ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών

Κωδικοποίηση Πιθανοτικώς Ελέγξιμων Αποδείξεων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ανδρέα Γ. Γαλάνη

Επιβλέπων: Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών

Κωδικοποίηση Πιθανοτικώς Ελέγξιμων Αποδείξεων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ανδρέα Γ. Γαλάνη

Επιβλέπων: Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η του Ιουλίου 2009.

.....	.....	.....
Ευστάθιος Ζάχος	Άρης Παγουρτζής	Δημήτρης Φωτάκης
Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2009.

.....

Ανδρέας Γ. Γαλάνης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τη δεκαετία του 1990, αποδείχτηκε ένα θεώρημα εξαιρετικής σημασίας στη θεωρία πολυπλοκότητας. Συγκεκριμένα, υπάρχει τρόπος κωδικοποίησης των αποδείξεων και ελέγχου της ορθότητάς τους, τέτοιος ώστε ο επαληθευτής να διαβάζει 3 μόνο δυαδικά ψηφία της απόδειξης και να αποκτά πιθανοτική αυτοπεποίθηση για την ισχύ της. Ακόμη, ο νέος αυτός τρόπος κωδικοποίησης αρκεί να έχει μόνο πολυωνυμικά μεγαλύτερο μέγεθος από αυτό της αρχικής απόδειξης. Θα εστιάσουμε σε ένα λιγότερα ισχυρό αποτέλεσμα, το οποίο αποκαλείται το PCP θεώρημα (θεώρημα Πιθανοτικώς Ελέγξιμων Αποδείξεων), στις οποίες ο επαληθευτής χρησιμοποιεί λογαριθμική τυχαιότητα και αναζητά ένα σταθερό αριθμό δυαδικών ψηφίων στην απόδειξη.

Η πρώτη απόδειξη του θεωρήματος συνδύασε προηγούμενα αποτελέσματα πάνω σε διάφορες αλγεβρικές τεχνικές και λίγο αργότερα ο Håstad παρουσίασε τον επαληθευτή των 3 δυαδικών ψηφίων που αναφέραμε προηγουμένως. Το 2005, η Irit Dinur έδωσε μια συνδυαστική απόδειξη του θεωρήματος. Ο κύριος στόχος αυτής της διπλωματικής εργασίας είναι η μελέτη της απόδειξης της Dinur. Από αυτή τη σκοπιά, θα επικεντρωθούμε στους τομείς που απαιτούνται για την κατανόηση της απόδειξης, μεταξύ των οποίων είναι οι επεκτατικοί γράφοι και η κωδικοποίηση κατά Hadamard.

Λέξεις κλειδιά

PCP θεώρημα, Πιθανοτικώς Ελέγξιμες Αποδείξεις, Απόδειξη Dinur, Επεκτατικοί γράφοι, Κωδικοποίηση Hadamard.

Abstract

In the 1990ŏs, a major theorem was proved in complexity theory. Namely, there is a format of writing proofs and checking their validity, such that the verifier reads only 3 bits of the proof to obtain probabilistic confidence in the validity of the proof. Furthermore, the new format needs only to be polynomially longer than the original proof. We will concentrate on a weaker result, stated as the PCP theorem (standing for Probabilistically Checkable Proofs), in which the verifier uses logarithmic randomness and queries a constant number of bits in the proof.

The first proof of the theorem combined previous results on various algebraic techniques and later Håstad presented the 3 bit verifier mentioned earlier. In 2005, Irit Dinur provided a combinatorial proof of the theorem. The main purpose of this diploma thesis is to overview Dinur's PCP construction. In this perspective, we shall focus on the subjects that are essential to understand the proof, amongst which are expander graphs and the Hadamard encoding.

Keywords

PCP theorem, Probabilistically Checkable Proofs, Dinur's proof, Expanders, Hadamard encoding.

Ευχαριστίες

Αυτή η διπλωματική δεν θα μπορούσε να είχε πραγματοποιηθεί χωρίς τη συμβολή του κ. Ζάχου σε πολλά επίπεδα. Ως καθηγητής στα προπτυχιακά μαθήματα της σχολής, μου έδωσε τα ερεθίσματα να ασχοληθώ με την επιστήμη των υπολογιστών. Ως επιβλέπων της διπλωματικής αυτής, με ενθάρρυνε και μου έδωσε τη δυνατότητα να ασχοληθώ με ένα πολύ ενδιαφέρον και πολύπλευρο θέμα. Τέλος, έκανε αυτή τη διπλωματική εργασία μια εμπειρία μοναδική φέρνοντας με σε επαφή με πλήθος ανθρώπων που μοιραζόμαστε κοινά ερευνητικά ενδιαφέροντα.

Θα ήθελα επίσης να ευχαριστήσω τα άλλα δύο μέλη της τριμελούς εξεταστικής επιτροπής, κ. Παγουρτζή και κ. Φωτάκη, που εκτός από διδάσκοντές μου, με καθοδήγησαν με τις συμβουλές τους για την καλύτερη δυνατή παρουσίαση της διπλωματικής εργασίας.

Ευχαριστώ επίσης την εργαστηριακή ομάδα του Corelab για το ευχάριστο και δημιουργικό κλίμα που μου προσέφεραν και τη συμμετοχή τους στη διόρθωση ατελειών της παρουσίασης.

Θέλω τέλος να ευχαριστήσω τους γονείς μου, που σε κάθε φάση της ζωής μου με στηρίζουν με τον καλύτερο δυνατό τρόπο.

Contents

Chapter 1

Introduction

1.1 Origins of PCP Verifiers

As it is widely believed, the class NP consists of problems for which there is no efficient algorithm. NP can be defined as the class of languages for which there exists a polytime deterministic verifier that can check membership proofs. The verifier can be viewed as a polynomial time deterministic Turing machine with oracle access to a string which encodes suitably the membership proof of a given input. It is not hard to see that this notion of NP is equivalent to its standard definition.

NP has played, and probably continue to play, a central role in complexity theory, since it is inextricably related to the current views on what can be computed efficiently. A journey through time can clearly convince us for this.

The theory of NP completeness due to Cook [Coo71], Levin and Karp [Kar72] in the 70's triggered instantaneously the interest of the computer science community on classifying problems according to their complexity. The results of this research showed that NP was the perfect theory model to unify a wide array of optimization problems arising in practice. The next question that immediately arose was "Ok, I proved that 3SAT is not efficiently solvable. What should I do with an instance of such a problem?".

Such type of questions in combination with the work of algorithm designers lead to the notion of approximation algorithms. That is, given an NP-complete problem, is an approximation of its optimal solution feasible and if yes, up to what degree. NP even provides the answer for some of these questions. But things became much more intricate, since the new type of questions were not that straightforward to answer. The classical reduction scheme seemed to reach a bottleneck which was hard to surpass.

This was a good reason for a good many people to start looking toward other classes of languages. Others started to introduce non-standard methods of checking membership proofs in a quest of different characterizations of NP.

The landmark of these approaches is the Interactive Proof system introduced by Goldwasser, Micali and Rackoff [GMR89] as well as the Arthur-Merlin games introduced by Babai [Bab85]. The main ingredient of their works is a verifier interacting with an all-powerful adversary, called the prover, aiming at convincing the verifier that a given input is in the language, even if this does not hold. Soon enough, the classes of languages defined by these proof checking protocols were proved to be equivalent.

The previous two protocols soon evolved into multi-prover interactive proof systems (MIP) [BOGKW88], where a single verifier interacts with multiple provers to verify an assertion. The provers cannot communicate with each other. Fortnow, Rompel and Sipser [FRS94], as part of understanding the complexity of MIP, proved that MIP is the set of languages for which membership proofs can be checked by a probabilistic polynomial time verifier that has random access to the proof. Since such a verifier can check exponential-sized proofs, MIP is a subset of the exponential analogue of NP, NEXPTIME.

In this context, the most important aspect of the previous work is the introduction of PCP verifiers in a primal form, where the interactive proof system was substituted by a single proof to be checked by a probabilistic verifier. From this point on, the more restrictions imposed on the verifier the more spectacular the results were. Imposing restrictions on the verifier was somewhat natural in terms of measuring its efficiency, starting with the work of Babai, Fortnow, Levin and Szegedy [Bab91], where the verifier was restricted in terms of its computation time and the proof had specified length. Feige, Goldwasser, Lovasz, Safra and Szegedy [FGL$^+$91] were the first to point out that the number of the queries into the proof, the randomness of the PCP verifier and the probability that the verifier accepts a proof membership can be associated with the hardness of approximating the MAX CLIQUE problem. Arora and Safra [AS92] explicitly defined the class PCP stressing its dependence on two resources: the random bits that the verifier uses and the number of queries that it is allowed to make into the proof.

1.2 The Class PCP

Following the previous discussion, we present the formal definition of the class PCP. Prior to that, we give the definitions of oracle access to a proof string y and restricted verifiers.

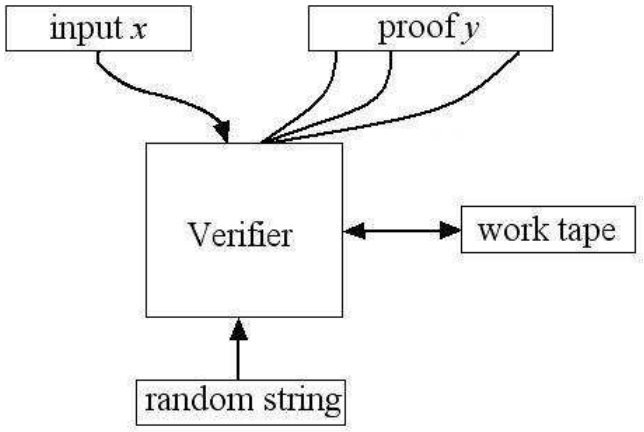
Definition 1.1 (Oracle Access). *An algorithm A, receiving as input x and a proof string y, has oracle access to the string y, if it can access a specific index within y with a cost of one step. This is denoted by $A^y(x)$.*

Definition 1.2 (Restricted Verifiers). *A verifier V is a (r, q)-restricted verifier for a language L if for input a string x, a proof string y and a random string*

ρ of length at most r, V decides whether $V^y(x,\rho) = 1$ or 0 by querying at most q bits in the proof.

Remark 1.1.
For our purposes, V will be non-adaptive, i.e. the queries into the proof depend only on the input x and the random string ρ and not on the way that V accesses the random bits in ρ.

A classic figure illustrating the verifier follows below.



Definition 1.3 (The Class PCP). *The class* $\mathrm{PCP}_{c,s}[r,q]$ *is the set of languages L for which there exists a (r,q)-restricted polynomial-time verifier V and a constant p such that:*

1. [Completeness] : $x \in L \Rightarrow \exists y, |y| \leq |x|^p; \Pr_{\rho\in\{0,1\}^r}[V^y(x,\rho) = 1] \geq c.$
2. [Soundness] : $x \notin L \Rightarrow \forall y, |y| \leq |x|^p; \Pr_{\rho\in\{0,1\}^r}[V^y(x,\rho) = 1] \leq s.$

The important parameters of the above definition are r, q, c, s. From these four parameters, the first two are of primary interest. In our scope of interest, c will be equal to 1, while s will be set to $1/2$. Unless otherwise stated, $\mathrm{PCP}[r,q]$ will denote the PCP class with the parameters c and s mentioned. Note that $c = 1$ implies perfect completeness with its standard notion in logic. Moreover, the soundness s is related to the number of queries allowed. The intuition behind this is that the k times repetition of the verification process, reduces the soundness parameter to s^k but increases the query complexity to $k \cdot q$ and the random bits used to $k \cdot r$.

We will not study PCP models with different soundness and completeness parameters. But the more general definition has lead to some sharp results which we will refer to in the next section.

1.3 The PCP Theorem

Arora, Lund, Motwani, Sudan and Szegedy [ALM+92] proved the following major theorem.

Theorem 1.4 (PCP Theorem).

$$\text{NP} = \text{PCP}[O(\log n), O(1)]$$

The theorem is striking in the sense that we can gain probabilistic confidence in the correctness of any NP membership proof by querying a constant number of bits in it. This result is highly nontrivial and the original proof was rather intricate too. Some of the techniques employed, to mention but a few, were the arithmetization of 3SAT, low-degree polynomials over fields, low-degree testing, sum-check protocols and Hadamard codes.

Note that proving the direction $\text{PCP}[O(\log n), O(1)] \subseteq \text{NP}$ is easy, as is captured by the following lemma.

Lemma 1.5.

$$\text{PCP}[O(\log n), O(1)] \subseteq \text{NP}$$

Proof Consider a verifier V for $L \in \text{PCP}[O(\log n), O(1)]$ and suppose that we are given an input x with $|x| = n$. Note that V uses $c \log n$ random bits and $k = O(1)$ bits. Thus, there are at most $2^{c \log n} = n^c$ different runs of the verifier on the input x. In each run V queries k bits into the proof. Thus, the proof should not have length greater than $O(1) \cdot n^c = O(n^c)$, which we assume to be without loss of generality n^d for some constant d.

Assign boolean variables $y_1, \ldots, y_{n^d}$ for each of the proof's positions. We construct a boolean function for each of the runs of V as follows.

- Fix a random string $\rho \in \{0,1\}^{c \log n}$.
- Assume the corresponding queries made by V given the string ρ are $y_{i_1}, \ldots, y_{i_k}$.
- Build a boolean function $f_\rho(y_{i_1}, \ldots, y_{i_k})$ which evaluates to 1, whenever the assignment to the variables $y_{i_1}, \ldots, y_{i_k}$ makes the verifier to accept. This can be accomplished by checking all the $2^k = O(1)$ possible assignments to $y_{i_1}, \ldots, y_{i_k}$ in constant time.

Thus we can construct in polynomial time n^c functions f_ρ which simulate each of the runs of V on input x. Clearly

- If $x \in L$, there is a proof which satisfies all the functions.

- If $x \notin L$, every proof satisfies at most $1/2$ of the functions.

It remains to observe that an assignment to the y_i's is an NP witness for x. ∎

The above proof has some interesting elements which are not obvious on first sight. First of all, note that each f_ρ is a boolean function on a *constant* number of variables, and thus it can be formulated as a 3SAT formula with a constant number of clauses (recall that $k = O(1)$). Thus, given a verifier for an NP language we can clearly build a 3SAT formula with polynomial number of clauses on polynomial number of variables. Since at most $1/2$ functions of the f_ρ can be satisfied, this can be used to prove that MAX-3SAT is hard to approximate. A detailed proof of the above intuition can be found in [Aro94]. We will present essentially the same argument from a different perspective in Chapter 5.

What is important to stress though is that, viewing the NP as the PCP verifier of Theorem 1.4 can lead in a straightforward manner to hardness of approximation results, which justifies the significance of the theorem.

1.4 History of Results

The story of the results that finally lead to the PCP theorem is a long one. We present a short overview over the sequence of such results.

Phase 1. The first non-trivial result was due to Babai, Fortnow and Lund [BFL90] who showed that NEXPTIME = PCP[poly(n), poly(n)]. Note that the PCP verifier queries the proof into polynomially many places, whereas the NEXPTIME verifier would look the whole exponential-sized proof.

Babai, Fortnow, Levin and Szegedy [Bab91] scaled this result down to NP, thus proving NP $\subseteq$ PCP[poly $\log n$,poly $\log n$]. Note that the randomness introduced in the PCP model, reduced the query complexity of the standard verifiers for these classes by a polylogarithmic factor. This was unexpected at the time. Still, these PCPs did not provide any new characterizations of NP since the containments were not equalities.

Phase 2. In phase 2, exact characterizations of NP were proved including the PCP theorem itself. The first such characterization was provided by Arora and Safra [AS92], namely NP = PCP$[O(\log n), o(\log n)]$. Their work was innovative in many aspects, the most important of which was the proof composition technique, a technique that is prevalent to subsequent PCP constructions.

It was in 1992 that Arora, Lund, Motwani, Sudan and Szegedy [ALM+92] in a joint work proved the PCP theorem. The fact that query complexity was independent of the proof size astonished the whole computer science community.

Phase 3. Research in this period examined the tightness of the parameters appearing in the PCP theorem. It was already known that any PCP for an NP-complete language should have query complexity at least 3. In fact, most works conjectured that $\mathrm{PCP}_{1,s}[O(\log n), 3]$ should belong in P for some $s > 1/2$. John Håstad [Hås97] settled these conjectures in a hard-cut manner proving that

$$\mathrm{NP} = \mathrm{PCP}_{1-\epsilon,\frac{1}{2}}[O(\log n), 3]$$

for any $\epsilon > 0$. Note that Håstad's PCP did not have perfect completeness but this was fixed in [GLST98], where $\mathrm{NP} = \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}[O(\log n), 3]$ was proved for any $\epsilon > 0$. Karloff and Zwick [KZ97] established the optimality of Håstad's result, by proving that $\mathrm{P} = \mathrm{PCP}_{1,\frac{1}{2}}[O(\log n), 3]$.

1.5 This Dissertation

The proof of the PCP theorem by Arora, Lund, Motwani, Sudan and Szegedy [ALM+92] was mainly algebraic, making extensive use of polynomial fields. This made the proof inaccessible to many computer scientists who were not that eager to follow the thorough PCP construction. This was the only "flaw" in the PCP theorem, for the astonishing theorem it was and the simple description it had, its proof was rather discouraging to go through. It is important that the reader should not get the wrong idea: the initial proof of the PCP theorem was startling by itself, using global techniques which can be applied to many fields.

In 2005, Irit Dinur [Din05], inspired by Reingold's work [Rei04] on L = SL, presented a less sophisticated proof providing a combinatorial argument for the NP-hardness of approximating a constraint satisfaction problem on graphs. This proof, while still applying the idea of proof composition, differed radically in grasp using an amplification scheme for the fraction of unsatisfied constraints, in contrast to previous approaches which preserved unsatisfiability in sufficiently big levels. The new proof has an immediate combinatorial essence which was absent in other PCP constructions.

In this diploma thesis, we study thoroughly Dinur's proof (chapter 5). To accomplish this goal, we study expander graphs and their properties to get a higher understanding of Dinur's construction (chapters 2 and 3). Moreover, we will go through some classical results in coding theory (chapter 4), so as to successfully follow the proof composition scheme. We will *not* pursue the applications to hardness of approximations. The interested reader may find such results in [Aro94], [Vaz01].

1.5.1 Knowledge assumed of the Reader

This diploma thesis is written in a self-contained manner so that the reader does not need to have high-level knowledge on any topic. Still, an elementary

background on linear Algebra will presumably be useful for chapters 2 and 3, and familiarity with probability theory will certainly be handful at the gap amplification proof. As far as complexity theory is concerned, we assume that the reader is acquainted with the standard notions of NP and polynomial time reductions. For an introduction to these, see [Pap94].

Chapter 2

Expanders

Expander graphs are a special class of graphs originating from the field of algebraic graph theory. Roughly speaking, an expander is a sparse graph in which the neighborhood of any set of vertices S is proportional to the size of S. As we shall see, this structural property of expanders has many useful extensions and together with the existence of constant degree expanders with high expansion, has lead to various beautiful and unexpected results. The proof of the PCP theorem by Dinur is heavily dependent on expanders and thus, it is more than important to study them in their own context.

2.1 Edge Expansion and Eigenvalues

Before we proceed into the formal definition of the edge expansion property, we will first introduce some notation. Assume that we have an undirected graph $G = (V, E)$ and two disjoint subsets A, B of V. We denote by $E(A, B)$ the set of edges with one endpoint in A and the other in B, i.e. the edges which separate A from B.

Definition 2.1 (Edge expansion). *Let $G = (V, E)$ a graph. The edge expansion of G, denoted by $h(G)$ is defined as*

$$h(G) := \min_{\substack{S \subset V \\ |S| \leq |V|/2}} \frac{|E(S, V - S)|}{|S|}$$

Note that the edge expansion property is a lower bound on the number of edges across cuts (normalized by the size of the sets involved). Other definitions of edge expansion are possible, as in [MR95] but most of them can be proved to be equivalent.

Definition 2.2 (Expander graph). *A graph $G = (V, E)$ is a (n, d, c) expander if $|V| = n$, G is d-regular and $h(G) \geq c$.*

The adjacency matrix of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix A in which the (u, v) element is equal to 1 iff $(u, v) \in E$. For simplicity, we will denote by $\mathbf{A}_{uv}$ the (u, v) element of $\mathbf{A}$. Observe that for an undirected graph, $\mathbf{A}$ is symmetric. In what follows, $\mathbf{A}$ will be the adjacency matrix of a d-regular undirected graph G and $n = |V|$ the number of vertices in G. Note that the column as well as the row sums are all equal to d since the graph is d-regular.

Since $\mathbf{A}$ is a symmetric real matrix, it has n eigenvalues. Denote them by $\lambda_1, \ldots, \lambda_n$. From now on, we will consider them in descending order

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$$

Even if the eigenvalues are not necessarily distinct, there exist n corresponding eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ which form an orthonormal basis of $\mathbb{R}^n$.

The following theorem sheds some light on the connection between the eigenvalues of the adjacency matrix and basic graph properties.

Theorem 2.3. *For a d-regular (multi)graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$ whose eigenvalues are $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$*

1. *$\lambda_1 = d$.*
2. *$\lambda_n \geq -d$.*
3. *G is connected iff $\lambda_1 > \lambda_2$.*
4. *Suppose that G is connected. Then G is bipartite iff $\lambda_n = -\lambda_1$.*

Proof Consider an eigenvalue λ of $\mathbf{A}$. Let $\mathbf{x}$ an eigenvector corresponding to λ. Denote by x_v the largest entry in $\mathbf{x}$. We can clearly assume that $\mathbf{x}$ has a positive entry (otherwise we may consider $-\mathbf{x}$). Thus $x_v > 0$. Note that

$$(\mathbf{Ax})_v = \lambda \cdot x_v \Rightarrow \sum_{(u,v)\in E} A_{uv} x_u = \lambda \cdot x_v$$

and by the selection of x_v

$$d \cdot x_v \geq \sum_{(u,v)\in E} A_{uv} x_u = \lambda x_v \Rightarrow (d - \lambda) x_v \geq 0 \tag{2.1}$$

and consequently $d \geq \lambda$. Considering the smallest entry in $\mathbf{x}$, one can prove in a similar fashion that $\lambda \geq -d$. Thus for every $1 \leq i \leq n$ we have that

$$|\lambda_i| \leq d$$

Since the graph is d-regular we immediately obtain that

$$\mathbf{A} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = d \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

and consequently $\lambda_1 = d$.

Let's return to (2.1). It is straightforward to see that equality holds iff $(u, v) \in E$ implies $x_u = x_v$. Thus if G is connected then the space of eigenvectors such that $\mathbf{Ax} = d \cdot \mathbf{x}$ has dimension 1 and consequently the eigenvalue d is of multiplicity 1.

