost orthogonal, the matrix is norm preserving up to an explicitly derived multiplicative constant.

The techniques we employ are quite different from the ones used earlier. We believe that they might be useful for proving local testability of other codes.

10.2 The Approach Applied in This Work

In this section we describe the approach applied in this work and provide a sufficient condition for local testability of linear codes. Let C be a linear code whose dual code is C^\perp . Consider all codewords of C^\perp of weight k , denoted C_k^\perp . Their number is $B_k^{C^\perp}$. The codewords C_k^\perp imply local tests of size k over C . Hence, in order to show that a code C is locally testable it is sufficient to show that there exists a constant

k such that for a vector w that is ϵ -far from C and for c' selected uniformly from C_k^\perp , the probability that $\langle w, c' \rangle \neq 0$, is a constant $\delta(\epsilon)$ that depends only on ϵ . In such case the code C is testable using $\Theta(\frac{k}{\delta(\epsilon)})$ queries.

Consider a codeword w that is ϵ -far from C , that is, $w = c + v_{\epsilon n}$, such that $c \in C$ and $v_{\epsilon n}$ is ϵ -far from any codeword in C . Suppose that we could show that at least $\delta(\epsilon)$ -fraction of codewords in C_k^\perp are such that their inner product with $v_{\epsilon n}$ is non-zero. Then we would get that by taking uniformly at random $c' \in C_k^\perp$, with probability $\delta(\epsilon)$, $\langle c', w \rangle \neq 0$. That is, with probability $\delta(\epsilon)$ the tester rejects. Hence we conclude that C is testable using $\Theta(k/\delta)$ queries.

In order to show that at least $\delta(\epsilon)$ -fraction of words in C_k^\perp are such that their inner product with $v_{\epsilon n}$ is non-zero, we could equivalently show that for at most $(1 - \delta(\epsilon))$ -fraction of the codewords in C_k^\perp , their inner product with $v_{\epsilon n}$ is zero. The last could be shown using the following approach.

The codewords of C^\perp that are orthogonal both to C and to $v_{\epsilon n}$ are codewords that are orthogonal to the code $C \cup v_{\epsilon n}$. That is, these are the weight k words of $[C \cup v_{\epsilon n}]^\perp$ (denoted $[C \cup v_{\epsilon n}]_k^\perp$). Their number is denoted as $B_k^{[C \cup v_{\epsilon n}]^\perp}$. In order to find how many such weight k words exist in $[C \cup v_{\epsilon n}]^\perp$, we need the study the weight distribution of $[C \cup v_{\epsilon n}]^\perp$. Following, is a definition capturing the properties discussed above.

Definition 10.2.1 Consider a linear code C of length n , such that its dual is C^\perp . The code C has (k, δ) -**Coset-Property** if for every $\frac{1}{n} \leq \epsilon \leq \frac{R(C)}{n}$, there exist non-decreasing $\delta = \delta(\epsilon, k) > 0$, such that for every vector $v_{\epsilon n}$ at distance ϵn from C , $B_k^{[C+v_{\epsilon n}]^\perp} \leq (1 - 2\delta)B_k^{C^\perp}$.

Note that for a vector $v_{\epsilon n} \notin C$, its distance from C is between 1 to $R(C)$. Thus, ϵ is chosen to cover all possible distances of $v_{\epsilon n}$ from C .

In the following we provide our sufficient condition for local testability of linear codes.

Theorem 10.2.1 A sufficient condition for local testability: Consider a linear code C of length n , such that its dual is C^\perp . If C has (k, δ) -Coset-Property then it is testable using $\Theta(\frac{k}{\delta})$ queries. Hence, if k is a constant independent of n then the code is locally testable.

Proof: We show that the condition $B_k^{[C+v_{\epsilon n}]^\perp} \leq (1 - 2\delta)B_k^{C^\perp}$ implies $B_k^{[C \cup v_{\epsilon n}]^\perp} \leq (1 - \delta)B_k^{C^\perp}$.

Recall that,

$$B_k^{[C \cup v_{\epsilon n}]^\perp} = \frac{1}{|C \cup v_{\epsilon n}|} \sum_{i=0}^n B_i^{C \cup v_{\epsilon n}} P_k(i) = \frac{1}{2|C|} \sum_{i=0}^n B_i^C P_k(i) + \frac{1}{2|C|} \sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_k(i) = \frac{B_k^{C^\perp}}{2} + \frac{B_k^{[C+v_{\epsilon n}]^\perp}}{2}$$

It remains to show that the condition $B_k^{[C \cup v_{\epsilon n}]^\perp} \leq (1 - \delta)B_k^{C^\perp}$ implies that C is testable using $\Theta(\frac{k}{\delta})$ queries. The last can be verified easily by the definitions. The details of the proof follows.

Assume that for code C and its dual C^\perp the condition applies. In the following we use the condition to construct a one-sided tester for the code C where its query complexity is $\Theta(\frac{k}{\delta})$, for k, δ as specified in the theorem. Given a word c the algorithm should accepts if $c \in C$, and it should rejects w.h.p if c is in distance of at least ϵn from any word in C .

Test- C -Algorithm (c)

1. Repeat the following check $\Theta(\frac{1}{\delta})$ times:
 - (a) Uniformly and independently select a word w of length k from C^\perp .
 - (b) Verify whether $\langle c, w \rangle = 0$. If not then the check failed.
2. If any of the above checks failed than reject, otherwise accept.

Following we prove that the algorithm **Test- C -Algorithm** (c) is a one-sided tester for the code C with query complexity $\Theta(\frac{k}{\delta})$. We need to show that if c is at distance at least ϵn from C , then the algorithm rejects with probability at least $2/3$.

Suppose c is ϵ' -far from the code, where $\epsilon' \geq \epsilon$. In this case c can be presented as $c = a + v_{\epsilon'n}$, where $a \in C$, and $v_{\epsilon'n}$ is at distance $\epsilon'n$ from C . Given a random k -weight $w \in C^\perp$, $\langle c, w \rangle = \langle a, w \rangle + \langle v_{\epsilon'n}, w \rangle = \langle v_{\epsilon'n}, w \rangle$. By the assumed conditions of the theorem, $\text{Prob}(\langle v_{\epsilon'n}, w \rangle \neq 0) = \delta(\epsilon') \geq \delta(\epsilon)$. Thus, the probability to reject in a single round of the algorithm is at least $\delta(\epsilon)$. Hence, after $\Theta(\frac{1}{\delta})$ trials c is rejected with probability at least $\frac{2}{3}$. Note that the query complexity of the described algorithm is $\Theta(\frac{k}{\delta})$ as required. ■

Next we show that the coset-property of C implies that C^\perp is spanned by short words.

Theorem 10.2.2 *Consider a code C with dual C^\perp . If C has (k, δ) -Coset-Property then C^\perp is spanned by words of weight k .*

Proof: (k, δ) -Coset-Property implies that for every $\frac{1}{n} \leq \epsilon \leq \frac{R(C)}{n}$, there exists a non-decreasing sequence $\delta = \delta(\epsilon, k) > 0$, such that for every vector $v_{\epsilon n}$ being at distance ϵn from C , $B_k^{[C \cup v_{\epsilon n}]^\perp} \leq (1 - \delta)B_k^{C^\perp}$. Assume that the weight k codewords C_k^\perp do not span C^\perp . Then, there exists C^* that contains C_k^\perp and some other codewords from C^\perp , such that its size is half of the size of C^\perp . Hence, we conclude that $C^{*\perp}$, that is dual to C^* , is $C^{*\perp} = C \cup v_{\epsilon n}$, where $v_{\epsilon n} \in C^\perp$. However, due to the (k, δ) -Coset-Property we know that the number of words of weight k in $C^* = [C \cup v_{\epsilon n}]^\perp$ is at most $(1 - \delta)B_k^{C^\perp}$, and this contradicts the fact that C^* contains $B_k^{C^\perp}$ words of weight k . Hence, C^\perp is spanned by its weight k words. ■

Clearly from definitions, if C has a (k, δ) -regular tester, then it has (k, δ) -Coset-Property. It was shown in Theorem 10.2.2 that if a code C has the (k, δ) -Coset Property, then its dual C^\perp , is spanned by words of weight k . Hence the following theorem is established:

Theorem 10.2.3 *If C has a (k, δ) -regular tester then its dual C^\perp is spanned by words of weight k .*

10.3 A Tester for the Dual-BCH Code - First Try

In this section we show that $C_{dBCH(t)}$ is locally testable using $O(\frac{t \log t}{\epsilon})$ queries. However the tester we describe uses local tests of weights varying with ϵ . That is, in order to test if a given vector is ϵ -far from $C_{dBCH(t)}$, the tester employs local tests that use $O(\frac{t \log t}{\epsilon})$ queries on the vector. Note that each local test performs number of queries that is dependent of ϵ . Thus, the tester is not regular and does not imply a proof that the $C_{BCH(t)}$ is spanned by short words. In the next section we show a way to improve our analysis and get a regular tester. Our analysis shows that $\text{Rej}(\epsilon) \geq \epsilon - O(\frac{1}{n})$ in the case of the BLR test [24].

