Property Testing and Combinatorial Approximation

Arie Matsliah

Property Testing and Combinatorial Approximation

Research Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Arie Matsliah

Submitted to the Senate of the
Technion - Israel Institute of Technology

Tamuz 5768 Haifa July 2008

The research thesis was done under the supervision of
Assoc. Prof. Eldar Fischer
in the Department of Computer Science.

Acknowledgements

I am grateful to my supervisor - Eldar Fischer, for guiding me devotedly for the last five years, for his continued encouragement and for many invaluable suggestions in this work. I would also like to include my gratitude to Eli Ben-Sasson, especially for his willingness to teach and assist in any given moment. Furthermore, I am deeply indebted to my collaborators in this work: Sourav Chakraborty, Prahladh Harsha, Oded Lachish, Ilan Newman, Asaf Shapira and Orly Yahalom.

The generous financial help of the Technion is gratefully acknowledged

Contents

List of Figures

Abstract

In this thesis we study property testers and their applications in the dense graph and hypergraph model, property testers for massively parameterized properties, and probabilistically checkable proofs of proximity – PCPPs.

First we consider the task of testing for graph isomorphism in the dense graph model. We investigate both one-sided error and two-sided error testers under two possible settings. The first is where both graphs need to be queried, and the second is where one of the graphs is given in advance. We prove nearly tight lower and upper bounds on the query complexity of isomorphism testers under each of the four possible combinations.

Then we show that any general partition-problem of dense hypergraphs has a sub-linear time ($O(n)$ where the input size is $\Omega(n^r)$ for some $r > 1$) approximate partitioning algorithm and a property tester with constant query complexity. This extends the results of Goldreich, Goldwasser and Ron who obtained similar algorithms for graph partition problems in their seminal paper [GGR98]. We use the partitioning algorithm to obtain a surprisingly simple sub-linear time algorithmic version of Szemerédi's regularity lemma, and for any $r \geq 3$, we also obtain an $O(n)$ time (where the input size is $\Omega(n^r)$) randomized algorithm for constructing weakly regular partitions of r-uniform hypergraphs, thus improving upon the previous $O(n^{2r-1})$ time deterministic algorithms. In addition, we use the hypergraph partition testing algorithm to unify many previous results in hypergraph property testing and CNF testing.

In massively parameterized property testing we concentrate on the orientation model. Our first result in this model solves an open question from [HLNT05], asking whether it is possible to test (with constant number of queries) that an orientation $\overrightarrow{G}$ is st-connected. We show that the property of being st-connected is testable by a one-sided error algorithm with a number of queries depending only on ε. That is, we construct a randomized algorithm such that for any underlying graph G, on input of an unknown orientation the algorithm queries only $O(1)$ edges for their orientation, and based on this distinguishes with success probability $\frac{2}{3}$ between the case that the orientation is st-connected and the case that it is ε-far from being st-connected.

Our second result within the topic of massively parameterized properties deals with the task of testing whether an orientation of a given graph is Eulerian. Despite the local nature of this property, it turns out to be significantly harder for testing than st-connectivity. In particular, we show a super-constant lower bound on the query complexity, even if 2-sided error is allowed and the underlying graph is a toroidal grid (meaning that there are very small negative witnesses).

In the last part of the thesis we study length-soundness tradeoffs in probabilistically checkable proofs of proximity (PCPPs). We show that any verifier obtaining the "best possible" soundness must query an exponentially long proof. One of the central ingredients in our proof is a tight connection between the query complexity of linear codes (in the property testing sense), and their optimal proof-length (in the PCPP sense).

Chapter 1

General introduction

With the recent advances in technology, we are faced with the need to process increasingly larger amounts of data increasingly faster. Originally, a problem was considered computable if there was an algorithm that could decide it in finite time given any input instance. Later came the notion of polynomial time computation, and then the possibility of making computations faster through use of parallel machines. However, the algorithms involved still face the obvious obstacle of reading the entire input prior to its assessment. There are practical situations in which the input is so large (and not accessible in parallel), so that even taking a linear time in its size to provide an answer is too much. In other situations the input is not easily accessible, or does not have an explicit representation, and an "oracle" procedure that computes its values in specific locations is provided instead.

The main line of research in this thesis revolves around designing and analyzing extremely efficient approximation algorithms, and in particular algorithms that make a decision concerning the input after reading only a small portion of it. In particular, the questions addressed in this thesis are related to combinatorial property testing in the dense-graphs model, testing massively parameterized properties, sub-linear time algorithms for dense graphs and hypergraphs, and probabilistically checkable proofs of proximity (PCPPs).

In this chapter we provide a brief introduction to the topics that are related to the content of this thesis. A more focused introduction and bibliographic background, based on the articles in which these results were published, are provided later in the specific parts.

1.1 Combinatorial property testing

The meta problem in the area of property testing is the following: Given a combinatorial structure S, distinguish between the case that S satisfies some property P and the case that S is *ε-far* from satisfying P. Roughly speaking, a combinatorial structure is said to be ε-far from satisfying some property P if an ε-fraction of its representation has to be modified in order to make S satisfy P. The main goal in property testing is to design randomized algorithms, which look at a very small portion of the input, and use this information to distinguish with high probability between the above two cases. Such algorithms are called *property testers* or simply *testers* for the property P.

Blum, Luby and Rubinfeld [BLR90] were the first to formulate a question of this type, and the general notion of property testing was first formulated by Rubinfeld and Sudan [RS93], who were interested in studying various algebraic properties such as the linearity of functions. The definitions and the first study of property testing as a framework were introduced in the seminal paper of Goldreich, Goldwasser and Ron [GGR98]. Since then, an extensive amount of work has been done on various aspects of property testing as well as on studying particular properties. For comprehensive surveys see [Ron01, Fis04].

1.1.1 Property testing in the dense graph (and hypergraph) model

The "dense" graph testing model that was defined in [GGR98] is also one of the well studied models in combinatorial property testing. Formally, the inputs in this model are functions $g : \{1, 2, \ldots, \binom{n}{2}\} \to \{0, 1\}$, which represent the edge set of a graph G with n vertices. The *distance* of a graph from a property P is measured by the minimum number of bits that have to be modified in the input in order to make it satisfy P, divided by the input length m, which in our case is taken to be $\binom{n}{2}$.

For the question of testing graphs with a constant number of queries in the dense model there are many recent advances, such as [AS05b], [FN05], [AS05] and [AFNS06]. All of these results use Szemerédi's regularity lemma [Sze78] as the main technical tool (see a brief introduction to the regularity lemma in Section 5.4.1).

For some of the properties that we consider here the number of required queries is of the form n^α for some $\alpha > 0$, and in these cases our interest will be to find bounds as tight as possible on α.

The dense graph model naturally generalizes to hypergraphs. For example, if properties of r-uniform hypergraphs are considered, then the inputs are functions $h : \{1, 2, \ldots, \binom{n}{r}\} \to \{0, 1\}$, representing the edge set of a hypergraph. Similarly to the case of graphs, the distance of a hypergraph from a property P is measured by the minimum number of bits that have to be modified in the input in order to make it satisfy P, divided by the input length.

1.1.2 Testing graph properties in other models

Other models in which graph properties were studied are the *bounded-degree* graph model, in which a sparse representation of sparse graphs is considered (instead of the adjacency matrix as in the dense model), and the *general density* model (also called the *mixed* model) which is a combination of the dense and the sparse (bounded degree) models. We will not deal with those models in the thesis. For further information on the sparse and the mixed models the reader is referred to [GR02], [PR02] and [KKR04].

1.1.3 Testing massively parameterized properties

The study of massively parameterized properties explicitly started in [HLNT05], where property testing of graphs in the orientation model was considered first [1]. This is a model that combines information that has to be queried with information that is known in advance, and so does not readily yield to general techniques such as that of the regularity lemma used in [AFNS06] and [AS05].

Specifically, the information given in advance is an underlying undirected graph $G = (V, E)$ (that may have parallel edges). The input is then constrained to be an orientation of G, and the distances are measured relative to $|E|$ and not to any function of $|V|$. An orientation of G is simply an orientation of its edges. That is, for every edge $e = u, v$ in $E(G)$ an orientation of G specifies which of u and v is the source vertex of e, and which is the target vertex. Thus an orientation defines a directed graph $\overrightarrow{G}$ whose undirected skeleton is G. Given the undirected graph G, a property of orientations is just a partial set of all orientations of G.

In the framework of property testing, the relevant combinatorial structure is an orientation $\overrightarrow{G}$ of the underlying graph G, and the distance between two orientations $\overrightarrow{G}_1$, $\overrightarrow{G}_2$ is the number of edges that are oriented differently in $\overrightarrow{G}_1$ and $\overrightarrow{G}_2$. Thus an orientation $\overrightarrow{G}$ is *ε-far* from a given property P if at least $\varepsilon|E(G)|$ edges have to be redirected in $\overrightarrow{G}$ to make it satisfy P. Ideally the number of queries that the tester makes depends only on ε and on nothing else (in particular it depends neither on $|E|$ nor the specific undirected graph G itself).

To put this model in context with previously discussed models, we note that in the dense, sparse and mixed models any property that has $o(n)$ witness size, and in particular the property of st-connectivity (either directed or undirected) is trivially testable as every input is close to the property.

Apart from [HLNT05], another related work to this model is that of [HLNT07] in which a graph $G = (V, E)$ is given and the properties are properties of boolean functions

[1]One can put the earlier works of Newman [New02] and Ben-Sasson et. al. [BSHR05] in the framework of testing massively parameterized properties as well, where the parameters are the branching program and the 3-CNF formula, respectively.

$f : E(G) \to \{0,1\}$. In [HLNT07] the interpretation of such a function is as an assignment to certain formulae that are associated with the underlying graph G, and in particular can viewed as properties of orientations (although the results in [HLNT07] concentrate on properties that are somewhat more "local" than our "global" property of being *st*-connected).

A common feature of the works in this model, which distinguishes these results from results in many other areas of property testing and in particular those of the dense graph models, is that the algorithms themselves are rather non-trivial in construction, and not just in their analysis.

1.2 Probabilistically checkable proofs of proximity - PCPPs

A *PCPP verifier* for a property $P \subset \{0,1\}^n$ is a randomized, sublinear-time algorithm that distinguishes with high probability between inputs that belong to P and inputs that are far in relative Hamming distance from all members of P. In this sense, PCPP verifiers are similar to property testers, as they perform the same task. However, in contrast to a tester the PCPP verifier may query an additional auxiliary proof, called a *proof of proximity.*

PCPPs were simultaneously introduced in [BSGH$^+$04] and (under the name *assignment testers*) in [DR04], and a similar notion also appeared earlier in [Sze99]. The interest in PCPPs stems first and foremost from the role they play within the proof of the celebrated PCP Theorem of [AS98, ALM$^+$98]. But PCPPs are also interesting beyond the scope of the PCP Theorem. They can be used to transform any error correcting code into a locally testable one and to construct "relaxed" locally decodable codes [BSGH$^+$04]. Additionally, as shown in [FF05, GR05], they have applications to questions in the theory of "tolerant" property testing that was introduced in [PRR06].

While the performance of a property tester is usually measured as the maximal number of queries it makes, PCPP verifiers have four parameters of interest — *proof length, query complexity, completeness* and a *soundness function.* The proof length is the length of the auxiliary proof that is queried by the verifier[2]. The query complexity is the maximal number of bits that can be read from *both* the input and the proof. The completeness parameter is the minimal probability with which inputs that belong to P are accepted when they are presented along with a "good" proof of proximity, and finally the soundness function $s(\delta)$ is the minimal rejection probability of inputs that are δ-far from P, where the minimum is taken over all such δ-far inputs and all possible proofs that may accompany them.[3] In this thesis we study the relations between the proof length and the soundness

[2]In PCP literature one often encounters *randomness complexity* as a means for bounding proof-length. The two parameters are closely related, i.e., proof-length $\approx 2^{\text{randomness}}$, and we stick to the former parameter.

[3]Often, in literature on PCPs, the term "soundness" refers to "soundness-error" which is defined to be

of PCPP verifiers that make 3 queries and have perfect completeness.

the *maximal* acceptance probability of a "bad" input. The connection between soundness (used here) and soundness-error, denoted s_{error}, is of course $s = 1 - s_{error}$.

Chapter 2

Summary of results

In this chapter we give an informal description of our main results. Sections 2.1 and 2.2 refer to the dense graphs model and related applications, Sections 2.3 and 2.4 are related to massively parameterized properties, and Section 2.5 summarizes our results related to PCPPs.

2.1 Testing graph isomorphism

In [AFKS00] it is proved that testing whether two graphs are isomorphic requires a non-constant number of queries, but the question of how many queries are indeed required has remained open. In Chapter 4 (based on [FM06]) we consider the task of distinguishing between the case that two graphs on n vertices are isomorphic, and the case that they are ε-far, that is they differ in more than $\varepsilon\binom{n}{2}$ pairs for all possible bijections of their vertices. During this work we investigated both one-sided error and two-sided error testers under two possible models. The first is where both graphs need to be queried, and the second is where one of the graphs is given in advance. We prove lower and upper bounds on the query complexity of isomorphism testing under each of the four possible settings, that are nearly tight for most of them.

2.2 Approximate hypergraph partitioning and applications

In Chapter 5 (based on [FMS07]) we show that any *partition-problem* of hypergraphs has an $O(n)$ time approximate partitioning algorithm and a property tester with constant query complexity. This extends the results of Goldreich, Goldwasser and Ron who obtained similar algorithms for graph partition problems in their seminal paper [GGR98].

The partitioning algorithm is used to obtain a surprisingly simple $O(n)$ time algorithmic version of Szemerédi's regularity lemma. Unlike all the previous approaches for this problem, which only guaranteed to find partitions of tower-size in ε (regardless of the nature of the graph), our algorithm will find a small regular partition in the case that one exists. In addition, for any $r \geq 3$, we give an $O(n)$ time randomized algorithm for constructing weakly regular partitions of r-uniform hypergraphs, thus improving upon the previous $O(n^{2r-1})$ time deterministic algorithms.

The hypergraph partition testing algorithm is used to unify many previous results in hypergraph property testing and CNF testing. It can also be used to obtain the partition densities for the regularity problem (rather than the partition itself) using only poly$(1/\varepsilon)$ queries (for partitions of fixed size) and constant running time.

2.3 Testing *st*-connectivity

A major question that has remained open in [HLNT05] is whether connectivity properties admit a constant query complexity test. For a fixed $s, t \in V(G)$, an orientation $\overrightarrow{G}$ is st-connected if there is a directed path from s to t in it. Connectivity and in particular st-connectivity is a very basic problem in graph theory which has been extensively studied in various models of computation.

Our first result in the study of massively parameterized properties (Chapter 7 – based on [CFL$^+$07]) is that the property of being st-connected is testable by a one-sided error algorithm with a number of queries depending only on ε. That is, we construct a randomized algorithm such that for any underlying graph G, on input of an unknown orientation the algorithm queries only $O(1)$ edges for their orientation and based on this decides with success probability $\frac{2}{3}$ (this of course could be amplified to any number smaller than 1) between the case that the orientation is st-connected and the case that it is ε-far from being st-connected. Our algorithm additionally has one-sided error, meaning that st-connected orientations are accepted with probability 1. Recall that in this setting the algorithm knows the underlying graph G in advance (and may do preprocessing on it) and G is neither alterable nor part of the input to be queried.

Our proof uses combinatorial reduction arguments with a concentration type lemma that is proven for this purpose. Unlike many traditional property testing results, here the resulting testing algorithm is highly non-trivial.

2.4 Testing orientations for being Eulerian

Our second result within the topic of massively parameterized properties (Chapter 8 – based on parts from [FMNY08]) solves an open problem from [CFL$^+$07] (the same paper that is summarized in the previous section). Again, the undirected graph (the skeleton) is known in advance, and we need to test whether the orientation of the graph is Eulerian. Despite the local nature of the property of being Eulerian, it turns out to be significantly harder for testing than other properties studied in the orientation model. In particular, we show a non-constant lower bound on the query complexity of even 2-sided error tests for this property (namely, tests that are allowed to reject inputs that satisfy the property).

Additional results related to this topic were published in [FMNY08], and also appear in the Ph.D. thesis of Orly Yahalom.

2.5 Length-Soundness tradeoffs for 3-query PCPPs

As mentioned earlier, PCPP verifiers have four parameters of interest – proof length, query complexity, completeness and a soundness function. Recall that the soundness function $s(\delta)$ is the minimal rejection probability of inputs that are δ-far from the property P, where the minimum is taken over all such δ-far inputs and all possible proofs that may accompany them. In Chapter 9 (based on [BHLM08]) we study the relations between the proof length and the soundness of PCPP verifiers that make three queries and have perfect completeness. In particular, we quantify the *soundness deficiency* as a function of the proof-length and show that any verifier obtaining the "best possible" soundness must query an exponentially long proof.

Chapter 3

Global definitions and preliminaries

3.1 General notations

We denote by $[n]$ the set $\{1, 2, \ldots, n\}$, and for $\ell < n$ we denote by $[\ell, n]$ the set $\{\ell, \ell + 1, \ldots, n\}$. For two finite sets A and B, we denote by $A \triangle B$ their symmetric difference (i.e. the set $A \setminus B \cup B \setminus A$). For an alphabet Σ and a pair of words $x, y \in \Sigma^n$, we define the fractional Hamming distance

$$\delta(x, y) \triangleq \frac{|\{i \in [n] : x_i \neq y_i\}|}{n}.$$

Fora property defined as a set $P \subseteq \Sigma^n$, we define

$$\delta(x, P) \triangleq \min_{y \in P}\{\delta(x, y)\}.$$

A word x is said to be *ε-far* from P whenever $\delta(x, P) \geq \varepsilon$.

3.2 Property testers

Definition 1. *An* ε-testing algorithm with q queries *for a property $P \subseteq \Sigma^n$ is a probabilistic algorithm, that for any input $x \in \Sigma^n$ makes up to q queries into x, and satisfies the following.*

- *If x satisfies P then the algorithm accepts x with probability at least $\frac{2}{3}$.*
- *If x is ε-far from P, then the algorithm rejects x with probability at least $\frac{2}{3}$.*

A property testing algorithm has one-sided error probability *if it accepts inputs that satisfy the property with probability* 1*. We also call such testers* one-sided error testers*, or* testers with perfect completeness.

A property testing algorithm is non-adaptive *if the outcomes of its queries do not affect the choice of the following queries, but only the decision of whether to reject or accept the input in the end.*

Later we will need a more general definition of a property tester, in which we also measure the *soundness function* of the testing algorithm. We postpone this definition to the parts where it becomes necessary (Section 9.2).

A property that is testable by looking at a portion of the input whose size is a function of ε only (and is independent of the input size n) is plainly called *testable*.

3.3 Graphs and graph properties

Unless stated otherwise, all of our graphs are simple, and contain no loops nor parallel edged. We always assume (even where not explicitly stated) that the number of vertices (usually denoted by n) of the input graph(s) is large enough, as a function of the other parameters. For a graph $G = (V, E)$ and a vertex $v \in V$, $\Gamma(v) \subseteq V$ denotes the set of v's neighbors in G. Given a subset U of the vertices of a graph G, we denote by $G(U)$ the induced subgraph of G on U.

We denote by $G \sim G(n, p)$ the random graph where each pair of vertices forms an edge with probability p, independently of other pairs. Note that this is in fact a probability space, but sometimes we will abuse notation and use $G(n, p)$ to denote a graph drawn according to this space.

3.3.1 Distance between graphs and properties

Definition 2. *Given two labeled graphs G and H on the same vertex set V, the* distance *between G and H is the size of the symmetric difference between the edge sets of G and H, divided by $\binom{|V|}{2}$.*

Given a graph G and a graph H on the same vertex set V, we say that H and G are ε-far*, if the distance between G and any permutation (vertex renaming) of H is at least ε.*

Given a graph G and a graph property (a set of graphs that is closed under graph isomorphisms) P, we say that G is ε-far *from satisfying the property P, if G is ε-far from any graph H on the same vertex set which satisfies P. Otherwise G is ε-close to P.*

Using this definition of the distance from graph properties together with Definition 3.2 we get the formal definition of a graph testing algorithm. Notice that this definition is relevant only for the dense graphs model, where the queries correspond to reading an entry in the adjacency matrix.

3.4 Restriction, variation distance and Yao's method

Proving lower bounds for the two-sided error testers involves Yao's method [Yao77], which for our context informally says that if there is a small enough statistical distance between the distributions of q query results, from two distributions over inputs that satisfy the property and inputs that are far from satisfying the property, then there is no tester for that property which makes at most q queries. We start with definitions that are adapted to property testing lower bounds.

Definition 3 (restriction, variation distance). *For a distribution D over inputs, where each input is a function $f : \mathcal{D} \to \{0,1\}$, and for a subset $\mathcal{Q}$ of the domain $\mathcal{D}$, we define the* restriction *$D|_{\mathcal{Q}}$ of D to $\mathcal{Q}$ to be the distribution over functions of the type $g : \mathcal{Q} \to \{0,1\}$, that results from choosing a random function $f : \mathcal{D} \to \{0,1\}$ according to the distribution D, and then setting g to be $f|_{\mathcal{Q}}$, the restriction of f to $\mathcal{Q}$.*

Given two distributions D_1 and D_2 of binary functions from $\mathcal{Q}$, we define the variation distance *between D_1 and D_2 as follows: $d(D_1, D_2) = \frac{1}{2}\sum_{g:\mathcal{Q}\to\{0,1\}} |\mathrm{Pr}_{D_1}[g] - \mathrm{Pr}_{D_2}[g]|$, where $\mathrm{Pr}_D[g]$ denotes the probability that a random function chosen according to D is identical to g.*

The next lemma follows from [Yao77] (see e.g. [Fis04]):

Lemma 3.4.1 (see [Fis04]). *Suppose that there exists a distribution D_P on inputs over $\mathcal{D}$ that satisfy a given property P, and a distribution D_N on inputs that are ε-far from satisfying the property, and suppose further that for any $\mathcal{Q} \subset \mathcal{D}$ of size q, the variation distance between $D_P|_{\mathcal{Q}}$ and $D_N|_{\mathcal{Q}}$ is less than $\frac{1}{3}$. Then it is not possible for a non-adaptive algorithm making q (or less) queries to ε-test for P.*

An additional lemma for adaptive testers is proven implicitly in [FNS04], and a detailed proof appears in [Fis04]. Here we strengthen it somewhat, but still exactly the same proof works in our case too.

Lemma 3.4.2 ([FNS04], see [Fis04]). *Suppose that there exists a distribution D_P on inputs over $\mathcal{D}$ that satisfy a given property P, and a distribution D_N on inputs that are ε-far from satisfying the property. Suppose further that for any $\mathcal{Q} \subset \mathcal{D}$ of size q, and any $g : \mathcal{Q} \to \{0,1\}$, we have $\mathrm{Pr}_{D_P|_{\mathcal{Q}}}[g] < \frac{3}{2}\mathrm{Pr}_{D_N|_{\mathcal{Q}}}[g]$. Then it is not possible for any algorithm making q (or less) queries to ε-test for P. The conclusion also holds if instead of the above, for any $\mathcal{Q} \subset \mathcal{D}$ of size q and any $g : \mathcal{Q} \to \{0,1\}$, we have $\mathrm{Pr}_{D_N|_{\mathcal{Q}}}[g] < \frac{3}{2}\mathrm{Pr}_{D_P|_{\mathcal{Q}}}[g]$.*

Part I

Dense Graphs and Hypergraphs

Chapter 4

Testing graph isomorphism

Two graphs G and H on n vertices are ε-far from being isomorphic if at least $\varepsilon\binom{n}{2}$ edges must be added or removed from $E(G)$ in order to make G and H isomorphic. In this chapter we deal with the question of how many queries are required to distinguish between the case that two graphs are isomorphic, and the case that they are ε-far from being isomorphic.

We investigate both one-sided error and two-sided error testers under two possible settings: The first setting is where both graphs need to be queried; and the second setting is where one of the graphs is fully known to the algorithm in advance.

We prove that the query complexity of the best one-sided error testing algorithm is $\widetilde{\Theta}(n^{3/2})$ if both graphs need to be queried, and that it is $\widetilde{\Theta}(n)$ if one of the graphs is known in advance (where the $\widetilde{\Theta}$ notation hides polylogarithmic factors in the upper bounds only). For two-sided error testers, we prove that the query complexity of the best tester is $\widetilde{\Theta}(\sqrt{n})$ when one of the graphs is known in advance, and we show that the query complexity lies between $\Omega(n)$ and $\widetilde{O}(n^{5/4})$ if both G and H need to be queried. All of our algorithms are additionally non-adaptive, while all of our lower bounds apply for adaptive testers as well as non-adaptive ones.

4.1 Background and introduction

Since we study graph isomorphism (a property of pairs of graphs), our input consists of two functions $g : \{1, 2, \ldots, \binom{n}{2}\} \to \{0, 1\}$ and $h : \{1, 2, \ldots, \binom{n}{2}\} \to \{0, 1\}$, which represent the edge sets of two corresponding graphs G and H over the vertex set $V = \{1, \ldots, n\}$.

We consider the following questions:

1. Given two input graphs G and H, how many queries to G and H are required to test that the two graphs are isomorphic? This property was already used in [AFKS00] for proving lower bounds on property testing, and a lower bound of the form n^{α} was

known for quite a while (see e.g. [Fis04]).

2. Given a graph G_k, which is known in advance (and for which any amount of preprocessing is allowed), and an input graph G_u, how many queries to G_u are required to test that G_u is isomorphic to G_k? Some motivation for this question comes from [Fis05], where upper and lower bounds that correlate this question with the "inherent complexity" of the provided G_k are proven. Here our main interest is in finding the bounds for the "worst possible" G_k.

For the case where the testers must have one-sided error, our results show tight (up to logarithmic factors) upper and lower bounds, of $\widetilde{\Theta}(n^{3/2})$ for the setting where both graphs need to be queried, and $\widetilde{\Theta}(n)$ for the setting where one graph is given in advance. The upper bounds are achieved by trivial algorithms of edge sampling and exhaustive search. As we are interested in the number of queries we make no attempt to optimize the running time (but our two-sided error algorithms have a better running time as well). The main work here lies in proving a matching lower bound for the first setting where both graphs need to be queried, as the lower bound for the second setting is nearly trivial.

Unusually for graph properties that involve no explicit counting in their definition, we can do significantly better if we allow our algorithms to have two-sided error. When one graph is given in advance, we show $\widetilde{\Theta}(n^{1/2})$ upper and lower bounds. The upper bound algorithm uses a technique that allows us to greatly reduce the number of candidate bijections that need to be checked, while assuring that for isomorphic graphs one of them will still be close to an isomorphism. For this to work we need to combine it with a distribution testing algorithm from [BFF+01], whose lower bound is in some sense the true cause of the matching lower bound here.

For two-sided error testers where the two graphs need to be queried, a gap in the bounds remains. We present here a lower bound proof of $\Omega(n)$ on the query complexity – it is in fact the lower bound proof already used previously, only here we analyze it to its fullest potential. The upper bound of $\widetilde{O}(n^{5/4})$ uses the ideas of the algorithm above for the setting where one of the graphs is known, with an additional mechanism to compensate for having to query from both graphs to find matching vertices.

To our knowledge, the best known algorithm for deciding the corresponding promise problem in the classical sense (i.e., given two graphs distinguish whether they are isomorphic or ε-far from being isomorphic) requires quasi-polynomial running time [AFK96]. Both our two-sided error testers have the additional property of a quasi-polynomial running time (similarly to the algorithm in [AFK96]) even with the restriction on the number of queries.

The following is the summary of our results for the query complexity in various settings. We made no effort to optimize the logarithmic factors in the upper bounds, as well as the exact dependance on ε (which is at most polynomial).

	Upper bound	Lower bound
One sided error, one graph known	$\widetilde{O}(n)$	$\Omega(n)$
One sided error, both graphs unknown	$\widetilde{O}(n^{3/2})$	$\Omega(n^{3/2})$
Two sided error, one graph known	$\widetilde{O}(n^{1/2})$	$\Omega(n^{1/2})$
Two sided error, both graphs unknown	$\widetilde{O}(n^{5/4})$	$\Omega(n)$

The rest of this chapter is organized as follows. We provide some additional (specific) preliminaries and definitions in Section 4.2. Upper and lower bounds for the one-sided algorithms are proven in Section 4.3, and the upper and lower bounds for the two-sided algorithms are proven in Section 4.4. Section 4.5 contains some discussion and comments on future work.

4.2 Specific definitions and preliminaries

First we formulate an extension of Definition 1 to properties of pairs of graphs. In our case, we will be interested in the property of two graphs being isomorphic.

Definition 4. *An ε-*testing algorithm with q queries *for a property P of pairs of graphs is a probabilistic algorithm, that for any input pair G,H makes up to q queries in G and H (a query consisting of finding whether two vertices u, v of G or H form an edge of that graph or not), and satisfies the following.*

- *If the pair G,H satisfies P then the algorithm accepts with probability at least $\frac{2}{3}$.*
- *If the pair G,H is ε-far from P, then the algorithm rejects with probability at least $\frac{2}{3}$.*

To simplify the arguments when discussing the properties of the query sets, we define *knowledge charts.*

Definition 5. *Given a query set Q to the adjacency matrix A of the graph $G = (V, E)$ on n vertices, we define the* knowledge chart *$I_{G,Q}$ of G as the subgraph of G known after making the set Q of queries to A. We partition the pairs of vertices of $I_{G,Q}$ into three classes: Q^1, Q^0 and Q^*. The pairs in Q^1 are the ones known to be edges of G, the pairs in Q^0 are those that are known not to be edges of G, and all unknown (unqueried) pairs are in Q^*. In other words, $Q^1 = E(G) \cap Q$, $Q^0 = Q \setminus E(G)$, and $Q^* = [V(G)]^2 \setminus Q$. For a fixed q, $0 \leq q \leq n$, and G, we define $I_{G,q}$ as the set of all possible knowledge charts $\{I_{G,Q} : |Q| = q\}$. For example, note that $|I_{G,0}| = |I_{G,\binom{n}{2}}| = 1$.*

We will ask the question of whether two query sets are consistent, i.e. they do not provide an evidence for the two graphs being non-isomorphic. We say that the knowledge charts are *knowledge-packable* if the query sets that they represent are consistent. Formally,

Definition 6. *A* knowledge-packing *of two knowledge charts I_{G_1,Q_1}, I_{G_2,Q_2}, where G_1 and G_2 are graphs with n vertices, is a bijection π of the vertices of G_1 into the vertices of G_2 such that for all $v, u \in V(G_1)$, if $\{v,u\} \in E(G_1) \cap Q_1$ then $\{\pi(v), \pi(u)\} \notin Q_2 \setminus E(G_2)$, and if $\{v,u\} \in Q_1 \setminus E(G_1)$ then $\{\pi(v), \pi(u)\} \notin E(G_2) \cap Q_2$.*

In particular, if G_1 is isomorphic to G_2, then for all $0 \le q_1, q_2 \le \binom{n}{2}$, every member of I_{G_1,q_1} is knowledge-packable with every member of I_{G_2,q_2}. In other words, if G_1 is isomorphic to G_2, then there is a knowledge-packing of I_{G_1,Q_1} and I_{G_2,Q_2} for any possible query sets Q_1 and Q_2.

Lemma 4.2.1. *Any one-sided error isomorphism tester, after completing its queries Q_1, Q_2, must always accept G_1 and G_2 if the corresponding knowledge charts I_{G_1,Q_1}, I_{G_2,Q_2} on which the decision is based are knowledge-packable. In particular, if for some G_1, G_2 and $0 \le q \le \binom{n}{2}$, any $I_{G_1,Q_1} \in I_{G_1,q}$ and $I_{G_2,Q_2} \in I_{G_2,q}$ are knowledge-packable, then every one-sided error isomorphism tester which is allowed to ask at most q queries must always accept G_1 and G_2.*

Proof. This is true, since if the knowledge charts I_{G_1,Q_1} and I_{G_2,Q_2} are packable, then there is an extension $G_1^{'}$ of G_1's restriction to Q_1 to a graph that is isomorphic to G_2. In other words, given $G_1^{'}$ and G_2 as inputs, there is a positive probability that the isomorphism tester obtained $I_{G_1',Q_1} = I_{G_1,Q_1}$ and I_{G_2,Q_2} after completing its queries, and hence, a one-sided error tester must always accept in this case. □

Often, given two isomorphic graphs G, H on n vertices, we want to estimate how many vertices from both graphs need to be randomly chosen in order to get an intersection set of size k with high probability.

Lemma 4.2.2. *Given two graphs G, H on n vertices, a bijection σ of their vertices, and two uniformly random subsets $C_G \subset V(G), C_H \subset V(H)$, the following holds: for any $0 < \alpha < 1$ and any positive integers c, k, if $|C_G| = kn^{\alpha} \log^c n$ and $|C_H| = n^{1-\alpha} \log^c n$, then with probability $1 - o(2^{-\log^c n})$ the size of $C_G \cap \sigma(C_H)$ is greater than k.*

Proof sketch. By the linearity of expectation, the expected size of the intersection set is $\frac{|C_G||C_H|}{n} = k \log^{2c} n$. Using large deviation inequalities, $C_G \cap \sigma(C_H) > k$ with probability $1 - o(2^{-\log^c n})$. □

4.3 One-sided testers

By Lemma 4.2.1, one-sided testers for isomorphism look at some query set Q of the input, and accept if and only if the restriction of the input to Q is extensible to some input satisfying the property. The main idea is to prove that if the input is far from satisfying the property, then with high probability its restriction Q will provide the evidence for it. To prove lower bounds for one-sided testers, it is sufficient to find an input that is ε-far from satisfying the property, but for which the restriction of the input to any possible set Q is extensible to some alternative input that satisfies the property. In this section we prove the following:

Theorem 4.3.1. *The query complexity of the best one-sided isomorphism tester is $\widetilde{\Theta}(n^{3/2})$ (up to coefficients depending only on the distance parameter ε) if both graphs are unknown, and it is $\widetilde{\Theta}(n)$ if one of the graphs is known in advance.*

We first prove Theorem 4.3.1 for the case where both graphs are unknown, and then move to the proof of the simpler second case where one of the graphs is known in advance.

4.3.1 One-sided testing of two unknown graphs

The upper bound

Algorithm 1.

1. *For both graphs G_1, G_2 construct the query sets Q_1, Q_2 respectively by choosing every possible query with probability $\sqrt{\frac{\ln n}{\varepsilon n}}$, independently of other queries.*
2. *If $|Q_1|$ or $|Q_2|$ is larger than $1000n^{3/2}\sqrt{\frac{\ln n}{\varepsilon}}$, accept without making the queries. Otherwise make the chosen queries.*
3. *If there is a knowledge-packing of I_{G_1,Q_1} and I_{G_2,Q_2}, accept. Otherwise reject.*

Clearly, the query complexity of Algorithm 1 is $O(n^{3/2}\sqrt{\log n})$ for every fixed ε.

Lemma 4.3.2. *Algorithm 1 accepts with probability 1 if G_1 and G_2 are isomorphic, and if G_1 and G_2 are ε-far from being isomorphic, Algorithm 1 rejects with probability $1 - o(1)$.*

Proof. Assume first that G_1 and G_2 are isomorphic, and let π be an isomorphism between them. Obviously π is also a knowledge-packing for any pair of knowledge charts of G_1 and G_2. Hence, if the algorithm did not accept in the second stage, then it will accept in the third stage.

Now we turn to the case where G_1 and G_2 are ε-far from being isomorphic. Due to large deviation inequalities, the probability that Algorithm 1 terminates in Step 2 is $o(1)$, and therefore we can assume in the proof that it reaches Step 3 without harming the correctness. Since G_1 and G_2 are ε-far from being isomorphic, every possible bijection π of their vertices has a set E_π of at least εn^2 pairs of G_1's vertices such that for every $\{u, v\} \in E_\pi$, either $\{u, v\}$ is an edge in G_1 or $\{\pi(u), \pi(v)\}$ is an edge in G_2 but not both. Now we fix π and let $\{u, v\} \in E_\pi$ be one such pair. The probability that $\{u, v\}$ was not queried in G_1 or $\{\pi(u), \pi(v)\}$ was not queried in G_2 is $1 - \frac{\ln n}{\varepsilon n}$. Since the queries where chosen independently, the probability that for all $\{u, v\} \in E_\pi$ that either $\{u, v\}$ was not queried in G_1 or $\{\pi(u), \pi(v)\}$ was not queried in G_2 is at most $(1 - \frac{\ln n}{\varepsilon n})^{\varepsilon n^2}$. Using the union bound, we bound the probability of not revealing at least one such pair in both graphs for all possible bijections by $n!(1 - \frac{\ln n}{\varepsilon n})^{\varepsilon n^2}$. This bound satisfies

$$n!(1 - \frac{\ln n}{\varepsilon n})^{\varepsilon n^2} \leq n!(e^{-\frac{\ln n}{\varepsilon n}})^{\varepsilon n^2} = n!\frac{1}{n^n} = o(1)$$

thus the algorithm rejects graphs that are ε-far from being isomorphic with probability $1 - o(1)$. □

The lower bound

Here we construct a pair G, H of $1/100$-far graphs on n vertices, such that every knowledge chart from $I_{G,n^{3/2}/200}$ can be packed with every knowledge chart from $I_{H,n^{3/2}/200}$, and hence by Lemma 4.2.1, any one-sided algorithm which is allowed to use at most $n^{3/2}/200$ queries must always accept G and H. Note that this holds for non-adaptive as well as adaptive algorithms, since we actually prove that there is no certificate of size $n^{3/2}/200$ for the non-isomorphism of these two graphs.

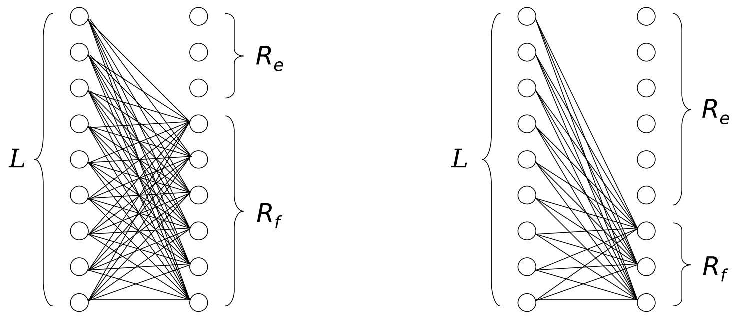
Lemma 4.3.3. *For every large enough n, there are two graphs G and H on n vertices, such that:*

1. *G is $1/100$-far from being isomorphic to H*
2. *Every knowledge chart from $I_{G,n^{3/2}/200}$ can be knowledge-packed with any knowledge chart from $I_{H,n^{3/2}/200}$*

Proof. We set both G and H to be the union of a complete bipartite graph with a set of isolated vertices. Formally, G has three vertex sets L, R_f, R_e, where $|L| = n/2, |R_f| = 26n/100$ and $|R_e| = 24n/100$, and it has the following edges: $\{\{u, v\} : u \in L \wedge v \in R_f\}$. H has the same structure, but with $|R_f| = 24n/100$ and $|R_e| = 26n/100$, as illustrated in Figure 8.1. Clearly, just by the difference in the edge count, G is $1/100$-far from being isomorphic to H, so G and H satisfy the first part of Lemma 4.3.3.

To prove that the second condition of Lemma 4.3.3 holds, we will show that for all possible query sets Q_G, Q_H of size $n^{3/2}/200$ there exist sets $Y_G \in R_f(G)$ and $Y_H \in R_e(H)$ that satisfy the following.

Figure 4.1: The graphs G and H (with the difference between them exaggerated)

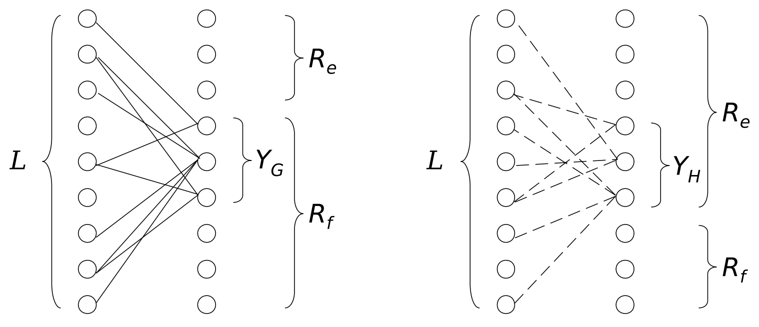


- $|Y_G| = |Y_H| = n/50$
- the knowledge charts I_{G,Q_G} and I_{H,Q_H} restricted to $L(G) \cup Y_G$ and $L(H) \cup Y_H$ can be packed in a way that pairs vertices from $L(G)$ with vertices from $L(H)$

In Figure 8.2 we illustrate these restricted knowledge charts, where solid lines are known (queried) edges, and dashed lines are known (queried) "non-edges". The existence of such Y_G and Y_H implies the desired knowledge-packing, since we can complete the partial packing from the second item by arbitrarily pairing vertices from $R_f(G) \setminus Y_G$ with vertices from $R_f(H)$, and pairing vertices from $R_e(G)$ with vertices from $R_e(H) \setminus Y_H$.

Remark 4.3.4. Note that there is a trivial algorithm that distinguishes between the two graphs in $O(n)$ queries by sampling vertices and checking their degrees. However, such an algorithm has two-sided error. Any one-sided error algorithm must find evidence to the non-isomorphism of the graphs, i.e. two knowledge charts that cannot be packed (in the sense that there is no isomorphism consistent with them).

Figure 4.2: Finding Y_G and Y_H



Proving the existence of Y_G and Y_H

For every vertex $v \in V(G)$, we define its query degree as

$$d_Q(v) = \left|\left\{\{v,u\} : u \in V(G) \wedge \{v,u\} \in Q_G\right\}\right|$$

We also denote by $N_Q(v)$ the set $\{u : \{v,u\} \in E(G) \cap Q_G\}$ and we denote by $\overline{N}_Q(v)$ the set $\{u : \{v,u\} \in Q_G \setminus E(G)\}$. In other words, $N_Q(v)$ is the set of known neighbors of v, $\overline{N}_Q(v)$ is the set of known non-neighbors of v, and $d_Q(v) = |\overline{N}_Q(v)| + |N_Q(v)|$. We define $d_Q(v)$, $N_Q(v)$ and $\overline{N}_Q(v)$ for H's vertices similarly.

Since $|Q_G|, |Q_H| \le n^{3/2}/200$, there must be two sets of vertices $D_G \in R_f(G)$ and $D_H \in R_e(H)$, both of size $n/10$, such that $\forall_{v\in D_G} : d_Q(v) \le n^{1/2}/2$ and $\forall_{v\in D_H} : d_Q(v) \le n^{1/2}/2$.

Now we prove the existence of Y_G and Y_H (as defined above) using a simple probabilistic argument. First we set an arbitrary pairing $B_D = \{\{v_G^1, u_H^1\}, \{v_G^2, u_H^2\}, \ldots, \{v_G^{n/10}, u_H^{n/10}\}\}$ of D_G's and D_H's elements. Then we choose a bijection $B_L : L(G) \to L(H)$ uniformly at random, and show that with some positive probability, there are at least $n/50$ consistent (packable) pairs in B_D. Formally, we define

$$Y = \left\{\{v_G, u_H\} \in B_D : B_L(N_Q(v_G)) \cap \overline{N}_Q(u_H) = \emptyset\right\}$$

as the set of consistent pairs, and show that $\Pr[|Y| \ge n/50] > 0$.

For a specific pair $\{v \in D_G, u \in D_H\}$, we have

$$\Pr_{B_L}[B_L(N_Q(v)) \cap \overline{N}_Q(u) = \emptyset] \ge \prod_{i=0}^{n^{1/2}/2-1} (1 - \frac{n^{1/2}/2}{n/2 - i})$$

$$\ge (1 - \frac{2n^{1/2}}{n})^{n^{1/2}/2} \ge (e + 0.001)^{-1} \ge 1/3$$

and by the linearity of expectation, $\mathrm{E}[|Y|] \ge |D_G|/3 > n/50$. Therefore, there is at least one bijection B_L for which the size of Y is no less than its expectation. We can now set

$$Y_G = \{u : \exists v \in V(H) \text{ such that } \{u,v\} \in Y\}$$

and

$$Y_H = \{v : \exists u \in V(G) \text{ such that } \{u,v\} \in Y\}$$

concluding the proof. □

4.3.2 One-sided testing where one of the graphs is known in advance

The algorithm for testing isomorphism between an unknown graph and a graph that is known in advance is similar to Algorithm 1 above. In this case the algorithm makes a quasi-linear number of queries, to accept with probability 1 if the graphs are isomorphic and reject with probability $1 - o(1)$ if they are ε-far from being isomorphic. We also prove an almost matching nearly trivial lower bound for this problem.

The upper bound

Denote by G_k and G_u the known and the unknown graphs respectively.

Algorithm 2.

1. *Construct a query set Q by choosing every possible query from G_u with probability $\frac{\ln n}{\varepsilon n}$, independently at random.*
2. *If $|Q|$ is larger than $\frac{10 n \ln n}{\varepsilon}$, accept without making the queries. Otherwise make the chosen queries.*
3. *If there is a knowledge-packing of $I_{G_u,Q}$ and $I_{G_k,[V(G_k)]^2}$, accept. Otherwise reject.*

Clearly the query complexity of Algorithm 2 is $O(n \log n)$, and it rejects in Step 2 with probability $o(1)$.

Lemma 4.3.5. *Algorithm 2 always accepts isomorphic graphs, and it rejects ε-far graphs with probability $1 - o(1)$.*

Proof. The proof is almost identical to that of Lemma 4.3.2. It is clear that isomorphic graphs are always accepted by Algorithm 2. Now we assume that the graphs G_k and G_u are ε-far and that the algorithm reached Step 3 (as it stops at Step 2 with probability $o(1)$) . Given a bijection π, the probability that no violating pair $\{u, v\} \in E_\pi$ was queried is at most $(1 - \frac{\ln n}{\varepsilon n})^{\varepsilon n^2} \leq e^{-n \ln n} = n^{-n}$. Applying the union bound over all $n!$ possible bijections, the acceptance probability is bounded by $n!/n^n = o(1)$ □

The lower bound

As before, to give a lower bound on one-sided error algorithms it is sufficient to show that for some G_k and G_u that are far, no "proof" of their non-isomorphism can be provided with $\Omega(n)$ queries. First we formulate the second part of Lemma 4.2.1 for the special case where one of the graphs is known in advance.

Lemma 4.3.6. *If for some G_k, G_u, where G_k is known in advance, and some fixed $0 \leq q \leq \binom{n}{2}$, $I_{G_k,[v(G_k)]^2}$ is knowledge-packable with every $I_{G_u,Q} \in I_{G_u,q}$, then every one-sided error isomorphism tester which is allowed to ask at most q queries must always accept G_k and G_u.*

We set G_k to be a disjoint union of $K_{n/2}$ and $n/2$ isolated vertices, and set G_u to be a completely edgeless graph.

Observation 4.3.7. *G_k and G_u are $1/4$-far, and every $I_{G_u,Q} \in I_{G_u,\frac{n}{4}}$ is knowledge-packable with $I_{G_k,[V(G_k)]^2}$.*

Proof. Clearly, just by the difference in the edge count, G_k is $1/4$ far from being isomorphic to G_u. But since $n/4$ queries cannot involve more than $n/2$ vertices from G_u (all isolated), and G_k has $n/2$ isolated vertices, the knowledge charts are packable. $\square$

Together with Lemma 4.3.6, we get the desired lower bound. This concludes the proof of the last part of Theorem 4.3.1.

4.4 Two-sided testers

In the context of graph properties, two-sided error testers are usually not known to achieve significantly lower query complexity than the one-sided error testers, apart from the properties that explicitly involve counting, such as *Max-Cut* and *Max-Clique* [GGR98]. However, in our case two-sided error isomorphism testers have substantially lower query complexity than their one-sided error counterparts.

4.4.1 Two-sided testing where one of the graphs is known in advance

Theorem 4.4.1. *The query complexity of two-sided error isomorphism testers is $\widetilde{\Theta}(\sqrt{n})$ if one of the graphs is known in advance, and the other needs to be queried.*

We prove the lower bound first. This way it will be easier to understand why certain stages of the upper bound testing algorithm are necessary.

The lower bound

Lemma 4.4.2. *Any isomorphism tester that makes at most $\frac{\sqrt{n}}{4}$ queries to G_u cannot distinguish between the case that G_k and G_u are isomorphic and the case that they are $1/32$-far from being isomorphic, where G_k is known in advance.*

We begin with a few definitions.

Definition 7. *Given a graph G and a set W of $\frac{n}{2}$ vertices of G, we define the* clone $G^{(W)}$ *of G in the following way:*

- *the vertex set of $G^{(W)}$ is defined as: $V(G^{(W)}) = W \cup \{w' : w \in W\}$*
- *the edge set of $G^{(W)}$ is defined as: $E(G^{(W)}) =$*

$$\Big\{\{v,u\} : \{v,u\} \in E(G)\Big\} \cup \Big\{\{v',u\} : \{v,u\} \in E(G)\Big\} \cup \Big\{\{v',u'\} : \{v,u\} \in E(G)\Big\}$$

In other words, $G^{(W)}$ is the product of the subgraph of G induced on W with the graph K_2.

For the two copies $v, v' \in V(G^{(W)})$ of $v \in W$, we say that v is the source *of v and v'.*

Lemma 4.4.3. *Let $G \sim G(n, 1/2)$ be a random graph. With probability $1 - o(1)$ the graph G is such that for every subset $W \subset V(G)$ of size $n/2$, the clone $G^{(W)}$ of G is $1/32$-far from being isomorphic to G.*

Proof. Let G be a random graph according to $G(n, 1/2)$, and let $W \subset V(G)$ be an arbitrary subset of G's vertices of size $n/2$. First we show that for an arbitrary bijection $\sigma : V(G^{(W)}) \to V(G)$ the graphs $G^{(W)}$ and G are $1/32$-close under σ with probability at most $2^{-\Omega(n^2)}$, and then we apply the union bound on all bijections and every possible subset W.

We split the bijection $\sigma : V(G^{(W)}) \to V(G)$ into two injections $\sigma_1 : W \to V(G)$ and $\sigma_2 : V(G^{(W)}) \setminus W \to V(G) \setminus \sigma_1(W)$. Note that either $|W \setminus \sigma_1(W)| \geq n/4$ or $|W \setminus \sigma_2(W)| \geq n/4$. Assume without loss of generality that the first case holds, and let U denote the set $W \setminus \sigma_1(W)$. Since every edge in G is chosen at random with probability $1/2$, the probability that for some pair $u, v \in U$ either $\{u, v\}$ is an edge in G and $\{\sigma(u), \sigma(v)\}$ is not an edge in G or $\{u, v\}$ is not an edge in G and $\{\sigma(u), \sigma(v)\}$ is an edge in G is exactly $1/2$. Moreover, these events are independent for all pairs in U. Therefore, using large deviation inequalities, the probability that in the set U there are less than $\binom{n}{2}/32$ such pairs is at most $2^{-\Omega(n^2)}$. There are at most $n!$ possible bijections, and $\binom{n}{n/2}$ possible choices for W, so using the union bound, the probability that for some W the graph $G \sim G(n, 1/2)$ is not $1/32$-far from being isomorphic to some $G^{(W)}$ is at most $2^{-\Omega(n^2)}\binom{n}{n/2}n! = o(1)$. $\square$

Given a graph G satisfying the assertion of Lemma 4.4.3, we set $G_k = G$ and define two distributions over graphs, from which we choose the unknown graph G_u:

- D_P: A permutation of G_k, chosen uniformly at random.
- D_N: A permutation of $G_k^{(W)}$, where both W and the permutation are chosen uniformly at random.

According to Lemma 4.4.3 and Lemma 3.4.2, it is sufficient to show that the distributions D_P and D_N restricted to a set of $\sqrt{n}/4$ queries are close. In particular, we intend to show that for any $\mathcal{Q} \subset \mathcal{D} = V^2$ of size $\sqrt{n}/4$, and any $Q : \mathcal{Q} \to \{0,1\}$, we have $\Pr_{D_P|_\mathcal{Q}}[Q] < \frac{3}{2}\Pr_{D_N|_\mathcal{Q}}[Q]$. This will imply a lower bound for adaptive (as well as non-adaptive) testing algorithms.

Observation 4.4.4. *For a set U of $G^{(W)}$'s vertices, define the event E_U as the event that there is no pair of copies w, w' of any one of G's vertices in U. For a given set Q of pairs of vertices, let U_Q be the set of all vertices that belong to some pair in Q. Then the distribution $D_N|_\mathcal{Q}$ conditioned on the event E_{U_Q} (defined above) and the unconditioned distribution $D_P|_\mathcal{Q}$ are identical.*

Proof. In D_N, if no two copies of any vertex were involved in the queries, then the source vertices of the queries to G_u are in fact a uniformly random sequence (with no repetition) of the vertices of G_k, and this (together with G_k) completely determines the distribution of the answers to the queries. This is the same as the unconditioned distribution induced by D_P. □

Intuitively, the next lemma states that picking two copies of the same vertex in a randomly permuted $G^{(W)}$ requires many samples, as per the well known birthday problem.

Lemma 4.4.5. *For a fixed set Q of at most $\sqrt{n}/4$ queries and the corresponding set $U = U_Q$ of vertices, the probability that the event E_U did not happen is at most* $1/4$.

Proof. The bound on $|Q|$ implies that $|U| \leq \sqrt{n}/2$. Now we examine the vertices in U as if we add them one by one. The probability that a vertex v that is added to U is a copy (with respect to the original graph G) of some vertex u that was already inserted to U (or vice versa) is at most $\frac{\sqrt{n}}{2n}$. Hence, the probability that eventually (after $\sqrt{n}/2$ insertions) we have two copies of the same vertex in U is at most $\frac{\sqrt{n}}{2n} \cdot \sqrt{n}/2 = 1/4$. □

From Observation 4.4.4, the distribution $D_N|_\mathcal{Q}$ conditioned on the event E_U and the unconditioned distribution $D_P|_\mathcal{Q}$ are identical. By Lemma 4.4.5 it follows that $\Pr[E_U] > 2/3$. Therefore, for any $g : \mathcal{Q} \to \{0,1\}$ we have

$$\Pr_{D_N|_\mathcal{Q}}[g] = \Pr[E_U]\cdot\Pr_{D_P|_\mathcal{Q}}[g] + (1-\Pr[E_U])\cdot\Pr_{D_N|_\mathcal{Q}}[g] \geq \Pr[E_U]\cdot\Pr_{D_P|_\mathcal{Q}}[g] > \frac{2}{3}\cdot\Pr_{D_P|_\mathcal{Q}}[g]$$

or equivalently

$$\Pr_{D_P|_\mathcal{Q}}[g] < \frac{3}{2}\Pr_{D_N|_\mathcal{Q}}[g]$$

hence the distributions D_P and D_N satisfy the conditions of Lemma 3.4.2. The following corollary completes the proof of Lemma 4.4.2.

Corollary 4.4.6. *It is not possible for any algorithm (adaptive or not) making $\sqrt{n}/4$ (or less) queries to test for isomorphism between a known graph and a graph that needs to be queried.*

The upper bound

We start with a few definitions. Given a graph G and a subset C of $V(G)$, we define the *C-labeling* of G's vertices as follows: Every vertex $v \in V(G)$ gets a label according to the set of its neighbors in C. Note that there are $2^{|C|}$ possible labels for a set C, but even if $2^{|C|} > n$ still at most n of the labels occur, since there are only n vertices in the graph. On the other hand, it is possible that several vertices will have the same label according to C. Such a labeling implies the following distribution over the vertices of G.

Definition 8. *Given a graph G and a C-labeling of its vertices (according to some $C \subset V(G)$), we denote by D_C the distribution over the actual labels of the C-labeling (at most n labels), in which the probability of a certain label γ is calculated as the number of vertices from $V(G)$ having the label γ under the C-labeling, divided by n.*

Given a graph G on n vertices and a graph C on $k < n$ vertices, we say that a one to one function $\eta : V(C) \to V(G)$ is an *embedding* of C in G . We also call $\eta(V(C))$ the *placement* of C in G. With a slight abuse of notation, from now on by a placement $\eta(V(C))$ we mean also the correspondence given by η, and not just the set.