Suppose now that $\lambda_1 > \lambda_2$ and G has wlog 2 components. Let V_1 and V_2 the sets of vertices in each component respectively and define the indicator vectors $\mathbf{x}_1, \mathbf{x}_2$

$$(\mathbf{x}_1)_v = \begin{cases} 1, & \text{if } v \in V_1 \\ 0, & \text{otherwise} \end{cases}$$

and similarly for $\mathbf{x}_2$. Then clearly $\mathbf{Ax}_1 = d \cdot \mathbf{x}_1$ and $\mathbf{Ax}_2 = d \cdot \mathbf{x}_2$. Since $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent, we have that the the space of eigenvectors such that $\mathbf{Ax} = d\mathbf{x}$ has dimension 2, and thus d has multiplicity 2 which is a contradiction to our assumption.

Finally, we prove 4. If G is bipartite then there exist $V_1, V_2 \subset V$ such that $E \subseteq (V_1 \times V_2)$ and $V_1 \cup V_2 = V$. Define $\mathbf{x}$ such that

$$(\mathbf{x})_v = \begin{cases} 1, & \text{if } v \in V_1 \\ -1, & \text{otherwise} \end{cases}$$

Clearly $\mathbf{Ax} = -d \cdot \mathbf{x}$ and since $|\lambda| \leq d$ for each eigenvalue, it must be the case that $\lambda_n = -d$.

For the converse, let $\mathbf{x}$ the eigenvector corresponding to $-d$ and note that for each vertex $v \in V$ we have that

$$\sum_{(u,v)\in E} A_{uv} x_u = -d \cdot x_v \Rightarrow \left| \sum_{(u,v)\in E} A_{uv} x_u \right| = d \cdot |x_v| \Rightarrow \sum_{(u,v)\in E} A_{uv} |x_u| \geq d \cdot |x_v|.$$

Summing over all $v \in V$ we obtain that each inequality must in fact be an equality. Consequently, if we denote by N_v the neighbors of vertex v, then it must be the case that either $x_u > 0$ for all $u \in N_v$ or $x_u < 0$ for all $u \in N_v$. Now, define V_1 the set of vertices for which $x_v > 0$ and V_2 the set of vertices for which $x_v < 0$ (note that x_v=0 for some vertex v would imply that G is not connected). It should be clear by the previous observation that there can be no edge inside V_1 or V_2. ∎

For our purposes the eigenvalue with the second largest magnitude has the most interesting properties. Namely, we will denote by $\lambda(G) = \max\{|\lambda_2|, |\lambda_n|\}$. For simplicity, when the graph G is fixed, we will use λ instead of $\lambda(G)$.

Next, we shall introduce some linear algebra notation. Consider two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$. Then, the *inner product* of $\mathbf{u}, \mathbf{v}$ is defined as

$$\mathbf{u}^{\mathrm{T}}\mathbf{v} = \sum_{i=1}^{n} u_i v_i$$

where $\mathbf{u}^{\mathrm{T}}$ denotes the transpose vector of $\mathbf{u}$. The vector $\mathbf{u}$ is *orthogonal* to $\mathbf{v}$, denoted by $\mathbf{u} \perp \mathbf{v}$, iff $\mathbf{u}^{\mathrm{T}}\mathbf{v} = 0$. The *Euclidean norm* of the vector u is equal to

$$\|\mathbf{u}\| := \sqrt{\mathbf{u}^{\mathrm{T}}\mathbf{u}} = \left(\sum_{i=1}^{n} u_i^2 \right)^{1/2}$$

whereas the l^1-*norm* is equal to

$$\|\mathbf{u}\|_1 := \sum_{i=1}^{n} |u_i|$$

Recall the Cauchy-Schwartz inequality $\mathbf{u}^{\mathrm{T}}\mathbf{v} \leq \|\mathbf{u}\| \cdot \|\mathbf{v}\|$.

The next theorem, will become handy in the proof of the main result in this section. A proof of the theorem can be found in Appendix.

Theorem 2.4 (Rayleigh Quotient). *Let* $\mathbf{A}$ *a symmetric real matrix with eigenvalues* $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ *and eigenvectors* $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ *respectively. Then*

1. $\lambda_1 = \max_{\mathbf{x} \in \mathbb{R}^n} \dfrac{\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|^2}$.

2. *For* $1 < i < n$, $\lambda_i = \max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \mathbf{x} \perp \mathbf{e}_1, \ldots, \mathbf{e}_{i-1}}} \dfrac{\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|^2}$.

We are now ready to proceed to the main result, which relates the *spectral gap* $d - \lambda(G)$ of a graph G with its edge expansion $h(G)$.

Theorem 2.5 (Cheeger's Inequality). *For a* d*-regular graph* $G = (V, E)$*, the following inequality holds*

$$\frac{h(G)^2}{2d} \leq d - \lambda(G) \leq 2h(G). \tag{2.2}$$

Proof We begin by proving the right part of the inequality. Let $\mathbf{A}$ the adjacency

matrix of G and recall that by Rayleigh's quotient,

$$\begin{aligned}\lambda(G) &= \max_{\mathbf{x}\perp\mathbf{e}_1} \frac{\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\ &= \max_{\mathbf{x}\perp\mathbf{e}_1} \frac{\sum_{u,v} A_{uv}x_u x_v}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\ &= \max_{\mathbf{x}\perp\mathbf{e}_1} \frac{\sum_{u,v} A_{uv}(x_u^2+x_v^2) - \sum_{u,v} A_{uv}(x_u-x_v)^2}{2\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\ &= \max_{\mathbf{x}\perp\mathbf{e}_1} \frac{2d\mathbf{x}^{\mathrm{T}}\mathbf{x} - \sum_{u,v} A_{uv}(x_u-x_v)^2}{2\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\ &= d - \max_{\mathbf{x}\perp\mathbf{e}_1} \frac{\sum_{u,v} A_{uv}(x_u-x_v)^2}{2\mathbf{x}^{\mathrm{T}}\mathbf{x}}\end{aligned}$$

Thus, we obtain the more convenient form for our purposes

$$d - \lambda = \max_{\mathbf{x}\perp\mathbf{e}_1} \frac{\sum_{u,v} A_{uv}(x_u-x_v)^2}{2\mathbf{x}^{\mathrm{T}}\mathbf{x}} \tag{2.3}$$

Note that $\mathbf{e}_1$ is actually the vector $\mathbf{1}_n = (1,\ldots,1)^{\mathrm{T}}$ multiplied by a scale factor of $1/\sqrt{n}$ (so as to have euclidean norm equal to 1). To see this, observe that by the regularity of G,

$$\mathbf{A}\mathbf{1}_n = d\mathbf{1}_n$$

which means that $\mathbf{1}_n$ is actually the vector corresponding to $\lambda_1 = d$, i.e. $\mathbf{e}_1$. Thus, the maximum in (2.3) is actually over the zero-sum vectors $\mathbf{x}$. Our task is to present such a vector $\mathbf{x}$, which renders the right part of (2.3) larger than $2h(G)$.

For this purpose, consider a set $S \subset V$ with $|S| \leq |V|/2$ such that $h(G) = \frac{|E(S,V-S)|}{|S|}$. Construct vector $\mathbf{x}$ as follows:

$$x_v = \begin{cases} |V|-|S|, & \text{if } v \in S \\ -|S|, & \text{otherwise} \end{cases}$$

Clearly the components of $\mathbf{x}$ add up to zero.

To make computations easier to follow, suppose that $n = |V|$ and $s = |S|$ and $|E(S,V-S)| = e$. Note that $e = s \cdot h(G)$ by the definition of S. It is straightforward to see that

$$\mathbf{x}^{\mathrm{T}}\mathbf{x} = \sum_{u\in V} x_u^2 = s(n-s)^2 + (n-s)s^2 = ns(n-s) \tag{2.4}$$

and, since $x_u - x_v = 0$ if u, v belong to the same partition of the cut defined by the set of edges $|E(S,V-S)|$,

$$\sum_{u,v} A_{uv}(x_u-x_v)^2 = 2|E(S,V-S)|(|V|-|S|+|S|)^2 = 2n^2e = 2h(G)n^2s \tag{2.5}$$

Using (2.4), (2.5) and recalling that $s \leq n/2$, (2.3) becomes

$$d - \lambda \leq \frac{n \cdot h(G)}{n - s} \leq 2h(G)$$

which is exactly the right part of (2.2).

For the left part of the inequality, consider a vector $\mathbf{x} \in \mathbb{R}^n$ equal to the eigenvector $\mathbf{e}_2$ and assume that at most $n/2$ of its entries are positive (otherwise we can work with $-\mathbf{e}_2$). Introduce vector $\mathbf{y} \in \mathbb{R}^n$ such that

$$y_v = \max\{x_v, 0\}.$$

Note that $\mathbf{Ay} \geq \lambda \mathbf{y}$ component-wise. To see that this is the case, observe that $y_v \geq 0$ and $y_v \geq x_v$ for each v so that

$$(\mathbf{Ay})_u = \sum_v A_{uv} y_v \geq \begin{cases} \sum_v A_{uv} x_v = \lambda x_u = \lambda y_u, & \text{if } x_u \geq 0 \\ 0 = \lambda y_u, & \text{if } x_u < 0 \end{cases}$$

It follows from the above property that

$$\mathbf{y}^{\mathrm{T}} \mathbf{Ay} \geq \lambda \mathbf{y}^{\mathrm{T}} \mathbf{y}$$

In order to get the expression $d - \lambda$, we rewrite $\mathbf{y}^{\mathrm{T}} \mathbf{Ay}$ as

$$\begin{aligned} \mathbf{y}^{\mathrm{T}} \mathbf{Ay} &= \sum_{(u,v)} A_{uv} y_u y_v \\ &= -\frac{1}{2} \sum_{u,v} A_{uv} (y_u - y_v)^2 + \frac{1}{2} \sum_{u,v} A_{uv} (y_u^2 + y_v^2) \\ &= -\frac{1}{2} \sum_{u,v} A_{uv} (y_u - y_v)^2 + d\mathbf{y}^{\mathrm{T}} \mathbf{y} \end{aligned}$$

so that

$$d - \lambda \geq \frac{\sum_{u,v} A_{uv} (y_u - y_v)^2}{2\mathbf{y}^{\mathrm{T}} \mathbf{y}} \tag{2.6}$$

Using Cauchy Schwarz inequality twice we can further write

$$\begin{aligned} \sum_{u,v} A_{uv} (y_u - y_v)^2 \underbrace{\sum_{u,v} 2A_{uv} (y_u^2 + y_v^2)}_{4d\mathbf{y}^{\mathrm{T}} \mathbf{y}} &\geq \sum_{u,v} A_{uv} (y_u - y_v)^2 \sum_{u,v} A_{uv} (y_u + y_v)^2 \\ &\geq \left(\sum_{u,v} A_{uv} \left| y_u^2 - y_v^2 \right| \right)^2 \end{aligned}$$

Thus, (2.6) becomes

$$d - \lambda \geq \frac{1}{2d} \left(\frac{\sum_{u,v} A_{uv} \left| y_u^2 - y_v^2 \right|}{2\mathbf{y}^{\mathrm{T}} \mathbf{y}} \right)^2 \tag{2.7}$$

Since the right hand side of (2.7) is symmetric, without loss of generality we may assume that y_v are in descending order, i.e. $y_1 \geq \ldots \geq y_n$ and consider t to be the largest index such that $y_t > 0$. Recall that $t \leq n/2$. We can now get rid of the absolute values in the sum

$$\begin{aligned}
\sum_{u,v} A_{uv} \left|y_u^2 - y_v^2\right| &= 2\sum_{i=1}^{t}\sum_{j=i}^{n} A_{ij}(y_i^2 - y_j^2) \\
&= 2\sum_{i=1}^{t}\sum_{j=i}^{n}\sum_{r=i}^{j-1} A_{ij}(y_r^2 - y_{r+1}^2) \\
&= 2\sum_{r=1}^{t}\sum_{i=1}^{r}\sum_{j=r+1}^{n} A_{ij}(y_r^2 - y_{r+1}^2) \\
&= 2\sum_{r=1}^{t}(y_r^2 - y_{r+1}^2)\sum_{i\leq r}\sum_{j>r} A_{ij} \qquad (2.8)
\end{aligned}$$

where the second step used the telescoping series $y_r^2 - y_{r+1}^2$ and the third a change of summation. Considering the sets $S_r = \{v_1, \ldots, v_r\}$ for each $1 \leq r \leq t$, it is straightforward to see that

$$\sum_{i\leq r}\sum_{j>r} A_{ij} = |E(S_r, V - S_r)|$$

and (2.8) yields

$$\begin{aligned}
\sum_{u,v} A_{uv}|y_u^2 - y_v^2| &= 2\sum_{r=1}^{t} |E(S_r, V - S_r)|(y_r^2 - y_{r+1}^2) \\
&\geq 2\sum_{r=1}^{t} h(G)|S_r|(y_r^2 - y_{r+1}^2) \\
&= 2h(G)\sum_{r=1}^{t} r(y_r^2 - y_{r+1}^2) \\
&= 2h(G)\sum_{r=1}^{t}(ry_r^2 - (r-1)y_r^2) \\
&= 2h(G)\sum_{r=1}^{t} y_r^2 \\
&= 2h(G)\mathbf{y}^{\mathrm{T}}\mathbf{y}
\end{aligned}$$

Using the last inequality and (2.7), the desired result follows. ∎

Remark 2.1.
The argument of the proof for the left part of Cheeger's inequality can easily be modified to present a cut of edge expansion at most $\sqrt{2d(d-\lambda)}$.

Remark 2.2.
Cheeger 's inequality bounds $h(G)$ so that

$$\frac{d-\lambda}{2} \leq h(G) \leq \sqrt{2d(d-\lambda)}.$$

Computing $h(G)$ exactly is coNP hard. An intuition for this is that it suffices to present a set S which makes $h(G) < c$. Cheeger's inequality is extremely useful because the expansion of most explicitly constructed families of expander graphs is proved by bounding the spectral gap first. Alon [Alo86] was the first to state this equivalence between edge expansion and spectral gap, though he worked with vertex expansion.

2.2 Random Walks

Random walks on expanders are particularly interesting due to their property that they are *rapidly mixing*: starting at any vertex, after a few steps, there is almost equal probability of being in any vertex.

More formally, consider a d-regular graph $G = (V, E)$ with n vertices and adjacency matrix $\mathbf{A}$ and let $\{X_i\}_{i=1}^{\infty}$ be a sequence of random variables such that $X_i \in V$.

The sequence $\{X_i\}_{i=0}^{\infty}$ is *a Markov chain* if for every sequence of vertices $\{v_i\}_{i=0}^{\infty}$ the following *memoryless property* holds for each $i \geq 0$:

$$\Pr[X_{i+1} = v_{i+1} | X_i = v_i, \ldots, X_1 = v_1] = \Pr[X_{i+1} = v_{i+1} | X_i = v_i]$$

In other words, X_i is a Markov chain iff the next step of the walk depends only on its current state and not on the steps taken so far to arrive at the current state. It follows that a Markov chain is determined by the parameters $T_{uv} = \Pr[X_{i+1} = u | X_i = v]$ and it is natural to call the matrix T the *probability transition* matrix.

A *random walk* on a d-regular graph G is defined as the Markov Chain with transition matrix $\tilde{\mathbf{A}} = \frac{1}{d}\mathbf{A}$. Note that each entry in $\tilde{\mathbf{A}}$ is between 0 and 1 and each row and column has sum equal to 1. Thus, in each transition step of the random walk and assuming that the present state is a vertex u, we pick as the next state one of the neighbors u of v according to $\tilde{\mathbf{A}}$, and repeat the transition step from u. For each step of the random walk, we may use a *probability vector* $\mathbf{p} \in \mathbb{R}^n$ such that the probability that the random walk is currently in vertex v is equal to the v-th component of $\mathbf{p}$. Thus, for every probability vector $\mathbf{p}$, $p_v \geq 0$ and $\sum_v p_v = 1$. We denote the probability vector with the uniform distribution $\frac{1}{n}\mathbf{1}$ by $\mathbf{u}$.

It is pretty straightforward to see that the eigenvalues of $\tilde{\mathbf{A}}$ are related to

the eigenvalues of $\mathbf{A}$ by a $1/d$ scaling factor, namely

$$\tilde{\lambda}_i = \frac{1}{d}\lambda_i,$$

whereas the (orthonormal) eigenvectors $\mathbf{e}_i$ remain the same. Note that $\tilde{\lambda}_1 = 1$ and $\mathbf{u} = \frac{1}{\sqrt{n}}\mathbf{e}_1$.

Lemma 2.6. *If at the i-th stage of a random walk the probability distribution on the vertices is $\mathbf{p}$, then at the $(i+1)$-th stage, the probability distribution is $\tilde{\mathbf{A}}\mathbf{p}$.*

Proof It suffices to show that $\Pr[X_{i+1} = u] = (\tilde{\mathbf{A}}\mathbf{p})_u$. But this is true since

$$\Pr[X_{i+1} = u] = \sum_v \Pr[X_{i+1} = u | X_i = v] \Pr[X_i = v] = \sum_v A_{uv} p_v = (\tilde{\mathbf{A}}\mathbf{p})_u \quad \blacksquare$$

Thus, if we start the random walk according to an initial distribution $\mathbf{p}$ on the set of vertices, the probability distribution after t steps is simply $\tilde{\mathbf{A}}^t\mathbf{p}$. Note that since $\mathbf{u}$ is the eigenvector corresponding to the eigenvalue 1 of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{A}}\mathbf{u} = \mathbf{u}$ and consequently a random walk which begins from a uniform distribution remains at a uniform distribution.

The following theorem shows that any random walk beginning from an arbitrary probability distribution $\mathbf{p}$ converges to the uniform distribution at an exponential rate.

Theorem 2.7. *For any probability vector $\mathbf{p}$ and any $t \geq 0$, $\left\|\tilde{\mathbf{A}}^t\mathbf{p} - \mathbf{u}\right\|_2 \leq \tilde{\lambda}^t$.*

Proof We induct on t. Set $\mathbf{v} = \mathbf{p} - \mathbf{u}$. For $t = 0$ the theorem states that $\|\mathbf{v}|_2 \leq 1$. To prove this, note that $\sum_v p_v = \sum_v u = 0$ and therefore $\sum_v \mathbf{v}_v = 0$. It follows that $\mathbf{v}^{\mathrm{T}}\mathbf{u} = 0$ and consequently $\mathbf{v}$ is orthogonal to $\mathbf{u}$. Since $\mathbf{p} = \mathbf{u} + \mathbf{v}$, $\mathbf{v}^T\mathbf{p} = \mathbf{v}^T\mathbf{v}$. Using the Cauchy Schwarz inequality,

$$\|\mathbf{v}\|_2 \|\mathbf{p}\|_2 \geq (\mathbf{v}^T\mathbf{p})^2 = \|\mathbf{v}\|_2^2$$

yielding $\|\mathbf{p}\|_2 \geq \|\mathbf{v}\|_2$ (a more laconic way to obtain this inequality would be to use the orthogonality of $\mathbf{u}, \mathbf{v}$ and Pythagoras theorem for the hypotenuse $\mathbf{p}$). Since $0 \leq p_v \leq 1$ we finally have that

$$\|\mathbf{v}\|_2 \leq \|\mathbf{p}\|_2 = \left(\sum_v p_v^2\right)^{1/2} \leq \left(\sum_v p_v\right)^{1/2} = 1$$

which proves the base case of the induction.

Assuming the claim true for t and noting that $\tilde{\mathbf{A}}^t\mathbf{p} = \tilde{\mathbf{A}}^t(\mathbf{u} + \mathbf{v}) = \mathbf{u} + \tilde{\mathbf{A}}^t\mathbf{v}$, the induction hypothesis implies $\left\|\tilde{\mathbf{A}}^t\mathbf{v}\right\|_2 \leq \tilde{\lambda}^t$. We would like to prove that $\left\|\tilde{\mathbf{A}}^{t+1}\mathbf{v}\right\|_2 \leq \tilde{\lambda}\left\|\tilde{\mathbf{A}}^t\mathbf{v}\right\|_2$ from which the induction step clearly follows.

As we saw above $\mathbf{v} \perp \mathbf{u}$ and therefore $\mathbf{v}$ is a linear combination of the eigenvectors $\mathbf{e}_2, \ldots, \mathbf{e}_n$ of $\tilde{\mathbf{A}}$. Consequently, there exist constants c_i such that $\mathbf{v} = \sum_{i=2}^{n} c_i \mathbf{e}_i$. Therefore,

$$\begin{aligned}
\left\| \tilde{\mathbf{A}}^{t+1}\mathbf{v} \right\|_1 &= \left\| \tilde{\mathbf{A}}^{t}(\tilde{\mathbf{A}}\mathbf{v}) \right\|_2 \\
&= \left\| \tilde{\mathbf{A}}^{t} \sum_{i=2}^{n} c_i \tilde{\mathbf{A}}\mathbf{e}_i \right\|_2 \\
&= \left\| \tilde{\mathbf{A}}^{t} \sum_{i=2}^{n} c_i \lambda_i \mathbf{e}_i \right\|_2 \\
&\leq \tilde{\lambda} \left\| \tilde{\mathbf{A}}^{t} \sum_{i=2}^{n} c_i \mathbf{e}_i \right\|_2 \\
&= \tilde{\lambda} \left\| \tilde{\mathbf{A}}^{t}\mathbf{v} \right\|_2
\end{aligned}$$

and the theorem follows. ∎

Remark 2.3.
Suppose that we are given a regular connected graph G. The above theorem implies that the deviation from the uniform distribution after $t = c(1-\tilde{\lambda})^{-1} \ln n$ steps will be

$$\left\| \tilde{\mathbf{A}}^{t}\mathbf{p} - \mathbf{u} \right\|_2 \leq \tilde{\lambda}^t = \left(1 - \frac{1}{(1-\tilde{\lambda})^{-1}}\right)^{c(1-\tilde{\lambda})^{-1}\ln n} \leq \mathrm{e}^{-c\ln n} = \frac{1}{n^c}.$$

Thus, after $O\left(\frac{1}{1-\tilde{\lambda}} \log n\right)$ steps, the random walk will be inverse polynomially close to the uniform distribution.

The interested reader is referenced to the excellent survey on random walks by Lovász [Lov93].

2.3 The Expander Mixing Lemma

Up to this point, we have focused our conversation on regular graphs which satisfy $\tilde{\lambda} < 1$. If we further impose λ to be close to 0, i.e. $\lambda \leq \epsilon$ for some $\epsilon > 0$, one can get nice unexpected properties on such graph families. The following lemma, proved by Alon and Chung [AC88] illustrates one of those properties.

Lemma 2.8 (Expander Mixing Lemma). *Let G be a d-regular graph with n vertices and second largest eigenvalue in absolute value $\tilde{\lambda}$ and S, T two disjoint subsets of its vertices. Then*

$$\left| |E(S,T)| - d\frac{|S||T|}{n} \right| \leq \tilde{\lambda} \cdot d \cdot \sqrt{|S||T|} \leq \frac{1}{2}\tilde{\lambda} \cdot d \cdot n$$

Proof Consider the indicator vector for S, denoted by $\mathbf{1}_S$, having a one in exactly those positions v for which $v \in S$. Define similarly $\mathbf{1}_T$. Let $\tilde{\mathbf{A}}$ denote the normalized adjacency matrix of G.