Theorem 10.3.1 *$C_{dBCH(t)}$ is locally testable using $O(\frac{t \log t}{\epsilon})$ queries. Moreover, there exists a regular-tester for the code that uses local tests of weight $O(\frac{t \log t}{\epsilon})$ such that $\text{Rej}(\epsilon) \geq \frac{1}{2}$.*

Proof: Consider $C_{dBCH(t)}$, $\epsilon > 0$. Let $k = \frac{t \log t}{\epsilon}$, k being an odd integer. We next show that for a vector $v_{\epsilon n}$ which is at distance ϵn from $C_{dBCH(t)}$, $B_k^{[C_{dBCH(t)} + v_{\epsilon n}]^\perp} \leq \frac{1}{2}B_k^{C_{BCH(t)}}$. This suffice by Theorem 10.2.1. Recall, that using the MacWilliams transform

$$B_k^{[C_{dBCH(t)} + v_{\epsilon n}]^\perp} = \sum_{i=0}^n B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_k(i)$$

Hence, we need to show that

$$\sum_{i=0}^n B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_k(i) \leq \frac{1}{2}B_k^{C_{BCH(t)}} \cdot |C_{dBCH(t)}| \quad (10.1)$$

In the following we bound the left-hand side of the last inequality.

By [29] the covering radius of $C_{dBCH(t)}$ is bounded from above by $\frac{n}{2} - \sqrt{tn}$. This bound on the covering radius can be proved using estimates on the minimum zero of a Krawtchouk polynomial, and the minimum distance of the dual code. We use this bound to conclude that any vector is at distance at most $(\frac{1}{2} - \frac{\sqrt{t}}{\sqrt{n}})n$ from $C_{dBCH(t)}$. Thus, we may assume that $1 \leq \epsilon n \leq \frac{n}{2} - \sqrt{tn}$. This will be useful in what follows.

Consider $C_{dBCH(t)} + v_{\epsilon n}$. Since $v_{\epsilon n}$ is at distance ϵn from $C_{dBCH(t)}$, the shortest word in $C_{dBCH(t)} + v_{\epsilon n}$ is of weight ϵn . Recall that k is odd, and note that the Krawtchouk polynomial $P_k(i)$ is negative for $i > \frac{n}{2} + \sqrt{tn}$. Using these observations and the bound from Claim 2.3.3 we get:

$$\sum_{i=0}^n B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_k(i) \leq \sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_k(i) \leq \sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B^{C_{dBCH(t)} + v_{\epsilon n}} \frac{|n - 2i|^k}{k!}$$

Note that for $\epsilon n \leq i \leq \frac{n}{2} + \sqrt{tn}$, we have $|n - 2i| \leq (1 - 2\epsilon)n$. Thus,

$$\sum_{i=0}^n B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_k(i) \leq \sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B^{C_{dBCH(t)} + v_{\epsilon n}} \frac{|n - 2i|^k}{k!} \leq \frac{(1 - 2\epsilon)^{k-2t} n^{k-2t}}{k!} \sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)} + v_{\epsilon n}} |n - 2i|^{2t}$$

In Claim 10.3.1 we will prove that $\sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)} + v_{\epsilon n}} |n - 2i|^{2t} \leq n^t |C_{dBCH(t)}|^{\frac{(2t)!}{2^t t!}}$. By showing that we get

$$\sum_{i=0}^n B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_k(i) \leq \frac{(1 - 2\epsilon)^{k-2t} n^{k-2t} \cdot n^t |C_{dBCH(t)}|^{\frac{(2t)!}{2^t t!}}}{k!} \leq \frac{1}{2} B_k^{C_{BCH(t)}} \cdot |C_{dBCH(t)}|.$$

The last inequality follows from Claim 2.3.5 which asserts that $B_k^{C_{BCH(t)}} = \frac{\binom{n}{k}}{(n+1)^t} (1 + O(\frac{1}{n}))$. Hence equation (10.1) follows.

Claim 10.3.1 Generalized Parseval Bound: For a linear code C of length n and minimum dual distance $2t + 1$, and a vector $v_{\epsilon n}$ being at distance ϵn from C ,

$$\sum_{i=\epsilon n}^n B_i^{C + v_{\epsilon n}} (n - 2i)^{2t} \leq n^t |C|^{\frac{(2t)!}{2^t t!}} (1 + o(1))$$

Note that for $t = 1$ we derive an inequality following from the Parseval identity up to a factor of $(1 + o(1))$. Since the dual of $C_{dBCH(t)}$ is $C_{BCH(t)}$ and its distance is $2t + 1$, the lemma can be applied, and the proof of Theorem 10.3.1 follows.

Proof: [of Claim 10.3.1] Note that if the distance of C^\perp is $2t + 1$, then the distance of $[C_{dBCH(t)} \cup v_{\epsilon n}]^\perp$ is at least $2t + 1$.

Consider a polynomial f of degree at most $2t$. Since the Krawtchouk polynomials form an orthogonal basis, f can be written as:

$$f(i) = \sum_{k=0}^{2t} \alpha_k P_k(i)$$

Let $B_0^{C + v_{\epsilon n}}, \dots, B_n^{C + v_{\epsilon n}}$ be the weight distribution of $C + v_{\epsilon n}$.

Hence,

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} f(i) = \sum_{i=0}^n B_i^{C+v_{\epsilon n}} \sum_{k=0}^{2t} \alpha_k P_k(i) = \sum_{k=0}^{2t} \alpha_k \sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_k(i) = \alpha_0 |C + v_{\epsilon n}|$$

The last equality is derived from the MacWilliams Transform. Recall that,

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_k(i) = |C + v_{\epsilon n}| (2B_k^{[C_{dBCH(t)} \cup v_{\epsilon n}]^\perp} - B_k^{C^\perp})$$

As we know that the minimum distance of C^\perp is at least $2t + 1$, we get that $\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_k(i) = 0$ for $k = 1, \dots, 2t$.

By taking $f = (n - 2i)^{2t}$ and using the fact from the Fourier Analysis that:

$$\alpha_0 = \frac{1}{2^n} \sum_{i=0}^n \binom{n}{i} f(i)$$

we get that,

$$\sum_{i=\epsilon n}^n B_i^{C+v_{\epsilon n}} (n - 2i)^{2t} \leq \alpha_0 |C + v_{\epsilon n}| = \frac{1}{2^n} \sum_{i=0}^n \binom{n}{i} (n - 2i)^{2t} |C + v_{\epsilon n}|$$

Note that $|C + v_{\epsilon n}| = |C|$. Hence, in order to complete the proof it is sufficient to show that

$$\frac{1}{2^n} \sum_{i=0}^n \binom{n}{i} (n - 2i)^{2t} \leq n^t \frac{(2t)!}{2^t t!} (1 + o(1))$$

Since the binomial distribution converges to the normal one we deduce that:

$$\frac{1}{2^n} \sum_{i=0}^n \binom{n}{i} (n - 2i)^{2t} \leq \frac{1}{2^n} \sum_{i=-n}^n \frac{2^n}{\sqrt{2\pi n}} e^{-\frac{i^2}{2n}} i^{2t} \leq \frac{1}{\sqrt{2\pi n}} \int_{x=-\infty}^{\infty} x^{2t} e^{-\frac{x^2}{2n}} dx (1 + o(1))$$

The last integral is the $2t$ -th central moment of the normal distribution which is known (see e.g. Dwight, Tables of Integrals, 860.16):

$$\int_{-\infty}^{\infty} x^{2t} e^{-r^2 x^2} dx = \sqrt{\pi} \frac{1 \cdot 3 \cdot 5 \dots \cdot (2t - 1)}{2^t r^{2t+1}}.$$

Noticing that

$$1 \cdot 3 \cdot 5 \dots \cdot (2t - 1) = \frac{(2t)!}{2^t t!}$$

we obtain the claimed bound.

■ ■

10.3.1 Back to the BLR Test

Following the same lines of the proof of Theorem 10.3.1, and using the fact that orthogonal matrix preserves norm, instead of using the Generalized Parseval Bound from above, we provide a simple alternative proof for the following theorem. The first proofs for this theorem appear in [17, 53]. The detailed proof follows.

Theorem 10.3.2 *In the BLR test [24] $\text{Re}j(\epsilon) \geq \epsilon - O(\frac{1}{n})$.*

Proof: Note that by [29] the covering radius of C is $\frac{n}{2} - \sqrt{n}$, hence an n length vector is at distance at most $(\frac{1}{2} - \frac{1}{\sqrt{n}})n$ from C . Recall that it is sufficient to show that

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_3(i) \leq (1 - 2\epsilon + O(\frac{1}{n})) B_3^C \cdot |C|$$

In the following we bound the first term in the above equation.

Consider the code $C + v_{\epsilon n}$. Since $v_{\epsilon n}$ is at distance ϵn from C , the shortest word in the code $C + v_{\epsilon n}$ is of weight ϵn . Moreover, all words in C but the "zero" word are of weight exactly $\frac{n}{2}$. Hence the heaviest word in $C + v_{\epsilon n}$ is of weight at most $\frac{n}{2} + \epsilon n$, Thus:

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_3(i) = \sum_{i=\epsilon n}^{\frac{n}{2}+\epsilon n} B_i^{C+v_{\epsilon n}} P_3(i)$$

By Claim 2.3.3:

$$\sum_{i=\epsilon n}^n B_i^{\frac{n}{2}+\epsilon n} P_3(i) \leq \sum_{i=\epsilon n}^{\frac{n}{2}} B_i^{C+v_{\epsilon n}} \frac{|n - 2i|^3}{6}$$

Since the Krawtchouk polynomial $P_3(i)$ gets negative values for $i > \frac{n}{2} + \sqrt{n}$, and since for $\epsilon n \leq i \leq \frac{n}{2} + \epsilon n$, $|n - 2i| \leq (1 - 2\epsilon)n$, we get the following:

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_3(i) \leq \sum_{i=\epsilon n}^{\frac{n}{2}+\epsilon n} B_i^{C+v_{\epsilon n}} \frac{|n - 2i|^3}{6} \leq \frac{(1 - 2\epsilon)n}{6} \sum_{i=\epsilon n}^{\frac{n}{2}+\epsilon n} B_i^{C+v_{\epsilon n}} |n - 2i|^2$$

We next show that $\sum_{i=\epsilon n}^{\frac{n}{2}+\epsilon n} B_i^{C+v_{\epsilon n}} |n - 2i|^2 = n^2$.