Given graphs G, H on n vertices, a subset C_G of $V(G)$ and a placement C_H of C_G in H under an embedding η, we define the *distance* between the C_G-labeling of G and the C_H-labeling of H as

$$\frac{1}{2}\sum_{\gamma \in 2^{C_G}} \big|\, |\{u \in V(G) : \Gamma(u) \cap C_G = \gamma\}| - |\{v \in V(H) : \Gamma(v) \cap \eta(C_G) = \gamma\}| \,\big|$$

this distance measure is equal to the usual variation distance between D_{C_G} and D_{C_H}, multiplied by n. We are now ready to prove the upper bound.

Lemma 4.4.7. *Given an input graph G_u and a known graph G_k (both of order n), there is a property tester A_{ku} that accepts with probability at least $2/3$ if G_u is isomorphic to G_k, and rejects with probability at least $2/3$ if G_u is ε-far from G_k. Furthermore, A_{ku} makes $\widetilde{O}(\sqrt{n})$ queries to G_u.*

We first outline the algorithm: The test is performed in two main phases. In Phase 1 we randomly choose a small subset C_u of G_u's vertices, and try all possible placements of C_u in the known graph G_k. The placements that imply a large distance between the labeling of G_u and G_k are discarded. After filtering the good placements of C_u in G_k, we move to Phase 2. In Phase 2 every one of the good placements is tested separately, by defining a random bijection $\pi : V(G_u) \to V(G_k)$ and testing whether π is close to being an isomorphism. Finally, if one of the placements passed both Phase 1 and Phase 2, the graphs are accepted. Otherwise they are rejected.

Phase 1

In the first phase we choose at random a core set C_u of $\log^2 n$ vertices from G_u (the unknown graph). For every embedding η of C_u in G_k and the corresponding placement C_k, we examine the distributions D_{C_u} and D_{C_k} as in Definition 8. Since the graph G_k is known in advance, we know exactly which are the actual labels according to C_k (in total no more than n labels), so from now on we will consider the restriction of both distributions to these actual labels only. Next we test for every embedding of C_u whether D_{C_u} is statistically close to D_{C_k}. Note that the distribution D_{C_k} is explicitly given, and the distribution D_{C_u} can be sampled by choosing a vertex v from $V(G_u)$ uniformly at random, and making all queries $\{v\} \times C_u$. If the label of some $v \in V(G_u)$ does not exist in the C_k-labeling of G_k, we immediately reject this placement and move to the next one. Now we use the following lemma from [BFF$^+$01], which states that $\widetilde{O}(\sqrt{n})$ samples are sufficient for testing if the sampled distribution is close to the explicitly given distribution.

Lemma 4.4.8. *There is an algorithm that given two distributions D_K,D_U over n elements and a distance parameter ε, where D_K is given explicitly and D_U is given as a black box that allows sampling according to the distribution, satisfies the following: If the distributions D_K and D_U are identical, then the algorithm accepts with probability at least $1 - 2^{-\log^7 n}$; and if the variation distance between D_K and D_U is larger than $\varepsilon/10$, then the algorithm accepts with probability at most $2^{-\log^7 n}$. For a fixed ε, the algorithm uses $\widetilde{O}(\sqrt{n})$ many samples.*

Actually, this is an amplified version of the lemma from [BFF$^+$01], which can be achieved by independently repeating the algorithm provided there polylog(n) many times and taking the majority vote. This amplification allows us to reuse the same $\widetilde{O}(\sqrt{n})$ samples for all possible placements of the core set. As a conclusion of Phase 1, the algorithm rejects the placements of C_u that imply a large variation distance between the above distributions, and passes all other placements of C_u to Phase 2. Naturally, if Phase 1 rejects all placements of C_k due to distribution test failures or due to the existence of labels in G_u that do not exist in G_k, then G_u is rejected without moving to Phase 2 at all. First we observe the following.

Observation 4.4.9. *With probability $1 - o(1)$, all of the placements that passed Phase 1 imply $\varepsilon/10$-close distributions, and all placements that imply identical distributions passed Phase 1. In other words, the distribution test did not err on any of the placements.*

Proof. There are at most $2^{\log^3 n}$ possible placements of C_u. Using the union bound with Lemma 4.4.8, we conclude that Phase 1 will not err with probability $1 - o(1)$. □

Phase 2

Following Observation 4.4.9, we need to design a test such that given a placement C_k of C_u in G_k that implies close distributions, the test satisfies the following conditions:

1. If the graphs are isomorphic and the embedding of C_u is expandable to some isomorphism, then the test accepts with probability at least $3/4$
2. If the graphs G_u and G_k are ε-far, then the test accepts with probability at most $o(2^{-\log^3 n})$.

If our test in Phase 2 satisfies these conditions, then we get the desired isomorphism tester. From now on, when we refer to some placement of C_u we assume that it has passed Phase 1 and hence implies close distributions.

In Phase 2 we choose a set W_u of $\log^4 n$ vertices from $V(G_u)$, and retrieve their labels according to C_u by making the queries $W_u \times C_u$. Additionally, we split W_u into $\frac{1}{2}\log^4 n$ pairs $\{\{u_1, v_1\}, \dots, \{u_{\frac{1}{2}\log^4 n}, v_{\frac{1}{2}\log^4 n}\}\}$ randomly, and make all $\frac{1}{2}\log^4 n$ queries according to these pairs. This is done once, and the same set W_u is used for all the placements of C_u that are tested in Phase 2. Then, for every placement C_k of C_u, we would like to define a random bijection $\pi_{C_u,C_k} : V(G_u) \to V(G_k)$ as follows. For every label γ, the bijection π_{C_u,C_k} pairs the vertices of G_u having label γ with the vertices of G_k having label γ uniformly at random. There might be labels for which one of the graphs has more vertices than the other. We call these remaining vertices *leftovers*. Note that the amount of leftovers from each graph is equal to the distance between the C_k-labeling and the C_u-labeling. Finally, after π_{C_u,C_k} pairs all matching vertices, the leftover vertices are paired arbitrarily. In practice, since we do not know the labels of G_u's vertices, we instead define a partial bijection $\widetilde{\pi}_{C_u,C_k}(W_u) \to V(G_k)$ as follows. Every vertex $v \in W_u$ that has the label γ_v is paired uniformly at random with one of the vertices of G_k which has the same label γ_v and was not paired yet. If this is impossible, we reject the current placement of C_u and move to the next one.

Denote by δ_{C_u,C_k} the fraction of the queried pairs from W_u for which exactly one of $\{u_i, v_i\}$ and $\{\widetilde{\pi}_{C_u,C_k}(u_i), \widetilde{\pi}_{C_u,C_k}(v_i)\}$ is an edge. If $\delta_{C_u,C_k} \leq \varepsilon/2$, then G_u is accepted. Otherwise we move to the next placement of C_u. If none of the placements was accepted, G_u is rejected.

Correctness

A crucial observation in our proof is that with high probability, any two vertices that have many distinct neighbors in the whole graph will also have distinct neighbors within a "large enough" random core set.

Formally, given a graph G and a subset C of its vertices, we say that C is *β-separating* if for every pair of vertices $u, v \in V(G)$ such that $d_{uv} \triangleq \frac{1}{n}|\{\Gamma(u) \triangle \Gamma(v)\}| \geq \beta$ the vertices u and v have different labels under the C-labeling of G.

Claim 4.4.10. *Let $\beta > 0$ be fixed, let G be a graph of order n and let $C \subset V(G)$ be a uniformly chosen random subset of size $\log^2 n$. Then C is β-separating with probability $1 - o(1)$.*

Proof. Fix a pair $u, v \in V(G)$. If u, v are such that $d_{uv} > \beta$, then the probability that they share exactly the same neighbors in C is bounded by $(1-\beta)^{\log^2 n} \leq e^{-\beta \log^2 n} = n^{-\beta \log n}$. Using the union bound, with probability $1 - o(1)$ every pair u, v of vertices with $d_{uv} > \beta$ will not have exactly the same neighbors in C, i.e. the vertices will have different labels under the C-labeling. $\square$

Lemma 4.4.11 (completeness)**.** *Conditioned over the event that C_u is $\varepsilon/8$-separating, if the graphs G_u and G_k are isomorphic and the placement C_k of C_u is expandable to some isomorphism, then* $\Pr[\delta_{C_u,C_k} \leq \varepsilon/2] = 1 - o(1)$*, and hence C_k is accepted in Phase 2 with probability* $1 - o(1)$*.*

Proof. Let $\phi : V(G_u) \to V(G_k)$ be an isomorphism to which the placement of C_u is expandable. By definition, for every pair v_1, v_2 of G_u's vertices, $\{v_1, v_2\}$ is an edge in G_u if and only if $\{\phi(v_1), \phi(v_2)\}$ is an edge in G_k. In addition, for every vertex $v \in V(G_u)$, the vertices v and $\phi(v)$ have exactly the same labels. Let σ be the permutation, such that π_{C_u,C_k} is the composition of σ and the isomorphism ϕ. In the rest of this proof, by distance we mean the absolute distance between two labeled graphs (which is between 0 and $\binom{n}{2}$).

First we show that the distance from $\sigma(G_u)$ to G_k is almost the same as the distance from $\phi(G_u)$ to G_k (which is zero since ϕ is an isomorphism), and then we apply large deviation inequalities to conclude that $\Pr[\delta_{C_u,C_k} \leq \varepsilon/2] = 1 - o(1)$.

To prove that the distance from $\sigma(G_u)$ to G_k is close to zero we show a transformation of ϕ into π_{C_u,C_k} by performing "swaps" between vertices that have the same label. Namely, we define a sequence of permutations ϕ_i, starting from $\phi_0 = \phi$, and ending with $\phi_t = \pi_{C_u,C_k}$. In each step, if there is some vertex v_0 such that $\phi_i(v_0) = u_1$ while $\pi_{C_u,C_k}(v_0) = u_0$, then we find a vertex v_1 such that $\phi_i(v_1) = u_0$, and set $\phi_{i+1}(v_0) = u_0$ and $\phi_{i+1}(v_1) = u_1$. The rest of the vertices are mapped by ϕ_{i+1} as they were mapped by ϕ_i.

Since in each step we only swap a pair of vertices with the same label, and since the core set C_u is $\varepsilon/8$-separating, every such swap can increase the distance by at most $\varepsilon n/8$, so eventually the distance between $\sigma(G_u)$ and G_k is at most $\varepsilon n^2/8$. Therefore, by large deviation inequalities, δ_{C_u,C_k} as defined in Phase 2 is at most $\varepsilon/2$ with probability $1-o(1)$, and so the placement C_k is accepted.

□

We now turn to the case where G_u and G_k are ε-far. Note that until now we did not use the fact that C_u and C_k imply close distributions. To understand why this closeness is important, recall the pairs of graphs from the lower bound proof. If we give up the distribution test in Phase 1, then these graphs will be accepted with high probability, since the algorithm cannot reveal two copies of the same vertex when sampling $o(\sqrt{n})$ vertices (recall that $|W_u| = O(\log^4 n)$). Intuitively, the problem is that in these pairs of graphs, the partial random bijection $\widetilde{\pi}_{C_u,C_k}$ will not simulate a restriction of the random bijection π_{C_u,C_k} to a set of $\log^4 n$ vertices. In the lower bound example, $\widetilde{\pi}_{C_u,C_k}$ will have no leftovers with high probability, even though π_{C_u,C_k} will always have $\Omega(n)$ leftovers. The reason is that in the cloned graph G_u, for each of about half of the labels from C_k there are two times more vertices, while for the second half there are no vertices at all. The distribution test in Phase 1 actually checks whether the clustering of the vertices according to the labels is into subsets of almost equal sizes in both G_u and G_k. If it is so, then the partial random bijection $\widetilde{\pi}_{C_u,C_k}$ is indeed similar to the restriction of a bijection π_{C_u,C_k} to a set of $\log^4 n$ vertices.

Lemma 4.4.12 (soundness)**.** *If the graphs G_u and G_k are ε-far, and the placement C_k implies $\varepsilon/10$-close distributions, then* $\Pr[\delta_{C_u,C_k} \le \varepsilon/2] \le o(2^{-\log^3 n})$*, and hence C_k is accepted in Phase 2 with probability at most $o(2^{-\log^3 n})$.*

Proof. Assume that for a fixed C_k the random bijection π_{C_u,C_k} is ε-far from isomorphism. We then need to show that δ_{C_u,C_k} as defined in Phase 2 is larger than $\varepsilon/2$ with probability $1 - o(2^{-\log^3 n})$.

Since the variation distance between the distributions D_{C_u} and D_{C_k} is at most $\varepsilon/10$, the amount of leftovers (which is exactly the distance between the C_u-labeling of G_u and the C_k-labeling of G_k) is at most $\varepsilon n/10$. Therefore, even if we first remove those $\varepsilon n/10$ (or less) leftovers, the fraction of pairs u, v for which exactly one of $\{u, v\}$ and $\{\widetilde{\pi}_{C_u,C_k}(u), \widetilde{\pi}_{C_u,C_k}(v)\}$ is an edge is not smaller by more than $4\varepsilon/10$ from that of π_{C_u,C_k}.

Let $\widetilde{\pi}_{C_u,C_k}$ be the random partial bijection as defined above. The distribution test of Phase 1 guaranties that $\widetilde{\pi}_{C_u,C_k}$ is a random restriction of a function that is $\varepsilon/10$-close to some bijection π_{C_u,C_k}. Since G_u is ε-far from G_k, the bijection π_{C_u,C_k} must be ε-far from being an isomorphism, and hence $\widetilde{\pi}_{C_u,C_k}$ must exhibit a $6\varepsilon/10$-fraction of mismatching edges. Note that the acceptance probability of C_k given $\widetilde{\pi}_{C_u,C_k}$ is equal to the probability that δ_{C_u,C_k} as defined in Phase 2 is at most $\varepsilon/2$. Large deviation inequalities show that this probability is at most $2^{-\Omega(\log^4 n)} = o(2^{-\log^3 n})$. □

As a conclusion, if G_k and G_u are isomorphic, then the probability that C_u is not $\varepsilon/8$-separating is at most $o(1)$, and for a correct (under some isomorphism) embedding of

C_u in G_k, the probability that the distribution test will fail is also $o(1)$, so in summary algorithm A_{ku} accepts with probability greater than $2/3$. In the case that G_k and G_u are ε-far from being isomorphic, with probability $1 - o(1)$ all placements that are passed to Phase 2 imply close label distributions. Then each such placement is rejected in Phase 2 with probability $1 - o(2^{-\log^3 n})$, and by the union bound over all possible placements the graphs are accepted with probability less than $1/3$. Algorithm A_{ku} makes $\widetilde{O}(\sqrt{n})$ queries in Phase 1 and $\widetilde{O}(n^{1/4})$ queries in Phase 2. This completes the proof of Lemma 4.4.7 and so of Theorem 4.4.1.

4.4.2 Two-sided testing of two unknown graphs

Theorem 4.4.13. *The query complexity of two-sided error isomorphism testers is between $\Omega(n)$ and $\widetilde{O}(n^{5/4})$ if both graphs need to be queried.*

The upper bound

Lemma 4.4.14. *Given two unknown graphs G and H on n vertices, there is a property tester A_{uu} that accepts with probability at least $2/3$ if G is isomorphic to H, and rejects with probability at least $2/3$ if G is ε-far from H. Furthermore, A_{uu} makes $\widetilde{O}(n^{5/4})$ queries to G and H.*

We use here ideas similar to those used in the upper bound proof of Lemma 4.4.7, but with several modifications. The main difference between this case and the case where one of the graphs is known in advance is that here we cannot write all label distributions with all possible core sets in either one of the unknown graphs (because doing that would require $\Omega(n^2)$ queries). We overcome this difficulty by sampling from both graphs in a way that with high probability will make it possible to essentially simulate the test for isomorphism where one of the graphs is known in advance.

Phase 1

First we randomly pick a set U_G of $n^{1/4}\log^3(n)$ vertices from G, and a set U_H of $n^{3/4}\log^3(n)$ vertices from H. Then we make all $n^{5/4}\log^3(n)$ possible queries in $U_G \times V(G)$. Note that if G and H have an isomorphism σ, then according to Lemma 4.2.2 with probability $1-o(1)$ the size of $U_G \cap \sigma(U_H)$ will exceed $\log^2(n)$.

For all subsets C_G of U_G of size $\log^2 n$ we try every possible placement $C_H \subset U_H$ of C_G. There are at most $2^{\log^3 n}$ subsets C_G, and at most $2^{\log^3 n}$ possible ways to embed each C_G in U_H. Since we made all $n^{5/4}\log^3(n)$ possible queries in $U_G \times V(G)$, for every $C_G \subset U_G$ the corresponding distribution D_{C_G} is entirely known.

Now for every possible placement of C_G in U_H, we test if the variation distance between the distributions D_{C_G} and D_{C_H} is at most $\varepsilon/10$. Since we know the entire distribution

D_{C_G}, we only need to sample the distribution D_{C_H}, and therefore we can still use the amplified distribution test of Lemma 4.4.8. The test there requires $\widetilde{O}(\sqrt{n})$ samples, so similarly to the proof of Lemma 4.4.7 we take a random set S of $\widetilde{O}(\sqrt{n})$ vertices from H and make all $n^{5/4}\text{polylog}(n)$ queries in $S \times U_H$.

We reject the pairs of a set C_G and a placement C_H that were rejected by the distribution test for D_{C_G} and D_{C_H}, and pass all other pairs to Phase 2. If Phase 1 rejects all possible pairs, then the graphs G and H are rejected without moving to Phase 2. The following observation is similar to the one we used in the case where one of the graphs is known in advance.

Observation 4.4.15. *With probability $1 - o(1)$, all of the placements that passed Phase 1 imply $\varepsilon/10$-close distributions, and all placements that imply identical distributions passed Phase 1. In other words, the distribution test did not err on any of the placements.*

Phase 2

As in Lemma 4.4.7, we need to design a test which given a placement C_H of C_G in H that implies close distributions, satisfies the following conditions:

1. If the graphs are isomorphic and the embedding of C_H is expandable to some isomorphism, then the test accepts with probability at least 3/4
2. If the graphs G and H are ε-far, then the test accepts with probability at most $o(2^{-2\log^3 n})$.

In Phase 2 we choose at random a set W_G of $n^{1/2}\log^{13} n$ vertices from $V(G)$, and a set W_H of $n^{1/2}\log^{6} n$ vertices from $V(H)$. We retrieve the labels in W_H according to any C_H by making the queries $W_H \times U_H$. Additionally, we make all queries inside W_H and all queries inside W_G. This is done once, and the same sets W_G, W_H are used for all of the pairs C_G, C_H that are tested in Phase 2. According to Lemma 4.2.2, if the graphs are isomorphic under some isomorphism σ, then $|W_H \cap \sigma(W_G)| > \log^7 n$ with probability $1 - o(1)$.

Then, similarly to what is done in Lemma 4.4.7, for every pair C_G, C_H, we would like to define a random bijection $\pi_{C_G,C_H} : V(G) \to V(H)$ as follows. For every label γ, π_{C_G,C_H} pairs the vertices of G having label γ with the vertices of H having label γ uniformly at random. After π_{C_G,C_H} pairs all matching vertices, the leftover vertices are paired arbitrarily. Then again, since we do not know the labels of H's vertices, we define a partial bijection $\widetilde{\pi}_{C_G,C_H}(W_H) \to V(G)$ instead, in which every vertex $v \in W_H$ that has the label γ_v is paired uniformly at random with one of the vertices of G which has the same label γ_v and was not paired yet. If this is impossible, we reject the current pair C_G, C_H and move to the next one.

Denote by I_H the set $\widetilde{\pi}_{C_G,C_H}(W_H) \cap W_G$, and denote by S_H the set $\widetilde{\pi}^{-1}_{C_G,C_H}(I_H)$. According to Lemma 4.2.2, $|I_H| > \log^7 n$ with probability $1 - o(2^{-\log^6 n})$, that is, with probability $1 - o(1)$ we have $|I_H| > \log^7 n$ for every pair C_G, C_H (if this is not the case, we terminate the algorithm and answer arbitrarily). Next we take $\frac{1}{2}\log^7 n$ pairs $\{\{u_1, v_1\}, \dots, \{u_{\frac{1}{2}\log^7 n}, v_{\frac{1}{2}\log^7 n}\}\}$ randomly from S_H, and denote by δ_{C_G,C_H} the fraction of S_H's pairs for which exactly one of $\{u_i, v_i\}$ and $\{\widetilde{\pi}_{C_G,C_H}(u_i), \widetilde{\pi}_{C_G,C_H}(v_i)\}$ is an edge. If $\delta_{C_G,C_H} \le \varepsilon/2$, then the graphs are accepted. Otherwise we move to the next pair C_G, C_H. If none of the pairs are accepted, then the graphs are rejected.

As noted above, if G and H are isomorphic, then according to Lemma 4.2.2 with probability $1-o(1)$ the size of $U_G \cap \sigma(U_H)$ is at least $\log^2(n)$. Therefore with probability $1-o(1)$, for some pair C_H, C_G the placement C_H of C_G is expandable to an isomorphism. We now need to show that in this case the pair C_H, C_G is accepted with sufficient probability.

Lemma 4.4.16 (completeness). *If the graphs G and H are isomorphic and σ is an isomorphism between them, then with probability at least $3/4$ there exists $C_G \subset U_G$ with a placement $C_H \subset U_H$ which is expandable to σ, and for which $\delta_{C_G,C_H} \le \varepsilon/2$.*

Proof sketch. First we look at the set $\Delta = U_G \cap \sigma^{-1}(U_H)$. By Lemma 4.2.2 the size of Δ is at least $\log^2 n$ with probability $1 - o(1)$. Conditioned on this event, we pick $C_G \subseteq \Delta \subseteq U_G$ uniformly from all subsets of Δ with size $\log^2 n$, and set $C_H = \sigma(C_G)$ to be its placement in U_H. We now prove that conditioned on the event that Δ is large enough, C_G and C_H will be as required with probability $1 - o(1)$.

Our main observation is that if we condition only on the event that Δ is large enough, then C_G is distributed uniformly among all subsets with this size of $V(G)$, so we proceed similarly to the case where one of the graphs is known in advance. We observe that if two vertices have many distinct neighbors, then with high probability they will not share exactly the same neighbors within a random core set of size $\log^2 n$ (see Lemma 4.4.10), so C_G has a separating property. When this happens, it is possible to switch between the vertices with identical labels and still retain a small enough bound on δ_{C_G,C_H}. $\square$

Lemma 4.4.17 (soundness). *If the graphs G and H are ε-far, and the pair C_G, C_H implies close distributions, then $\Pr[\delta_{C_G,C_H} \le \varepsilon/2] \le o(2^{-\log^6 n})$, and hence the pair C_G, C_H is accepted in Phase 2 with probability at most $o(2^{-\log^6 n})$.*

Proof sketch. As before, assume that for a fixed pair C_G, C_H the random bijection π_{C_G,C_H} is ε-far from isomorphism. We then need to show that δ_{C_G,C_H} as defined in Phase 2 is at most $\varepsilon/2$ with probability only $o(2^{-\log^6 n})$.

Since the variation distance between the distributions D_{C_G} and D_{C_H} is at most $\varepsilon/10$, the amount of leftovers (which is exactly the distance between the C_G-labeling and the C_H-labeling) is at most $\varepsilon n/10$. After removing those $\varepsilon n/10$ (or less) leftovers, the fraction of pairs u, v for which exactly one of $\{u, v\}$ and $\{\widetilde{\pi}_{C_G,C_H}(u), \widetilde{\pi}_{C_G,C_H}(v)\}$ is an edge is still not smaller than that of π_{C_G,C_H} by more than $4\varepsilon/10$. Now the distribution test of Phase 1 guaranties that $\widetilde{\pi}_{C_G,C_H}$ is $\varepsilon/10$-close to the restriction of some random bijection π_{C_G,C_H}. Since the graph G is ε-far from being isomorphic to the graph H, the bijection π_{C_G,C_H} must be ε-far from an isomorphism, and hence $\widetilde{\pi}_{C_G,C_H}$ must exhibit a $6\varepsilon/10$-fraction of incompatible edges, while the acceptance probability of the pair C_G, C_H given $\widetilde{\pi}_{C_G,C_H}$ is equal to the probability that δ_{C_G,C_H} as defined in Phase 2 is at most $\varepsilon/2$. Applying large deviation inequalities shows that this probability is at most $2^{-\Omega(\log^7 n)} = o(2^{-\log^6 n})$. $\square$

The isomorphism testing algorithm A_{uu} makes $\widetilde{O}(n^{5/4})$ queries in total, completing the proof of Theorem 4.4.13.

The lower bound

A lower bound of $\Omega(n)$ queries is implicitly stated in [Fis04] following [AFKS00]. Here we provide the detailed proof for completeness.

Lemma 4.4.18. *Any adaptive (as well as non-adaptive) testing algorithm that makes at most $\frac{n}{4}$ queries cannot distinguish between the case that the unknown input graphs G and H are isomorphic, and the case that they are $\frac{1}{8}$-far from being isomorphic.*

Proof. We construct two distributions over pairs of graphs. The distribution D_P is constructed by letting the pair of graphs consist of a random graph $G \sim G(n, 1/2)$ and a graph H that is a random permutation of G. The distribution D_N is constructed by letting the pair of graphs consist of two independently chosen random graphs $G, H \sim G(n, 1/2)$.

Clearly D_P satisfies the property with probability 1. By large deviation inequalities, it is also clear that in an input chosen according to D_N, the graphs G and H are $\frac{1}{8}$-far with probability $1 - 2^{\Omega(n^2)}$. The next step is to replace D_N with D'_N, in which the graphs are $\frac{1}{8}$-far from being isomorphic with probability 1. We just set D'_N to be the distribution that results from conditioning D_N on the event that G is indeed $\frac{1}{8}$-far from H.

We now consider any fixed set $Q = \{p_1, \ldots, p_{\frac{n}{4}}\}$ of vertex pairs, some from the first graph, and others from the second graph. For an input chosen according to the distribution D_N, the values of these pairs (the answers for corresponding queries) are $\frac{n}{4}$ uniformly and independently chosen random bits. We now analyze the distribution D_P. Let $e_1, \ldots, e_k$ and $f_1, \ldots, f_l$ be all vertex pairs of the first and the second graph respectively, that appear in Q. Clearly $k, l \leq |Q| = \frac{n}{4}$. Let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be the permutation according to which the second graph is chosen in D_P. Let E denote the event that $\sigma(e_i) \neq f_j$ for every $1 \leq i \leq k$ and $1 \leq j \leq l$, where for $e = \{u, v\}$ we denote by $\sigma(e)$ the pair $\{\sigma(u), \sigma(v)\}$. Clearly, if E occurs then $\{p_1, \ldots, p_{\frac{n}{4}}\}$ will be a set of $\frac{n}{4}$ uniformly and independently chosen random bits.

Claim 4.4.19. *The event E as defined above occurs with probability at least* $3/4$.

Proof. For a single pair e_i and a random permutation σ, the probability that $e_i = \sigma(f_j)$ for some $1 \leq j \leq l$ is bounded by $\frac{n}{2\binom{n}{2}}$. Hence by the union bound, $\Pr[E] \geq 1 - \frac{kn}{2\binom{n}{2}} > 3/4$. $\square$

Since E occurs with probability at least $3/4$, and since the event upon which we conditioned D_N to get D'_N occurs with probability $1 - 2^{-\Omega(n^2)} = 1 - o(2^{-|\mathcal{Q}|})$, we get that for any $g : \mathcal{Q} \to \{0,1\}$, we have $\Pr_{D'_N|\mathcal{Q}}[g] < \frac{3}{2}\Pr_{D_P|\mathcal{Q}}[g]$ and therefore the distributions D_P and D'_N satisfy the conditions of of Lemma 3.4.2.

$\square$

4.5 Future work

While our two-sided error algorithms run in time quasi-polynomial in n (like the general approximation algorithm of [AFK96]), the one-sided algorithms presented here require an exponential running time. It would be interesting to reduce the running time of the one-sided algorithms to be quasi-polynomial while still keeping them one-sided.

Another issue goes back to [AFKS00]. There, the graph isomorphism question was used to prove that certain first order graph properties are impossible to test with a constant number of queries. However, in view of the situation with graph isomorphism, the question now is whether every first order graph property is testable with $O(n^{2-\alpha})$ many queries for some $\alpha > 0$ that depends on the property to be tested.

Finally, it would be interesting to close the remaining gap between $\Omega(n)$ and $\widetilde{O}(n^{5/4})$ in the setting of two graphs that need to be queried, and a two-sided error algorithm. It appears (with the aid of martingale analysis on the same distributions D_P, D_N as above) that at least for non-adaptive algorithms the lower bound can be increased a little to a bound of the form $\Omega(n \log^{\alpha} n)$, but we are currently unable to give tighter bounds on the power of n.

Chapter 5

Approximate hypergraph partitioning and applications

In this chapter we show that any *partition-problem* of hypergraphs has an $O(n)$ time approximate partitioning algorithm and an efficient property tester. This extends the results of Goldreich, Goldwasser and Ron who obtained similar algorithms for the special case of graph partition problems in their seminal paper [GGR98].

The partitioning algorithm is used to obtain the following results:

- We derive a surprisingly simple $O(n)$ time algorithmic version of Szemerédi's regularity lemma. Unlike all the previous approaches for this problem [AN06, DLR95, FK99b, FK96, KRT03], which only guaranteed to find partitions of tower-size, our algorithm will find a small regular partition in the case that one exists.

- For any $r \geq 3$, we give an $O(n)$ time randomized algorithm for constructing regular partitions of r-uniform hypergraphs, thus improving the previous $O(n^{2r-1})$ time (deterministic) algorithms [CR00, FK96].

The property testing algorithm is used to unify several previous results, and to obtain for a fixed partition size the partition densities for the above problems (rather than the partitions themselves) using only poly$(1/\varepsilon)$ queries and constant running time.

5.1 Background and introduction

Graph partition problems are some of the most well-studied problems both in graph theory and in computer-science. Standard examples of partition problems include k-colorability, Max-Clique and Max-Cut. Most of these problems are computationally hard even to approximate, but it was observed in the 90's [AKK99, dlV96] that many of these partition problems have good approximations when the input graph is dense. In this chapter we introduce an efficient $O(n)$ algorithm for partitioning *hypergraphs*, with an accompanying $O(1)$ query complexity test for the existence of a partition with given parameters. We show that several previous results on graph and hypergraph partition problems follow as special cases of our main result. In some cases the our results will actually improve upon the previously known ones.

Our framework for studying hypergraph partition problems generalizes the framework of graph partition problems that was introduced by Goldreich, Goldwasser and Ron [GGR98]. Let us briefly discuss the graph partitioning algorithm of [GGR98]. A graph *partition-instance* Ψ is composed of an integer k specifying the number of sets in the required partition $V_1, \ldots, V_k$ of the graph's vertex set, and intervals specifying the allowed ranges for the number of vertices in every V_i and the number of edges between every V_i and V_j for $i \leq j$.

Goldreich, Goldwasser and Ron [GGR98] showed that for any partition-instance Ψ with k parts, and for any fixed ε, there is an $O(2^{(k/\varepsilon)^{O(k)}} + (k/\varepsilon)^{O(k)}n)$ time algorithm that produces a partition of an input graph that is ε-close to satisfying Ψ, assuming that a satisfying partition exists (the distance is measured by the differences between the actual densities and the required ones). Note that one can formulate many problems, such as k-colorability, Max-Cut and Max-Clique, in this framework of partition-instances. Therefore, the algorithm of [GGR98], which we will henceforth refer to as the *GGR-algorithm*, implies for example that there is an $O(\exp(1/\varepsilon^3) + (1/\varepsilon^2)n)$ time algorithm that approximates the size of the maximum cut of a graph to within an additive error of εn^2. It also implies that for any $\varepsilon > 0$ there is an $O(\exp(1/\varepsilon^3) + (1/\varepsilon^2)n)$ time algorithm that, given a graph G, either reports that G is not 3-colorable, or 3-colors the vertices of G in such a way that all but at most εn^2 of the edges are properly colored.

The second main result of [GGR98] was that in fact, for any partition problem Ψ as above, there is a randomized algorithm making only $(k/\varepsilon)^{O(k)}$ many queries to the input graph, and running in time $2^{(k/\varepsilon)^{O(k)}}$, which distinguishes with probability $\frac{2}{3}$ between graphs satisfying Ψ and graphs that are ε-far from satisfying Ψ. Thus the result of [GGR98] implies that one can distinguish in *constant time* between graphs satisfying a partition instance and graphs that are far from satisfying it.

Here we extend the main results of [GGR98] to the case of hypergraphs by showing that there is a randomized algorithm for the general hypergraph partition-problem. The running time of our algorithm is $O(n)$ (where n is the number of vertices, making this sublinear) and it has the following property: Given an input hypergraph H, which satisfies the partition problem, the algorithm produces a partition of H that is “close” to satisfying it. In the case that no such partition of H exists, the algorithm rejects the input. We also obtain property testing algorithms for such problems making only poly$(1/\varepsilon)$ queries, and with a constant running time.

We present several applications of our result, and in the foremost a new application related to Szemerédi’s regularity lemma [Sze78]. By using an appropriate hypergraph modeling of the problem we design a surprisingly simple $O(n)$ time algorithm for constructing regular partitions of graphs. An added benefit is that unlike the previous approaches for constructing regular partitions [AN06, DLR95, FK99b, FK96, KRT03], which proved the lemma “algorithmically”, our algorithm will find a small regular partition in the case that one exists in our input graph, rather than being guaranteed to find only a partition of the tower-size upper bound given by Szemerédi’s lemma itself.

We also design an $O(n)$ time randomized algorithm for constructing regular partitions of r-uniform hypergraphs for any $r \geq 3$. This improves over the previous (deterministic) algorithms [CR00, FK96] that have a running time of $O(n^{2r-1})$.

Our property testing algorithm provides a common generalization for many previously known results. We show how special cases of the now-testable partition problem can be easily used to derive some results that were previously proved using specialized methods, namely testing properties of hypergraphs [CS05, Lan04] and estimating k-CNF satisfiability [AVKK03].

5.2 Extension of the GGR-algorithm

Our main result in this chapter is a generalization of the GGR-algorithm to the case of hypergraphs. Let us start by defining our framework for studying hypergraph partition problems. We consider directed hypergraphs $H = (V, E_1, E_2, \dots, E_s)$ with n vertices and s directed edge sets (the generalization to several edge sets also has a role in our main application). Every edge set $E_i(H) \subseteq V^{r_i}$ is a set of *ordered* r_i-tuples (ordered sets with possible repetitions) over V. [1] That is, for every edge set $E_i(H)$ we think of any of the edges $e \in E_i$ as an ordered subset of $V(H)$ of size r_i. Let us put $r = \max_i\{r_i\}$. Note that the usual notion of a directed graph corresponds to hypergraphs with only one edge set E whose edges are of size 2. For an r_i-tuple $e = (v_1, \dots, v_{r_i}) \in E_i$ we say that v_j is in the j^{th} *place* of e (or is simply the j^{th} *vertex* of e). We use $[k]$ to denote the set $\{1, \dots, k\}$, and we use $a = b \pm c$ as a shorthand for $b - c \leq a \leq b + c$.

[1] The logicians among the readers should recognize these as arity r_i relations.

5.2.1 The partition property

Let H be a hypergraph as above and let $\Pi = \{V_1^\Pi, \dots, V_k^\Pi\}$ be a partition of $V(H)$. Let us introduce a notation for counting the number of edges of $E_i(H)$ with a specific placement of their vertices within the partition classes of Π (remember that the edges are ordered). For every $i \in [s]$ we denote by Φ_i the set of all possible mappings $\phi : [r_i] \to [k]$. We think of every $\phi \in \Phi_i$ as mapping the vertices of an r_i-tuple to the components of Π. We denote by $E_{i,\phi}^\Pi \subseteq E_i$ the following set of r_i-tuples:

$$E_{i,\phi}^\Pi = \{(v_1, \dots, v_{r_i}) \in E_i : v_j \in V_{\phi(j)}^\Pi,\ \forall\ 1 \le j \le r_i\}$$

We now introduce a notion that generalizes the partition instances of graphs that were discussed earlier in the context of graphs. A *density tensor* is a sequence $\psi = \left\langle \langle \rho_j \rangle_{j \in [k]}, \langle \mu_{i,\phi} \rangle_{i \in [s], \phi \in \Phi_i} \right\rangle$ of reals (between 0 and 1), specifying the presumed normalized sizes of $|V_i^\Pi|$ and $|E_{i,\phi}^\Pi|$ of a partition of a hypergraph H. In particular, given a partition $\Pi = \{V_1^\Pi, V_2^\Pi, \dots, V_k^\Pi\}$ of a hypergraph H, we set ψ^Π to be the density tensor $\left\langle \langle \rho_j^\Pi \rangle_{j \in [k]}, \langle \mu_{i,\phi}^\Pi \rangle_{i \in [s], \phi \in \Phi_i} \right\rangle$ with the property that for all j, $\rho_j^\Pi = \frac{1}{n} \cdot |V_j^\Pi|$ and for all i and ϕ, $\mu_{i,\phi}^\Pi = \frac{1}{n^{r_i}} \cdot |E_{i,\phi}^\Pi|$.

For a fixed hypergraph H, a set Ψ of general density tensors (with respect to k, s and $r_1, \dots, r_s$) defines a property of $V(H)$'s partitions as follows. We say that a partition Π of $V(H)$ (exactly) *satisfies* Ψ if there exists a density tensor $\psi = \left\langle \langle \rho_j \rangle_{j \in [k]}, \langle \mu_{i,\phi} \rangle_{i \in [s], \phi \in \Phi_i} \right\rangle \in \Psi$, such that ψ and the density tensor ψ^Π of Π are equal. Namely, the following equalities hold: for all $j \in [k]$, $\quad \rho_j^\Pi = \rho_j$; and for all $i \in [s]$ and $\phi \in \Phi_i$, $\quad \mu_{i,\phi}^\Pi = \mu_{i,\phi}$.

We say that Π ε-*closely* satisfies Ψ if there exists $\psi \in \Psi$, such that for all $j \in [k]$ $\rho_j^\Pi = \rho_j$ and for all $i \in [s]$ and $\phi \in \Phi_i$ $\mu_{i,\phi}^\Pi = \mu_{i,\phi} \pm \varepsilon$. In addition, we say that Π ε-*approximately* satisfies Ψ if there exists $\psi \in \Psi$ such that for all $j \in [k]$ $\quad \rho_j^\Pi = \rho_j \pm \varepsilon$ and for all $i \in [s]$ and $\phi \in \Phi_i$ $\quad \mu_{i,\phi}^\Pi = \mu_{i,\phi} \pm \varepsilon$.

By extension (and with a slight abuse of notation), we say that the hypergraph H itself *satisfies* the property Ψ if there exists a partition Π of H's vertices that satisfies Ψ, and similarly we say that H itself ε-*closely* (respectively ε-*approximately*) satisfies the property Ψ if there exists a partition of H's vertices that ε-closely (respectively ε-approximately) satisfies the property Ψ. In addition, we may refer to a specific density tensor ψ as the singleton set $\{\psi\}$, and accordingly consider it as a property of partitions.

In the following we make some computational assumptions on the representation of the considered set Ψ, which may be infinite. In particular, we assume that given a density tensor ψ, computing whether ψ is close to some $\psi' \in \Psi$ can be done efficiently.

Definition 9. *For a set Ψ of density tensors, we say that Ψ* is checkable for proximity in time $O(t)$, *or shortly $TC(\Psi) = O(t)$, if there exists an algorithm that for any density tensor ψ and any $\varepsilon > 0$ decides in time at most $O(t)$ whether the tensor ψ ε-approximately* *satisfies Ψ, and if so, outputs a density tensor $\psi_T \in \Psi$ which is ε-approximated by ψ.*

We say that a set Ψ of density tensors is efficiently checkable for proximity *if $TC(\Psi)$ is bounded by some polynomial in s and k^r.*

Note that the sets Ψ resulting from upper and lower bound constraints as in [GGR98] are indeed efficiently checkable for proximity. For instance, given a density tensor $\psi = \left\langle \langle \rho_j \rangle_{j\in[k]}, \langle \mu_{i,\phi} \rangle_{i\in[s],\phi\in\Phi_i} \right\rangle$, one can verify in time $O(k + s \cdot k^r)$ whether ψ ε-approximately satisfies Ψ by going over the $k + s \cdot k^r$ values of the parameters of ψ, and checking if all of these values satisfy the lower and upper bounds within the allowed deviation of ε. It is easy to verify that the sets of density tensors defined in the proofs of Theorem 5.4.2 and Theorem 5.5.1 below are also efficiently checkable for proximity.

Also, we assume throughout that a uniformly random choice of a vertex $v \in V$, as well as an edge query, can all be done in constant time. In the following we make no attempt to optimize the coefficients, only the function types.

We are now ready to state the main technical theorem of this chapter.

Theorem 5.2.1 (Hypergraph Partitioning Algorithm)**.** *Let $H = (V, E_1, \dots, E_s)$ be a hypergraph with n vertices and s edge sets of maximal size r, let Ψ be an efficiently checkable set of density tensors, of partitions into k sets, and let $\varepsilon > 0$ and $\delta > 0$ be fixed constants. There is a randomized algorithm A such that*

- *If H satisfies Ψ then with probability at least $1-\delta$ the algorithm A outputs a partition of $V(H)$ that ε-closely satisfies Ψ.*
- *For every H, the algorithm A outputs a partition of $V(H)$ that* does not *ε-closely satisfies Ψ with probability at most δ. In particular, if H does not ε-closely satisfy Ψ, then the algorithm returns "No Partition" with probability at least $1-\delta$.*

Furthermore, the running time of A is bounded by $\log^2(\frac{1}{\delta}) \cdot \exp\left((\frac{r}{\varepsilon})^{O(s\cdot r\cdot k^r)}\right) + n \cdot \mathrm{poly}(k^r, s, 1/\varepsilon)$.

Observe that the above algorithm has only poly-logarithmic dependence on the success probability δ, a fact that will be important in the applications of Theorem 5.2.1 we discuss later.

Let us now discuss the property testing variant of Theorem 5.2.1. To this end, we define one additional measure of closeness to the property Ψ for the property testing context. Let $H = (V, E_1, \dots E_s)$ and $H' = (V, E'_1, \dots E'_s)$ be two hypergraphs over the same vertex set V, with matching arities of edge sets (i.e. for all $i \in [s]$, $r_i = r'_i$). Let $\Delta(E_i, E'_i)$ denote the size of the symmetric difference between E_i and E'_i. The distance between H and H' is defined as $dist(H, H') = \frac{1}{s}\sum_{i\in[s]} \Delta(E_i, E'_i)/n^{r_i}$. The distance of a hypergraph H from the property Ψ is defined as $dist(H, \Psi) = \min_{H'}\{dist(H, H') : H' \text{ satisfies } \Psi\}$. For

$\varepsilon > 0$ we say that H is *ε-far* from satisfying the property Ψ when $dist(H, \Psi) > \varepsilon$, and otherwise, we say that H is *ε-close* to Ψ. The testing algorithm follows immediately from the following.

Theorem 5.2.2 (Testing)**.** *Let $H = (V, E_1, \ldots, E_s)$ be a hypergraph with n vertices and s edge sets of maximal size r, let Ψ be an efficiently checkable set of density tensors, of partitions into k sets, and let $\varepsilon > 0$ and $\delta > 0$ be fixed constants. There is a randomized algorithm A_T such that*

- *If H satisfies Ψ then with probability at least $1-\delta$ the algorithm A_T outputs* Accept*, and in addition provides a density tensor $\psi_T \in \Psi$ such that ψ_T is ε-closely satisfied by H.*
- *If H does not even ε-closely satisfy the property Ψ then with probability at least $1-\delta$ the algorithm A_T outputs* Reject*.*

The query complexity of the algorithm A_T is bounded by $\log^2(\frac{1}{\delta}) \cdot \mathrm{poly}(k^r, s, 1/\varepsilon)$, and its running time is bounded by $\log^2(\frac{1}{\delta}) \cdot \exp\left((\frac{r}{\varepsilon})^{O(s \cdot r \cdot k^r)}\right)$.

To see why the algorithm above can be used as a testing algorithm in the traditional sense, just note that a hypergraph that ε/k^r-closely satisfies a property is clearly ε-close to it. Note that the GGR-algorithm [GGR98] follows from Theorem 5.2.1 by setting $r = 2$ and $s = 1$ in the statement of the theorem. Similarly, the property testing algorithm of [GGR98] follows easily from Theorem 5.2.2.

5.3 Immediate applications of the theorems

Just as the GGR-algorithm [GGR98] and its testing variant can be used to give $O(n)$ time algorithms and testing algorithms for graph partition problems like Max-Cut, k-colorability and Max-Clique, so can Theorems 5.2.1 and 5.2.2 be used to give analogous algorithms for the corresponding problems in hypergraphs. For example, consider the following immediate corollary of Theorems 5.2.1 and 5.2.2.

Corollary 5.3.1. *For any fixed r and c, the following algorithms immediately follow from using Theorems 5.2.1 and 5.2.2 for the corresponding special cases:*

- *An ε-test of an n-vertex r-uniform (simple) hypergraph for the property of being c-colorable (with no monochromatic edges), making $\mathrm{poly}(1/\varepsilon)$ queries and taking $\exp(\mathrm{poly}(1/\varepsilon))$ time, and an accompanying approximate coloring algorithm taking $\exp(\mathrm{poly}(1/\varepsilon)) + \mathrm{poly}(1/\varepsilon)n$ time.*

- *An ε-test of an n-vertex r-uniform hypergraph for the property of having an independent set of size at least αn (for any fixed α), making* $\mathrm{poly}(1/\varepsilon)$ *queries and taking* $\exp(\mathrm{poly}(1/\varepsilon))$ *time, and an accompanying approximate algorithm taking* $\exp(\mathrm{poly}(1/\varepsilon)) + \mathrm{poly}(1/\varepsilon)n$ *time and finding a set of size αn spanning less than εn^r edges.*

- *An algorithm for approximating the maximum r-way cut of an n-vertex r-uniform hypergraph up to an εn^r additive error, making* $\mathrm{poly}(1/\varepsilon)$ *queries and taking* $\exp(\mathrm{poly}(1/\varepsilon))$ *time, and an accompanying algorithm taking* $\exp(\mathrm{poly}(1/\varepsilon)) + \mathrm{poly}(1/\varepsilon)n$ *time for finding a cut witnessing the approximate maximum.*

We note that some of these results are not new. For example, a test for a maximum independent set was first obtained by Langberg [Lan04], and a test for hypergraph c-colorability was first obtained by Czumaj and Sohler [CS05]. Moreover, the ad-hoc versions of the results of [CS05] and [Lan04] have a better dependence on $1/\varepsilon$ than the algorithms derivable from our method of obtaining general hypergraph partitions, although we could also improve our dependency by doing ad-hoc optimizations in the proof for the special cases. The main motivation for the above proposition is to show that our result can be thought of as a common generalization of many previous results.

5.3.1 Estimating the maximum number of satisfiable clauses in a k-CNF

Up to now it would have been enough to prove a version of our main theorem that would apply only to undirected hypergraphs. The next application, related to the work of [AVKK03], shows an instance in which the ordering of the edges is important.

Corollary 5.3.2. *For a fixed k, the following follows from using Theorems 5.2.1 and 5.2.2 for the appropriate setup: An algorithm for approximating the minimum number of unsatisfiable clauses in an n-variable k-CNF up to an εn^k additive error, making* $\mathrm{poly}(1/\varepsilon)$ *queries and taking* $\exp(\mathrm{poly}(1/\varepsilon))$ *time, and an accompanying algorithm taking* $\exp(\mathrm{poly}(1/\varepsilon)) + \mathrm{poly}(1/\varepsilon)n$ *time for finding an assignment witnessing the approximate minimum.*

Proof. We need to use here $k+1$ "edge sets", $E_0, \dots, E_k$, where E_i will correspond to all clauses for which exactly i of the k literals are negated. Moreover, we order each clause in E_i so that the i negated literals are first. We consider the n variables as "vertices", and seek a partition of them into two sets V_0 and V_1.

We now formulate the property Ψ_α, as the property that in the desired partition no more than a total of αn^k edges are in some E_i while their first i vertices are in V_1 and all their other vertices are in V_0. It is now not hard to see that a partition witnessing Ψ_α can be converted to an assignment leaving no more than αn^k clauses unsatisfied, by assigning to each variable x_i the value j_i for which the corresponding vertex v_i is in V_{j_i}.

Finally, to estimate the minimum number, we run our partitioning algorithm for any α which is an integer multiple of $\varepsilon/2$, with approximation parameter $\varepsilon/2$ and confidence parameter $\varepsilon/6$ (so with probability at least $\frac{2}{3}$ we make no error on any run of the partitioning algorithm). $\square$

We note that here again the degree of our polynomial is worse than the one in [AVKK03], but also here we could have made it better with ad-hoc optimizations. Also, the original result of [AVKK03] refers to general k-CSP instances rather than k-CNF ones, but it is not hard to either reduce the former to the latter, or use a slightly more complicated instance of the partitioning problem.

5.4 Finding a regular partition of a graph

5.4.1 Background and statement

The most interesting application we have of Theorem 5.2.1 is a new algorithmic approach for constructing a regular partition of a graph (in the sense of Szemerédi [Sze78]). This approach leads to an algorithm that improves upon the previous algorithms for this problem both in the running time (note that we make a tradeoff here by constructing a sublinear time *probabilistic* algorithm), and in the guarantee that a small regular partition will be found if one exists, with a running time that corresponds to the actual output size. Although this is a result on graphs, it cannot be derived from the graph version of Theorem 5.2.1 (i.e. the original GGR-algorithm) because a key feature of the algorithm is that it considers relations between more than two vertices of the graph.

The regularity lemma of Szemerédi [Sze78] is one of the most important results in graph theory, as it guarantees that every graph has an ε-approximation of constant descriptive size, namely a size that depends only on ε and not on the size of the graph. This approximation "breaks" the graph into a constant number of pseudo-random bipartite graphs. This is very useful in many applications since dealing with random-like graphs is much easier than dealing with arbitrary graphs. For a comprehensive survey of the applications of the lemma, the interested reader is referred to [KS96].

We first state the lemma. Recall that for two nonempty disjoint vertex sets A and B of a graph G, we define $E(A, B)$ to be the set of edges of G between A and B. The *edge density* of the pair is defined by $d(A, B) = |E(A, B)|/(|A||B|)$. The original notion of regularity, to which we refer to as *subset regularity*, was defined based on an easy to observe property of random graphs.

Definition 10 (ε-subset-regular pair)**.** *A pair (A, B) is* ε-subset-regular *if for every $A' \subseteq A$ and $B' \subseteq B$ satisfying $|A'| \geq \varepsilon|A|$ and $|B'| \geq \varepsilon|B|$, we have $d(A', B') = d(A, B) \pm \varepsilon$.*

An ε-regular pair can be thought of as a pseudo-random bipartite graph in the sense that it behaves almost as we would expect from a random bipartite graph of the same density, see e.g. [DLR95]. The intuition behind the above definition is that if (A, B) behaves like a random bipartite graph with edge density d, then all large enough sub-pairs should have similar densities.

A partition $V_1, \ldots, V_k$ of the vertex set of a graph is called an *equipartition* if $|V_i|$ and $|V_j|$ differ by no more than 1 for all $1 \leq i < j \leq k$ (so in particular every V_i has one of two possible sizes). The *order* of an equipartition denotes the number of partition classes (k above). An equipartition $V_1, \ldots, V_k$ of the vertex set of a graph is called *ε-subset-regular* if all but at most $\varepsilon\binom{k}{2}$ of the pairs (V_i, V_j) are ε-subset-regular. Szemerédi's regularity lemma can be formulated as follows.

Lemma 5.4.1 ([Sze78])**.** *For every $\varepsilon > 0$ there exists $T = T_{5.4.1}(\varepsilon)$, such that any graph with $n \geq 1/\varepsilon$ vertices has an ε-subset-regular equipartition of order $\hat{k}$, where $1/\varepsilon \leq \hat{k} \leq T$.*

It is far from surprising that a lemma that supplies an approximation of constant size for arbitrarily large graphs will also have algorithmic applications. The original proof of the regularity lemma was non-constructive, in the sense that it did not supply an efficient polynomial time algorithm for obtaining a regular partition of the graph. The first polynomial time algorithm for constructing such a partition of a graph was obtained by Alon et al. [ADL$^+$94]. Additional algorithmic versions of the lemma were later obtained in [AN06, DLR95, FK99b, FK96, KRT03].

The main observation used in most of the above was that although subset-regularity is computationally hard even to detect, it is essentially equivalent to an alternative definition that is based on a simple count of local structures.

Given (A, B), we denote by $d_{C_4}(A, B)$ the density of C_4 instances (cycles of size 4) between A and B, namely, the number of copies of C_4 whose vertices alternate between A and B, divided by their possible maximum number $\binom{|A|}{2}\binom{|B|}{2}$. [2]

Definition 11 (ε-locally-regular pair and partition)**.** *A pair (A, B) of vertex sets is* ε-locally-regular *if $d_{C_4}(A, B) = d^4(A, B) \pm \varepsilon$.*

An equipartition $V_1, \ldots, V_k$ of the vertex set of a graph is called ε-locally-regular *if all but at most $\varepsilon\binom{k}{2}$ of the pairs (V_i, V_j) are ε-locally-regular.*

In the sequel we will refer to local-regularity simply as *regularity* whenever there is no ambiguity in the context. The main observation about this definition is that an $\varepsilon^{O(1)}$-regular pair is also ε-subset-regular, and the other direction similarly holds. This was first

[2]When counting the number of C_4 instances between A and B we only consider the edges connecting A and B. Therefore, $0 \leq d_{C_4}(A, B) \leq 1$.

proved in [ADL$^+$94]. Our algorithm will center on finding partitions in which the pairs conform to local regularity.

The main drawback of the regularity lemma is that the bounds that the lemma guarantees have an enormous dependence on ε. More precisely, denote the tower of exponents function as $\mathrm{Tower}(i) = 2^{\mathrm{Tower}(i-1)} = 2^{\left.2^{\cdot^{\cdot^{2}}}\right\}i}$. Then the bounds on $T_{5.4.1}(\varepsilon)$ are given by $\mathrm{Tower}(1/\varepsilon^5)$. Furthermore, Gowers [Gow97] proved that these bounds are not far from the truth (in the qualitative sense), as he constructed a graph that has no ε-regular partition of size less than $\mathrm{Tower}(1/\varepsilon^{1/16})$.

Of course, this takes a huge toll on the dependency on ε of the running time of all the previous algorithms for constructing regular partitions. Also, as these algorithms essentially follow the original proof of Szemerédi's lemma (while substituting the newer and easier to check definition of regularity as in Definition 11), they are not guaranteed to run faster even if the graph admits a very small regular partition, as it may be overlooked considering the way the lemma was proved.

In contrast, the algorithm we design in this chapter takes a different approach, and manages to directly check the graph (up to some tolerance) for the existence of a regular partition with a prescribed order k. Thus, a smaller regular partition, if one exists, will indeed be returned. For the same reason, the running time is also reduced. For a fixed ε, the running time will be linear in the size of the actual output (making this a sublinear time algorithm in the size of the input). The worst case running time will only occur if the graph is indeed a worst-case graph, such as the one constructed in [Gow97]. Another added feature is that if we only care about the densities of the regular partition and not the partition itself, then there is a variant of the algorithm that takes constant time, assuming that a uniformly random choice of a vertex and an edge query can be done in constant time. The following is the new algorithmic version of Lemma 5.4.1.

Theorem 5.4.2. *There is a randomized algorithm, that given $\varepsilon > 0$ and an n vertex graph G, that has a $\frac{1}{2}\varepsilon$-regular partition of order $\hat{k} \geq 1/\varepsilon$,* [3] *produces with high probability an ε-regular partition of G of order k, where $1/\varepsilon \leq k \leq \hat{k}$.*

The expected running time of the algorithm is $O(\exp(\exp(O(\hat{k}^5))) + n \cdot \mathrm{poly}(\hat{k}))$. Also, the densities of the partition can be found and output in time $\exp(\exp(O(\hat{k}^5)))$, independently of n.