We write $\mathbf{1}_S$ as a linear combination of the eigenvectors $\mathbf{e}_i$ of $\tilde{\mathbf{A}}$, namely

$$\mathbf{1}_S = \sum_{i=1}^{n} c_i \mathbf{e}_i = c_1 \mathbf{e}_1 + \mathbf{s}$$

where $\mathbf{s} = \sum_{i=2}^{n} c_i \mathbf{e}_i$. In a similar fashion, write $\mathbf{1}_T$ as $c_1' \mathbf{e}_1 + \mathbf{t}$. Note that $c_1 = \mathbf{e}_1^{\mathrm{T}} \mathbf{1}_S = |S|/\sqrt{n}$ and likewise, $c_1' = |T|/\sqrt{n}$. Observing that $|E(S,T)| = \mathbf{1}_S^{\mathrm{T}}(d\tilde{\mathbf{A}})\mathbf{1}_T$ we have

$$\begin{aligned} E(S,T) &= \mathbf{1}_S^{\mathrm{T}}(d\tilde{\mathbf{A}})\mathbf{1}_T \\ &= d(c_1\mathbf{e}_1 + \mathbf{s})^{\mathrm{T}}\tilde{\mathbf{A}}(c_1'\mathbf{e}_1 + \mathbf{t}) \\ &= d(c_1c_1'\mathbf{e}_1^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{e}_1 + c_1\mathbf{e}_1^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t} + c_1'\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{e}_1 + \mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}) \end{aligned}$$

Each of the terms $\mathbf{e}_1^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}, \mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{e}_1$ is equal to 0, since, for instance, $\tilde{\mathbf{A}}\mathbf{t}$ is a linear combination of $\mathbf{e}_2, \ldots, \mathbf{e}_n$ and $\mathbf{e}_1$ is orthonormal to each one of them. Thus

$$\begin{aligned} |E(S,T)| &= d(c_1c_1'\mathbf{e}_1^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{e}_1 + \mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}) \\ &= d\left(\frac{|S||T|}{n} + \mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}\right) \end{aligned}$$

Thus it suffices to prove that

$$\left|\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}\right| \leq \tilde{\lambda} \cdot \sqrt{|S||T|}$$

Using once again the Cauchy Schwarz inequality,

$$\left|\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}\right| = \left\|\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}\right\| \leq \|\mathbf{s}^{\mathrm{T}}\| \left\|\tilde{\mathbf{A}}\mathbf{t}\right\| \tag{2.9}$$

It is easy to see that $\|\tilde{\mathbf{A}}\mathbf{t}\| = \left\|\sum_{i=2}^{n} c_i\tilde{\lambda}_i\mathbf{e}_i\right\| \leq \tilde{\lambda}\|\mathbf{t}\|$. Moreover $\|\mathbf{1}_S\| \geq \mathbf{s}$, because $\mathbf{1}_S$ is the hypotenuse of the orthogonal triangle formed by the vectors $\mathbf{s}$ and $c_1\mathbf{e}_1$. Similarly, $\|\mathbf{1}_T\| \geq \mathbf{t}$. (2.9) yields the desired inequality:

$$\left|\mathbf{s}^{\mathrm{T}}\tilde{\mathbf{A}}\mathbf{t}\right| \leq \lambda \|\mathbf{1}_S\| \|\mathbf{1}_T\| = \lambda\sqrt{|S||T|}$$ ∎

Remark 2.4.
Identifying the term $d|S||T|/n$ as the expected number of edges from S to T, we can see that the expander mixing lemma bounds the deviation from the behavior of a random graph. Indeed, a subset S of a random d-regular graph has $d|S|$ edges with at least one endpoint in $|S|$. The probability of such an edge to have the other point in T is exactly $|T|/n$.

In a good expander, the second largest eigenvalue in absolute value λ is low and consequently the number of edges between two subsets of vertices is approximately what it would be in a random d-regular graph.

There is a partial converse to the expander mixing lemma stating that for a d-regular graph, λ is essentially (up to a $\log d$ factor), the best constant that can occur in the expander mixing lemma. Namely:

Theorem 2.9 ([BL06]). *Let G be a $d-$regular graph with n vertices. If for all $S, T \subset V$ the inequality*

$$\left| |E(S,T)| - d\frac{|S||T|}{n} \right| \leq \theta \cdot d \cdot \sqrt{|S||T|}$$

holds for some fixed $\theta > 0$ then

$$\lambda = O\left(\theta\left(1 + \log\frac{d}{\theta}\right)\right)$$

2.4 Random Walks on Expanders

In this section we prove a really useful lemma which illustrates the power of expanders. In fact, this lemma will be used in chapter 5 to show that the correlation of two paths in an expander is low.

Lemma 2.10. *Let $G = (V, E)$ be a d-regular graph with $\tilde{\lambda}(G) \leq \tilde{\lambda} < 1$ and let $F \subset E$. Let $u_0, \ldots, u_k$ be a random walk in G where the starting point u_0 is chosen by picking a random edge in F and then a random endpoint of the edge. Then the probability that (u_{k-1}, u_k) is in F is at most*

$$\frac{|F|}{|E|} + \tilde{\lambda}^{k-1}$$

Proof Let $\tilde{\mathbf{A}}$ denote as usual the normalized transition matrix of G. Denote by $d_F(v)$ the number of edges in F incident to vertex v. Then the vector describing the initial distribution u_0 can easily be seen that it can be described be a vector $\mathbf{x}$ whose v-th component is equal to $\frac{d_F(v)}{2|F|}$.

It must also be clear by now that the distribution of any u_t in the path is given by $\tilde{\mathbf{A}}^t\mathbf{x}$. Specifically, the distribution of u_{k-1} is $\mathbf{y} = \tilde{\mathbf{A}}^{k-1}\mathbf{x}$. Moreover, if the walk is at vertex v after $k-1$ steps, then the probability that the last step will be along an edge in F is $\frac{d_F(v)}{d}$. Thus

$$\begin{aligned}\Pr[(u_{k-1}, u_k) \in F] &= \sum_{v \in V} y_v \frac{d_F(v)}{d} = \sum_{v \in V} y_v \frac{2|F|x_v}{d} = \frac{2|F|}{d}\sum_{v \in V} y_v x_v \\ &= \frac{2|F|}{d}\mathbf{x}^{\mathrm{T}}\mathbf{y} = \frac{2|F|}{d}\mathbf{x}^{\mathrm{T}}\tilde{\mathbf{A}}^{k-1}\mathbf{x} \qquad (2.10)\end{aligned}$$

We now write $\mathbf{x}$ as a linear combination of the eigenvectors of $\tilde{\mathbf{A}}$, namely

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{e}_i = c_1 \mathbf{e}_1 + \mathbf{x}'$$

where $\mathbf{x}' = \sum_{i=2}^{n} c_i \mathbf{e}_i$. Then

$$\begin{aligned}\mathbf{x}^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x} &= (c_1 \mathbf{e}_1 + \mathbf{x})^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} (c_1 \mathbf{e}_1 + \mathbf{x}') \\ &= c_1^2 \mathbf{e}_1^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{e}_1 + 2 c_1 \mathbf{e}_1^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x}' + \mathbf{x}'^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x}'\end{aligned}$$

and observing that $\tilde{\mathbf{A}}\mathbf{x}'$ is a linear combination of $\mathbf{e}_2, \ldots, \mathbf{e}_n$ we obtain that $\mathbf{e}_1^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x}' = 0$ and thus

$$\mathbf{x}^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x} = c_1^2 \mathbf{e}_1^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{e}_1 + \mathbf{x}'^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x}'$$

Note that $\mathbf{e}_1^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{e}_1 = \mathbf{e}_1^{\mathrm{T}} \mathbf{e}_1 = 1$, whereas $\mathbf{x}'^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x}' \leq \lambda^k \mathbf{x}'^{\mathrm{T}} \mathbf{x}'$ from Rayleigh's quotient. In addition, $\|\mathbf{x}'^{\mathrm{T}}\| \leq \|\mathbf{x}^{\mathrm{T}}\|$ due to Pythagoras' inequality and

$$c_1 = \mathbf{x}^{\mathrm{T}} \mathbf{e}_1 = \frac{1}{\sqrt{n}} \sum_v \left(\frac{d_F(v)}{2|F|} \right) = \frac{1}{\sqrt{n}}$$

It follows that

$$\mathbf{x}^{\mathrm{T}} \tilde{\mathbf{A}}^{k-1} \mathbf{x} \leq \frac{1}{n} + \lambda^k \mathbf{x}^T \mathbf{x}$$

But we can see that

$$\mathbf{x}^T \mathbf{x} = \sum_v \left(\frac{d_F(v)}{2|F|} \right)^2 \leq \max_v \frac{d_F(v)}{2|F|} \cdot \sum_v \frac{d_F(v)}{2|F|} = \max_v \frac{d_F(v)}{2|F|} \leq \frac{d}{2|F|}$$

(2.10) transforms into

$$\Pr[(u_{k-1}, u_k) \in F] = \frac{2|F|}{d} \left(\frac{1}{n} + \tilde{\lambda}^k \frac{d}{2|F|} \right) = \frac{|F|}{|E|} + \tilde{\lambda}^k \qquad \blacksquare$$

Chapter 3

Existence of Expanders

We have studied quite extensively expander graphs and their properties. But so far we have not proved their existence. In this chapter, we address this topic which has been of special research interest the last years since the need for explicit expander construction emerged in several applications. The interested reader can find more information about advances on this field in [LW03].

3.1 A Probabilistic Argument

Indeed, as a first result we prove ([Pin73]) that a random regular graph is an expander with high probability.

Theorem 3.1. *For any even $d \geq 3$ there exists a family of d-regular expander graphs.*

Proof We will use the probabilistic method to prove that there exists $\epsilon > 0$ such that for every $n > 0$, if we pick a random d-regular graph G, then with positive probability $h(G) > \epsilon$.

Consider a perfect matching on dn vertices.

1. Randomly partition these vertices into n sets $V_1, \ldots, V_n$ each of size d.
2. Merge the vertices in each V_i to a single vertex i without eliminating any edge.

We have obtained a d-regular (multi)graph G with n vertices $\{1, \ldots, n\}$. Pick an S such that $|S| \leq n/2$. We want to bound the edges that go out of S, thus it suffices to provide an upper bound the number of edges inside S. Wlog label the vertices in S as $\{1, \ldots, |S|\}$.

Denote by $P_{|S|}$ the event that the number of edges inside $|S|$ is at least $|S|(d/2 - \epsilon)$. This means that $V_1 \cup \ldots V_{|S|}$ includes $|S|(d/2 - \epsilon)$ edges. Thus, we seek the

probability of picking $|S|(d/2-\epsilon)$ edges in a random selection of $|S|d$ vertices from the original graph. This is equivalent to picking a set X of vertices such that $|X| = d|S|$ using the following random process.

- Pick $|S|(d/2-\epsilon)$ edges and assign their endpoints in X.
- Pick the rest $2\epsilon|S|$ vertices in X from the rest $n - 2|S|(d/2-\epsilon)$ vertices.

Thus the desired probability is bounded above by

$$\frac{\binom{dn/2}{|S|(d/2-\epsilon)}\binom{n-2|S|(d/2-\epsilon)}{2\epsilon|S|}}{\binom{dn}{d|S|}}$$

The probability that at least one of the $P_{|S|}$ occurs (for $|S| = 1, \ldots, n/2$) is hence at most

$$\sum_{|S|=1}^{n/2} \frac{\binom{dn/2}{|S|(d/2-\epsilon)}\binom{n-2|S|(d/2-\epsilon)}{2\epsilon|S|}}{\binom{dn}{d|S|}} < 1$$

for sufficiently small ϵ. Consequently, there exists a graph G on n vertices such that none of the $P_{|S|}$ occurs. Clearly, for this graph G it holds that

$$h(G) = \min_{\substack{S \subset V \\ |S| \leq |V|/2}} \frac{E(S, V-S)}{|S|} \geq \epsilon$$

which proves the claim. ∎

Remark 3.1.
We could clearly pick either d even or n even. We chose d since we are interested in proving the existence of a d-regular n-vertex for a constant d and every $n \in \mathbb{N}$.

Thus existence of expander graphs has come easier than one would probably expect. Still, constructing expander graphs *explicitly* is far from easy. In the next sections we provide such constructions.

3.2 The Zig Zag Product

Reingold [Rei04] in his proof of $L = SL$, introduced an interesting technique to produce expander graphs. His work was the first incentive for Dinur for proving the PCP theorem by gap amplification. Below we present the zig zag method.

Definition 3.2. *Let $G = (V, E)$ an n-vertex, D-regular undirected multigraph and $H = (V', E')$ a D-vertex, d-regular undirected (multi)graph. The zig-zag product of the two graphs, denoted by $G ⓩ H$, is defined as follows.*

- *Replace each vertex of G with a copy of H, which will be called a cloud. For $v \in V$, $j \in V'$, we denote by (v,j) the j-th vertex in the cloud of v. Denote by $[v]$ the set of vertices in v's cloud.*
- *(u,i) and (v,j) are connected by an edge in $G\textcircled{z}H$ if (v,j) can be reached from (u,i) by taking a step in the first cloud, then a step between the clouds and then a step in the second cloud. More formally, $((u,i),(v,j)) \in E(G\textcircled{z}H)$ if there exist k,l such that:*
  - $(i,k) \in E'$.
  - *v is the k-th neighbor of u in G*
  - *u is the l-th neighbor of v in G.*
  - $(l,j) \in E'$.

Remark 3.2.
Clearly $G\textcircled{z}H$ is a nD-vertex graph. The graph is d^2-regular, since from each vertex in $G\textcircled{z}H$ we can choose by d ways the first step within the cloud. Once the first step is chosen, the second is uniquely defined since each edge in G is associated with exactly two vertices in $G\textcircled{z}H$. The third step can again be pulled out with d ways, giving a total of d^2 neighbors.

Before proving the main result of this section, let us describe in a neat way the normalized adjacency matrix of $G\textcircled{z}H$. Denote by $\tilde{\mathbf{A}}_H$ the normalized adjacency matrix of H.

Introduce the $(nD \times nD)$ matrix $\mathbf{A}$ such that

$$\mathbf{A}[(u,i),(v,j)] = \begin{cases} 1, & v \text{ is the } i\text{-th neighbor of } u,\ u \text{ is the } j\text{-th neighbor of } v \\ 0, & \text{otherwise} \end{cases}$$

Note that $\mathbf{A}$ is a permutation matrix, i.e. in every row and column there is exactly one non-zero element. Next, introduce the $(nD \times nD)$ bock matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} \tilde{\mathbf{A}}_H & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}}_H & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_H & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}}_H \end{bmatrix}$$

where $\mathbf{B}$ has n copies of $\tilde{\mathbf{A}}_H$. Note that both $\mathbf{B}$ is a symmetric matrix. It is easy to verify that the normalized adjacency matrix of $G\textcircled{z}H$ is equal to $\mathbf{BAB}$.

Theorem 3.3. *If G is a D-regular graph on n vertices and H is a d-regular graph on D vertices, then*

$$\tilde{\lambda}\left(G\textcircled{z}H\right) \leq \tilde{\lambda}(G) + \tilde{\lambda}(H) + \left(\tilde{\lambda}(H)\right)^2$$

Proof Let $\tilde{\mathbf{M}}$ denote the normalized adjacency matrix of $G\textcircled{z}H$ and consider the matrices $\mathbf{A}, \mathbf{B}$ mentioned earlier, so that

$$\tilde{\mathbf{M}} = \mathbf{BAB}$$

Since $G\textcircled{z}H$ is a regular graph, from the Rayleigh quotient (see Appendix) we have that

$$\tilde{\lambda}\left(G\textcircled{z}H\right) = \max_{\mathbf{x}\perp\mathbf{1}_{nD}, \|\mathbf{x}\|=1} \left|\mathbf{x}^{\mathrm{T}}\mathbf{BABx}\right| \tag{3.1}$$

In order to correlate $\tilde{\lambda}\left(G\textcircled{z}H\right)$ with $\tilde{\lambda}(G)$ and $\tilde{\lambda}(H)$, we need to split the vector in such a way that we will be able to use the respective Rayleigh quotients for G and H. As such, we split the vector $\mathbf{x}$ in the above equation into two vectors $\mathbf{x}_{||}$ and $\mathbf{x}_{\perp}$ such that

$$\mathbf{x}_{||}(u,i) = \frac{1}{D}\sum_{j\in[u]} \mathbf{x}(u,j)$$
$$\mathbf{x}_{\perp} = \mathbf{x} - \mathbf{x}_{||}$$

Observe that if $\mathbf{x}\perp\mathbf{1}_{nD}$ then $\mathbf{x}_{||}\perp\mathbf{1}_{nD}$ too, since the sum of the entries in $\mathbf{x}_{||}$ is the same as that of $\mathbf{x}$. Denote by $\mathbf{x}_{\perp}^{[v]}$ the $D\times 1$ vector which is the restriction of $\mathbf{x}_{\perp}$ onto the cloud $[v]$. Then, since $\mathbf{x}_{\perp} = \mathbf{x} - \mathbf{x}_{||}$ and by the way $\mathbf{x}_{||}$ is defined, for each v it holds that $\mathbf{x}_{\perp}^{[v]}$ is perpendicular to $\mathbf{1}_D$ and since $\mathbf{B}$ is block diagonal we have that

$$\begin{aligned}
\left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{B}\mathbf{x}_{\perp}\right| &= \left|\sum_v \sum_{j\in[v]} \left(\mathbf{x}_{\perp}^{[v]}\right)^{\mathrm{T}} \tilde{\mathbf{A}}_H \mathbf{x}_{\perp}^{[v]}\right| \\
&\leq \sum_v \sum_{j\in[v]} \left|\tilde{\lambda}(H)\left\|\mathbf{x}_{\perp}^{[v]}\right\|^2\right| \\
&\leq \tilde{\lambda}(H)\|\mathbf{x}_{\perp}\|^2
\end{aligned}$$

In a similar fashion, define $\mathbf{x}_{||}^{[v]}$ the $nD\times 1$ vector which is the restriction of $\mathbf{x}_{||}$ onto the vertices of the cloud $[v]$. Then

$$\mathbf{B}\mathbf{x}_{||} = \sum_v \tilde{\mathbf{A}}_H \mathbf{x}_{||}^{[v]} = \sum_v \tilde{\lambda}_1(H)\mathbf{x}_{||}^{[v]} = \sum_v \mathbf{x}_{||}^{[v]} = \mathbf{x}_{||} \tag{3.2}$$

and

$$\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{x}_{||}^{[v]} = \sum_v \mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{x}_{||}^{[v]} = \sum_v \sum_{j\in[v]} \left(\frac{1}{D}\sum_{j\in[u]} \mathbf{x}(u,j)\right) \mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{1}_D = 0.$$

Thus $\mathbf{x}_{\perp}\perp\mathbf{x}_{||}$ and Pythagoras' theorem yields

$$\|\mathbf{x}_{||}\| \leq \|\mathbf{x}\| = 1$$
$$\|\mathbf{x}_{\perp}\| \leq \|\mathbf{x}\| = 1.$$

The expression to be maximized in the right part of (3.1) transforms into

$$\begin{aligned}\left|\mathbf{x}^{\mathrm{T}}\mathbf{BABx}\right| &= \left|(\mathbf{x}_{||}+\mathbf{x}_{\perp})^{\mathrm{T}}\mathbf{BAB}(\mathbf{x}_{||}+\mathbf{x}_{\perp})\right| \\ &= \left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{||}+\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{\perp}+\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{||}+\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| \\ &\leq \left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{||}\right|+2\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right|+\left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| \qquad (3.3)\end{aligned}$$

We analyze each term in (3.3) separately.

The term $\left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right|$. Applying Cauchy Schwarz inequality

$$\left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| \leq \left\|\mathbf{A}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{x}_{\perp}\right\| \cdot \|\mathbf{Bx}_{\perp}\|$$

and noting that $\left\|\mathbf{A}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{x}_{\perp}\right\| = \|\mathbf{Bx}_{\perp}\|$ since $\mathbf{A}$ is a permutation matrix and $\mathbf{B}$ is symmetric,

$$\left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| \leq \|\mathbf{Bx}_{\perp}\|^2$$

It remains to observe that that $\|\mathbf{Bx}_{\perp}\|^2 = \left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{B}^2\mathbf{x}_{\perp}\right| \leq (\tilde{\lambda}(H))^2\|\mathbf{x}_{\perp}\|^2$, hence

$$\left|\mathbf{x}_{\perp}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| \leq (\tilde{\lambda}(H))^2. \qquad (3.4)$$

The term $\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right|$. Using (3.2) and the Cauchy Schwarz inequality, we have that

$$\begin{aligned}\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| &= \left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{ABx}_{\perp}\right| \\ &\leq \left\|\mathbf{A}^{\mathrm{T}}\mathbf{x}_{||}\right\| \cdot \|\mathbf{Bx}_{\perp}\| \\ &= \left\|\mathbf{x}_{||}\right\| \cdot \|\mathbf{Bx}_{\perp}\| \\ &\leq \tilde{\lambda}(H) \cdot \left\|\mathbf{x}_{||}\right\| \cdot \|\mathbf{x}_{\perp}\|\end{aligned}$$

Using the inequality $a^2+b^2 \geq 2ab$ we have that

$$\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{\perp}\right| \leq \frac{1}{2}\tilde{\lambda}(H) \cdot \left(\left\|\mathbf{x}_{||}\right\|^2+\|\mathbf{x}_{\perp}\|^2\right) = \frac{1}{2}\tilde{\lambda}(H) \cdot \|\mathbf{x}\|^2 = \frac{1}{2}\tilde{\lambda}(H). \qquad (3.5)$$

The term $\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{||}\right|$. Define $\mathbf{y} \in \mathbb{R}^n$ such that

$$\mathbf{y}(u) = \frac{1}{D}\sum_{j \in [u]} \mathbf{x}(u,i)$$

Note that $\mathbf{y} \perp \mathbf{1}_n$ and that for each $j \in [u]$ it holds that $\mathbf{y}(u) = \mathbf{x}_{||}(u)$. Thus $\|\mathbf{x}_{||}\|^2 = D\|\mathbf{y}\|^2$. Using (3.2), we obtain

$$\begin{aligned}
\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{||}\right| &= \left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{Ax}_{||}\right| \\
&= \left|\sum_{u'=(u,i),v'=(v,j)} \mathbf{x}_{||}(u')\mathbf{A}(u',v')\mathbf{x}_{||}(v')\right| \\
&= D \cdot \left|\sum_{u,v} \mathbf{y}(u)\tilde{\mathbf{A}}_G\mathbf{x}_{||}(v)\right| \\
&= D \cdot \left|\mathbf{y}^{\mathrm{T}}\tilde{\mathbf{A}}_G\mathbf{y}\right| \\
&= D \cdot \tilde{\lambda}(G)\|\mathbf{y}\|^2
\end{aligned}$$

and noting that $\|\mathbf{y}\|^2 = \frac{1}{D}\|\mathbf{x}\|^2 \leq \frac{1}{D}$, we obtain

$$\left|\mathbf{x}_{||}^{\mathrm{T}}\mathbf{BABx}_{||}\right| \leq \tilde{\lambda}(G). \tag{3.6}$$

Plugging (3.4),(3.5),(3.6) into (3.3), the result follows. ∎

3.3 Constructing Expanders via the Zig Zag Road

In this section, we construct explicitly a family of expander graphs using the zig zag method. To apply the method we need an expander to start with. This is provided by the following important theorem, which we prove in section 3.4.