Note that if this is indeed the case $\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_3(i) \leq \frac{(1-2\epsilon)n^3}{6}$, and the lemma follows. The last is true due to the following. Recall that we needed to show

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_3(i) \leq (1 - 2\epsilon + O(\frac{1}{n})) B_3^C \cdot |C|$$

However by lemma 2.3.5 $B_3^{C^\perp} \cdot |C| = \frac{n^3}{6}(1 - O(\frac{1}{n}))$. Thus,

$$\sum_{i=0}^n B_i^{C+v_{\epsilon n}} P_3(i) \leq \frac{(1 - 2\epsilon)n^3}{6} \leq (1 - 2\epsilon + O(\frac{1}{n})) B_3^{C^\perp} \cdot |C|$$

as required. Thus, it remains to show the following claim:

Claim 10.3.2 Parseval Identity: *For an n length binary code C . and a vector $v_{\epsilon n}$ at distance ϵn from C , the following holds: $\sum_{i=\epsilon n}^n B_i^{C+v_{\epsilon n}} |n - 2i|^2 = n^2$.*

Proof: Consider the weight distribution of $C + v_{\epsilon n}$. Denote by v' the $1, -1$ vector obtained from a binary vector v where every 0 is transformed into 1, and every 1 is transform into -1 . For $c \in C$ and $v_{\epsilon n}$, We mark by $\langle c', v_{\epsilon n}' \rangle$ the inner products of the corresponding vectors. Note that, $n - 2w(c + v_{\epsilon n}) = \langle c', v_{\epsilon n}' \rangle$. Thus,

$$\sum_{i=\epsilon n}^n B_i^{C+v_{\epsilon n}} |n - 2i|^2 = \sum_{c \in C} \langle c', v_{\epsilon n}' \rangle^2$$

Note that it remains to show that $\sum_{c \in C} \langle c', v_{\epsilon n}' \rangle^2 = n^2$.

Recall that if we consider the words of the Hadamard code written in a matrix where every 0 is transformed into 1, and every 1 is transform into -1 , we get the Hadamard matrix H_n of order $n \cdot n$. Since H_n is orthogonal it is norm preserving. Thus, for a $1, -1$ vector v' , $|v'|_2^2 = \frac{1}{n} |H_n \cdot v'|_2^2 = n$.

Hence,

$$\sum_{c \in C} \langle c', v_{\epsilon n}' \rangle^2 = |H_n \cdot v_{\epsilon n}'|_2^2 = n^2.$$

■ ■

10.4 Improved Tester for Dual-BCH Code

In this section we show that $C_{dBCH(t)}$ is locally testable using $O(\frac{t}{\epsilon})$ queries. Moreover, we describe a regular-tester that uses local tests each of weight $2t + 3$. Hence, we conclude that $C_{BCH(t)}$ is spanned by weight $2t + 3$ words. We provide our analysis for $C_{dBCH(t)}$ codes. However, in the end of the proof we observe that this proof actually applies to every almost orthogonal code. We next show that the coset-property holds in the case of $C_{dBCH(t)}$ and codewords of weight $2t + 3$ of its dual. The idea we use is as follows. By arguments similar to those used in the previous section it is sufficient to show that for a vector $v_{\epsilon n}$ that is ϵ -far from $C_{dBCH(t)}$:

$$\sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_{2t+3}(i) \leq (1 - 2\epsilon) \sum_{i=0}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)}} P_{2t+3}(i)$$

The intuition behind the last claim follows. As all non-zero codewords of $C_{dBCH(t)}$ are of weight close to $\frac{n}{2}$, and as the corresponding Krawtchouk polynomial has values close to zero around $\frac{n}{2}$, we conclude that the zero word contributes significantly to the summation above, while other codewords have negligible effect. The zero word does not appear in the coset of $C_{dBCH(t)} + v_{\epsilon n}$ since the shortest word of $C_{dBCH(t)} + v_{\epsilon n}$ is of weight ϵn . Since the rest of codewords do not contribute much to the summation we obtain the desired property.

In order to bound the left-hand side expression we use the following strategy. We partition it into two sums that we address as head (first sum) and tail (second sum).

$$\sum_{i=\epsilon n}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_{2t+3}(i) = \sum_{i=\epsilon n}^{\frac{n}{2} - a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_{2t+3}(i) + \sum_{i=\frac{n}{2} - a}^{\frac{n}{2} + \sqrt{tn}} B_i^{C_{dBCH(t)} + v_{\epsilon n}} P_{2t+3}(i)$$

Here a , to be defined in the proof, is close to $\frac{n}{2} - \sqrt{n}$. Hence, we immediately conclude that the contribution of the tail is negligible. To deal with the head we use some generalization of the Johnson Bound, to estimate $\sum_{i=\epsilon n}^{\frac{n}{2} - a} B_i^{C_{dBCH(t)} + v_{\epsilon n}}$. Then we use the Karamata inequality to deduce a bound on the head. The Karamata inequality is of help when one starts from an inequality and wishes it to be preserved under application of

a convex function to both sides (in our case the convex function is $(n - 2i)^{2t+3}$ appearing in the bound for $P_{2t+3}(i)$).

We consider the case $t > 1$. Note that for $t = 1$, $C_{dBCH(t)}$ corresponds to linear functions, and it is known to be testable by its shortest words.

10.4.1 Useful Lemmas for the Improved Tester

Lemma 10.4.1 [A Generalization of Johnson Bound]: Consider $C_{dBCH(t)} + v_{\epsilon n}$. Let $a = n^{\frac{1}{2} + \frac{1}{4(2t+3)}}$. Let c_i for $i = 1, \dots, (n+1)^t$, be the codewords of $C_{dBCH(t)} + v_{\epsilon n}$. Let $x_i = n - 2w(c_i)$. Assume that $x_1 \geq x_2 \geq \dots \geq x_{(n+1)^t}$. Denote $Q = \sum_{\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}}$. Under the given conditions the following holds:

$$Q \leq n^{1 - \frac{1}{2(2t+3)}} (1 + o(1)).$$

For $s \leq Q$, such that $s \leq n^{\frac{1}{2} - \frac{1}{100}}$, $\sum_{i=1}^s x_i \leq n\sqrt{s}(1 + o(1))$.

For $s \leq Q$, such that $s \geq n^{\frac{1}{2} + \frac{1}{100}}$, $\sum_{i=1}^s x_i \leq n^{\frac{3}{4}}s(1 + o(1))$.

For $s \leq Q$, such that $n^{\frac{1}{2} - \frac{1}{100}} \leq s \leq n^{\frac{1}{2} + \frac{1}{100}}$, $\sum_{i=1}^s x_i \leq n^{1 + \frac{1}{100}}\sqrt{s}(1 + o(1))$.

The following claim will be used in the proof of Lemma 10.4.1.

Claim 10.4.2 Consider a code C of length n with distance d and weight distribution B_0, B_1, \dots, B_n . Let $a < \frac{n}{2}$. Let $j = n - 2d$. Let $m = \sum_{i=0}^{\frac{n}{2}-a} B_i$. For such code the following holds:

$$m = \sum_{i=0}^{\frac{n}{2}-a} B_i \leq \frac{dn}{2(\frac{n}{2} + a)^2 - 2n(\frac{n}{2} + a) + dn} \quad (10.2)$$

$$\left(\sum_{i=0}^{\frac{n}{2}-a} B_i (n - 2i) \right)^2 \leq nm((m - 1)j + n) \quad (10.3)$$

This claim enables us to get bounds on $\sum_{i=0}^{\frac{n}{2}-a} B_i (n - 2i)$, while we have bounds on $\sum_{i=0}^{\frac{n}{2}-a} B_i$.

Proof: Consider the code C' that is complement to the code C in a way that the codewords of C' are the codewords of C after the transformation where 0's are replaced by 1's and 1's are replaced by 0's. Let B'_0, B'_1, \dots, B'_n be the weight distribution of C' .

Let $m' = \sum_{\frac{n}{2}+a}^n B'_i$. Clearly from the definition of C' it follows that the distance of C' is $d = \frac{n-j}{2}$ and $m' = m$. Note that m' can be bounded from above by the maximum possible words of weight $\frac{n}{2} + a$ in a code with distance d . Hence, by using Claim 2.3.1, $m = m' \leq \frac{dn}{2(\frac{n}{2}+a)^2 - 2n(\frac{n}{2}+a) + dn}$, and equation (10.2) is established.