We stress that in Theorem 5.4.2, the algorithm does *not* receive the number $\hat{k}$ as part of the input. Also, the hidden constants in the running time are *absolute* and do not depend on ε or $\hat{k}$, and correspondingly there is no unconditional Tower-dependence on ε as in the previous algorithms. In addition, it is not hard to see that the constant $\frac{1}{2}$ in

[3]The reason for the requirement that $\hat{k} \geq 1/\varepsilon$ is that the same requirement is present in Lemma 5.4.1. For the same reason we return a partition of size $k \geq 1/\varepsilon$.

the statement can be replaced with any constant smaller than 1. Finally, although the algorithm does not know $\hat{k}$ in advance, its running time does depend, in a good sense, on $\hat{k}$. That is, if $\hat{k}$ is small then the running time of the algorithm will also be small, compared to the unconditional Tower-type dependence on ε of the previous algorithms.

Note that if one takes $\hat{k}$ in Theorem 5.4.2 to be the upper bound on the size of a regular partition that must exist in any graph, that is $\hat{k} = T_{5.4.1}(\varepsilon) = \text{Tower}(\text{poly}(1/\varepsilon))$, then we simply get a randomized algorithm for constructing a regular partition of a graph. An interesting feature of this algorithm is that its running time is $O(n)$ for any fixed ε. This algorithm is faster than the previous algorithms that ran in time $\Omega(n^2)$ (and higher). The price that we pay is that our algorithm is randomized while the previous algorithms were deterministic.

5.4.2 The proof idea

Our main idea is that instead of trying to reprove the regularity lemma "algorithmically" as in the previous approaches, we take Lemma 5.4.1 for a fact and just try to find the smallest regular partition that the graph has. Starting from $k = 1/\varepsilon$ we try to find a regular partition of G of order k, and if none exists we move to $k + 1$. Lemma 5.4.1 guarantees that we will eventually find a regular partition. To implement the above approach, we need an algorithm that will produce a regular partition of G of order k if one exists, and will not output any partition if none exists. Let's see how we can use Theorem 5.2.1 to obtain such an algorithm.

A key difference between a partition-instance in the sense of [GGR98] and a regular partition is that in a partition-instance we only care about the density of each pair (V_i, V_j), while in a regular partition we care about the *distribution* of the edges between V_i and V_j given in terms of a sort of a second moment condition, that of Definition 11. The framework of the graph partition problems of [GGR98] thus cannot by itself provide a check for regularity (unless "negations" of partition properties are checked as in [FN05] and in the hypergraph regularity algorithms here, but these again may lead to a small regular partition being overlooked). However, as we show below, a hypergraph partition theorem such as Theorem 5.2.1 is very useful for checking the regularity condition of Definition 11.

Given a graph G let us *implicitly* construct a hypergraph $H = H(G)$ on the vertex set $V(H) = V(G)$, that contains the 2-edges of G as well as 4-edges corresponding to the 4-cycles of G. This is not an undirected hypergraph, so we need at least some part of the additional generality provided in Theorem 5.2.1. Suppose that G has an ε-regular partition $V_1, \dots, V_k$. Then, for any pair (V_i, V_j) which is ε-regular, Definition 11 says that there is a real d such that $d = d(A, B)$ and the number of 4-cycles spanned by (A, B) is $(d^4 \pm \varepsilon)\binom{|V_i|}{2}\binom{|V_j|}{2}$. Hence, the same partition $V_1, \dots, V_k$ when considered on H, would have the property that if (V_i, V_j) is regular (in G) with density d, then the density of 4-edges in H connecting a pair of vertices from V_i with a pair of vertices from V_j is $d^4 \pm \varepsilon$.

Therefore, our algorithm for constructing a regular partition of G simply looks for a partition of H into k sets $V_1, \ldots, V_k$ such that most pairs (V_i, V_j) have the property that the density of 4-edges connecting them is close to $d^4(V_i, V_j)$. By the above discussion we know that if a graph G has an ε-regular partition then H satisfies the partition-instance. One should be careful at this point, as Theorem 5.2.1 only guarantees that if the hypergraph satisfies the partition-instance then the algorithm will return a possibly *different* partition that is close to satisfying the partition-instance. Luckily, it is not difficult to see that if the algorithm returns a partition of H that is close to satisfying the above partition-instance, then it is in fact regular with slightly worse parameters.

We finally note that the above explanation describes the method of designing an algorithm whose running time is as stated with *high probability* and not in *expectation.* To maintain a low expected running time, without knowing the value of $\hat{k}$ (which may be very large), we need one final trick: As we go and try higher and higher values of k, we go back and try again the lower values, to get another chance at small partitions that may have been missed during the first try. We note that in this use of Theorem 5.2.1 we rely on the fact that the dependence on δ in Theorem 5.2.1 is only poly-logarithmic. Thus we execute the more costly iterations with sufficiently small probability relative to their cost.

5.4.3 Proof of Theorem 5.4.2

We first formally describe the reduction, through which we reduce the problem of finding a regular partition of a graph G to the problem of finding a partition of a hypergraph $H = H(G)$ satisfying certain density conditions. Given a graph G let us define the following hypergraph $H = H(G)$.

Definition 12. *For a graph G, the hypergraph $H = H(G)$ has the same vertex set as G, and two sets of edges, a set E of* 2*-edges and a set C of (ordered)* 4*-edges. $E(H)$ is made identical to $E(G)$ (if we insist on the ordered edge setting of Theorem 5.2.1, then for every $\{u, v\} \in E(G)$ we have both (u, v) and (v, u) in H). We set C to include all sequences (u_1, u_2, u_3, u_4) that form a* 4*-cycle in that order in G.*

We note that also C has symmetries due to redundancy, as for example $(u_1, u_2, u_3, u_4) \in C$ if and only if $(u_2, u_3, u_4, u_1) \in C$. If we want to keep a hypergraph-like structure without redundancy, then we should define $E(H)$ as a set of unoredered pairs (just like $E(G)$) and $C(H)$ as a set of unordered pairs of unordered pairs, where the cycle (u_1, u_2, u_3, u_4) would be represented by the pair of pairs $\{\{u_1, u_3\}, \{u_2, u_4\}\}$. This is the representation that we adopt from now on (moving back to the fully ordered representation would just entail adding a constant coefficient in Definition 13 below).

Let us note that while discussing regular partitions, we measure the density of edges and C_4's between sets V_i, V_j relative to the size of V_i and V_j, see for example Definitions 10 and 11. On the other hand, when discussing the partition problems related to Theorem 5.2.1, we considered the density of edges relative to the number of vertices in the entire graph. In order to keep the following discussion aligned with the definitions of Subsection 5.2.1, let us use the convention of Theorem 5.2.1 of considering densities relative to the number of vertices in the entire graph. Therefore, in our density tensors we set $\mu(i,j)$ as the number of edges in $E(H)$ between V_i and V_j divided by n^2 and set $\mu_{C_4}(i,j)$ as the number of directed edges in $C(H)$ consisting of a pair from V_i and a pair from V_j, divided by n^4. Our hypergraph partition property is now defined as the following.

Definition 13. *Let $\Psi(k,\varepsilon)$ denote the set of density tensors for partitions $(V_1,\dots,V_k)$ which are equipartitions, and in addition for at least a $1-\varepsilon$ fraction of the pairs $1 \le i < j \le k$ we have $\mu_{C_4}(i,j) = \frac{1}{4}k^4\mu^4(i,j) \pm \frac{\varepsilon}{4k^4}$.*

It is not hard to see that the above Ψ is efficiently checkable for proximity as per Definition 9, as required for Theorem 5.2.1 and Theorem 5.2.2.

We claim that a partition satisfying Ψ over H indeed satisfies the (local) regularity condition over G. For simplicity, we ignore additive $O(\frac{k}{n})$ factors all throughout the following.

Claim 5.4.3. *Given a graph G, let $H = H(G)$ be the hypergraph defined above. Then, a partition of $V(G)$ into sets $V_1,\dots,V_k$ is ε-regular if and only if the same partition of $V(H)$ satisfies $\Psi(k,\varepsilon)$. Also, if a partition of $V(H)$ $\frac{\varepsilon}{16k^4}$-closely satisfies $\Psi(k,\frac{1}{2}\varepsilon)$ then the same partition of $V(G)$ is ε-regular.*

Proof. For two disjoint vertex sets W_1, W_2, Let us define by $E(W_1,W_2)$ the set of edges of $E(H)$ from W_1 to W_2, and by $C_4(W_1,W_2)$ the set of 4-edges of $C(H)$ of type $\{\{u_1,u_2\},\{v_1,v_2\}\}$ with $u_1,u_2 \in V_i$ and $v_1,v_2 \in V_j$. It is now not hard to see that (up to $O(\frac{k}{n})$ additive factors) we have $d(V_i,V_j) = \frac{|E(V_i,V_j)|}{(n/k)^2} = k^2\mu(i,j)$, and similarly $d_{C_4}(V_i,V_j) = \frac{|C(V_i,V_j)|}{\binom{n/k}{2}^2} = 4k^4\mu_{C_4}(i,j)$. Now, the ε-regularity condition of Definition 11 requires that $d_{C_4}(V_i,V_j) = d(V_i,V_j)^4 \pm \varepsilon$. By the above this is equivalent to the condition $\mu_{C_4}(i,j) = \frac{1}{4}k^4\mu^4(i,j) \pm \frac{\varepsilon}{4k^4}$, as needed.

As for the second assertion of the lemma, note that $\Psi(k,\frac{1}{2}\varepsilon)$ requires that for all but at most $\frac{1}{2}\varepsilon\binom{k}{2}$ of the pairs (V_i,V_j) we have $\mu_{C_4}(i,j) = \frac{1}{4}k^4\mu^4(i,j) \pm \frac{\varepsilon}{8k^4}$. If a partition $\frac{\varepsilon}{16k^4}$-closely satisfies this condition, it means that for all these pairs (V_i,V_j) we have $\mu_{C_4}(i,j) = \frac{1}{4}k^4\mu^4(i,j) \pm \frac{\varepsilon}{4k^4}$, which means by the above discussion that this partition is ε-regular. □

The algorithm: We only prove the version that provides the actual partition, as proving the version that provides densities in constant running time is almost word-for-word identical, only instead of Theorem 5.2.1 one would use Theorem 5.2.2 respectively. We start by describing a version of the algorithm that will run in the stated time with high probability (say, 3/4) rather than in expectation. We will later add one more idea that will give the required bound on the expectation. Given a graph G and a parameter $\varepsilon > 0$, our goal is to produce an ε-regular partition of G of size at least $1/\varepsilon$ and at most $\hat{k}$, assuming that G has a $\frac{1}{2}\varepsilon$-regular partition of size at most $\hat{k}$ (remember that the algorithm does not receive $\hat{k}$ as part of the input). Starting from $k = 1/\varepsilon$, we execute the hypergraph partitioning algorithm on the hypergraph $H = H(G)$ that was described at the beginning of this section with the partition instance $\Psi(k, \frac{1}{2}\varepsilon)$, with success probability δ_k (that will be specified later) and with approximation parameter $\frac{\varepsilon}{16k^4}$. Note that each call to the algorithm of Theorem 5.2.1 is done with a different value of k. Let us name the step where we call the partition algorithm with partition-instance $\Psi(k, \frac{1}{2}\varepsilon)$ the k^{th} *iteration* of the algorithm.

A crucial point here is that we *do not* explicitly construct H as that may require time $\Theta(n^4)$. Rather, whenever the hypergraph partition algorithm of Theorem 5.2.1 asks whether a pair of vertices $\{v_1, v_2\}$ is connected to another pair of vertices $\{u_1, u_2\}$, we just answer by inspecting the four corresponding edges of G (in constant time). If for some integer k the hypergraph partition algorithm returns a partition of $V(H)$, then we return the same partition for $V(G)$. Otherwise, we move to the next integer $k+1$. If we reached $k = T_{5.4.1}(\frac{1}{2}\varepsilon) = \text{Tower}(\text{poly}(1/\varepsilon))$ we stop and return "fail".[4]

Correctness: Let us now prove the correctness of the algorithm. Recall that we denote by δ_k the error probability with which we apply the partition algorithm of Theorem 5.2.1 at the k^{th} iteration. Remember that the algorithm does not know r in advance, and it may be the case that r is as large as $\text{Tower}(1/\varepsilon)$ (due to the lower bound of Gowers [Gow97]). Therefore, one way to resolve this is to take each δ_k to be $1/\text{Tower}(1/\varepsilon)$. However, that would mean that the running time of the partition algorithm would be $\text{Tower}(1/\varepsilon)$ even if r is small (as the running time depends on δ_k).

A more economic way around this problem is to take $\delta_k = \frac{1}{4} \cdot 2^{-k}$. This way the probability of making an error when considering all possible values of k is bounded by $\sum_{k=1}^{\infty} \frac{1}{4} \cdot 2^{-k} \leq \frac{1}{4}$. Note that this in particular means that with high probability we will not return a partition of G that does not $\frac{\varepsilon}{16k^4}$-closely satisfy $\Psi(k, \frac{1}{2}\varepsilon)$. Combined with the second assertion of Claim 5.4.3 we get that with high probability the algorithm returns only partitions that are ε-regular.

We thus only have to show that if G has a $\frac{1}{2}\varepsilon$-regular partition of size $\hat{k}$, then the algorithm will find an ε-regular partition of G of size at most $\hat{k}$. Suppose then that G has

[4]The choice of the maximum k is because by Lemma 5.4.1 we know that any graph has a $\frac{1}{2}\varepsilon$-regular partition of size at most $T_{5.4.1}(\frac{1}{2}\varepsilon)$. Therefore, if we reached that value of k then we know that we made a mistake along the way.

such a partition of size $\hat{k}$. We show that with high probability the algorithm will find such a partition during the $\hat{k}^{th}$ iteration (of course it may be the case that it will find a smaller partition during one of the previous iterations). Let $H = H(G)$ be the hypergraph defined above. By Claim 5.4.3 we know that as G has a $\frac{1}{2}\varepsilon$-regular partition of size $\hat{k}$, H satisfies $\Psi(\hat{k}, \frac{1}{2}\varepsilon)$. Therefore, with probability at least $1 - \delta_{\hat{k}} \geq 3/4$, the partition algorithm will return a partition of H that $\frac{\varepsilon}{16\hat{k}^4}$-closely satisfies $\Psi(\hat{k}, \frac{1}{2}\varepsilon)$. By the second assertion of Claim 5.4.3 we know that such a partition is ε-regular.

Running time with high probability: Let us bound the running time of the algorithm. Since with high probability the hypergraph partition algorithm will generate an ε-regular partition of G when reaching the $\hat{k}^{th}$ iteration (or earlier), we get that with high probability the algorithm will terminate after at most $\hat{k}$ iterations. When executing the algorithm of Theorem 5.2.1 on $\Psi(\hat{k}, \frac{1}{2}\varepsilon)$ we need to set $s = 1$, $r = 4$, $\delta = \delta_k = \frac{1}{4} \cdot 2^{-k}$. The number of partitions is k while the proximity parameter should be $\frac{\varepsilon}{16k^4} \geq \frac{1}{16k^5}$ (remember that we start with $k = 1/\varepsilon$). It follows from Theorem 5.2.1 that the running time of the hypergraph partition algorithm in the k^{th} iteration is $O(2^{2^{O(k^5)}} + n \cdot \text{poly}(k))$. As the running time of k iterations is clearly dominated by the running time of the k^{th} one, we get that with probability at least $\frac{3}{4}$ both the answer will be correct and the running time will be bounded by $O(2^{2^{O(\hat{k}^5)}} + n \cdot \text{poly}(\hat{k}))$.

Bounding the expected running time: The version of the algorithm described above runs in time $O(2^{2^{O(\hat{k}^5)}} + n \cdot \text{poly}(\hat{k}))$ with high probability, but this is not its expectation. The reason is that the error of not finding one small existing regular partition could be very costly, as it could be followed with many iterations of searching in vain for higher values of k (remember that the larger k is, the more costly it is to execute the algorithm of Theorem 5.2.1). Suppose then that we partition the execution of the algorithm into *phases*, where in the k^{th} phase we execute the algorithm described above from the first iteration till iteration k, only now we use $\delta_k = 2^{-2^{b \cdot k^6}}$ for *all* k iterations, with b a constant to be chosen.

The modified algorithm clearly has a larger probability of outputting the required regular partition and a smaller probability of returning a wrong partition (because $2^{-2^{b \cdot k^6}} \leq k^{-1}2^{-k}$). Let us compute the expected running time, which we bound using $\sum_{k=m}^{\infty} p_k \cdot t_k$ where p_k is the probability of the algorithm performing the k^{th} phase, and t_k is the "cost" of the k^{th} phase, that is the running time of that phase. Similarly to what we have discussed above, the time it takes to execute the iterations $1, \ldots, k$ of the original algorithm, even with the new δ_k, is bounded by $O(2^{2^{O(k^5)}} + n \cdot \text{poly}(k))$. The contribution of the first $\hat{k}$ phases is clearly dominated by this expression for $k = \hat{k}$.

Now, let us focus on p_k for some $k > \hat{k}$. Suppose that the graph has a $\frac{1}{2}\varepsilon$-regular partition of size $\hat{k}$. To reach phase k for some $k > \hat{k}$, the algorithm must have in particular failed the attempt made in phase $k-1$ to find a partition of size $\hat{k}$. Remember that the failure probability of that attempt is at most δ_{k-1}. Therefore, the probability of reaching phase k is at most $\delta_{k-1} = 2^{-2^{b\cdot(k-1)^6}}$. Hence, the total time expectancy for this algorithm (where we set b to be a large enough constant) is bounded by

$$\begin{aligned}
\sum_{k=1/\varepsilon}^{\infty} p_k \cdot t_k &= \sum_{k=1/\varepsilon}^{\infty} p_k \cdot O(2^{2^{O(k^5)}} + n \cdot \mathrm{poly}(k)) \\
&= O(2^{2^{O(\hat{k}^5)}} + n \cdot \mathrm{poly}(\hat{k})) + \sum_{k=\hat{k}+1}^{\infty} 2^{-2^{b\cdot(k-1)^6}} \cdot O(2^{2^{O(k^5)}} + n \cdot \mathrm{poly}(k)) \\
&= O(2^{2^{O(\hat{k}^5)}} + n \cdot \mathrm{poly}(\hat{k})) + O(n) \\
&= O(2^{2^{O(\hat{k}^5)}} + n \cdot \mathrm{poly}(\hat{k}))\ .
\end{aligned}$$

5.5 Regular partition of a hypergraph

Another main application of Theorem 5.2.1 is finding regular partitions of r-uniform hypergraphs. [5] Just like the algorithms for graph regularity have many applications for graph problems, the algorithms for hypergraph regularity have applications for hypergraph problems.

Hypergraph regularity has more than one version. The version we investigate here is the one discussed e.g. in [FK96] (the "vertex partition" version), which is not strong enough for some applications such as proving Szemerédi's Theorem on r-term arithmetic progressions [Sze75], but still has many applications. For example, it was used in [FK96] to obtain additive approximations for all Max-SNP problems. This notion of regularity is defined in an analog manner to the definition of ε-subset-regularity for graphs. The following is a quick rundown of the relevant definitions.

Definition 14. *For an r-uniform hypergraph H and an r-tuple $(U_1,\ldots,U_r)$ of vertex sets of H, the* edge density *$d(U_1,\ldots,U_r)$ is defined by the number of edges with one vertex from every U_i, divided by $\prod_{i=1}^{r}|U_i|$. An r-tuple $(U_1,\ldots,U_r)$ as above is called* ε-regular*, if every r-tuple $U'_1,\ldots,U'_r$ such that $U'_i \subseteq U_i$ and $|U'_i| \geq \varepsilon|U_i|$ satisfies $d(U'_1,\ldots,U'_r) = d(U_1,\ldots,U_r) \pm \varepsilon$.*

Finally, an equipartition $V_1,\ldots,V_k$ of the vertices of H is called ε-regular *if for all but at most an ε fraction of the possible r-tuples $1 \leq i_1 < i_2 < \cdots < i_r \leq k$, we have that $(V_{i_1},\ldots,V_{i_r})$ is ε-regular.*

[5] A hypergraph $H = (V,E)$ is r-uniform if all its edges have exactly r distinct vertices of V. These edges are *unordered*, that is, they are just subsets of $V(H)$ of size r. Hence a simple graph is just a 2-uniform hypergraph.

As is the case for graphs, an ε-regular partition of an r-uniform hypergraph into a bounded number of sets always exists. However, obtaining an algorithmic version of the hypergraph version of the regularity lemma turns out to be more involved than the graph case, and in particular here we can no longer guarantee to find a small regular partition if one exists.

Theorem 5.5.1. *For any fixed r and $\varepsilon > 0$, there exists an $O(n)$ time probabilistic algorithm that finds an ε-regular partition of an r-uniform hypergraph with n vertices. Moreover, if we only want to find the densities of an ε-regular partition, then there exists an algorithm obtaining them in constant time.*

The improvement over previous results that comes from the linearity of the algorithm in its output becomes more apparent here: While the previous algorithms (see, for example, [CR00, FK96]) for constructing a regular partition of an r-uniform hypergraph have running time $O(n^{2r-1})$ (note that this is close to being quadratic in the input size), our algorithm has running time $O(n)$ for any r. Like the previous algorithms, and unlike the algorithm given in Theorem 5.4.2, the algorithm in Theorem 5.5.1 still has a tower-type dependence on ε.

Unlike the algorithm for graph regularity from the previous section that directly finds a regular partition, the algorithm of Theorem 5.5.1 makes use of some aspects of the iterative procedure for proving the existence of a regular partition, which repeatedly refines a partition of the hypergraph until a regular one is obtained. A direct implementation of such a procedure would suffer from NP-Completeness issues, and even if this is resolved in the style of previous works it would still take at least $\Omega(n^r)$ time as all deterministic algorithms have to (and in the previous works it actually takes longer). The crux of our proof is to apply an idea from [FN05] along with Theorem 5.2.1 in order to implement (an aspect of) the iterative procedure in time $O(n)$.

5.5.1 Proof for Theorem 5.5.1

The main result in [FN05] required a way to quickly find the densities of a regular partition of a graph.[6] While the full result there does not seem translatable to our current body on knowledge on hypergraphs, the part about finding a regular partition is indeed translatable to a proof of Theorem 5.5.1. In the following we describe how to adapt the appropriate arguments from [FN05] to hypergraph regularity.

Equipartitions and the index function: In all that follows, we consider an r-uniform simple and unordered hypergraph H with a vertex set V, and we assume throughout that $|V|$ is large enough as a function of the other parameters (for a smaller $|V|$, we can just do

[6] In fact, in [FN05] a partition conforming to a strengthened notion of regularity was required.

an exhaustive search). An *equipartition* $V_1, \ldots, V_k$ of the vertex set V is defined as before. The main tool in proving regularity is the following numeric function, that in some sense measures how much the densities of the r-tuples in the partition vary.

Definition 15. *For an r-tuple of vertex sets $(W_1, \ldots, W_r)$ of an r-uniform hypergraph G, we define the* density *$d(W_1, \ldots, W_r)$ of the r-tuple as the number of edges with one vertex from every W_i, divided by $\prod_{j=1}^{r} |W_j|$.*

For an equipartition $\mathcal{A} = (V_1, \ldots, V_k)$ of an r-uniform hypergraph G into l sets, its index *$\mathrm{ind}(\mathcal{A})$ is defined as $k^{-r} \sum_{1 \le i_1 < i_2 < \cdots < i_r \le k} (d(V_{i_1}, \ldots, V_{i_r}))^2$.*

The index of any equipartition is clearly always between 0 and 1. One immediate but useful observation is the continuity of the index function with respect to the densities.

Claim 5.5.2. *If an equipartition $\mathcal{A} = (V_1, \ldots, V_k)$ of G and an equipartition $\mathcal{A}' = (V'_1, \ldots, V'_k)$ of G' satisfy $|d(V_{i_1}, \ldots, V_{i_r}) - d(V'_{i_1}, \ldots, V'_{i_r})| \le \varepsilon$ for all $1 \le i_1 < i_2 < \cdots < i_r \le k$, then the respective indexes satisfy $|\mathrm{ind}(\mathcal{A}) - \mathrm{ind}(\mathcal{A}')| \le 4\varepsilon$.*

Robustness, finality, and regularity: The main technical observation of Szemerédi in [Sze78] is that non-regular partitions can be refined in a way that substantially increases their index, while (because the index is bounded by 1) there must be partitions which cannot be refined that way. This observation carries to hypergraphs. Let us state this formally.

Definition 16. *A* refinement *of an equipartition $\mathcal{A} = (V_1, \ldots, V_k)$ is an equipartition $\mathcal{B} = (W_1, \ldots, W_l)$ such that every set W_j is fully contained in some V_{i_j}.*

*Given $\delta > 0$ and a function $f : N \to N$, we say that an equipartition $\mathcal{A} = (V_1, \ldots, V_k)$ is (δ, f)-*robust *if there exists no refinement $\mathcal{B} = (W_1, \ldots, W_l)$ for which $k \le l \le f(k)$ and $\mathrm{ind}(\mathcal{B}) \ge \mathrm{ind}(\mathcal{A}) + \delta$. The equipartition $\mathcal{A} = (V_1, \ldots, V_k)$ is called (δ, f)-*final *if there exists no $\mathcal{B} = (W_1, \ldots, W_l)$ as above regardless of whether it is a refinement of $\mathcal{A}$ or not.*

It is immediate to see that any (δ, f)-final partition is also (δ, f)-robust. We work with the definition of a final partition because it is somewhat easier to handle in the framework of Theorem 5.2.1. By using the fact that the index is always bounded between 0 and 1, it is not hard to see the following.

Claim 5.5.3. *For every $\delta > 0$, t and $f : N \to N$ there exists $T = T_{5.5.3}(\delta, f, t)$ such that every graph G with $n > T$ vertices has an equipartition $\mathcal{A} = (V_1, \ldots, V_k)$ with $t \le k \le T$ which is (δ, f)-final.*

The main technical step in [Sze78] is using the "defect form" of the Cauchy-Schwartz inequality to derive a connection between robustness and regularity. The proof is based on refining the equipartition according to the algebra generated by the regularity counter examples for the irregular r-tuples, and works almost word for word for hypergraphs. We will not reproduce it here as it has been proved in several papers, see e.g. [Chu91] and [CR00].

Lemma 5.5.4. *For every r and ε there exist $\delta = \delta_{5.5.4}(r, \varepsilon)$ and $f(k) = f^{(r,\varepsilon)}_{5.5.4}(k)$ so that any equipartition of an r-uniform hypergraph that is (δ, f)-robust is also ε-regular.*

The above lemma immediately suggests a way to generate a regular partition. Starting from an arbitrary partition, we can repeatedly refine the partition until a regular one is obtained. Indeed, that is the main idea in the original proofs of Szemerédi's theorem for graphs and hypergraphs. The first problem with implementing this strategy algorithmically is that it is not clear how to efficiently detect if an r-tuple is not regular. This problem, however, can be overcome (see e.g. [ADL$^+$94] and [CR00]). The more relevant problem to our investigation here is that just calculating edge counts for a partition of the hypergraph already takes time $\Theta(n^r)$, and our goal is an $O(n)$ time algorithm. As we explain in the sequel, if we are looking for a partition satisfying a somewhat stronger condition than regularity, then one such partition can be produced in $O(n)$.

Finding a regular partition: The proof of Theorem 5.5.1 now manifests itself as checking a sequence of density tensor sets that each describes a partition into k sets which is (δ, f)-final for the parameters we obtain from Lemma 5.5.4. However, finality (or robustness) by itself cannot be described by a set of density tensors. What we can do is test for density tensors which conform to a possible value of the index function, and then try to verify that an equipartition is final by testing for finer partitions with somewhat higher indexes and *rejecting* our equipartition if they exist. Claim 5.5.2 allows us to use our approximate algorithms, while ensuring that the "negation" procedure will also work (i.e. that for an equipartition that is not final we will indeed detect this).

For an integer k and $0 \leq \alpha \leq 1$, we let $\Psi^{(k,\alpha)}$ be the set of all density tensors of possible equipartitions into k sets whose index is at least α. We set $\delta = \delta_{5.5.4}(r, \varepsilon)$ and $f(k) = f^{(r,\varepsilon)}_{5.5.4}(k)$, and now for every $1/\varepsilon \leq k \leq f(T_{5.5.3}(\frac{1}{3}\delta, f, 1/\varepsilon))$ and every $\alpha \in \{0, \frac{1}{6}\delta, \frac{2}{6}\delta, \ldots, 1\}$ we run either the algorithm of Theorem 5.2.1 or the algorithm of Theorem 5.2.2 (depending on whether we want to find the actual partition or just the densities) on $\Psi^{(k,\alpha)}$, with approximation parameter $\frac{1}{24}k^{-r}\delta$, and with success probability sufficient to ensure that with probability at least $\frac{2}{3}$ none of the iterations provides an erroneous answer.[7]

We are now interested in finding k_0 and $\alpha_0 = \frac{\ell}{6}\delta$ for which we received a positive answer for $\Psi^{(k_0,\alpha_0)}$ and received a negative answer for all $\Psi^{(k,\alpha_0+2\delta/3)}$ for $k_0 \leq k \leq f(k_0)$.

[7]The additional k^{-r} factor in the approximation parameter is because of the difference in the normalization between the definition of the density tensors in Ψ and the normalization of the density measure $d(W_1, \ldots, W_r)$.

To conclude the proof, it is enough to convince ourselves that the following observations hold assuming that none of our iterations has provided an erroneous answer.

- For every k we can approximate the maximum index of an equipartition into k parts up to an error of $\frac{1}{6}\delta$ (in either direction). The lower bound follows directly from our procedure, while the upper bound follows from Claim 5.5.2
- A k_0 as above is obtained. This is because the k' for which there is a $(\frac{1}{3}\delta, f)$-final partition into k' sets (which exists by Claim 5.5.3) will in particular be detected due to the first item above (this does not mean that we will necessarily obtain $k_0 = k'$).
- For the k_0 that is obtained, the corresponding witnessing equipartition is (δ, f)-final and hence ε-regular. This is again due to the bounds of the first item above on the error in our index estimates.

The last two items above imply that we can then (by extracting the required data about the partition witnessing a maximum index for k_0) find our regular partition.

5.6 Overview of the proof of Theorems 5.2.1 and 5.2.2

Before going to the formal proof of Theorem 5.2.1 and Theorem 5.2.2 we briefly describe the general idea of how it is done.

The outermost layer of the proof borrows from the proof of [GGR98] for graph partitions. Assuming that the hypergraph admits a partition $\Pi = \{V_1, \ldots, V_k\}$ of the vertices with the desired densities, we first split V into $\ell = O(1/\varepsilon)$ parts $Y^1, Y^2, \ldots, Y^\ell$ of equal size. Then, for every part Y^i, a sample set $U \subset V \setminus Y^i$ is chosen at random with the hope of obtaining sufficiently many vertices from every V_j. Assume for now that we know the intersection of U with Π, i.e. the partition $\Pi_U = \{U \cap V_1, \ldots, U \cap V_k\}$. Then we try to reconstruct Π from its intersection with U as follows. First, the vertices in Y^i are clustered into a finite number of clusters, according to their degrees in the various components of Π_U (the degree of v is the number of hyperedges in which v participates, counted for all sets of edges and all possible configurations with respect to Π_U). Intuitively, each cluster groups together the vertices that look similar with respect to U and Π_U. Assume in addition that for every cluster $C \subset Y^i$ of similar vertices we also know their approximate distribution in the components of Π, i.e. we know some approximate $\bar{\beta} = \{\beta_1, \ldots, \beta_k\}$, where for every j, $\beta_j \approx \frac{|C \cap V_j|}{|C|}$. Then we partition every cluster of Y^i into the k components in a way that preserves the normalized intersection sizes, according to the corresponding $\bar{\beta}$.

Since we do not know the intersection Π_U, we simply try every possible partition of U. Similarly for the second assumption, since we do not know the intersection sizes for each cluster, we try all possibilities (up to some allowed deviation). For every such combination we get a different partition of V, and we test (by sampling) if one of them would give a reconstruction with the required densities.

The motivation for this kind of reconstruction is based on the following central observation. Suppose that $Y \subset V$ is a small enough set (but still linear in n) containing vertices that have similar degrees in the various V_i's. Assume that $\Pi' = \{V_1', \ldots, V_k'\}$ is a new partition of V that is constructed from Π by redistributing the vertices of Y arbitrarily, so long as the component sizes are almost preserved (i.e. for all i, $|V_i' \cap Y| \approx |V_i \cap Y|$). Then the densities of the partition Π' are almost similar to the densities of Π. We should also note that, like in [GGR98], it is not possible to classify all vertices of V at once while maintaining a small approximation error, so every Y^i is classified according to a partition of a different sample set U^i.

Although the overall strategy of the algorithm is similar to the GGR-algorithm, there are several differences between our analysis and the original one in [GGR98]. One of the reasons is that there are many ways in which an edge with r vertices can intersect the k vertex sets, complicating the procedure of classifying the vertices according to the edges they have with the sets of the partition. This reflects on which statistics we keep track of, and also on how we obtain an approximation thereof. Another reason is that Theorem 5.2.1 can have as an input a general set of density tensors, rather than intervals of allowed densities as in the GGR-algorithm [GGR98]. This extension of the GGR formalism allows us to use the algorithm more easily and efficiently in our applications.

5.7 The proof of Theorem 5.2.1 and Theorem 5.2.2

First we define the notion of partitioning oracles, and state the central lemma that implies the proof of Theorem 5.2.1 and Theorem 5.2.2.

Definition 17. *A mapping $\pi : V(H) \to [k]$ is called a (q, c)-*partition oracle *if:*

- *for any $v \in V(H)$, the query complexity of computing $\pi(v)$ is bounded by $O(q)$*
- *for any $v \in V(H)$, the time complexity of computing $\pi(v)$ is bounded by $O(c)$*

For a subset $Y \subset V(H)$, we define a partial partition oracle *$\pi : Y \to [k]$ similarly.*

Given a sequence $\{\pi_i\}_{i=1}^m$ of m partition oracles, we say that the sequence $\{\pi_i\}_{i=1}^m$ has shared query complexity *f, if for every $v \in V(H)$ computing the m values $\pi_1(v), \pi_2(v), \ldots, \pi_m(v)$ requires at most f queries in all.*

Now we are ready to state the main lemma required for the proof of Theorems 5.2.1 and 5.2.2.

Lemma 5.7.1. *Let $H = (V, E_1, \ldots, E_s)$ be a hypergraph, let $0 < \varepsilon < 1$ be a fixed constant and let Ψ be a set of density tensors. There exist $m = m_{5.7.1}(\varepsilon, s, r, k) = \exp\left(\varepsilon^{-O(s \cdot r \cdot k^r)}\right)$*

and $f = f_{5.7.1}(\varepsilon, s, r, k) = \mathrm{poly}(k^r, s, 1/\varepsilon)$ *for which there is a randomized algorithm* A_{R} *that generates a sequence of size* m *of* (f, f)*-partition oracles* $\{\pi_i : V(H) \to [k]\}_{i=1}^m$ *with shared query complexity* f*, such that:*

- *If* H *satisfies* Ψ *then with probability at least* $\frac{1}{2}$ *one or more of the* m *partition oracles induces a partition which* ε*-approximately satisfies* Ψ*.*
- *The algorithm* A_R *itself makes* no *queries to* H*, and it does not depend on the property* Ψ*.*

The running time of algorithm A_R *is bounded by* $O(m)$*.*

Note that the algorithm A_R does not depend on H or Ψ at all. It only depends on the general parameters $(s, r_1, \ldots, r_s, \varepsilon, k)$ of the problem, and none of the parameters of the specific input.

Informally, our partition oracles will be constructed as follows.

Representation: every partition oracle π will be associated with some small subset $U \subset V(H)$ of H's vertices, a partition Π_U of U into k components, and a vector $\bar{\beta}$. The set U will be usually shared among the partition oracles within a sequence.

Query complexity: for every vertex $v \in V(H)$, some subset of edges within $U \cup \{v\}$ will be queried. As we mentioned above, in case of sequences of partition oracles the set U will be shared, and hence the *shared query complexity* of the considered sequences will be small (when U is small).

Operation: every vertex $v \in V(H)$ is classified according to the outcomes of the queries in the previous stage and the partition Π_U. Then it is mapped to some $j \in [k]$ at random, according to the vector $\bar{\beta}$ which is interpreted as a distribution.

Before proving Lemma 5.7.1, we state two additional lemmas and through them prove Theorem 5.2.1 and Theorem 5.2.2.

Lemma 5.7.2. *Let* H *be a hypergraph and let* Ψ *be a set of density tensors. For every* $\varepsilon > 0$*, if* H ε'*-approximately satisfies* Ψ*, where* $\varepsilon' \triangleq \frac{\varepsilon}{2r}$*, then* H *also* ε*-closely satisfies* Ψ*.*

Lemma 5.7.3. *Let* Ψ *be a property that is checkable for proximity in time* $TC(\Psi)$*, and let* $\{\pi_i\}_{i=1}^m$ *be a sequence of* m (q, c)*-partition oracles with shared query complexity* f*. For every* $0 < \delta, \varepsilon < 1$*, there is a randomized algorithm* A_S *that satisfies the following.*

- *If one of the partition oracles* π_i *defines a partition* Π_i *that* $\varepsilon/2$*-approximately satisfies the property* Ψ*, then with probability at least* $1 - \frac{\delta}{2}$ *algorithm* A_S *outputs* i *and a density tensor* $\psi \in \Psi$ *which is* ε*-approximated by* ψ^{Π_i} *– the true density tensor of* Π_i*.*

- *If for every π_i the induced partition Π_i does not even ε-approximately satisfies Ψ, then algorithm A_S outputs* False *with probability at least $1 - \frac{\delta}{2}$.*

Furthermore, the query complexity of A_S is bounded by

$$O\Big((r \cdot f + s) \cdot (\frac{1}{\varepsilon})^2 \cdot \log(\frac{m \cdot sk^r}{\delta})\Big)$$

and the time complexity of A_S is bounded by

$$O\Big(m \cdot c \cdot (r \cdot f + s) \cdot (\frac{1}{\varepsilon})^2 \cdot \log(\frac{m \cdot sk^r}{\delta})\Big) + m \cdot TC(\Psi)$$

We proceed with proving Theorem 5.2.1 and Theorem 5.2.2, and postpone the proof of Lemmas 5.7.2 and 5.7.3 to Subsections 5.7.4 and 5.7.4 respectively.

Proof of Theorem 5.2.1

Proof. First we apply Lemma 5.7.1 with accuracy parameter $\varepsilon' = \frac{\varepsilon}{4r}$, repeating it $2\log(1/\delta)$ independent times, and thus we obtain $m' = m \cdot 2\log(1/\delta)$ partition oracles with shared query complexity $2\log(1/\delta) \cdot f$, where we know that if H satisfies Ψ, then with probability at least $1 - \frac{1}{2}\delta$ at least one of the m' partition oracles induces a partition which ε'-approximately satisfies Ψ. Then we can apply the algorithm from Lemma 5.7.3 on all m' partitions and write down a $2\varepsilon'$-approximately satisfying partition if we find one, and output "No Partition" otherwise. In case we found a $2\varepsilon'$-approximately satisfying partition oracle π, we partition all vertices of H in time $O(n \cdot f)$. Then we modify the resulting partition by making a minimal number of vertex moves according to Lemma 5.7.2 (see its proof below), so if one of the oracles indeed $2\varepsilon'$-approximately satisfies Ψ then with probability $1 - \delta/2$ Lemma 5.7.3 detected it and hence its corresponding modified partition ε-closely satisfies Ψ. As a conclusion, the time complexity of our algorithm is bounded by

$$O\Big(m' \cdot \Big[f_{5.7.1}^2 \cdot \Big(\frac{1}{\varepsilon'}\Big)^2 \cdot \log\Big(\frac{m'sk^r}{\delta}\Big) \cdot r \cdot s + TC(\Psi)\Big] + n \cdot f_{5.7.1}\Big) =$$

$$\exp\Big((r/\varepsilon)^{O(s \cdot r \cdot k^r)}\Big) \cdot \log^2(1/\delta) + n \cdot \mathrm{poly}(k^r, s, 1/\varepsilon)$$

and the probability of success is at least $1-\delta$, matching the assertions of Theorem 5.2.1. □

Proof of Theorem 5.2.2

Proof. First we apply Lemma 5.7.1 with $\varepsilon' = \frac{\varepsilon}{4r}$, repeating it $2\log(1/\delta)$ independent times and obtaining $m' = m \cdot 2\log(1/\delta)$ partition oracles with shared query complexity $2\log(1/\delta) \cdot f$, where with probability at least $1 - \frac{1}{2}\delta$, at least one of them induces an

ε'-approximately satisfying partition in the case that H satisfies Ψ. Then we can apply the algorithm from Lemma 5.7.3 on all m' partitions and output the density tensor $\psi \in \Psi$ if we received one from algorithm A_S described there. Observe that by Lemma 5.7.2, the density tensor ψ (in the case that the algorithm A_S did not err) is ε-closely satisfied by H. Hence the total probability of success is at least $1-\delta$. The time complexity of our algorithm is bounded by

$$\exp\left((r/\varepsilon)^{O(s\cdot r\cdot k^r)}\right)\cdot\log^2(1/\delta)$$

and its query complexity is bounded by

$$\mathrm{poly}(k^r, s, 1/\varepsilon)\cdot\log^2(1/\delta)$$

matching the assertions of Theorem 5.2.2. □

5.7.1 Proof of Lemma 5.7.1

As we mentioned in Section 5.6, in order to maintain a small approximation error the partitioning algorithm A_R is going to classify the vertices of H incrementally, by first splitting $V(H)$ into sets $Y^1, Y^2, \dots, Y^{8/\varepsilon}$ and then partitioning every set Y^i separately. Therefore, we first state a technical lemma related to this partial partitioning algorithm, and using it we prove Lemma 5.7.1.

Lemma 5.7.4. *Let $H = (V, E_1, \dots, E_s)$ be a hypergraph, let $0 < \alpha < \varepsilon < 1$ be fixed constants and let Ψ be a set of density tensors. Let in addition $\Pi = \{V_1, \dots, V_k\}$ be a partition of H's vertices that α-approximately satisfies the property Ψ and let $Y \subset V(H)$ be a set of size $\frac{\varepsilon}{8}n$.*

There exists $\hat{m} = \hat{m}_{5.7.4}(\varepsilon, s, k, r) = \exp\left(\varepsilon^{-O(s\cdot k^r)}\right)$ and $f = f_{5.7.4}(\varepsilon, s, k, r) =$ $\mathrm{poly}(k^r, s, 1/\varepsilon)$ for which there is a randomized algorithm $A_{1/\varepsilon}$ that outputs a sequence of $\hat{m}$ partial *(f,f)-partition oracles for Y with shared query complexity f,* without looking at Π, *so that with probability at least $1-\frac{\varepsilon}{16}$ at least one of the partition oracles induces a partition $\Pi_U = \{Y_1, \dots, Y_k\}$ such that combining it with the original partition Π gives a partition $\hat{\Pi} = \{\hat{V}_1, \dots, \hat{V}_k\}$ (where for every i, $\hat{V}_i = \big((V_i \setminus Y) \cup Y_i\big)$), which $(\alpha + \frac{\varepsilon^2}{8})$-approximately satisfies Ψ.*

Furthermore, the partial partitioning algorithm $A_{1/\varepsilon}$ makes no *queries to H at all, is not dependent on Ψ, and its time complexity is bounded by $O(\hat{m})$.*

Supposing that we have such a *partial* partitioning algorithm $A_{1/\varepsilon}$, we show how to construct the main partitioning algorithm A_{R} from Lemma 5.7.1.

Proof (of Lemma 5.7.1).

Proof. Assume that H has a partition Π that satisfies the property Ψ. First we arbitrarily split V into $l = 8/\varepsilon$ equal-sized sets $\{Y^1, \ldots, Y^l\}$. Then we execute for every Y^i the partial partitioning algorithm $A_{1/\varepsilon}$, and write down the $\hat{m}$ partition oracles of each Y^i. Recall that the partial partitioning algorithm $A_{1/\varepsilon}$ does not require knowledge of the existing partition Π. Using these $\hat{m} \cdot l$ partial partition oracles $\{(\pi_1^1, \ldots, \pi_{\hat{m}}^1), \ldots, (\pi_1^l, \ldots, \pi_{\hat{m}}^l)\}$ we construct a set of $\hat{m}^l$ *complete* partition oracles for V by going over all possible combinations. Namely, we combine every possible sequence $\pi_{i_1}^1, \pi_{i_2}^2, \ldots, \pi_{i_l}^l$ of l partial oracles into a complete oracle, that partitions all of H's vertices into k components. We claim that with probability at least $\frac{1}{2}$, one of these $\hat{m}^l$ complete partition oracles induces a partition that ε-approximately satisfies Ψ.

Assume that we start with an implicit partition $\Pi^0 = \{V_1^0, \ldots, V_k^0\}$ of H that satisfies the property Ψ, or equivalently α-approximately satisfies Ψ for $\alpha = 0$. Since every execution of $A_{1/\varepsilon}$ fails with probability at most $\frac{\varepsilon}{16} = \frac{1}{2l}$, with probability at least $1/2$ we have a sequence of l "correct" partial partition oracles that admits the following construction. To construct the correct complete partition oracle, we take the first "correct" induced partition $\Pi_{Y^1} = \{Y_1^1, \ldots, Y_k^1\}$ of Y^1, and build (implicitly) a new complete partition $\Pi^1 = \{V_1^1, \ldots, V_k^1\}$, where $V_i^1 = (V_i^0 \setminus Y^1) \cup Y_i^1$, and where we now know (by Lemma 5.7.4) that Π^1 is a partition that $\alpha + \frac{\varepsilon^2}{8} = \frac{\varepsilon^2}{8}$-approximately satisfies Ψ. Now we continue with a "correct" induced partition of Y^2 with respect to Π^1, and build a complete partition Π^2 that $2 \cdot \frac{\varepsilon^2}{8}$-approximately satisfies Ψ and so on. Eventually, with probability $1/2$, there is a "right" sequence of the induced partial partitions Π_{Y^i}, from which we get a complete partition Π^l which (due to the triangle inequality) $\frac{8}{\varepsilon} \cdot \frac{\varepsilon^2}{8} = \varepsilon$-approximately satisfies Ψ as required. Moreover, the way that the partitions are constructed is such that for this to work (by trying out all combinations of the provided partition oracles of $Y^1, \ldots, Y^l$) we require no knowledge of Π at all, apart from that such a partition exists. □

5.7.2 Algorithm $A_{1/\varepsilon}$ - proof of Lemma 5.7.4

Overview

The main idea behind the partial partitioning algorithm is the following. Let $\Pi = \{V_1, \ldots, V_k\}$ be the (unknown) partition which approximately satisfies the property Ψ, and let $\tilde{Y}$ be some small enough set of vertices. We refer to the *type* of an edge $e = (v_1, \ldots, v_{r_i})$ as its configuration with respect to Π, i.e. the number of vertices that it has in every component, and their positions in e. If almost all vertices in $\tilde{Y}$ participate in the same number of edges of any type (with respect to $V_1 \setminus \tilde{Y}, \ldots, V_k \setminus \tilde{Y}$), then we can redistribute $\tilde{Y}$ in any way, as long as the component sizes are almost preserved, and still get a partition Π' which is almost as good as Π.

Once we convinced ourselves that this is true, we can think of a somewhat larger set

$Y \subset V$, and its own partition $\{Y_1, \ldots, Y_c\}$ where in every Y_i (similarly to the set $\tilde{Y}$ above) almost all vertices are "similar" (note that this partition of Y is not related to the main partition Π, and the partition size c depends among other things on the level of accuracy in our notion of "almost"). Now, if we redistribute each Y_i in a way that preserves the component sizes in Π, then still (after the c redistributions) we get a partition which approximately satisfies the property Ψ.

In the following section we define the notion of "almost similar vertices" more precisely, and show how to cluster a given set in a way that groups almost similar vertices together.

Clustering vertices

Let $Y^0 \subset V$ be a fixed set of vertices and let $\Pi = \{V_1, \ldots, V_k\}$ be some partition of V. Denote by $\{W_1, \ldots, W_k\}$ the partition that Π induces on $V \setminus Y^0$, i.e. $\{V_1 \setminus Y^0, \ldots, V_k \setminus Y^0\}$. Our aim in this section is to cluster every vertex $v \in Y^0$ according to its relative density scheme with respect to $\{W_1, \ldots, W_k\}$. Referring to our discussion above, we are going to group the "similar" vertices together. The clustering procedure is described precisely below, but first we start with some definitions.

We denote by Φ_i^p the restriction of Φ_i to the domain $[r_i] \setminus \{p\}$. Namely, Φ_i^p is the set of all mappings from $[r_i] \setminus \{p\}$ to $[k]$. Fix an edge set E_i, an index $p \in [r_i]$, a vertex $v \in Y^0$ and a mapping $\phi \in \Phi_i^p$. Let $\Gamma_{i,\phi}^p(v)$ denote the set

$$\left\{(v_1, \ldots, v_{r_i}) \in E_i : \left(\forall_{j \in [r_i] \setminus \{p\}}\ v_j \in W_{\phi(j)}\right) \wedge \left(v_p = v\right)\right\}$$

Intuitively, for every extension $\phi' \in \Phi_i$ of ϕ, the size of $\Gamma_{i,\phi}^p(v)$ measures how the density $\mu_{i,\phi'}^\pi$ is affected when putting the vertex v in the set $V_{\phi'(p)}$. Since we are interested in the normalized densities of such edges, we define

$$\gamma_{i,\phi}^p(v) = \frac{|\Gamma_{i,\phi}^p(v)|}{n^{r_i - 1}}$$

Let $\gamma(v) = \langle \gamma_{i,\phi}^p(v) \rangle_{i \in [s], p \in [r_i], \phi \in \Phi_i^p}$ denote the density tensor of the vertex v considering all possible edge sets, places and mappings with respect to the partition $\{W_1, \ldots, W_k\}$. This density tensor encodes all information on v that we will use in the following (going back to our previous discussion, "similar" vertices are the vertices that have "similar" density tensors). Now we build an auxiliary set of quantitative density tensors. First let $\mathcal{Q}_\varepsilon = \left\{0, \frac{\varepsilon}{32}, \frac{2\varepsilon}{32}, \ldots, 1\right\}$ be the set of integer multiples of $\frac{\varepsilon}{32}$ in $[0, 1]$. Then the set of possible clusters is defined as

$$\mathcal{A} = \{\bar{\alpha} = \langle \alpha_{i,\phi}^p \rangle_{i \in [s], p \in [r_i], \phi \in \Phi_i^p} : \forall_{i \in [s], p \in [r_i], \phi \in \Phi_i^p}\ \ \alpha_{i,\phi}^p \in \mathcal{Q}_\varepsilon\}$$

In this set of $(\frac{1}{\varepsilon})^{O(k^r \cdot s \cdot r)}$ clusters (recall that r is the maximal arity of the edge sets) we group together the vertices that have approximately (up to $\frac{\varepsilon}{16}$) the same density tensors. The precise clustering of a vertex $v \in Y^0$ is performed as follows. We say that the tensor $\gamma(v)$ of v is *close* to the cluster $\bar{\alpha} \in \mathcal{A}$ if for every i, ϕ, p we have $\gamma^p_{i,\phi}(v) = \alpha^p_{i,\phi} \pm \varepsilon/32$. Note that one tensor $\gamma(v)$ can be close to multiple clusters. The final clustering will be based on sampled approximate tensors of H's vertices (rather than the real tensors), and all we will require is that almost every vertex v goes to a cluster that is close to $\gamma(v)$.

In the following we show that redistributing vertices from the same cluster cannot change by much the resulting edge densities from those of the initial partition of V.

Redistributing vertices from the same cluster

Let $\Pi = \{V_1, \dots, V_k\}$ be any fixed partition of V, and let $Y^0 \subset V$ be a set of at most $\frac{\varepsilon}{8}n$ vertices. As above, denote by $\{W_1, \dots, W_k\}$ the partition induced by Π on $V \setminus Y^0$. Let $Y^{0,\bar{\alpha}}$ be a subset of Y^0 such that all but at most $\xi|Y^{0,\bar{\alpha}}|$ vertices $v \in Y^{0,\bar{\alpha}}$ are close to the same cluster $\bar{\alpha}$ with respect to the partition $\{W_1, \dots, W_k\}$. This means that there is a tensor $\bar{\alpha} = \langle \alpha^p_{i,\phi} \rangle_{i \in [s], p \in [r_i], \phi \in \Phi^p_i}$ such that for all but at most $\xi|Y^{0,\bar{\alpha}}|$ vertices $v \in Y^{0,\bar{\alpha}}$ we have $\forall_{i \in [s], p \in [r_i], \phi \in \Phi^p_i} \ \gamma^p_{i,\phi}(v) = \alpha^p_{i,\phi} \pm \varepsilon/32$. We claim that if the vertices of such a $Y^{0,\bar{\alpha}}$ are redistributed over the k components of Π in a way that approximately preserves the number of vertices in every component, then the edge densities (according to any index i and mapping $\phi \in \Phi_i$) remain almost the same as before the redistribution of $Y^{0,\bar{\alpha}}$. Formally:

Claim 5.7.5. *Let $(Y^{0,\bar{\alpha}}_1, \dots, Y^{0,\bar{\alpha}}_k)$ be any partition of $Y^{0,\bar{\alpha}}$ that for each $1 \le j \le k$ satisfies $\big||V_j \cap Y^{0,\bar{\alpha}}| - |Y^{0,\bar{\alpha}}_j|\big| < \eta|Y^{0,\bar{\alpha}}|$, and let $\hat{\Pi} = \{\hat{V}_1, \dots, \hat{V}_k\} = \{W_1 \cup Y^{0,\bar{\alpha}}_1, \dots, W_k \cup Y^{0,\bar{\alpha}}_k\}$ be the redistributed partition of V. Then the following holds.*

- $\forall_{i \in [k]}, \ \big||V_j| - |\hat{V}_j|\big| < \eta|Y^{0,\bar{\alpha}}|$
- $\forall_{i \in [s], \phi \in \Phi_i}, \quad \Big||E^{\Pi}_{i,\phi}| - |E^{\hat{\Pi}}_{i,\phi}|\Big| \le (\frac{3\varepsilon}{16} + \xi + \eta)|Y^{0,\bar{\alpha}}|n^{r_i - 1}$

Proof. The first inequality follows directly from our assumptions, so we move to the second one. Fix i and ϕ. We call an edge e *relevant* with respect to Π, i and ϕ if $e \in E^{\Pi}_{i,\phi}$. Our purpose is to show that for all i and ϕ, the size of $E^{\Pi}_{i,\phi}$ is close to the size of $E^{\hat{\Pi}}_{i,\phi}$. The amount of relevant edges which have no vertices in $Y^{0,\bar{\alpha}}$ at all is preserved. We partition the rest of the relevant edges into two types:

$$I_1 = \{e \in (E^{\Pi}_{i,\phi} \cup E^{\hat{\Pi}}_{i,\phi}) : |e \cap Y^{0,\bar{\alpha}}| = 1\}$$

$$I_{\ge 2} = \{e \in (E^{\Pi}_{i,\phi} \cup E^{\hat{\Pi}}_{i,\phi}) : |e \cap Y^{0,\bar{\alpha}}| \ge 2\}$$

Clearly the size of $I_{\ge 2}$ is bounded by $|Y^{0,\bar{\alpha}}|^2 n^{r_i - 2} \le \frac{\varepsilon}{8}|Y^{0,\bar{\alpha}}|n^{r_i - 1}$. As for I_1, its influence on the change in the density may be caused by one of the following.

- At most $\xi|Y^{0,\bar{\alpha}}|$ vertices that do not reside in the same cluster as the others. These vertices can participate in at most $\xi|Y^{0,\bar{\alpha}}|n^{r_i-1}$ relevant edges.
- The difference between the vertices that reside in the same cluster. This can change the number of relevant edges by at most $\frac{\varepsilon}{16}|Y^{0,\bar{\alpha}}|n^{r_i-1}$.
- The difference in the sizes of the components before and after the redistribution. Since these differences are bounded by $\eta|Y^{0,\bar{\alpha}}|$, the change in the number of crossing edges due to them is at most $\eta|Y^{0,\bar{\alpha}}|n^{r_i-1}$.