Theorem 3.4. *Let $p \in \mathbb{N}$ a prime. For every $t \leq p$ there is an explicit construction of a p^2-regular graph $G_{p,t}$ on p^{t+1} vertices such that $\tilde{\lambda}(G_{p,t}) \leq \frac{t}{p}$.*

Applying Theorem 3.4 with $p = 37$ and $t = 7$ yields a graph H such that:

1. H has 37^8 vertices.
2. H is 37^2-regular.
3. $\tilde{\lambda}(H) \leq \frac{7}{37} < \frac{1}{5}$.

Theorem 3.5. *There is a constant d such that for every $k \geq 0$ there exists an $(d^{4(k+1)}, d^2, \tilde{\lambda})$-expander with $\tilde{\lambda} \leq \frac{1}{2}$.*

Proof The expander graph is constructed inductively.

Set $G_0 = H^2$, where H is the graph described earlier in this section. Note that H is a $(d^4, d, \frac{1}{5})$ expander, where $d = 37^2 = 1369$. Trivially, G_0 is a d^2-regular on d^4 vertices and

$$\tilde{\lambda}(G_0) = \tilde{\lambda}(H) < \frac{1}{25} < \frac{1}{2}$$

Define $G_k = G_{k-1}^2 ⓩ H$ for each $k \geq 1$. We claim that the desired family is G_k.

The proof is by induction on k. Assume that the property holds for k, then G_k is a $(d^{4(k+1)}, d^2, \tilde{\lambda})$-expander such that $\tilde{\lambda}(G_k) \leq \frac{1}{2}$.

Note that G_k^2 is a d^4 regular graph so that the zig zag operation is well defined. Moreover, the number of vertices in the graph G_{k+1} are $d^{4(k+1)} \cdot d^4 = d^{4(k+2)}$. Finally, the zig zag method guarantees that

$$\tilde{\lambda}(G_{k+1}) \leq \tilde{\lambda}(G_k^2) + \tilde{\lambda}(H) + (\tilde{\lambda}(H))^2 < \frac{1}{4} + \frac{1}{5} + \frac{1}{25} < \frac{1}{2}$$ ∎

3.4 An Explicit Construction

In this section we provide a proof for Theorem 3.4. The proof is essentially the same as the one presented in [Rei04].

Proof (Theorem 3.4) We first describe the graph $G(p,t)$. To simplify notation, let $\mathbb{F} = \mathbb{F}_p$. The vertices of $G_{p,t}$ are defined to be $V = \mathbb{F}^{t+1}$. The set of neighbors of a vertex $v = (v_0, \ldots, v_{t+1})$ is the set of vertices $\{(v_0 + b, v_1 + ab, \ldots, v_t + a^t b) | a, b \in \mathbb{F}\}$. Clearly, the graph $G_{p,t}$ is p^2-regular on p^{t+1} vertices.

Next, we will prove that $\tilde{\lambda}(G_{p,t}) \leq \frac{t}{p}$. We first construct a set of $|V|$ orthonormal eigenvectors. To define the eigenvectors denote by ω the p-th root of unity. Then, by definition

$$1 + \omega + \ldots + \omega^{p-1} = 0$$

and $\omega^p = 1$. Note also that

$$\sum_{k=0}^{p-1} \omega^{kj} = 0 \tag{3.7}$$

for any $j \in \{1, \ldots, p-1\}$ since the function $f(k) = kj$ for any j such that $\gcd(j,p) = 1$ is a bijection in $\mathbb{F}$.

The set of eigenvectors is defined as follows.

$$\chi_\alpha(v) = \omega^{\sum_{j=1}^{t} \alpha_j v_j} \text{ for each } \alpha, v \in \mathbb{F}^{t+1}$$

Note that $\chi_a(b_1) \cdot \chi_a(b_2) = \chi_a(b_1 + b_2)$ and $\chi_{a_1}(b) \cdot \chi_{a_2}(b) = \chi_{a_1+a_2}(b)$. Moreover if $z*$ denotes the complex conjugate of z, $\omega^p = 1$ implies that $\chi_a^*(b) = \chi_{-a}(b)$.

We next prove that $\chi_a \perp \chi_b$ whenever $a \neq b$. Namely,

$$\begin{aligned}
\langle \chi_\alpha^*, \chi_\beta \rangle &= \sum_{v \in \mathbb{F}^{t+1}} \chi_\alpha^*(v) \chi_\beta(v) \\
&= \sum_{v \in \mathbb{F}^{t+1}} \chi_{-\alpha}(v) \chi_\beta(v) \\
&= \sum_{v \in \mathbb{F}^{t+1}} \chi_{\beta-\alpha}(v) \\
&= \sum_{v \in \mathbb{F}^{t+1}} \omega^{\sum_j (\beta_j - \alpha_j) v_j}
\end{aligned}$$

Observe that the last sum is the result if we write out the product

$$\prod_j \sum_{v_j} \omega^{(\beta_j - \alpha_j) v_j},$$

and noting that $\alpha_j \neq \beta_j$ for at least one index j, one of the factors in the last equation is 0, thus proving the claim.

Let $\tilde{\mathbf{A}}$ be the normalized adjacency matrix of $G_{p,t}$. We prove that each χ_α is an eigenvector of $\tilde{\mathbf{A}}$.

$$\begin{aligned}
\tilde{\mathbf{A}} \chi_\alpha(v) &= \sum_{c \in \mathbb{F}^{t+1}} \tilde{\mathbf{A}}(v, c) \chi_\alpha(c) \\
&= \frac{1}{p^2} \sum_{a,b} \chi_\alpha(v + (b, ab, \ldots, a^t b)) \\
&= \frac{1}{p^2} \sum_{a,b} \chi_\alpha(v) \chi_\alpha((b, ab, \ldots, a^t b)) \\
&= \frac{\chi_\alpha(v)}{p^2} \sum_{a,b} \chi_\alpha((b, ab, \ldots, a^t b))
\end{aligned}$$

and consequently the eigenvalue corresponding to χ_α is

$$\tilde{\lambda}_\alpha = \frac{1}{p^2} \sum_{a,b} \chi_\alpha((b, ab, \ldots, a^t b))$$

Thus, it suffices to prove that $|\tilde{\lambda}_\alpha| \leq \frac{t}{p}$ for every $\alpha \in \mathbb{F}^{t+1}$.

Assuming that $\alpha = (\alpha_0, \ldots, \alpha_t)$, we obtain

$$\begin{aligned}|\tilde{\lambda}_\alpha| &= \left|\frac{1}{p^2}\sum_{a,b}\chi_\alpha((b, ab, \ldots, a^t b))\right| \\ &= \frac{1}{p^2}\left|\sum_{a,b}\omega^{\alpha_0 b+\alpha_1 ab+\ldots+\alpha_t a^t b}\right|\end{aligned}$$

Denoting by $P_\alpha(x)$ the degree t polynomial $\alpha_0 + \alpha_1 t + \ldots + \alpha_t x^t$, we have that

$$\begin{aligned}\tilde{\lambda}_\alpha &= \frac{1}{p^2}\left|\sum_{a,b}\omega^{b\cdot P_\alpha(a)}\right| \\ &= \frac{1}{p^2}\left(\left|\sum_{a:P_\alpha(a)=0}\sum_{b}\omega^{b\cdot P_\alpha(a)}\right| + \left|\sum_{a:P_\alpha(a)\neq 0}\sum_{b}\omega^{b\cdot P_\alpha(a)}\right|\right)\end{aligned}$$

Observe that each sub sum in the sum

$$\sum_{a:P_\alpha(a)\neq 0}\sum_{b}\omega^{b\cdot P_\alpha(a)}$$

is equal to 0 by property (3.7), while each sub sum in the sum

$$\sum_{a:P_\alpha(a)=0}\sum_{b}\omega^{b\cdot P_\alpha(a)}$$

evaluates to p. Since P_α can have at most t roots, we finally obtain that

$$|\tilde{\lambda}_\alpha| \leq \frac{t}{p}$$ ∎

The above proof would work for any p a power of prime, as pointed out in [Rei04]. This leads to the following corollary which is vital for the Dinur's proof.

Corollary 3.6.
There exist $d_0 \in \mathbb{N}$ and $h_0 > 0$, such that there is a polynomial-time constructible family $\{X_n\}_{n\in N}$ of d_0-regular graphs X_n on n vertices with $h(X_n) \geq h_0$.

Proof (Dinur05) By Theorem 3.4 and the previous remark we can get expanders on 2^k vertices for any k. If $2^k < n < 2^{k+1}$ then we can merge $2^{k+1} - n$ vertices in $X_{2^{k+1}}$ and make the resulting graph regular by adding arbitrary edges to the non-merged vertices. Edge expansion is maintained up to a constant factor, thus the resulting graph is an expander. ∎

Chapter 4

Error Correction

The proof composition scheme is highly related to coding theory. In this respect, it is essential to study some of its aspects.

4.1 Code Theory Basics

Definition 4.1. *Given an alphabet Σ and two integers $k, n > 0$, a code is a function $C : \Sigma^k \to \Sigma^n$.*

Suppose that we are given a word a in Σ^k and a code C as in the definition. Then a is a *source word*, whereas $C(a)$ is a *code word*.

Any code should have the following properties.

1. The code is decodable, i.e. given a code word we can easily find the source of it. This implies that the code must be one-to-one and thus $n \geq k$.

2. The code is error correcting, i.e. if a code word is changed to some extent, we can still track the source word. This implies that two different codewords should differ as much as possible.

Instead of working with alphabets on letters, we consider only alphabets on a binary alphabet, i.e. $\Sigma = \{0, 1\}$. In order for such codes to be error correcting, it should be clear that two different codewords should differ in at least 3 places. To see this, if there exist codewords which differ in just one bit, then changing that bit would result into a legal codeword. Moreover, if there exist codewords which differ in exactly two places, then changing one of the two bits would not allow us to be sure which of the two code words was the initial code word.

The above observation leads naturally to our first example of a code.

Example 4.1.1.
Denote the binary string $a_1 \dots a_k$ by a. Then $C(a) = a_1 a_1 a_1 a_2 \dots a_{n-1} a_n a_n a_n$.

C is clearly a function from $\{0,1\}^k$ into $\{0,1\}^{3k}$. Clearly two distinct code-words differ in at least 3 places. This code can easily correct any single error. On the other hand, correcting two errors in a code word is clearly out of reach for this code. We could possibly increase the number of times each bit is duplicated so as to correct more errors, but this would clearly increase the length of a code word.

The above example indicates the importance of a code to be efficient and leads naturally to the notion of *informatio ratio*.

Definition 4.2. *The information ratio of a code is the fraction* k/n.

Clearly the information ratio should be as large as possible.

But we would also like to have a measure of the error correcting capabilities of the code. For this purpose, let us define the weight of a code word and the Hamming distance between two code words.

Definition 4.3. *For* $x \in \{0,1\}^r$, *the weight of* x, *denoted by* $\mathrm{wt}(x)$, *is the number of bits of* x *which are equal to 1, i.e.* $\mathrm{wt}(x) = |\{1 \leq i \leq r : \ x_i = 1\}|$.

Definition 4.4. *For* $x, y \in \{0,1\}^r$, *the Hamming distance between* x, y, *denoted by* $\mathrm{dist}(x,y)$ *is the number of bits in which* x, y *differ, i.e.* $\mathrm{dist}(x,y) = |\{1 \leq i \leq r : \ x_i \neq y_i\}|$.

We are now ready to define codes more explicitly.

Definition 4.5. *An* $(n,k,d)_\Sigma$*-code is a code* $C : \Sigma^k \to \Sigma^n$ *such that*

$$d = \min_{x \neq y} \mathrm{dist}(C(x), C(y)).$$

The parameter d *is the code distance. For an* (n,k,d)*-code* C, *the rate of* C *is defined as* $r = k/n$ *and the relative distance of* C *is defined as* $\delta = d/n$.

Thus the repetition code we saw earlier is a $(3k, k, 3)$-code with rate $r = 1/3$ and relative distance $\delta = 1/k$. Generalizing the 3 bit idea for a single error correcting code, it is easy to see that in order to be able to correct t bits a necessary and sufficient condition is that $2t < d$.

Moreover, it seems natural that to be able to correct many errors, the code word length should be sufficiently large. This intuition is affirmed by the following theorem, which demonstrates a trade off between r and δ. Before stating the theorem we introduce the ball around a word.

Definition 4.6. *For a word* $x \in \{0,1\}^r$, *the ball of radius* t *around* x *is defined as* $B(x,t) = \{z \in \{0,1\}^r : \mathrm{dist}(x,z) \leq t\}$.

We proceed into stating the bound mentioned earlier.

Theorem 4.7 (Hamming Bound). *For an $(n,k,d)_{\{0,1\}}$-code we have*

$$r + H\left(\frac{\delta}{2}\right) \leq 1,$$

where $H(p)$ denotes the entropy function $H(p) = p\log_2\left(\frac{1}{p}\right) + (1-p)\log\left(\frac{1}{1-p}\right)$.

Proof Consider the ball with radius $d/2$ around a code word x, $B(x, \lfloor \frac{d}{2} \rfloor)$. This ball should be disjoint from any other ball $B(y, \lfloor \frac{d}{2} \rfloor)$ since otherwise, the two code words would have distance $< d$. Thus, for each source word the corresponding balls are disjoint which means that

$$2^k \cdot \left| B\left(x, \frac{d}{2}\right) \right| \leq 2^n \tag{4.1}$$

But

$$|B(x,z)| = \binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{z} \approx 2^{n \cdot H(\frac{z}{n})}$$

Using the above approximation and taking logarithms in (4.1) we obtain

$$k + n \cdot H\left(\frac{d}{2n}\right) \leq n \iff r + H\left(\frac{\delta}{2}\right) \leq 1$$ ∎

4.2 Hadamard Code

We next define the Hadamard code.

Definition 4.8. *A Hadamard code is a function $H : \{0,1\}^k \to \{0,1\}^{2^k}$ such that, if $a \in \{0,1\}^k$ then for every $\beta \in \{0,1\}^k$ with binary the β-th component of $H[a]$ is equal to*

$$H(a)[\beta] =< a, \beta >= \sum_{i=0}^{k-1} a_i\beta_i \mod 2$$

where a_i, β_i denote the bits of a, β respectively.

It is easy to see that a Hadamard code is a linear code in the sense that for $a, b \in \{0,1\}^k$,

$$H(a) \oplus H(b) = H(a \oplus b)$$

In the above equation $\oplus$ represents the bitwise xor operation. Alternatively, $\oplus$ can be thought of

$$(a_1, \ldots, a_n) \oplus (b_1, \ldots, b_n) = (a_1 + b_1, \ldots, a_n + b_n)$$

where addition in the right part of the above equation is the usual addition in the Boolean field.

A different way to look at the Hadamard code is in terms of linear functions, i.e. $H[a]$ is the list of all possible outcomes of the linear function

$$H[a](x) = \sum_i a_i x_i$$

where again all operations are on the Boolean field.

Example 4.2.1.
Suppose $a = 101$. The Hadamard code of a is shown in the following table.

x_i	$H[a](x_i)$
000	0
001	1
010	0
011	1
100	1
101	0
110	1
111	0

4.2.1 The Parameters of Hadamard Code

The rate of a Hadamard code is logarithmic since $r = \frac{k}{2^k}$.

We would like to calculate the distance between two Hadamard code words. This is equivalent to calculating the minimum weight of $H(a) - H(b)$ for $a, b \in \{0,1\}^k, a \neq b$. Since the code is linear, it suffices to find the minimum weight of $H(a)$ where a has at least one nonzero bit.

Denote by a_j the first such non-zero bit and by $e_j = 2^{j-1}$ the word in $\{0,1\}^k$ with exactly one non-zero element in the j-th position. Then, for every number $0 \leq \beta \leq 2^k - 1$, using the linearity property, it holds that

$$< a, \beta >=< a, e_j > + < a, \beta + e_j >=< a, \beta + e_j > +1 \neq < a, \beta + e_j >$$

Consequently, there is a bijection between the nonzero elements of $H(a)$ to its zero elements. Thus $\text{wt}(H(a)) = 2^{k-1}$ and the relative distance of a Hadamard code is equal to

$$\delta = \frac{2^{k-1}}{2^k} = \frac{1}{2}$$

4.3 Local Testing

Suppose that we are given a set $P \subset \{0,1\}^n$, which we call a *property*. In this section we will get some insight into deciding whether a $x \in \{0,1\}^n$ belongs in P without looking at all bits of x. It should be clear that this question is interconnected with our ability to generate effective codes.

More formally, let us define local testing algorithms.

Definition 4.9 (Local Testing Algorithms). *Given a property P, a non deterministic polynomial algorithm A is a (q,ϵ)-Local Testing Algorithm (LTA) for P if for every $x \in \{0,1\}^n$, A computes indices $(i_1,\ldots,i_q) \in \{1,\ldots,n\}$ and according to a Boolean predicate $\Phi : \{0,1\}^q \to \{0,1\}$ outputs $\Phi(x_{i_1},\ldots,x_{i_q})$. A LTA has the following attributes:*

- $x \in P \Rightarrow \Pr[A(x)=1] = 1$.
- $x \notin P \Rightarrow \Pr[A(x)=0] \geq \epsilon \cdot \mathrm{dist}(x,P)$.

Remark 4.1.
From now on, $\mathrm{dist}(x,P)$ will denote the minimum *fraction* of bits needed to be changed in x so as to have property P.

We have intentionally left obscure the number of random bits a LTA uses. This is because it is not necessary all of the indices $(i_1,\ldots,i_q)$ to be randomly selected, since some indices might occur as a function on an already picked subset of the indices. Covering this possibility would lead to a much more complex definition, which for our purposes, is unnecessary since we are far more interested in the queries that an LTA makes than its randomness properties. At practice, we will examine LTAs for which $q = O(1)$. Hence, the random bits used in any case will be $O(\log n)$.

4.4 LTA for Hadamard Encodings

Blum, Ruby and Rubinfeld [BLR90] were the first to provide a local testing algorithm for Hadamard codes. The test is vital in both proofs of the PCP theorem. A stronger version was later proved by Bellare, Coppersmith, Håstad, Kiwi and Sudan [BCH$^+$95].

Denote by H to be the set of words in the Hadamard code, i.e.

$$H = \left\{ w \in \{0,1\}^{2^l} \,\middle|\, \exists a \forall x \left(x \in \{0,1\}^l, w(x) =< a,x > \right) \right\}$$

and consider the following algorithm.

Algorithm A (LTA for Hadamard Codes).
Test whether a given string is a legal Hadamard Code.

Input: A string $w \in \{0,1\}^{2^l}$.

Procedure:

Choose a random $x \in \{0,1\}^l$.

Choose a random $y \in \{0,1\}^l$.

Output:

Answer yes if $w(x) \oplus w(y) = w(x \oplus y)$.

Theorem 4.10. *Algorithm A satisfies:*

1. *If* $w \in P$ *then* $\Pr_{x,y}[A(w) = 1] = 1$.
2. *If* $\text{dist}(w, P) = \delta > 0$ *then* $\Pr_{x,y}[A(w) = 0] \geq \min\left(\frac{\delta}{2}, \frac{2}{9}\right)$.

Proof ([BLR90]) We have already proved 1 in section , so we proceed into proving 2. Consider $\hat{w}$ defined as

$$\hat{w}(x) = \arg\max_{a \in \{0,1\}} \Pr_y[w(x) \oplus w(x \oplus y) = a]$$

Set $P_x = \Pr_y[\hat{w}(x) = w(y) \oplus w(x \oplus y)]$. By definition of $\hat{w}$, we have that

$$P_x \geq \frac{1}{2} \text{ for all } x.$$

We prove next that

$$\Pr_{x,y}[A(w) = 0] \geq \frac{1}{2} \cdot \text{dist}(w, \hat{w}) \tag{4.2}$$

Set $B = \left\{x \in \{0,1\}^l : \Pr_y[w(x) \neq w(y) \oplus w(x \oplus y)] > \frac{1}{2}\right\}$. If $x \notin B$ then $\hat{w}(x)$ is assigned the value $w(x)$ and consequently $\Pr_x[w(x) = \hat{w}(x) | x \notin B] = 0$. Thus

$$\text{dist}(w, \hat{w}) = \Pr_x[w(x) = \hat{w}(x) | x \in B] \Pr_x[x \in B] \leq \Pr_x[x \in B] \tag{4.3}$$

If $x \in B$ then Algorithm A rejects by definition with probability greater than $\frac{1}{2}$ and therefore, using Bayes formula,

$$\Pr_{x,y}[A(w) = 0] \geq \Pr_y[A(w) = 0 | x \in B] \Pr_x[x \in B] = \frac{1}{2} \cdot \Pr_x[x \in B] \tag{4.4}$$

Combining (4.3) and (4.4), we obtain (4.2). Notice that it remains to prove that

$$\Pr_{x,y}[A[w] = 0] < \frac{2}{9} \Rightarrow \hat{w} \in H,$$

since then it must be the case that $\text{dist}(w, \hat{w}) \geq \delta$.

Hence, assume that $\Pr_{x,y}[A[w] = 0] < \frac{2}{9}$. We first prove that

$$P_x \geq \frac{2}{3} \text{ for all } x.$$

Choose y_1 and y_2 independently and let

$$P'_x = \Pr_{y_1,y_2}[w(y_1) \oplus w(x \oplus y_1) = w(y_2) \oplus w(x \oplus y_2)]$$

Considering the case where both expressions equal $\hat{w}(x)$ or $1 - \hat{w}(x)$, we obtain that

$$P'_x = P_x^2 + (1 - P_x)^2 \tag{4.5}$$

Moreover P'_x can also be written as

$$P'_x = \Pr_{y_1,y_2}[w(y_1) \oplus w(x \oplus y_2) = w(y_2) \oplus w(x \oplus y_1)]$$

Observe that both $x+y_1, x+y_2$ are randomly distributed on $\{0,1\}^l$ and therefore, by assumption,

$$\Pr_{y_1,y_2}[\underbrace{w(y_1) \oplus w(x \oplus y_2) = w(x \oplus y_1 \oplus y_2)}_{E_1}] > 1 - \frac{2}{9}$$

$$\Pr_{y_1,y_2}[\underbrace{w(y_2) \oplus w(x \oplus y_1) = w(x \oplus y_1 \oplus y_2)}_{E_2}] > 1 - \frac{2}{9}$$

Thus

$$P'_x \Pr_{y_1,y_2}[E_1 \cap E_2] = \Pr_{y_1,y_2}[E_1] + \Pr_{y_1,y_2}[E_2] - \Pr_{y_1,y_2}[E_1 \cup E_2] > 2 - \frac{4}{9} - 1 = \frac{5}{9} \tag{4.6}$$

Since $x^2 + (1-x)^2$ is increasing for $x \geq \frac{1}{2}$, (4.5) and (4.6) give

$$P_x^2 + (1 - P_x)^2 > \frac{5}{9} \Longrightarrow P_x > \frac{2}{3}$$

We are now ready to prove that $\hat{w} \in H$, i.e.

$$\forall x \forall z \in \{0,1\}^l : \ \hat{w}(x) \oplus \hat{w}(z) = \hat{w}(x \oplus z)$$

Note that

$$P_x = \Pr_y[\hat{w}(x) = w(y) \oplus w(x \oplus y)] > \frac{2}{3}$$
$$P_z = \Pr_y[\hat{w}(z) = w(y) \oplus w(z \oplus y)] > \frac{2}{3}$$
$$P_{x\oplus z} = \Pr_y[\hat{w}(x \oplus z) = w(y \oplus z) \oplus w(\underbrace{x \oplus z \oplus y \oplus z}_{x\oplus z})] > \frac{2}{3}$$

where the last equation follows by observing that $y \oplus z$ has the same distribution as y. The three equations above tell us that there must be a y satisfying all of the in bracket expressions (otherwise the union of the above probabilities would be ≤ 1).