Note that from the definition of C' it follows that $(\sum_{i=0}^{\frac{n}{2}-a} B_i (n - 2i))^2 = (\sum_{\frac{n}{2}+a}^n B'_i (2i - n))^2$. Hence for proving equation (10.3) it is sufficient to show that

$$\left(\sum_{\frac{n}{2}+a}^n B'_i (2i - n) \right)^2 \leq nm'((m' - 1)j + n) = nm((m - 1)j + n) \quad (10.4)$$

Consider the matrix V^{+-} that its rows contain the codewords of C' of weight at least $\frac{n}{2} + a$, with zero's replaced by -1 's. The number of rows of V^{+-} is $m' = m$. Let v_1, \dots, v_m denote the rows of V^{+-} . For $1 \leq i \leq n$, denote by g_i the number of 1 's in the i -th column of V^{+-} . Let g be the average number of 1 's in the columns of V^{+-} . For $1 \leq j \leq m$, denote by r_j the number of 1 's in the j -th column of V^{+-} . Let r be the average number of 1 's in the rows of V^{+-} . By definition,

$$g = \frac{1}{n} \sum_{i=1}^n g_i = \frac{1}{n} \sum_{k=\frac{n}{2}+a}^n kB'_k \quad (10.5)$$

Recall that the distance between every two rows of V^{+-} is $d = \frac{n-j}{2}$, hence the usual scalar product of any two distance rows is at least $n - 2d = j$, therefore:

$$S = \sum_{i=1}^m \sum_{k=1}^m (v_i, v_k) \leq m(m-1)j + mn \quad (10.6)$$

On the other hand,

$$S = \sum_{i=1}^n (g_i^2 + (m - g_i)^2 - 2g_i(m - g_i)) = \sum_{i=1}^n (2g_i - m)^2 \quad (10.7)$$

By Jensen inequality, $S = \sum_{i=1}^n (2g_i - m)^2 \geq n(2g - m)^2$, therefore we get

$$n^2(2g - m)^2 \leq nm((m-1)j + n) \quad (10.8)$$

However,

$$\begin{aligned} n^2(2g - m)^2 &= n\left(2\frac{\sum_{i=1}^n g_i}{n} - m\right)^2 = \left(2\sum_{i=1}^n g_i - mn\right)^2 \\ &= \left(2\sum_{i=1}^m r_i - mn\right)^2 = \left(2\sum_{i=1}^m (r_i - n)\right)^2 = \left(\sum_{i=\frac{n}{2}+a}^n B'_i(2i - n)\right)^2 \end{aligned} \quad (10.9)$$

Hence we get that $(\sum_{i=\frac{n}{2}+a}^n B'_i(2i - n))^2 \leq nm((m-1)j + n)$ and equation (10.3) is established. ■

Proof of Lemma 10.4.1: Note that the distance d of the $C_{dBCH(t)}$ code obeys $\frac{n}{2} - (t-1)\sqrt{n} \leq d \leq \frac{n}{2} + (t-1)\sqrt{n}$, by Claim 2.3.4. Hence we can get the same bounds on the distance between every two words in a coset of the code. Thus, the first equation of the lemma follows directly from the bound on the distance of $C_{dBCH(t)} + v_{en}$ and application of equation (10.2) in Claim 10.4.2. The second and third equations follow from equation (10.3) in Claim 10.4.2, where m is assigned the value s , and j is assigned the value $2(t-1)\sqrt{n}$. ■

Lemma 10.4.3 [Karamata Inequality, see [60]]: Let $x_1, \dots, x_s, y_1, \dots, y_s$ be two non-negative sequences, such that for every $q < s$, $\sum_{i=1}^q x_i \leq \sum_{i=1}^q y_i$, and such that $\sum_{i=1}^s x_i = \sum_{i=1}^s y_i$. If $f(\cdot)$ is a convex function then, $\sum_{i=1}^s f(x_i) \leq \sum_{i=1}^s f(y_i)$.

10.4.2 The Improved Tester

Theorem 10.4.1 $C_{dBCH(t)}$ is locally testable using $O(\frac{t}{\epsilon})$ queries. Moreover, there exists a regular-tester for the code that uses local tests of length $2t + 3$ such that $\text{Rej}(\epsilon) \geq \epsilon$.

Proof: For $t = 1$, $C_{dBCH(t)}$ is the Hadamard code and hence the proof is known. Consider $C_{dBCH(t)}$, $t > 1$, $\epsilon > 0$.

We next show that for a vector $v_{\epsilon n}$ that is at distance ϵn from $C_{dBCH(t)}$.

$$B_{2t+3}^{[C_{dBCH(t)}+v_{\epsilon n}]^\perp} \leq (1 - 2\epsilon)B_{2t+3}^{C_{dBCH(t)}}.$$

Thus, by Theorem 10.2.1 the proof of the current theorem applies. Similarly to the arguments used in the proof of Theorem 10.3.1 it is sufficient to show that:

$$\sum_{i=\epsilon n}^{\frac{n}{2}+\sqrt{tn}} B_i^{C_{dBCH(t)}+v_{\epsilon n}} P_{2t+3}(i) \leq (1 - 2\epsilon)B_{2t+3}^{C_{dBCH(t)}} |C_{dBCH(t)}|.$$

Note that due to the minimum distance of $C_{dBCH(t)}$:

$$B_{2t+3}^{C_{dBCH(t)}} |C_{dBCH(t)}| \leq \sum_{i=0}^{\frac{n}{2}+\sqrt{tn}} B_i^{C_{dBCH(t)}} P_{2t+3}(i) = \frac{n^{2t+3}}{(2t+3)!} + O(n^{2t+2}).$$

Hence, it is sufficient to show that:

$$\sum_{i=\epsilon n}^{\frac{n}{2}+\sqrt{tn}} B_i^{C_{dBCH(t)}+v_{\epsilon n}} P_{2t+3}(i) \leq (1 - 2\epsilon) \frac{n^{2t+3}}{(2t+3)!} + O(n^{2t+2}) \quad (10.10)$$

In the following we bound the left-hand side. Choose $a = n^{\frac{1}{2} + \frac{1}{4(2t+3)}}$. Note that $a^{2t+3} = o(n^{t+2})$. By the bound from Claim 2.3.3, and by the bound on the number of codewords in $C_{dBCH(t)}$ the following is true:

$$\begin{aligned} \sum_{i=\epsilon n}^{\frac{n}{2}+\sqrt{tn}} B_i^{C_{dBCH(t)}+v_{\epsilon n}} P_{2t+3}(i) &\leq \sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)}+v_{\epsilon n}} P_{2t+3}(i) + (n+1)^t \max_{\frac{n}{2}-a < i \leq n} (P_{2t+3}(i)) \\ &\leq \frac{1}{(2t+3)!} \left(\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)}+v_{\epsilon n}} (n-2i)^{2t+3} + (2a)^{2t+3} (n+1)^t \right) \end{aligned} \quad (10.11)$$

In the following we present two properties, whose validity is sufficient for the bound (10.10) to hold:

Head Bound: $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)}+v_{\epsilon n}} (n-2i)^{2t+3} \leq (1 - 3\epsilon)n^{2t+3} + O(n^{2t+2})$.

Tail Bound : $(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2}(\epsilon n^{2t+3} + O(n^{2t+2}))$.

We provide different proofs for different values of ϵ . We deal with two cases:

Case 1: $\frac{1}{4} \leq \epsilon \leq \frac{1}{2}$. **Case 2:** $\frac{1}{n} \leq \epsilon \leq \frac{1}{4}$.

Note that by the bounds on the covering radius of the code (see e.g. [29]), ϵ is at most $\frac{1}{2}$. Hence, the two cases cover the possible values of ϵ . Both cases are proved along the same lines. The proof is based on the Karamata inequality presented in Lemma 10.4.3. In either of the cases we shall show that the **Head Bound** and **Tail Bound** apply.

In the proof of the first case we use the fact that using claims 2.3.1 and 2.3.4, the number of words of weight exactly ϵn in $C_{dBCH(t)} + v_{\epsilon n}$ is at most

$$\frac{dn}{2(\epsilon n)^2 - 2n(\epsilon n) + dn} \leq \frac{1}{(1 - 2\epsilon)^2} (1 + o(1))$$

In the second case we use the fact that there can be only one word c_i in $C_{dBCH(t)} + v_{\epsilon n}$ of weight ϵn . The rest of the words of $C_{dBCH(t)} + v_{\epsilon n}$ are of weight at least $(\frac{1}{2} - \epsilon)n$. By Claims 2.3.1 and 2.3.4 the number of words of weight $(\frac{1}{2} - \epsilon)n$ in this code is at most

$$\frac{dn}{2(\frac{1}{2} - \epsilon)^2 - 2n(\frac{1}{2} - \epsilon) + dn} \leq \frac{1}{(2\epsilon)^2} (1 + o(1)).$$

The complete proofs of case 1 and case 2 follow.

10.4.3 Proof of Case 1 in Theorem 10.4.1

Let $a = n^{\frac{1}{2} + \frac{1}{4(2t+3)}}$, for $t > 1$. Consider the $C_{dBCH(t)} + v_{\epsilon n}$. Let c_i for $i = 1, \dots, (n+1)^t$ be the codewords of the $C_{dBCH(t)} + v_{\epsilon n}$ code. Let $x_i = n - 2w(c_i)$ and assume that $x_1 \geq x_2 \geq \dots \geq x_{(n+1)^t}$. Recall that we need to show that the following two bounds apply for the case that $\frac{1}{4} \leq \epsilon \leq \frac{1}{2}$:

Head Bound: $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} (n - 2i)^{2t+3} \leq (1 - 3\epsilon)n^{2t+3} + O(n^{2t+2})$.

Tail Bound: $(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2}(\epsilon n^{2t+3} + O(n^{2t+2}))$.

Proof of Case 1, $\frac{1}{4} \leq \epsilon \leq \frac{1}{2}$: In order to bound the head sum we use a generalization of Johnson Bound to bound the sum of codewords $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}}$.