Summing up, the difference in the number of relevant edges before and after the redistribution is at most $(\frac{3\varepsilon}{16} + \xi + \eta)|Y^{0,\bar{\alpha}}|n^{r_i-1}$. □

Redistributing general sets of vertices

Until now we referred to sets $Y^{0,\bar{\alpha}}$ in which almost all vertices are similar. Now we turn to the case where we need to redistribute an arbitrary set Y^0 of size at most $\frac{\varepsilon}{8}n$ (let the V_j's, W_j's and the $\hat{V}_j$'s denote the sets of the above partitions with respect to Y^0). Assume for now that we have an oracle that provides us the following information:

(i) A clustering $\{Y^{0,\bar{\alpha}_1}, \ldots, Y^{0,\bar{\alpha}_c}\}$ of Y^0, such that all but at most an $\varepsilon/4$ fraction of the vertices in Y^0 are assigned to a close cluster with respect to $\{W_1, \ldots, W_k\}$

(ii) For every $Y^{0,\bar{\alpha}_i}$ and V_j, the approximate normalized size of $|Y^{0,\bar{\alpha}_i} \cap V_j|$. Namely, for each $Y^{0,\bar{\alpha}_i}$ and V_j define $\zeta_j^{\bar{\alpha}_i} \triangleq \frac{|Y^{0,\bar{\alpha}_i} \cap V_j|}{|Y^{0,\bar{\alpha}_i}|}$ as the true intersection size. Then the oracle provides a sequence $\left(\beta_j^{\bar{\alpha}_i}\right)_{i\in[c], j\in[k]}$ of reals such that

$$\sum_{j=1}^{k}\sum_{i=1}^{c}\left(\left|\zeta_j^{\bar{\alpha}_i} - \beta_j^{\bar{\alpha}_i}\right| \cdot |Y^{0,\bar{\alpha}_i}|\right) < \frac{\varepsilon}{4} \cdot |Y^0|$$

Denoting by ξ_h the fraction of vertices in every $Y^{0,\bar{\alpha}_h}$ that are not close to the cluster $\bar{\alpha}_h$, we have $\sum_h \xi_h |Y^{0,\bar{\alpha}_h}| \le \frac{\varepsilon}{4}|Y^0|$, and denoting by η_h the sum $\sum_{j=1}^{k}\left|\zeta_j^{\bar{\alpha}_h} - \beta_j^{\bar{\alpha}_h}\right|$ we have $\sum_h \eta_h |Y^{0,\bar{\alpha}_h}| \le \frac{\varepsilon}{4}|Y^0|$. Then, after any redistribution of the vertices of each $Y^{0,\bar{\alpha}_h}$ according to the sequence $\left(\beta_j^{\bar{\alpha}_i}\right)_{i\in[c], j\in[k]}$, from (ii) we have:

$$\forall_{j\in[k]} : \left||V_j| - |\hat{V}_j|\right| < \frac{\varepsilon}{4}|Y^0| \le \frac{\varepsilon^2}{32}n$$

and according to (i), by Claim 5.7.5 we have:

$$\forall_{i\in[s], \phi\in\Phi_i} : \left||E_{i,\phi}^{\Pi}| - |E_{i,\phi}^{\hat{\Pi}}|\right| \le \sum_h (\frac{3\varepsilon}{16} + \xi_h + \eta_h)|Y^{0,\bar{\alpha}_h}|n^{r_i-1} \le \frac{\varepsilon}{2}|Y^0|n^{r_i-1} \le \frac{\varepsilon^2}{16}n^{r_i}$$

In terms of density tensors, we have: $\forall_{j\in[k]} : |\rho_j^{\Pi} - \rho_j^{\hat{\Pi}}| \le \frac{\varepsilon^2}{32}$ and $\forall_{i\in[s],\phi\in\Phi_i} : |\mu_{i,\phi}^{\Pi} - \mu_{i,\phi}^{\hat{\Pi}}| \le \frac{\varepsilon^2}{16}$.

The only thing left is to describe how to simulate the oracles for (i) and (ii) above. As for the second item (ii), we can just provide corresponding partitions for all possible sets of intersection sizes as follows. Let

$$\mathcal{B} = \Big\{ \bar{\beta} = \big(\beta_1^{\bar{\alpha}}, \ldots, \beta_k^{\bar{\alpha}}\big)_{\bar{\alpha}\in\mathcal{A}} : (\beta_j^{\bar{\alpha}} \in \{0, \frac{\varepsilon}{32}, \frac{2\varepsilon}{32}, \ldots, 1\}) \wedge (\sum_j \beta_j^{\bar{\alpha}} = 1) \Big\}$$

The size of $\mathcal{B}$ is bounded from above by $(32/\varepsilon)^{k\cdot|\mathcal{A}|} = \exp\left(\varepsilon^{-O(s\cdot r\cdot k^r)}\right)$. For every possible $\bar{\beta} \in \mathcal{B}$, we partition the vertices of every $Y^{0,\bar{\alpha}}$ into the k components according to the distribution defined by $\big(\beta_1^{\bar{\alpha}}, \ldots, \beta_k^{\bar{\alpha}}\big)$, i.e. when queried about a vertex $v \in Y^{0,\bar{\alpha}}$ we put it into V_j with probability $\beta_j^{\bar{\alpha}}$, for every $j \in [k]$. Observe that there exists $\bar{\beta} = \big(\beta_1^{\bar{\alpha}}, \ldots, \beta_k^{\bar{\alpha}}\big)_{\bar{\alpha}\in\mathcal{A}} \in \mathcal{B}$ which approximates the true intersection values $\big(\zeta_1^{\bar{\alpha}}, \ldots, \zeta_k^{\bar{\alpha}}\big)_{\bar{\alpha}\in\mathcal{A}}$ even closer than needed, and then for $n = \omega(s\cdot k^r)$, with probability $1 - o(1)$, redistributing according to this vector $\bar{\beta}$ will make the intersection sizes as close as required. Note that this avoids having to know the actual sizes of the intersections of $Y^{0,\bar{\alpha}}$ with the components $V_1, \ldots, V_k$ while constructing a partial partition oracle for the vertices in Y^0.

For the first item (i), we are going to choose a random set U of vertices in $V \setminus Y^0$, and approximate the clusters of the vertices in Y^0 according to their density tensors with respect to the partition of U induced by Π. Once again, since we do not know the partition Π, we shall try all possible partitions of U as follows. Let $U \in (V \setminus Y^0)$ be a set of size T. We denote by $\mathcal{P}_U$ the set of all possible k-way partitions of U, namely $\mathcal{P}_U = \{\Pi_U : \Pi_U$ is a k-way partition of $U\}$. Note that the size of $\mathcal{P}_U$ is at most k^T. We are going to partition the set U according to each one of the partitions in $\mathcal{P}_U$, and clearly one of them is the correct one, which is the one that is induced by the initial partition $\Pi = \{V_1, \ldots, V_k\}$. The following lemma states that sampling from a small set U gives a good approximation of the clusters for most of the vertices in Y^0.

Lemma 5.7.6. *Let $\Pi = \{V_1, \ldots, V_k\}$ be a partition of V, let $Y^0 \subset V$ be a set of size $\frac{\varepsilon}{8}n$, and let $U \subset V \setminus Y^0$ be a uniformly chosen random set of $r \cdot t = r \cdot \frac{2^{16}}{\varepsilon^2} \log \frac{l\cdot s\cdot r\cdot k^r}{\varepsilon}$ vertices (possibly with repetitions). There is a (deterministic) algorithm that outputs a sequence of $k^{rt} \cdot |\mathcal{B}| = \exp\left(\varepsilon^{-O(s\cdot r\cdot k^r)}\right)$ partial oracles, that have shared query complexity $s \cdot t$ and that query only edges contained in $U \cup Y^0$ to provide presumed density tensors for vertices in Y^0. With probability at least $1 - 1/(2l)$ over the choice of U at least one of the oracles clusters the vertices of Y^0 so that all but at most $\frac{\varepsilon}{4}|Y^0|$ of them are clustered correctly with respect to Π and $\mathcal{A}$. Moreover, both the production and the operation of these oracles do not depend on Π at all.*

Estimating the clusters by sampling – Proof of Lemma 5.7.6

Let $\Pi = \{V_1, \ldots, V_k\}$ (we first assume here that the underlying partition of the vertices is known and later show how to remove this dependency), let $\{W_1, \ldots, W_k\} = \{V_1 \setminus Y^0, \ldots, V_k \setminus Y^0\}$, and let us fix a vertex $v \in Y^0$, an edge set E_i, $i \in [s]$, a position $p \in [r_i]$ and a mapping $\phi \in \Phi_i^p$. We are going to estimate the value of $\gamma_{i,\phi}^p(v)$ by sampling random sets of vertices from V, and calulating the fraction of the relevant edges within the chosen random subsets and our fixed vertex v. Recall that

$$\gamma_{i,\phi}^p(v) \triangleq \frac{1}{n^{r_i-1}}\left|\left\{(v_1, \ldots, v_{r_i}) \in E_i : \left(\forall_{j\in[r_i]\setminus\{p\}}\ v_j \in W_{\phi(j)}\right) \wedge \left(v_p = v\right)\right\}\right|$$

We should now note the following: If we choose uniformly, independently and with repetition a sequence of $r_i - 1$ vertices $v_1, \ldots, v_{p-1}, v_{p+1}, \ldots, v_{r_i}$, and denote by F the event that for every $i \neq p$ between 1 and r_i we have $v_i \in W_{\phi(i)}$ and $(v_1, \ldots, v_{p-1}, v, v_{p+1}, \ldots, v_{r_i}) \in E_i$, then the probability of F is exactly $\gamma_{i,\phi}^p(v)$. Now, we would like to repeat this step $t = \frac{2^{16}}{\varepsilon^2} \log \frac{l\cdot s\cdot r\cdot k^r}{\varepsilon}$ times independently, and compute $\hat{\gamma}_{i,\phi}^p(v)$, which is the fraction of the number of times where the event F occurred (later we will show what to do here without prior knowedge of $W_1, \ldots, W_k$). Applying the additive Chernoff bound we have

$$\Pr\left[|\hat{\gamma}_{i,\phi}^p(v) - \gamma_{i,\phi}^p(v)| > \frac{\varepsilon}{32}\right] < 2\cdot\exp(-2(\frac{\varepsilon}{32})^2 t) < \frac{\varepsilon}{32\cdot l\cdot s\cdot r\cdot k^r}$$

To be able to classify v completely by our oracle, we first choose uniformly and independently (with repetitions, although those would clearly occur with probability $o(1)$) a sequence $U = \{w_1, \ldots w_{(r-1)t}\}$ of $(r-1)t$ vertices. For a requested vertex v, we can for every $i \in [s]$, $p \in [r_i]$ and $\phi \in \Phi_i^p$ compute the estimation $\hat{\gamma}_{i,\phi}^p(v)$ by dividing U into subsequences of length $r_i - 1$ and using the first t of them in the estimation procedure above. By the union bound, the probability for a given v that there exists *any* i, p and ϕ for which the deviation is more than $\frac{\varepsilon}{32}$ is bounded by $\frac{\varepsilon}{32l}$. Hence, by the Markov inequality, with probability at least $1 - \frac{1}{3l}$ the sequence U is such that for all vertices $v \in Y^0$ but at most an $\varepsilon/4$ fraction of them, we have $\hat{\gamma}_{i,\phi}^p(v) = \gamma_{i,\phi}^p(v) \pm \frac{\varepsilon}{32}$ for all i, p and ϕ. We also note that the number of input queries that we need in order to produce the classification of v is clearly bounded by $t\cdot s\cdot r\cdot k^r$

Now, to remove the dependency on Π we still have to deal with the problem of not knowing $W_1, \ldots, W_k$. We do know which vertices are not in any W_i at all (because we know Y^0, which is given by the algorithm and is not dependent on the input hypergraph), so we can classify those vertices from U which are not in these sets. For the other vertices, we have no choice but to construct appropriate oracles for any possibility of partitioning them into k sets, as one of these partitions would be the correct one conforming to $W_1, \ldots, W_k$.

Here it is important that U is chosen only once, and then re-used for all our oracles in all of their oracle queries.

5.7.3 Some possible ad-hoc optimizations for properties

In all our treatment, we made no attempt to optimize the dependency functions themselves but only to optimize the function types. Moreover, we applied the general versions of Theorem 5.2.1 and Theorem 5.2.2 also for properties where clearly our requirements on the partition tensor ψ^{Π} are rather sparse. Before we conclude the proof details for these theorems, this is a good place to sketch how we can use such specific features of special cases to substantially optimize the algorithm.

For several of the special cases discussed in this chapter there is a way to ignore many of the values $\gamma_{i,\phi}^{p}(v)$, thus reducing the size of $\mathcal{A}$ and obtaining much shorter running times. Here we sketch how it can be done for two of the properties.

k-colorability of r-uniform hypergraphs: We note that this partition property depends only on the density of edges *inside* every partition component V_j. These are only affected by edges including a vertex v whose other $r-1$ vertices are in the same V_j. In other words, we need only to count the values $\gamma_{1,\phi}^{p}(v)$ for which $\phi : [r] \setminus \{p\} \to [k]$ is a constant function, and there are only rk of those. Moreover, taking into consideration that this is a property of simple (undirected) hypergraphs, we can also do away with the index p. Thus, the power of $1/\varepsilon$ that is hidden in the complexity estimates of the colorability algorithm would be $\tilde{O}(k)$ rather than depend on k^r (only multiplicative coefficients would depend on r).

A similar optimization, to a somewhat lesser extent, are also possible for the problem of estimating the number of satisfiable clauses in a k-CNF instance.

Optimizations for Theorem 5.4.2: The property Ψ used in its proof (see Section 5.4.3) concerns hypergraphs with 4-edges, but these are always between two sets. Therefore, we only need to consider values $\gamma_{2,\phi}^{p}(v)$ where the function ϕ has a range of size 2. Along with some additional care taken in the proof itself, it appears possible to replace the $\hat{k}^5$ in the estimates there with $\hat{k}^{2+o(1)}$.

5.7.4 The last missing details of the proof

From ε'-approximation to ε-closeness – Proof of Lemma 5.7.2

In this section we show that any partition Π, which $\varepsilon' = \frac{\varepsilon}{2r}$-approximately satisfies Ψ, can be turned into a partition which ε-closely satisfies Ψ by correcting the number of vertices in each component, where the correction is done by moving around a minimal number of vertices according to some close density tensor in Ψ. The idea behind this observation is that a partition which ε'-approximately satisfies Ψ can be corrected (by definition) by moving vertices in a way that every V_j either gains up to $\varepsilon' n$ vertices or loses up to $\varepsilon' n$ vertices (but not both). Now we show that as a consequence of such moves the difference in the measured edge densities before and after the correction are small enough.

Formally, let Π be a partition which ε'-approximately satisfies Ψ, let ψ^{Π} be the density tensor of Π and let $\psi \in \Psi$ be the density tensor which is ε'-approximately satisfied by Π. By definition:

- for all $j \in [k]$, $\quad \rho_j^{\Pi} = \rho_j \pm \varepsilon'$
- for all $i \in [s]$ and $\phi \in \Phi_i$, $\quad \mu_{i,\phi}^{\Pi} = \mu_{i,\phi} \pm \varepsilon'$

Let $\hat{\Pi}$ be the *corrected* partition, which results from the following process: Let S be a temporary set of vertices, which is initially empty. For each $j \in [k]$ such that $\rho_j^{\Pi} > \rho_j$, we take $|V_j^{\Pi}| - \rho_j \cdot n$ vertices from the component V_j^{Π} of Π, and put them into the set S. Then, for each $j \in [k]$ such that $\rho_j^{\Pi} < \rho_j$, we move $\rho_j \cdot n - |V_j^{\Pi}|$ vertices from S to the component V_j.

We are going to prove that the partition $\hat{\Pi}$ ε-closely satisfies ψ. Recall that by definition of ε-closely satisfying, we need to show that:

- for all $j \in [k]$, $\quad \rho_j^{\hat{\Pi}} = \rho_j$
- for all $i \in [s]$ and $\phi \in \Phi_i$, $\quad \mu_{i,\phi}^{\hat{\Pi}} = \mu_{i,\phi} \pm \varepsilon$

The first item holds by the definition of $\hat{\Pi}$, so we just need to prove the second one. Let us fix $i \in [s]$ and $\phi \in \Phi_i$ and prove that $\mu_{i,\phi}^{\hat{\Pi}} = \mu_{i,\phi} \pm \varepsilon$.

Let $j_h = \phi(h)$ for all $h \in [r_i]$, and let $V_{j_1}, V_{j_2}, \ldots, V_{j_{r_i}}$ be the set (with possible repetitions) of components of Π that participate in the edges from $|E_{i,\phi}^{\Pi}|$. Observe that any vertex $v \in V_{j_h}$ can participate in the h'th place of at most $n^{r_i - 1}$ edges of $|E_{i,\phi}^{\Pi}|$. On the other hand, the number of vertices that were added or removed from each V_{j_h} is at most $\varepsilon' n$. Therefore, $\left| |E_{i,\phi}^{\Pi}| - |E_{i,\phi}^{\Pi}| \right| \leq \varepsilon' \cdot r_i \cdot n^{r_i}$, and consequently, $\mu_{i,\phi}^{\hat{\Pi}} = \mu_{i,\phi}^{\Pi} \pm \varepsilon/2$. Combining it with $\mu_{i,\phi}^{\Pi} = \mu_{i,\phi} \pm \frac{\varepsilon}{2r}$ we conclude that $\mu_{i,\phi}^{\hat{\Pi}} = \mu_{i,\phi} \pm \varepsilon$ as required.

Testing proximity to Ψ – Proof of Lemma 5.7.3

Proof. The main idea is straightforward. Fix one of the partition oracles π, and let Π denote the partition induced by it. Our algorithm will estimate the following measures by sampling:

- for all $j \in [k]$, we compute a value ρ_j^S that estimates the normalized size of the component V_j in the partition Π.

- for every $i \in [s]$ and $\phi \in \Phi_i$, we compute a value $\mu^S_{i,\phi}$ that estimates the corresponding hyperedge density.

In the next step, we check these estimated values as follows. Let ψ^S denote the density tensor corresponding to the estimated values above. If ψ^S $\frac{3}{4}\varepsilon$-approximately satisfies Ψ, then we output $\psi \in \Psi$ that is $\frac{3}{4}\varepsilon$-approximated by ψ^S (recall that finding such ψ takes $O(TC(\Psi))$ time). Otherwise, we move to the next partition oracle from the sequence $\{\pi_i\}_{i=1}^m$. If we did not find any $\frac{3}{4}\varepsilon$-approximately satisfying partition, then we output *False*.

We now define and analyze the estimation procedure formally. First we choose a sequence T_v (with possible repetitions) of $t = (\frac{4}{\varepsilon})^2 \cdot \log\left(\frac{4m\cdot(sk^r+k)}{\delta}\right)$ vertices uniformly at random, and we choose one additional sequence T_e of $r \cdot t$ vertices in a similar fashion. Next we make the following queries and write down their results.

- For every partition oracle π and every vertex $v \in T_v \cup T_e$ we compute $\pi(v)$. This requires making $O(r \cdot t \cdot f)$ queries.
- For every $i \in [s]$ we split the first $r_i \cdot t$ vertices of T_e into t consequent r_i-tuples $\{\xi^h = (v^h_1, \dots, v^h_{r_i})\}_{h=1}^t$, and for every tuple $\xi^h = (v^h_1, \dots, v^h_{r_i})$ we check if $(v^h_1, \dots, v^h_{r_i}) \in E_i$. This requires making $O(t \cdot s)$ queries.

Summing up, the total number of queries we make is bounded by

$$O\Big(t \cdot (r \cdot f + s)\Big) = O\bigg((r \cdot f + s) \cdot (\frac{1}{\varepsilon})^2 \cdot \log(\frac{m \cdot sk^r}{\delta})\bigg)$$

Now, once we made all required queries, for every partition oracle π we perform the following.

- For every $j \in [k]$ we set

$$\rho^S_j = \frac{1}{t}|\{v \in T_v : \pi(v) = j\}|$$

- For every $i \in [s]$ and $\phi \in \Phi_i$, we set

$$\mu^S_{i,\phi} = \frac{1}{t}\bigg|\Big\{\xi^h = (v^h_1, \dots, v^h_{r_i}) : \Big((v^h_1, \dots, v^h_{r_i}) \in E_i\Big) \wedge \Big(\forall_{j\in[r_i]}\pi(v_j) = \phi(j)\Big)\Big\}\bigg|$$

For each $j \in [k]$, let F_j denote the event $|\rho^S_j - \rho^\Pi_j| \geq \varepsilon/4$. Similarly, for every $i \in [s]$ and $\phi \in \Phi_i$, let $F_{i,\phi}$ denote the event $|\mu^S_{i,\phi} - \mu^\Pi_{i,\phi}| \geq \varepsilon/4$. Observe that if with probability $1 - \frac{\delta}{2}$ none of these "bad" events occur for any one of the partition oracles π, then algorithm A_S satisfies the assertions of the lemma. We concentrate on a somewhat simpler case (which implies this one by the union bound), where we want to bound the probability for one specific π and one specific event F of this type by $\frac{\delta}{2m(sk^r+k)}$.

Note that for every $v \in T_v$ and $j \in [k]$, $\Pr[\pi(v) = j] = \rho_j^\Pi$. Similarly, for every r_i-tuple $\xi^h = (v_1^h, \ldots, v_{r_i}^h)$ from T_e, $i \in [s]$ and $\phi \in \Phi_i$,

$$\Pr\left[\Big((v_1^h, \ldots, v_{r_i}^h) \in E_i\Big) \wedge \Big(\forall_{j\in[r_i]}\pi(v_j) = \phi(j)\Big)\right] = \mu_{i,\phi}^\Pi$$

Since the vertices in T_v and T_e were chosen independently at random, we can apply Chernoff's inequality to bound the probability of the event F.

$$\Pr[F] \le 2 \cdot \exp(-2 \cdot (\varepsilon/4)^2 \cdot t) \le \frac{\delta}{2m \cdot (s \cdot k^r + k)}$$

as required.

Since the operations of A_S consume negligible running time, its time complexity is bounded by m times the query complexity, plus m times the time required for checking proximity to Ψ. So in total, the time complexity of A_S is bounded by

$$O\bigg(m \cdot c \cdot (r \cdot f + s) \cdot \Big((\frac{1}{\varepsilon})^2 \cdot \log(\frac{m \cdot sk^r}{\delta})\Big)\bigg) + m \cdot TC(\Psi)$$

□

5.8 Future work

Strong hypergraph regularity: Very recently, a lot of attention was given to obtaining hypergraph regularity lemmas [Gow97, NRS06, RS93, Tao06] that are strong enough to reprove Szemerédi's number-theoretic theorem [Sze75] and to prove combinatorial deletion results. As it turns out, in these lemmas one needs to consider not only partitions of the vertices of the graph, but also partitions of pairs of vertices, triples of vertices and so on. Furthermore, one needs to consider various requirements on the interactions between the partitions of different arities concerning their densities and beyond (e.g. on the count of certain small substructures). See for example [HNR05] for an algorithmic version of a strong regularity lemma for 3-uniform hypergraphs.

We believe that we can extend our main result about vertex partitions to the case of partitions of pairs and beyond, but (regretfully) we currently cannot use them to give algorithmic versions of the new variants of the regularity lemma discussed above. The main reason is that our "control" of the interactions between partitions of different arities is not strong enough to match the one typically required for a strong hypergraph regularity lemma. It would be very interesting to obtain a generalization of Theorem 5.2.1 that will be strong enough to yield algorithmic versions of the results of [Gow97, NRS06, RS93, Tao06].

Improving the output for hypergraph regularity: While we can apply Theorem 5.2.1 in a simple way to obtain an algorithmic version of the regularity lemma for graphs (given in Theorem 5.4.2), the generalization for hypergraphs (Theorem 5.5.1) is more complicated, and has worse output guarantees as it will generally not find small regular partitions if they exist. It would be interesting to see if one can prove a variant of Theorem 5.5.1 similar in nature to Theorem 5.4.2, perhaps by formulating and proving an adequate notion of "local regularity" first.

Part II

Massively Parameterized Properties

Chapter 6

Introduction

6.1 The orientation model

In this part of the thesis we concentrate on the Orientation model introduced in [HLNT05]. In this model the massive-parameter, i.e. the information given in advance, is an underlying undirected graph $G = (V, E)$. The input is then constrained to be an orientation of G, and the distances are measured relative to $|E|$ and not to any function of $|V|$. An orientation of G is simply an orientation of its edges. That is, for every edge $e = \{u, v\}$ in $E(G)$ an orientation of G specifies which of u and v is the source vertex of e, and which is the target vertex. Thus an orientation defines a directed graph $\overrightarrow{G}$ whose undirected skeleton is G. Given the undirected graph G, a property of orientations is just a subset of the set of all $2^{|E|}$ possible orientations of G.

In the context of property testing, the relevant combinatorial structure to be tested is an orientation $\overrightarrow{G}$ of the underlying graph G, and the distance between two orientations $\overrightarrow{G}_1$, $\overrightarrow{G}_2$ is the number of edges that are oriented differently in $\overrightarrow{G}_1$ and $\overrightarrow{G}_2$. Thus an orientation $\overrightarrow{G}$ is *ε-far* from a given property P if at least $\varepsilon|E(G)|$ edges have to be redirected in $\overrightarrow{G}$ to make it satisfy P. Ideally the number of queries that the property tester makes depends only on ε and on nothing else (in particular it depends neither on $|E|$ nor on the specific undirected graph G itself), but as we shall see, this is not always the case, even if we consider properties that have very small negative witnesses.

A common feature of the results (ours and others) in the orientation model is that the algorithms themselves are rather non-trivial in their construction, and not just in their analysis. This feature distinguishes them from results in many other areas of property testing, such as the dense graph models. In particular, the testing algorithms in this model do not readily yield to general techniques such as that of the regularity lemma used in the dense graph model.

In this part of the thesis we study the query complexity of two natural graph properties in the orientation model. The first is the property of st-connectivity, where s and t are two special vertices, and a digraph satisfies the property if there is a directed path between s and t. In Chapter 7 we show that st-connectivity can be tested with constant (depending only on the distance parameter ε) number of queries. Then, in Chapter 8, we study the property of being Eulerian, namely that of all vertices having the same in-degree as their out-degree. Although this property has a more "local" nature than st-connectivity, our results show that it is harder to test in the orientation model. In particular, in Chapter 8 we show that testing orientations for being Eulerian requires making $\omega(1)$ queries, even if two sided error is allowed and the underlying graph G has maximal degree 4 (note that in this case there is always a negative witness of size 4). Before proceeding to these chapters, we give some common notations and definitions in the following section.

6.2 Specific definitions and preliminaries

6.2.1 Notations

In what follows, our graphs are going to possibly include parallel edges, so we use 'named' pairs for edges, i.e. we use $e = e\{u, v\}$ for an undirected edge named e whose end points are u and v. Similarly, we use $e = e(u, v)$ to denote the directed edge named e that is directed from u to v. Let $G = (V, E)$ be an undirected multi-graph (parallel edges are allowed), and denote by n the number of vertices in V. We say that a directed graph $\overrightarrow{G}$ is an *orientation* of the graph G, or in short a G-orientation, if we can derive $\overrightarrow{G}$ from G by replacing every undirected edge $e = e\{u, v\} \in E$ with either $e(u, v)$ or $e(v, u)$, but not both. We also call G the *underlying graph* of $\overrightarrow{G}$.

Given an undirected graph G and a subset $W \subset V$ of G's vertices, we denote by $G(W)$ the induced subgraph of G on W, and we denote by $E(W) = E(G(W))$ the edge set of $G(W)$. The distance between two vertices u, v in G is denoted by $\mathrm{dist}_G(u, v)$ and is set to be the length of the shortest path between u and v. Similarly, for a directed graph $\overrightarrow{G}$, $\mathrm{dist}_{\overrightarrow{G}}(u, v)$ denotes the length of the shortest *directed* path from u to v. The distance of a vertex from itself is $\mathrm{dist}_{\overrightarrow{G}}(v, v) = \mathrm{dist}_G(v, v) = 0$. In the case where there is no directed path from u to v in $\overrightarrow{G}$, we set $\mathrm{dist}_{\overrightarrow{G}}(u, v) = \infty$. The diameter of an undirected graph G is defined as $\mathrm{diam}(G) = \max_{u,v \in V}\{\mathrm{dist}_G(u, v)\}$.

For a graph $\overrightarrow{G}$ and a vertex $v \in V$, let $\Gamma_{in}(v) = \{u : \exists e(u, v) \in E\}$ and $\Gamma_{out}(v) = \{u : \exists e(v, u) \in E\}$ be the set of incoming and outgoing neighbors of v respectively, and let $\Gamma(v) = \Gamma_{in}(v) \cup \Gamma_{out}(v)$ be the set of neighbors in the underlying graph G. Let $\deg_{in}(v)$, $\deg_{out}(v)$ and $\deg(v)$ denote the sizes of $\Gamma_{in}(v)$, $\Gamma_{out}(v)$ and $\Gamma(v)$ respectively. We denote the *i-neighborhood* (in the underlying undirected graph G) of a vertex v by $N_i(v) = \{u : \mathrm{dist}_G(u, v) \leq i\}$. For example, $N_1(v) = \{v\} \cup \Gamma(v)$, and for all $v \in V$, $V = N_{\mathrm{diam}(G)}(v)$.

6.2.2 Orientation distance, properties and testers

Given two G-orientations $\overrightarrow{G}_1$ and $\overrightarrow{G}_2$, the *distance* between $\overrightarrow{G}_1$ and $\overrightarrow{G}_2$, denoted by $\Delta(\overrightarrow{G}_1, \overrightarrow{G}_2)$, is the number of edges in $E(G)$ having different directions in $\overrightarrow{G}_1$ and $\overrightarrow{G}_2$.

Given a graph G, a property $\mathcal{P}_G$ of G's orientations is a subset of all possible G-orientations. We say that an orientation $\overrightarrow{G}$ *satisfies* the property $\mathcal{P}_G$ if $\overrightarrow{G} \in \mathcal{P}_G$. The *distance* of $\overrightarrow{G}_1$ from the property $\mathcal{P}_G$ is defined by $\delta(\overrightarrow{G}_1, \mathcal{P}_G) = \min_{\overrightarrow{G}_2 \in \mathcal{P}_G} \frac{\Delta(\overrightarrow{G}_1, \overrightarrow{G}_2)}{|E(G)|}$. We say that $\overrightarrow{G}$ is ε*-far* from $\mathcal{P}_G$ if $\delta(\overrightarrow{G}, \mathcal{P}_G) \geq \varepsilon$, and otherwise we say that $\overrightarrow{G}$ is ε*-close* to $\mathcal{P}_G$. We omit the subscript G when it is obvious from the context.

Definition 18. [(ε, q)**-orientation tester**] *Let G be a fixed undirected graph and let P be a property of G's orientations. An (ε, q)*-tester *T for the property P is a randomized algorithm, that for any $\overrightarrow{G}$ that is given via oracle access to the orientations of its edges operates as follows.*

- *The algorithm T makes at most q orientation queries to $\overrightarrow{G}$ (where on a query $e \in E(G)$ it receives as an answer the orientation of e in $\overrightarrow{G}$).*
- *If $\overrightarrow{G} \in P$, then T accepts it with probability at least* $2/3$.
- *If $\overrightarrow{G}$ is ε-far from P, then T rejects it with probability at least* $2/3$.

The query complexity of an (ε, q)-tester T is the maximal number of queries q that T makes on any input. Here too, we say that T has *one-sided error* if it accepts every $\overrightarrow{G} \in P$ with probability 1. We also say that a property P is *testable* if for every $\varepsilon > 0$ it has an $(\varepsilon, q(\varepsilon))$-test, where $q(\varepsilon)$ is a function depending only on ε (and independent of the graph size n).

Chapter 7

Testing st-connectivity

7.1 Background

A major question that has remained open in [HLNT05] is whether connectivity properties admit a test in the orientation model. For a fixed $s, t \in V(G)$, an orientation $\overrightarrow{G}$ is *st-connected* if there is a directed path from s to t in it. Connectivity problems and in particular st-connectivity are very basic problems in graph theory and have been extensively studied in various models of computation.

Our main result in this chapter is that the property of being st-connected is testable by a one-sided error algorithm with a number of queries depending only on ε. That is, we construct a randomized algorithm such that for any underlying graph G, on input of an unknown orientation the algorithm queries only $O(1)$ edges for their orientation and based on this decides with success probability $\frac{2}{3}$ between the case that the orientation is st-connected and the case that it is ε-far from being st-connected. Our algorithm additionally has one-sided error, meaning that st-connected orientations are accepted with probability 1. Note that the algorithm knows the underlying graph G in advance and G is neither alterable nor part of the input to be queried. The dependence of the number of queries in our test on ε is triply exponential, but it is independent of the size of the graph.

To put this result in context with previous works in the area of property testing, we recall that the model that was mainly studied is the dense graphs model in which an input is a graph represented as a subgraph of the complete graph. As such, for n-vertex graphs, the input representation size is $\binom{n}{2}$ which is the number of all possible unordered pairs. Thus, any property that has $o(n^2)$ witness size, and in particular the property of undirected st-connectivity, is trivially testable as every input is close to the property. Properties of directed graphs were studied in the same context mostly by [AFNS06] and [AS05]. Inputs in this model are subgraphs of the complete directed graph (with or without anti parallel edges). In this case, directed st-connectivity is again trivial.

Apart from [HLNT05], the most related work is that of [HLNT07], in which a graph $G = (V, E)$ is given and the properties are properties of boolean functions $f : E(G) \to \{0, 1\}$. In [HLNT07] the interpretation of such a function is as an assignment to certain formulae that are associated with the underlying graph G, and in particular can viewed as properties of orientations (although the results in [HLNT07] concentrate on properties that are somewhat more "local" than our "global" property of being *st*-connected). Hence, the results here should be viewed as moving along the lines of the investigation that was started in [HLNT05] and [HLNT07].

The algorithm that we present here for *st*-connectivity involves a preprocessing stage that is meant to reduce the problem to that of testing a branching program of bounded width. Once this is achieved, a randomized algorithm simulating the test for constant width branching programs from [New02] is executed to conclude the result.

In general, the decision problem of *st*-connectivity of orientations of a given graph is not known to be reducible to constant width branching programs. In fact, it is most probably not the case, as *st*-connectivity is complete for NL (non-deterministic LOG space) while deciding constant width branching programs is in L.

In particular, it is not clear how to deal with high degree vertices or with large cuts. The purpose of the preprocessing stage is to get rid of these difficulties (here it will be crucial that we only want to distinguish between inputs that have the property and inputs that are quite far from having the property). This is done in several steps that constitute the main part of this chapter. In particular we have an interim result in which most but not all edges of the graph are partitioned into constant width layers. This is proved using a weak concentration lemma for sequences of integers, which is formulated and proved for this purpose.

After the small portion of edges not in constant width layers is dealt with (using a reduction based on graph contractions), we can reduce the connectivity problem to a constant width read once branching program. Once such a branching program is obtained, the result of [New02] can be used essentially in a black box manner.

Some interesting related open problems still remain. We still do not know if the property of being *strongly st*-connected is testable with a constant number of queries. The orientation $\overrightarrow{G}$ is *strongly st*-connected if there is a directed path in $\overrightarrow{G}$ from s to t as well as a directed path from t to s. A more general problem is whether in this model we can test the property of being *all-pairs strongly-connected* using a constant number of queries. Another related property is the property that for a given $s \in V(G)$ every vertex is reachable by a directed path from s. The complexity of these problems is unknown, although there are some indications that similar methods as those used here may help in this regard.

The rest of this chapter is organized as follows. Section 7.2 contains the statement of the main result and an overview of the proof. In Section 7.3 we reduce the problem of testing *st*-connectivity in general graphs to the problem of testing *st*-connectivity in nicely structured bounded-width graphs (we later call them *st-connectivity programs*). In Section 7.4 we reduce from testing *st*-connectivity programs to testing *clustered* branching programs, and then we show how to convert these clustered branching programs into regular ones, to which we can apply the testing algorithm from [New02]. Then in Section 7.5 we combine these ingredients to wrap up the proof. Finally, in Section 7.6 we provide the missing proofs for the auxiliary lemmas.

7.1.1 Connectivity programs and Branching programs

Our first sequence of reductions converts general *st*-connectivity instances to well structured bounded-width *st*-connectivity instances, as formalized in the next definition.

Definition 19 (*st*-Connectivity Program)**.** *An* *st*-Connectivity Program *of width w and length m over n vertices (or $CP(w,m,n)$ in short), is a tuple $\langle G,\mathcal{L}\rangle$, where G is an* undirected *graph with n vertices and $\mathcal{L}$ is a partition of G's vertices into layers $L_0,\dots,L_m$. There are two special vertices in G: $s \in L_0$ and $t \in L_m$, and the edges of G are going only between vertices in consecutive layers, or between the vertices of the same layer, i.e. for each $e = e\{u,v\} \in E(G)$ there exists $i \in [m]$ such that $u \in L_{i-1}$ and $v \in L_i$, or $u,v \in L_i$. The partition $\mathcal{L}$ induces a partition $E_1,\dots,E_m$ of $E(G)$, where E_i is the set of edges that have both vertices in L_i, or one vertex in L_{i-1} and another in L_i. In this partition of the edges the following is required in $CP(w,m,n)$: $\max_i\{|E_i|\} \le w$.*

Any orientation of G's edges (that maps every edge $e\{u,v\} \in E(G)$ to either $e(u,v)$ or $e(v,u)$) defines a directed graph $\overrightarrow{G}$ in the natural way. An st-connectivity program $C = \langle G,\mathcal{L},\rangle$ defines a property P_C of G's orientations in the following way: $\overrightarrow{G} \in P_C$ if and only if in the directed graph $\overrightarrow{G}$ there is a directed path from s to t.

Next we define branching programs. These are the objects to which we can apply the testing algorithm of [New02].

Definition 20 (Branching Program)**.** *A* Read Once Branching Program *of width w over an input of n bits (or $BP(w,n)$ in short), is a tuple $\langle G,\mathcal{L},X\rangle$, where G is a* directed *graph with 0/1-labeled edges, $\mathcal{L}$ is a partition of G's vertices into layers $L_0,\dots,L_n$ such that $\max_i\{|L_i|\} \le w$, and $X = \{x_0,\dots,x_{n-1}\}$ is a set of n Boolean variables. In the graph G there is one special vertex s belonging to L_0, and a subset $T \subset L_n$ of* accepting *vertices. The edges of G are going only between vertices in consecutive layers, i.e. for each $e = e(u,v) \in E(G)$ there is $i \in [n]$ such that $u \in L_{i-1}$ and $v \in L_i$. Each vertex in G has at most two outgoing edges, one of which is labeled by '0' and the other is labeled by '1'. In addition, all edges between two consecutive layers are associated with one distinct member of $X = \{x_0,\dots,x_{n-1}\}$. An assignment $\sigma : X \to \{0,1\}$ to X defines a subgraph G_σ of G,* *which has the same vertex set as G, and for every layer L_{i-1} (whose outgoing edges are associated with the variable x_{j_i}), the subgraph G_σ has only the outgoing edges labeled by $\sigma(x_{j_i})$. A read once branching program $B = \langle G, \mathcal{L}, X\rangle$ defines a property $P_B \subset \{0,1\}^n$ in the following way: $\sigma \in P_B$ if and only if in the subgraph G_σ there is a directed path from the starting vertex s to any of the accepting vertices in T.*

Branching programs that comply with the above definition can be tested by the algorithm of [New02]. However, as we will see in Section 7.4, the branching programs resulting from the reduction from our st-connectivity programs have a feature that they require reading more than one bit at a time to move between layers. The next definition describes these special branching programs formally.

Definition 21 (Clustered Branching Program)**.** *A* c-clustered Read Once Branching Program *of width w and length m over an input of n bits (or shortly $BP_c(w, m, n)$) is a tuple $\langle G, \mathcal{L}, X, \mathcal{I}\rangle$, where similarly to the previous definition, G is a directed graph with labeled edges (see below for the set of labels), $\mathcal{L} = (L_0, \dots, L_m)$ is a partition of G's vertices into m layers such that* $\max_i\{|L_i|\} \leq w$*, and $X = \{x_0, \dots, x_{n-1}\}$ is a set of n Boolean variables. Here too, G has one special vertex s belonging to L_0, and a subset $T \subset L_n$ of* accepting *vertices. The additional element $\mathcal{I}$ is a partition $(I_1, \dots, I_m)$ of X into m components, such that* $\max_i\{|I_i|\} \leq c$*.*

All edges in between two consecutive layers L_{i-1} and L_i are associated with the component I_i of $\mathcal{I}$. Each vertex in L_{i-1} has $2^{|I_i|}$ outgoing edges, each of them labeled by a distinct $\alpha \in \{0,1\}^{|I_i|}$.

An assignment $\sigma : X \to \{0,1\}$ to X defines a subgraph G_σ of G, which has the same vertex set as G, and for every layer L_i (whose outgoing edges are associated with the component I_i), the subgraph G_σ has only the edges labeled by $\Big(\sigma(x_i)\Big)_{i\in I_i}$. A c-clustered read once branching program $B = \langle G, \mathcal{L}, X, \mathcal{I}\rangle$ defines a property $P_B \subset \{0,1\}^n$ in the following way: $\sigma \in P_B$ if and only if in the subgraph G_σ there is a directed path from the starting vertex s to one of the accepting vertices in T.

Observe that $BP(w, m)$ is equivalent to $BP_1(w, m, m)$.

7.2 The main result

For an undirected graph G and a pair $s, t \in V(G)$ of distinct vertices, let P_G^{st} be a set of G-orientations under which there is a directed path from s to t. Formally, $P_G^{st} = \{\overrightarrow{G} : \mathrm{dist}_{\overrightarrow{G}}(s,t) < \infty\}$.

Theorem 7.2.1. *The property P_G^{st} is testable. In particular, for any undirected graph G, two vertices $s, t \in V(G)$ and every $\varepsilon > 0$, there is an (ε, q)-tester T for P_G^{st} with query complexity $q = (2/\varepsilon)^{2^{(1/\varepsilon)\cdot 2^{O(\varepsilon^{-2})}}}$*

Note that the property P_G^{st} is trivially testable whenever the undirected distance from s to t in G is less than $\varepsilon|E(G)|$. In particular, our result is interesting only for sparse graphs, i.e. graphs for which $|E(G)| \leq |V(G)|/\varepsilon$.

7.2.1 Proof overview

The main idea of the proof is to reduce the problem of testing st-connectivity in the orientation model to the problem of testing a Boolean function that is represented by a small width read once branching program. For the latter we have the result of [New02] asserting that each such Boolean function is testable.

Theorem 7.2.2 ([New02])**.** *Let $P \subseteq \{0,1\}^n$ be the language accepted by a read-once branching program of width w. Then testing P with one-sided error requires at most $\left(\frac{2^w}{\varepsilon}\right)^{O(w)}$ queries.*

By the definition above of $BP(w,n)$, one could already notice that testing the acceptance of a branching program resembles testing st-connectivity, and that the two problems seem quite close. However, there are several significant differences:

1. In branching programs, every layer is associated with a variable whose querying reveals all the edges going out from this layer. In st-connectivity instances, in order to discover the orientation of these edges we need to query each of them separately.

2. The length of the input in branching programs is the number of layers rather than the total number of edges.

3. The edges in branching program graphs are always directed from L_{i-1} to L_i for some $i \in [n]$. In our case, the graph is not layered, and a pair u, v of vertices might have any number of edges in both directions.

4. In branching programs the graphs have out-degree exactly 2, while an input graph of the st-connectivity problem might have vertices with unbounded out-degree.

5. The most significant difference is that the input graphs of the st-connectivity problem may have unbounded width. This means that the naive reduction to branching programs may result in an unbounded width BPs, which we cannot test with a constant number of queries.

We resolve these points in several steps. First, given an input graph G, we reduce it to a graph $G^{(1)}$ which has the following property: for every induced subgraph W of $G^{(1)}$, the diameter of W is larger than ε times the number of edges in W. Then we prove that $G^{(1)}$ can be layered such that most of its edges lie within bounded-width layers. In particular, the total number of edges in the "wide" layers is bounded by $\frac{\varepsilon}{2}E(G^{(1)})$. For this we need a concentration type lemma which is stated and proven here for this purpose. Next we reduce $G^{(1)}$ to a graph $G^{(2)}$, which can be layered as above, but without wide layers at all. This in particular means that the number of edges in $G^{(2)}$ is of the order of the number of layers, similarly to bounded width branching programs. In addition, in this layering of $G^{(2)}$ the vertex t is in the last layer (while the vertex s remains in the first layer). Then we reduce $G^{(2)}$ (which might be thought of as the *bidirectional* analogue of a bounded width branching program) to a clustered read once bounded width branching program. Finally we show that these clustered branching programs can be converted into non-clustered branching programs, to which we can apply the test from [New02].

7.3 Reducing general graphs to connectivity programs

In this section we prove our first step towards proving Theorem 7.2.1. We reduce the problem of testing st-connectivity in a general graph to the problem of testing st-connectivity of an st-Connectivity Program. First we define the notion of reducibility in our context, and then we describe a sequence of reductions that will eventually lead us to the problem of testing read once bounded width BPs.

7.3.1 Reducibility between st-connectivity instances

Let $\mathcal{G}^{st}$ denote the class of undirected graphs having two distinct vertices s and t, and let $G, G' \in \mathcal{G}^{st}$. We say that G is (ε, η)*-reducible* to G' if there is a function ρ that maps orientations of G to orientations of G' (from now on we denote by $\overrightarrow{G}'$ the orientation $\rho(\overrightarrow{G})$) such that the following holds.

- If $\overrightarrow{G} \in P_G^{st}$ then $\overrightarrow{G}' \in P_{G'}^{st}$
- If $\delta(\overrightarrow{G}, P_G^{st}) \geq \varepsilon$ then $\delta(\overrightarrow{G}', P_{G'}^{st}) \geq \eta$
- Any orientation query to $\overrightarrow{G}'$ can be simulated by a single orientation query to $\overrightarrow{G}$.

We say that G is (ε)*-reducible* to G' if it is $(\varepsilon, \varepsilon)$-reducible to G'. Notice that whenever G is (ε, η)-reducible to G', any (η, q)-tester T' for $P_{G'}^{st}$ can be converted into an (ε, q)-tester T for P_G^{st}. Or in other words, (ε, q)-testing P_G^{st} is reducible to (η, q)-testing $P_{G'}^{st}$.

In the following section we introduce our first reduction, which is referred to as the reduction from G to $G^{(1)}$ in the proof overview.

7.3.2 Reduction to graphs having high-diameter subgraphs

An undirected graph G is called ε*-long* if for every subset $W \subset V(G)$, $\operatorname{diam}(G(W)) > \varepsilon|E(W)|$.

Lemma 7.3.1. *Any graph $G \in \mathcal{G}^{st}$ is (ε)-reducible to a graph $G' \in \mathcal{G}^{st}$ which is ε -long.*

We first define a general contraction operator for graphs, and then use it for the proof. Given a graph G let $W \subset V$ be a subset of its vertices, and let $C_1, \dots, C_r$ be the vertex sets of the connected components of $G(W)$. Namely, for each $C_i \subset W$, the induced subgraph $G(C_i)$ of the underlying graph G is connected, and for any pair $u \in C_i, v \in C_j$ $(i \neq j)$ of vertices, $e\{u,v\} \notin E(G)$. We define a graph G/W as follows. The graph G/W has the vertex set $V/W = (V \setminus W) \cup \{c_1, \dots, c_r\}$ and its edges are

$$E/W = \Big\{e\{u,v\} : (u,v \in V \backslash W) \wedge (e \in E)\Big\} \cup \Big\{e\{c_i,v\} : (v \in V \backslash W) \wedge \exists_{u \in C_i}(e\{u,v\} \in E)\Big\}$$

Intuitively, in G/W we contract every connected component C_i of $G(W)$ into a single (new) vertex c_i without changing the rest of the graph. Note that such a contraction may create parallel edges, but loops are removed. Whenever $s \in C_i$ for some $i \in [r]$, we rename the vertex c_i by s, and similarly if $t \in C_j$ then we rename c_j by t. In the following a connected component containing both s and t will not be contracted, so we can assume that the distinguished vertices s and t remain in the new graph G/W. Given an orientation $\overrightarrow{G}$ of G we define the orientation $\overrightarrow{G}/W = \rho(\overrightarrow{G})$ of G/W as the orientation induced from $\overrightarrow{G}$ in the natural way (note that there are no "new" edges in G/W). Before proving Lemma 7.3.1, we state and prove an auxiliary lemma.

Lemma 7.3.2. *Let $W \subset V$ be a subset of G's vertices, such that* $\operatorname{diam}(G(W)) \leq \varepsilon|E(W)|$. *Then G is (ε)-reducible to the graph G/W.*

In particular, if $v \in V$ is a vertex of $\overrightarrow{G}$ such that $\deg(v) \geq 2/\varepsilon$, *then G is (ε)-reducible to the graph $G/N_1(v)$.*

Proof. Fix an orientation $\overrightarrow{G}$ of G. It is clear that if $\overrightarrow{G} \in P^{st}_G$ then $\overrightarrow{G}/W \in P^{st}_{G/W}$. Now assume that $\delta(\overrightarrow{G}, P^{st}_G) \geq \varepsilon$. Let d and d' denote $\delta(\overrightarrow{G}, P^{st}_G) \cdot |E(G)|$ and $\delta(\overrightarrow{G}/W, P^{st}_{G/W}) \cdot |E(G/W)|$ respectively. From the definition of the graph G/W it follows that $d' \geq d - \operatorname{diam}(G(W))$. This is true since any st-path in $\overrightarrow{G}/W$ can be extended to an st-path in $\overrightarrow{G}$ by reorienting at most $\operatorname{diam}(G(W))$ edges in W (by definition, $\operatorname{diam}(G(W))$ is an upper bound on the undirected distance from any "entry" vertex to any "exit" vertex in $G(W)$). From the condition on W we have $|E(\overrightarrow{G}/W)| = |E| - |E(W)| \leq |E| - \frac{\operatorname{diam}(G(W))}{\varepsilon}$. Combining these two together we have

$$\delta(\overrightarrow{G}/W, P^{st}_{G/W}) = \frac{d'}{|E(\overrightarrow{G}/W)|} \geq \frac{d - \operatorname{diam}(G(W))}{|E| - \operatorname{diam}(G(W))/\varepsilon} \geq \frac{d - \operatorname{diam}(G(W))}{d/\varepsilon - \operatorname{diam}(G(W))/\varepsilon} = \varepsilon$$

In addition, it is clear that we can simulate each query to $\overrightarrow{G}/W$ by making at most one query to $\overrightarrow{G}$. □

Proof of Lemma 7.3.1. We apply this contraction (iteratively) for each "bad" subgraph W, until eventually we get a graph G' in which all vertex subsets W satisfy $\mathrm{diam}(G(W)) > \frac{|E(W)|}{\varepsilon}$. If in some stage we have both s and t contained in the contracted subgraph W, then we just output a tester that accepts all inputs (since in this case all orientations are ε-close to being st-connected). Note that this process may result in several different graphs (depending on choices of the set W in each iteration), but we are only interested in any such graph G'. □

7.3.3 Properties of ε-long graphs

Next we show that an ε-long graph G can be "layered" so that the total number of edges in the "wide" layers is bounded by $\frac{\varepsilon}{2}E(G)$. We first define graph layering.

Definition 22 (Graph layering and width)**.** *Given a graph G and a vertex $s \in V(G)$, let m denote the maximal (undirected) distance from s to any other vertex in G. We define a* layering *$\mathcal{L} = (L_0, L_1, \ldots, L_m)$ of G's vertices as follows. $L_0 = \{s\}$ and for every $i > 0$, $L_i = N_i(s) \setminus N_{i-1}(s)$. Namely L_i is the set of vertices which are at (undirected) distance exactly i from s. Note that for every edge $e\{u, v\}$ of G either both u and v are in the same layer, or $u \in L_{i-1}$ and $v \in L_i$ for some $i \in [m]$.*

We also denote by $E_i^{\mathcal{L}}$ the subset of G's edges that either have one vertex in L_i and the other in L_{i-1}, or edges that have both vertices in L_i. Alternatively, $E_i^{\mathcal{L}} = E(L_i \cup L_{i-1}) \setminus E(L_{i-1})$. We refer to the sets L_i and $E_i^{\mathcal{L}}$ as vertex-layers *and* edge-layers *respectively. We omit the superscript $\mathcal{L}$ from the edge-layers notation whenever it is clear from the context.*

The vertex-width *of a layering $\mathcal{L}$ is* $\max_i\{|L_i|\}$*, and the* edge-width *of $\mathcal{L}$ is* $\max_i\{|E_i|\}$.

Bounding the number of edges within wide edge-layers

The following lemma (which is proved in Section 7.6.1) states that in a layering of an ε-long graph most of the edges are captured in edge-layers of bounded width.

Lemma 7.3.3. *Consider the layering $\mathcal{L}$ of an ε-long graph G as defined above, and let $I = \{i : |E_i| > 2^{100/\varepsilon^2}/\varepsilon\}$ be the set of indices of the wide edge-layers. Then the following holds: $\sum_{i \in I} |E_i| \le \frac{\varepsilon}{2}|E|$.*

In the following two sections we prove that ε-long graphs can be reduced to graphs that have bounded width. In terms of the proof overview, we are going to reduce the graph $G^{(1)}$ to the graph $G^{(2)}$.

7.3.4 Reduction to bounded width graphs

Let $G = (V, E) \in \mathcal{G}^{st}$, and let $\mathcal{L} = (L_0, L_1, \ldots, L_m)$ be the layering of G as above. We call an edge-layer E_i *wide* if $|E_i| > \frac{1}{\varepsilon} \cdot 2^{100/\varepsilon^2}$. Let $\mathcal{W}$ be the set of all wide edge-layers.

Lemma 7.3.4. *If $G \in \mathcal{G}^{st}$ satisfies $\sum_{E_i \in \mathcal{W}} |E_i| \leq \frac{\varepsilon}{2}|E|$ then G is $(\varepsilon, \varepsilon/2)$-reducible to a graph G' which has no wide edge-layers at all.*

The proof of Lemma 7.3.4 appears in Section 7.6.2.

7.3.5 Reducing bounded width graphs to *st*-connectivity programs

So far we reduced the original graph G to a graph G' which has a layering $\mathcal{L} = (L_0, L_1, \dots, L_m)$ of edge-width at most $w = \frac{1}{\varepsilon} \cdot 2^{100/\varepsilon^2}$, and in which the source vertex s belongs to layer L_0. The remaining difference between G' and an st-connectivity program is that in G' the target vertex t might not be in the last vertex-layer L_m. The following lemma states that we can overcome this difference by another reduction.

Lemma 7.3.5. *Let G' be a graph as described above. Then G' is $(\varepsilon, \varepsilon/2)$-reducible to an st-connectivity program S of width at most $w+1$.*

Proof. Let L_r be the layer to which t belongs in the layering $\mathcal{L}$ of the graph G'. Let $P = \{p_1, p_2, \dots, p_{m-r}\}$ be a set of $m - r$ new vertices. We define S as follows.

- $V(S) = V(G') \cup P$
- $E(S) = E(G') \cup \left(\bigcup_{i=1}^{m-r-1} \{e_i(p_i, p_{i+1})\}\right) \cup \{e_t(t, p_1)\}$

Any orientation $\overrightarrow{G}'$ of G' induces a natural orientation $\overrightarrow{S}$ of S; all edges $e \in E(S)$ that are also in $E(G')$ have the same orientation as in $\overrightarrow{G}'$, while the orientation of the new edges were defined explicitly above. We also rename t to p_0 and rename p_{m-r} to t in S. Basically we have added a sufficiently long path from the original target vertex to the new target vertex to get S. Now it is easy to verify that G' is indeed $(\varepsilon, \varepsilon/2)$-reducible to S (assuming that G has at least $1/\varepsilon$ edges), and that the width of S is as required. $\square$

7.4 Reducing *st*-connectivity programs to branching programs

We now show how to reduce an st-connectivity program to a clustered branching program (recall Definition 19 and Definition 21). First observe that we can assume without loss of generality that if an st-connectivity program has edge-width w, then its vertex-width is at most $2w$ (since removing vertices of degree 0 essentially does not affect the st-connectivity program, and a vertex in L_i with edges only between it and L_{i+1} can be safely moved to L_{i+1}).