The result now comes for free, since using this common y, we have that

$$\hat{w}(x)\oplus\hat{w}(y) = w(y)\oplus w(x\oplus y)\oplus w(y)\oplus w(y\oplus z) = w(x\oplus y)\oplus w(y\oplus z) = \hat{w}(x\oplus z) \quad \blacksquare$$

Remark 4.2.
Since the Hadamard code can be thought as the encoding of a linear function, we will often refer to Algorithm A as a *linearity testing* algorithm.

4.5 Local Decodability

Hadamard codes are not only locally testable but also locally decodable. A Locally decodable code, in a similar fashion with locally testable codes, are codes such that we can find with high probability a specific bit of the source word of an input codeword without reading all the bits in the latter. We will study the notion of locally decodability in terms of Hadamard codes.

Theorem 4.11 (Local Decodability of Hadamard code).
Let $H = \{H(a) | a \in \{0,1\}^l\}$ *and* $w \in \{0,1\}^{2^l}$ *such that* $\text{dist}(w, H) \leq \epsilon$. *Let* $z \in H$ *such that* $\text{dist}(w, z) \leq \epsilon$ *and suppose that an index* $x \in \{0,1\}^l$ *be given. Then we can find a bit* b *such that*

$$\Pr[b = z(x)] \geq 1 - 2\epsilon$$

We first give the algorithm and then analyze its correctness.

Algorithm B.
Given $w \in \{0,1\}^{2^l}$: $\mathrm{dist}(w, P) \leq \epsilon$ *and* $x \in \{0,1\}^l$ *find* b *such that* $\Pr[b = z(x)] \geq 1 - 2\epsilon$.

Input: A string $w \in \{0,1\}^{2^l}$.

An index $x \in \{0,1\}^l$.

Procedure:

Choose a random $y \in \{0,1\}^l$.

Output:

Return $b = w(y) \oplus w(x \oplus y)$.

Let $E_y = \Pr_y[w(y) = z(y)]$ and similarly $E_{x \oplus y} = \Pr_y[w(x \oplus y) = z(x \oplus y)]$. Since y and $x \oplus y$ follow the same distribution, by our assumptions we have that

$$E_y \geq 1 - \epsilon \text{ and } E_{x \oplus y} \geq 1 - \epsilon$$

Since z is a legal Hadamard codeword we have that

$$\Pr_y[b = z(x)] \geq E_y \cdot E_{x \oplus y} \geq (1 - \epsilon)^2 \geq 1 - 2\epsilon$$

and the claim follows.

Remark 4.3.
Algorithms such as Algorithm B which are capable of returning with high probability a legal Hadamard bit and, as an extension, a legal Hadamard codeword are often referred to as Self Correcting algorithms. The result presented above is a special case of the test appearing in [AS92], [ALM+92] for decoding degree d polynomials.

Chapter 5

Dinur's Proof of the PCP theorem

In this chapter, we present a step-by-step proof of the PCP theorem. Unless stated otherwise, we follow [Din05].

5.1 Constraint Satisfaction Problems

In this section we define constraint satisfaction problems. Consider a finite set of symbols Σ called the alphabet and $V = \{v_1, \ldots, v_n\}$ a set of variables.

Definition 5.1 (Constraint). *A q-ary constraint is a tuple $(i_1, \ldots, i_q, \phi)$ such that $i_j \in [n]$ and $\phi : \Sigma^q \to \{0,1\}$. A constraint $(i_1, \ldots, i_q, \phi)$ is satisfied by an assignment $a : V \to \Sigma$ iff $\phi(a(v_{i_1}), \ldots, a(v_{i_q})) = 1$.*

Definition 5.2 (Constraint Satisfaction Problems). *Given as input an alphabet Σ, a set of variables $V = \{v_1, \ldots, v_n\}$, and a set of q-ary constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$, denote by CSP$[q, \Sigma]$ the problem of finding an assignment $a : V \to \Sigma$ which satisfies all of the constraints.*

Numerous known problems can be formulated easily as constraint satisfaction problems.

Example 5.1.1 (3SAT as a CSP).
Consider a 3SAT formula $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m$ over a set of boolean variables $V = \{x_1, \ldots, x_n\}$, where each ϕ_i is of the form $x'_k \vee x'_l \vee x'_m$ (x'_k is either x_k or $\neg x_k$). Then a CSP equivalent to this formula can be constructed as follows.

- $\Sigma = \{0, 1\}$.
- $V = \{x_1, \ldots, x_n\}$.

- (i, j, k, ϕ) is a constraint iff there exists ϕ_i such that $\phi_i \equiv x'_k \vee x'_l \vee x'_m$ and $\phi(a, b, c) = a \vee b \vee c$.

Example 5.1.2 (3COL as a CSP).
Consider the decision problem of whether a graph $G = (V, E)$ is 3-colorable, i.e. whether there exists an assignment $a : V \to \{1, 2, 3\}$ such that if $(u, v) \in E$ then $a(u) \neq a(v)$. A CSP instance for this problem can be constructed as follows.

- $\Sigma = \{1, 2, 3\}$.
- $V = \{v_1, \ldots, v_n\}$.
- (i, j, ϕ) is a constraint iff $(v_i, v_j) \in E$ and $\phi : \{1, 2, 3\}^2 \to \{0, 1\}$ such that $\phi(a, b) = 1$ if and only if $a \neq b$.

Following the 3COL formulation as a CSP, we observe that CSP instances with $q = 2$ can be better perceived in terms of graphs.

Definition 5.3 (Constraint Graph Problems). *A constraint graph satisfaction problem is an instance $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ such that (V, E) is a graph, the alphabet Σ is a finite set and $\mathcal{C} : E \to \{\phi : \Sigma \times \Sigma \to \{0, 1\}\}$.*

The notion of maximum satisfiability problems naturally leads to introducing the satisfaction value of a CSP. Namely, we denote by $\mathrm{SAT}_a(\mathcal{C})$ the fraction of constraints in $\mathcal{C}$ satisfied by the assignment a.

Definition 5.4 (Max-CSP). *Given as input an alphabet Σ, a set of variables $V = \{v_1, \ldots, v_n\}$, and a set of q-ary constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$, denote as Max-CSP$[q, \Sigma]$ the problem of finding an assignment $a : V \to \Sigma$ which maximizes the fraction of satisfied constraints. Define the sat value of $\mathcal{C}$ as*

$$\mathrm{SAT}(\mathcal{C}) = \max_{a:V\to\Sigma} \Pr_{i\in 1,\ldots,m} [a \text{ satisfies constraint } c_i]$$

In other words, the SAT *value of a CSP is the fraction of constraints satisfied by an optimal assignment. Finally, define the unsatisfaction value of $\mathcal{C}$ as*

$$\textsc{Unsat}(\mathcal{C}) = 1 - \mathrm{SAT}(\mathcal{C}),$$

standing for the minimum fraction of constraints which are left unsatisfied by an optimal assignment.

Clearly, the examples demonstrated previously provide reductions from Max-3SAT to Max-CSP$[q = 3, |\Sigma| = 2]$ and from Max-3COL to Max-CSP$[q = 2, |\Sigma| = 3]$. Thus the corresponding instances of Max-CSP are NP-Hard problems.

5.2 PCP theorem and Hardness of Approximation

We next introduce the gap version of CSP.

Definition 5.5 (Gap-CSP). *Given as input an alphabet Σ, a set of variables $V = \{v_1, \ldots, v_n\}$, and a set of q-ary constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$ and $0 < s < 1$, Gap-CSP$_s[q, \Sigma]$ is the problem of deciding between the following two possibilities:*

- $\mathrm{SAT}(\mathcal{C}) = 1$.
- $\mathrm{SAT}(\mathcal{C}) \leq s$.

The next theorem will reduce the PCP theorem to proving a hardness of approximation result.

Theorem 5.6. *The following claims are equivalent.*

1. $\mathrm{NP} \subseteq \mathrm{PCP}[O(\log n), O(1)]$.
2. *There exist constants q, $0 < s < 1$, and $|\Sigma|$ such that Gap - CSP$_s[q, \Sigma]$ is* NP*-hard.*

Proof
$(1) \Longrightarrow (2)$

Assuming that $\mathrm{NP} \subseteq \mathrm{PCP}[O(\log n), O(1)]$, for any NP language there is a $[O(\log n), O(1)]$-verifier. Consider such a verifier V for 3COL which uses $c \log n$ random bits and $k = O(1)$ queries into the proof. We will reduce 3COL to an instance of $\mathrm{gap} - \mathrm{CSP}_s[q, \Sigma]$ for an appropriate selection of s, q, Σ. Then, any $L \in \mathrm{NP}$ has a reduction to $\mathrm{gap} - \mathrm{CSP}_s[q, \Sigma]$ since 3COL is NP-hard.

The main idea of the proof is to build a constraint for each of the possible runs of the verifier, thus simulating its function.

- V reads the input graph G, reads the random binary string ρ of length $r = O(\log n)$ and computes the indices $\{i_1, \ldots, i_q\}$ which will query into the proof y. Then according to the values $\{y[i_1], \ldots, y[i_k]\}$ it outputs whether it accepts or not.
- Clearly, the possible random strings are $2^{O}(\log n) = n^c$ and consequently the number of distinct queries into the proof is at most $k \cdot n^c = O(n^c) = n^d$ (c, d constants). We introduce variables $y_1, \ldots, y_{n^d}$ for each of the possible distinct queries.
- For each random string ρ simulate all the $2^k = O(1)$ possible assignments to $y[i_1], \ldots, y[i_k]$ and construct a boolean predicate ϕ which evaluates to 1 if and only if the verifier accepts. Thus, build a constraint $(i_1, i_2, \ldots, i_k, \phi)$. Consider the set of all such constraints.

Denote by C_G the above instance of CSP. If the graph G is 3-colorable then there exists a proof y such that the verifier accepts and consequently the CSP in the above construction satisfies $\mathrm{SAT}(\mathrm{C_G}) = 1$. On the other hand, if G is not 3-colorable, then for every proof y the verifier accepts with probability $< 1/2$ and thus $\mathrm{SAT}(\mathrm{C_G}) \leq 1/2$.

It follows that Gap-$\mathrm{CSP}_{1/2}[q = k, |\Sigma| = 2]$ is NP-hard.

(2) $\Longrightarrow$ (1)

(2) states that for any $L \in NP$, it is possible to map any instance x of the language L to an instance C_x of gap-$\mathrm{CSP}_s[q, \Sigma]$.

We build a PCP verifier for L as follows: given an instance x map it to C_x and then for a potential assignment a, pick one of C_x's constraints at random and accept if that constraint is satisfied by a. Otherwise, reject.

More formally, consider an instance of an NP-hard language, say 3COL. Then, it suffices to prove that 3COL has a PCP verifier since any other language in NP can be mapped polynomially into an instance of 3COL.

Given an input x for 3COL and a proof y, the verifier V for 3COL works as follows.

1. Use the polynomial reduction to gap-$\mathrm{CSP}_s[q, \Sigma]$ to get an instance C_x. C_x is polynomially related to the size of the input x. Denote by m the number of constraints in C_x. If $|x| = n$ then $m = \mathrm{poly(n)}$.

2. V uses a random binary string ρ of length $\lceil \log m \rceil$ to pick a random constraint in C_x, say $c_j = (i_1, \ldots, i_q, \phi)$. Note that $\log m = O(\log n)$.

3. V queries the bits corresponding to $y[i_1], \ldots, y[i_q]$ and then outputs according to $\phi(y[i_1], \ldots, y[i_q])$. Note that each of the $y[i_j]$ is encoded by $\log |\Sigma| = O(1)$ bits, since $|\Sigma|$ is constant.

From the above construction, V is a $[O(\log n), O(1)]$-verifier. Moreover it runs in polynomial time. Next, we prove its soundness and correctness.

- If $x \in L$ then $\mathrm{SAT}(C_x) = 1$, so there is a proof y such that the verifier always accepts (the encoding of the optimal assignment).

- If $x \notin L$ then $\mathrm{SAT}(C_x) \leq s$, thus for every proof y

$$\Pr_{\rho \in \{0,1\}^{O(\log n)}} [V^y(x, \rho) = 1] \leq s.$$

If $s > 1/2$, in order to enhance the soundness parameter, repeat steps 2, 3 $\lceil \log_s \frac{1}{2} \rceil$ times. Since s is a constant, a constant number of repetitions is required and consequently the verifier still uses $O(\log n)$ random bits and $O(1)$ queries into the proof. ∎

Remark 5.1.
It is important to stress the importance of having constant values for $s, q, |\Sigma|$. The reader should go through the proof of Theorem 5.6 again and clarify this before proceeding into the next sections.

5.3 The PCP theorem by Gap Amplification

In the previous section we proved that the PCP theorem is equivalent to proving the following theorem.

Theorem 5.7 (Gap-CSP Hardness). *There exist constants q, $0 < s < 1$, and $|\Sigma|$ such that Gap - $CSP_s[q, \Sigma]$ is* NP*-hard.*

We will restrict ourselves to constraint graph problems. As we saw in section 1.1 there is a trivial reduction which transforms an instance $G = (V, E)$ of 3COL to a constraint graph C_G. Note that:

- G is 3-colorable if $\textsc{Unsat}(C_G) = 0$.
- G is not 3-colorable if $\textsc{Unsat}(C_G) \geq \frac{1}{|E|}$.

The latter claim just states that at least one edge constraint should be left unsatisfiable. Observe that, ideally, if the term $\frac{1}{|E|}$ was a constant the Gap-CSP hardness would come for free. Our goal will be to amplify the above gap to a constant γ. The following theorem is vital toward this objective.

Theorem 5.8 (Gap Amplification). *There exists Σ_0 such that for all Σ there exist constants c and $0 < \gamma < 1$ and a polynomial reduction which given a constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ outputs $G' = \langle (V', E'), \Sigma_0, \mathcal{C}' \rangle$ such that*

1. $|V'| + |E'| \leq c(|V| + |E|)$.
2. [Completeness] : $\textsc{Unsat}(G) = 0 \Rightarrow \textsc{Unsat}(G') = 0$.
3. [Soundness] : $\textsc{Unsat}(G) > 0 \Rightarrow \textsc{Unsat}(G') \geq \min\{\gamma, 2 \cdot \textsc{Unsat}(G)\}$.

Assuming the Gap Amplification theorem to be true, one can prove that Gap-$\text{CSP}_\gamma[q = 2, \Sigma_0]$ is NP-hard.

Theorem 5.9. *Let $G = (V, E)$ denote some graph. There exists a polynomial reduction from 3COL to Gap-$CSP_\gamma[q = 2, \Sigma_0]$ which maps G into a constraint graph G' such that:*

- [Completeness] : $G \in \text{3COL} \Rightarrow \textsc{Unsat}(G') = 0$.
- [Soundness] : $G \notin \text{3COL} \Rightarrow \textsc{Unsat}(G') \geq \gamma$.

Remark 5.2.
Note that the constants γ and Σ_0 appearing in the theorem are the ones guaranteed by the Gap Amplification theorem.

Proof To see this, take the reduction from 3COL to the constraint graph instance $C_G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ which we presented earlier. Apply Gap Amplification $k = \log |E| + 1$ times to obtain a sequence of constraint graphs $G_1, G_2, \ldots, G_k$. We claim that the constraint graph $G_k = \langle (V_k, E_k), \Sigma, \mathcal{C}_k \rangle$ is an instance of Gap-CSP$_\gamma[q = 2, \Sigma_0]$.

- $|V_k| + |E_k| \leq c^k(|V| + |E|)$. This shows that G_k is only polynomially bigger than the original constraint graph since $c^k = c^{\log |E|+1} = O(|E|^{\log c})$.
- $\textsc{Unsat}(G) = 0 \Rightarrow \textsc{Unsat}(G_k) = 0$, since Gap Amplification preserves completeness.
- $\textsc{Unsat}(G) > 0 \Rightarrow \textsc{Unsat}(G_k) \geq \gamma$. To see this, recall that if the initial graph is not satisfiable then $\textsc{Unsat}(G) \geq \frac{1}{|E|}$. If $\textsc{Unsat}(G_i) \geq 2 \cdot \textsc{Unsat}(G_{i-1})$ for each $i \leq k$ then

$$\textsc{Unsat}(G_k) \geq 2^k \cdot \textsc{Unsat}(G) = 2^k \cdot \frac{1}{|E|} > |E| \cdot \frac{1}{E} = 1,$$

  which is absurd. Thus, for some index i less than k, it must be the case that $\textsc{Unsat}(G_i) \geq \gamma$ which proves that $\textsc{Unsat}(G_k) \geq \gamma$. ∎

It should be clear that Theorem 5.7 follows in a straighforward manner from Theorem 5.9.

In the rest of this chapter, we will concentrate on proving the Gap Amplification theorem.

5.4 Overview of the Gap Amplification Proof

The Gap Amplification theorem will follow in 3 steps.

1. In the first step, we transform the constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ into another one G' such that the underlying graph of this new instance is an expander. Alphabet Σ remains the same and the unsatisfaction level decreases by a constant. Despite this, the new graph has much better combinatorial properties. This step can be called as the preprocessing step since its main goal is to beautify the graph.
2. In the second step, which is the main gap amplification procedure, G' is transformed into G'' such that the gap amplifies by an arbitrary constant t. This would be enough if the alphabet of G'' remained invariant. Unfortunately, the alphabet increases exponentially on the constant t, which leads to the third step.

3. Proof Composition. In this step, the alphabet becomes constant and the unsatisfaction level decreases only by a constant.

As we shall see, the constants by which the unsatisfaction level decreases in steps 1 and 3 are well defined and thus can easily be compensated from step 2.

5.5 Preprocessing Step

The preprocessing step consists of two substeps.

1. In the first substep, we transform the graph into a d-regular one for some constant d.

2. In the second substep, we transform the d-regular graph into an expander.

5.5.1 Regularization

The main result that we use is that there is an (n, d, h_0)-expander family for some constants d, h_0 for each $n \in \mathbb{N}$, as established in Corollary 3.6. The construction follows.

Definition 5.10. *Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $G' = \mathrm{PREP}_1(G) = \langle (V', E'), \Sigma, \mathcal{C}' \rangle$ is constructed as follows.*

- *Vertices in V': for each vertex $v \in V$ introduce $\deg_G(v)$ nodes in V', representing each of the edges incident to v. Denote these set of new variables as $[v]$. To picture this better, each vertex v in G is replaced by a smaller graph on $\deg_G(v)$ vertices. We will refer to such a set as v's cloud. Define $V' = \bigcup_{v \in G} [v]$.*

- *Edges in E': we will have two sets of edges, E_G and E_c. The set E_G denotes edges among clouds and will be in one-to-one correspondence with edges of G, while the set E_c denotes edges within clouds.*

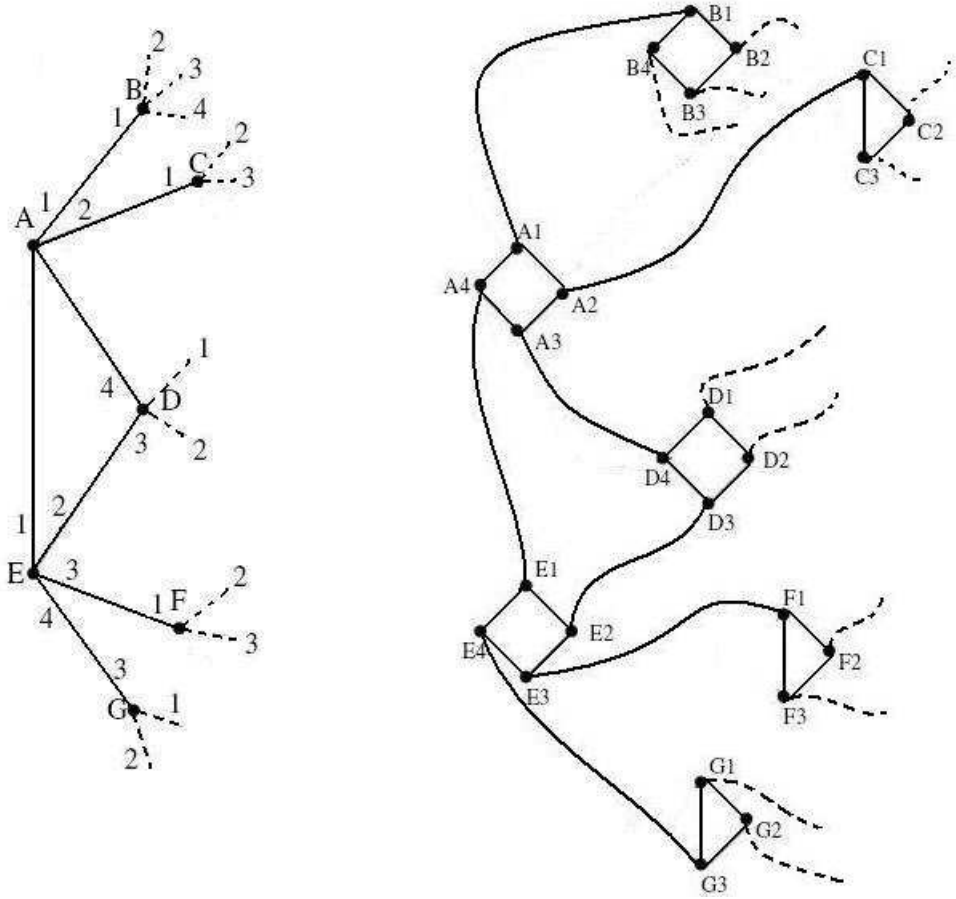
    1. *E_G: Consider the cloud of $[v]$ nodes of a vertex v in G. Then each of the new nodes is adjacent to one edge of the initial graph G. In other words, suppose that $u_i \in [u]$ and $v_j \in [v]$. Then $(u_i, v_j) \in E_1$ iff v is the i-th neighbor of u in G and u is the j-th neighbor of v. The set E_G is the union of all such edges between any two clouds.*

    2. *E_c: Consider the cloud of a vertex v. Then among the $[\deg(v)]$ nodes add edges such that the induced subgraph of these nodes is a $(\deg_G(v), d, c)$-expander. Note that such an expander exists. E_c is the union of all such edges within any cloud.*

- *Constraints: each edge in E_G will preserve its original constraint in G, while every edge in E_c will try to enforce an identical assignment. Thus,*

$$\mathcal{C}'(e \in E_c)(a,b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{otherwise} \end{cases}$$

An example of the new constraint graph is illustrated in the following figure, where instead of a d-regular expander, we use a (2-regular) cycle.