That is, Let $Q = \sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}}$. Using Lemma 10.4.1,

$$Q \leq n^{1 - \frac{1}{2(2t+3)}} (1 + o(1)) \tag{10.12}$$

Here we have got bound on $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}}$. However, we need to bound

$$\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} (n - 2i)^{2t+3}.$$

For this we will be using the Karamata Inequality which is of help when one has some inequalities and one needs them to be preserved under a convex function that is applied to their both sides (which is our case is $(n - 2i)^{2t+3}$).

Recall that we need to bound $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} (n - 2i)^{2t+3} = \sum_{i=1}^Q x_i^{2t+3}$.

In the following we define y_i 's. $i = 1, \dots, (n+1)^t$, such that for every $s \leq Q$: $\sum_{i=1}^s x_i \leq \sum_{i=1}^s y_i$, and such that $\sum_{i=1}^Q x_i = \sum_{i=1}^Q y_i$.

We then use the Karamata Inequality and deduce that $\sum_{i=1}^Q x_i^{2t+3} \leq \sum_{i=1}^Q y_i^{2t+3}$. The y_i 's are defined in some way that enables us to upper bound the sum $\sum_{i=1}^Q y_i^{2t+3}$, that is, to bound the head sum as required.

In the following we define the y_i 's. $i = 1, \dots, (n+1)^t$:

For $1 \leq i \leq s = \frac{1}{(1-2\epsilon)^2}$: $y_i \stackrel{\text{def}}{=} (1-2\epsilon)n$.

Note that the shortest words in $C_{dBCH(t)} + v_{\epsilon n}$ are of weight ϵn . Using Claims 2.3.1 and 2.3.4, the number of words of weight ϵn in $C_{dBCH(t)} + v_{\epsilon n}$ is at most $\frac{dn}{2(\epsilon n)^2 - 2n(\epsilon n) + dn} \leq \frac{1}{(1-2\epsilon)^2}(1+o(1))$. Hence we get that for $1 \leq i \leq s = \frac{1}{(1-2\epsilon)^2}$, $x_i \leq y_i = (1-2\epsilon)n$.

For $\frac{1}{(1-2\epsilon)^2} \leq i \leq s = n^{\frac{1}{2} - \frac{1}{100}}$ such that $s \leq Q$: $y_i \stackrel{\text{def}}{=} n(\sqrt{i} - \sqrt{i-1})$. Using Lemma 10.4.1, we get that in this region $x_i \leq y_i \leq \frac{n}{\sqrt{i}}$.

For $n^{\frac{1}{2} - \frac{1}{100}} < i \leq n^{\frac{1}{2} + \frac{1}{100}}$: $y_i \stackrel{\text{def}}{=} 2n^{\frac{3}{4} + \frac{1}{100}}$. Using Lemma 10.4.1, we get that in this region $x_i \leq y_i \leq 2n^{\frac{3}{4} + \frac{1}{100}}$.

For $n^{\frac{1}{2} + \frac{1}{100}} < i \leq Q$: $y_i \stackrel{\text{def}}{=} 2n^{\frac{3}{4}}$. Using Lemma 10.4.1, we get that in this region $x_i \leq y_i \leq 2n^{\frac{3}{4}}$.

Hence by the definition of y_i : For $i = 1, \dots, Q$ we get that $\sum_{i=1}^Q x_i \leq \sum_{i=1}^Q y_i$. Let W be the biggest number such that $\sum_{i=1}^W y_i \leq \sum_{i=1}^Q x_i$. For $1 \leq i \leq W$ the value of y_i remains as it was defined. We next assign new values to y_{W+1}, \dots, y_Q .

$$y_{W+1} = \sum_{i=1}^Q x_i - \sum_{i=1}^W y_i, y_{W+2} = y_{W+3} = \dots = y_Q = 0.$$

Thus we have

$$\sum_{i=1}^Q x_i = \sum_{i=1}^Q y_i.$$

In the following we shall show that the **Head Bound** holds: That is:

$$\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} (n-2i)^{2t+3} \leq (1-3\epsilon)n^{2t+3} + O(n^{2t+2}).$$

Consider the definition of x_i 's and y_i 's for $i = 1, \dots, Q$. Note that for every $s < Q$, $\sum_{i=1}^s x_i \leq \sum_{i=1}^s y_i$ and that $\sum_{i=1}^Q x_i = \sum_{i=1}^Q y_i$. Hence, by the Karamata Inequality of Lemma 10.4.3 we can conclude that

$$\sum_{i=1}^Q x_i^{2t+3} \leq \sum_{i=1}^Q y_i^{2t+3}.$$

We use this last inequality in the following equation.

$$\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)}+v_{\epsilon n}} (n-2i)^{2t+3} = \sum_{i=1}^Q x_i^{2t+3} \leq \sum_{i=1}^Q y_i^{2t+3} \quad (10.13)$$

Hence,

$$\begin{aligned} \sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)}+v_{\epsilon n}} (n-2i)^{2t+3} &\leq \sum_{i=1}^Q y_i^{2t+3} \leq \frac{n^{2t+3}(1-2\epsilon)^{2t+3}}{(1-2\epsilon)^2} + \sum_{i=\frac{1}{(1-2\epsilon)^2}}^{\sqrt{n}} \frac{n^{2t+3}}{\sqrt{i}^{2t+3}} + \sum_{i=\sqrt{n}}^Q (2n^{\frac{3}{4}})^{2t+3} + n^{\frac{6t+12}{4}} \leq \\ n^{2t+3}(1-2\epsilon)^{2t+1} + n^{2t+3} \int_{i=\frac{1}{(1-2\epsilon)^2}}^{\infty} \frac{dx}{x^{\frac{2t+3}{2}}} + 4^{2t+3} n^{1-\frac{1}{2(2t+2)}} n^{\frac{3(2t+3)}{4}} &= n^{2t+3} \left(\frac{2t+3}{2t+1} (1-2\epsilon)^{2t+1} \right) + 4^{2t+3} n^{\frac{6t+13}{4} - \frac{1}{2(2t+2)}} \end{aligned}$$

Note that for constant t and $\frac{1}{4} \leq \epsilon \leq \frac{1}{2}$, the last term is bounded by $(1-3\epsilon)n^{2t+3} + O(n^{2t+2})$. Hence, **Head Bound** holds.

In the following we prove that the **Tail Bound** holds: That is:

$$(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2} \epsilon (n^{2t+3} + O(n^{2t+2})).$$

Recall that $\frac{1}{4} \leq \epsilon \leq \frac{1}{2}$, $a = n^{\frac{1}{2} + \frac{1}{4(2t+3)}}$, and t is constant.

Thus, $(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2} \epsilon (n^{2t+3} + O(n^{2t+2}))$, and the **Tail Bound** holds.

10.4.4 Proof of Case 2 in Theorem 10.4.1

Let $a = n^{\frac{1}{2} + \frac{1}{4(2t+3)}}$, for $t > 1$. Consider the $C_{dBCH(t)} + v_{\epsilon n}$. Let c_i for $i = 1, \dots, (n+1)^t$ be the codewords of the $C_{dBCH(t)} + v_{\epsilon n}$ code. Let $x_i = n - 2w(c_i)$ and assume that $x_1 \geq x_2 \geq \dots \geq x_{(n+1)^t}$. Recall that we need to show that the following two bounds apply for the case that $\frac{1}{n} \leq \epsilon \leq \frac{1}{4}$:

Head Bound: $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)}+v_{\epsilon n}} (n-2i)^{2t+3} \leq (1-3\epsilon)n^{2t+3} + O(n^{2t+2})$.

Tail Bound : $(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2} (\epsilon n^{2t+3} + O(n^{2t+2}))$.

Proof of Case 2, $\frac{1}{n} \leq \epsilon \leq \frac{1}{4}$: Here we follow the same line of proof as in the first case. The only difference is the following. For $\frac{1}{n} \leq \epsilon \leq \frac{1}{4}$, there can be only one word c_i in $C_{dBCH(t)} + v_{\epsilon n}$ with weight ϵn . The rest of the codewords in $C_{dBCH(t)} + v_{\epsilon n}$ are of weight at least $(\frac{1}{2} - \epsilon)n$. By using Claims 2.3.1 and 2.3.4, the number of words of weight $(\frac{1}{2} - \epsilon)n$ in this code is at most

$$\frac{dn}{2(\frac{1}{2} - \epsilon)^2 - 2n(\frac{1}{2} - \epsilon) + dn} \leq \frac{1}{(2\epsilon)^2} (1 + o(1)).$$

Hence we define y_i 's. $i = 1, \dots, (n+1)^t$ in the following way:

For $i = 1$: $y_1 \stackrel{\text{def}}{=} (1-2\epsilon)n$, hence $x_1 \leq y_1$.

For $2 \leq i \leq s = \frac{1}{(2\epsilon)^2}$: $y_i \stackrel{\text{def}}{=} 2\epsilon n$, hence in this region, $x_i \leq y_i = (2\epsilon)n$.

For $\frac{1}{(2\epsilon)^2} \leq i \leq s = n^{\frac{1}{2} - \frac{1}{100}}$: $y_i \stackrel{\text{def}}{=} n(\sqrt{i} - \sqrt{i-1})$. Using Lemma 10.4.1, we get that in this region $x_i \leq y_i \leq \frac{n}{\sqrt{i}}$.