Before moving to the formal description of the reduction, we start with a short intuitive overview. A branching program corresponds to a (space bounded) computation that moves from the start vertex s, which represents no information about the input at all, and proceeds (via the edges that are consistent with the input bits) along a path to one of the final vertices. Every vertex of the branching program represents some additional information gathered by reading more and more pieces of the input. Thus, the best way to understand the reduction is to understand the implicit meaning of each vertex in the constructed branching program.

Given a graph G of a bounded width st-connectivity program, and its layering $L_0, L_1, \dots L_m$, we construct a graph G' (with layering $L'_0, L'_1, \dots L'_m$) for the bounded-width branching program. The graph G' has the same number of layers as G. Each level L'_i in G' will represent the conditional connectivity of the vertices in the subgraph $G_i = G(\bigcup_{j=0}^{i} L_i)$ of G. To be specific, the knowledge we want to store in a typical vertex at layer L'_i of G' is the following.

- for every $u \in L_i$ whether it is reachable from s in G_i.
- for every $v, u \in L_i$ whether v is reachable from u in G_i.

Hence, the amount of information we store in each node $x \in L'_i$ has at most $2w + (2w)^2$ many bits, and so there will be at most 4^{2w^2+w} vertices in each L'_i, meaning that the graph G' of the branching program is of bounded width as well.

Lemma 7.4.1. *Let $\varepsilon > 0$ be a positive constant. Given a $CP(w, m, n)$ instance $C = \langle G, \mathcal{L} \rangle$, we can construct a $BP_w(4^{2w^2+w}, m, n)$ instance $B = \langle G', \mathcal{L}', X', \mathcal{I}' \rangle$ and a mapping ρ from G-orientations to assignments on X such that the following holds,*

- *if $\overrightarrow{G}$ satisfies P_C then $\sigma = \rho(\overrightarrow{G})$ satisfies P_B.*
- *if $\overrightarrow{G}$ is ε-far from satisfying P_C then $\sigma = \rho(\overrightarrow{G})$ is ε-far from satisfying P_B.*
- *any assignment query to σ can be simulated using a single orientation query to $\overrightarrow{G}$.*

Proof. First we describe the construction, and then show that it satisfies the requirements above.

The vertices of G': We fix i and show, based on the layer L_i of G, how to construct the corresponding layer L'_i of G'. Each vertex in L'_i corresponds to a possible value of a pair $(S_i,\ R_i)$ of sets. The first set $S_i \subseteq L_i$ contains vertices $v \in L_i$ for which there is a directed path from s to v in the subgraph of G induced on $\bigcup_{j=0}^{i} L_j$. The second set $R_i \subseteq L_i \times L_i$ is a set of ordered pairs of vertices, such that every ordered pair (u, v) is in R_i if there is a directed path from u to v in the subgraph of G induced on $\bigcup_{j=0}^{i} L_j$ (the path can be of length 0, meaning that the R_i's contain all pairs (v, v), $v \in L_i$). Notice that $|L'_i| = 2^{|L_i|^2+|L_i|} \leq 4^{2w^2+w}$ for all i.

The edges of G': Now we construct the edges of G'. Recall that E'_{i+1} denotes the set of (labeled) edges having one vertex in L'_{i+1} and the other in L'_i. Fix i and a vertex $v \in L'_i$.

Let (S, R) be the pair of sets that correspond to v. Let S'' be the set $L_{i+1} \cap \Gamma_{out}(S)$, namely the neighbors of vertices from S that are in L_{i+1}, and set $R'' = \{(v, v) : v \in L_{i+1}\} \cup \{(u, v) : (u, v \in L_{i+1}) \wedge (v \in \Gamma_{out}(u))\} \cup \{(u, v) : (u, v \in L_{i+1}) \wedge \Big((\Gamma_{out}(u) \times \Gamma_{in}(v)) \cap R \neq \emptyset\Big)\}$. Now define R' as the transitive closure of R'', and set $S' = S'' \cup \{v \in L_{i+1} : \exists_{u \in S''}(u, v) \in R'\}$. Let $v' \in L'_{i+1}$ be the vertex corresponding to the pair (S', R') that we defined above. Then the edges of E'_{i+1} are given by all such pairs of vertices (v, v').

The variables in X': Each variable $x'_i \in X'$ is associated with an edge $e_i \in E(G)$. This association is actually the mapping ρ above, i.e. every orientation $\overrightarrow{G}$ of G defines an assignment σ on X'.

The partition $\mathcal{I}'$ of X': Recall that E_i denotes the set of edges of G having either one vertex in L_{i-1} and the other in L_i, or both vertices in L_i. The partition $\mathcal{I}'$ of X' is induced by the partition $\mathcal{L}$ of $V(G)$. Namely, the component I_i of $\mathcal{I}$ contains the set of variables in X' that are associated with edges in E_i. Thus w is also a bound on the sizes of the components in $\mathcal{I}$.

The set $T' \subset L'_m$ of accepting vertices: The set T' is simply the subset of vertices in L'_m whose corresponding set S contains the target vertex t of G.

Note that each value of a variable in X' corresponds exactly to an orientation of an edge in G. This immediately implies the third assertion in the statement of the lemma. Distances between inputs are clearly preserved, so to prove the other assertions it is enough to show that the branching program accepts exactly those assignments that correspond to orientations accepted by the connectivity program. It is straightforward (and left to the reader) to see that the vertex reached in each L'_i indeed fits the description of the sets R and S, and so an assignment is accepted if and only if it corresponds to a connecting orientation. □

The branching programs resulting from the above reduction have a feature that they require reading more than one bit at a time to move between layers. Specifically, they conform to Definition 21. The result in [New02], however, deals with standard branching programs (see Definition 20), which in relation to the above are a special case in which essentially $m = n$ and all the I_i's have size 1. In the next section we deal with the final reduction, into a standard (non-clustered) branching program (in which the edges between two layers depend on just one Boolean variable).

7.4.1 Converting clustered branching programs to non-clustered ones

Lemma 7.4.2. *Any $BP_c(w, m, n)$ instance can be converted to a $BP(w2^c, n)$ instance accepting the very same language.*

Proof. Throughout the proof it is convenient to refer to the vertices of a layer L_i in the clustered branching program as a set of *states*, and to the edges between L_{i-1} and L_i

as a *transition function* $f_i : L_{i-1} \times \{0,1\}^{|I_i|} \to L_i$. Namely, $f_i(v, b_1, \ldots, b_{|I_i|})$ is equal to the vertex w in L_i so that (v, w) is an edge labeled with $(b_1, \ldots, b_{|I_i|})$. We shall use an analogue notation for the transition functions in the constructed branching program, $f'_i : L'_{i-1} \times \{0,1\} \to L'_i$. The basic idea of the proof is that instead of reading the entire cluster I_i at once, we read it bit by bit, and use additional states in intermediate layers to store the bits that we have read so far. We need to read at most c bits before we can use the original transition functions, which causes the blowup of up to 2^c in the number of states in each layer. We define the new layers $\{s\} = L'_0, \ldots, L'_n$ of the program inductively. First we define $L_0 = L'_0 = \{s\}$. Now, assuming that we have already converted $L_0, \ldots, L_{i-1}$ into $L'_0, \ldots, L'_j$ such that $L'_j = L_{i-1}$, and calculated the corresponding transition functions $f_k : L'_{k-1} \times \{0,1\} \to L'_k$, we show how to convert L_i into layers $L'_{j+1}, \ldots, L'_{j+|I_i|}$ so that $L'_{j+|I_i|} = L_i$.

For $0 < k < |I_i|$ we set $L'_{j+k} = L_{i-1} \times \{0,1\}^k$, and set $L'_{j+|I_i|} = L_i$. Each layer L'_{j+k} will be associated with the bit corresponding to the k'th member of I_i, which to reduce indexes we shall re-label as y_{j+k}. In the following we denote members of a cross product $A \times B$ as tuples (a, b) with $a \in A$ and $b \in B$. The transition functions $f'_{j+1}, \ldots, f'_{j+|I_i|}$ are set as follows.

- For $k = 1$ we set $f'_{j+1}(v, y_{j+1}) = (v, y_{j+1})$, that is the transition is to the tuple resulting from pairing v with the bit y_{j+1}
- Accordingly, for $1 < k < |I_i|$ we set $f'_{j+k}((v, b_1, \ldots, b_{k-1}), y_{j+k}) = (v, b_1, \ldots, b_{k-1}, y_{j+k})$, i.e. we concatenate the value of y_{j+k} to the previously collected values.
- Finally, for $k = |I_i|$ we set $f'_{j+|I_i|}(v, b_1, \ldots, b_{|I_i|-1}) = f_i(v, b_1, \ldots, b_{|I_i|-1}, y_{j+|I_i|})$, i.e. we employ the function f_i on all the collected bit values and $y_{j+|I_i|}$.

The accepting subset T' of $L'_n = L_m$ remains the same as the original T. It is now not hard to see that both programs accept the exact same language over $x_1, \ldots, x_n$. $\square$

7.5 Wrapping up – proof of Theorem 7.2.1

We started with a graph G and wanted to construct an (ε, q)-test for st-connectivity of orientations of G. In Section 7.3.1, Section 7.3.2 and Section 7.3.3 we constructed a graph $G^{(1)}$ such that if we have an (ε, q)-test for st-connectivity in $G^{(1)}$, then we have an (ε, q)-test for G. Additionally $G^{(1)}$ has the property that most of the edge-layers in $G^{(1)}$ are of size at most $w = \frac{1}{\varepsilon} \cdot 2^{100/\varepsilon^2}$. Then in Section 7.3.4 we constructed a graph $G^{(2)}$ such that if we have an $(\frac{\varepsilon}{2}, q)$-test for st-connectivity in $G^{(2)}$ then we have an (ε, q)-test for $G^{(1)}$ and hence we have one for G. Moreover $G^{(2)}$ has all its edge-layers of size at most

w. Finally in Section 7.3.5 we built a graph $G^{(3)}$ which has all its edge-layers of width at most $w+1$, and in addition, the vertices s and t are in the first and the last vertex-layers of $G^{(3)}$ respectively. We also showed that having an $(\frac{\varepsilon}{4}, q)$-test for st-connectivity in $G^{(3)}$ implies an $(\frac{\varepsilon}{2}, q)$-test for $G^{(2)}$, and hence an (ε, q)-test for G. This ends the first part of the proof, which reduces general graphs to st-connectivity programs.

Then in Section 7.4 from $G^{(3)}$ we constructed a read once $(w+1)$-clustered Branching Program that has width $4^{2(w+1)^2+w+1}$ so that an $(\frac{\varepsilon}{4}, q)$-test for this BP gives an $(\frac{\varepsilon}{4}, q)$-test for st-connectivity in $G^{(3)}$. Then we converted the $(w+1)$-clustered Branching Program to a non-clustered Branching Program which has width $w_1 = 4^{2(w+1)^2+(w+1)}2^{(w+1)}$. Once we have our read once bounded width branching program, by applying the algorithm of [New02] for testing branching programs we get an $(\frac{\varepsilon}{4}, q)$-test with $q = (\frac{2^{w_1}}{\varepsilon/4})^{O(w_1)}$ queries for our problem. Hence by combining all of the above, we get an (ε, q) testing algorithm for our original st-connectivity problem, where $q = (2/\varepsilon)^{2^{O((1/\varepsilon)\cdot 2^{(100/\varepsilon^2)})}}$

. □

7.6 Completing the missing proofs

7.6.1 Proof of Lemma 7.3.3

Before proving Lemma 7.3.3 we prove an auxiliary concentration lemma. Denote by $\mathcal{A} = \langle a_0, a_1, \ldots, a_m \rangle$ a sequence of integers, where $a_0 = 1$ and for every $i \geq 1$, $a_i = |E_i|$.

Definition 23 (ε-good sequence). *Let $0 < \varepsilon < 1$ be a positive constant. A sequence $1 = a_0, a_1, \ldots, a_m$ of positive natural numbers is* ε-good *if for every $0 \leq k < m$ and $\ell \in [m-k]$ we have that*

$$\sum_{i=k+1}^{k+\ell} a_i \leq \frac{\ell \cdot a_k}{\varepsilon}.$$

Claim 7.6.1. *Let G be a graph in which for any induced subgraph W we have $|E(W)| < \operatorname{diam}(W)/\varepsilon$. Then the sequence $\mathcal{A} = \langle a_0, a_1, \ldots, a_m \rangle$ defined above is $\varepsilon/4$-good.*

Proof. Let us assume the contrary of the claim. Let k and ℓ be such that

$$\sum_{i=k+1}^{k+\ell} a_i > \frac{4\ell \cdot a_k}{\varepsilon}$$

Consider the subgraph W defined by the vertices $\cup_{i=k}^{k+\ell} L_i$ and the edges $\cup_{i=k+1}^{k+\ell} E_i$. Now the number of edges in W is clearly $\sum_{i=k+1}^{k+\ell} a_i$. For each vertex v in L_k consider the neighborhood of distance $\ell+1$ from v, $N_{\ell+1}(v)$, and denote the subgraph it spans by W_v. Notice that each W_v is of diameter at most $2(\ell+1) \leq 4\ell$, and that the union of the edge sets of the W_v includes the whole edge set of W, so we have $\sum_{v \in L_k} |E(W_v)| \geq \sum_{i=k+1}^{k+\ell} a_i$. The number of vertices in L_k is at most a_k, so by the pigeonhole principle we know that at least one of the vertices v in L_k has at least $\frac{\sum_{i=k+1}^{k+\ell} a_i}{a_k}$ edges in W_v. By our assumption on $\sum_{i=k+1}^{k+\ell} a_i$ we have $|E(W_v)| > \frac{4\ell}{\varepsilon} \geq \operatorname{diam}(W_v)/\varepsilon$, a contradiction. □

Lemma 7.6.2 (The weak concentration lemma). *Let $\langle a_0, \ldots, a_m \rangle$ be an ε-good sequence and let $B = \{i \mid a_i > 2^{5/\varepsilon^2}/\varepsilon\}$. Then*

$$\sum_{i \in B} a_i \leq \varepsilon \sum_{i=1}^{m} a_i.$$

Proof. Let A be the sum of $a_1, \ldots, a_m$. According to Claim 7.6.3, that we state and prove further on, we may without loss of generality assume that $a_0, \ldots, a_m$ are monotone non-decreasing as a function of their index. Now we assume that m is a power of 2, and prove that in this case $\sum_{i \in B} a_i \leq \frac{\varepsilon}{2} \sum_{i=1}^{m} a_i$. This is sufficient because if m is not a power of 2 then we can add more copies of $a_0 = 1$ in the beginning until m is a power of 2, and doing so will no more than double the sum of the sequence while keeping it ε-good.

We first note that $A \leq m/\varepsilon$ since the sequence is ε-good. In particular it is safe to assume that $m > 2^{4/\varepsilon^2}$, because otherwise we would have $B = \emptyset$. For the sake of contradiction, now assume that

$$\sum_{i \in B} a_i > \frac{\varepsilon}{2} A. \tag{7.1}$$

Set $p(k) = m(1 - 2^{-(k+1)})$ and $R(k) = \sum_{i=p(k)+1}^{p(k+1)} a_i$. Since the sum ranges in the definition of $R(k)$ are disjoint,

$$A \geq \sum_{k=1}^{4/\varepsilon^2} R(k). \tag{7.2}$$

We next show that Assumption (7.1) implies that for any $k \in [4/\varepsilon^2]$ we have that $R(k) > \frac{\varepsilon^2 \cdot A}{4}$. This is sufficient since it implies that $\sum_{j=1}^{4/\varepsilon^2} R(k) > A$, which contradicts Equation (7.2).

Let $k_0 = 4/\varepsilon^2$. The assumption that the sequence is non-decreasing implies that

$$A \geq a_{p(k_0)} \cdot (m - p(k_0)) = a_{p(k_0)} \cdot m \cdot 2^{-(k_0+1)}.$$

Using that $A \leq m/\varepsilon$ we get that $a_{p(k_0)} \leq \frac{2^{(k_0+1)}}{\varepsilon} = \frac{2^{4/\varepsilon^2+1}}{\varepsilon}$.

Consequently, if $a_i > 2^{5/\varepsilon^2}/\varepsilon$ then $i > p(4/\varepsilon^2 + 1)$. Hence for $k \leq k_0$, by Assumption (7.1) we get that $\sum_{i=p(k)+1}^{m} a_i > \sum_{i \in B} a_i > \frac{\varepsilon}{2} A$. On the other hand as $a_0, \ldots, a_m$ is an ε-good sequence we know that $\varepsilon^{-1} \cdot a_{p(k)} \cdot m \cdot 2^{-(k+1)} \geq \sum_{i=p(k)+1}^{m} a_i$, and therefore by plugging this into the previous inequality we get that $\varepsilon^{-1} \cdot a_{p(k)} \cdot m \cdot 2^{-(k+1)} > \frac{\varepsilon}{2} A$. Thus, $a_{p(k)} > \frac{\varepsilon^2 \cdot 2^k \cdot A}{m}$. Since $a_0, \ldots, a_m$ are monotone non-decreasing as a function of their index we conclude that $R(k) \geq a_{p(k)} \cdot (p(k+1) - p(k)) = a_{p(k)} \cdot m \cdot 2^{-(k+2)}$. Together with the lower bound on $a_{p(k)}$ we get that $R(k) > \frac{\varepsilon^2 \cdot A}{4}$. □

Claim 7.6.3. *If $\langle a_0, \ldots, a_m \rangle$ is an ε-good sequence, then the sequence $\langle b_0, \ldots, b_m \rangle$ obtained by sorting $a_0, \ldots, a_m$ is also ε-good.*

Proof. As the b_i's are monotone nondecreasing as a function of their index, in order to show that they are an ε-good sequence we only need to show that for any $k \in [m]$ we have

$$\frac{1}{m-k} \sum_{i=k+1}^{m} b_i \leq \frac{b_k}{\varepsilon}$$

as for monotone sequences the average only decreases if a subsequence is chopped off from the right.

Fix $k \leq m$ and consider the average $\frac{1}{m-k} \sum_{i=k+1}^{m} b_i$. We may assume that each b_i is a renaming of some a_j in the original sequence. Thus the subsequence $B = \langle b_{k+1}, \ldots, b_m \rangle$ corresponds to members in the original subsequence:

$$\langle a_{i_1+1}, \ldots, a_{i_1+j_1} \rangle, \langle a_{i_2+1}, \ldots, a_{i_2+j_2} \rangle, \ldots, \langle a_{i_t+1}, \ldots, a_{i_t+j_t} \rangle$$

where each subsequence $\langle a_{i_r+1}, \ldots a_{i_r+j_r} \rangle$ is a maximal contiguous subsequence in the original sequence whose members were renamed to members of B, and hence their value is at least b_k (as all members in B are such).

On the other hand, the values of $a_{i_1}, a_{i_2}, \ldots, a_{i_t}$ are all bounded by b_k as their renaming does not put them in B. Since $\langle a_1, \ldots, a_m \rangle$ is ε-good, this means that for every $1 \leq r \leq t$, the average of the subsequence $\langle a_{i_r+1}, \ldots a_{i_r+j_r} \rangle$ is at most b_k/ε. Note that it is safe to assume that b_0 is the renaming of a_0 (which for a good sequence is equal to the minimum 1) and hence $i_1 \geq 0$. Finally, as the average of the subsequence B is clearly a weighted average of the averages of the $\langle a_{i_r+1}, \ldots a_{i_r+j_r} \rangle$ subsequences, it is bounded by b_k/ε as well. □

Now the proof of Lemma 7.3.3 follows directly from Claim 7.6.1 and Lemma 7.6.2.

7.6.2 Proof of Lemma 7.3.4

Proof. Recall the definition of G/W (a graph in which a subset of vertices is contracted) from Section 7.3.2. Let E_i be a wide edge-layer. For every edge $e = e\{u, v\} \in E_i$ define $W_e = \{u, v\}$. We iteratively contract the subgraphs W_e in G, for all edges in all wide edge-layers. Denote the final graph as $G' = (V', E')$.

We claim that after this process G' has no wide edge-layers at all. Formally let $p(i)$ denote the size of the set $\{j | j \leq i \text{ and } E_j \text{ is wide}\}|$. Namely $p(i)$ is number of wide edge-layers in G preceding the vertex-layer L_i. Now we have the following claim:

Observation 7.6.4. *Let $v \in L_i$ be a vertex in the i'th vertex-layer of G. Then v's representing vertex v' in G' is in the vertex-layer $L'_{i-p(i)}$ of G'.*

Proof. The proof follows by induction on i. $\square$

The mapping μ sending i to $i' = (i - p(i))$ is monotone non-decreasing, and onto the number of layers in G'. For an index i' of a vertex-layer in G', let $\ell(i')$ denote the largest index in the set $\mu^{-1}(i')$. Recall that E'_i is the set of edges from layer $L'_{i'-1}$ to layer $L'_{i'}$ and within layer $L'_{i'}$ in G'. Note that these edges are exactly the edges that correspond to the edges between layers $L_{\ell(i')}$ and $L_{\ell(i')-1}$ and within $L_{\ell(i')}$ in G. Thus assuming towards a contradiction that E'_i is wide in G', means that $E_{\ell(i)}$ is wide in G . But this is not possible, as $E_{\ell(i)}$ is in the set of edges that were not contracted. As a conclusion, G' cannot have any wide edge-layers.

It is clear that any orientation query to $\overrightarrow{G}'$ can be simulated by a single orientation query to $\overrightarrow{G}$, and since all we did was contracting edges, if there is a path from s to t in $\overrightarrow{G}$ then there is a path from s to t in $\overrightarrow{G}'$. Now we only need to show is that the second condition of reducibility holds.

We now show that by changing the direction of at most $\frac{\varepsilon}{2}|E'|$ edges we can have a path from s to t in $\overrightarrow{G}$ if there was one in $\overrightarrow{G}'$. We can always assume that the path in $\overrightarrow{G}'$ is simple, i.e. it passes through each vertex at most once. Now each vertex in G' corresponds to a connected subgraph in G. We call a non-trivial subgraph of G *shrunk* if all its vertices, and only them, are represented by a single vertex in G'. Clearly we can extend an st-path in G' to an st-path in G by only reorienting the edges in the shrunk subgraphs of G. We only contracted edges in the wide edge-layers, so the total number of edges in the shrunk components is at most the total number of edges in the wide edge-layers. By the assumption on G we have that only $\frac{\varepsilon}{2}|E|$ of the edges lie in the shrunk subgraphs of G. If by changing the orientation of only $\frac{\varepsilon}{2}|E'|$ edges in $\overrightarrow{G}'$ we get a path from s to t in $\overrightarrow{G}'$, then by changing only $\frac{\varepsilon}{2}|E'| + \frac{\varepsilon}{2}|E| \leq \varepsilon|E|$ edges in $\overrightarrow{G}$ we can get a path from s to t in $\overrightarrow{G}$, and the second reducibility condition is proved. $\square$

Chapter 8

Testing for Eulerian orientations

8.1 Background

In this chapter we consider the property of being Eulerian, which was presented in [HLNT07] as one of the natural orientation properties whose query complexity was still unknown. A directed graph $\overrightarrow{G}$ is called *Eulerian* if for every vertex v in the graph, the in-degree of v is equal to its out-degree. An undirected graph G has an Eulerian orientation $\overrightarrow{G}$ if and only if all the degrees of G are even. Such an undirected graph is called Eulerian also. Throughout this chapter we assume that our underlying undirected graph G is Eulerian.

Despite the local nature of the property of being Eulerian, it turns out to be significantly harder for testing than other properties in the orientations model. In particular, in this chapter we show a super-constant lower bound on the query complexity of this problem, even if the tester has two-sided error.

Eulerian graphs and Eulerian orientations have attracted researchers since the dawn of graph theory in 1736, when Leonard Euler published his solution for the famous "Königsberg bridge problem". Throughout the years, Eulerian graphs have been the subject of extensive research (e.g. [Rob69, Lov76, Tut84, MW96, BW05, Bab06]; see [Fle90, Fle91] for an extensive survey). Aside from their appealing theoretic characteristics, Eulerian graphs have been studied in the context of networking [IKN88] and genetics [PTW01].

Testing for being Eulerian in the orientation model is equivalent to the following problem. We have a known network (a communication network, a transportation system or a piping system) where every edge can transport a unit of "flow" in both directions. Our goal is to know whether the network is "balanced", or far from being balanced, where being balanced means that the number of flows entering every node in the network is equal to the number of flows exiting it (so there is no "accumulation" in the nodes). To examine the network, we detect the flow direction in selected individual edges, which is deemed to be the expensive operation.

The main difficulty in testing orientations for being Eulerian arises from the fact that an orientation might have a small number of unbalanced vertices, and each of them with only a small small imbalance, and yet be far from being Eulerian. This is since trying to balance an unbalanced vertex by inverting some of its incident edges may violate the balance of its balanced neighbors. Thus, we must continue to invert edges along a directed path between a vertex with a positive imbalance and one with a negative imbalance. We call such a path a *correction path*.

In this context we mention the work of Babai [Bab06], who studied the ratio between the diameter of Eulerian digraphs and the diameter of their underlying undirected graphs. While he gave an upper bound for this ratio for vertex-transitive graphs, he showed an infinite family of undirected graphs with diameter 2 which have an Eulerian orientation with diameter $\Omega(n^{1/3})$.

To prove the lower bounds, we concentrate on bounded-degree graphs, and use the toroidal grid to prove non-constant 1-sided and 2-sided lower bounds. These bounds are quite surprising, as bounded-degree graphs have a constant size witness for not being Eulerian, namely, the edges incident with one unbalanced vertex. In contrast, we saw that the *st*-connectivity property studied in the previous chapter, whose witness must include a cut in the graph, is testable with a constant number of queries in the orientation model. However, in other testing models there are known super-constant lower bounds also for properties that have constant-size witness. For instance, in [BSHR05] it is proved that testing whether a truth assignment satisfies a known 3CNF formula requires a linear number of queries for some formulas.

Before proceeding to the proofs of the lower bounds, we refer the reader to Section 6.2 above for basic preliminaries and notations used in the orientation model. Additional specific preliminaries for this chapter are given in the next section.

8.2 Specific preliminaries

Consider an algorithm that tests an orientation $\overrightarrow{G}$ of G. At a given moment, we represent the edges that the algorithm has queried so far by a directed *knowledge graph* $\overrightarrow{H} = (V, \overrightarrow{E_H})$, where $\overrightarrow{E_H} \subseteq \overrightarrow{E}$. We say that a cut $M = (U, V \setminus U)$ of G is *valid* with respect to a knowledge graph $\overrightarrow{H}$ if

$$|\overrightarrow{E_H}(U, V \setminus U)| \;\le\; \frac{1}{2}|E(U, V \setminus U)| \quad \text{and} \quad |\overrightarrow{E_H}(V \setminus U, U)| \;\le\; \frac{1}{2}|E(U, V \setminus U)|.$$

Otherwise, M is called *invalid.* Clearly, if $\overrightarrow{G}$ is Eulerian, then every knowledge graph $\overrightarrow{H}$ of $\overrightarrow{G}$ contains only valid cuts. In the following we also show that the only negative witness of $\overrightarrow{G}$ (for *not* being Eulerian) is a knowledge graph that contains some invalid cut.

We say that a (valid) cut $M = (U, V \setminus U)$ of G is *restricting* with respect to $\overrightarrow{H}$ if

$$|\overrightarrow{E_H}(U, V \setminus U)| \;=\; \frac{1}{2}|E(U, V \setminus U)| \quad \text{or} \quad |\overrightarrow{E_H}(V \setminus U, U)| \;=\; \frac{1}{2}|E(U, V \setminus U)|.$$

Note that, given that $\overrightarrow{G}$ is Eulerian, a restricting cut with respect to $\overrightarrow{H}$ forces the orientations of all the unqueried edges in the cut. We say that two restricting cuts M_1 and M_2 are *conflicting* (with respect to a knowledge graph $\overrightarrow{H}$) if they force contrasting orientations of at least one unqueried edge.

Lemma 8.2.1. *Let $\overrightarrow{H}$ be a knowledge graph of $\overrightarrow{G}$ and suppose that all the cuts in G are valid with respect to $\overrightarrow{H}$. Then there are no conflicting cuts with respect to $\overrightarrow{H}$.*

Proof. Assume, on the contrary, that $M_1 = (V_1, V \setminus V_1)$ and $M_2 = (V_2, V \setminus V_2)$ are conflicting with respect to $\overrightarrow{H}$. That is, there exists an edge $\{u, w\} \in E$, which was not queried and hence is not oriented in $\overrightarrow{H}$, which is forced to have contrasting orientations by M_1 and M_2. Without loss of generality, assume that $u \in V_1 \setminus V_2,\ w \in V_2 \setminus V_1$, and

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| \;=\; \frac{1}{2} \cdot |E(V_1,\ V \setminus V_1)|, \tag{8.1}$$

$$|\overrightarrow{E_H}(V_2,\ V \setminus V_2)| \;=\; \frac{1}{2} \cdot |E(V_2,\ V \setminus V_2)|. \tag{8.2}$$

Thus, M_1 forces e to be oriented from w to u, whereas M_2 forces e to be oriented from u to w.

Recall now that all the cuts in G are valid with respect to $\overrightarrow{H}$. Consider the cuts $(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))$ and $(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))$. We have

$$|\overrightarrow{E_H}(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \;\leq\; \frac{1}{2} \cdot |E(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \tag{8.3}$$

and

$$|\overrightarrow{E_H}(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \;\leq\; \frac{1}{2} \cdot |E(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \tag{8.4}$$

since these cuts are valid. Note that

$$|E(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \;+\; |E(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \tag{8.5}$$

$$= |E(V_1,\ V \setminus V_1)| + |E(V_2,\ V \setminus V_2)| - 2 \cdot |E(V_1 \setminus V_2,\ V_2 \setminus V_1)|$$

and

$$|\overrightarrow{E_H}(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \;+\; |\overrightarrow{E_H}(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| = \tag{8.6}$$

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| + |\overrightarrow{E_H}(V_2,\ V \setminus V_2)| - |\overrightarrow{E_H}(V_1 \setminus V_2,\ V_2 \setminus V_1)| - |\overrightarrow{E_H}(V_2 \setminus V_1,\ V_1 \setminus V_2)|.$$

Summing Equation (8.1) with Equation (8.2) yields

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| + |\overrightarrow{E_H}(V_2,\ V \setminus V_2)| = \frac{1}{2}\left(|E(V_1,\ V \setminus V_1)|\ +\ |E(V_2,\ V \setminus V_2)|\right). \quad (8.7)$$

Summing Inequality (8.3) with Inequality (8.4) yields

$$\overrightarrow{E_H}(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))|\ +\ |\overrightarrow{E_H}(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \leq \quad (8.8)$$

$$\frac{1}{2}\left(|E(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))|\ +\ |E(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))|\right).$$

Substituting Equations (8.5) and (8.6) in Inequality (8.8) we obtain:

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| + |\overrightarrow{E_H}(V_2,\ V \setminus V_2)| - |\overrightarrow{E_H}(V_1 \setminus V_2,\ V_2 \setminus V_1)| - |\overrightarrow{E_H}(V_2 \setminus V_1,\ V_1 \setminus V_2)| \leq$$

$$\frac{1}{2}\left(|E(V_1,\ V \setminus V_1)|\ +\ |E(V_2,\ V \setminus V_2)|\right) - |E(V_1 \setminus V_2,\ V_2 \setminus V_1)|.$$

Now, from Equation (8.7) we have:

$$|\overrightarrow{E_H}(V_1 \setminus V_2,\ V_2 \setminus V_1)|\ +\ |\overrightarrow{E_H}(V_2 \setminus V_1,\ V_1 \setminus V_2)|\ \geq\ |E(V_1 \setminus V_2,\ V_2 \setminus V_1)|.$$

That is, all the edges in $E(V_1 \setminus V_2,\ V_2 \setminus V_1)$ are oriented in $\overrightarrow{H}$. This is a contradiction to our assumption that $\{u, w\} \in E(V_1 \setminus V_2,\ V_2 \setminus V_1)$ was not yet oriented. □

Lemma 8.2.2. *Suppose that $\overrightarrow{H}$ is a knowledge graph that does not contain invalid cuts. Then $\overrightarrow{H}$ is extensible to an Eulerian orientation $\overrightarrow{G} = (V, \overrightarrow{E_G})$ of G. That is, $\overrightarrow{E_H} \subseteq \overrightarrow{E_G}$.*

Proof. We orient unoriented edges in the following manner. If there exists a restricting cut with unoriented edges, we orient one of them as obliged by the cut. According to Lemma 8.2.1, this will not invalidate any of the other cuts in the graph, and so we may continue. If there are no restricting cuts in the graph, we arbitrarily orient one unoriented edge and repeat (and this cannot violate any cut in the graph since there were no restricting cuts). Eventually, after orienting all the edges, we receive a complete orientation of G whose cuts are all valid, and thus it is Eulerian. □

8.3 Lower bounds for bounded-degree graphs

In this section we prove the following theorems.

Theorem 8.3.1. *Non-adaptive testing for Eulerian orientations of bounded degree graphs with 2-sided error requires $\Omega\left(\sqrt{\frac{\log m}{\log \log m}}\right)$ queries. Consequently, adaptive testing requires*

$\Omega(\log\log m)$ *queries.*

Theorem 8.3.2. *Non-adaptive testing for Eulerian orientations of bounded degree graphs with* 1*-sided error requires* $\frac{1}{100}m^{1/4}$ *queries. Consequently, adaptive testing with* 1*-sided error requires* $\Omega(\log m)$ *queries.*

We start by proving Theorem 8.3.1. Next, based on the proof of Theorem 8.3.1, we prove Theorem 8.3.2.

8.3.1 Overview of the proof of Theorem 8.3.1

The proof of Theorem 8.3.1 is by Yao's principle, as formalized in Lemma 3.4.1. Namely, for infinitely many natural numbers m we define a graph G_m with m edges, and two distributions over the orientations of G_m. The first distribution, $\mathcal{P}_m$, will contain only Eulerian orientations of G_m, and the second distribution, $\mathcal{F}_m$, will contain orientations that are with high probability ε-far from being Eulerian, where $\varepsilon = 1/64$. Then we show that any non-adaptive deterministic algorithm that makes $o\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ orientation queries cannot distinguish between the distributions $\mathcal{P}_m$ and $\mathcal{F}_m$ with probability larger than 1/5.

All our underlying graphs G_m are two dimensional *tori*, which are 4-regular graphs having a highly symmetric structure (the exact definition is given below). We exploit this symmetry to construct distributions $\mathcal{P}_m$ and $\mathcal{F}_m$ such that in both cases, for any fixed set Q of $o\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ edges, the orientation of every pair in Q has (with high probability) either no correlation at all, or a correlation that is identical in both cases.

To construct these distributions we build the orientations from repeated "patterns" of varying sizes, and show that in order to succeed, a deterministic algorithm must know the approximate size of these patterns.

8.3.2 Torus – formal definition

Recall that we denote by $[\ell]$ the set $\{1, 2, \ldots, \ell\}$. For $i, j \in [\ell]$ we let $\oplus$ denote addition modulo ℓ, that is:

$$i \oplus j = \begin{cases} i + j & , \ i + j \leq \ell \\ i + j - \ell & , \ i + j > \ell \end{cases} .$$

Given a graph G and two edges $e_1, e_2 \in E(G)$, we define the *distance* between e_1 and e_2 (or shortly $\mathrm{dist}(e_1, e_2)$) as the minimal distance between an endpoint of e_1 and an endpoint of e_2. For an edge $e = \{u, v\} \in E(G)$ and a vertex $w \in V(G)$ we define the *distance* of e from w, or shortly $\mathrm{dist}(e, w)$, as the minimum of $\mathrm{dist}(u, w)$ and $\mathrm{dist}(v, w)$. We stress that in this chapter even in oriented graphs $\overrightarrow{G}$, the distances between edges and vertices are still measured on the underlying (undirected) graph G.

We define the *imbalance* of a vertex v in a directed graph as $\mathrm{ib}(v) \overset{\mathrm{def}}{=} \deg_{out}(v) - \deg_{in}(v)$. We say that a vertex $v \in V$ is a *spring* if $\mathrm{ib}(v) > 0$. We say that v is a *drain* if $\mathrm{ib}(v) < 0$. If $\mathrm{ib}(v) = 0$ then we say that v is *balanced* in $\overrightarrow{G}$. Clearly, a digraph is *Eulerian* if all its vertices are balanced, and whenever all the vertices of an undirected graph have even degrees, there always exists some Eulerian orientation on it.

Definition 24 (Torus). *A torus is a two dimensional cyclic grid. Formally, an $\ell \times \ell$ torus is the graph $T = (V, E)$ on $n = \ell^2$ vertices $V = \{v_{i,j} : i \in [\ell],\ j \in [\ell]\}$ and $m = 2n$ edges $E = E_H \cup E_V$, where $E_H = \Big\{\{v_{i,j_1}, v_{i,j_2}\} : j_2 = j_1 \oplus 1\Big\}$ and $E_V = \Big\{\{v_{i_1,j}, v_{i_2,j}\} : i_2 = i_1 \oplus 1\Big\}$. We also call E_H the set of* horizontal edges*, and call E_V the set of* vertical edges*. Two edges $e_1, e_2 \in E$ are* perpendicular *if one of them is horizontal and the other is vertical, and otherwise they are* parallel*.*

Given an orientation $\overrightarrow{T}$ of T, we say that a horizontal edge $e = \{v_{i,j}, v_{i,j\oplus 1}\}$ is directed to the right *if $v_{i,j}$ is the start-point of e, and otherwise we say that e is* directed to the left*. Similarly, we say that a vertical edge $e = \{v_{i,j}, v_{i\oplus 1,j}\}$ is* directed upwards *if $v_{i,j}$ is the start-point of e, and else it is* directed downwards*.*

To avoid irrelevant special case arguments, throughout this section we always assume that ℓ is even. Now we define a graph operation that is later used in the construction of the distributions $\mathcal{P}_m$ and $\mathcal{F}_m$.

Definition 25 (shifting). *Let T be an $\ell \times \ell$ torus, and let $a, b \in [\ell]$. We say that a mapping $\pi : V(T) \to V(T)$ is an (a,b)-*shifting *of T if for every $v_{i,j} \in V(T)$ we have $\pi(v_{i,j}) = v_{i\oplus a, j\oplus b}$. Note that this is an isomorphism of T. Given an orientation $\overrightarrow{T}$ of T, we define its shifting to preserve the directions of the edges. Namely, if $e = \{v_{i,j}, v_{i',j'}\}$ is directed from $v_{i,j}$ to $v_{i',j'}$ in T, then in the (a,b)-shifting of T the edge $e' = \{\pi(v_{i,j}), \pi(v_{i',j'})\}$ is directed from $\pi(v_{i,j})$ to $\pi(v_{i',j'})$.*

8.3.3 Defining two auxiliary distributions

In this section we describe two simple distributions, $\mathcal{R}_m$ and $\mathcal{C}_m^{(k)}$, over the orientations of an $\ell \times \ell$ torus T with $m = 2\ell^2$ edges. We later build the final distributions $\mathcal{F}_m$ and $\mathcal{P}_m$ based on $\mathcal{R}_m$ and $\mathcal{C}_m^{(k)}$ respectively.

The distribution $\mathcal{R}_m$ is simply a random orientation of T's edges. Namely, in $\overrightarrow{T} \sim \mathcal{R}_m$ the orientation of each edge $e \in E(T)$ is chosen uniformly at random, independently of the other edges.

Lemma 8.3.3. *Let $\overrightarrow{T}$ be an orientation of a torus T with m edges, distributed according to the distribution $\mathcal{R}_m$. With probability $1 - o(1)$, there are at least $n/4 = m/8$ unbalanced vertices in $\overrightarrow{T}$.*

Proof. Recall that $V = \{v_{i,j} : i, j \in [\ell]\}$ is the set of T's vertices. Define a subset

$$I = \{v_{i,j} : i + j \text{ is even}\} \subseteq V$$

of $n/2$ vertices. Observe that I is an independent set in T, and so every vertex $v_i \in I$ is balanced with probability $x_i = \binom{4}{2}/2^4 = 3/8$ independently from all other vertices in I. By Chernoff's inequality, the probability that at least half of the vertices in I are balanced is bounded by $\exp(-n/64)$. Namely, with probability $1 - o(1)$ there are at least $n/4$ unbalanced vertices in I. $\square$

The second distribution, $\mathcal{C}_m^{(k)}$, is a distribution over Eulerian orientations of T. The parameter k is assumed to divide $\frac{\ell}{2} = \frac{\sqrt{m/2}}{2}$. The Eulerian orientations given by $\mathcal{C}_m^{(k)}$ are constructed as follows. First, the torus T is decomposed into $m/4k$ edge-disjoint $4k$-cycles $c_1, c_2, \ldots, c_{m/4k}$, where each cycle c_i has exactly four "corner" vertices, that is vertices that are adjacent to both vertical and horizontal edges. Then, for each cycle c_i, one of its two possible Eulerian orientations is chosen uniformly at random, independently of other cycles. Let $\overrightarrow{T}'$ denote the orientation of T at this stage. Finally, $a, b \in [\ell]$ are chosen uniformly at random, and $\overrightarrow{T}$ is set to be an (a, b)-shift of $\overrightarrow{T}'$.

In what follows, for a pair of edges $e_i, e_j \in T$ and an orientation $\overrightarrow{T}$ of T, we say that e_i and e_j are *independent* if either $\overrightarrow{T} \sim \mathcal{R}_m$, or if $\overrightarrow{T} \sim \mathcal{C}_m^{(k)}$ and the edges e_i,e_j reside in different $4k$-cycles c_i, c_j. A set $Q \subseteq E(T)$ is independent if all pairs $e_1, e_2 \in Q$ are independent. Observe that if Q is independent, then the orientation of every $e \in Q$ is distributed uniformly at random, independently of the other members of Q. Clearly if $\overrightarrow{T}$ is distributed according to $\mathcal{R}_m$ then every set $Q \subseteq E(T)$ is independent, but this is not the case for orientations distributed according to $\mathcal{C}_m^{(k)}$. In the following lemmas we claim that under some conditions on the set Q, with high probability it is independent even if $\overrightarrow{T}$ is distributed according to $\mathcal{C}_m^{(k)}$.

Lemma 8.3.4. *Let T be an $\ell \times \ell$ torus with m edges, and let $e_1, e_2 \in E(T)$ be two perpendicular edges of T. Let $\overrightarrow{T}$ be an orientation of T distributed according to $\mathcal{C}_m^{(k)}$, for an integer k that divides $\ell/2$. Then the probability that e_1 and e_2 are independent is at least $1 - \frac{2}{k}$.*

Proof. Suppose that e_1 and e_2 are not independent. Then they must reside in the same cycle c_0. Observe that there is a unique vertex v_0 defined by e_1 and e_2, which must be one of the "corner" vertices of c_0, namely, v_0 must be an endpoint of both horizontal and vertical edges of c_0 . The number of $4k$-cycles that are required to cover all edges of T is $\frac{2\ell^2}{4k} = \frac{\ell^2}{2k}$, so the total number of corner vertices is $4 \cdot \frac{\ell^2}{2k} = \frac{2\ell^2}{k}$. Therefore, the fraction of corner vertices is exactly $2/k$. Since in the last step of the definition of $\mathcal{C}_m^{(k)}$ the orientation is randomly shifted, the probability that v_0 is not a corner of any of the $4k$-cycles is exactly

$1 - 2/k$. □

Lemma 8.3.5. *Let T be an $\ell \times \ell$ torus with m edges and let k be an integer that divides $\ell/2$. Let $Q \subseteq E(T)$ be a set of $o(\sqrt{k})$ edges such that for every pair $e_1, e_2 \in Q$ either* $\operatorname{dist}(e_1, e_2) > 2k$*, or e_1 and e_2 are perpendicular. Then for an orientation $\overrightarrow{T}$ of T distributed according to $\mathcal{C}_m^{(k)}$, the probability that Q is independent is $1 - o(1)$.*

Proof. Fix a pair $e_1, e_2 \in Q$. If $\operatorname{dist}(e_1, e_2) > 2k$ then e_1 and e_2 must reside in different $4k$-cycles, and hence they are independent. Otherwise, e_1, e_2 are perpendicular, and by Lemma 8.3.4 they are independent with probability at least $1 - \frac{2}{k}$. Now the proof is completed by applying the union bound for all $o(k)$ pairs $e_1, e_2 \in Q$. □

8.3.4 Defining the distributions $\mathcal{P}_m$ and $\mathcal{F}_m$

First we need to define the following operation, that allows us to construct orientations of large tori based on orientations of small ones. This operation preserves the distance of the orientations we use from being Eulerian. In particular, if it is applied on $\overrightarrow{T} \sim \mathcal{R}_m$, then the resulting orientation is far from being Eulerian as long as $\overrightarrow{T}$ contains many unbalanced vertices (and then we can use Lemma 8.3.3), and on the other hand, if it is applied on the Eulerian orientation $\overrightarrow{T} \sim \mathcal{C}_m^{(k)}$ then the resulting orientation is also Eulerian.

Definition 26 (t-tiling)**.** *Let $\overrightarrow{T}$ be a directed $\ell \times \ell$ torus, and let t be a natural number. We define the* t-tiling of *$\overrightarrow{T}$ as a $2t\ell \times 2t\ell$ directed torus $\overrightarrow{T}^t$ which is constructed as follows.*

For convenience, let $v_{i,j}$, $i, j \in [\ell]$ denote the vertices of $\overrightarrow{T}$ and let $u_{i,j}$, $i, j \in [2t\ell]$ denote the vertices of $\overrightarrow{T}^t$. First, we partition the $2t\ell \times 2t\ell$ torus $\overrightarrow{T}^t$ into ℓ^2 disjoint $2t \times 2t$ grids $\{G_{i,j}\}_{i,j \in [\ell]}$, where every grid $G_{i,j}$ is associated with the vertex $v_{i,j} \in V(\overrightarrow{T})$. The partition of $\overrightarrow{T}^t$ into ℓ^2 grids $G_{i,j}$ is according to the original layout of the vertices $v_{i,j}$ in the $\ell \times \ell$ torus $\overrightarrow{T}$. Formally, For every $i, j \in [\ell]$, the grid $G_{i,j}$ contains the vertices

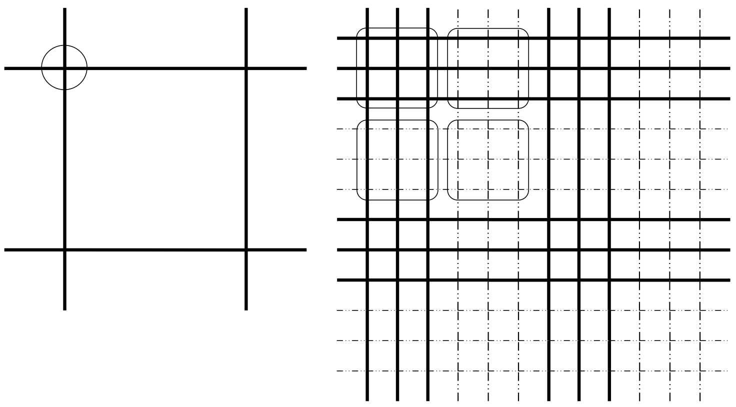
$$V(G_{i,j}) = \{u_{i',j'} : 2t(i-1) < i' \leq 2ti,\ 2t(j-1) < j' \leq 2tj\}.$$

Next, every grid $G_{i,j}$ is partitioned into the following four disjoint $t \times t$ grids.

- *The grid $R_{i,j}$, which we call the* representative *of the vertex $v_{i,j} \in V(\overrightarrow{T})$.*
- *The grid $P^{\mathrm{r}}_{i,j}$, which we call the* padding to the right *of $R_{i,j}$.*
- *The grid $P^{\mathrm{b}}_{i,j}$, which we call the* padding below *$R_{i,j}$.*
- *The grid $P^{\mathrm{rb}}_{i,j}$, which we call the* padding diagonally *to $R_{i,j}$.*

These four $t \times t$ grids are composed into a $2t \times 2t$ grid $G_{i,j}$ by placing $R_{i,j}$ in the upper left corner, placing $P^{\mathrm{r}}_{i,j}$ to the right of $R_{i,j}$, placing $P^{\mathrm{b}}_{i,j}$ below $R_{i,j}$ and placing $P^{\mathrm{rb}}_{i,j}$ below $P^{\mathrm{r}}_{i,j}$. See an example of how a 2×2 torus corresponds to its 3-tiling in Figure 8.1.

Figure 8.1: A 2×2 torus and its 3-tiling. The vertex $v_{2,1}$ is encircled in the original torus, and its corresponding sub-grids $R_{2,1}, P^{\mathrm{r}}_{2,1}, P^{\mathrm{b}}_{2,1}$ and $P^{\mathrm{rb}}_{2,1}$ are encircled in the 3-tiling.



The orientation $\overrightarrow{T}^t$ is defined as follows. For every $v_{i,j} \in V(\overrightarrow{T})$, let $r^1_{i,j}, r^2_{i,j}, \ldots, r^t_{i,j} \in V(\overrightarrow{T}^t)$ be the t vertices on the main diagonal of the representative grid $R_{i,j}$. For every edge $e = \{v_{i,j}, v_{i',j'}\} \in E(\overrightarrow{T})$ directed from $v_{i,j}$ to $v_{i',j'}$ and every $h \in [t]$, we orient the edges on the shortest path from $r^h_{i,j}$ to $r^h_{i',j'}$ in a way that forms a directed path from $r^h_{i,j}$ to $r^h_{i',j'}$. For every edge $e' \in E(\overrightarrow{T}^t)$ that participates in this path, we call e the originating edge *of e', and we use the notation* $\mathrm{org}(e') \overset{\text{def}}{=} e$.

As for the remaining edges (the "padding" part), all horizontal edges are directed to the right, and all vertical edges are directed upwards, as per Definition 24. See an example in Figure 8.2. For every padding edge e we define $\mathrm{org}(e) \overset{\text{def}}{=} \emptyset$*, since they have no origin in T.*

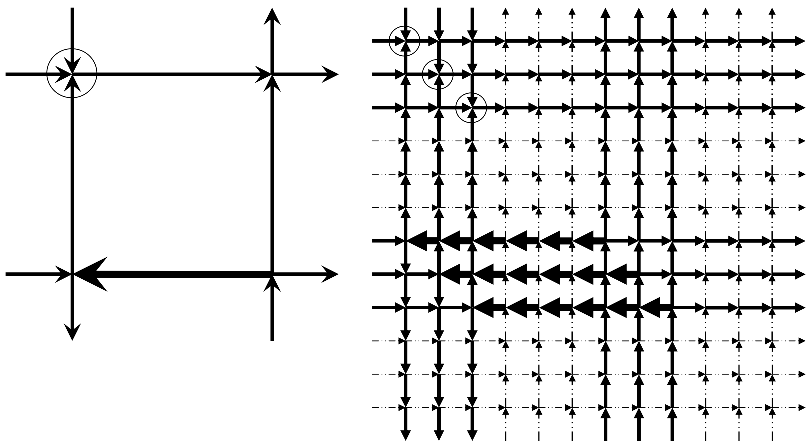
The next lemma states that a tiling of an Eulerian torus is also Eulerian, while on the other hand, a tiling of a torus with many unbalanced vertices results with a torus which is far from being Eulerian.

Lemma 8.3.6. *Let $\overrightarrow{T}$ be a directed $\ell \times \ell$ torus on $n = \ell^2$ vertices, and let $\overrightarrow{T}^t$ be the t-tiling of $\overrightarrow{T}$ for some natural number t. Then ,*

- *If $\overrightarrow{T}$ is Eulerian, then $\overrightarrow{T}^t$ is also Eulerian.*
- *For every $0 < \delta < 1$, if $\overrightarrow{T}$ contains $\delta n = \delta \ell^2$ unbalanced vertices, then $\overrightarrow{T}^t$ is $\frac{\delta}{16}$-far from being Eulerian.*

Proof. The first statement of the lemma follows directly from Definition 26. Assume now that $\overrightarrow{T}$ has δn unbalanced (spring or drain) vertices. According to the definition of a

Figure 8.2: A 2×2 directed torus and the corresponding 3-tiling. *The vertex $v_{2,1}$ is encircled in the original torus, and the corresponding vertices $r^1_{2,1}, r^2_{2,1}$ and $r^3_{2,1}$ are encircled in its tiling. In addition, the edge $\{v_{1,2}, v_{1,1}\}$ is emphasized in the original torus, and the corresponding edges are emphasized in the 3-tiling. Notice the circular orientation of the dashed edges (the padding part).*



t-tiling, for every unbalanced vertex $v_{i,j} \in V(\overrightarrow{T})$ we have exactly t unbalanced vertices $r^1_{i,j}, r^2_{i,j}, \ldots, r^t_{i,j}$ on the main diagonal of $v_{i,j}$'s representative grid $R_{i,j}$ in $\overrightarrow{T}^t$, so the number of unbalanced vertices in $\overrightarrow{T}^t$ is δnt. In addition, whenever $v_{i,j}$ is a spring (respectively drain) vertex in $\overrightarrow{T}$, the vertices $r^1_{i,j}, r^2_{i,j}, \ldots, r^t_{i,j}$ are also springs (respectively drains) in $\overrightarrow{T}^t$, so every pair of spring-drain vertices must reside in different grids $R_{i,j}$ and $R_{i',j'}$. This implies that (due to the padding parts) the distance from any spring vertex to any drain vertex in $\overrightarrow{T}^t$ is at least t. Consequently, every correction path in $\overrightarrow{T}^t$ must be of length at least t. Since every correction path in $\overrightarrow{T}^t$ can balance at most two unbalanced vertices, and since the length of every such path is at least t, we conclude that $\overrightarrow{T}^t$ is $\frac{t\delta nt/2}{|E(\overrightarrow{T}^t)|} = \frac{\delta nt^2/2}{8nt^2} = \frac{\delta}{16}$-far from being Eulerian. □

Lemma 8.3.7. *Let $\overrightarrow{T}^t$ be a t-tiling of a randomly oriented $\ell \times \ell$ torus $\overrightarrow{T} \sim \mathcal{R}_m$. Then with probability $1 - o(1)$, $\overrightarrow{T}^t$ is $1/64$-far from being Eulerian.*

Proof. Follows by combining Lemma 8.3.6 (with $\delta = 1/4$) and Lemma 8.3.3. □

Now we describe the distributions $\mathcal{P}_m$ and $\mathcal{F}_m$ over the orientations of an $\ell \times \ell$ torus T, where $m = 2\ell^2$. To avoid divisibility concerns, we assume that $\ell = 2^k$, and $k = 2^b$ for some natural number $b > 1$. It is easy to verify that the same proof works also for general values of ℓ and k by using rounding as appropriate.

Distribution $\mathcal{P}_m$: Choosing $\overrightarrow{T} \sim \mathcal{P}_m$ is done according to the following steps.

- Choose s uniformly at random from the range $[k/4, k/2]$. Let $t = 2^s$, that is, t can take $\frac{\log \ell}{4}$ values in the range $[n^{1/4}, n^{1/2}]$.
- For an $\frac{\ell}{2t} \times \frac{\ell}{2t}$ torus H, choose a random orientation $\overrightarrow{H}$ according to the distribution $\mathcal{C}_m^{(k)}$.
- Set $\overrightarrow{T}'$ to be the t-tiling of $\overrightarrow{H}$.
- Choose $a, b \in [\ell]$ uniformly at random, and set $\overrightarrow{T}$ to be the (a, b)-shift of $\overrightarrow{T}'$.

Distribution $\mathcal{F}_m$: Choosing $\overrightarrow{T} \sim \mathcal{F}_m$ is done according to the following steps.