It should be clear that $\text{PREP}_1(G)$ is a $(d+1)$-regular graph, where d is a constant. The main theorem we prove for the above construction follows.

Theorem 5.11. *Let $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The reduction $G \to G' = \text{PREP}_1(G) = \langle (V',E'), \Sigma, \mathcal{C}' \rangle$ is such that*

1. *$|V'| + |E'| \leq c(|V| + |E|)$ for some constant c.*
2. *$\text{UNSAT}(G) \cdot \beta_1 \leq \text{UNSAT}(G') \leq \text{UNSAT}(G)$ for a constant $0 < \beta_1 < 1$.*

Remark 5.3.
Note that the second property implies that the unsatisfaction level of G' decreases only by a constant when G is not satisfiable, while if G is satisfiable then G' is too.

Proof First of all, note that $|V'| = \sum_v \deg_G(v) = 2|E|$ and $|E'| = (d+1)|E|$ and since d is a constant, 1 follows.

Next, we prove that $\textsc{Unsat}(G) \geq \textsc{Unsat}(G')$. To see this, denoting by $g : V \to \Sigma$ the optimal assignment in G, we construct an assignment for G' such that all vertices in a cloud $[v]$ are assigned the value $g(v)$. Then, it clearly holds that

$$\textsc{Unsat}(G') \leq \textsc{Unsat}_{g'}(G') = \frac{\textsc{Unsat}_g(G) \cdot |E|}{|E'|} \leq \textsc{Unsat}(G)$$

which is what we wanted to prove.

Next, we prove that $\textsc{Unsat}(G) \cdot \beta_1 \leq \textsc{Unsat}(G')$ for some constant β_1. This is quite more complicated. Note that constructing an assignment in G' from an assignment in G will not help since this proves the reverse inequality. Thus, we need to construct an assignment in G from an assignment in G'.

Given an assignment $g' : V' \to \Sigma$ in G', define $g(v)$ for a vertex $v \in G$ to be the majority value in v's cloud, i.e.

$$g(v) = \arg \max_{u \in [\deg(v)]} g'(u)$$

To aid the analysis below, consider U, U' the set of edges left unsatisfied by g in G and g' in G' correspondingly. Moreover, for a vertex v denote by $S^{[v],\sigma}$ the nodes in v's cloud which are assigned to $\sigma \in \Sigma$ and do not agree with the majority decision, i.e.

$$S^{[v],\sigma} = \{u \in [\deg(v)] : g'(u) = \sigma,\ \sigma \neq g(v)\}$$

Denote by $S^{[v]} = \bigcup_{\sigma \in \Sigma} S^{[v],\sigma}$ and $S = \bigcup_{v \in V} S^{[v]}$.

Let us consider an edge $e = (u, v) \in U$. We will denote by $e' = (u_i, v_j)$ the corresponding edge in E'. Then there are two possibilities.

- Either e' is unsatisfied in H too.
- Either e' is satisfied in H and at least one of u_i, v_j disagree with the majority assignment.

Thus

$$|U| \leq |U'| + |S|. \tag{5.1}$$

Note that $\textsc{Unsat}_g(G) = \frac{|U|}{|E|}$ and $\textsc{Unsat}_{g'}(G') = \frac{|U'|}{|E'|}$. Therefore, if $|U'| > |U|/2$ it follows rather easily that

$$\textsc{Unsat}_{g'}(G') = \frac{|U'|}{|E'|} \geq \frac{|U|}{2(d+1)|E|} = \frac{\textsc{Unsat}_g(G)}{2(d+1)} \geq \frac{1}{2(d+1)} \cdot \textsc{Unsat}(G)$$

and consequently the same holds for the optimal assignment g'.

Consider now the case $|U'| \leq |U|/2$. From (5.1) it follows that $|S| \geq |U|/2$. Note that $S^{[v],\sigma}$ contains at most $\deg(v)/2$ vertices since g is the majority assignment. Since each cloud is an expander with edge expansion h each $S^{[v],\sigma}$ has (within the cloud) at least $h \cdot |S^{[v],\sigma}|$ edges going out of it. These edges are clearly unsatisfied. Being wary not to count twice the same edge, summing over all $\sigma \in \Sigma$ and all clouds we get a bound on U', namely

$$|U'| \geq \frac{1}{2} \sum_{v \in V} \sum_{\sigma \in \Sigma} h_0 \cdot \left| S^{[v],\sigma} \right| = \frac{h_0}{2}|S| \geq \frac{h}{4}|U|$$

Thus

$$\textsc{Unsat}_{g'}(G') = \frac{|U'|}{|E'|} \geq \frac{h_0|U|}{4(d+1)|E|} = \frac{h_0 \cdot \textsc{Unsat}_g(G)}{4(d+1)} \geq \frac{h_0}{4(d+1)} \cdot \textsc{Unsat}(G)$$

and the same holds for the optimal assignment in G'.

We can therefore derive that

$$\textsc{Unsat}(G) \cdot \beta_1 \leq \textsc{Unsat}(G') \leq \textsc{Unsat}(G),\ \beta_1 = \min\left\{ \frac{1}{2(d+1)}, \frac{h_0}{4(d+1)} \right\} \blacksquare$$

Concluding, the $\textsc{Prep}_1$ step turned the original graph G into a d-regular graph, decreasing the unsatisfaction value only by a constant β_1.

5.5.2 Expanderizing

The second step will turn the output of the $\textsc{Prep}_1$ step into an expander. Again, the main result that we use is Corollary 3.6, that there is an (n, d, h_0)-expander family for some constants d, h_0 for each $n \in \mathbb{N}$.

Definition 5.12. *Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a d-regular constraint graph. The constraint graph $G' = \textsc{Prep}_2(G) = \langle (V', E'), \Sigma, \mathcal{C}' \rangle$ is constructed as follows.*

- $V' = V$.
- *Edges in E': consider a d'-regular expander graph $X = (V, E_X)$. Then $E' = E \cup E_X$, where multiple edges between two vertices are allowed.*
- *Constraints: each edge in E preserves its original constraint in G, while every edge in E_X has null constraints, i.e. it is satisfied by any assignment to its endpoints.*

Remark 5.4.
G' is $(d+d')$-regular.

The next theorem illustrates the properties of the above construction.

Theorem 5.13. *Let $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The reduction $G \to G' = \text{PREP}_1(G) = \langle (V',E'), \Sigma, \mathcal{C}' \rangle$ is such that*

1. $\lambda(G') < \lambda + d$.
2. $|V'| + |E'| \leq c(|V| + |E|)$ *for some constant* c.
3. $\text{UNSAT}(G) \cdot \beta_2 \leq \text{UNSAT}(G') \leq \text{UNSAT}(G)$ *for some constant* $0 < \beta_2 < 1$.

Proof Note that $|E'| = \frac{(d+d')|V|}{2}$, therefore the new graph is only bigger by a constant since both d, d' are constants.

It should be obvious that $\text{UNSAT}(G') \leq \text{UNSAT}(G)$ since the optimal assignment for G clearly satisfies the same number of edges in G' and leaves a smallest fraction of the constraints unsatisfied since G' has additional trivial constraints.

The same argument works pretty much for bounding $\text{UNSAT}(G')$. Namely, suppose that $g' : V' \to \Sigma$ an assignment in G' which satisfies S edges of E', then clearly $S \subset E$ and S is exactly the set of vertices left unsatisfied by g' in G'. Thus

$$\text{UNSAT}_{g'}(G') = \frac{|S|}{|E|} = \frac{2|S|}{(d+d')V} = \frac{d}{d+d'} \cdot \frac{2|S|}{dV} = \frac{d}{d+d'} \text{UNSAT}_{g'}(G),$$

which clearly leads to

$$\text{UNSAT}(G') \geq \frac{d}{d+d'} \text{UNSAT}(G)$$

and therefore

$$\text{UNSAT}(G) \cdot \beta_2 \leq \text{UNSAT}(G') \leq \text{UNSAT}(G), \ \beta_2 = \frac{d}{d+d'}.$$

Thus, it is left to prove that $\lambda(G') < \lambda + d$. Note that on a intuitive level, G' will certainly be an expander since adding edges to an expander clearly cannot reduce the edge expansion property.

Formally, denote by $\mathbf{A}_{G'}, \mathbf{A}_G, \mathbf{A}_X$ the adjacency matrices of G', G, X respectively. Then, it is clear that $\mathbf{A}_{G'} = \mathbf{A}_G + \mathbf{A}_X$ and recall that by the Rayleigh

quotient we have that

$$\begin{aligned}\lambda(G') &= \max_{\|\mathbf{x}\|=1,\mathbf{x}\perp\mathbf{1}} \left|\mathbf{x}^T\mathbf{A}_{G'}\mathbf{x}\right| \\ &= \max_{\|\mathbf{x}\|=1,\mathbf{x}\perp\mathbf{1}} \left|\mathbf{x}^T(\mathbf{A}_G + \mathbf{A}_X)\mathbf{x}\right| \\ &\leq \max_{\|\mathbf{x}\|=1,\mathbf{x}\perp\mathbf{1}} \left|\mathbf{x}^T\mathbf{A}_G\mathbf{x}\right| + \max_{\|\mathbf{x}\|=1,\mathbf{x}\perp\mathbf{1}} \left|\mathbf{x}^T\mathbf{A}_X\mathbf{x}\right| \\ &= \lambda(G) + \lambda(X)\end{aligned}$$

and the claim follows. ∎

Summarizing the second step, we have succeeded in turning the graph into an expander.

5.5.3 Combining the two Preprocessing Steps

Let H be the output graph if we apply the two transformations in chain, namely

$$G \to \textsc{Prep}_1(G) \to \textsc{Prep}_2(\textsc{Prep}_1(G)) = H.$$

Combining Theorem 5.11 and Theorem 5.13, we derive that

$$\textsc{Unsat}(G) \cdot \beta_1\beta_2 \leq \textsc{Unsat}(H) \leq \textsc{Unsat}(G),$$

where $\beta_1,\ \beta_2$ are constants. We may finalize the preprocessing step with the following theorem.

Theorem 5.14. *Let $G = \langle (V, E), \Sigma, \mathcal{C}\rangle$ be a constraint graph. Then the constraint graph*

$$H = \textsc{Prep}_2(\textsc{Prep}_1(G)) = \langle (V', E'), \Sigma, \mathcal{C}'\rangle$$

is such that

1. *(V', E') is $(d + d' + 1)$-regular for some constants $d,\ d'$.*
2. *$\lambda(H) < \lambda + d$ for some constant $\lambda < d'$.*
3. *$|V'| + |E'| \leq c(|V| + |E|)$ for some constant c.*
4. *$\textsc{Unsat}(G) \cdot \beta \leq \textsc{Unsat}(G') \leq \textsc{Unsat}(G)$ for some constant $0 < \beta < 1$.*

5.6 Powering Step

5.6.1 The Original Construction

In this step we amplify the unsatisfation value of the input graph G by an arbitrary constant. We will follow the work in [Jut06], which is an improvement to the initial Dinur's construction. Before presenting the final construction, we will first discuss the construction in [Din05], which makes the final one a little easier to perceive.

Definition 5.15. *Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a d-regular constraint graph, and let $t \in \mathbb{N}$ be an arbitrary constant. Define $G^t = \langle (V, E'), \Sigma', \mathcal{C}' \rangle$ to be the following constraint graph:*

- *The set of vertices remains the same.*
- *The set of edges E' is defined as follows: (u, v) belongs in E' if there is a walk of length t from u to v. Note that multiple edges are allowed between distinct vertices, as well as self loops. Thus, if $\mathbf{A}$ denotes the adjacency matrix of (V, E), the adjacency matrix of (V, E') is $\mathbf{A}^t$.*
- *$\Sigma' = \Sigma^{d^0+d^1+\ldots+d^t}$. As a first intuition for the selection of Σ', note that a symbol of Σ' encodes appropriately the assignment of all vertices in G reachable from a vertex, in a walk of length at most t. Such a possible encoding could be the d-adic tree which starts at a vertex u and visits in a breadth first search manner all the nodes reachable in at most t steps. Note that the number of nodes in this d-adic tree is $d^0 + d^1 + \ldots + d^t$ and recording each possible assignment $a : V \to \Sigma$ with a single symbol requires an alphabet of size $|\Sigma|^{d^0+d^1+\ldots+d^t}$.*
- *Constraints: Denote by $B(v, t)$ the set of vertices which are at distance at most t from v. Clearly $B(v, t)$ coincides with the set of vertices appearing in the d-adic tree described above and $|B(v, t)| \leq d^0+d^1+\ldots+d^t$. Consider an edge $e = (u, v) \in E'$ and $\sigma_1, \sigma_2 \in \Sigma'$ such that*

  $$\sigma_1 : B(u, t) \to \Sigma \text{ and } \sigma_2 : B(v, t) \to \Sigma$$

  The constraint on an edge $e = (u, v)$ is satisfied iff

  - *For every $w \in B(u, t) \cap B(v, t)$ it holds that $\sigma_1(w) = \sigma_2(w)$. Then, clearly there is $\sigma : B_1 \cup B_2 \to \Sigma$ such that $\sigma|_{B_1} = \sigma_1$ and $\sigma|_{B_2} = \sigma_2$.*
  - *Every edge in $(u, v) \in E \cap (B_1 \cup B_2) \times (B_1 \cup B_2)$ is satisfied from the assignment σ.*

First we convince ourselves that the constraint graph G^t is bigger than G by just a constant factor. To see this, note that

1. The set of vertices is invariant.

2. The graph G^t is d^t-regular since G is d-regular. Thus $|E'| = |V|\frac{d^t}{2}$. Note that both d and t are constants, the former due to the preprocessing step and the latter due to our assumptions.

Moreover, the alphabet has increased a great deal, but since both Σ, d and t are constants its cardinality is still a constant. Likewise, the set of constraints $\mathcal{C}'$ is much more complex, but since it grows only on constants (Σ, d and t) it can be constructed in linear time on the size of the input.

Remark 5.5.
Recall that the gap amplification lemma in the proof of $\text{Gap} - \text{CSP}$ hardness was applied several times depending on the size of the input graph. Thus, if not for the correcting the alphabet step, on different inputs we would end up in different instances of $\text{Gap} - \text{CSP}$ which would be a significant problem.

Up to now, we are assured that the size of the new graph is satisfying our requirements. The next and even more important question is why this transformation amplifies the UNSAT value of the original graph.

To gain some insight into this subject, we consider first the rather obvious solution to amplifying the probability of picking an unsatisfied edge in G by an arbitrary assignment. Note that when $\text{UNSAT}(G) = 0$, it must also be the case that $\text{UNSAT}(G^t) = 0$ since, given as input a satisfying assignment for G, we can clearly construct step by step a satisfying assignment for G^t.

Assume that $\text{UNSAT}(G) > 0$. Then, given an arbitrary assignment a in G, the probability that a random edge will be unsatisfied is at least $\text{UNSAT}(G)$. To amplify this probability, we could clearly pick t random edges which would result in a probability of magnitude

$$1 - (1 - \text{UNSAT}(G))^t \approx t \cdot \text{UNSAT}(G)$$

The black point of this solution is that in order to build the corresponding constraint graph we would one way or another be obliged to build $|E|^t$ constraints for each multiset of t edges which implies a non-linear transformation.

Instead, as we have already pointed out, walks of length t pass around this output size problem easily. Moreover, picking a random walk of length t in G is the same as picking a random edge in G^t. And even more significantly, the edges corresponding to a random walk of length t in an expander graph is in a manner (which will be clear later) as if we picked the edges at random.

The previous intuition is more accurately (but still in a non rigorous manner) presented in the following proof skeleton:

- First consider an assignment A for G^t based on an assignment a for G. Since the construction is based on the assignment h, consistency of A

on mutual neighbors is guaranteed. Consequently, the only way for an edge $(u, v) \in G^t$ to be unsatisfied is that an edge between two vertices in $B(u,t) \cup B(v,t)$ is unsatisfied in G. Hence, the probability of an unsatisfied edge in G is greater than the probability that a random walk of length t in G will contain an unsatisfied edge. Since G is an expander, the latter probability is roughly $c \cdot t \cdot \textsc{Unsat}(G)$ for some constant c. This however is less than half of the road to the proof since the point is to assure that any assignment to G^t keeps the unsatisfaction level of G^t sufficiently above than that of G.

- Let us consider the optimal assignment H for G^t. Constructing an assignment h for G such that

$$\textsc{Unsat}_H(G^t) \geq c \cdot t \cdot \textsc{Unsat}_h(G)$$

would clearly prove the desired claim since $\textsc{Unsat}_H(G^t) = \textsc{Unsat}(G)$ and $\textsc{Unsat}_h(G) \geq \textsc{Unsat}(G)$.

In the original construction of Dinur, the assignment h is defined as follows.

$$h(u) = \arg\max_{\sigma \in \Sigma} \Pr[H(v)_u = \sigma | (u, v) \in E']$$

This means that u is assigned the majority opinion on all of its neighbors in G^t. In the following section this assignment will be constructed in a much more peculiar way.

Consider now an unsatisfied edge (u, v) of G. We would like to bound from below the probability that a random walk o length t in G contains this edge and that this specific edge is unsatisfied by H. Clearly this probability is greater than the probability that a random walk if length t in G, which is resembled by an edge (v_0, v_t) in G^t,

1. contains this edge and,
2. $H(v_0)_u = h(u)$ and $H(v_t)_v = h(v)$, i.e. v_0 and v_t assignments agree on u, v with the majority assignments in G^t.

The former constraint can easily be surpassed by taking a walk of length i starting at u, ending up in a vertex v_0, and taking a walk of length $t-1-i$ starting at v, ending up in a vertex v_t. Then clearly the resulting walk is of length t between v_0 and v_t.

The latter constraint is the one making things difficult. For the majority assignment h defined above (which is the one Irit Dinur used in her proof) it can be proved that

$$\textsc{Unsat}_H(G^t) \geq c \cdot \sqrt{t} \cdot \textsc{Unsat}_h(G)$$

For the rather more complicated h that we will define afterwards (and the one that we will analyze), the square root in the term $\sqrt{t}$ will disappear. The main idea of this improvement is that h will be defined in such a way that for the paths described earlier the probability that the second requirement is satisfied is constant and independent of i. Thus, adding over all the t possible values of i the desired bound will come for free.

5.6.2 The Modified Version of Graph Powering

Preliminaries: The distribution L and J

Assume for the rest of this section that $t \in \mathbb{N}$ is divisible by 8 and consider the intervals

$$T_2 = \left[-\frac{t}{2}, \frac{t}{2}\right],\ T_4 = \left[-\frac{t}{4}, \frac{t}{4}\right], T_8 = \left[-\frac{t}{8}, \frac{t}{8}\right]$$

Definition 5.16. *The random distribution $\mathcal{L}$ is defined by the following random process.*

1. *Choose a random integer $i \in T_4$.*
2. *Choose a random integer $j \in T_4$.*
3. *Return $l = i + j$.*

Clearly l takes values in T_2. We prove that for each $l \in T_2$, the following lemma holds.

Lemma 5.17. $\Pr_{\mathcal{L}}(l) = \dfrac{\frac{t}{2} + 1 - |l|}{\left(\frac{t}{2} + 1\right)^2}$ *for each $l \in T_2$.*

Proof Clearly the number of pairs (i, j) such that $i, j \in T_4$ is

$$|T_4|^2 = \left(\frac{t}{2} + 1\right)^2 \tag{5.2}$$

and each pair occurs with equal probability.

The pairs (i, j) such that $i, j \in T_4$ and $i = l - j$ are the ones that

$$-\frac{t}{4} \leq l - j \leq \frac{t}{4} \Leftrightarrow l - \frac{t}{4} \leq j \leq l + \frac{t}{4} \tag{5.3}$$

If $l \geq 0$ then $l + \frac{t}{4} \geq \frac{t}{4}$ and $l - \frac{t}{4} \geq -\frac{t}{4}$. Thus, since $Pr[j > \frac{t}{4}] = 0$, (5.3) is equivalent to

$$l - \frac{t}{4} \leq j \leq \frac{t}{4}$$

Likewise, if $l \leq 0$ (5.3) is equivalent to

$$-\frac{t}{4} \leq j \leq l + \frac{t}{4}$$

Note that the above equations just describe the fact that j is in $\left[l - \frac{t}{4}, l + \frac{t}{4}\right] \cap T_4$. By the way the above inequalities were constructed, each value of j corresponds to a unique valid value of $i = l - j$. Moreover, the number of such j's is in both cases

$$\frac{t}{2} + 1 - |l|, \tag{5.4}$$

each occurring with equal probability. Combining (5.2) and (5.4), we obtain that for each $l \in T_2$

$$\Pr_{\mathcal{L}}(l) = \frac{\frac{t}{2} + 1 - |l|}{\left(\frac{t}{2} + 1\right)^2}$$

■

Remark 5.6.
Note that the most probable value of $l \in T_2$ under the distribution L is clearly 0 and more generally, the closer l is to 0 the bigger the probability $\Pr_{\mathcal{L}}(l)$.

The edges of G^t are peculiar to define. In the construction we described in section these were t-length walks. In this construction the length of the walk will be defined by means of the distribution L and the distribution J which follows.

Consider the set W^* of all walks in G. Clearly the set W^* has infinite cardinality.

Definition 5.18 (Distribution J). *Define the distribution $\mathcal{J}$ on $W^* \times T_4$ by the following random process.*

1. *Choose $l \in T_2$ according to $\mathcal{L}$ distribution.*
2. *Choose a random walk $\bar{w}$ on G of length $t + l$.*
3. *Choose a random $s \in \left[l - \frac{t}{4}, l + \frac{t}{4}\right] \cap T_4$.*
4. *Return $(\bar{w}, s)$.*

The following lemma illustrates that the distribution $\mathcal{J}$ can be obtained in a more explicit way.

Lemma 5.19 (Edges). *There exists a multiset W such that*

- *$W = O(|E|)$. Recall that E is the set of the edges in the original graph G.*
- *The next distribution is identical to $\mathcal{J}$:*

1. *Choose a random* $\bar{w} \in W$ *uniformly.*
2. *Set* $l = |\bar{w}| - t$ *and choose a random* $s \in \left[l - \frac{t}{4}, l + \frac{t}{4}\right] \cap T_4$.
3. *Return* $(\bar{w}, s)$.