For $n^{\frac{1}{2} - \frac{1}{100}} < i \leq n^{\frac{1}{2} + \frac{1}{100}}$: $y_i \stackrel{\text{def}}{=} 2n^{\frac{3}{4} + \frac{1}{100}}$. Using Lemma 10.4.1, we get that in this region $x_i \leq y_i \leq 2n^{\frac{3}{4} + \frac{1}{100}}$.

For $n^{\frac{1}{2} + \frac{1}{100}} < i \leq Q$: $y_i \stackrel{\text{def}}{=} 2n^{\frac{3}{4}}$. Using Lemma 10.4.1, we get that in this region $x_i \leq y_i \leq 2n^{\frac{3}{4}}$.

We then update the y_i 's similarly as before, and get that for every $s < Q$,

$$\sum_{i=1}^s x_i \leq \sum_{i=1}^s y_i.$$

and that

$$\sum_{i=1}^Q x_i = \sum_{i=1}^Q y_i$$

.

Hence, by the Karamata Inequality of Lemma 10.4.3 we can conclude that

$$\sum_{i=1}^Q x_i^{2t+3} \leq \sum_{i=1}^Q y_i^{2t+3}.$$

We use this last inequality in the following equation.

$$\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} (n-2i)^{2t+3} = \sum_{i=1}^Q x_i^{2t+3} \leq \sum_{i=1}^Q y_i^{2t+3} \quad (10.14)$$

Hence, $\sum_{i=\epsilon n}^{\frac{n}{2}-a} B_i^{C_{dBCH(t)} + v_{\epsilon n}} (n-2i)^{2t+3}$ is at most

$$\begin{aligned} \sum_{i=1}^Q y_i^{2t+3} &\leq n^{2t+3}(1-2\epsilon)^{2t+3} + \frac{n^{2t+3}(2\epsilon)^{2t+3}}{(2\epsilon)^2} + \sum_{i=\frac{1}{(2\epsilon)^2}}^{\sqrt{n}} \frac{n^{2t+3}}{\sqrt{i}^{2t+3}} + \sum_{i=\sqrt{n}}^Q (2n^{\frac{3}{4}})^{2t+3} + n^{\frac{6t+12}{4}} \\ &\leq n^{2t+3}(1-2\epsilon)^{2t+3} + n^{2t+3}(2\epsilon)^{2t+1} + n^{2t+3} \int_{i=\frac{1}{(2\epsilon)^2}}^{\infty} \frac{dx}{x^{\frac{2t+3}{2}}} + 4^{2t+3} n^{1-\frac{1}{2(2t+3)}} n^{\frac{3(2t+3)}{4}} \\ &= n^{2t+3}(1-2\epsilon)^{2t+3} + n^{2t+3} \left(\frac{2t+3}{2t+1} (2\epsilon)^{2t+1} \right) + 4^{2t+3} n^{\frac{6t+13}{4} - \frac{1}{2(2t+3)}} \end{aligned} \quad (10.15)$$

Note that for constant t and $\frac{1}{n} \leq \epsilon \leq \frac{1}{4}$, the last term is at most $(1-3\epsilon)n^{2t+3} + O(n^{2t+2})$, and hence the **Head Bound** follows.

In the following we prove that the **Tail Bound** holds: That is:

$$(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2} (\epsilon n^{2t+3} + O(n^{2t+2})).$$

Recall that $\frac{1}{n} \leq \epsilon \leq \frac{1}{4}$, $a = n^{\frac{1}{2} + \frac{1}{4(2t+3)}}$, and t is constant.

Thus, $(2a)^{2t+3} \cdot (n+1)^t \leq \frac{1}{2}(\epsilon n^{2t+3} + O(n^{2t+2}))$, and the **Tail Bound** holds. ■

As an immediate corollary from Theorem 10.4.1 and Theorem 10.2.2 we get:

Theorem 10.4.2 *Consider a constant $t \geq 1$. The $C_{BCH(t)}$ code is spanned by its words of weight $2t + 3$. (where its shortest words are of weight $2t + 1$).*

10.4.5 Local Testability of Almost Orthogonal Codes

The proof of Theorem 10.4.1 used only two facts about $C_{dBCH(t)}$: the first is that its distance is at least $\frac{n}{2} - (t-1)\sqrt{n}$ and the second is that the number of its codewords is $O(n^t)$. Note that by a known bound from [56], a code C with minimum distance $\frac{n}{2} - \sqrt{tn}$, contains at most $O(n^t)$ codewords. Hence, we could repeat the previous proof for general almost orthogonal codes. We then need to use Krawtchouk polynomial P_{t+2} instead of P_{2t+3} that was used for $C_{dBCH(t)}$. All the rest works the same and the proof of Theorem 10.1.1 is obtained.

10.4.6 The $C_{BCH(t)}$ is Spanned by its Codewords of Weight at most $2t + 2$

Extended $C_{BCH(t)}$ code, $C_{eBCH(t)}$, obtained by appending all-coordinates parity-check to each codeword of $C_{BCH(t)}$. The code $C_{edBCH(t)}$ is doubly transitive. The shortest words of $C_{eBCH(t)}$ are of weight $2t + 2$. Using the techniques presented in the the proof of Theorem 10.4.1 and using the fact that the Krawtchouk polynomial $P_{2t+2}(i)$ is symmetric around $\frac{n}{2}$ we obtain the following theorem:

Theorem 10.4.3 *The $C_{edBCH(t)}$ is locally testable using $O(\frac{t}{\epsilon})$ queries. Moreover there exist a regular-tester for the code that uses local tests of weight $2t + 2$, such that $Rej(\epsilon) \geq \epsilon$.*

Hence by the definition of the extended code, if $C_{eBCH(t)}$ is spanned by codewords of weight exactly $2t + 2$ then $C_{BCH(t)}$ is spanned by codewords of weight at most $2t + 2$. This leads to the following theorem.

Theorem 10.4.4 *Consider a constant $t \geq 1$. $C_{BCH(t)}$ is spanned by its words of weight at most $2t + 2$.*

10.5 Efficient Construction of a Random Short Codeword of $C_{BCH(t)}$

In the BLR test (i.e. Hadamard test) [24], there is a procedure that uses $O(\log n)$ bits operation to find a random 3 length word from the $C_{BCH(t)}$ code where $t = 1$. This procedure, in the case of BLR, does the following. It selects uniformly at random $x, y \in \{0, 1\}^{\log n}$. Then, it computes $x + y$ using $O(\log n)$ bits operations. In [24] it is shown that the weight 3 codeword that has 1's in coordinates $x, y, x + y$ is a random codeword of weight 3 selected uniformly from the dual code (dual to the Hadamard code). Since the BLR test works in $O(\frac{1}{\epsilon})$ rounds, it needs overall $O((\log n)/\epsilon)$ bits operations.

In the following we show that there exists a procedure that uses $O(\text{poly}(t \log n))$ bits operations for selecting a random word of length $2t + 3$ from $C_{BCH(t)}$. Hence, we show that the $C_{dBCH(t)}$ -tester can be implemented using $O(\text{poly}(t \log n)/\epsilon)$ bits operations.

Our procedure could be used for finding any random fixed weight codeword efficiently.

Theorem 10.5.1 *There exists a random procedure that uses $O(\text{poly}(t \log n))$ bits operations and finds a random word of length $2t + 3$ from $C_{BCH(t)}$ with a constant probability.*

Proof: The proof of the theorem follows from the following claim.

Claim 10.5.1 Consider $t + 3$ vectors $x_1, \dots, x_{t+3} \in \{0, 1\}^{\log n}$. There exists a procedure *Fast-BCH-decoding* that receives these vectors and does the following. Consider an n length binary vector u , such that u has 1's in the locations determined by x_1, \dots, x_{t+3} , and all other coordinates are zero. If there exists a codeword $c \in C_{BCH(t)}$ at distance t from u , then the non-zero coordinates of c can be found using $O(\text{poly}(t \log n))$ bits operations. *Fast-BCH-decoding*(x_1, \dots, x_{t+3}) finds the $2t + 3$ non-zero coordinates of such c if exists.

Proof: Note that the distance of $C_{BCH(t)}$ is $2t + 1$, hence, a decoding algorithm for $C_{BCH(t)}$, if succeeds, returns a codeword of weight at most $2t + 3$. A decoding algorithm for $C_{BCH(t)}$ has three stages (for details see [58]): First is *Syndrome calculation*. Second is *Key-equation solution*. Third is *Finding the roots of the locator polynomial*. We note here that when working in a field of size n every operation in the field can be performed using $O(\log n)$ bits operations. Consider a length n binary vector u , such that u has 1's in the locations determined by x_1, \dots, x_{t+3} , and all other coordinates of u are zeros. Next we show that for such u , the decoding algorithm can be performed using $O(\text{poly}(t \log n))$ bits operations. Since u is of weight $t + 3$, Syndrome calculation involves $O(t^2)$ operations in the field. Key-equation solution using the Berlekamp Massey Algorithm [22, 61] requires $(t^2 \log t)$ operations in the field. Finding the roots can be implemented using fast factorization since the polynomial at hand is of degree at most $2t + 3$ (where the degree is independent of the size of the field). Fast factorization as described in [28] can be implemented using $\text{poly}(t \log n)$ operations in the field. Hence, overall the $2t + 3$ non zero coordinates of c (c as described above) can be found using $O(\text{poly}(t \log n))$ bits operations. ■

Based on the *Fast-BCH-decoding* (x_1, \dots, x_{t+3}) we describe a randomized procedure *Find-Random-Codeword*, that returns the non zero locations of a codeword of length $2t + 3$ uniformly selected from the $C_{BCH(t)}$ code. The procedure described succeeds to find such a word with a constant probability.