- Choose s uniformly at random from the range $[k/4, k/2]$ and set $t = 2^s$.
- For an $\frac{\ell}{2t} \times \frac{\ell}{2t}$ torus H, choose a random orientation $\overrightarrow{H}$ according to the distribution $\mathcal{R}_m$.
- Set $\overrightarrow{T}'$ as the t-tiling of $\overrightarrow{H}$.
- Choose $a, b \in [\ell]$ uniformly at random, and set $\overrightarrow{T}$ to be the (a, b)-shift of $\overrightarrow{T}'$.

8.3.5 Bounding the probability of distinguishing between $\mathcal{P}_m$ and $\mathcal{F}_m$

According to Lemma 8.3.6, $\overrightarrow{T} \sim \mathcal{P}_m$ is always Eulerian, and according to Lemma 8.3.7, $\overrightarrow{T} \sim \mathcal{F}_m$ is with high probability $\frac{1}{64}$-far from being Eulerian. Our aim is to show that any non-adaptive deterministic algorithm making $o\left(\sqrt{\frac{\log \ell}{\log \log \ell}}\right)$ queries will fail to distinguish between orientations that are distributed according to $\mathcal{P}_m$, and those that are distributed according to $\mathcal{F}_m$.

Let $Q \subseteq E(T)$ be a fixed set of at most $\frac{1}{10}\sqrt{\frac{\log \ell}{\log \log \ell}}$ edges queried by a deterministic algorithm. Let $\text{org}(Q) \stackrel{\text{def}}{=} \{\text{org}(e) : e \in Q\} \subseteq E(H)$ be the set of Q's originating edges (see Definition 26). Clearly, if $\overrightarrow{T}$ is distributed according to $\mathcal{F}_m$, then the set $\text{org}(Q) \subseteq E(H)$ is independent in H, and hence the distribution of the orientations of the edges in $\text{org}(Q) \subseteq E(H)$ is uniform. Therefore, it is enough to show that whenever $\overrightarrow{T}$ is distributed according to $\mathcal{P}_m$, with probability at least $\frac{4}{5}$ the set $\text{org}(Q)$ is independent in H too. In order to prove this we need the next lemma.

Lemma 8.3.8. *Let $q_1, q_2 \in E(T)$ be a pair of edges within distance x. Let $\overrightarrow{T}$ be a random orientation of T, chosen according to distribution $\mathcal{P}_m$. Then*

$$\Pr_s[\frac{t}{4k} \leq x \leq 4tk] < \frac{8(\log \log \ell + 2)}{\log \ell}.$$

Proof. Observe that there are at most $2\log k + 4 = 2\log\log \ell + 4$ values of s for which a fixed number x can satisfy $2^{s-\log k-2} = \frac{t}{4k} \leq x \leq 4tk = 2^{s+\log k+2}$. On the other hand, s is distributed uniformly among $\frac{k}{4} = \frac{\log \ell}{4}$ possible values, so the lemma follows. $\square$

For a set $Q \subseteq E(T)$ and an orientation $\overrightarrow{T}$ of T, we define the following event I_Q: For all pairs $q_1, q_2 \in Q$ either $\operatorname{dist}(q_1, q_2) < \frac{t}{4k}$ or $\operatorname{dist}(q_1, q_2) > 4tk$ in T. The next lemma is a direct consequence of Lemma 8.3.8.

Lemma 8.3.9. *Let $Q \subseteq E(T)$ be a fixed set of $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}}$ edges, and let $\overrightarrow{T}$ be a random orientation of T, according to distribution $\mathcal{P}_m$. Then the event I_Q occurs with probability at least* $9/10$.

Proof. Follows by applying the union bound on the inequality from Lemma 8.3.8 for all pairs $q_1, q_2 \in Q$. $\square$

Recall that since $|\operatorname{org}(Q)| = o(\sqrt{\log \ell}) = o(k)$, according to Lemma 8.3.5 it is enough to show that with probability at least $\frac{4}{5}$ the set $\operatorname{org}(Q)$ is such that for every pair $\operatorname{org}(e_1), \operatorname{org}(e_2) \in \operatorname{org}(Q)$ either $\operatorname{dist}\Big(\operatorname{org}(e_1), \operatorname{org}(e_2)\Big) > 2k$ in H, or $\operatorname{org}(e_1)$ and $\operatorname{org}(e_2)$ are perpendicular. Clearly, if $\operatorname{dist}(e_1, e_2) > 4tk$ in T, then $\operatorname{dist}\Big(\operatorname{org}(e_1), \operatorname{org}(e_2)\Big) > 2k$ in H. We take care of the other case in the next lemma.

Lemma 8.3.10. *Conditioned on the event I_Q, with probability $1 - o(1)$ for every pair of edges $e_1, e_2 \in Q$ such that $\operatorname{dist}(e_1, e_2) < \frac{t}{4k}$ one of the following holds:* (1) *e_1, e_2 are perpendicular;* (2) *one of e_1, e_2 has no origin in H;* (3) $\operatorname{org}(e_1) = \operatorname{org}(e_2)$.

Before proving the lemma, observe that since I_Q occurs with probability at least $9/10$, the conditions of Lemma 8.3.5 are satisfied by Q with (total) probability at least $\frac{9}{10} - o(1)$, and hence $\operatorname{org}(Q)$ is independent with probability at least $\frac{9}{10} - 2 \cdot o(1) > 4/5$, where the probabilities are taken over $\mathcal{P}_m$.

Proof. Fix a pair $e_1, e_2 \in Q$ such that $\operatorname{dist}(e_1, e_2) < \frac{t}{4k}$. If one of them has no originating edge in H then we are done. Otherwise, since $\operatorname{dist}(e_1, e_2) < \frac{t}{4k}$, by the definition of the t-tiling $\operatorname{org}(e_1)$ and $\operatorname{org}(e_2)$ must have a common endpoint in H, say $v_{i,j}$. If e_1 and e_2 are perpendicular, then again we are done. On the other hand, if e_1 and e_2 are parallel, then in order to have different origins they must be separated by the the main diagonal of $R_{i,j}$ (the representative grid of the common vertex $v_{i,j}$). Notice that this may happen only if the distance of both e_1 and e_2 from the main diagonal is at most $\frac{t}{4k}$. But the probability that an edge is within that distance from the main diagonal of some representative grid is at most $\frac{t}{4k} t \frac{1}{t^2} = \frac{1}{4k} = o(\frac{\log\log \ell}{\log \ell})$. Now the proof is completed by applying the union bound for all pairs $e_1, e_2 \in Q$. $\square$

8.3.6 Proof of Theorem 8.3.1

Proof. Let $Q \subseteq E(T)$ be the fixed set of $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}}$ edge queries that a deterministic algorithm makes. For every t let $\mathcal{P}_m^t$ and $\mathcal{F}_m^t$ be respectively the conditioning of $\mathcal{P}_m$ and $\mathcal{F}_m$ on a fixed value of t. Note that fixing t, $\mathrm{org}(Q)$ becomes defined and, in particular, it is the same fixed set for inputs drawn according to $\mathcal{P}_m^t$ and $\mathcal{F}_m^t$. Now, for every t, if $\overrightarrow{T}$ is distributed according to $\mathcal{F}_m^t$, then the set $\mathrm{org}(Q)$ of originating edges is independent. On the other hand, by Lemmas 8.3.8, 8.3.9, and 8.3.10 the set $\mathrm{org}(Q)$ is independent with probability at least $4/5$ (with respect to the choice of t). Thus with probability at least $4/5$ (over the choice of s which determines t), the set $\mathrm{org}(Q)$ is independent which implies that the restriction of $\mathcal{P}_m^t$ on Q is identical to that of $\mathcal{F}_m^t$. Altogether, this implies that distinguishing between the distributions with probability larger than $1/5$ requires more than $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}} = \Omega\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ queries. $\square$

8.3.7 Overview of the proof of Theorem 8.3.2

As opposed to testers with 2-sided error, a tester with 1-sided error is not allowed to reject the input unless a negative witness was found. In our case, as claimed in Lemma 8.2.1, the only possible witness that an orientation is not Eulerian is an invalid cut, i.e. a (possibly partial) cut that cannot be made balanced under any orientation of the non-queried edges.

Following this observation, we prove Theorem 8.3.2 using the distribution $\mathcal{F}_\ell$ defined in Section 8.3.4. First, we define distributions $\mathcal{F}'_\ell$ that are similar to the distributions $\mathcal{F}_\ell$, except that t is fixed to be $\ell/16$, and the orientation $\overrightarrow{H}$ of an 8×8 torus H is fixed to be one that makes all 64 vertices fully unbalanced. Then we show that for orientations that are distributed according to $\mathcal{F}'_\ell$, any non-adaptive deterministic algorithm that makes $o(\sqrt{\ell}) = o(\sqrt{m})$ orientation queries cannot find an invalid cut (a negative witness) with probability larger than $1/5$. This will imply that there exists an $\frac{1}{16}$-far orientation on which any randomized tester fails with probability at least $4/5$.

The main idea is as follows. A cut can be invalid (and hence unbalanced) only if both its components contain unbalanced vertices. Let us now fix a cut (A, B) of an $\ell \times \ell$ torus T, and let $\overrightarrow{T}$ be an orientation of T chosen according to $\mathcal{F}'_\ell$. Suppose that indeed both A and B contain unbalanced vertices, and let Q be a subset of the edges in the cut (A, B) that witness its invalidity. Since the underlying graph T is a torus, one can show that either Q contains $\Omega(m^{1/4})$ edges, or otherwise, one of the edges $e \in Q$ must be within distance at most $O(m^{1/4})$ from one of the unbalanced vertices of $\overrightarrow{T}$. Since the number of unbalanced vertices in $\overrightarrow{T} \sim \mathcal{F}'_\ell$ is $O(\ell) = o(m^{1/4})$, and since they are grouped into 64 diagonals of length $\ell/32$, the number of edges that are within distance $O(m^{1/4})$ or less from these unbalanced vertices is bounded by $O(m^{3/4})$. Finally, since $\overrightarrow{T}$ is randomly shifted, the probability that a set Q of size $o(m^{1/4})$ contains any such edge goes to zero.

8.3.8 Proof of Theorem 8.3.2

First we formally define the distribution $\mathcal{F}'_\ell$.

Distribution $\mathcal{F}'_\ell$: Choosing $\overrightarrow{T} \sim \mathcal{F}'_\ell$ is done according to the following steps.

- Set $t = \ell/16$.
- Fix the orientation $\overrightarrow{H}$ of the $\frac{\ell}{2t} \times \frac{\ell}{2t} = 8 \times 8$ torus H, such that all 64 vertices of H are fully unbalanced in $\overrightarrow{H}$ (i.e. no vertex has both incoming and outgoing edges).
- Set $\overrightarrow{T}'$ to be the t-tiling of $\overrightarrow{H}$.
- Pick $a, b \in [\ell]$ uniformly at random and set $\overrightarrow{T}$ to be the (a,b)-shifting of $\overrightarrow{T}'$.

Lemma 8.3.11. *Let T be an $\ell \times \ell$ torus, and let $\overrightarrow{T}$ be an orientation of T distributed according to $\mathcal{F}'_\ell$. Let Q be a fixed set of $\frac{1}{100}m^{1/4}$ edges from $E(T)$. Then the probability (over $\overrightarrow{T} \sim \mathcal{F}'_\ell$) that any of the edges in Q is within distance at most $\sqrt{\ell}$ from an unbalanced vertex $v \in V(\overrightarrow{T})$ is at most $1/5$.*

Proof. Let U denote the set of unbalanced vertices in $\overrightarrow{T} \sim \mathcal{F}'_\ell$. Observe that $|U| = 64t = 4\ell = 4\sqrt{m/2}$, and recall that the vertices in $U \subseteq V(T)$ are grouped into 64 diagonals of length t (see Definition 26). Thus, the number of vertices $v \in V(T)$ that are within distance at most $\sqrt{\ell}$ from some vertex $u \in U$ is bounded by $64 \cdot (t + 2\sqrt{\ell}) \cdot 2\sqrt{\ell} \le 10\ell^{3/2}$. Hence, the probability of a single edge $e \in Q$ satisfying $\mathrm{dist}(e,u) \le \sqrt{\ell}$ for some $u \in U$ is bounded by $20m^{-1/4}$ and the lemma follows. $\square$

We establish the proof of Theorem 8.3.2 using a few lemmas, in which we point out some significant properties of the torus. But first, we give a general lemma about witnesses for not being Eulerian.

Lemma 8.3.12. *Let $G = (V,E)$ be a graph and let $\overrightarrow{G} = (V, \overrightarrow{E})$ be an orientation of G. If a set $Q \subseteq E$ is a witness that $\overrightarrow{G}$ is not Eulerian then Q contains more than half of the edges of some invalid cut (A,B) in $\overrightarrow{G}$, where both A and B are connected sets of vertices.*

Proof. Recall that, by Lemma 8.2.1, Q contains more than half of the edges of an invalid cut, say (A', B'). Without loss of generality we assume that $|\overrightarrow{E}(A',B')| > \frac{1}{2}|E(A',B')|$. Hence, Q contains more than $\frac{1}{2}|E(A',B')|$ edges going from A' to B'. Let $A_1, \ldots, A_r$ be the connected components of A'. Note that (A',B') is a disjoint union of $(A_1,B'), \ldots, (A_k,B')$. Using averaging calculations, we obtain that there exists a connected component A_i such that Q contains more than $\frac{1}{2}|E(A_i,B')|$ edges going from A_i to B'. Note in addition that there are no edges between A_i and other connected components A_k's of A', and thus $(A_i,B') = (A_i, V \setminus A_i)$. We conclude that Q contains more than $\frac{1}{2}|E(A_i, V \setminus A_i)|$ edges going from A_i to $V \setminus A_i$. Now, let $B_1, \ldots, B_s$ be the connected components of $V \setminus A_i$. Note that $(A_i, V \setminus A_i)$ is a disjoint union of $(A_i, B_1), \ldots, (A_i, B_s)$. Using averaging calculations, we obtain that there exists a connected component B_j such that Q contains more than $\frac{1}{2}|E(A_i, B_j)|$ edges going from A_i to B_j. Note in addition that there are no edges between B_j and other connected components B_k's of $V \setminus A_i$, and therefore $(A_i, B_j) = (V \setminus B_j, B_j)$. We conclude that Q contains more than $\frac{1}{2}|E(V \setminus B_j, B_j)|$ edges going from $V \setminus B_j$ to B_j, and hence, Q is a witness to the invalidity of $(V \setminus B_j, B_j)$. B_j is clearly connected. To complete the proof, we need to show that $V \setminus B_j$ is connected as well. Recall that $V \setminus B_j$ is a union of A_i and all the connected components B_k's of $V \setminus A_i$ for $k \neq j$. The B_k's are not connected to each other. However, the torus T is a connected graph, and therefore, every B_k must be connected to A_i. Since A_i is connected, $V \setminus B_j$ is connected. To conclude, we set $A = V \setminus B_j$ and $B = B_j$. $\square$

In the following, we let $T = (V, E)$ be an $\ell \times \ell$ torus and use the notation of Definition 24. For every $i \in [\ell]$, define the i^{th} *row* of T as $R_i \overset{\text{def}}{=} \{v_{i,j} \in V \mid j \in [\ell]\}$. For every $j \in [\ell]$, define the j^{th} *column* of T as $C_j \overset{\text{def}}{=} \{v_{i,j} \in V \mid i \in [\ell]\}$. Given a set $A \subseteq V$, let $R(A)$ be the set of rows R_i of T such that $A \cap R_i \neq \emptyset$, and let $C(A)$ be the set of columns C_j of T such that $A \cap C_j \neq \emptyset$.

Given a cut (A, B) of V we say that a row R_i is *mixed* if $R_i \subseteq R(A) \cap R(B)$, that is, if R_i includes vertices in A as well as vertices in B. Similarly, we say that a column C_j is *mixed* if $C_j \subseteq C(A) \cap C(B)$. Let r_{mix} be the number of mixed rows with respect to (A, B) and let c_{mix} be the number of mixed columns with respect to (A, B).

Observation 8.3.13. $|E(A, B)| \geq 2(r_{mix} + c_{mix})$.

Proof. Looking at the cycle of vertical edges connecting all the vertices in every mixed column, it is easy to see that every mixed column has at least two vertical edges in (A, B). Similarly, it can be shown that every mixed row has at least two horizontal edges in (A, B). $\square$

Observation 8.3.14.

1. *If* $|R(A)| < \ell$ *then* $c_{mix} = |C(A)|$.
2. *If* $|C(A)| < \ell$ *then* $r_{mix} = |R(A)|$.

The analogous claims also hold for B.

Proof. We give the proof of the first item. The proof of the second item is similar. Let R_i be a row of T that is not in $R(A)$. Then $v_{i,j} \in B$ for every $j \in [\ell]$. Hence, every column $C_j \in C(A)$ has a vertex in A as well as a vertex in B (namely, $v_{i,j}$), which proves the claim. $\square$

Observation 8.3.15.

1. *If $|R(A)| = \ell$ then $r_{mix} = |R(B)|$.*
2. *If $|C(A)| = \ell$ then $c_{mix} = |C(B)|$.*

Proof. We give the proof of the first item. The proof of the second item is similar. Suppose that $|R(A)| = \ell$. Then every row includes a vertex in A. Let $R_i \in R(B)$. Then R_i includes a vertex in A as well as a vertex in B, which completes the proof. □

We say that a set $A \subseteq V$ of vertices in T is *grid-bounded* if $|R(A)| < \ell$ and $|C(A)| < \ell$.

Lemma 8.3.16. *Let (A, B) be a cut of V where $|E(A, B)| < 2\ell$. Then at least one of A and B is grid-bounded.*

Proof. From Observation 8.3.13 we have that $r_{mix} + c_{mix} < \ell$. Note that $|R(A)| + |R(B)| - r_{mix} = \ell$ and $|C(A)| + |C(B)| - c_{mix} = \ell$. Hence $|R(A)| + |C(A)| + |R(B)| + |C(B)| \leq 2\ell + r_{mix} + c_{mix} < 3\ell$, and thus, at most two of the sets $R(A), C(A), R(B), C(B)$ are of size ℓ. Assuming that both A and B are not grid-bounded, we have $\max(|R(A)|, |C(A)|) = \ell$ and $\max(|R(B)|, |C(B)|) = \ell$. Therefore, we must have $|R(A)| = \ell$ and $|C(A)| < \ell$ or $|R(A)| < \ell$ and $|C(A)| = \ell$. The proofs of both cases are similar. We give the proof for the case where $|R(A)| = \ell$ and $|C(A)| < \ell$. From Observation 8.3.15 we have $|R(B)| = r_{mix} < \ell$, and thus, from Observation 8.3.14, $c_{mix} = |C(B)|$. Now, $|C(B)| = \ell$, as otherwise B is grid-bounded. We hence obtain $c_{mix} = \ell$, a contradiction. □

Observation 8.3.17. *If A is connected then there exists a row index $i^* \in [\ell]$ such that $R(A) = \{R_{i^*}, R_{i^* \oplus 1}, \ldots, R_{i \oplus (s-1)}\}$ where $s = |R(A)|$, and there exists a column index $j^* \in [\ell]$ such that $C(A) = \{C_{j^*}, C_{j^* \oplus 1}, \ldots, C_{j \oplus (t-1)}\}$ where $t = |R(B)|$. Hence, A is contained in a subgraph G of T which is an $|R(A)| \times |R(B)|$ grid. Renaming i^* as 1 and j^* to 1, we have that G is a grid with the vertex set $V_G = \{v_{i,j} \mid i \in [s], j \in [t]\}$.*

Proof. Let R_{i_1}, R_{i_2} be rows in $R(A)$. Hence, both R_{i_1} and R_{i_2} include at least one vertex in A. Since A is connected, there exists a path of vertices in A between R_{i_1} and R_{i_2}. Clearly, for every edge in the path, the endpoints are in the same row (in case of a horizontal edge) or in subsequent rows (in case of a vertical edge). We thus conclude that $R(A)$ is a set of successive rows in the torus. Similarly, $C(A)$ is a set of successive columns in the torus. □

Lemma 8.3.18. *Let T be an $\ell \times \ell$ torus, and let $\overrightarrow{T}$ be a non-Eulerian orientation of T. Let $U \subseteq V(T)$ denote the set of unbalanced vertices with respect to $\overrightarrow{T}$. Let $Q \subseteq E(T)$ be a set of edges forming a witness that $\overrightarrow{T}$ is not Eulerian, where $|Q| < \frac{1}{2}\ell$. Let q denote the minimal distance of an edge in Q to an unbalanced vertex, that is, $q \overset{\text{def}}{=} \min_{e \in Q,\ u \in U}\{\text{dist}(e, u)\}$. Then $|Q| \geq q$.*

Proof. By Lemma 8.3.12, we assume without loss of generality that Q contains more than half of the edges of an invalid cut (A, B), where both A and B are connected. Since $|Q| < \frac{1}{2}\ell$, we have $|E(A, B)| < \ell$, and hence, from Lemma 8.3.16, one of the sets A and B is grid-bounded. Assume, without loss of generality, that A is grid-bounded. Denote $s = |R(A)|$ and $t = |C(A)|$. Then $s, t < \ell$. Since A is connected, from Observation 8.3.17, A is contained in an $s \times t$ grid G.

Suppose that $|Q| < q$. Then $|E(A, B)| < 2q$, and from Observation 8.3.13 we have $r_{mix} + c_{mix} < q$. Let $e = (w_A, w_B)$ be an edge in $Q \cap E(A, B)$, where $w_A \in A$ and $w_B \in B$. Since A is invalid, there exists an unbalanced vertex $u \in A$. By the definition of q we have $\text{dist}(e, u) \geq q$ and hence $\text{dist}(u, w_A) \geq q$. Using the notation of Observation 8.3.17, we denote $u = v_{i_1,j_1}$ and $w_A = v_{i_2,j_2}$, where $i_1, i_2 \in [s]$ and $j_1, j_2 \in [t]$. Then clearly $|i_1 - i_2| + |j_1 - j_2| \geq q$. We thus have $s = |R(A)| \geq |i_1 - i_2| + 1$ and $t = |C(A)| \geq |j_1 - j_2| + 1$. Since A is grid-bounded, from Observation 8.3.14 we obtain $|C(A)| + |R(A)| = r_{mix} + c_{mix} \geq |i_1 - i_2| + 1 + |j_1 - j_2| + 1 > q$. Finally, Observation 8.3.13 gives that $|E(A, B)| > 2q$, a contradiction. □

Proof. of Theorem 8.3.2 Let $Q \subseteq E(T)$ be the fixed set of $\frac{1}{100}m^{1/4}$ edge queries that a deterministic non-adaptive algorithm makes. By Lemma 8.3.18, in order to form a witness that $\overrightarrow{T}$ is not Eulerian, one of the edges in Q must be within distance at most $|Q| = \frac{1}{100}m^{1/4} < \sqrt{\ell}$ from an unbalanced vertex in $\overrightarrow{T} \sim \mathcal{F}'_\ell$. But according to Lemma 8.3.11, the probability of the above is at most 1/5. We thus conclude that discovering a witness that $\overrightarrow{T} \sim \mathcal{F}'_\ell$ is not Eulerian with probability larger than 1/5 requires more than $\frac{1}{100}m^{1/4}$ nonadaptive queries. □

Part III

Probabilistically Checkable Proofs of Proximity

Chapter 9

Length-Soundness tradeoffs for 3-query PCPPs

In this chapter we study the tradeoff between the *length* of a probabilistically checkable proof of proximity (PCPP) and the maximal *soundness* that can be guaranteed by a 3-query verifier with oracle access to the proof. Our main observation is that a verifier limited to querying a short proof cannot obtain the same soundness as that obtained by a verifier querying a long proof. Moreover, we quantify the *soundness deficiency* as a function of the proof-length and show that any verifier obtaining the "best possible" soundness must query an exponentially long proof.

In terms of techniques, we focus on the special class of *inspective* verifiers that read at most two proof-bits per invocation. For such verifiers we prove *exponential* length-soundness tradeoffs that are later on used to imply our main results for the case of general (i.e., not necessarily inspective) verifiers. To prove the exponential tradeoff for inspective verifiers we show a surprising connection between PCPP proof length and property-testing query complexity. The connection is that any property that can be verified with proofs of length ℓ by inspective verifiers must be testable with query complexity $\approx \log \ell$.

9.1 Background and introduction

This chapter discusses the relationship between two basic parameters of *probabilistically checkable proofs of proximity* (PCPPs) – their *proof length* and *soundness.* PCPPs were simultaneously introduced in [BSGH$^+$04] and (under the name *assignment testers*) in [DR04] and a similar notion also appeared earlier in [Sze99]. The interest in PCPPs stems first and foremost from the role they play within the proof of the celebrated PCP Theorem of [AS98, ALM$^+$98]. All recent constructions of PCPs, starting with [BSGH$^+$04, DR04], use PCPPs to simplify the proof of the PCP theorem and improve certain aspects of it, most notably, to decrease the length of proofs as in [BSGH$^+$04, BSS05, Din07]. All previous proofs of the PCP theorem implicitly use PCPPs and can be augmented to yield them. (See, e.g., [BSGH$^+$04, Theorem 3.2] for a conversion of the original PCP system of [AS98, ALM$^+$98] into a PCPP). But PCPPs are also interesting beyond the scope of the PCP Theorem. They can be used to transform any error correcting code into a locally testable one and to construct "relaxed" locally decodable codes [BSGH$^+$04]. Additionally, as shown in [FF05, GR05], they have applications to questions in the theory of "tolerant" property testing that was introduced in [PRR06].

A *PCPP verifier*, (or, simply, verifier) for a property $P \subset \{0,1\}^n$ is a randomized, sublinear-time algorithm that distinguishes with high probability between inputs that belong to P and inputs that are far in relative Hamming distance from all members of P. In this respect a verifier is similar to a *property-tester* as defined in Section 3.2. However, in contrast to a tester, the verifier may query an auxiliary proof, called a *proof of proximity.* A PCPP system has four basic parameters of interest, described next – *length, query complexity, completeness* and a *soundness function.* The *proof length* is the length of the auxiliary proof that is queried by the verifier[1]. The *query complexity* is the maximal number of bits that can be read from *both* the input and the proof. The *completeness* parameter is the minimal probability with which inputs that belong to P are accepted when they are presented along with a "good" proof of proximity. Finally, the *soundness function* $s(\delta)$ is the minimal rejection probability of inputs that are δ-far (in relative Hamming distance) from P, where the minimum is taken over all such δ-far inputs and all possible proofs that may accompany them.[2] One additional property of a verifier is adaptiveness, but in this thesis we will only focus on non-adaptive verifiers. See Section 9.2 for a formal definition of PCPPs and further discussion of their parameters.

9.1.1 Informal description of the results

To describe our results, let us discuss the range of parameters we can expect from a verifier for a *linear* property over the binary alphabet, i.e., a property that is closed under addition modulo 2 (this amounts to saying P is a linear subspace of $\mathbb{F}_2^n$ where $\mathbb{F}_2$ denotes the two-element field). We look at nonadaptive 3-query verifiers with perfect (probability 1) completeness, thereby fixing two of the four basic parameters, and look at the tradeoff between proof length and soundness. We point out that all known constructions of PCPPs naturally yield nonadaptive 3-query verifiers with perfect completeness, so the results

[1]In PCP literature one often encounters *randomness complexity* as a means for bounding proof-length. The two parameters are closely related, i.e., proof-length $\approx 2^{\text{randomness}}$, and we stick to the former parameter.

[2]Often, in literature on PCPs, the term "soundness" refers to "soundness-error" which is defined to be the *maximal* acceptance probability of a "bad" input. The connection between soundness (used here) and soundness-error, denoted s_{error}, is given by $s = 1 - s_{error}$.

described next apply to all of them.

Suppose we are interested in minimizing the proof length. The results of [Din07, BSS05] give constructions with proofs of length at most $m \cdot \text{polylog } n$ where m is the minimal size of circuit deciding P (notice that the linearity of P implies $m = O(n^2)$). Regarding the soundness function, consider a random word that can be shown to have, with high probability, distance $\delta \approx \frac{1}{2}$ from P. The "short PCPP" construction mentioned above gives $s(\delta) > \varepsilon$ for some small and unspecified constant $\varepsilon > 0$ that depends only on δ and neither on P, nor on n.

Next, let us try to increase the soundness. We show in Theorem 9.2.2 that the soundness can be boosted to $s(\delta) \geq \delta$ and that this soundness is obtained by a *linear* verifier. A verifier is called linear if the set of answer-bits that cause it to accept forms a linear space. For $\mathbb{F}_2$ this amounts to saying the verifier accepts iff the sum (mod 2) of the queried bits is 0. For such verifiers, it can be shown that $s(\delta)$ is at most $\frac{1}{2}$ and thus the soundness of our construction is optimal. On the down side, the length of the proof used by this verifier is exponential in n (this soundness-optimal construction can be carried out over any finite field of prime size – see Theorem 9.2.2 for details).

To sum up the situation so far, we have constructions that are nearly optimal in length, but are deficient in soundness, and we have constructions that are optimal in soundness but are deficient in length. One could conjecture that a "super-PCPP" with short proofs *and* optimal soundness exists. Our first main result, stated in Theorem 9.2.3 and Corollary 9.2.4, rules this out. We show a tradeoff between proof length and soundness that essentially matches our soundness-optimal construction. In plain words, for some properties (discussed below) any PCPP verifier that queries a short proof of length ℓ must incur a *soundness deficiency*, and this deficiency increases as ℓ decreases (see Definition 31 for a formal definition of deficiency).

Our next main result, stated in Theorem 9.2.5 and Corollary 9.2.6, proves a tighter tradeoff similar to the one mentioned above for the case of $\mathbb{F}_p$-linear verifiers for $\mathbb{F}_p$-linear properties over a finite field of size p. Our results in this case are stronger even though the query complexity, when measured in bits, is greater than 3 (however, the bits are read from three "blocks", where each block encodes a field element).

So far we have not specified which properties cause this kind of tradeoff to arise, i.e., which properties are "hard to verify". These are in fact properties that are "hard to test". Informally, we say that $P \subset \{0,1\}^n$ is "hard to test" if any property-tester for P (as defined in [GGR98]) that rejects (say) $\frac{1}{3}$-far inputs with probability greater than 1/100, requires query complexity $q \gg 3$. Our main theorems (Theorems 9.2.3 and 9.2.5) show an *exponential* tradeoff between the property-testing query complexity q and the minimal length of a 3-query verifier with large soundness (e.g., achieving a soundness function $s(\delta) \geq \delta - 1/100$). In a certain sense we show that any property that is hard to test is also hard to verify.

9.1.2 Context and motivation

We are motivated by the attempt to understand the limitations of PCP constructions. One interesting open question related to our research is that of obtaining 3-query PCPs with quasilinear length, completeness $1 - \varepsilon$ and soundness $\frac{1}{2} - \varepsilon$ for any language in **NP**. For the sake of reference, we informally call such a construction a "super-PCP". The celebrated result of [Hås97] obtains three out of four of these parameters (the proof length there is a large polynomial). Numerous other works such as [GLST98, HK01, ST00, EH05, KS06, ST06] investigate optimal, or nearly optimal, tradeoffs between the three parameters of query complexity, completeness and soundness, while settling for polynomial length proofs. A different line of research focused on optimizing the tradeoff between proof length and query complexity [PS94, HS01, GS02, BSSVW03, BSGH$^+$04, BSS05, Din07, MR06, MR07], and all of these constructions obtain perfect completeness. Several of these works, most notably [HS01, GS02, MR06, MR07], also strive to optimize the fourth parameter, soundness, but have stopped short of constructing a "super-PCP".

Our results show why a certain natural class of PCP constructions will not be suitable for reaching our goal. All constructions of "short" PCPs (i.e., with proof length $n^{1+o(1)}$ for **NP** instances of size n) start by encoding a witness for an **NP**-instance by some good error correcting code, usually based on univariate or multivariate polynomials. These codes are inherently "hard to test" because they have relatively high degree and are converted into locally testable codes by composition with a PCPP. Our results show that no matter how one tries to compose such codes with a PCPP, the resulting soundness will not come close to $\frac{1}{2}$ unless the proof is exponentially long. If a different error correcting code will someday replace the aforementioned codes as a starting point for PCP constructions, our results imply this code should be locally testable, at least if we hope to use it to obtain a "super-PCP" construction.

This work can also be placed within the larger context of the study of limitations of PCPs and objects related to them. There are preciously few results that give nontrivial tradeoffs between the basic parameters of a PCP system. One notable example presented in [Zwi98] shows that the soundness of a 3-query PCP verifier with perfect-completeness cannot exceed 3/8 unless $\mathbf{NP} \subseteq \mathbf{BPP}$. A larger number of works try to understand the limitations of PCP systems by either showing limitations of specific techniques used in PCP constructions, or proving limitations on computational and combinatorial objects that are closely related to PCPs. Along the first line of research one can mention [FK95] that shows limitations on derandomizing the parallel repetition method of [Raz98], and [Bog05] that shows upper bounds on the soundness that can be obtained from the gap amplification technique of [Din07]. The second line of research includes the study of the limits of various basic parameters of locally decodable codes [KT00, KdW03], locally testable codes [BSGS03], unique games [Kho02, Tre05, CMM06], and a large number of results regarding the limits of property testing (see the survey [Fis04] for further information). Our work relates to both of these lines of research because PCPPs are computational objects that are closely related to PCPs and constitute the method of choice for constructing them. We also hope that the research initiated here will contribute to a better understanding of the inherent limits of the PCP theorem.

Last but not least, the actual soundness parameter one obtains from a small query PCPP (and the PCPs and LTCs resulting from it) may someday in the future tell whether such objects can be put to practical use in proof checking (as in [BFLS91]), communication and cryptography (as in [Kil92, Mic00]). Therefore, the study of tradeoffs between soundness and proof length is of inherent importance.

9.1.3 Proof techniques

Inspective PCPPs Consider a 3-query verifier that rejects inputs that are δ-far from P with probability $\approx \delta$. At first sight it may seem that reaching soundness $s(\delta) \geq \delta$ is impossible because such high soundness forces the verifier to make at least one out of three queries to the input, leaving only two queries for "checking" the proof. Indeed, a verifier that seldom queries the input can be easily fooled to accept with high probability a "legitimate" proof accompanying an input that is δ-far from P. The need to look at the input naturally leads us to define an *inspective* verifier as one that *inspects* the input on every invocation. Formally, an inspective verifier is one that makes at most two queries to the proof; all other queries are to the input.[3] Our main *positive* result, Theorem 9.2.2, says that every $\mathbb{F}_p$-linear property over a prime field of size p has a 3-query $\mathbb{F}_p$-linear inspective verifier with soundness function $s(\delta) \geq \delta$ and proof length at most $p^{\dim(P)}$. "Good" proofs for inputs $w \in P$ turn out to be certain "folded" Hadamard codewords, and we analyze soundness using the Fourier analytic approach to linearity testing that was introduced in [BCH$^+$95] (see Section 9.3 for more details). The soundness obtained by the verifier of Theorem 9.2.2 is the benchmark against which we measure all other 3-query verifiers.

Exponential tradeoffs between soundness and proof length for inspective PCPPs All our results about the soundness deficiency of short PCPPs are based on exponential tradeoffs between soundness and proof length for *inspective* PCPPs. Since these results are similar in spirit let us informally describe how we obtain them in the simplest setting – that of $\mathbb{F}_2$-linear verifiers.

[3]Alternatively, an inspective verifier could be defined as one that makes at least one query to the input. For query complexity 3 the two definitions coincide, but for larger query complexity there is a big difference. In particular, our main technical lower bound can be extended to any q-query inspective PCPP, as long as we limit the number of proof-queries to be at most two.

Roughly speaking, we show that if the linear property $P \subset \mathbb{F}_2^n$ has a linear inspective verifier that makes q queries[4] to a proof of length ℓ and achieves soundness function $s(\delta)$, then for every $\varepsilon > 0$ the property P has a *tester*, i.e., a verifier that queries only input bits (not requiring proof), with query complexity $O((q \log \ell)/\varepsilon)$ and soundness function $s(\delta) - \varepsilon$. The contrapositive formulation for $\delta \approx 1/2$ and $\varepsilon = 0.01$ gives the following statement. Suppose that P is "hard to test", i.e., any tester for P with large soundness requires large query complexity. Then any inspective linear verifier for P with small query complexity must use proofs of exponential length. Examples of "hard to test" properties include most random Low Density Parity Check (LDPC) codes as defined in [Gal62] and linear spaces P for which the dual space, denoted P^*, has no elements of small support (in coding terminology, P is a linear code with large dual distance). As mentioned earlier, most error correcting codes actually used as the starting point for constructing PCPs, PCPPs and LTCs fall within this latter class.

From inspective to general PCPP tradeoffs Given the exponential tradeoff between soundness and proof length for inspective verifiers, the proof of our main results (stated in Section 9.2) goes along the following lines. A verifier is forced to choose between two "bad" options. Either the probability that it reads only proof-bits is large. In this case we fool it by presenting a legitimate proof for some word and make use of the fact that the verifier seldom looks at the input (that is δ-far from P). Otherwise, the probability that the verifier makes an inspective query is large. In this case we use the tradeoff for the inspective case to fool verifiers that use short proofs. In either of these two cases we manage to fool the verifier into accepting words that are δ-far from P with probability approaching $1 - \delta/2$, i.e., the *soundness-deficiency* of short-proof verifiers when compared to the exponential length verifier of Theorem 9.2.2 is close to $\delta/2$. To complete the overview of our proof techniques we describe next how we obtain exponential length-soundness tradeoffs for inspective verifiers.

Proving Tradeoff Theorems for inspective verifiers Informally, we convert a q-query *inspective verifier* for P that uses a proof of length ℓ and obtains soundness function s, into a *tester* (not requiring proof) with query complexity $O(q \log \ell)/\varepsilon$ and soundness $s - \varepsilon$. We first notice that an inspective verifier gives rise to a natural induced labeled multigraph. The vertices of this graph are indices of proof bits, so the number of vertices equals the length of the proof. For simplicity assume that each query-tuple reads exactly two bits of the proof. Thus, every query-tuple defines an edge whose endpoints are the proof bits read, and we label this edge by the set of indices of input bits read when making the query (note that the resulting graph may have multiple edges between two vertices

[4]Our tradeoffs for inspective PCPPs hold for query complexity larger than 3, even though for the proof of our three main theorems query complexity 3 suffices.

and these edges may have different labels). The induced graph is actually a representation of the verifier in the sense that a single invocation of the verifier corresponds to picking a random edge of the graph and making the set of queries given by the names of the end-vertices and the edge-label. More to the point, the labeled graph also constitutes a "partially-defined" *constraint graph*, meaning that if all input bits are read then the resulting set of constraints (over proof bits) forms a constraint satisfaction problem with two variables per constraint.

We apply a *decomposition lemma* (Lemma 9.5.3) due to [LR99] to the constraint graph and remove some of its edges. The decomposition lemma guarantees that if the graph was small to start with (i.e., the proof was short), then after removing a tiny fraction of edges we are left with disconnected components of small radius[5]. The "decomposed" graph corresponds to a new linear inspective verifier whose soundness has not decreased significantly because it has pretty much the same possible queries as the original verifier. Our analysis is completed (in Lemma 9.5.2) by showing that inspective PCPPs whose induced graph has radius R can be converted without loss in soundness into (proofless) testers with query complexity $O(R)$. Summing up, if the proof is short to start with, then its decomposed graph has small radius, also P has a tester with small query complexity and good soundness.

The decomposition lemma mentioned above was previously used in a closely related context in [Tre05] to provide algorithms for approximating unique games. We use it for similar purposes, namely for analyzing constraint graphs, but our setting differs from that of [Tre05] in three important aspects. First, in our setting the constraints that label edges of the constraint graph are not given to the verifier and only the structure of the graph itself is known in advance. This difference also explains why the techniques relying on linear and semidefinite programming that were used in [Kho02, Tre05, CMM06, GT06] do not seem appropriate for our setting. The second difference is that for our constraint graphs that are induced by 3-query verifiers, perfect completeness can be assumed. In the context of the unique games conjecture, assuming perfect completeness makes the problem trivial to solve. Finally, we use the decomposition lemma to construct a tester for the constraint graph rather than just decide whether the constraint graph is close to be satisfiable.

We end our discussion of the proof techniques by pointing out Lemma 9.4.1, a generalization of the decomposition lemma to the case of non-unique constraint graphs. This lemma, which is required for obtaining our main result for general verifiers (Theorem 9.2.3), may be of independent interest. It says that any 2-CSP with ℓ constraints over the binary alphabet that is ε-far from being satisfiable, must contain a contradiction with $O(\log \ell/\varepsilon)$

[5]The radius of a connected graph is the minimum maximal distance between any vertex and any other vertex, i.e, $\mathrm{rad}(G) = \min_v \max_u d(u, v)$, where $d(u, v)$ denotes the distance between the vertices u and v.

constraints.

This chapter is organized as follows. In the next section we give some specific definitions and statements of our main results. Section 9.3 constructs 3-query verifiers with optimal soundness and exponentially long proofs. The last sections 9.4 and 9.5 prove our main tradeoffs for general and linear verifiers respectively.

9.2 Specific definitions and statement of the main results

We start by recalling the basic definitions and parameters of a PCPP system. Then, in Subsection 9.2.2 we introduce and define the *best soundness* and the *soundness deficiency* which are the quantities that we use to measure the tradeoff between proof length and soundness. In Subsection 9.2.3 we summarize our main results for both general PCPPs over the binary alphabet, and linear PCPPs over finite fields. Finally, in Subsection 9.2.4 we formally define *inspective* PCPPs and state the tradeoffs for these PCPPs.

9.2.1 Probabilistically checkable proofs of proximity (PCPPs)

Let Σ be a finite alphabet. A set $P \subseteq \Sigma^n$ is called a *property* of length n over Σ. We are interested in deciding the promise problem whose set of YES instances is P and whose set of NO instances is $\mathrm{NO}_{\delta_0} = \{w \in \Sigma^n \mid \delta(w, P) > \delta_0\}$, where $\delta(\cdot)$ denotes fractional Hamming distance and δ_0 is called the *proximity parameter*. In the context of property testing, decision should be made after making a small number of queries into the input *word* $w \in \Sigma^n$ and the decision should be correct with high probability. In the context of *proximity testing* we try to decide the very same promise problem but the difference is that we allow oracle access to an additional *proof of proximity* $\pi \in \Sigma^\ell$ of length ℓ, and restrict the total number of queries that can be made to both w and π. In the following, we call the usual property tester simply a *tester*, and whenever we allow oracle access to a proof we call it a *verifier*. The formal definition follows (see [BSGH$^+$04] for more information on PCPPs).

To simplify exposition we view w and π as functions, from $[n] = \{1, \ldots, n\}$ and from $[n+1, n+\ell] = \{n+1, \ldots, n+\ell\}$ respectively to Σ, and define the *word-proof pair* as the function $(w \sqcup \pi) : [n+\ell] \to \Sigma$ that is the concatenation of w and π. We call $(w \sqcup \pi)[i]$ a *word-symbol* whenever $i \leq n$ and a *proof symbol* when $i \in \{n+1, \ldots, n+\ell\}$. For a set of indices $I \subseteq [n+\ell]$ let $(w \sqcup \pi)|_I : I \to \Sigma$ denote the restriction of $w \sqcup \pi$ to I.

Definition 27 (Verifier, Tester). *A* query *of size q into a word of length n and a proof of length ℓ is a pair $Q = (I, C)$ where $I \subseteq [n+\ell], |I| \leq q$ denotes the query's* index-set *and $C : \Sigma^I \to \{\mathsf{accept}, \mathsf{reject}\}$ is the query's* constraint. *Given a word w and a proof π, let $Q(w \sqcup \pi) = C((w \sqcup \pi)|_I)$. A (q, n, ℓ)-*verifier *for a property of length n is a pair $\mathcal{V} = \langle \mathcal{Q}, D \rangle$ where*

- *$\mathcal{Q}$ is a finite set of queries of size at most q into a word of length n and proof of length ℓ.*
- *D is a distribution over $\mathcal{Q}$. We use $Q \sim_D \mathcal{Q}$ to denote that Q is sampled from $\mathcal{Q}$ according to distribution D.*

*A q-*tester *is a $(q, n, 0)$-verifier, i.e., a verifier that queries only the input, and a (q, n, ∞)-*verifier *is a (q, n, ℓ)-verifier for some value of ℓ (this notation will be used as a shorthand to verifiers querying proofs of unbounded length).*

Notice that as opposed to other parts of this thesis, in this chapter the testers have restricted query complexity, and we measure their quality according to their error probability (soundness).

Often we will restrict our attention to a subclass of verifiers that use special kinds of constraints. In particular, we will be interested in *linear* verifiers, defined next.

Definition 28 (Linear verifiers)**.** *A query is called $\mathbb{F}$-linear if $\Sigma = \mathbb{F}$ is a finite field and the set of assignments accepted by the query-constraint forms an $\mathbb{F}$-linear space.*

A verifier is called linear *if all its queries are linear. Let $\mathbb{F}$-**linV** denote the set of $\mathbb{F}$-linear verifiers.*

Informally, if a (q, ℓ)-verifier solves the promise problem associated with P "with high probability" then we say that P "has a PCPP" with query complexity q and length ℓ. The *completeness* and *soundness* parameters quantify the success probability of the verifier. The formal definition follows.

Definition 29 (PCPP, Testability)**.** *A property $P \subset \Sigma^n$ is said to have a PCPP of* length *ℓ,* query complexity *q,* completeness *parameter c and* soundness function *$s : (0, 1] \to [0, 1]$ if there exists a (q, n, ℓ)-verifier for the property satisfying the following requirements.*

- **Completeness:** *For all $w \in P$,*

$$\max_{\pi \in \Sigma^\ell} \Pr_{Q \sim_D \mathcal{Q}}[Q(w \sqcup \pi) = \mathsf{accept}] \geq c.$$

  If $c = 1$, we say the verifier has perfect *completeness.*

- **Soundness:** *For all $w \in \Sigma^n \setminus P$,*

$$\min_{\pi \in \Sigma^\ell} \Pr_{Q \sim_D \mathcal{Q}}[Q(w \sqcup \pi) = \mathsf{reject}] \geq s(\delta(w, P)),$$

  where $\delta(w, P)$ denotes the minimal fractional Hamming distance between w and an element of P.

If P has a PCPP of length 0, query complexity q, completeness parameter c and soundness function s, we say that P is q-testable *with completeness c and soundness s.*

A verifier is said to be *adaptive* if it's query indices depend on answers given to previous queries. The verifier defined above is *nonadaptive.* All results in this chapter refer to nonadaptive verifiers with perfect completeness. We point out that all known PCPP constructions use nonadaptive verifiers and achieve perfect completeness, so our deficiency bounds apply to all of them (see [BHLM08] for further discussion).

9.2.2 Soundness deficiency

We study the tradeoff between *proof length* and *soundness.* Our aim is to show that short PCPPs cannot attain the same soundness as long ones. To quantify this tradeoff we start by defining the *best soundness* that can be obtained by a class of verifiers with restricted proof length.

Definition 30 (Best Soundness)**.** *Let $P \subseteq \Sigma^n$ be a property. For integers q, ℓ and $\delta \in [0,1]$, define the* best soundness *$\mathcal{S}^P(q,\ell,\delta)$ to be the maximum – taken over all (q,n,ℓ)-verifiers $\mathcal{V}$ – of the soundness of $\mathcal{V}$ with respect to inputs that are δ-far from P. Formally,*

$$\mathcal{S}^P(q,\ell,\delta) = \max_{(q,n,\ell)\text{-}verifiers} \; \min_{w\sqcup\pi\in\Sigma^{n+\ell},\ \delta(w,P)=\delta} \; \Pr_{Q\sim_D\mathcal{Q}}[Q(w\sqcup\pi) = \mathsf{reject}].$$

The best tester soundness *is $\mathcal{S}^P(q,0,\delta)$.*

The best soundness with respect to a class of verifiers $\mathbf{V}$, denoted $\mathcal{S}^P_{\mathbf{V}}(q,\ell,\delta)$, is defined by taking the maximum above over all (q,n,ℓ)-verifiers in $\mathbf{V}$. Notice that $\mathcal{S}^P_{\mathbf{V}}(q,\ell,\delta) \leq \mathcal{S}^P(q,\ell,\delta)$.

The *soundness-deficiency*, defined next, is the reduction from the best soundness incurred by 3-query verifiers limited to using short proofs.[6] As customary in computational complexity, we measure the asymptotic deficiency over a family of properties of increasing length. In the remark following the definition, we further explain the need for complexity assumptions.

Definition 31 (Soundness deficiency)**.** *For a family of properties $\mathcal{P} = \{P \subseteq \Sigma^n \mid n \in \mathbb{Z}^+\}$, a class of verifiers $\mathbf{V}$ and a function measuring proof length $\ell : \mathbb{Z}^+ \to \mathbb{Z}^+$, let the* soundness-deficiency *be the function measuring the decrease in soundness due to limited proof length. Formally, it is the function from $(0,1]$ to $[0,1]$ defined by*

$$\text{s-Def.}_{\mathbf{V}}[\mathcal{P},\ell](\delta) = \liminf_{n\to\infty} \mathcal{S}^{P_n}_{\mathbf{V}}(3,\infty,\delta) - \mathcal{S}^{P_n}_{\mathbf{V}}(3,\ell(n),\delta)\,.$$

[6]The definition could be naturally generalized to query complexity greater than 3. However, since all our results are limited to $q = 3$ we omit the query complexity parameter to simplify notation.

For a complexity class $\mathcal{C}$ and a family of complexity functions $\mathcal{L}$, let s-Def.$_{\mathbf{V}}[\mathcal{C},\mathcal{L}](\delta)$ *be the maximal soundness deficiency function taken over all $\mathcal{P} \subseteq \mathcal{C}$ and $\ell \in \mathcal{L}$. Let in addition* max-s-Def.$_{\mathbf{V}}[\mathcal{C},\mathcal{L}] = \max_{\delta\in(0,1]}$ s-Def.$_{\mathbf{V}}[\mathcal{C},\mathcal{L}](\delta)$ *be the maximal value that this function obtains over all $\delta \in (0,1]$. As before, whenever there is no restriction to a specific class of verifiers, the subscript $\mathbf{V}$ is omitted.*

Remark 9.2.1 (Complexity restrictions)**.** *If no restriction is placed on the complexity of $\mathcal{P}$, then one may end up with trivial and uninteresting results. For instance, if $P_n \subset \{0,1\}^n$ is random, then with high probability any nondeterministic circuit deciding the promise problem associated with P_n requires size $2^{\Omega(n/\log n)}$. This implies that there are no constant query PCPPs with positive soundness and proof length $2^{o(n/\log n)}$. Thus, to get meaningful results, we focus on properties $\mathcal{P} \in \mathbf{P}/poly$ for which the existence of polynomial-length PCPPs is guaranteed.*

9.2.3 Summary of results

In this section, we summarize our main results bounding the maximum soundness deficiency for two different classes of verifiers – general verifiers and linear verifiers. Deficiency bounds are obtained by bounding from below the soundness of inspective verifiers that have access to long proofs and then bounding from above the soundness obtained by verifiers limited to short proofs. The next theorem shows the first bound, namely that large soundness is obtainable if no restriction is placed on proof length. Its proof is based on the Fourier analytic approach introduced in [BCH+95] and appears in Section 9.3.

Theorem 9.2.2 (Best soundness with unbounded proof length)**.** *Let $\mathbb{F}_p$ be a prime field. Every $\mathbb{F}_p$-linear property $P \subseteq \mathbb{F}_p^n$ has a 3-query $\mathbb{F}_p$-linear verifier using a proof of length at most $|\mathbb{F}|^{\dim(P)} \leq |\mathbb{F}|^n$ that achieves a soundness function $s(\delta) \geq \delta$. Formally,*

$$\mathcal{S}^P_{\mathbf{linV}}\left(3, |\mathbb{F}_p|^{\dim(P)}, \delta\right) \geq \delta.$$

Deficiency of short PCPPs

Our first main theorem says that for some properties, proofs of sub-exponential length incur constant soundness-deficiency. This deficiency can be reduced, but only at the expense of using exponentially long proofs.

Theorem 9.2.3 (Main)**.** *Let $\alpha \in (0,1)$ be a positive constant and let $\mathcal{P} \triangleq \{P_n \subseteq \{0,1\}^n : n \in \mathbb{Z}^+\}$ be a family of binary linear properties (codes) with dual distance[7] at least αn. The properties in $\mathcal{P}$ have no sub-exponential PCPP's achieving soundness larger than $1/3$.*

[7]The dual distance of a linear property P is defined to be the minimal support-size of a nonzero vector in the space dual to P.

Namely, for every $\varepsilon > 0$ there are $\beta > 0$ and $n_0 \in \mathbb{N}$ such that for any property $P_n \in \mathcal{P}$, $n > n_0$ the following is satisfied for all $\delta \in [0,1]$:

$$\mathcal{S}^{P_n}\left(3, 2^{\beta n}, \delta\right) \leq \frac{1}{3} + \varepsilon.$$

We show in Theorem 9.2.2 that every (in particular) binary linear property $P \subseteq \{0,1\}^n$ of dimension $k \leq n$ has a $(3, 2^k)$-verifier with soundness function $s(\delta) \geq \delta$. This implies constant deficiency for short PCPPs over the binary alphabet as formalized in the following corollary.

Corollary 9.2.4 (Soundness deficiency)**.** *Let* **SUBEXP** *denote the set of sub-exponential functions, i.e., functions satisfying $f(n) = 2^{o(n)}$. There exists a family $\mathcal{P}$ of linear properties over the binary alphabet such that*

$$\text{s-Def.}[\mathcal{P}, \mathbf{SUBEXP}](\delta) \geq \delta - \frac{1}{3}.$$

Consequently, since there are words that are roughly $\frac{1}{2}$-far from $\mathcal{P}$, the maximal deficiency with sub-exponential proofs is at least $\frac{1}{6}$, i.e.,

$$\text{max-s-Def.}[\mathbf{P}/\mathit{poly}, \mathbf{SUBEXP}] \geq \frac{1}{6}.$$

Deficiency of short Linear PCPPs

Our next main theorem presents stronger deficiency bounds for linear PCPPs and states the following intuitively appealing implication: Let p be a prime. Every $\mathbb{F}_p$-linear property that is "untestable" – in the sense that testers with small query complexity for it have low soundness – is also "unverifiable", i.e., 3-query $\mathbb{F}_p$-linear verifiers with short proofs must incur a large loss in soundness. Limiting our attention to linear verifiers seems natural in light of the fact that all current PCPP constructions produce linear verifiers for linear properties.

Theorem 9.2.5 (Main, linear case)**.** *Let $P \subseteq \mathbb{F}^n$ be a $\mathbb{F}$-linear property. Let $s[\ell](\delta)$ denote the best soundness of a $(3,\ell)$-linear verifier for P, i.e., $s[\ell](\delta) = \mathcal{S}^P_{\mathbf{linV}}(3, \ell, \delta)$. Let $t[q](\delta)$ denote the best soundness of a q-tester for P, i.e., $t[q](\delta) = \mathcal{S}^P(q, 0, \delta)$. Then*

$$s[\ell](\delta) \leq \min_{0 < \varepsilon \leq 1} \left\{ t\left[\frac{36 \log \ell}{\varepsilon}\right](\delta) + \frac{1}{2} \cdot \left(1 - \frac{1}{|\mathbb{F}|} + \varepsilon\right)\right\}.$$

Using Theorem 9.2.2 again for arbitrary prime p we get the following bound on the deficiency of linear verifiers.

Corollary 9.2.6 (Soundness deficiency, linear case)**.** *Let* **SUBEXP** *denote the set of subexponential functions, i.e., functions satisfying $f(n) = 2^{o(n)}$. For every prime field $\mathbb{F}_p$ there exists a family of $\mathbb{F}_p$-linear properties $\mathcal{P}$ such that*

$$\text{s-Def.}_{\mathbb{F}_p-\mathbf{linV}}[\mathcal{P}, \mathbf{SUBEXP}](\delta) \geq \delta - \frac{1}{2} \cdot \left(1 - \frac{1}{p}\right).$$

Consequently, the maximal deficiency of linear verifiers with sub-exponential proofs is at least $\frac{1}{2} \cdot (1 - 1/p)$. In other words,

$$\text{max-s-Def.}_{\mathbb{F}_p-\mathbf{linV}}[\mathbb{F}_p - \textit{linear}, \mathbf{SUBEXP}] \geq \frac{1}{2} \cdot \left(1 - \frac{1}{p}\right).$$

We point out that even if we restrict our attention to families of linear properties with constant dual distance, the soundness deficiency can be very large. This last point is explained in detail in the proof of Corollary 9.2.6.