Proof Denote by W_i the walks of length i in G. Since the walks defined by $\mathcal{J}$ are of length between $\frac{t}{2}$ and $\frac{3t}{2}$, the walks that W' includes must coincide with the union $W_{\frac{t}{2}}, \ldots, W_t, \ldots, W_{\frac{3t}{2}}$. Thus, the only thing that we need to care about is to pick the multiplicity of the each walk accordingly so that the random process defined in the lemma is identical to the random process which generates $\mathcal{J}$.

But this can be done rather easily since there are $\frac{t}{2} + 1 - |l|$ possible values for the s that A picks (s satisfies the same inequalities as j in the proof of Lemma 5.17). Thus W is the multiset defined by the union of the sets

$$W = \bigcup_{i=\frac{t}{2}}^{\frac{3t}{2}} \underbrace{W_i \cup \ldots \cup W_i}_{\frac{t}{2}+1-|i-t|}$$

Notice that l is actually the quantity $i - t$. In other words, the above equation says that if $\bar{w} \in W_i$, W' contains $\frac{t}{2} + 1 - |i - t|$ copies of $\bar{w}$.

It is easy to see that $|W| = O(|E|)$. Namely, $|W_i| = |V| \cdot \frac{d^t}{2}$ and since d and t are constants (recall that d is the regularity of the expander G) and each W_i has at most $\frac{t}{2}$ copies into W. ∎

The next lemma provides a different way to look at the distribution $\mathcal{J}$.

Lemma 5.20 (Distribution J). *Let* $i \in T_8$. *The following distribution* $\mathcal{B}_i$ *is identical to* $\mathcal{J}$*:*

1. *Choose* $(u, v) \in E$.
2. *Choose* $j_1, j_2 \in T_4$ *independently.*
3. *Pick a walk of length* $\frac{t}{2} + i + j_1$ *starting from* u. *Let*

$$(v_0, \ldots, v_{\frac{t}{2}+i+j_1} = u)$$

denote the resulting walk.

4. *Pick a walk of length* $\frac{t}{2} - i - 1 + j_2$ *starting from* v. *Let*

$$(v = v_{\frac{t}{2}+i+j_1+1}, \ldots, v_{t+j_1+j_2})$$

denote the resulting walk.

5. *Return* $\bar{w} = (v_0, \ldots, v_{t+j_1+j_2})$ *and* $s = j_1$.

Proof First identify $j_1 + j_2$ with l of Definition 5.18, since $j_1 + j_2 \sim \mathcal{L}$ by definition. Moreover, since

$$l - \frac{t}{4} = j_1 + j_2 - \frac{t}{4} \leq j_1 \leq j_1 + j_2 + \frac{t}{4} = l + \frac{t}{4}$$

it follows that $j_1 \in \left[l - \frac{t}{4}, l + \frac{t}{4}\right]$. Notice also that each inequality above may hold as equality for an appropriate choice of j_2. Therefore j_1 is uniformly distributed on $\left[l - \frac{t}{4}, l + \frac{t}{4}\right] \cap T_4$ and consequently s is equivalently defined to Definition 5.18. Thus we are left to prove that the way the $(t + l)$-walk $\bar{w}$ was generated is equivalent to picking it uniformly from all the $(t+l)$- length walks. But this can easily be verified by a counting argument, keeping in mind that the graph G is regular. Namely,

- Step 1 has $|V| \cdot d/2$ choices.
- Step 2 has $d^{\frac{t}{2}+i+j_1}$ choices.
- Step 3 has $d^{\frac{t}{2}-i-1+j_2}$ choices.

Thus, given j_1, j_2 there are $|V|d^{t+l}/2$ possible choices for the walk $\bar{w}$, which means that the above procedure is equivalent to picking $\bar{w}$ from all the possible $|V|d^{t+l}/2$ walks of length $t + l$. ∎

The Final Construction

The final construction is almost the same as the one in Definition 5.15 with the difference that the set of edges is defined differently. We rewrite the whole definition for easy reference. If the reader shall get the feeling that some parts of the final construction described in this section are too laconic, it is highly recommended to go through the extended version of Definition 5.15.

Definition 5.21. *Let* $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ *be a* d*-regular constraint graph, and let* $t \in \mathbb{N}$ *be an arbitrary constant. Define* $G^t = \langle (V, E'), \Sigma', \mathcal{C}' \rangle$ *to be the following constraint graph:*

- *The set of vertices remains the same.*
- *The set of edges* E' *is defined as follows: Let* W *the multiset of random walks established in Lemma 5.19. There are exactly* k *parallel edges* (u, v) *in* E' *iff the number of walks in* W *between* u *and* v *equals* k*. Note that multiple edges are allowed between distinct vertices, as well as self loops.*
- $\Sigma' = \Sigma^{d^0 + d^1 + \ldots + d^t}$.

- *Constraints: Consider an edge* $e = (u, v) \in E'$ *and* $\sigma_1, \sigma_2 \in \Sigma'$ *such that*

$$\sigma_1 : B(u,t) \to \Sigma \text{ and } \sigma_2 : B(v,t) \to \Sigma$$

The constraint on an edge $e = (u, v)$ *is satisfied iff*

  - *For every* $w \in B(u,t) \cap B(v,t)$ *it holds that* $\sigma_1(w) = \sigma_2(w)$. *Then, clearly there is* $\sigma : B_1 \cup B_2 \to \Sigma$ *such that* $\sigma|_{B_1} = \sigma_1$ *and* $\sigma|_{B_2} = \sigma_2$.
  - *Every edge in* $(u, v) \in E \cap (B_1 \cup B_2) \times (B_1 \cup B_2)$ *is satisfied from the assignment* σ.

Lemma 5.19 guarantees that the above construction keeps the size of G^t linearly bigger than that of the input graph G. The arguments presented for the original construction of Definition 5.15 apply here too, so that G^t can be computed in linear time. Furthermore, if $\text{UNSAT}(G) = 0$ then $\text{UNSAT}(G^t) = 0$. Hence, in order to complete the correctness of the final constructions, it remains to prove the following theorem.

Theorem 5.22 (Gap Amplification). *There exist constants* c *and* γ *dependent on* d *and* $|\Sigma|$ *such that*

$$\text{UNSAT}(G) > 0 \Rightarrow \text{UNSAT}(G^t) \geq \min\{\gamma, c \cdot t \cdot \text{UNSAT}(G)\}$$

Let $H : V \to \Sigma'$ the optimal assignment for G^t. We construct the assignment $h : V \to \Sigma$ for G as follows:

1. Choose integer k in $\left[\frac{t}{2} - \frac{t}{8}, \frac{t}{2} + \frac{t}{8}\right]$ uniformly.
2. For $u \in V$, choose a walk of length k in G. Denote by v the end vertex of this walk.
3. $h(u) = \arg\max_{\sigma \in \Sigma} \Pr[H(v)_u = \sigma | (u, v) \in E']$

Thus, the majority opinion on u is modified in comparison to the one presented in the original construction as it takes into consideration only the opinion of vertices close enough to u.

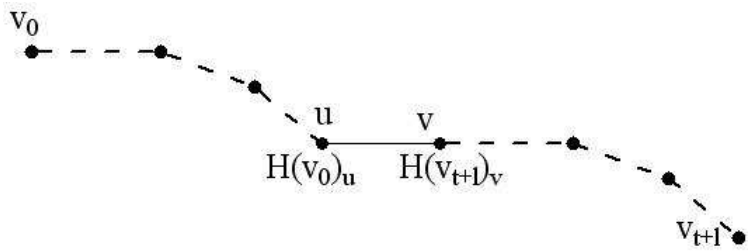
Denoting by U, U' the sets of edges left unsatisfied by h and H respectively, our aim is to prove that

$$\text{UNSAT}(G^t) = \frac{|U'|}{|E'|} \geq c \cdot t \cdot \frac{|U|}{|E|} = c \cdot t \cdot \text{UNSAT}_h(G) \geq c \cdot t \cdot \text{UNSAT}(G)$$

Definition 5.23. *For each* $i \in T_8$, *denote by* $J_i(\bar{w}, s)$ *the following event on* $W \times T_4$ *:*

1. $\bar{w} \in W$. *Thus* $\bar{w} = (v_0, \ldots, v_{t+l})$ *for some* $l \in T_2$.
2. $s \in \left[l - \frac{t}{4}, l + \frac{t}{4}\right] \cap T_4$. *Let* $u = v_{i+s+\frac{t}{2}}$ *and* $v = c_{i+s+\frac{t}{2}+1}$. *Note that* u, v *are successive vertices in the walk* $\bar{w}$ *and consequently* $(u, v) \in E$.
3. $H(v_0)_u = h(u)$.
4. $H(v_{t+l})_v = h(v)$.
5. $(u, v) \in U$.

Notice that the first two conditions are immediately satisfied by any pair $(\bar{w}, s) \in W \times T_4$ *by definition.*



If $J_i(\bar{w}, s)$ occurs for some i then clearly $(v_0, v_{t+l}) \in U'$. Define an indicator variable $x_i(\bar{w}, s)$ such that

$$x_i(\bar{w}, s) = \begin{cases} 1, & \text{iff } J_i(\bar{w}, s) \text{ occurs} \\ 0, & \text{otherwise} \end{cases}$$

and

$$N(\bar{w}, s) = \sum_{i \in T_8} x_i(\bar{w}, s)$$

We claim that the following lemma holds.

Lemma 5.24. *There exists constants* γ, c *such that*

$$\Pr_{(\bar{w},s)\sim\mathcal{J}}[N(\bar{w}, s) > 0] \geq \min\left\{\gamma, c \cdot t \cdot \frac{|U|}{|E|}\right\}$$

Assuming the validity of Lemma 5.24, Theorem 5.22 follows.

Proof (Gap Amplification) It is easy to see that $\textsc{Unsat}(G^t)$ is bounded from below by $\Pr_{(\bar{w},s)\sim\mathcal{J}}[N(\bar{w}, s) > 0]$. Namely, when ranging over all $(\bar{w}, s)$ $N(\bar{w}, s) > 0$ is the event of $J_i(\bar{w}, s)$ occur

$$\textsc{Unsat}(G^t) \geq \Pr_{(\bar{w},s)\sim\mathcal{J}}[N(\bar{w}, s) > 0] \geq c \cdot t \cdot \frac{|U|}{|E|} = c \cdot t \cdot \textsc{Unsat}_h(G) \qquad \blacksquare$$

To prove Lemma 5.24 we use the inequality $\Pr[Y > 0] \geq \frac{(\mathbb{E}[Y])^2}{\mathbb{E}[Y^2]}$ (see Appendix for a proof), namely

$$\Pr_{(\bar{w},s)\sim\mathcal{J}}[N(\bar{w},s) > 0] \geq \frac{(\mathbb{E}_{\mathcal{J}}[N])^2}{\mathbb{E}_{\mathcal{J}}[N^2]} \tag{5.5}$$

Proposition 5.25 (Expected Value).

$$\mathbb{E}_{\mathcal{J}}[N] \geq \sum_i \mathbb{E}_{\mathcal{J}}[x_i] \geq \frac{1}{16|\Sigma|^2} \cdot t \cdot \frac{|U|}{|E|}$$

Proposition 5.26 (Second Moment). *If* $\tilde{\lambda} = \tilde{\lambda}(G)$ *then*

$$\mathbb{E}_{\mathcal{J}}[N^2] \leq 3 \max\left\{ \sum_i \mathbb{E}_{\mathcal{J}}[x_i], \left[\left(\frac{t}{4}\right)^2 + \frac{t}{4}\right] \cdot \frac{|U|^2}{|E|^2}, \ \frac{t}{2} \cdot \frac{1}{1-\tilde{\lambda}} \cdot \frac{|U|}{|E|} \right\}$$

Using Proposition 5.25 and Proposition 5.26, Lemma 5.24 follows from (5.5).

Proof (Expected Value) We first prove that

$$\mathbb{E}_{\mathcal{J}}[x_i] \geq \frac{1}{16|\Sigma|^2} \cdot \frac{|U|}{|E|} \tag{5.6}$$

By the definition of event J_i and using Lemma 5.20, we have that

$$\begin{aligned}
\Pr_{\mathcal{J}}[x_i > 0] &= \Pr[(u,v) \in U] \cdot \Pr_{\mathcal{J}}\left[H(v_0)_u = h(u) | u = v_{i+s+\frac{t}{2}}\right] \cdot \\
&\qquad \cdot \Pr_{\mathcal{J}}\left[H(v_{t+l})_v = h(v) | v = v_{i+s+\frac{t}{2}+1}\right] \\
&= \Pr[(u,v) \in U] \cdot \Pr_{\mathcal{J}}\left[H(v_0)_u = h(u) | (u,v)\right] \cdot \\
&\qquad \cdot \Pr_{\mathcal{J}}\left[H(v_{t+j_1+j_2})_v = h(v) | (u,v)\right] \qquad (5.7)
\end{aligned}$$

Let us consider $\Pr_{\mathcal{J}}\left[H(v_0)_u = h(u) | (u,v)\right]$. Define on the distribution $\mathcal{J}$ the event

$$P_{j_1} = \left\{ \left(i + j_1 + \frac{t}{2}\right) \in \left[\frac{t}{2} - \frac{t}{8}, \frac{t}{2} + \frac{t}{8}\right] \right\}$$

Then, using Bayes formula, we have that

$$\Pr_{\mathcal{J}}\left[H(v_0)_u = h(u) | (u,v)\right] \geq \Pr_{\mathcal{J}}\left[P_{j_1}\right] \cdot \Pr_{\mathcal{J}}\left[H(v_0)_u = h(u) | P_{j_1}\right]$$

Note that $i + j_1 + \frac{t}{2}$ is uniformly distributed on $\left[i + \frac{t}{2} - \frac{t}{4}, i + \frac{t}{2} + \frac{t}{4}\right]$, thus

$$\Pr_{\mathcal{J}}\left[P_{j_1}\right] = \frac{\frac{t}{4}+1}{\frac{t}{2}+1} > \frac{1}{2}$$

Moreover, by the definition of h and since there are at most $|\Sigma|$ symbols in the alphabet, it follows that

$$\Pr_{\mathcal{J}}[H(v_0)_u = h(u)|P_{j_1}] \geq \frac{1}{|\Sigma|}$$

Therefore

$$\Pr_{\mathcal{J}}[H(v_0)_u = h(u)|(u,v)] > \frac{1}{2|\Sigma|}$$

and in a similar fashion

$$\Pr_{\mathcal{J}}[H(v_{t+j_1+j_2})_v = h(v)|(u,v)] > \frac{1}{2|\Sigma|}.$$

Thus, equation (5.7) yields

$$\Pr_{\mathcal{J}}[x_i > 0] > \frac{1}{4|\Sigma|^2} \cdot \frac{|U|}{|E|}$$

and trivially

$$\mathbb{E}_{\mathcal{J}}[x_i] > \frac{1}{4|\Sigma|^2} \cdot \frac{|U|}{|E|}$$

Finally, linearity of expectation shows that

$$\mathbb{E}_{\mathcal{J}}[N] = \sum_{i \in T_8} \mathbb{E}_{\mathcal{J}}[x_i] > \frac{1}{4|\Sigma|^2} \cdot \left(\frac{t}{4} + 1\right) \cdot \frac{|U|}{|E|} > \frac{1}{16|\Sigma|^2} \cdot t \cdot \frac{|U|}{|E|} \qquad \blacksquare$$

Proof (Second Moment) Notice that

$$\mathbb{E}_{\mathcal{J}}[N^2] = \mathbb{E}_{\mathcal{J}}\left[\left(\sum_{i \in T_8} x_i\right)\right] = \mathbb{E}_{\mathcal{J}}\left[\sum_{(i,j) \in T_8 \times T_8} x_i x_j\right] = \sum_{(i,j) \in T_8 \times T_8} \mathbb{E}_{\mathcal{J}}[x_i x_j] \tag{5.8}$$

Instead of calculating the probability that $J_i(\bar{w}, s)$ occurs, we relax our requirements and seek only for walks in G which contain an edge in U. This relaxation is in our interests, since we are trying to bound $N(\bar{w}, s)$ from above.

To be more specific, for each $i \in T_8$, denote by $J_i'(\bar{w}, s)$ the following event on $W \times T_4$:

- $\bar{w} = (v_0, \ldots, v_{t+l})$ with $u = v_{\frac{t}{2}+i+s}$ and $v = v_{\frac{t}{2}+i+s+1}$.
- $(u, v) \in U$.

and define the indicator variable

$$y_i(\bar{w}, s) = \begin{cases} 1, & \text{iff } J_i'(\bar{w}, s) \text{ occurs} \\ 0, & \text{otherwise} \end{cases}$$

Clearly, if $J_i(\bar{w}, s)$ occurs then $J'_i(\bar{w}, s)$ occurs too, and consequently

$$x_i(\bar{w}, s) \leq y_i(\bar{w}, s)$$

Noting that $y_i^2 = y_i$ and $x_i^2 = x_i$, (5.8) yields

$$\mathbb{E}_{\mathcal{J}}[N^2] \leq \sum_{i \in T_8} \mathbb{E}_{\mathcal{J}}[x_i] + \sum_{\substack{(i,j) \in T_8 \times T_8 \\ i \neq j}} \mathbb{E}_{\mathcal{J}}[y_i y_j] \tag{5.9}$$

A simple calculating argument, as the one presented in the proof of Lemma 5.20, can easily establish that

$$\mathbb{E}_{\mathcal{J}}[y_i] = \Pr_{\mathbb{J}}[y_i > 0] = \frac{|U|}{|E|}$$

and therefore,

$$\sum_{i \in T_8} \mathbb{E}_{\mathcal{J}}[y_i] = \left(\frac{t}{4} + 1\right) \cdot \frac{|U|}{|E|} \tag{5.10}$$

Next, we calculate the terms $\mathbb{E}_{\mathcal{J}}[y_i y_j]$ appearing in (5.9). Using Bayes formula

$$\begin{aligned}
\mathbb{E}[y_i y_j] &= \Pr_{\mathcal{J}}[y_i y_j > 0] \\
&= \sum_{(l,s)} \Pr \mathcal{J}[y_i y_j > 0 | l, s] \cdot \Pr_{\substack{l \in T_2 \\ s \in T_4}}[l, s] \\
&= \sum_{(l,s)} \Pr\left[(v_i, v_{i+1}) \in U \wedge (v_j, v_{j+1}) \in U | l, s\right] \cdot \Pr_{\substack{l \in T_2 \\ s \in T_4}}[l, s] \\
&= \sum_{(l,s)} \Pr\left[(v_j, v_{j+1}) \in U | l, s\right] \cdot \Pr\left[(v_i, v_{i+1}) \in U | l, s, (v_j, v_{j+1}) \in U\right] \cdot \Pr_{\substack{l \in T_2 \\ s \in T_4}}[l, s] \\
&= \frac{|U|}{|E|} \cdot \sum_{(l,s)} \Pr\left[(v_i, v_{i+1}) \in U | l, s, (v_j, v_{j+1}) \in U\right] \cdot \Pr_{\substack{l \in T_2 \\ s \in T_4}}[l, s]
\end{aligned}$$

Again, using the fact that G is d-regular, it can easily be proved that the distribution of the edge (v_i, v_{i+1}) on a random walk conditioned on (v_j, v_{j+1}) is the same as the distribution defined by the following random process $\mathcal{D}$.

1. Choose a random edge in U and pick a random endpoint of the edge, say v_0.
2. Take a random walk of length $(i - j + 1)$ in G and let (v_{i-j}, v_{i-j+1}) denote the last step.

Thus $\Pr\left[(v_i, v_{i+1}) \in U | l, s, (v_j, v_{j+1}) \in U\right]$ does not depend on l, s and since

$$\sum_{(l,s)} \Pr_{\substack{l \in T_2 \\ s \in T_4}}[l, s] = 1$$

it suffices to find an upper bound for $\Pr[(v_i, v_{i+1}) \in U | (v_j, v_{j+1}) \in U]$. But this has already been treated in section 2.4. Using the bound for an $(i-j+1)$-length walk, we obtain

$$\mathbb{E}[y_i y_j] \leq \frac{|U|}{|E|} \cdot \left(\frac{|U|}{|E|} + \tilde{\lambda}^{i-j} \right)$$

and consequently

$$\begin{aligned}
\sum_{\substack{(i,j) \in T_8 \times T_8 \\ i \neq j}} \mathbb{E}_{\mathcal{J}}[y_i y_j] &\leq 2 \sum_{\substack{(i,j) \in T_8 \times T_8 \\ i > j}} \frac{|U|}{|E|} \cdot \left(\frac{|U|}{|E|} + \tilde{\lambda}^{i-j} \right) \\
&\leq 2 \sum_{\substack{(i,j) \in T_8 \times T_8 \\ i > j}} \frac{|U|^2}{|E|^2} + 2 \sum_{\substack{(i,j) \in T_8 \times T_8 \\ i > j}} \frac{|U|}{|E|} \cdot \tilde{\lambda}^{i-j} \\
&\leq \left[\left(\frac{t}{4} \right)^2 + \frac{t}{4} \right] \cdot \frac{|U|^2}{|E|^2} + 2 \frac{|U|}{|E|} \sum_i \frac{t}{4} \tilde{\lambda}^i \\
&\leq \left[\left(\frac{t}{4} \right)^2 + \frac{t}{4} \right] \cdot \frac{|U|^2}{|E|^2} + \frac{t}{2} \cdot \frac{1}{1-\tilde{\lambda}} \cdot \frac{|U|}{|E|}
\end{aligned}$$

Finally (5.9) gives

$$\begin{aligned}
\mathbb{E}_{\mathcal{J}}[N^2] &\leq \sum_i \mathbb{E}_{\mathcal{J}}[x_i] + \left[\left(\frac{t}{4} \right)^2 + \frac{t}{4} \right] \cdot \frac{|U|^2}{|E|^2} + \frac{t}{2} \cdot \frac{1}{1-\tilde{\lambda}} \cdot \frac{|U|}{|E|} \\
&\leq 3 \max \left\{ \sum_i \mathbb{E}_{\mathcal{J}}[x_i], \left[\left(\frac{t}{4} \right)^2 + \frac{t}{4} \right] \cdot \frac{|U|^2}{|E|^2}, \frac{t}{2} \cdot \frac{1}{1-\tilde{\lambda}} \cdot \frac{|U|}{|E|} \right\} \qquad \blacksquare
\end{aligned}$$

5.7 Proof Composition

So far, we have managed to turn the initial input constraint graph G to another one whose unsatisfaction value can be selected to be arbitrary bigger than $\textsc{Unsat}(G)$. The only drawback of this reduction is the increase on the alphabet, which is undesirable since the reduction will be applied $\log |E| + 1$ times resulting in Σ depending on the input.

In this step, we correct the alphabet via proof composition, a powerful technique which was first introduced by Arora and Safra.

The idea of proof composition is an elegant argument but getting the gist of it, due to the many technical details, is hard. Before presenting it in its full form, we will try to demonstrate in an abstract form the various aspects of its application.