Find-Random-Codeword

1. Repeat $\Theta(t!)$ times:
 - (a) Uniformly and independently select $t + 3$ vectors $x_1, \dots, x_{t+3} \in \{0, 1\}^{\log n}$.
 - (b) If *Fast-BCH-decoding* (x_1, \dots, x_{t+3}) returns $2t + 3$ vectors exit and return these vectors.
2. If this point is reached output fail.

The correctness of the theorem stems from the following set of claims.

Claim 10.5.2 The $2t + 3$ codewords of $C_{BCH(t)}$ that are obtained by the *Find-Random-Codeword* algorithm are uniformly distributed in the set of $2t + 3$ length codewords of $C_{BCH(t)}$.

Proof: Note that if we consider some $t + 3$ coordinates of an n length vector, then there can be only one $C_{BCH(t)}$ codeword of length $2t + 3$ that has 1's in all these coordinates. The reason for that is that if there were two such words then by the linearity of $C_{BCH(t)}$ we get that the $C_{BCH(t)}$ contains a word of weight $2t$ which is a contradiction. Hence, every $t + 3$ coordinates may belong to at most one weight $2t + 3$ codeword. Thus, the probability to find each of the weight $2t + 3$ codewords is identical. ■

Claim 10.5.3 The probability that a single round is successful is at least $\frac{1+o(1)}{t!}$.

Proof: Note that by the definition of the procedure, the decoding succeeds only if it finds a codeword of length $2t + 3$ that has 1's in the given $t + 3$ coordinates. Note that by Claim 2.3.5, the number of words of length $2t + 3$ in $C_{BCH}(t)$ is $\frac{\binom{n}{2t+3}}{(n+1)^t} (1 + o(1))$, while the number of sets of $t + 3$ coordinates is $\binom{n}{t+3}$. Hence, the probability that the decoding procedure succeeds is

$$\frac{\frac{\binom{n}{2t+3}}{(n+1)^t} (1 + o(1)) \cdot \binom{2t+3}{t+3}}{\binom{n}{t+3}} = \frac{1 + o(1)}{t!}$$

■

Claim 10.5.4 *The number of steps performed in a single round of **Find-Random-Codeword** algorithm is $\text{poly}(t \log n)$*

Proof: Clearly follows from 10.5.1. ■ ■

Part IV

Open Problems

Chapter 11

Further Research

It would be interesting to further understand the *testability nature of graph properties*, when the underlying graphs are not promised to have certain pre-defined densities. The goal is to understand the specific impact that the density of the graph has on the ability to test it for various properties. In this work we considered testing of several graph properties in the general case. Some of our results are not tight and there exists a gap between the known upper and lower bounds. Closing those gaps may shed some light on testing general graph properties.

There are several known techniques for *proving lower bounds in property testing*. Among them are Yao's *principle*, methods based on *additive number theory*, constructions of *gap-preserving local reductions* and constructions of *objects with local view different than their global view* (e.g. a graph that is not k -colorable but each of its subgraphs of linear size is k -colorable). It seems that lower bounds for testing graphs with varying densities require some new lower bounds techniques.

There are well studied connections between *graph testing and sub-linear approximation algorithms for graphs* for the case of dense graphs. Specifically, given an NP-hard problem, suppose that instead of approximating it using polynomial algorithms (such as Semi-Definite Programming), we are willing to "invest" only sub-linear queries. Could we get reasonable approximation in that case? Some positive results are known for the case that the graphs are dense. It would be interesting to understand similar connections for graphs with general densities.

The fact that the Hadamard code is locally testable played a major role in the proof of the PCP theorem. Since the dual-BCH code is a generalization of the Hadamard code, it could be the case the local testability of the dual-BCH code may have some broader implications. A more general question in codes testing is the following. What is the *characterization of linear codes that can be tested* using query complexity which is independent of the code length?

Bibliography

- [1] N. Alon. Explicit ramsey graphs and orthonormal labelings. *The Electronic J. Combinatorics*, 1(12), 1994.
- [2] N. Alon. Testing subgraphs of large graphs. *Random Structures and Algorithms*, 21:359–370, 2002.
- [3] N. Alon, E. Fischer, M. Krivelevich, and M Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.
- [4] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing low-degree polynomials over $GF(2)$. In *Proceedings of the Seventh Annual Workshop on Randomization and Approximation Techniques in Computer Sciences (RANDOM)*, pages 188–199, 2003.
- [5] N. Alon, T. Kaufman, M. Krivelevich, and D. Ron. Testing triangle-freeness in general graphs. In *Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 279–288, 2006.
- [6] N. Alon and M. Krivelevich. Testing k -colorability. *SIAM Journal on Discrete Math*, 15(2):211–227, 2002.
- [7] N. Alon and A. Orlitsky. Repeated communication and ramsey graphs. *IEEE Transactions on Information Theory*, 41:1276–1289, 1995.
- [8] N. Alon and Y. Roichman. Random Cayley graphs and expanders. *Random Structures and Algorithms*, 5:271–284, 1994.
- [9] N. Alon and A. Shapira. A characterization of easily testable induced subgraphs. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.
- [10] N. Alon and A. Shapira. Testing subgraphs in directed graphs. *Journal of Computer and System Sciences*, 69:354–482, 2004.
- [11] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, Inc., 1992.
- [12] S. Arora and M. Sudan. Improved low-degree testing and its applications. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 485–495, 1997.
- [13] E. F. Assmus, Jr. and J. D. Key. *Handbook of Coding theory: Polynomial codes and finite geometries*. Elsevier, 1998. In V.S. Pless and W. C. Huffman, editors, Volume 2, Part 2, Chapter 16.
- [14] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

- [15] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.
- [16] F. A. Behrend. On sets of integers which contain no three terms in arithmetic progression. *Proc. National Academy of Sciences USA*, 32:331–332, 1946.
- [17] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing over characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, 1996.
- [18] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 294–304, 1993.
- [19] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 184–193, 1994.
- [20] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 1–10, 2004.
- [21] E. Ben-Sasson, P. Harsha, and S. Raskhodnikova. 3CNF properties are hard to test. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 345–354, 2003.
- [22] E. R. Berlekamp. *Ch. 7 in Algorithmic Coding Theory*. New York: McGraw-Hill, 1968.
- [23] E. R. Berlekamp. Goppa codes. *IEEE Transaction on Information Theory*, 19:590–592, 1973.
- [24] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993.
- [25] A. Bogdanov, Kenji Obata, and L. Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pages 93–102, 2002.
- [26] A. Bogdanov and L. Trevisan. Lower bounds for testing bipartiteness in dense graphs. Technical Report TR02-064, Electronic Colloquium on Computational Complexity (ECCC), 2002. Available from <http://www.eccc.uni-trier.de/eccc/>.
- [27] B. Bollobas, B. Erdos, M. Simonovitz, and E. Szemerédi. Extremal graphs without large forbidden subgraphs. *Annals of Discrete Mathematics*, 3:29–41, 1978.
- [28] D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36:587–592, 1981.
- [29] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering Codes*. North Holland, 1997.
- [30] P. Delsarte, J.M. Goethals, and F.J. MacWilliams. On generalized Reed-Muller codes and their relatives. *Information and Control*, 16:760–769, 1970.
- [31] P. Ding and J.D. Key. Minimum-weight codewords as generators of generalized Reed-Muller codes. *IEEE Transactions on Information Theory*, 46(6):2152–2157, 2000.
- [32] U. Feige. Sampling vertex degrees and estimating network load parameters. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, 2004. to appear.

- [33] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Journal of the Association for Computing Machinery*, pages 268–292, 1996.
- [34] K. Friedl and M. Sudan. Some improvements to total degree tests. In *Proceedings of the 3rd Annual Israel Symposium on Theory of Computing and Systems*, pages 190–198, 1995. Corrected version available online at <http://theory.lcs.mit.edu/~madhu/papers/friedl.ps>.
- [35] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 32–42, 1991.
- [36] O. Goldreich. Short locally testable codes and proofs (survey). Available from: <http://www.wisdom.weizmann.ac.il/~oded>, 2005.
- [37] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the Association for Computing Machinery*, 45(4):653–750, 1998.
- [38] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.
- [39] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.
- [40] O. Goldreich and D. Ron. On estimating the average degree of a graph. Technical Report TR04-013, Electronic Colloquium on Computational Complexity (ECCC), 2004. Available from <http://www.eccc.uni-trier.de/eccc/>.
- [41] O. Goldreich and M. Sudan. Locally testable codes and PCPs of almost-linear length. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pages 13–22, 2002.
- [42] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23(1):23–57, 2003.
- [43] B. Green. Counting sets with small sumset, and the clique number of random Cayley graphs. Can be downloaded from: <http://www.maths.bris.ac.uk/~mabjg/preprints.html>.
- [44] S. Janson, T. Łuczak, and A. Ruciński. *Random graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000.
- [45] C. S. Jutla, A. C. Patthak, A. Rudra, and D. Zuckerman. Testing low-degree polynomials over prime fields. In *Proceedings of the Forty-Fifth Annual Symposium on Foundations of Computer Science*, 2004.
- [46] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.
- [47] T. Kaufman, M. Krivelevich, and D. Ron. Testing k -colorability in general graphs. Manuscript, 2005.
- [48] T. Kaufman and S. Litsyn. Almost orthogonal linear codes are locally testable. In *Proceedings of the Forty-sixth Annual Symposium on Foundations of Computer Science*, pages 317–326, 2005.
- [49] T. Kaufman and S. Litsyn. Long extended BCH codes are spanned by minimum weight words. In *Proceedings of the 16th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC)*, pages 285–294, 2006.