9.2.4 Inspective PCPPs

The deficiency bounds stated above follow from much stronger bounds on the soundness achieved by a special family of *inspective* verifiers, defined next. Informally, inspective verifiers are called so because every 3-query they make *inspects* the word w in at least one location.

Definition 32 (Inspective PCPP)**.** *A query $Q = (I, C)$ is called* inspective *if its index-set involves at most two symbols of the proof, i.e., $|I \cap [n + 1, n + \ell]| \leq 2$. We refer to the above quantity as the* inspective size *(i-size) of the query Q.*

A verifier $\mathcal{V} = \langle \mathcal{Q}, D \rangle$ is said to be inspective *if all its queries are inspective. We denote by* $\mathbf{V_i}$ *be the set of inspective verifiers, and by* $\mathbf{linV_i}$ *the set of inspective linear verifiers.*

A property $P \subset \Sigma^n$ is said to have an inspective PCPP of length ℓ, query complexity q and soundness function $s : (0, 1] \to [0, 1]$ *if there exists a (q, n, ℓ)-inspective verifier with soundness function s. Inspective linear PCPPs are similarly defined.*

Remark 9.2.7. *We note that the linear verifier mentioned in Theorem 9.2.2 is in fact an inspective verifier that makes inspective queries of size exactly two. Thus, $\mathcal{S}^{P}_{\mathbf{linV_i}}\left(3, |\mathbb{F}_p|^{\dim(P)}, \delta\right) \geq \delta$.*

The main technical components in the proofs of Theorems 9.2.3 and 9.2.5 are the following respective upper bounds on the soundness of inspective verifiers limited to querying only short proofs. The proof of these theorems rely on defining a natural *inspective graph* (Definition 36) and applying a decomposition lemma to it. In the case of general PCPPs over the binary alphabet we use Lemma 9.4.1, and in the remaining two cases we apply Lemma 9.5.3 which is very similar to the original decomposition lemma of [LR99].

Definition 33 (d-Universal Properties). *A property $P \subseteq \Sigma^n$ is d-*universal *if for all subsets $I \subset [n], |I| \leq d$, the restriction of P to I equals Σ^I, i.e. $\{w|_I \mid w \in P\} = \Sigma^I$. Observe that any linear property P with dual distance d is also d-universal.*

Theorem 9.2.8 (Best soundness with inspective verifiers). *Let $P \subseteq \{0,1\}^n$ be a d-universal property, and let $q \in \mathbb{Z}^+$. Let s_{i} denote the best soundness of a (q,ℓ)-inspective verifier for P, i.e., $s_{\mathrm{i}}(\delta) = \mathcal{S}^P_{\mathbf{V_i}}(q,\ell,\delta)$. Then for every $\delta \in [0,1]$,*

$$s_{\mathrm{i}}(\delta) \leq \min_{\varepsilon>0} \left\{ \frac{4\log(\varepsilon^{-2}(n+\ell))}{\frac{d}{q-1}-2} + \varepsilon \right\}.$$

Theorem 9.2.9 (Best soundness with inspective linear verifiers). *Let $P \subseteq \mathbb{F}^n$ be an $\mathbb{F}$-linear property. Let $s_{\mathrm{i}}(\delta)$ denote the best soundness of a $(3,\ell)$-linear inspective verifier for P, i.e., $s_{\mathrm{i}}(\delta) = \mathcal{S}^P_{\mathbf{linV_i}}(3,\ell,\delta)$. Let $t[q](\delta)$ denote the best soundness of a q-tester for P, i.e., $t[q](\delta) = \mathcal{S}^P(q,0,\delta)$. Then*

$$s_{\mathrm{i}}(\delta) \leq \min_{\varepsilon>0} \left\{ t\left[\frac{36\log\ell}{\varepsilon}\right](\delta) + \varepsilon \right\}.$$

9.3 Long PCPPs with best possible soundness

In this section, we will prove that any $\mathbb{F}_p$-linear property $P \subseteq \mathbb{F}_p^n$ over a prime field $\mathbb{F}_p$ has a 3-query linear inspective PCPP of length at most $p^{\dim(P)}$. Furthermore, the soundness of this verifier on words that are δ-far from P satisfies $s(\delta) \geq \delta$, thereby proving Theorem 9.2.2. We point out that if P is "nontrivial", meaning there is no $i \in [n]$ such that $w_i = 0$ for all $w \in P$, then the soundness of linear verifiers can be shown to be bounded from above by $1-1/p$. This shows that for δ approaching $1-1/p$ the term "best possible" aptly describes the soundness function of our verifier.

9.3.1 Fourier transform – preliminaries

We interpret $\mathbb{Z}_p$ as the multiplicative group of pth complex roots of unity. Let $\omega \triangleq e^{\frac{2\pi i}{p}}$, and let $\mu_p = \{\omega^0, \omega^1, \ldots, \omega^{p-1}\}$ be the pth complex roots of unity. For every $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_p^n$ we define the function $\chi_\alpha : \mathbb{Z}_p^n \to \mathbb{C}$ as

$$\chi_\alpha(x_1, \ldots, x_n) = \omega^{(x\cdot\alpha)} = \omega^{\sum_i x_i\alpha_i}$$

For two functions $f : \mathbb{Z}_2^n \to \mathbb{C}$ and $g : \mathbb{Z}_p^n \to \mathbb{C}$, we define their *inner product* as

$$\langle f, g \rangle \triangleq \frac{1}{p^n} \sum_{x \in \mathbb{Z}_p^n} f(x) \cdot \overline{g(x)} = \mathbb{E}_{x \in \mathbb{Z}_p^n} \left[f(x) \cdot \overline{g(x)} \right]$$

It is easy to verify that the functions $\chi_\alpha : \mathbb{Z}_p^n \to \mathbb{C}$ are *orthonormal* with respect to this inner product. Namely, that for every $\alpha \in \mathbb{Z}_p^n$,

$$\langle \chi_\alpha, \chi_\alpha \rangle = 1$$

and for every $\alpha, \beta \in \mathbb{Z}_p^n, \alpha \neq \beta$,

$$\langle \chi_\alpha, \chi_\beta \rangle = 0$$

Therefore the functions $\{\chi_\alpha\}_{\alpha \in \mathbb{Z}_p^n}$ form a *basis* for the space of functions $f : \mathbb{Z}_p^n \to \mathbb{C}$ (the dimension of which is exactly p^n). Hence, every function $f : \mathbb{Z}_p^n \to \mathbb{C}$ can be written as a linear combination of the elements of this basis

$$f(x) = \sum_\alpha \hat{f}_\alpha \cdot \chi_\alpha(x)$$

where the coefficients $\hat{f}_\alpha$ (called the *Fourier coefficients* of f) are defined as follows:

$$\hat{f}_\alpha = \langle f, \chi_\alpha \rangle$$

We have the following equality (*Parseval's identity*).

$$\sum_{\alpha \in \mathbb{Z}_p^n} |\hat{f}_\alpha|^2 = \langle f, f \rangle = \mathbb{E}_{x \in \mathbb{Z}_p^n} \left[|f(x)|^2 \right].$$

In particular, if $f : \mathbb{Z}_p^n \to \mu_p$, then $\sum_{\alpha \in \mathbb{Z}_p^n} |\hat{f}_\alpha|^2 = 1$ and for all α, $|\hat{f}_\alpha| \leq 1$.

We also have the following well known lemma.

Lemma 9.3.1. *Let $\eta \in \mu_p$ be a p^{th} root of unity. Then the sum $\sum_{i \in [p] \setminus \{0\}} \eta^i$ equals $p - 1$ if $\eta = 1$, and equals -1 for any $\eta \neq 1$.*

9.3.2 Proof of Theorem 9.2.2

Let $P \subseteq \mathbb{Z}_p^n$ be a $\mathbb{Z}_p$-linear space of dimension k. Fix $G \in \mathbb{Z}_p^{n \times k}$ to be a matrix such that P equals the span of the columns of G:

$$P = \{Gx : x \in \mathbb{Z}_p^k\}.$$

Let $g_i \in \mathbb{Z}_p^k$ denote the i^{th} row of G. Thus, if $w = Gx$, then we have that $w_i = (g_i \cdot x)$ for all i. In the terminology of error correcting codes G is a *generating matrix* for the $[n,k]_p$-code P, and so we refer to the elements $w \in P$ as "codewords".

For every $x \in \mathbb{Z}_p^k$ we denote by $H_x : \mathbb{Z}_p^k \to \mathbb{C}$ the *Hadamard* encoding of x, which is defined as $H_x(y) = \omega^{(x \cdot y)} = \omega^{\sum_i x_i y_i}$. The function H_x can be explicitly written as a vector of values (of the exponents) in $\mathbb{Z}_p^{p^k}$. However, the following *folded* representation of H_x will be simpler to analyze. We partition the set $\mathbb{Z}_p^k \setminus \{0\}$ into disjoint classes of the form $\Big\{ j \cdot y : j \in \{1, \dots, p-1\} \Big\}$, each of size $p-1$. Then for each of these classes we choose one of its elements as a representative, and keep the values of H_x only for these representative elements. Now we can extract the value of $H_x(y)$ for every $y \in \mathbb{Z}_p^k$ as follows.

- If $y = 0$ then $H_x(y) = \omega^0 = 1$.
- If y is one of the representatives, then we read the appropriate value according to the folded encoding.
- Otherwise, we find a representative u and j such that $y = j \cdot u$, read $H_x(u)$ by the previous rule, and set $H_x(y) = \Big(H_x(u) \Big)^j$.

Since H_x is a linear function, these extraction rules are consistent with the original function.

For every codeword $w \in P$, we denote by $x_w \in \mathbb{Z}_p^k$ the vector that satisfies $w = Gx_w$, and we denote by $\pi_w : \mathbb{Z}_p^k \to \mathbb{C}$ the Hadamard encoding of x_w, i.e. $\pi_w = H_{x_w}$. We assume that π_w is represented in its folded form, so the actual representation of π_w takes $\frac{p^k-1}{p-1}$ values in $\mathbb{Z}_p$.

Consider the following 3-query linear inspective verifier V for P

```
INSPECTIVE VERIFIER  $V$

INPUT (AS ORACLES): $w \in \mathbb{Z}_p^n, \pi : \mathbb{Z}_p^k \to \mathbb{C}$

  1. Choose $y \in \mathbb{Z}_p^k$ and $i \in [n]$ uniformly at random
  2. Output accept if and only if $\pi(y) \cdot \omega^{w_i} = \pi(y + g_i)$
```

Claim 9.3.2. *The inspective verifier V satisfies the following properties*

- Completeness: *If $w \in P$ and $\pi = \pi_w$ then* $\Pr\left[V^{(w,\pi_w)} = \mathsf{accept}\right] = 1$
- Soundness: *For any $w \in \mathbb{Z}_p^n$ and any (folded) $\pi \in \mathbb{Z}_p^{\frac{p^k-1}{p-1}}$,* $\Pr\left[V^{(w,\pi)} = \mathsf{reject}\right] \geq \delta(w, P)$

Before proceeding to the proof of Claim 9.3.2, we note that Theorem 9.2.2 follows immediately from this claim.

Proof. For a codeword $w = G \cdot x_w \in P$ and a legal proof $\pi_w = H_{x_w}$ we have $w_i = (g_i \cdot x_w)$, and together with the fact that H_{x_w} is linear we have

$$\pi_w(y + g_i) = \pi_w(y) \cdot \pi_w(g_i) = \pi_w(y) \cdot \omega^{(g_i \cdot x_w)} = \pi(y) \cdot \omega^{w_i}$$

and the completeness condition is satisfied. Now we prove that the soundness of V is as required.

In the following we use the fact that the function π is represented in folded form, and hence for every $y \in \mathbb{Z}_p^k$ and $j \in [p]$ we have $\pi(j \cdot y) = \Big(\pi(y)\Big)^j$. Denote by s the soundness of V, i.e., the probability that it rejects a word-proof pair. We are going to express s in terms of $\delta(w, P)$ by making some manipulations on the Fourier expansion of π. According to the description of algorithm V,

$$s = \Pr_{y,i}[\pi(y)\omega^{w_i}\overline{\pi(y + g_i)} = 1]$$

and according to Lemma 9.3.1, if η is a p^{th} root of unity, then the sum $\sum_{j \in [p] \setminus \{0\}} \eta^j$ equals $p - 1$ when $\eta = 1$, and it equals -1 otherwise. Thus for all pairs (w, π) we have

$$(p-1)(1-s) - s = \mathbb{E}_{y,i}\Big[\sum_{j \in [p] \setminus \{0\}} \Big(\pi(y)\omega^{w_i}\overline{\pi(y+g_i)}\Big)^j\Big] =$$

$$\mathbb{E}_{y,i}\Big[\sum_{j \in [p] \setminus \{0\}} \pi(jy)\omega^{jw_i}\overline{\pi(jy + jg_i)}\Big] =$$

$$\mathbb{E}_{y,i}\Big[\sum_{j \in [p] \setminus \{0\}} \omega^{jw_i}\big(\sum_{\alpha} \hat{\pi}_\alpha \chi_\alpha(jy)\big)\big(\sum_{\beta} \overline{\hat{\pi}_\beta \chi_\beta(jy)\chi_\beta(jg_i)}\big)\Big] =$$

$$\sum_{\alpha,\beta} \hat{\pi}_\alpha \overline{\hat{\pi}_\beta} \sum_{j \in [p] \setminus \{0\}} \mathbb{E}_i\Big[\omega^{jw_i}\overline{\chi_\beta(jg_i)}\Big]\mathbb{E}_y\Big[\chi_\alpha(jy)\overline{\chi_\beta(jy)}\Big].$$

By the orthonormality of the character functions this equals

$$\sum_{\alpha} |\hat{\pi}_\alpha|^2 \sum_{j \in [p] \setminus \{0\}} \mathbb{E}_i\Big[\omega^{jw_i}\overline{\chi_\alpha(jg_i)}\Big] =$$

$$\sum_{\alpha} |\hat{\pi}_\alpha|^2 \mathbb{E}_i\Big[\sum_{j \in [p] \setminus \{0\}} \omega^{jw_i}\overline{\chi_\alpha(jg_i)}\Big] =$$

$$\sum_{\alpha} |\hat{\pi}_\alpha|^2 \mathbb{E}_i\Big[\sum_{j \in [p] \setminus \{0\}} \Big(\omega^{w_i}\overline{\chi_\alpha(g_i)}\Big)^j\Big] =$$

$$\sum_{\alpha} |\hat{\pi}_\alpha|^2 \mathbb{E}_i\Big[\sum_{j \in [p] \setminus \{0\}} \Big(\omega^{w_i - \alpha \cdot g_i}\Big)^j\Big].$$

By Lemma 9.3.1, for every i such that $w_i = \alpha g_i$ (the agreeing indices) the sum $\sum_{j\in[p]\setminus\{0\}} \left(\omega^{w_i-\alpha\cdot g_i}\right)^j$ evaluates to $p-1$, and for all other indices i this sum evaluates to -1, therefore the above equals to

$$\sum_{\alpha} |\hat{\pi}_\alpha|^2 \Big(\big(1-\delta(w,G\alpha)\big)(p-1) - \delta(w,G\alpha)\Big) \leq$$

$$\Big(\big(1-\delta(w,P)\big)(p-1) - \delta(w,P)\Big) \sum_{\alpha} |\hat{\pi}_\alpha|^2 =$$

$$p-1-p\delta(w,P)$$

The last inequality is due to Parseval's identity. To conclude, we have $(p-1) - ps \leq (p-1) - p\delta(w,P)$, or simply $s \geq \delta(w,P)$ as required. $\square$

9.4 Proof of Length-Soundness tradeoff (Theorem 9.2.3)

The proof is organized as follows. In Section 9.4.1 we define *constraint graphs*, which are later used to analyze inspective verifiers. In Section 9.4.2 we prove an auxiliary lemma that allows us to convert any verifier $\mathcal{V} = \langle \mathcal{Q}, D\rangle$ to a verifier $\mathcal{V}' = \langle \mathcal{Q}', D'\rangle$ such that $\mathcal{V}'$ achieves almost the same soundness as $\mathcal{V}$, but the size of $\mathcal{Q}$ is linear in the length of the proof, and the distribution D' is uniform over $\mathcal{Q}$. In Section 9.4.3 we prove that the soundness of inspective verifiers goes to zero as long as the proof length is sub-exponential. Based on these, we prove Theorem 9.2.3 in Section 9.4.4 and complete several missing proofs in Section 9.4.5.

9.4.1 Constraint graphs and the generalized Decomposition Lemma

Definition 34 (Constraint Graphs)**.** *A constraint graph is a pair $\phi = (G, C)$, where $G = (V, E)$ is a directed multigraph and $C = \Big\{c_e : \{0,1\}^2 \to \{\mathsf{accept}, \mathsf{reject}\} \mid e \in E\Big\}$ is a set of binary constraints associated with the edges of G.*

*If an assignment $\pi : V \to \{0,1\}$ satisfies a δ-fraction of the constraints in ϕ then we say that π δ-*satisfies *ϕ. Namely, π is δ-satisfying if $\left|\Big\{e = (u,v) \in E : c_e\Big(\pi(u), \pi(v)\Big) = \mathsf{accept}\Big\}\right| = \delta|E|$.*

A constraint graph ϕ is unsatisfiable *if there is no assignment that 1-satisfies it. We also say that ϕ is ε-*far *from being satisfiable if there is no assignment $\pi : V \to \{0,1\}$ that $(1-\varepsilon)$-satisfies ϕ.*

We say that a constraint graph $\phi' = (G', C')$ is a *subgraph* of $\phi = (G, C)$ if G' is a subgraph of G, and in addition, for every $e \in E(G')$ the corresponding constraints $c_e \in C$ and $c'_e \in C'$ are identical.

The following main lemma is a natural generalization of the decomposition lemma from [LR99], which is useful when analyzing graphs with general edge-constraints (rather than linear ones). The lemma states that any constraint graph which is far from being satisfiable has a small unsatisfiable subgraph (witness of unsatisfiability).

Lemma 9.4.1. *Let $\phi = (G, C)$ be a constraint graph which is ε-far from being satisfiable. Then ϕ has an unsatisfiable subgraph ϕ' with at most $\frac{4 \log |E(G)|}{\varepsilon} + 2$ edges.*

Observe that an immediate corollary of Lemma 9.4.1 is that if a 2-CSP formula with m constraints is ε-far from being satisfiable (meaning that any assignment falsifies at least εm constraints) then it has an unsatisfiable subset of at most $\frac{4 \log m}{\varepsilon} + 2$ constraints.

Before proving the lemma we need some definitions.

Definition 35 (Forcing)**.** *Let $\phi = (G, C)$ be a constraint graph, and let $u \in V(G)$ and $b_u \in \{0,1\}$ be a vertex of G and a value assigned to it, respectively. For every vertex $v \in V(G) \setminus \{u\}$ and any value $b_v \in \{0,1\}$, we say that $(u \leftarrow b_u)$* forces *$(v \leftarrow b_v)$ if*

- *the partial assignment $\pi : \{u, v\} \to \{0,1\}$ defined as $\pi(u) = b_u$ and $\pi(v) = b_v$ does not violate any constraint in C*
- *the partial assignment $\pi' : \{u, v\} \to \{0,1\}$ defined as $\pi'(u) = b_u$ and $\pi'(v) = 1 - b_v$ violates at least one constraint $c_e \in C$ (and the violated constraints are called the* forcing constraints*).*

Observe that $(u \leftarrow b_u)$ forces $(v \leftarrow b_v)$ if and only if $(v \leftarrow 1 - b_v)$ forces $(u \leftarrow 1 - b_u)$.

We can naturally extend the notion of forcing for subsets of vertices as follows. Let $U \subset V(G)$ be a subset of G's vertices, and let $\pi_U : U \to \{0,1\}$ be a partial assignment on U. For every vertex $v \in V(G) \setminus U$ and every value $b_v \in \{0,1\}$ we say that π_U forces $(v \leftarrow b_v)$ if there exists a vertex $u \in U$ such that $(u \leftarrow \pi_U(u))$ forces $(v \leftarrow b_v)$.

In some cases there is no immediate forcing between assignments, but there is an indirect implication. We say that $(u \leftarrow b_u)$ *implies* $(v \leftarrow b_v)$ if there are $k > 0$ vertices $x_1, x_2, \ldots, x_k \in V \setminus \{u, v\}$ and k values $b_1, b_2, \ldots, b_k \in \{0,1\}$ such that:

- $(u \leftarrow b_u)$ forces $(x_1 \leftarrow b_1)$
- for all $1 \le i < k$, $(x_i \leftarrow b_i)$ forces $(x_{i+1} \leftarrow b_{i+1})$
- $(x_k \leftarrow b_k)$ forces $(v \leftarrow b_v)$.

We also define the *implication path* from $(u \leftarrow b_u)$ to $(v \leftarrow b_v)$ as the corresponding path of $k + 1$ forcing edges from u to v.

If for some pair of vertices $u, v \in V$ and a value $b_u \in \{0,1\}$ the assignment $(u \leftarrow b_u)$ implies both $(v \leftarrow 0)$ and $(v \leftarrow 1)$, it means that $(u \leftarrow b_u)$ leads to contradiction, and hence any assignment π for which $\pi(u) = b_u$ cannot satisfy ϕ. In this case we call the pair of corresponding implication paths a *contradiction cycle*. Furthermore, if both $(u \leftarrow 0)$ and $(u \leftarrow 1)$ lead to contradiction, then clearly the constraint graph is unsatisfiable. In this case we call the pair of corresponding contradiction cycles a *witness of unsatisfiability*.

Given a subset $U \subset V$, a partial assignment $\pi_U : U \to \{0,1\}$ has no consistent extensions if one of the following holds:

- π_U forces two different values on some $v \in V \setminus U$
- there exists an edge $e = (v_1, v_2) \in E(V \setminus U)$ such that π_U forces the values b_1, b_2 on v_1, v_2 respectively, and $c_e(b_1, b_2) = \mathsf{reject}$

Notice that in both cases there is a contradiction cycle witnessing the inextensibility of π_U.

If π_U has a consistent extension, then we denote by $f(U) \triangleq \{v_1, \ldots, v_k\} \subseteq V \setminus U$ the set of all vertices that are forced by π_U to have the values $b_{v_1}, \ldots, b_{v_k}$ respectively, and we define the *forced extension* of π_U which is an assignment $\pi_{U \cup f(U)} : U \cup f(U) \to \{0,1\}$ given by

$$\pi_{U \cup f(U)}(v) = \begin{cases} \pi_U(v) & , \quad v \in U \\ b_v & , \quad v \in f(U) \end{cases}.$$

Proof of Lemma 9.4.1. Assume for the sake of contradiction that $\phi = (G, C)$ is the smallest constraint graph that violates the conditions of Lemma 9.4.1. Namely, ϕ is ε-far from being satisfiable, but it has no unsatisfiable subgraph with at most $\frac{4 \log |E(G)|}{\varepsilon} + 2$ edges. Pick an arbitrary vertex $r \in V(G)$ and consider the executions **FindContradiction**$(r, 0)$ and **FindContradiction**$(r, 1)$ of the following algorithm, which is basically a BFS algorithm starting from vertex r that proceeds along forcing edges.

```
FindContradiction(r, b)

    1. Set $U = \{r\}$, $i = 0$, and define a partial assignment $\pi_U$ as $\pi_U(r) = b$.
    2. $i = i + 1$.
    3. If $i > \frac{\log|E(G)|}{\varepsilon}$ output FAIL.
    4. If $\pi_U$ has a consistent extension $\pi_{U \cup f(U)}$ to the set $f(U)$ of the forced
       neighbors of $U$:
        (a) If $|E(f(U), U)| \geq \varepsilon |E(U)|$ then set $U = U \cup f(U)$, set $\pi_U = \pi_{U \cup f(U)}$
            and go to step 2.
        (b) Else output FAIL.
```

5. *Else there must be a contradiction cycle $\mathcal{W}$ of length at most $2i + 1 \leq \frac{2\log|E(G)|}{\varepsilon} + 1$ for the assignment $(r \leftarrow b)$[8]. Output $\mathcal{W}$.*

If both executions **FindContradiction**$(r, 0)$ and **FindContradiction**$(r, 1)$ reached Step 5 then we have a pair of contradiction cycles (each of length at most $\frac{2\log|E(G)|}{\varepsilon}+1$) for both $(r \leftarrow 0)$ and $(r \leftarrow 1)$. Joined together, these cycles form a witness of unsatisfiability of length at most $\frac{4\log|E(G)|}{\varepsilon} + 2$, contradicting our assumption that ϕ has no unsatisfiable subgraph with at most $\frac{4\log|E(G)|}{\varepsilon} + 2$ edges. Therefore, one of the executions must output FAIL either in Step 3 or in Step 4b.

Since in every iteration of the algorithm $|E(U)|$ grows by a multiplicative factor of at least $(1+\varepsilon)$, after $\frac{\log|E(G)|}{\varepsilon} > \log_{(1+\varepsilon)}|E(G)|$ iterations we get $|E(U)| > |E(G)|$, which is of course impossible. This completely rules out the possibility of outputting FAIL in Step 3.

Finally, assume towards a contradiction that one of the executions outputs FAIL in Step 4b. Consider the induced subgraphs $G_U = G(U)$ and $G_{V\setminus U} = G(V \setminus U)$, and the corresponding induced constraint graphs $\phi_U = (G_U, C_U)$ and $\phi_{V\setminus U} = (G_{V\setminus U}, C_{V\setminus U})$ where C_U and $C_{V\setminus U}$ are the sets of all original constraints associated with $E(U)$ and $E(V \setminus U)$ respectively.

According to Algorithm **FindContradiction(r,b)**, the set U is enlarged only when the assignment π_U has a consistent extension. This fact preserves the invariant that the constraints $\{c_e : e \in E(U)\}$ are always satisfied by π_U. Therefore π_U completely satisfies the subgraph ϕ_U. On the other hand, by the minimality condition $\phi_{V\setminus U}$ must be $1-\varepsilon$ satisfiable by some assignment $\pi_{V\setminus U}$. Let $\pi : V(G) \to \{0,1\}$ be the union of π_U and $\pi_{V\setminus U}$, defined as

$$\pi(v) = \begin{cases} \pi_U(v) & , \quad v \in U \\ \pi_{V\setminus U}(v) & , \quad v \in V \setminus U \end{cases}.$$

Since the execution was terminated in Step 4b, π falsifies at most $\varepsilon|E(U)|$ of the constraints on $E(U, V \setminus U)$. So the total number of unsatisfied constraints by π is bounded by $\varepsilon|E(V \setminus U)| + \varepsilon|E(U, V \setminus U)| \leq \varepsilon|E(G)|$, contradicting our initial assumption. $\square$

9.4.2 The uniform (sparse) verifier lemma

In this section we claim that without loss of generality we can concentrate on (q, n, ℓ)-verifiers that make roughly $O(n + \ell)$ uniformly distributed queries. This assumption eases the application of Lemma 9.4.1, which bounds the size of contradiction witnesses as a function of number of edges (rather than number of vertices as in Lemma 9.5.3).

We note that a similar lemma was already proved in [GS02] for $(q, n, 0)$-verifiers (property testers).

[8]The bound on the cycle length is due to the fact that every implication in U has a corresponding implication path of length at most i that follows the iterative extension of π_U.

Lemma 9.4.2. *For every $\gamma > 0$ and property $P \subset \Sigma^n$, if P has a (q, n, ℓ)-verifier $\mathcal{V} = \langle \mathcal{Q}, D\rangle$ with perfect completeness and soundness function $s : (0, 1] \to [0, 1]$ then P also has a (q, n, ℓ)-verifier $\mathcal{V}' = \langle \mathcal{Q}', U\rangle$ with the following properties.*

1. *$\mathcal{V}'$ has perfect completeness.*
2. *$\mathcal{V}'$ has soundness function s' that for all δ satisfies $s'(\delta) \geq s(\delta) - \gamma$.*
3. *The number of queries in $\mathcal{Q}'$ is $\gamma^{-2}(n + \ell)\log|\Sigma|$.*
4. *U is the uniform distribution over $\mathcal{Q}'$.*

Proof. We prove the lemma by the following probabilistic argument. Construct a multi-set $\mathcal{Q}'$ by choosing independently at random $\gamma^{-2}(n + \ell)\log|\Sigma|$ queries $Q \in \mathcal{Q}$ according to distribution D. Given $\mathcal{Q}'$, the new verifier $\mathcal{V}'$ operates similarly to $\mathcal{V}$, but instead of choosing queries from $\mathcal{Q}$ according to distribution D, it chooses them from $\mathcal{Q}'$ according to the uniform distribution.

Since the original verifier $\mathcal{V}$ had perfect completeness and since $\mathcal{Q}' \subseteq \mathcal{Q}$, $\mathcal{V}'$ has perfect completeness too. Conditions 3 and 4 of the lemma follow (with probability 1 over the choice of $\mathcal{Q}'$) from the definition of $\mathcal{Q}'$ and $\mathcal{V}'$. We only need to show that the soundness function s' of $\mathcal{V}'$ satisfies $s'(\delta) \geq s(\delta) - \gamma$ for all $\delta > 0$. Clearly, this is satisfied for all δ for which $s(\delta) \leq \gamma$ because the rejection probability is always non-negative. Therefore, to complete the proof it is enough to show that with positive probability the set $\mathcal{Q}'$ satisfies the following: For every word w such that $s\Big(\delta(w, P)\Big) > \gamma$ and every proof π, at least an $\Big(s\Big(\delta(w, P)\Big) - \gamma\Big)$-fraction of the queries in $\mathcal{Q}'$ reject the pair $w \sqcup \pi$ (we say that the query $Q = (I, C)$ *rejects* the pair $w \sqcup \pi$ if $C(w \sqcup \pi|_I) = \mathsf{reject}$).

Fix a word $w \in \Sigma^n$ such that $s\Big(\delta(w, P)\Big) > \gamma$ and a proof $\pi \in \Sigma^\ell$. For every $Q \in \mathcal{Q}$, we define the indicator variable $x_{Q,w\sqcup\pi}$ which is equal to 1 if Q rejects the pair $w \sqcup \pi$. Notice that once w is fixed, for any proof π we have $\mathbb{E}_{Q\sim_D \mathcal{Q}}[x_{Q,w\sqcup\pi}] \geq s\Big(\delta(w, P)\Big)$.

We also define an indicator variable $I_{w\sqcup\pi}$ which equals 1 if the fraction of queries in $\mathcal{Q}'$ that reject the pair $w \sqcup \pi$ is at least $s\Big(\delta(w, P)\Big) - \gamma$. Since the queries in $\mathcal{Q}'$ were chosen independently (according to the distribution D), by Chernoff's bound for any w and any π we have

$$\Pr_{\mathcal{Q}'}[I_{w,\pi} = 0] = \Pr_{\mathcal{Q}'}\left[\Big(\frac{1}{|\mathcal{Q}'|}\sum_{Q\in\mathcal{Q}'} x_{Q,w\sqcup\pi}\Big) < s\Big(\delta(w, P)\Big) - \gamma\right] \leq$$

$$\leq \exp(-2\gamma^2|\mathcal{Q}'|) = \exp(-2\gamma^2\gamma^{-2}(n + \ell)\log|\Sigma|) <$$

$$< |\Sigma|^{-n-\ell}$$

and if we apply the union bound over all word-proof pairs $w \sqcup \pi$ we get

$$\Pr_{\mathcal{Q}'}[I_{w,\pi} = 0 \text{ for some pair } w \sqcup \pi \text{ as above}] < |\Sigma|^{n+\ell} \cdot |\Sigma|^{-n-\ell} < 1.$$

We conclude that there must be a query set $\mathcal{Q}'$ that satisfies the required soundness condition.

□

9.4.3 Best soundness for inspective verifiers (proof of Theorem 9.2.8)

Before proceeding to the proof we need to define the following object, which is basically a graph that is induced by a verifier. This graph plays a crucial role also in the proof of Lemma 9.5.2.

Definition 36 (Inspective Graph). *Let $\mathcal{V} = \langle \mathcal{Q}, D \rangle$ be a (q, n, ℓ)-verifier. For $Q = (I, C)$ of i-size 2 we say Q* generates *the pair $I \cap [n+1, n+\ell]$. Similarly, if Q is of i-size 1 then we say that it* generates *the pair $0, I \cap [n+1, n+\ell]$. A query of i-size different from 1, 2 generates no pair. The* inspective graph *of $\mathcal{V}$, denoted $G_{\mathcal{V}}$, is the multigraph with vertex set $V = \{0\} \cup [n+1, n+\ell]$ and edge set E being the multiset of pairs generated by $\mathcal{Q}$.*

Proof of Theorem 9.2.8. Let $P \subset \{0,1\}^n$ be a d-universal property, and let us fix $\varepsilon \in (0,1)$ and $\delta \in (0,1)$. Let $\mathbf{V_i}$ be an inspective (q, n, ℓ) verifier for P and let $\mathbf{V_i}' = \langle \mathcal{Q}', U \rangle$ be the corresponding "sparse" verifier (which is also inspective) described in Lemma 9.4.2 for $\gamma = \varepsilon$.

Fixing a δ-far word w defines a constraint graph $\phi_w = (G, C)$ over $\ell + 1$ vertices as follows:

- G is the inspective graph induced by $\mathbf{V_i}'$ as per Definition 36.
- for every $e = (u, v) \in E(G)$, the constraint c_e evaluates to accept whenever the valuation $\pi(u), \pi(v)$ and the word w satisfy the query in $\mathcal{Q}'$ (with i-size 2) that generates the edge e.
- for every $e = (0, v) \in E(G)$, the (unary) constraint c_e evaluates to accept whenever the valuation $\pi(v)$ and the word w satisfy the query in $\mathcal{Q}'$ (with i-size 1) that generates the edge e.

Note that according to Lemma 9.4.2, the number of edges in $E(G)$ is bounded by $\varepsilon^{-2}(n+\ell)$. In addition, every constraint c_e depends on at most $q - 1$ word bits.

Since the minimal rejection probability of δ-far words by $\mathbf{V_i}'$ is $s_{\mathrm{i}}(\delta) - \varepsilon$, the constraint graph ϕ_w must be $(s_{\mathrm{i}}(\delta) - \varepsilon)$-far from being satisfiable. Hence by Lemma 9.4.1, ϕ_w has an unsatisfiable subgraph ϕ with at most

$$\frac{4\log|E(G)|}{s_{\mathrm{i}}(\delta)-\varepsilon}+2 \le \frac{4\log(\varepsilon^{-2}(n+\ell))}{s_{\mathrm{i}}(\delta)-\varepsilon}+2$$

edges. Let $i_1, i_2, \ldots, i_k \in [n]$ be the word bits associated with the constraints (edges) of the unsatisfiable subgraph ϕ, where $k \le (q-1)\cdot(\frac{4\log(\varepsilon^{-2}(n+\ell))}{s_{\mathrm{i}}(\delta)-\varepsilon}+2)$. It is clear that any word $w' \in \{0,1\}^n$ that agrees with w on indices $i_1, i_2, \ldots, i_k$ cannot be in the property P. Therefore, because of the universality condition k must be larger than d, implying

$$(q-1)\cdot(\frac{4\log(\varepsilon^{-2}(n+\ell))}{s_{\mathrm{i}}(\delta)-\varepsilon}+2) > d$$

or equivalently

$$s_{\mathrm{i}}(\delta) < \frac{4\log(\varepsilon^{-2}(n+\ell))}{\frac{d}{q-1}-2}+\varepsilon.$$

□

Corollary 9.4.3. *Let $\alpha \in (0,1)$ be a positive constant and let $\mathcal{P} \triangleq \{P_n \subseteq \{0,1\}^n : P_n \text{ is } \alpha n\text{−universal}\}$ be a family of αn-universal properties. The properties in $\mathcal{P}$ have no sub-exponential* **inspective** *PCPP's achieving constant soundness. Namely, for every $\varepsilon' \in (0,1]$ there are $\beta > 0$ and $n_0 \in \mathbb{N}$ such that for any property $P_n \in \mathcal{P}$, $n > n_0$ the following is satisfied for all $\delta \in [0,1]$:*

$$\mathcal{S}^{P_n}_{\mathbf{V_i}}\Big(3, 2^{\beta n}, \delta\Big) \le \varepsilon'.$$

Proof. Fix an arbitrary $\varepsilon' > 0$, and set $\beta > 0$ and $n_0 \in \mathbb{N}$ such that all $n > n_0$ satisfy the inequality

$$2^{\beta n} < 2^{\frac{\varepsilon'}{8}(\frac{\alpha n}{2}-2)+2\log\varepsilon'-2} - n.$$

Since P_n is an αn-universal property, we can apply Theorem 9.2.8 (with $q = 3$ and $\varepsilon = \varepsilon'/2$) and get that for every $\delta \in [0,1]$:

$$\mathcal{S}^{P_n}_{\mathbf{V_i}}\Big(3, 2^{\beta n}, \delta\Big) \le \frac{4\Big(\log(n+2^{\beta n}) - 2\log\varepsilon' + 2\Big)}{\frac{\alpha n}{2}-2} + \varepsilon'/2,$$

additionally, according to our choice of β and n_0 we also have:

$$\frac{4\Big(\log(n+2^{\beta n}) - 2\log\varepsilon' + 2\Big)}{\frac{\alpha n}{2}-2} \le \varepsilon'/2,$$

completing the proof. □

9.4.4 Proof of Theorem 9.2.3

Theorem 9.2.3 (restated) *Let $\alpha \in (0,1)$ be a positive constant and let $\mathcal{P} \triangleq \{P_n \subseteq \{0,1\}^n : n \in \mathbb{N}\}$ be a family of linear properties (codes) with dual distance at least αn. The properties in $\mathcal{P}$ have no sub-exponential PCPP's achieving soundness larger than $1/3$. Namely, for every $\varepsilon \in (0,1]$ there are $\beta > 0$ and $n_0 \in \mathbb{N}$ such that for any property $P_n \in \mathcal{P}$, $n > n_0$ the following is satisfied for all $\delta \in [0,1]$:*

$$\mathcal{S}^{P_n}\left(3, 2^{\beta n}, \delta\right) \leq \frac{1}{3} + \varepsilon.$$

Before proceeding to the proof of Theorem 9.2.3 we need the following lemma, which is proved in the next section.

Lemma 9.4.4. *Let $\mathcal{V}$ be a $(3, n, \ell)$ verifier for an $\mathbb{F}_p$-linear property $P \subseteq \mathbb{F}_p^n$ with dual distance at least 4. Let μ be the probability that $\mathcal{V}$ makes an inspective query (i.e., one that makes at most two queries into the proof). Then, using $s^{\mathcal{V}}$ to denote the soundness function of $\mathcal{V}$, we have for every $\delta < 1/2$*

$$s^{\mathcal{V}}(\delta) \leq \min\left\{1 - \mu + \mathcal{S}^{P}_{\mathbf{V}_\mathrm{i}}\left(3, \ell, \delta\right), (1 - \frac{1}{p})\mu\right\}.$$

Proof of Theorem 9.2.3. Fix any $\varepsilon \in (0,1]$, and let $\beta > 0$ and n_0 be the parameters promised by Corollary 9.4.3, so that $\mathcal{S}^{P_n}_{\mathbf{V}_\mathrm{i}}\left(3, 2^{\beta n}, \delta\right) < \varepsilon$ for every $n > n_0$.

Notice that the right hand side of the inequality in Lemma 9.4.4 ($p = 2$ in our case) is maximized when the two terms are equal, i.e., when $\mu = \frac{2}{3}\left(1 + \mathcal{S}^{P}_{\mathbf{V}_\mathrm{i}}\left(3, \ell, \delta\right)\right)$. Therefore, for $n > n_0$ and proofs of length $2^{\beta n}$,

$$s^{\mathcal{V}}(\delta) \leq \frac{1}{3}(1 + \mathcal{S}^{P_n}_{\mathbf{V}_\mathrm{i}}\left(3, 2^{\beta n}, \delta\right)) < \frac{1}{3} + \varepsilon,$$

where the second inequality follows from Corollary 9.4.3.

$\square$

9.4.5 Proof of Lemma 9.4.4

Proof. To see why $s^{\mathcal{V}}(\delta) \leq 1 - \mu + \mathcal{S}^{P}_{\mathbf{V}_\mathrm{i}}\left(3, \ell, \delta\right)$ convert $\mathcal{V} = \langle \mathcal{Q}, D\rangle$ into an inspective verifier $\mathcal{V}'$ as follows. $\mathcal{V}'$ picks $Q \sim D$ in the same manner that $\mathcal{V}$ does. If Q is an inspective query, $\mathcal{V}'$ performs it. Otherwise, $\mathcal{V}'$ performs the trivial (inspective) query that always accepts (without reading any information). Since $\mathcal{V}'$ is inspective, we conclude $s^{\mathcal{V}'} \leq \mathcal{S}^{P}_{\mathbf{V}_\mathrm{i}}\left(3, \ell, \delta\right)$, i.e., there exists some input w that is δ-far from $\mathcal{C}$ and a proof π such that $(w \sqcup \pi)$ is rejected by $\mathcal{V}'$ with probability at most $\mathcal{S}^{P}_{\mathbf{V}_\mathrm{i}}\left(3, \ell, \delta\right)$. Even if $\mathcal{V}$ rejects all non-inspective queries on this particular pair, this can only increase the soundness by an additive factor $1-\mu$, implying the first inequality.

To show that $s^{\mathcal{V}}(\delta) \leq (1-\frac{1}{p})\mu$ we need the following two lemmas, which we prove shortly.

Lemma 9.4.5. *Let $\mathcal{C} \subset \mathbb{F}_p^n$ be a linear code. For any $x \in \mathbb{F}_p^n$ and any codeword $w \in \mathcal{C}$,*

$$\delta(x+w,\mathcal{C}) \geq \delta(x,\mathcal{C}).$$

Lemma 9.4.6. *Let $\mathcal{C} \subset \mathbb{F}_p^n$ be a linear code with dual distance at least $d+1$, and let $I \subset [n]$ be a set of at most d indices. For any $x \in \mathbb{F}_p^n$ and any $y \in \mathbb{F}_p^d$,*

$$\Pr_{w \sim_U \mathcal{C}}[(x+w)|_I = y] = p^{-d},$$

and in particular, for any $y \in \mathbb{F}_p^d$,

$$\Pr_{w \sim_U \mathcal{C}}[w|_I = y] = p^{-d}.$$

The proof proceeds as follows. First we fix a δ-far word $x \in \mathbb{F}_p^n$, and pick $\hat{w} \in \mathcal{C}$ uniformly at random. Let π denote the legitimate proof for the codeword $\hat{w}$. Then, we pick another codeword $w' \in \mathcal{C}$ uniformly at random, and set $w \triangleq x + w'$. Recall that according to Lemma 9.4.5, w is δ-far from $\mathcal{C}$. We use the word-proof pair $(w \sqcup \pi)$ to fool the verifier $\mathcal{V} = \langle \mathcal{Q}, D \rangle$, i.e. to make it reject with probability at most $(1-\frac{1}{p})\mu$.

Let $\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3$ be a partition of $\mathcal{Q}$, where $\mathcal{Q}_i$ contains all queries that read i bits from the proof. Since the verifier $\mathcal{V}$ has perfect completeness, all queries in $\mathcal{Q}_3$ must be satisfied because π is a legitimate proof and all queries in $\mathcal{Q}_0$ (tester queries) must be satisfied because the dual distance of $\mathcal{C}$ is larger than three. In addition, the queries in $\mathcal{Q}_2$ are satisfied with probability at least $1/p$, since according to Lemma 9.4.6 for every $i \in [n]$, $w_i = \hat{w}_i$ with probability $1/p$. To complete the proof, it is enough to show that every query $Q \in \mathcal{Q}_1$ is satisfied with probability at least $1/p$ over the choice of $\hat{w}$ and w'.

Let $Q = (I, C)$ be a query in $\mathcal{Q}_1$. Let i_1, i_2 be the indices in $I \cap [n]$ and let j be the index in $I \cap [n+1, n+\ell]$, so that the query Q is satisfied whenever $C(\alpha_1, \alpha_2, \pi_j) = \mathsf{accept}$. For every $\beta \in \mathbb{F}_p$, let k_β denote the number of assignments $(\alpha_1, \alpha_2) \in \mathbb{F}_p^2$ for which $C(\alpha_1, \alpha_2, \beta) = \mathsf{accept}$. Since the dual distance of $\mathcal{C}$ is larger than two, we know that for each one of the p^2 possible assignments $(\alpha_1, \alpha_2) \in \mathbb{F}_p^2$ there exists a value $\pi_j \in \mathbb{F}_p$ such that $C(\alpha_1, \alpha_2, \pi_j) = \mathsf{accept}$, therefore $\sum_{\beta \in \mathbb{F}_p} k_\beta \geq p^2$.

Recall that we chose π by the following distribution: by picking a codeword $\hat{w} \in \mathcal{C}$ uniformly at random, and setting π to be the legitimate proof for the codeword $\hat{w}$. According to Lemma 9.4.6, the values of all pairs of indices in the word w are distributed uniformly. Therefore, once $\hat{w}$ is chosen (and the corresponding proof π is set), the query

Q is satisfied by $\left(w \sqcup \pi\right)$ with probability k_{π_j}/p^2 over the choice of $w' \in \mathcal{C}$.

Let η_β denote the probability (over the random choice of $\hat{w} \in \mathcal{C}$) that $\pi_j = \beta$. By Lemma 9.4.6 the values $\hat{w}_{i_1}$ and $\hat{w}_{i_2}$ are distributed uniformly and independently of each other, and therefore

$$\eta_\beta = \Pr[\pi_j = \beta] \geq \frac{k_\beta}{\sum_\gamma k_\gamma}.$$

So the overall acceptance probability is

$$\Pr_{\hat{w},w'}[C(w_{i_1}, w_{i_2}, \pi_j) = \mathsf{accept}] = \sum_\beta \eta_\beta \cdot \frac{k_\beta}{p^2} \geq \sum_\beta \left(\frac{k_\beta}{\sum_\gamma k_\gamma} \cdot \frac{k_\beta}{p^2}\right) = \frac{1}{p^2 \sum_\gamma k_\gamma} \sum_\beta k_\beta^2.$$

Recall that $\sum_\beta k_\beta \geq p^2$. In addition, by Cauchy-Schwartz inequality we know that

$$\sum_\beta k_\beta^2 \geq \frac{1}{p}(\sum_\beta k_\beta)^2 \geq p \sum_\beta k_\beta$$

hence the acceptance probability is at least $1/p$ as required.

We constructed a distribution of word-proof pairs $(w \sqcup \pi)$ in which all words are δ-far from $\mathcal{C}$, and all proofs are legitimate proofs. Any query from $\mathcal{Q}_3$ is satisfied with probability 1 under this distribution, and all other queries are satisfied with probability at least $1/p$. So by linearity of expectation, we conclude that there must be a pair $(w \sqcup \pi)$ (where w is δ-far from $\mathcal{C}$) that is accepted by the verifier $\mathcal{V}$ with probability at least $(1-\mu) \cdot 1 + \mu \cdot \frac{1}{p} = 1 - (1 - \frac{1}{p})\mu$.

□

Proofs of Lemma 9.4.5 and Lemma 9.4.6

Proof of Lemma 9.4.5. Assume towards a contradiction that for some $x \in \mathbb{F}_p^n$ and $w \in \mathcal{C}$ we have $\delta(x+w, \mathcal{C}) < \delta(x, \mathcal{C})$. Let $w' \in \mathcal{C}$ be the closest codeword to $x+w$, i.e. a codeword for which $\delta(x+w, w') = \delta(x+w, \mathcal{C})$. Observe that $\delta(x+w, w') = \delta(x, w' + (-w))$, and that $w' + (-w) \in \mathcal{C}$. This, together with our initial assumption, leads to the following contradiction,

$$\delta(x, \mathcal{C}) > \delta(x+w, \mathcal{C}) = \delta(x+w, w') = \delta(x, w' + (-w)) \geq \delta(x, \mathcal{C}).$$

□

Proof of Lemma 9.4.6. The second part of the lemma follows from the fact that $\mathcal{C}$ has no linear constraints of weight less than $d+1$, so any projection to d (or less) indices is a surjective linear function. The first part of the lemma follows from the second part, since a constant shift of a uniform distribution yields a uniform distribution. □

9.5 Proof of Length-Soundness tradeoff for linear verifiers (Theorem 9.2.9)

We start by restating our main theorem regarding linear verifiers and its main corollary. In Subsection 9.5.1 we reduce both of these results to our main technical lemma, Lemma 9.5.2. To prove the lemma we need a variant of the decomposition lemma of [LR99], which is proved in Subsection 9.5.2. We then complete our proof by proving the last details in Subsection 9.5.3.

Theorem 9.2.5 (restated) *Let $P \subseteq \mathbb{F}^n$ be a $\mathbb{F}$-linear property. Let $s[\ell](\delta)$ denote the best soundness of a $(3, \ell)$-linear verifier for P, i.e., $s[\ell](\delta) = \mathcal{S}^P_{\mathbf{linV}}(3, \ell, \delta)$. Let $t[q](\delta)$ denote the best soundness of a q-tester for P, i.e., $t[q](\delta) = \mathcal{S}^P(q, 0, \delta)$. Then*

$$s[\ell](\delta) \le \min_{\varepsilon > 0} \left\{ t\left[\frac{36 \log \ell}{\varepsilon}\right](\delta) + \frac{1}{2} \cdot \left(1 - \frac{1}{|\mathbb{F}|} + \varepsilon\right)\right\}.$$

Corollary 9.2.6 (restated) *Let* **SUBEXP** *denote the set of subexponential functions, i.e., functions satisfying $f(n) = 2^{o(n)}$. For every prime field $\mathbb{F}_p$ there exists a family of $\mathbb{F}_p$-linear properties $\mathcal{P}$ such that*

$$\text{s-Def.}_{\mathbb{F}_p-\mathbf{linV}}[\mathcal{P}, \mathbf{SUBEXP}](\delta) \ge \delta - \frac{1}{2} \cdot \left(1 - \frac{1}{p}\right),$$

Consequently, the maximal deficiency of linear verifiers with subexponential proofs is at least $\frac{1}{2} \cdot (1 - 1/p)$:

$$\text{max-s-Def.}_{\mathbb{F}_p-\mathbf{linV}}[\mathbb{F}_p - \mathit{linear}, \mathbf{SUBEXP}] \ge \frac{1}{2} \cdot \left(1 - \frac{1}{p}\right).$$

We start by proving that the main theorem implies the corollary.

Proof of Corollary 9.2.6. Take $\mathcal{P} = \{P_n \mid n \in \mathbb{Z}^+\}$ to be a family of linear properties satisfying both *(a)* $(\dim(P_n)/n)\xrightarrow[n\to\infty]{}0$ and *(b)* the best soundness of an $o(n)$-tester for P_n goes to 0 as n goes to ∞. One construction of such a family is based on properties that have linear dual distance, i.e., where the minimal weight of a nonzero element in the dual space of P_n is $\Omega(n)$. Any $o(n)$-tester with perfect completeness for such a property must have soundness function 0. A different construction is obtained by taking $\mathcal{P}$ to be a family of random Low Density Parity Check (LDPC) codes that satisfy *(a)*. These codes were shown in [BSHR05] to satisfy *(b)*. Let $w_n \in \mathbb{F}^n$ be δ-far from P_n. The verifier in Theorem

9.2.2 achieves soundness at least δ on w when the proof-length is exponential in n. On the other hand, take ε_n to be a sequence approaching 0 when n approaches ∞ while satisfying $\frac{36 \log \ell(n)}{\varepsilon_n} = o(n)$. Such a sequence exists because $\ell(n) = 2^{o(n)}$. In this case Theorem 9.2.5 shows that the soundness of $(3, \ell(n))$-verifiers approaches $\frac{1}{2} \cdot \left(1 - \frac{1}{p}\right)$ as n approaches ∞. This proves the first part of the corollary. To get the second part notice that *(a)* implies that a random $w' \in \mathbb{F}_p^n$ has distance $\delta = ((1 - 1/p) - o(1))$ from P^n. This completes the proof. $\square$

9.5.1 Proof of Theorem 9.2.5

Overview Given a verifier $\mathcal{V}$ and a word w that is δ-far from P, we need to describe a proof π such that $\mathcal{V}$ accepts $w \circ \pi$ with relatively high probability. We divide this into two cases. If a large fraction of the queries of $\mathcal{V}$ are inspective, then we try to satisfy these queries and care little about the rejection probability on the other queries. This part is argued in Lemma 9.5.2. On the other hand, if $\mathcal{V}$ rarely queries w, then we present a proof that is good for some codeword $w' \in P$ and making it so that $\mathcal{V}$ doesn't notice the difference between w and w'. Details follow.

Notation When discussing $\mathbb{F}$-linear verifiers, we view a word-proof pair as a vector $w \sqcup \pi \in \mathbb{F}^{n+\ell}$ by setting $(w \sqcup \pi)_i = (w \sqcup \pi)[i]$. A q-query constraint $Q = (I, C)$ can be represented by a vector $v_Q \in \mathbb{F}^{n+\ell}$ such that the support of v_Q, denoted $\mathrm{supp}(v_Q)$, is I and

$$C(w \sqcup \pi|_I) = \mathsf{accept} \Leftrightarrow \langle v_Q, w \sqcup \pi \rangle = \sum_{i=1}^{n+\ell} (v_Q)_i (w \sqcup \pi)_i = 0.$$

Abusing notation, we identify Q with its representing vector and say that $(w \sqcup \pi)$ satisfies Q whenever $\langle Q, (w \sqcup \pi) \rangle = 0$. For $I' \subset [n + \ell]$ we denote $\mathrm{supp}(Q) \cap I'$ by $\mathrm{supp}_{I'}(Q)$. Similarly, let $\langle Q, w \circ \pi \rangle_{I'} = \sum_{i \in I'} Q_i \cdot (w \circ \pi)_i$, where Q_i denotes the i^{th} entry of the vector Q. Finally, for a linear space P we denote its dual space by P^*.

To simplify the proof of Theorem 9.2.5, we assume that our verifier makes no *redundant* queries according to the following definition and claim.

Definition 37. *A query $Q \in \mathbb{F}^{n+\ell}$, with $|\mathrm{supp}(Q)| \leq 3$, is called* redundant *for the property P if $|\mathrm{supp}_{[n]}(Q)| > 0, |\mathrm{supp}_{[n+1,n+\ell]}(Q)| > 0$ and there exists $u \in P^*, u \neq 0$ with $\mathrm{supp}(u) \subseteq \mathrm{supp}_{[n]}(Q)$.*

If the dual distance of P is greater than 2 then all queries are nonredundant. The next claim says that even if the dual distance of P is 2, we may assume without loss of generality that its verifier makes no redundant queries.

Claim 9.5.1. *If P has a $(3, \ell)$-linear verifier with soundness function s, then P has a $(3, \ell)$-linear verifier that makes no redundant query and has soundness function s as well.*

Proof. Let $\mathcal{V}$ be a $(3,\ell)$-linear verifier for P using redundant queries. We replace these queries, one at a time, without increasing query complexity and length and without decreasing soundness.