- [50] T. Kaufman and D. Ron. Testing polynomials over general fields. In *Proceedings of the Forty-Fifth Annual Symposium on Foundations of Computer Science*, pages 413–422, 2004.
- [51] T. Kaufman and D. Ron. A characterization of low-weight words that span Generalized Reed Muller codes. *IEEE Transaction on Information Theory*, 51(11):4039–4043, 2005.
- [52] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *Proceedings of the Forty-Fifth Annual Symposium on Foundations of Computer Science*, pages 126–135, 2004.
- [53] M. Kiwi. Algebraic testing and weight distributions of codes. *Theoretical Computer Science*, 299:81–106, 2003.
- [54] I. Krasikov and S. Litsyn. On spectra of BCH codes. *IEEE Transaction on Information Theory*, 41(3):786–788, 1995.
- [55] I. Krasikov and S. Litsyn. Survey of binary krawtchouk polynomials. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 56, 2001.
- [56] V. I. Levenshtein. *Handbook of Coding Theory, Volume 1, Chapter 9*. North Holland, 1997.
- [57] A. Lubotzky, R. Phillips, and P. Sarnak. Explicit expanders and the Ramanujan conjectures. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 240–246, 1986.
- [58] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North Holland, 1977.
- [59] Gregory A. Margulis. Explicit constructions of expanders. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973.
- [60] A. W. Marshall. and I. Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, New York, 1979.
- [61] J. L. Massey. Shift-register synthesis and bch decoding. *IEEE Transaction on Information Theory*, 15:122–127, 1969.
- [62] M. Mihail. Conductance and convergence of Markov chains - A combinatorial treatment of expanders. In *Proceedings 30th Annual Conference on Foundations of Computer Science*, pages 526–531, 1989.
- [63] J. Hastad N. Alon, O. Goldreich and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3:289–304, 1992.
- [64] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.
- [65] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 194–203, 1994.
- [66] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.
- [67] R. Salem and D. C. Spencer. On sets of integers which contain no three terms in arithmetical progression. *Proc. National Academy of Sciences USA*, 28:561–563, 1942.
- [68] A. Shpilka and A. Wigderson. Derandomizing homomorphism testing in general groups. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 427–435, 2004.

- [69] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004.
- [70] A.C. Yao. Probabilistic computation, towards a unified measure of complexity. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

Appendix A

A formal definition of $M_{\ell_1}^{\ell_2}(H)$

For every vertex v in H we have a state v in $M_{\ell_1}^{\ell_2}(H)$. For simplicity, we shall continue referring to these states as vertices. Let the *border* of H , denoted $B(H)$, be the set of vertices in H that have at least one neighbor in G that is not in H . Then, for every vertex $v \in B(H)$, we have a set $a_{v,1}, \dots, a_{v,\ell_1}$ of *auxiliary* states. Let $p_{v,u}^H(t)$ denote the probability of a walk of length t that starts at v and ends at u without passing through any other vertex in H . Namely, it is the sum over all such walks w , of the product, taken over all steps in w , of the transition probabilities of these steps. In particular, $p_{v,v}^H(1) \geq \frac{1}{2}$ (where equality holds in case v has degree d), and for every $u \in \Gamma(v)$, $p_{v,u}^H(1) = \frac{1}{2d}$ (here we assume that we can choose a random neighbor of a vertex with in time which is $O(1)$). The transition probabilities, $q_{x,y}$, in $M_{\ell_1}^{\ell_2}(H)$ are defined as follows:

- For every v and u in H , $q_{v,u} = \sum_{t=1}^{\ell_2-1} p_{v,u}^H(t)$.

Thus, $q_{v,u}$ is a sum of $p_{v,u}^H(1)$ and $\sum_{t=2}^{\ell_2-1} p_{v,u}^H(t)$. The first term implies that for every v in H , $q_{v,v} \geq \frac{1}{2}$ and for every pair of neighbors v and u , $q_{v,u} \geq \frac{1}{2d}$. The second term, which we refer to as the *excess* probability is due to walks of length less than ℓ_2 (from v to u) passing through vertices outside of H , and can be viewed as *contraction* of these walks.

Hence, for every pair of vertices v and u , $q_{v,u} = q_{u,v}$.

- For every $v \in B(H)$, $q_{v,(a_{v,1})} = \sum_{u \in H} \sum_{t \geq \ell_2} p_{v,u}^H(t)$; for every ℓ , $1 \leq \ell < \ell_1$, $q_{(a_{v,\ell}), (a_{v,\ell+1})} = 1$; and for every $u \in H$, $q_{(a_{v,\ell_1}), u} = \frac{1}{q_{v,(a_{v,1})}} \cdot \sum_{t \geq \ell_2} p_{v,u}^H(t)$. (The parentheses added in the notation above (e.g., $q_{(a_{v,\ell}), (a_{v,\ell+1})}$) are only for sake of readability.)

In other words, $q_{v,(a_{v,1})}$ is the probability that a random walk in G that starts from v takes at least ℓ_2 steps outside of H before returning to H , and $q_{(a_{v,\ell_1}), u}$ is the conditional probability of reaching u in such a walk. Thus, the auxiliary states form auxiliary paths in $M_{\ell_1}^{\ell_2}(H)$, where these paths correspond to walks of length at least ℓ_2 outside of H .

We shall restrict our attention to walks of length at most ℓ_1 in $M_{\ell_1}^{\ell_2}(H)$, and hence any walk that starts at a vertex of H and enters an auxiliary path never returns to vertices of H .

For any two states y, z in $M_{\ell_1}^{\ell_2}(H)$ let $q_{y,z}(t)$ be the probability that a walk of length t starting from y ends at z . In particular $q_{y,z} \equiv q_{y,z}(1)$, and for any two vertices u and v and any integer t , we have $q_{u,v}(t) = q_{v,u}(t)$. We further let the parity of the lengths of paths corresponding to walks in G be carried on to $M_{\ell_1}^{\ell_2}(H)$. That is, each transition between vertices v and u that corresponds to walks outside of H consists of two transitions – one due to even-length paths corresponding to walks from v to u outside of H ,

and one to odd-length paths. For any two vertices in H we let $q_{v,u}^b(t)$ denote the probability in $M_{\ell_1}^{\ell_2}(H)$ of a walk of length t starting from v , ending at u , and corresponding to a path whose length has parity b .

In all that follows we assume that G is connected. Our analysis can easily be modified to deal with the case in which G is not connected, simply by treating separately each of its connected components. Under the assumption that G is connected, for every v and u in H , there exists a t such that $q_{u,v}(t) > 0$, and hence $M_{\ell_1}^{\ell_2}(H)$ is irreducible. Furthermore, because for each $v \in H$ $q_{v,v} \geq \frac{1}{2}$, $M_{\ell_1}^{\ell_2}(H)$ is also aperiodic. Thus it has a unique stationary distribution.

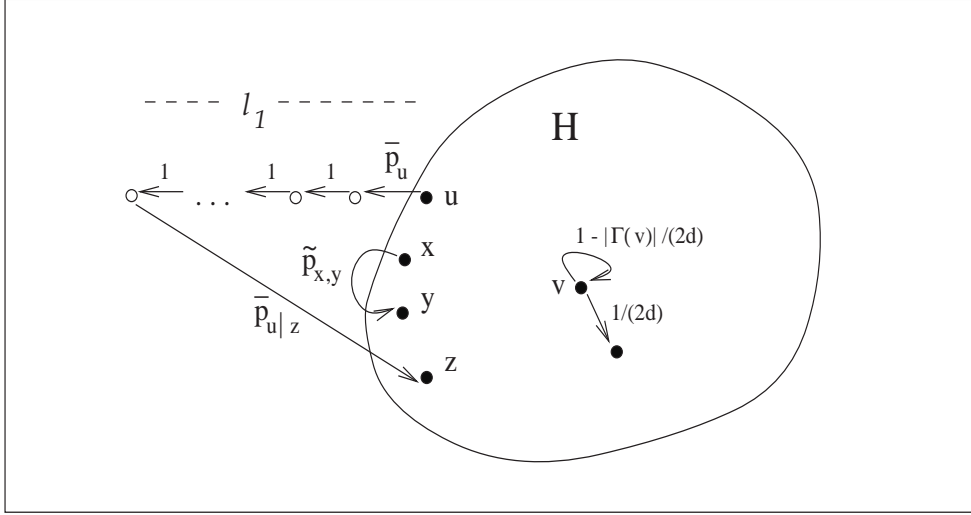


Figure A.1: The structure of $M_{\ell_1}^{\ell_2}(H)$. The states corresponding to vertices of H are depicted as black dots, and the auxiliary states as white ones. Here $\tilde{p}_{x,y}$ denotes the transition probability between any two vertices $x, y \in B(H)$, which equals $\sum_{t=1}^{\ell_2-1} p_{x,y}^H(t)$, \bar{p}_u denotes the probability of entering an auxiliary path starting from $u \in B(H)$, which equals $\sum_{z \in H} \sum_{t \geq \ell_2} p_{u,z}^H(t)$, and $\bar{p}_{u|z}$ denotes the probability of returning from the last state on this auxiliary path to $z \in B(H)$, which equals $\frac{1}{\bar{p}_u} \cdot \sum_{t \geq \ell_2} p_{u,z}^H(t)$.