Let Q be redundant. Since $\left|\text{supp}_{[n]}(Q)\right| \leq 2$ and there exists $u \in P^*$ with $\text{supp}(u) \subseteq \text{supp}_{[n]}(Q)$ there exists a nonzero vector $Q' \in \text{span}(P^*, Q)$ such that $\left|\text{supp}_{[n]}(Q')\right| < \left|\text{supp}_{[n]}(Q)\right|$. Replace Q by Q'. Note that $|\text{supp}(Q')| \leq 2$ and $\left|\text{supp}_{[n+1,n+\ell]}(Q')\right| \geq 1$, so Q' is a constraint that requires a proof symbol, say $\pi_{n+\ell}$, be equal to one of the following three possibilities: *(i)* the constant 0 (if $|\text{supp}(Q')| = 1$); *(ii)* a different proof symbol $\pi_{i'}$ (if $|\text{supp}(Q')| = \left|\text{supp}_{[n+1,n+\ell]}(Q')\right| = 2$); or *(iii)* a word symbol $w_{i'}$ (if $|\text{supp}(Q')| = |\text{supp}(Q)| = 1$). In each of these three cases we can eliminate the use of π_i and calculate its value by querying a single different proof-symbol or word-symbol. By construction, the query complexity does not increase and the proof length decreases because $\pi_{n+\ell}$ is not queried anymore. By linearity, the new verifier retains perfect completeness, because every new query lies in $\text{span}(\mathcal{Q}, P^*)$. Finally, to argue soundness notice that a proof π' of length $\ell - 1$ can be extended to a proof of length ℓ such that $w \circ \pi$ satisfies a query $\hat{Q}$ of $\mathcal{V}$ if and only if $w \circ \pi'$ satisfies the modified form of $\hat{Q}$. □

Proof (of Theorem 9.2.5). Let $\mathcal{V} = \langle \mathcal{Q}, D\rangle$ be a 3-query linear verifier. Let $\mu = \Pr_{Q\sim_D \mathcal{Q}}[\text{supp}_{[n]}(Q) \neq \emptyset]$. Fix $\varepsilon > 0$. We prove the following bound:

$$s[\ell](\delta) \leq \min\left\{t\left[\frac{36\log\ell}{\varepsilon}\right](\delta) + \varepsilon + (1-\mu)\cdot\left(1 - \frac{1}{|\mathbb{F}|}\right), t\left[\frac{36\log\ell}{\varepsilon}\right](\delta) + \mu\cdot\left(1-\frac{1}{|\mathbb{F}|}\right)\right\}. \tag{9.1}$$

The right hand side attains its maximal value when

$$\mu = \frac{1}{2} + \frac{\varepsilon}{2\left(1 - \frac{1}{|\mathbb{F}|}\right)}.$$

Plugging this value of μ back into (9.1) completes the proof.

Now we argue (9.1). The first element on the right hand side of (9.1) is given by the following lemma which is proved in the next subsection.

Lemma 9.5.2. *Let $\mathcal{V} = \langle \mathcal{Q}, D\rangle$ be a $\mathbb{F}$-linear verifier for the $\mathbb{F}$-linear property $P \subseteq \mathbb{F}^n$ with soundness function s, let $\varepsilon > 0$ and let $\mu = \Pr_{Q\sim_D \mathcal{Q}}[\text{supp}_{[n]}(Q) \neq \emptyset]$. Then*

$$s(\delta) \leq t\left[\frac{36\log\ell}{\varepsilon}\right](\delta) + \varepsilon + (1-\mu)\cdot\left(1-\frac{1}{|\mathbb{F}|}\right).$$

To complete the proof we only need to show

$$s[\ell](\delta) \leq t\left[\frac{36\log\ell}{\varepsilon}\right](\delta) + \mu\cdot\left(1-\frac{1}{|\mathbb{F}|}\right). \tag{9.2}$$

Let w_0 be δ-far from P. By linearity, the all-zero proof $\pi_0 = \mathbf{0}$ is a legitimate proof (accompanying the zero codeword). Consider the soundness of $\mathcal{V}$ when presented with $w \circ \pi_0$ where w is the sum of w_0 and a random word $w' \in P$. Every query Q with $\mathrm{supp}_{[n]}(Q) = \emptyset$ is satisfied by the legitimate proof π_0. Additionally, every query Q with $\mathrm{supp}_{[n+1,n+\ell]}(Q) = \emptyset$ corresponds to a test, so the accumulated rejection probability of such tests is at most $t\left[\frac{36\log\ell}{\varepsilon}\right](\delta)$ because increasing query complexity does not decrease soundness. Finally, consider a query Q such that both $\mathrm{supp}_{[n]}(Q)$ and $\mathrm{supp}_{[n+1,n+\ell]}(Q)$ are not empty. By Claim 9.5.1 we may assume that $\mathcal{V}$ is nonredundant, so there is no $u \in P^*, u \neq 0$ such that $\mathrm{supp}(u) \subseteq \mathrm{supp}_{[n]}(Q)$. Since P is linear, by Lemma 9.4.6 for a random $w' \in P$ we know that $\langle Q, w' \rangle_{[n]}$ is a random element of $\mathbb{F}$. This implies that the rejection probability over such tests is at most $\mu \cdot (1 - 1/|\mathbb{F}|)$. This gives (9.2) and Theorem 9.2.5 follows. □

We end this subsection with the formal proof of Theorem 9.2.9.

Proof of Theorem 9.2.9. This follows from Lemma 9.5.2 by noticing that in the case of an inspective verifier we have $\mu = 1$. □

9.5.2 The Decomposition Lemma

In the proof of Lemma 9.5.2 we use the decomposition lemma of [LR99], stated next. The proof is included because we use a stronger version than the one appearing in [LR99, Tre05]. Our version deals with multigraphs while still bounding the radius of the decomposed graph as a function of the number of vertices. The proof follows along the lines of [LR99].

Before stating the lemma we need to introduce some notation. For any subset $V' \subseteq V$ of vertices of a multigraph G, let $G(V')$ denote the induced subgraph of G on the vertex set V'. Also, let $E(V') = E(G(V'))$. Similarly, let $E(V', V \setminus V')$ denote the set of edges between V' and $V \setminus V'$ (i.e., $E(V', V \setminus V') = E \cap (V' \times (V \setminus V'))$. For any connected graph G, define the *radius of* G as follows:

$$\mathrm{rad}(G) = \min_{v \in V} \max_{u \in V} d(u, v),$$

where $d(u, v)$ denotes the length of the shortest path between vertices u and v. Note that for any connected graph, the distance between any two vertices is at most twice the radius of the graph.

Lemma 9.5.3 (Decomposition [LR99])**.** *For every $\varepsilon \in (0, 1)$ and every multigraph $G = (V, E)$, there exists a subset of edges $E' \subseteq E$ of size at most $\varepsilon|E|$, such that every connected component of the graph $G_{\mathsf{Decomp}} = (V, E \setminus E')$ has radius strictly lesser than* $\log|V|/\varepsilon$. *The graph G_{Decomp} is said to be an* ε-decomposition *of G.*

Proof. Assume on the contrary that for some $\varepsilon > 0$, there exists a graph G which cannot be decomposed into components of radius less than $\log |V|/\varepsilon$ by removing at most an ε-fraction of the edges. Let G be such a graph with the minimum number of vertices.

Let v be a vertex of maximum degree in V. Hence, $\deg(v) \geq 2|E|/|V|$. Now, consider the set of vertices V' defined by the following sequence of operations. In the following, $\Gamma(V')$ denotes the neighborhood of V' (i.e., $\Gamma(V') = \{u \in V' | (u, v) \in E$ for some $v \in V'\}$).

Algorithm 3.

```
1. Set $V' \leftarrow \{v\} \cup \Gamma(v)$
2. While $|E(V', V \setminus V')| > \varepsilon |E(V')|$ do
        Set $V' \leftarrow V' \cup \Gamma(V')$
3. Output $V'$
```

Clearly, $|E(V', V \setminus V)| \leq \varepsilon |E(V')|$. Let t be the number of iterations of the while loop in the above procedure. Clearly, $t+1$ upper bounds the radius of the induced subgraph $G(V')$ because $d(v,u) \leq t+1$ for all $u \in G(V')$. Furthermore, each iteration of the while loop increases the number of edges in $G(V')$ by a multiplicative factor of at least $(1+\varepsilon)$. Hence,

$$|E(V')| > (1+\varepsilon)^t \deg(v) \geq (1+\varepsilon)^{(\mathrm{rad}(G(V'))-1)} \left(\frac{2|E|}{|V|}\right) \geq (1+\varepsilon)^{\mathrm{rad}(G(V'))} \cdot \frac{|E|}{|V|}$$

where in the last inequality we have used the fact $2 > (1+\varepsilon)$. However, since $E(V') \subseteq E$, we have that $\mathrm{rad}(G(V')) < \log |V| / \log(1+\varepsilon) < \log |V|/\varepsilon$. Here, we have used the fact that $\log_2(1+\varepsilon) > \varepsilon$ for all $\varepsilon \in (0,1)$.

Now, consider the induced subgraph $G' = G(V \setminus V')$. Since $|V \setminus V'| < |V|$, by the minimality condition we have that there exists a set of edges $E'' \subseteq E(G')$ of size at most $\varepsilon |E(G')|$, such that every connected component of the graph $G'_{\mathsf{Decomp}} = (V \setminus V', E(G') \setminus E'')$ has radius strictly less than $\log |V \setminus V'|/\varepsilon$.

Let $E' = E(V', V \setminus V') \cup E''$. We first observe that $|E'| \leq \varepsilon |E(V')| + \varepsilon |E(G')| \leq \varepsilon |E|$. Furthermore, the components of the graph $G_{\mathsf{Decomp}} = (V, E \setminus E')$ are $G(V')$ and the components of G'_{Decomp}. Hence, their radius is strictly less than $\log |V|/\varepsilon$. This contradicts the assumption that G is a counterexample to the lemma. $\square$

9.5.3 Proof of Lemma 9.5.2

Overview Given a verifier $\mathcal{V} = \langle \mathcal{Q}, D \rangle$, we construct a tester $\mathcal{V}' = \langle \mathcal{Q}', D \rangle$ with a one-to-one correspondence between the queries of $\mathcal{V}$ and those of $\mathcal{V}'$. The query complexity of $\mathcal{V}'$ is $O\left(\frac{\log \ell}{\varepsilon}\right)$. Additionally, we construct a set of proofs Π such that for every proof $\pi \in \Pi$, a $(1-\varepsilon)$-fraction of the inspective queries Q satisfy $\langle Q, w \circ \pi \rangle = \langle Q', w \circ \pi \rangle$, where

Q' is the test of $\mathcal{V}'$ corresponding to Q. Finally, we show that if π is a random proof from Π then the expected acceptance probability of a noninspective query is at least $1 - 1/|\mathbb{F}|$. Summing up, the differece between the rejection probability of the tester $\mathcal{V}'$ and that of the verifier $\mathcal{V}$ is at most $\varepsilon + (1 - 1/|\mathbb{F}|)(1 - \mu)$, completing our proof. The construction of $\mathcal{V}'$ and Π uses the $\mathbb{F}$-linearity of the constraints and the ε-decomposition of the inspective graph of $\mathcal{V}$ given in Lemma 9.5.3. We now focus on these two aspects.

Decomposed $\mathbb{F}$-linear verifiers Let $\mathcal{V}$ be an $\mathbb{F}$-linear verifier and let $G = G(\mathcal{V})$ be its inspective graph. Recall from Definition 36 that if $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| = 1$ then Q induces an edge between 0 and a vertex $i \in [n+1, n+\ell]$, whereas if $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| = 2$ then both vertices of the edge generated by Q lie in $[n+1, n+\ell]$ (if $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| \neq 1, 2$ then Q generates no edge).

Let G' be an ε-decomposition of G as per Lemma 9.5.3 with E' being the set of removed edges, $|E'| \leq \varepsilon|E|$. Let $V_0, V_1, \ldots, V_m$ be the set of connected components of G', where V_0 is the component to which the vertex 0 belongs. Let $F_0, \ldots, F_m$ be a set of corresponding spanning trees, one per component, of radius at most $\frac{\log \ell}{\varepsilon}$ each and let $F = \cup_j F_j$ (the existence of these trees is guaranteed by Lemma 9.5.3). Let $r_1, \ldots, r_m$ be arbitrary roots for $F_1, \ldots, F_m$ and set $r_0 = 0$ to be the root of F_0. To describe $\mathcal{V}'$ and Π we define two types of constraints that belong to $\mathrm{span}(\mathcal{Q})$. They are described next.

Vertex constraints For $i \in V_j \setminus \{r_j\}$ let $\mathcal{Q}(i)$ be the set of constraints that generate the edges along the unique path in F_j leading from r_j to i. Let $Q(i)$ be the unique nonzero vector in $\mathrm{span}(\mathcal{Q}(i))$ satisfying

$$(Q(i))_{i'} = \begin{cases} -1 & i' = i \\ 0 & i' \in [n+1, n+\ell] \setminus \{r_j, i\} \end{cases} \tag{9.3}$$

Such a constraint can be shown to exist by performing Gaussian elimination to remove the variables appearing in internal nodes $i_1, \ldots, i_t$ along the path from r_j to i. We call $Q(i)$ the *vertex constraint* corresponding to i.

Claim 9.5.4 (Basic properties of vertex constraint). *For $i \in V_j \setminus \{r_j\}$ we have*
(a) $\{i\} \subseteq \mathrm{supp}_{[n+1,n+\ell]}(Q(i)) \subseteq \{i, r_j\}$,
(b) $|\mathrm{supp}_{[n]}(Q(i))| \leq \frac{4 \log \ell}{\varepsilon}$ *and*
(c) $r_j \in \mathrm{supp}_{[n+1,n+\ell]}(Q(i))$ *if and only if* $j \neq 0$.

Proof. Part *(a)* follows by construction. Part *(b)* holds because a query Q that generates an edge has $\left|\mathrm{supp}_{[n]}(Q)\right| \leq 2$ and $Q(i)$ lies in the span of at most $\frac{2 \log \ell}{\varepsilon}$ constraints. Regarding part *(c)*, clearly $j = 0$ implies $r_j \notin \mathrm{supp}_{[n+1,n+\ell]}(Q(i))$ because 0 is not in the support of any query. For the other direction, if $j \neq 0$ note that every constraint has exactly two vertices in its support. Additionally, every internal vertex along the path from r_j to i, except i and r_j themselves, appears in the support of exactly two constraints. Thus, any $Q \in \operatorname{span}(\mathcal{Q}(i))$ satisfying (9.3) must have r_j in its support. $\square$

Edge constraints For an edge $e = (i, i') \in V_j \times V_j$ in G' generated by Q, let

$$\hat{Q}(e) = \begin{cases} Q + Q_i \cdot Q(i) & i' = r_j \\ Q + Q_{i'} \cdot Q(i') & i = r_j \\ Q + Q_i \cdot Q(i) + Q_{i'} \cdot Q(i') & i, i' \neq r_j \end{cases} \quad \text{and} \quad Q(e) = \begin{cases} \hat{Q}(e) & (\hat{Q}(e))_{r_j} = 0 \\ \frac{-\hat{Q}(e)}{(\hat{Q}(e))_{r_j}} & (\hat{Q}(e))_{r_j} \neq 0 \end{cases}.$$

In words, $Q(e)$ is the unique linear combination of Q and $Q(i), Q(i')$ (if one or both of the latter two are defined) that satisfies

$$Q(e)_{r_j} \in \{-1, 0\} \text{ and } Q(e)_{i''} = 0 \text{ for } i'' \in [n+1, n+\ell] \setminus \{r_j\}. \tag{9.4}$$

We call $Q(e)$ the *edge constraint* corresponding to e.

Claim 9.5.5. *For $e = (i, i') \in V_j \times V_j$ we have (a) $\operatorname{supp}_{[n+1,n+\ell]}(Q(e)) \subseteq \{r_j\}$, (b) $|\operatorname{supp}_{[n]}(Q)| \leq \frac{8 \log \ell}{\varepsilon}$ and (c) if $j = 0$ then $\operatorname{supp}_{[n+1,n+\ell]}(Q(e)) = \emptyset$.*

Proof. Let Q be the constraint that generates e and note that $\operatorname{supp}_{[n+1,n+\ell]}(Q) \subseteq \{i, i'\}$. For part *(a)* assume that $i \in \operatorname{supp}_{[n+1,n+\ell]}(Q)$. Recall from Claim 9.5.4 that $\operatorname{supp}_{[n+1,n+\ell]}(Q(i)) \subseteq \{r_j, i\}$ and $Q(i)_i = -1$. This implies that $\operatorname{supp}(Q + Q_i \cdot Q(i)) \subseteq \{i', r_j\}$. The case of i' is handled identically and this proves part *(a)*. Part *(b)* follows because $Q(e)$ lies in the span of at most $\frac{4 \log \ell}{\varepsilon}$ constraints and each constraint has $\left|\operatorname{supp}_{[n]}(Q)\right| \leq 2$. Part *(c)* follows from part *(a)* by observing that 0 is not in the support of any constraint. $\square$

Forced components The construction of the tester $\mathcal{V}'$ and the corresponding proofs Π depend on a partition of the components of G' into *forced* and *unforced* components, defined next.

Definition 38 (Forced component). *If $e \in V_j \times V_j$ satisfies $\operatorname{supp}_{[n+1,n+\ell]}(Q(e)) = \{r_j\}$ then we say that e* forces *V_j. If V_j contains an edge that forces it then we say that V_j is* forced. *Pick an arbitrary ordering of the edges and set the* designated *forcing edge of V_j to be the smallest edge that forces it. Components without a forcing edge are called* unforced.

Construction of the Tester $\mathcal{V}'$ We construct $\mathcal{V}' = \langle \mathcal{Q}', D \rangle$ from $\mathcal{V} = \langle \mathcal{Q}, D \rangle$ in three consecutive steps. Assume without loss of generality that $V_1, \ldots, V_k$ are the forced components of G'. (Notice that Claim 9.5.5*(c)* implies that V_0 is unforced.) First we convert each query Q into a query $Q^{(1)}$ with $\operatorname{supp}_{[n+1,n+\ell]}(Q^{(1)}) \subseteq \{r_1, \ldots, r_m\}$. Then we convert

$Q^{(1)}$ into a $Q^{(2)}$ with $\mathrm{supp}_{[n+1,n+\ell]}(Q^{(2)}) \subseteq \{r_{k+1}, \ldots, r_m\}$. Finally, we replace $Q^{(2)}$ by Q' with $\mathrm{supp}_{[n+1,n+\ell]}(Q') = \emptyset$, i.e., Q' is a test. All the time we keep the same distribution over tests, i.e., $D(Q') = D(Q^{(2)}) = D(Q^{(1)}) = D(Q)$. The detailed construction follows.

1. For every query Q set

$$Q^{(1)} = Q + \sum_{i \in [n+1,n+\ell] \setminus \{r_1,\ldots,r_m\}} Q_i \cdot Q(i).$$

2. For every query $Q^{(1)}$ set

$$Q^{(2)} = Q^{(1)} + \sum_{j=1}^{k} (Q^{(1)})_{r_j} \cdot Q(e_j).$$

3. For every query $Q^{(2)}$ set

$$Q' = \begin{cases} 0 & |\mathrm{supp}_{[n+1,n+\ell]}(Q^{(2)})| > 0 \\ Q^{(2)} & \text{otherwise} \end{cases}$$

Next we bound all of the important parameters of $\mathcal{V}'$ but for it's soundness function.

Claim 9.5.6 (Basic properties of $\mathcal{V}'$)**.** *$\mathcal{V}'$ is a tester with perfect completeness and query complexity $\leq \frac{36 \log \ell}{\varepsilon}$.*

Proof. $\mathcal{V}'$ is a tester because the last conversion step enforces $\mathrm{supp}(Q') \subseteq [n]$ for all $Q' \in \mathcal{Q}'$. Perfect completeness of $\mathcal{V}'$ follows from the perfect completeness of $\mathcal{V}$ by $\mathbb{F}$-linearity, because $\mathcal{Q}' \subseteq \mathrm{span}(\mathcal{Q})$.

Finally, the bound on the query complexity follows from Claims 9.5.4*(b)*, 9.5.5*(b)* by noting that Q' lies in the span of Q and at most 3 vertex constraints and 3 edge constraints. Indeed,

$$Q^{(1)} \in \mathrm{span}(Q, \left\{ Q(i) \mid i \in \mathrm{supp}_{[n+1,n+\ell]}(Q) \setminus \{r_1, \ldots, r_m\} \right\}),$$

and since $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| \leq 3$ we conclude that $Q^{(1)}$ is in the span of Q and at most 3 vertex constraints. By Claim 9.5.4*(a)* and Equation (9.4) we have

$$\mathrm{supp}_{[n+1,n+\ell]}(Q^{(1)}) \subseteq \left\{ r_j \mid \exists i \in \mathrm{supp}_{[n+1,n+\ell]}(Q) \cap V_j \right\},$$

so $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| \leq 3$ also implies $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q^{(1)})\right| \leq 3$. This implies that $Q^{(2)}$ lies in the span of $Q^{(1)}$ and at most 3 edge constraints and our proof is complete. $\square$

Construction of the proof-set Π To argue the soundness of $\mathcal{V}'$ we introduce a family of proofs designed to fool inspective verifiers.

Definition 39. *Let $V_1, \ldots, V_k$ be the forced components of G' and let $e_1, \ldots, e_k$ be their respective designated forcing edges. A proof π is called* F-compliant for w *if $w \circ \pi$ satisfies every constraint that generates an edge in $F \cup \{e_1, \ldots, e_k\}$. Let $\Pi = \Pi(w)$ denote the set of F-compliant proofs for w.*

The next claim shows that F-compliant proofs exist for any word and describes the structure of these proofs. This structure will be used to analyze the soundness of $\mathcal{V}'$.

Claim 9.5.7. *For every $w \in \mathbb{F}^n$ and $\alpha_{k+1}, \ldots, \alpha_m \in \mathbb{F}$ there exists a unique F-compliant proof for w such that $\pi_{r_j} = \alpha_j$ for $k < j \leq m$.*

Proof. The set of constraints that generate the edges of F, denoted $\mathcal{Q}(F)$, is linearly independent and any setting of values for $\pi_{r_1}, \ldots, \pi_{r_j}$ can be extended in a unique way to a proof that satisfies $\mathcal{Q}(F)$ (this can be proved by induction along paths in F; we omit the full details).

To complete the proof we have to argue uniqueness. To do so we show that all F-compliant proofs assign the same values to $\pi_i, i \notin V_1 \cup \ldots \cup V_k$

First, consider V_0, the special component whose root is 0. Let $e = (0, i) \in F_0$ be generated by Q. There is a unique setting of π_i that satisfies Q because $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| = 1$. Once all vertices at distance 1 from 0 have been fixed, there is a unique assignment to $\pi_i, i \in V_0$ that satisfies $\mathcal{Q}(F_0)$ – the set of constraints that generate edges in F_0.

Next, consider $e = (i, i')$ – generated by Q – that is the designated forcing edge of V_j. By definition 38 we have $\mathrm{supp}_{[n+1,n+\ell]}(Q(e)) = \{r_j\}$, so there is a unique setting for π_{r_j} that satisfies Q. By the linear independence of $\mathcal{Q}(F_j)$ this can be extended to an assignment to $\pi_i, i \in V_j$ that satisfies $\mathcal{Q}(F_j)$. This completes the proof. □

F-compliant proofs are important because on certain types of queries the output of Q on $w \circ \pi$ is equal to the output of the test Q' performed on w. This is argued in our next claim.

Claim 9.5.8. *If π is F-compliant for w and $Q \in \mathcal{Q}$ has one of the following properties*

1. $\mathrm{supp}_{[n+1,n+\ell]}(Q) = \emptyset$, *or*

2. *Every $i \in \mathrm{supp}_{[n+1,n+\ell]}(Q)$ belongs to a forced component, or*

3. *Q generates an edge $e \in E \setminus E'$.*

Then

$$\langle Q', w \circ \pi \rangle = \langle Q, w \circ \pi \rangle.$$

Proof. We prove each case separately.

1. By construction $Q' = Q^{(2)} = Q^{(1)} = Q$ and the claim follows.

2. By assumption and Claim 9.5.4*(a)* we have $\mathrm{supp}_{[n+1,n+\ell]}(Q^{(1)}) \subseteq \{r_1, \ldots, r_k\}$. Suppose that $r_j \in \mathrm{supp}_{[n+1,n+\ell]}(Q^{(1)})$. Definition 38 and Equation (9.4) imply $(Q(e_j))_{r_j} = -1$, so by construction $r_j \notin \mathrm{supp}_{[n+1,n+\ell]}(Q^{(2)})$. This is argued for each $r_j \in \mathrm{supp}_{[n+1,n+\ell]}(Q^{(1)})$ and shows $\mathrm{supp}_{[n+1,n+\ell]}(Q^{(2)}) = \emptyset$. By construction this implies $Q' = Q^{(2)}$. Note that $Q^{(2)} = Q + Q''$ where Q'' is a linear combination of constraints that generate edges in $F \cup \{e_1, \ldots, e_k\}$. We conclude that

$$\langle Q', w \circ \pi \rangle = \langle Q^{(2)}, w \circ \pi \rangle = \langle Q, w \circ \pi \rangle + \langle Q'', w \circ \pi \rangle = \langle Q, w \circ \pi \rangle, \tag{9.5}$$

The last equality follows because π is F compliant for w.

3. We may assume that e belongs to a component V_j that is not forced, because the other case (of forced V_j) was argued in part 2. By construction $Q^{(1)} = Q(e)$. By assumption e does not force V_j, so by Definition 38 we have $\mathrm{supp}_{[n+1,n+\ell]}(Q(e)) = \emptyset$. By construction $Q' = Q^{(2)} = Q^{(1)}$ and the F-compliancy of π implies as argued in Equation (9.5) that $\langle Q', w \circ \pi \rangle = \langle Q^{(2)}, w \circ \pi \rangle = \langle Q, w \circ \pi \rangle$. This completes the proof.

□

We are ready to argue the soundness of $\mathcal{V}'$ and complete the proof of Lemma 9.5.2.

Claim 9.5.9 (Soundness). *Let* $\sigma = \Pr_{Q \sim_D \mathcal{Q}}[\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| = 3]$. *There exists an F-compliant proof π such that*

$$\Pr[\mathcal{V}'^{w \circ \pi} = \mathsf{reject}] \geq \Pr[\mathcal{V}^{w \sqcup \pi} = \mathsf{reject}] - \varepsilon - (1 - 1/|\mathbb{F}|) \cdot \sigma.$$

Proof. If π is F-compliant for w then by Claim 9.5.8 the output of $\mathcal{V}$ and $\mathcal{V}'$ on $w \circ \pi$ may differ only if the query performed is one of two types. The first type is a query that generates an edge $e \in E'$. The fraction of these queries is at most ε. The second type is a query with $\left|\mathrm{supp}_{[n+1,n+\ell]}(Q)\right| = 3$ where there exists $i \in \mathrm{supp}_{[n+1,n+\ell]}(Q)$ such that i belongs to an unforced component V_j. Let σ' denote the fraction of queries of the second type and note that $\sigma' \leq \sigma$. We can already conclude

$$\Pr[\mathcal{V}'^{w \circ \pi} = \mathsf{reject}] \geq \Pr[\mathcal{V}^{w \sqcup \pi} = \mathsf{reject}] - \varepsilon - \sigma,$$

but to reach the stronger claim stated above we need one additional observation regarding constraints of the second type.

Let Q be such a constraint and suppose that $i \in \mathrm{supp}_{[n+1,n+\ell]}(Q)$ belongs to the unforced component V_j. Consider the uniform distribution over F-compliant proofs obtained by randomly fixing values $\alpha_{k+1}, \ldots, \alpha_m$ for $\pi_{r_{k+1}}, \ldots, \pi_{r_m}$ and extending these values to an F-compliant proof for w. Notice the value assigned to π_i depends linearly on the value of π_{r_j}. Thus, assigning a uniformly random value to π_{r_j} implies that $\langle Q, w \circ \pi \rangle$ is a random variable ranging uniformly over $\mathbb{F}$, i.e. Q accepts $w \circ \pi$ with probability $1/|\mathbb{F}|$. This implies the expected number of constraints of the second type that are satisfied is $1/|\mathbb{F}|$. We conclude the existence of an F-compliant proof which is rejected by at most a $(1 - 1/|\mathbb{F}|)$-fraction of the queries of the second type. This completes our proof. □

Proof of Lemma 9.5.2. Let w be δ-far from P. Let $\mathcal{V}'$ be the tester constructed from $\mathcal{V}$ as described earlier in this subsection. Let π be the F-compliant proof for w satisfying Claim 9.5.9. Note that $\sigma \leq 1 - \mu$ so this claim implies

$$s(\delta) \leq \Pr[\mathcal{V}^{w \circ \pi} = \mathsf{reject}] \leq \Pr[\mathcal{V}'^{w \circ \pi} = \mathsf{reject}] + \varepsilon + (1 - 1/|\mathbb{F}|)(1 - \mu).$$

The proof is completed by recalling from Claim 9.5.6 that $\mathcal{V}'$ is a $\left(\frac{36 \log \ell}{\varepsilon}\right)$-tester, and hence $\Pr[\mathcal{V}'^{w \circ \pi} = \mathsf{reject}] \leq t\left[\frac{36 \log \ell}{\varepsilon}\right](\delta)$. □

Part IV

Bibliography

Bibliography

[Alon02] N. Alon, Testing subgraphs in large graphs, *Random Structures and Algorithms* 21 (2002), 359–370.

[ADL$^+$94] N. Alon, R. A. Duke, H. Lefmann, V. Rödl and R. Yuster, The algorithmic aspects of the regularity lemma, J. of Algorithms 16 (1994), 80-109.

[AFKS00] N. Alon, E. Fischer, M. Krivelevich and M. Szegedy, Efficient testing of large graphs, *Combinatorica* 20 (2000), 451–476.

[AFNS06] N. Alon, E. Fischer, I. Newman and A. Shapira, A Combinatorial Characterization of the Testable Graph Properties: It's All About Regularity, *Proceedings of the* 38th *ACM STOC* (2006), 251–260.

[AKKR06] N. Alon, T. Kaufman, M. Krivilevich, and D. Ron, Testing triangle-freeness in general graphs, *Proceedings of the* 17th *SODA*, (2006), 279–288.

[AN06] N. Alon and A. Naor, Approximating the cut-norm via Grothendieck's inequality, SIAM J. on Computing 35 (2006), 787-803.

[AS05] N. Alon and A. Shapira, A Characterization of the (natural) Graph Properties Testable with One-Sided Error, *Proceedings of the* 46th *IEEE FOCS* (2005), 429–438, Also *SIAM Journal on Computing*, to appear.

[AS05b] N. Alon and A. Shapira, Every monotone graph property is testable, *Proceedings of the* 37th *ACM STOC* (2005), 128–137, Also *SIAM Journal on Computing*, to appear.

[AS00] N. Alon and J. H. Spencer, *The probabilistic method.* Wiley-Interscience (John Wiley & Sons), New York, 1992 (1st edition) and 2000 (2nd edition).

[AVKK03] N. Alon, W. F. de la Vega, R. Kannan and M. Karpinski, Random sampling and approximation of MAX-CSP problems, Proc. of the 34 ACM STOC, ACM Press (2002), 232-239. Also: JCSS 67 (2003), 212-243.

[AFK96] S. Arora, A. Frieze and H. Kaplan, A new rounding procedure for the assignment problem with applications to dense graph arangement problems. *Mathematical Programming* 92:1 (2002), 1–36. Preliminary version in the 37th *Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press* (1996), 21–30.

[AKK99] S. Arora, D. R. Karger and M. Karpinski, Polynomial time approximation schemes for dense instances of NP-Hard problems, J. Comput. Syst. Sci. 58 (1999), 193-210.

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[AS98] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[Bab06] L. Babai, On the diameter of Eulerian orientations of graphs, *Proceedings of the* 17th *SODA*, (2006), 822–831.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin and M. Szegedy. Checking computations in polylogarithmic time. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, New York, NY, USA, 1991. ACM Press.

[BFF$^+$01] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld and P. White, Testing random variables for independence and identity, *Proceedings of the* 42nd *IEEE FOCS* (2001), 442–451.

[BCH$^+$95] M. Bellare, D. Coppersmith, J. Håstad, M. A. Kiwi and M. Sudan. Linearity testing in characteristic two. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 432–441, Washington, DC, USA, 1995. IEEE Computer Society.

[BSGH$^+$04] 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 Theory of Computing (STOC-04)*, pages 1–10, New York, June 13–15 2004. ACM Press.

[BSGS03] E. Ben-Sasson, O. Goldreich and M. Sudan. Bounds on 2-query codeword testing. In *RANDOM-APPROX 2003*, pages 216–227, 2003.

[BHLM08] E. Ben-Sasson, P. Harsha, O. Lachish and A. Matsliah, Sound 3-query PCPPs are Long, manuscript.

[BSHR05] E. Ben-Sasson, P. Harsha and S. Raskhodnikova, Some 3CNF properties are hard to test, *SIAM Journal on Computing*:35(1), 1–21, 2005 (a preliminary version appeared in Proc. 35th STOC, 2003).

[BSS05] E. Ben-Sasson and M. Sudan. Short PCPs with poly-log rate and query complexity. In *STOC*, pages 266–275, 2005.

[BSSVW03] E. Ben-Sasson, M. Sudan, S. P. Vadhan and A. Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *STOC*, pages 612–621, 2003.

[Bog05] A. Bogdanov. Gap amplification fails below 1/2. In *ECCC: Electronic Colloquium on Computational Complexity, technical reports*, 2005.

[BLR90] M. Blum, M. Luby and R. Rubinfeld, Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences* 47 (1993), 549–595 (a preliminary version appeared in Proc. 22nd STOC, 1990).

[BW05] G. R. Brightwell and P. Winkler, Counting Eulerian circuits is #P-complete, *Proc. 7th ALENEX and 2nd ANALCO 2005 (Vancouver BC)*, C. Demetrescu, R. Sedgewick and R. Tamassia, eds, SIAM Press, 259–262.

[CFL$^+$07] S. Chakraborty, E. Fischer, O. Lachish, A. Matsliah and I. Newman: Testing *st*-Connectivity, *Proceedings of the* 11th *RANDOM and the* 10th *APPROX (2007)*: 380–394.

[CMM06] M. Charikar, K. Makarychev and Y. Makarychev. Near-optimal algorithms for unique games. In Jon M. Kleinberg, editor, *STOC*, pages 205–214. ACM, 2006.

[Chu91] F.R.K. Chung, Regularity lemmas for hypergraphs and quasi-randomness, Random Structures and Algorithms, 2 (1991), 241-252.

[CS05] A. Czumaj and C. Sohler, Testing hypergraph colorability, Theor. Comput. Sci. 331 (2005), 37-52.

[CR00] A. Czygrinow and V. Rödl, An algorithmic regularity lemma for hypergraphs, SIAM Journal on Computing, 30 (2000), 1041-1066.

[Din07] I. Dinur. The PCP theorem by gap amplification. volume 54, page 12, 2007.

[DR04] I. Dinur and O. Reingold. Assignment testers: Towards a combinatorial proof of the PCP-theorem. In *FOCS*, pages 155–164, 2004.

[DLR95] R. Duke, H. Lefman and V. Rödl, A fast approximation algorithm for computing the frequencies of subgraphs in a given graph, SIAM J. on Computing 24 (1995) 598-620.

[EH05] L. Engebretsen and J. Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. In Volker Diekert and Bruno Durand, editors, *STACS*, volume 3404 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2005.

[FK95] U. Feige and J. Kilian. Impossibility results for recycling random bits in two-prover proof systems. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing, STOC'95 (Las Vegas, Nevada, May 29 - June 1, 1995)*, pages 457–468, New York, 1995. ACM Press.

[dlV96] W. Fernandez de la Vega, Max-Cut has a randomized approximation scheme in dense graphs, Random Struct. Algorithms 8 (1996), 187-198.

[Fis04] E. Fischer, The art of uninformed decisions: A primer to property testing, *Current Trends in Theoretical Computer Science: The Challenge of the New Century, G. Paun, G. Rozenberg and A. Salomaa (editors), World Scientific Publishing* (2004), Vol. I 229-264.

[Fis05] E. Fischer, The difficulty of testing for isomorphism against a graph that is given in advance, *SIAM Journal on Computing* 34 (2005), 1147-1158.

[FF05] E. Fischer and L. Fortnow. Tolerant versus intolerant testing for boolean properties. In *IEEE Conference on Computational Complexity*, pages 135–140. IEEE Computer Society, 2005.

[FM06] E. Fischer and A. Matsliah, Testing graph isomorphism, *Proceedings of the* 17th *ACM-SIAM SODA* (2006), 299–308.

[FMNY08] E. Fischer, A. Matsliah, I. Newman and O. Yahalom, On the Query Complexity of Testing for Eulerian Orientations, *To appear in RANDOM*, (2008).

[FMS07] E. Fischer, A. Matsliah and A. Shapira, Approximate Hypergraph Partitioning and Applications, *FOCS* (2007).

[FN05] E. Fischer and I. Newman, Testing versus estimation of graph properties, *Proceedings of the* 37th *ACM STOC* (2005), Also *SIAM Journal on Computing*, to appear.

[FNS04] E. Fischer, I. Newman and J. Sgall, Functions that have read-twice constant width branching programs are not necessarily testable, *Random Structures and Algorithms* 24 (2004), 175–193.

[FY07] E. Fischer and O. Yahalom, Testing convexity properties of tree colorings, *Proc. of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, LNCS 4393, Springer-Verlag 2007, 109–120.

[Fle90] H. Fleishcner, *Eulerian graphs and related topics*, Part 1. Vol. 1. Annals of Discrete Mathematics 45, 1990.

[Fle91] H. Fleishcner, *Eulerian graphs and related topics*, Part 1. Vol. 2. Annals of Discrete Mathematics 50, 1991.

[FK99b] A. Frieze and R. Kannan, A simple algorithm for constructing Szemerédi's regularity partition, Electr. J. Comb. 6: (1999).

[FK96] A. Frieze and R. Kannan, The regularity lemma and approximation schemes for dense problems, Proc. of FOCS 1996, 12-20.

[FK99] A. Frieze and R. Kannan, Quick approximation to matrices and applications, *Combinatorica* 19 (1999), 175-220.

[Gal62] R. G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[GGR98] O. Goldreich, S. Goldwasser and D. Ron, Property testing and its connection to learning and approximation, *Journal of the ACM* 45 (1998), 653–750 (a preliminary version appeared in Proc. 37^{th} FOCS, 1996).

[GR02] O. Goldreich and D. Ron, Property testing in bounded degree graphs, *Algorithmica*, (2002), 32(2):302–343

[GS02] O. Goldreich and M. Sudan. Locally testable codes and pcps of almost-linear length. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 13–22, Washington, DC, USA, 2002. IEEE Computer Society.

[GT03] O. Goldreich and L. Trevisan, Three theorems regarding testing graph properties, *Random Structures and Algorithms* 23 (2003), 23–57.

[Gow97] T. Gowers, Lower bounds of tower type for Szemerédi's uniformity lemma, GAFA 7 (1997), 322-337.

[Gow06] T. Gowers, Hypergraph regularity and the multidimensional Szemerédi theorem, manuscript, 2006.

[GLST98] V. Guruswami, D. Lewin, M. Sudan and L. Trevisan. A tight characterization of np with 3 query pcps. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS-98)*, pages 8–17, Los Alamitos, CA, 1998. IEEE Computer Society.

[GR05] V. Guruswami and A. Rudra. Tolerant locally testable codes. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *APPROX-RANDOM*, volume 3624 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 2005.

[GT06] A. Gupta and K. Talwar. Approximating unique games. In *SODA*, pages 99–106. ACM Press, 2006.

[HS92] P. Hajnal and M. Szegedy, On packing bipartite graphs. *Combinatorica* 12 (1992), 295–301.

[HLNT07] S. Halevy, O. Lachish, I. Newman and D. Tsur, Testing Properties of Constraint-Graphs, *Proceedings of the* 22^{nd} *IEEE Annual Conference on Computational Complexity (CCC 2007).*

[HLNT05] S. Halevy, O. Lachish, I. Newman and D. Tsur, Testing Orientation Properties, *Electronic Colloquium on Computational Complexity (ECCC)* , (2005), 153.

[HS01] P. Harsha and M. Sudan. Small PCPs with low query complexity. In Afonso Ferreira and Horst Reichel, editors, *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science, STACS'2001 (Dresden, Germany, February 15-17, 2001)*, volume 2010 of *LNCS*, pages 327–338. Springer-Verlag, Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan-Paris-Singapore-Tokyo, 2001.

[Hås97] J. Håstad. Some optimal inapproximability results. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1997. ACM Press.

[HK01] J. Håstad and S. Khot. Query efficient PCPs with perfect completeness. In Bob Werner, editor, *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS-01)*, pages 610–619, Los Alamitos, CA, October 14–17 2001. IEEE Computer Society.

[HNR05] P. Haxell, B. Nagle and V. Rödl, An algorithmic version of the hypergraph regularity method, Proc of FOCS 2005, 439-448.

[IKN88] T. Ibaraki, A. V. Karzanov and H. Nagamochi, A fast algorithm for finding a maximum free multiflow in an inner Eulerian network and some generalizations, *Combinatorica 18(1)* (1988), 61–83.

[KT00] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In ACM, editor, *Proceedings of the thirty second annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 80–86, pub-ACM:adr, 2000. ACM Press.

[KKR04] T. Kaufman, M. Krivelevich and D. Ron, Tight bounds for testing bipartiteness in general graphs, *SICOMP* , (2004), 33(6):1441–1483.

[KdW03] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *STOC*, pages 106–115. ACM, 2003.

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, New York, NY, USA, 2002. ACM Press.

[KS06] S. Khot and R. Saket. A 3-query non-adaptive PCP with perfect completeness. In *IEEE Conference on Computational Complexity*, pages 159–169. IEEE Computer Society, 2006.

[Kil92] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 723–732, Victoria, British Columbia, Canada, 4–6 May 1992.

[KRS02] Y. Kohayakawa, V. Rödl and J. Skokan, Hypergraphs, quasi-randomness, and conditions for regularity, J. of Combinatorial Theory A, 97 (2002), 307-352.

[KRT03] Y. Kohayakawa, V. Rödl and L. Thoma, An optimal algorithm for checking regularity, SIAM J. on Computing 32 (2003), no. 5, 1210-1235.

[KS96] J. Komlós and M. Simonovits, Szemerédi's regularity lemma and its applications in graph theory. In: *Combinatorics, Paul Erdős is Eighty*, Vol II (D. Miklós, V. T. Sós, T. Szönyi eds.), János Bolyai Math. Soc., Budapest (1996), 295–352.

[Lan04] M. Langberg, Testing the independence number of hypergraphs, Proc. of RANDOM 2004, 405-416.

[LR99] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[Lov76] L. Lovász, On some connectivity properties of Eulerian graphs, *Acta Math. Hung. 28* (1976) 129–138.

[Luks82] E. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences* 25 (1982), 42–65.

[Mic00] S. Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, August 2000.

[MW96] M. Mihail, P. Winkler, On the number of Eulerian orientations of a graph, *Algorithmica 16(4/5)* (1996), 402–414.

[MR06] D. Moshkovitz and R. Raz. Sub-constant error low degree test of almost-linear size. In Jon M. Kleinberg, editor, *STOC*, pages 21–30. ACM, 2006.

[MR07] D. Moshkovitz and R. Raz. Sub-constant error probabilistically checkable proof of almost linear size. *Electronic Colloquium on Computational Complexity (ECCC)*, (026), 2007.

[NRS06] B. Nagle, V. Rödl and M. Schacht, The counting lemma for regular k-uniform hypergraphs, Random Structures and Algorithms 28 (2006), 113-179.

[New02] I. Newman, Testing Membership in Languages that Have Small Width Branching Programs, *SIAM Journal on Computing* 31(5):1557–1570, 2002.

[PR02] M. Parnas and D. Ron, Testing the diameter of graphs, *Random Structures and Algorithms*, (2002), 20(2):165–183.

[PRR06] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci*, 72(6):1012–1042, 2006.

[PTW01] P. A. Pevzner, H. Tang and M. S. Waterman, An Eulerian path approach to DNA fragment assembly, *Proc. Natl. Acad. Sci. USA 98*, (2001), 9748–9753.

[PS94] A. Polishchuk and D. A. Spielman. Nearly-linear size holographic proofs. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 194–203, New York, NY, USA, 1994. ACM Press.

[Raz98] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998.

[Rob69] R. W, Robinson, Enumeration of Euler graphs, In *Proof Techniques in Graph Theory* (Ed. F. Harary), New York: Academic Press, 1969, 147–153.

[Ron01] D. Ron, Property testing (a tutorial), In: *Handbook of Randomized Computing* (S. Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolim eds), Kluwer Press (2001), Vol. II Chapter 15.

[RS93] R. Rubinfeld and M. Sudan, Robust characterization of polynomials with applications to program testing, *SIAM Journal on Computing* 25 (1996), 252–271 (first appeared as a technical report, Cornell University, 1993).

[ST00] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, STOC'2000 (Portland, Oregon, May 21-23, 2000)*, pages 191–199, New York, 2000. ACM Press.

[ST06] A. Samorodnitsky and L. Trevisan. Gowers uniformity, influence of variables, and PCPs. In Jon M. Kleinberg, editor, *STOC*, pages 11–20. ACM, 2006.

[Sze99] M. Szegedy. Many-valued logics and holographic proofs. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP*, volume 1644 of *Lecture Notes in Computer Science*, pages 676–686. Springer, 1999.

[Sze75] E. Szemerédi, Integer sets containing no k elements in arithmetic progression, Acta Arith. 27 (1975), 299-345.

[Sze78] E. Szemerédi, Regular partitions of graphs, In: *Proc. Colloque Inter. CNRS* (J. C. Bermond, J. C. Fournier, M. Las Vergnas and D. Sotteau, eds.), 1978, 399–401.

[Tao06] T. Tao, A variant of the hypergraph removal lemma, J. Combin. Theory, Ser. A 113 (2006), 1257-1280.

[Tre05] L. Trevisan. Approximation algorithms for unique games. In *FOCS*, pages 197–205. IEEE Computer Society, 2005.

[Tut84] W. T. Tutte, *Graph theory*, Addison-Wesley, New York, 1984.

[Yao77] A. C. Yao, Probabilistic computation, towards a unified measure of complexity. *Proceedings of the* 18^{th} *IEEE FOCS* (1977), 222–227.

[Zwi98] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'98 (San Francisco, California, January 25-27, 1998)*, pages 201–210, Philadelphia, PA, 1998. ACM SIGACT, SIAM, Society for Industrial and Applied Mathematics.

שאילתות קבוע של כיווני קשתות האם קיים מסלול מכוון מ-s ל-t תחת האוריינטציה של הגרף. בעיה זו נוסחה כבעיה פתוחה באחד מהמאמרים הראשונים בנושא של המחברים הלוי, לכיש, נוימן וצור (Halevy, Lachish, Newman, Tzur, ECCC 2005). בתזה הזו אנו עונים על השאלה הנ"ל בחיוב. ליתר דיוק, אנו מראים אלגוריתם בדיקה במספר שאילתות קבוע ועם שגיאה חד צדדית, אשר יודע להבדיל בין המקרה בו קיים מסלול מכוון מ-s ל-t לבין המקרה שבו האוריינטציה רחוקה מלהכיל מסלול כזה.

התוצאה השנייה בנושא בדיקת אוריינטציות עוסקת בבדיקת אוילריאניות (ז"א את קיומו של מסלול Euler) של גרפים מכוונים. למרות שתכונה זו מתאפיינת על ידי תנאים לוקאלים (גרף הוא אוילריאני אם ורק אם בכל צמתיו דרגת הכניסה שווה לדרגת היציאה) אנו מראים שלמעשה תכונה זו לא ניתנת לבדיקה במספר שאילתות קבוע. עובדה זו נכונה גם עבור בודקים עם שגיאה דו-צדדית, ואפילו אם גרף התשתית הוא גרף הטורוס (Toroidal grid), מה שמבטיח שישנם עדים מגודל קבוע במקרה שהגרף לא אוילריאני.

חלק ג' – הוכחות קרוב הניתנות לווידוא הסתברותי

החלק האחרון בתזה עוסק בהוכחות קרוב הניתנות לווידוא הסתברותי (Probabilistically Checkable Proof of Proximity – PCPP). בודק הוכחות קרוב עם ווידוא הסתברותי פועל בדומה לבודק תכונות, אלא שכאן מותר לו להסתייע בהוכחת עזר הניתנת בנוסף לקלט, ועליו לקבל קלט אשר מקיים את התכונה אם הוכחה זו היא "נכונה". קלטים רחוקים מהתכונה עדיין חייבים להדחות על ידי הבודק בהסתברות גבוהה, ללא קשר לתוכן ההוכחה הנוספת.

המיקוד שלנו הוא על הקשר בין הסתברות השגיאה של בודק כנ"ל, לבין אורך ההוכחה שאליה יש לו גישה. התוצאות שלנו מראות שאם דורשים שלבודק תהיה הסתברות שגיאה הקרובה לאופטימום – אזי להוכחה חייב להיות אורך אקספוננציאלי. אחד המרכיבים המרכזיים בהוכחת משפט זה הוא קשר הדוק בין מספר השאילתות הדרוש עבור בדיקת תכונות לבין אורך ההוכחה המינימאלי (במובן של הוכחות קרוב הניתנות לווידוא) של קודים ליניאריים מסוימים.

חלק א' – בדיקת תכונות במודל הצפוף

במודל הגרפים וההיפרגרפים הצפוף מתעניינים בתכונות של גרפים והיפרגרפים שלהם מספר קשתות פרופורציונאלי למספר המקסימאלי שיכול להיות. למשל, עבור גרפים עם n צמתים מדובר על $\Omega(n^2)$ קשתות, ועבור היפרגרפים r-יוניפורמים עם n-צמתים מדובר על $\Omega(n^r)$ היפר-קשתות. הקלט במודל זה הוא מטריצת השכנויות של הגרף (או טנסור השכנויות של ההיפרגרף), והשאילתה שהבודק יכול לשאול היא האם זוג נתון של צמתים (או סדרה מאורך r במקרה של היפרגרף) מהווה קשת או לא.

השאלה הראשונה בה אנו עוסקים היא בדיקת איזומורפיזם של גרפים. בהינתן שני גרפים פשוטים, אנו שואלים מהי סיבוכיות השאילתות של הבדיקה של תכונת היות הגרפים הנ"ל איזומורפים. אנו חוקרים גם בודקי תכונות עם שגיאה חד צדדית וגם עם שגיאה דו צדדית. אנחנו מטפלים בשני המקרים הבאים: המקרה הראשון שבו גרף אחד ידוע מראש, והשאילתות שהבודק עושה הן רק על הגרף הלא ידוע; והמקרה השני שבו שני הגרפים לא ידועים, והבודק חייב לשאול שאילתות על שניהם. אנו מראים חסמים תחתונים ועליונים כמעט הדוקים לשלוש מתוך ארבע הקומבינציות האפשריות.

השאלה השנייה שבה אנו עוסקים במודל הצפוף קשורה לבעיות חלוקה של היפרגרפים ושימושיהם. אנו מראים כי כל בעיית חלוקה כלליות של היפרגרפים צפופים ניתנת לפתרון בזמן ליניארי במספר הצמתים (שהוא תת-ליניארי בגודל הקלט), וכן אנו מראים כי תכונת הקיום של חלוקה כזו ניתנת לבדיקה במספר שאילתות שהוא קבוע (אינו תלוי בגודל הקלט). תוצאה זו מכלילה את עבודתם המייסדת של גולדרייך, גולדוואסר ורון (Goldreich, Goldwasser, Ron, JACM 1998), אשר הראו תוצאה דומה עבור בעיות חלוקה של גרפים פשוטים. בנוסף לכך אנו מראים שימוש של אלגוריתם החלוקה הנ"ל למציאת חלוקות רגולריות קטנות של גרפים בזמן תת-ליניארי, במובן של למת הרגולריות של סמרדי (Szemérédi). נראה גם שחלוקות רגולריות (במובן החלש) של היפרגרפים ניתנות למציאה באותו אופן, ובזמן קצר יותר ממה שהיה ידוע עד כה. שימושים נוספים של אלגוריתם החלוקה שלנו נובעים באופן טבעי, ובהם בדיקת תכונות של היפרגרפים צפופים, מציאת השמה שממקסמת בקירוב את מספר הפסוקיות המסופקות בנוסחת 3CNF, וגם איחוד של מספר תוצאות קודמות בתחום בדיקת התכונות.

חלק ב' – בדיקת תכונות במודל האוריינטציה

במודל האוריינטציה הקלטים הם אוריינטציות של גרף תשתית הנתון לבודק מראש. ניתן לחשוב על גרף התשתית כעל פרמטר קבוע של הבעיה, זאת אומרת המרחק של הקלט לתכונה נמדד כמספר השינויים שצריך לעשות באוריינטציה של הקשתות הנתונות מראש, בעוד שאין לבצע כל הוספה או הורדה של הקשתות עצמן.

בנושא של בדיקת תכונות במודל הנ"ל אנחנו מתרכזים בשתי בעיות בסיסיות. הראשונה היא בעיית ה-st קשירות: בהינתן גרף לא מכוון ושני צמתים s,t, עלינו לבדוק במספר

תקציר

מידת היעילות של חישוב פתרון לבעיה כלשהי מתבטאת ביחס שבין כמות הנתונים (אורך הקלט) לבין הזמן הדרוש לעיבוד שיש לבצע על הנתונים הללו. מובן שככל שהיקף העיבוד יהיה קטן יותר, החישוב המוצע ייחשב יעיל יותר. בעבר הלא רחוק, חישובים שמסתפקים בעיבוד שזמן ריצתו ליניארי באורך הקלט נחשבו היעילים ביותר (למעשה, אפילו זמן עיבוד פולינומי באורך הקלט יכול היה להיחשב יעיל גם כן). עם זאת, ישנן כיום בעיות מעשיות שעבורם אפילו זמן חישוב ליניארי באורך הקלט אינו יכול להיחשב יעיל דיו. במקרים אלה פתרון המבוסס על זמן עיבוד הקטן מאורך הקלט עשוי להיחשב לאופציה יחידה עבור בעיות מסוג זה. ברור שאלגוריתם כזה חייב לתת שערוך של הקלט מבלי שהספיק לקרוא את כולו.

בדיקת תכונות הוא תחום חדש יחסית במדעי המחשב העוסק באלגוריתמים הסתברותיים שזמן הריצה שלהם, או לפחות מספר הביטים שהם קוראים מהקלט, קטן משמעותית מאורך הקלט (תת-ליניארי). אלגוריתמים לבדיקת תכונות אינם יכולים לספק תשובה מדויקת ביחס לקיום התכונה הנבדקת, שכן הם אינם קוראים את הקלט במלואו (בחלק מהמקרים מדובר באלגוריתמים הקוראים מהקלט מספר קבוע של ביטים בלבד), אולם בהרבה מקרים ניתן למצוא אלגוריתמים הנותנים תשובה מקורבת. ליתר דיוק, בודק תכונות חייב להבדיל בין המקרה בו הקלט מספק תכונה כלשהי, לבין המקרה שבו הקלט רחוק מלספק את התכונה הנ"ל. לרוב המרחק נמדד כמספר הביטים המינימאלי שצריך לשנות בקלט על מנת להביאו לקלט אשר מקיים את התכונה. את זאת חייב בודק התכונות לספק באמצעות מדגם חלקי בלבד של הקלט, באופן הסתברותי.

תזה זו עוסקת במספר שאלות הקשורות באלגוריתמים לבדיקת תכונות ושימושיהם. החלק הראשון עוסק בבדיקת תכונות במודל הגרפים הצפופים, ובאלגוריתמים עם זמן ריצה תת-ליניארי הניתנים לבניה בהסתמך על בודקי תכונות במודל זה. החלק השני של התזה עוסק בבדיקת תכונות במודל האוריינטציה, שבו הקלטים הם אוריינטציות של גרף תשתית בלתי ניתן לשינוי הנתון לבודק מראש. החלק השלישי עוסק בשאלות הקשורות לבודקי תכונות עם הוכחות עזר (לרוב משתמשים במונח "הוכחות קרוב הניתנות לווידוא הסתברותי" לתיאור בעיות אלו), ובעיקר בשאלת הקשר בין הסתברות השגיאה (נאותות) לבין אורך ההוכחה של בודקים מהסוג הנ"ל.

המחקר נעשתה בהנחיית פרופסור חבר אלדר פישר
בפקולטה למדעי המחשב

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי

בדיקת תכונות וקירובים קומבינאטורים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

אריה מצליח

הוגש לסנט הטכניון - מכון טכנולוגי לישראל

תמוז תשס"ח חיפה יולי 2008

בדיקת תכונות וקירובים קומבינאטורים

אריה מצליח