Let's see how a PCP verifier V would work in G^t. Since an edge by our so far reduction is unsatisfied with good probability for our scope of interest, it would

pick a random edge and query into the proof the assignment to its endpoints and then check whether the chosen edge is satisfied or not. This, as we have already pointed out, would not work generally since gap amplification is applied several times and hence a polynomial number of bits would be required to check a constraint. Despite this, G^t's alphabet has constant size and each constraint is on two variables.

Suppose now that we have a reduction $\mathcal{P}$ from any NP language to a constraint graph satisfaction problem which keeps the alphabet low but creates a super-polynomial number of constraints. Seeing each constraint of G^t as an individual NP statement which we want to check if it is satisfiable, we could cast the constraint V chooses into $\mathcal{P}$ and ask for a proof that it is satisfiable. $\mathcal{P}$ given as input one of our constraints would definitely create a large number of constraints but since each constraint of G^t is of constant size the output would be of constant size too. Hence, we could create a constraint graph G_i for each edge in G^t and ask for a proof which will satisfy every G_i. Clearly the alphabet is by now corrected.

Does this suffice? The answer is no. The reason is that a proof of the satisfiability of a single constraint does not guarantee by any means that a single assignment can satisfy all of the constraints (think the analogue of clauses and a 3SAT formula). Hence, we should glue together all these G_i's and generate appropriate constraints in such a way that consistency is secured.

Having studied the intuition behind proof composition, we are now ready to state the main result of this section.

Theorem 5.27 (Proof Composition). *There exist constants $q > 1$, $0 < \epsilon < 1$ such that for every constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ there exists a constant c_Σ such that a constraint system $\mathcal{C}'$ of q-ary constraints on a binary alphabet can be constructed such that:*

- $|\mathcal{C}'| \leq c_\Sigma(|V| + |E|)$.
- $\textsc{Unsat}(G) = 0 \Rightarrow \textsc{Unsat}(\mathcal{C}') = 0$.
- $\textsc{Unsat}(G) > 0 \Rightarrow \textsc{Unsat}(\mathcal{C}') \geq \epsilon \cdot \textsc{Unsat}(G)$.

The transformation of the constraint system $\mathcal{C}'$ into a constraint graph problem (binary constraints) can be accomplished easily. Namely:

- Consider an alphabet Σ' such that all possible assignments to a constraint of $\mathcal{C}'$ can be encoded with a single $\sigma \in \Sigma'$. Clearly, $|\Sigma'| = 2^q$.
- For each constraint in $\mathcal{C}'$ introduce a node.
- $(u, v) \in E'$ iff the corresponding constraints in $\mathcal{C}'$ share at least one variable.

- $(u, v) \in E'$ is satisfied iff the assignments to u, v are consistent on their common variables.

Since q is constant and $|\mathcal{C}'| = O(|V| + |E|)$, it should be clear that the output constraint graph is linearly bigger than the size of the input graph.

A sketch of the proof of Theorem 5.27 follows:

1. Replace each variable v with a set of Boolean variables $[v]$ and expect a proper encoding from an assignment to v to an assignment to $[v]$.

2. We construct constraints such that:

   - For every $v \in V$ the constraints check whether an assignment to $[v]$ roughly encodes an assignment to v.
   - For every $(u, v) \in E$ the constraints check whether the assignments to $[u]$, $[v]$ roughly encode a satisfying assignment for (u, v).

5.7.1 Local Testing Revisited

In this subsection, we prove three lemmata which will be handy in proving Theorem 5.27. The reader not interested in technical details, though not ecouraged to do so, may omit the proofs of these lemmata and jump to the next subsection.

Recall that in section 4.5, we saw that Hadamard code is locally decodable. We repeat the algorithm for easy reference.

Algorithm 1 (LinTest).

Input: A string $w \in \{0,1\}^{2^l}$.

Procedure:

Choose a random $x \in \{0,1\}^l$.

Choose a random $y \in \{0,1\}^l$.

Output:

LinTest$(w) = 1$ iff $w(x) \oplus w(y) = w(x \oplus y)$.

We are ready to state the first lemma.

Lemma 5.28. *There exists $L_1^\Sigma \in \mathbb{N}$ and an encoding $H : \Sigma \to \{0,1\}^{L_1^\Sigma}$ such that the property $P = \{H(a)|a \in \Sigma\}$ has an LTA A_1, and the encoding H has relative distance at least $1/3$.*

Proof Take H to be the Hadamard code. Then H has, as we have already seen in section 4.4, relative distance of $1/2$ and **LinTest** is a $(3, \frac{1}{5})$-LTA for H.ard code. Then H has, as we have already seen, relative distance of $1/2$ and **LinTest** is a $(3, \frac{1}{5})$-LTA for H. ∎

The next lemma is an extension of Lemma 5.28.

Lemma 5.29. *For every $\Phi : \Sigma \times \Sigma \to \{0,1\}$ there exists $L_2^\Sigma \in N$ and an encoding $H_\Phi : \Sigma \times \Sigma \in \{0,1\}^{L_2^\Sigma}$ such that the property $P_\Phi = \{H_\Phi(a,b)|\Phi(a,b) = 1\}$ has an LTA A_2, and the encoding H_Φ have relative distance at least $1/3$.*

Remark 5.7.
Note that Lemma 5.29 is a generalization of Lemma 5.28, since the latter follows from the former by letting $\Phi(a,b) = 1$ for each a, b. Namely, Lemma 5.29 allows to check whether an assignment to a constraint satisfies it.

Recall that in section , we saw that Hadamard code is locally decodable. We repeat the algorithm for easy reference.

Algorithm 2 (SelfCorr).

Input: A string $w \in \{0,1\}^{2^l}$.
An index $x \in \{0,1\}^l$.

Procedure:
Choose a random $y \in \{0,1\}^l$.

Output:
Return $\mathbf{SelfCorr}^w(x) = w(y) \oplus w(x \oplus y)$.

Proof (Lemma 5.29) First of all, we construct a family of quadratic functions which simulate the boolean predicate Φ. The most intuitive way to construct this function is using its boolean circuit C_Φ.

Suppose X are the input variables of the circuit C_Φ. We add a variable for each of the internal wires in C_Φ, and denote these variables by Y. Each gate in C_ϕ can now be described by a quadratic equation in its input and output variables. Namely,

- NOT: $z_1 + z_2 - 1 = 0$.
- AND: $z_1 z_2 - z_3 = 0$.
- OR: $z_1 + z_2 - z_1 z_2 - z_3 = 0$.

Thus, for every gate in C_Φ assign an equation $f_i = 0$ for $i = \{1, \ldots, m\}$, where m denotes the number of gates. Note that m depends only on the size of $|\Sigma|$. To be more specific since each symbol in Σ can be encoded by $\log |\Sigma|$ binary variables, $|X| \leq |\Sigma|$. Thus, each Φ can be written in conjunctive normal form using at most $2^{|\Sigma|}$ clauses in disjunctive normal form and splitting each clause into 2-input gates shows that $m \leq O(|\Sigma| 2^{|\Sigma|})$. Since Σ is of constant size, m is a constant.

Observe that the f_i's are quadratic functions but when the variables are assigned Boolean values, each of them is Boolean. Clearly, if C_Φ by an assignment $a : X \to \{0, 1\}$ to its input variables, it can be extended to a unique assignment $a' : X \cup Y \to \{0, 1\}$ such that each f_i is satisfiable. Thus, satisfying $E(a, b)$ is equivalent to satisfying the family $\{f\}_{i=1}^m$.

Denoting by $Z = X \cup Y$ checking f_i could be simplified by asking not for an assignment to z_j but also the terms $z_j z_k$.

More formally, if $n = |Z|$ we denote by $z \otimes z$ the string, which for clarity we represent as a vector,

$$z \otimes z = [z_1 z_1, z_1 z_2, \ldots, z_1 z_n, z_2 z_1, z_2 z_2, \ldots, z_n z_{n-1}, z_n z_n].$$

Note that $n \leq 2m = O(1)$, so the size of $z \otimes z$ is again a constant ($n^2 = O(1)$). Moreover, observe that each f_i is linear with respect to $z \otimes z$.

We will use the Hadamard encoding of $z \otimes z$. The size of the Hadamard encoding of $z \otimes z$ is again a constant (though of order $|\Sigma|^2 2^{2^{2|\Sigma|}}$!). Thus the final encoding for an assignment a to the boolean variables X is of the form

$$H_\phi = a \to a' \to H(a' \otimes a')$$

Clearly the Hadamard encoding of $z \otimes z$ still has relative distance $1/3$. Thus we would be pleased if for some constants q, ϵ we found (q, ϵ)-LTA for $H(z \otimes z)$.

Note that $z_i^2 = z_i$ thus the terms $z_i z_j$ with $i = j$ are linear with respect to z. Denote by z_{lin} the string which consists of the assignments to z_i^2. Below follows an algorithm to test whether a $w \in \{0, 1\}^{2^{n^2}}$ is a legal Hadamard encoding of a word $z \otimes z$. We study its correctness right afterward.

Algorithm 3 (QdTest).

Input:

A string $w \in \{0,1\}^{2^{n^2}}$.

Procedure:

Choose a random $\alpha \in \{0,1\}^n$.

Choose a random $\beta \in \{0,1\}^n$.

Output:

$\mathbf{QdTest}(w) = 1$ iff $\mathbf{SelfCorr}^{w_{\text{lin}}}(\alpha) \cdot \mathbf{SelfCorr}^{w_{\text{lin}}}(\beta) = \mathbf{SelfCorr}^{w}(\alpha \otimes \beta)$

- If $w \in P$ then there is a z such that $w = H(z \otimes z)$. Clearly $w_{\text{lin}} = H(z)$. Thus

$$\mathbf{SelfCorr}^{w_{\text{lin}}}(\alpha) = \sum_{i=1}^{n} \alpha_i z_i$$

$$\mathbf{SelfCorr}^{w_{\text{lin}}}(\beta) = \sum_{j=1}^{n} \beta_j z_j$$

$$\mathbf{SelfCorr}^{w_{\text{lin}}}(\alpha \otimes \beta) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \beta_j z_i z_j$$

  It should be clear now $\mathbf{SelfCorr}^{w_{\text{lin}}}(\alpha) \cdot \mathbf{SelfCorr}^{w_{\text{lin}}}(\beta) = \mathbf{SelfCorr}^{w}(\alpha \otimes \beta)$ and consequently $\mathbf{QdTest}(w) = 1$.

- If $\text{dist}(w, P) \geq \delta$ then denote by x the word such that

$$\text{dist}(H(x), w_{\text{lin}}) \geq \delta$$

  and similarly y for w. Define the matrices M_1, M_2 such that $(M_1)_{ij} = x(i) \cdot x(j)$ and $(M_2)_{ij} = y(i,j)$. By hypothesis, we have that $M_1 \neq M_2$.

  Each of the self correcting algorithms give the right answer for the respective Hadamard word with probability $1 - 2\delta$ and thus the output step checks with probability at least $(1 - 2\delta)^3 \geq 1 - 6\delta$ that

$$\sum_{i,j} x(i)x(j)\alpha(i)\beta(j) = \sum_{i,j} y(i,j)\alpha(i)\beta(j) \Rightarrow \alpha^{\mathrm{T}} M_1 \beta = \alpha^{\mathrm{T}} M_2 \beta \Rightarrow$$

$$\alpha^{\mathrm{T}}(M_1 - M_2)\beta = 0$$

Since $M_1 \neq M_2$, this happens with probability at most $3/4$ for a random choice of α, β. The easiest way to see this is to note that $\gamma = (M_1 - M_2)\beta$ is nonzero with probability at least $1/2$ and in this case $\alpha^{\mathrm{T}}\gamma$ is nonzero with probability $1/2$. Thus the overall acceptance probability in this case is bounded above by $\frac{3}{4} + 6\delta$ which proves the soundness of **QdTest.**

Using **QdTest** we can finally construct an LTA for $P_\Phi = (H_\Phi(a,b)|\Phi(a,b) = 1)$.

Algorithm 4.

Input:

An assingment $w \in \{0,1\}^{2^{n^2}}$.

Procedure:

1. Use **QdTest**(w) to check that w is a legal Hadamard code.
2. Choose a random $\alpha \in \{0,1\}^m$ and set $f_\alpha(z) = \sum \alpha_i f_i(z)$.

Output:

1. If **SelfCorr**$^w(f_\alpha) = 1$ then return 0.
2. Denote by z_k the output of C_Φ. Return **SelfCorr**$^w(0 \ldots 010 \ldots 0)$, where the 1 is in the $n(k-1)+k$ position (the bit corresponding to z_k^2).

Remark 5.8.
If we write out $f_\alpha(z)$ it will be of the form $\sum z_i z_j$ where the sum is over a set of pairs (i,j) which are defined by those $f_i(z)$ such that $\alpha_i \neq 0$. Thus $f_\alpha(z)$ is a linear function on $z \otimes z$ and thus step 1 in the output level is well-defined.

The key fact about $f_\alpha(z)$ is captured by the following lemma.

Lemma 5.30.

1. *If* $f_i(z) = 0$ *for all* i *then* $\Pr_\alpha[f_\alpha(z) = 0] = 1$ *for any* $\alpha \in \{0,1\}^m$.
2. *If* $f_i(z) \neq 0$ *for some* i *then* $\Pr_\alpha[f_\alpha(z) \neq 0] = \frac{1}{2}$ *for any* $\alpha \in \{0,1\}^m$.

Proof The first part of the lemma should be obvious. For the second part, consider the Hadamard encoding of the string $f = [f_1(z), \ldots, f_m(z)]$. We have seen in section that if f is not the all zero string, then exactly half of the entries in $H(f)$ are equal to 0. But this is exactly the probability we seek. ∎

Completeness of Algorithm 4

If $w \in P_\Phi$ then there exists $w = H(z \otimes z)$ and z is a satisfying assignment for C_Φ. Consequently, the procedure step 1 clearly succeeds. Note that for each $i \in \{1, \ldots, m\}$ $f_i(z) = 0$ and consequently $\mathbf{SelfCorr}^w(f_\alpha) = 0$. Finally, $z_k = 1$ and the last self correction will return the right value of z_k^2 with probability 1.

Soundness of Algorithm 4

Suppose that $\text{dist}(w, P_\Phi) = \delta$.

- Step 1 in the procedure fails with probability $\Theta(\delta)$.
- If z is not an encoding of a feasible computation in C_ϕ, then $\{f_i(z)\}$ is a non-zero Boolean vector, and then $\Pr_\alpha[f\alpha(z) = 1] = \frac{1}{2}$.
- Step 1 in the output gives the right value with probability $1 - 2\delta$.
- Step 2 in the output gives the right value with probability $1 - 2\delta$. ∎

Thus soundness is guaranteed too.

The next lemma is an extension of both Lemma 5.28, Lemma 5.29.

Lemma 5.31. *For every $\Phi : \Sigma \times \Sigma \to \{0,1\}$ there exist $L_1^\Sigma, L_2^\Sigma \in N$ and encodings $H : \Sigma \in \{0,1\}^{L_1^\Sigma}$ and $H_\Phi : \Sigma \times \Sigma \in \{0,1\}^{L_2^\Sigma}$ such that the property $P_\Phi = \{(H(a), H_\Phi(a,b)) | a, b \in \Sigma, \Phi(a,b) = 1\} \cup \{(H(b), H_\Phi(a,b)) | a, b \in \Sigma, \Phi(a,b) = 1\}$ has an LTA A_3, and the encodings H, H_Φ have relative distance at least $1/3$.*

Proof In Lemma 5.29 we used $H_\Phi = H(z \otimes z)$ where z was an extension of the binary representation of (a,b). Thus, we can add in Algorithm 4 to locally test the linear part of $z \otimes z$ which encodes (a,b) using LTA A_1 from Lemma 5.28. ∎

5.8 Alphabet Reduction

We are now ready to prove Theorem 5.27. We first describe the reduction and then prove its correctness. Assume without loss of generality that L_1^Σ, L_2^Σ provided by Lemma 5.28 and Lemma 5.29 are equal, since we can we can use a repetition of the encodings to obtain equality. Moreover, for the reductions, we will use the LTA's A_1, A_2, A_3 of the lemmata.

Let $G = \langle (V, E), \Sigma, \mathcal{C}$. Then the constraint system $\langle V', \Sigma_0 = \{0,1\}, \mathcal{C}' \rangle$ is constructed as follows.

Variables. For every $v \in V$ define L_1^Σ binary variables $[v]$. For every $e = (u, v) \in E$ define L_2^Σ binary variables $[e]$. Define

$$V' = \cup_{v \in V}[v] \bigcup \cup_{e \in E}[e]$$

Constraints. Consider first the following LTA A:

Algorithm 5.
LTA A
Input:

- An assignment $H : V' \to \{0, 1\}$.

Procedure:

- Choose a random $e = (v_1, v_2) \in E$. Test the restriction of a to e using LTA A_2.
- Choose a random $v \in \{v_1, v_2\}$. Test the restriction of a to v using LTA A_1.

Output:

- Test the restriction of a to $[v] \cup [e]$ using LTA A_3. If any of the above tests fails output 0. Otherwise 1.

Note that A reads a constant number of variables, since each of the LTA's does so. Denote this number by q. Moreover, A needs $\log |E|$ bits to choose an edge, 1 bit to choose v plus the bits that each of A_1, A_2, A_3 needs to run. We have seen that the last number depends only on Σ, thus we may assume that it is a constant r_Σ. Thus, A needs $\log |E| + a + r_\Sigma$ bits to run.

It should be clear now that the constraints will simulate the different runnings of A. This can be done in linear time by computing for each $\rho \in \{0, 1\}^{\log |E| + a + r_\Sigma}$ a predicate Φ_ρ for those assignments which cause A to accept (note that there are $O(|E|)$ different possible such strings ρ and $2^q = O(1)$ needed to be checked for each ρ). Define $\mathcal{C}'$ to be

$$\mathcal{C}' = \left\{ \Phi_\rho | \rho \in \{0, 1\}^{\log |E| + a + r_\Sigma} \right\}$$

We have already seen that both $V', \mathcal{C}'$ are 'just' a constant bigger than the input. To complete the proof of Theorem 5.27, we need to prove the completeness and soundness of the final construction.

- Completeness is once again easy, since each of the LTA's that A uses has perfect completeness.
- Suppose that $\textsc{Unsat}(G) > 0$ and let $H : V' \to \{0,1\}$ the optimal assignment for G. Define the assignment $h : V \to \Sigma$ for G such that $h(u)$ to be the value in Σ whose encoding $H(a(v))$ is the closest to the restriction of A to $[v]$.

  Consider an edge $e = (v_1, v_2) \in E$, whose constraint Φ is not satisfied by a, and denote by w the restriction of H to $[e]$. Denote by ϵ the soundness parameter of A_2. We distinct two cases:

  1. If $\Pr[A_2(w) = 0] \geq \frac{\epsilon}{6}$ then A fails with probability at least $\epsilon/6$.
  2. If $\Pr[A_2(w) = 0] \leq \frac{\epsilon}{6}$ then $\exists \sigma_1, \sigma_2 \in \Sigma$ such that $\Phi(\sigma_1, \sigma_2) = 1$ and

  $$\text{dist}(w, H_\Phi) \leq \frac{\Pr[A_2(w) = 0]}{\epsilon} \leq \frac{1}{6}.$$

  Since the assignment h does not satisfy Φ, then it cannot be the case that both $\sigma_1 = h(v_1)$ and $\sigma_2 = h(v_2)$ hold. Note that we peek the wrong assigned $v \in \{v_1, v_2\}$ with probability at least $1/2$. Denote the restriction of A to $[v] \cup [e]$ by w'.

  We will prove that $\Pr[A_3(w) = 0] \geq \frac{\epsilon}{12}$. Note that it suffices to prove that w' has at least $1/12$ distance from the property tested by A_3. Indeed, since the encoding H has relative distance at least $1/3$, $H(a(v))$ must be changed in at least $L_1^\Sigma/6$ bits to encode the correct $\sigma \in \Sigma$. But since L_1^Σ and L_2^Σ are equal (recall the argument in the beginning of the section) w is half the length of w' and consequently w' must be changed in at least $L_3^\Sigma/12$ so as to be a proper encoding that will be accepted by A_3. Hence, w' has at least $1/12$ distance from the property tested by A_3.

  In both cases, $\textsc{Unsat}_H(\mathcal{C}') = \Pr[A = 0] \geq \frac{1}{12}\textsc{Unsat}_h(G) \geq \frac{1}{12}\textsc{Unsat}(G)$.

Thus, Theorem 5.27 holds and alphabet correction was successfully accomplished.

Bibliography

[AC88] Noga Alon and Fan R. K. Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72(1-3):15–19, 1988.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *FOCS*, pages 14–23, 1992.

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

[Aro94] Sanjeev Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. CS Division, UC Berkeley, 1994.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; a new characterization of np. In *FOCS*, pages 2–13, 1992.

[Bab85] László Babai. Trading group theory for randomness. In *STOC*, pages 421–429, 1985.

[Bab91] *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, 6-8 May 1991, New Orleans, Louisiana, USA*. ACM, 1991.

[BCH+95] Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos A. Kiwi, and Madhu Sudan. Linearity testing in characteristic two. In *FOCS*, pages 432–441, 1995.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. In *FOCS*, pages 16–25, 1990.

[BL06] Yonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap*. *Combinatorica*, 26(5):495–519, 2006.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *STOC*, pages 73–83, 1990.

[BOGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *STOC*, pages 113–131, 1988.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.

[Din05] Irit Dinur. The pcp theorem by gap amplification. *Electronic Colloquium on Computational Complexity (ECCC)*, (046), 2005.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *FOCS*, pages 2–12, 1991.

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theor. Comput. Sci.*, 134(2):545–557, 1994.

[GLST98] Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A tight characterization of np with 3 query pcps. *Electronic Colloquium on Computational Complexity (ECCC)*, 5(34), 1998.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[Hås97] Johan Håstad. Some optimal inapproximability results. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(37), 1997.

[Jut06] Charanjit S. Jutla. A simple biased distribution for dinur's construction. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(121), 2006.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KZ97] Howard J. Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *FOCS*, pages 406–415, 1997.

[Lov93] László Lovász. Random walks on graphs: A survey, 1993.

[LW03] N. Linial and A. Wigderson. Expander graphs and their applications. http://www.math.ias.edu/ boaz/ExpanderCourse, 2003.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Pap94] Christos Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[Pin73] M. Pinsker. On the complexity of a concentrator. In *Proceedings of the 7th International Teletraffic Conferenc*, pages 318/1–318/4, 1973.

[Rei04] Omer Reingold. Undirected st-connectivity in log-space. *Electronic Colloquium on Computational Complexity (ECCC)*, (094), 2004.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

Chapter 6

Appendix

6.1 Useful Inequalities

Theorem 6.1 (The Cauchy-Schwarz Inequality). *Let $a_i, b_i \in \mathbb{R}$ for $1 \leq i \leq n$. Then*

$$(a_1 b_1 + \ldots + a_n b_n)^2 \leq (a_1^2 + \ldots + a_n^2)(b_1^2 + \ldots + b_n^2)$$

In vector form the inequality can be written as $\mathbf{x}^{\mathrm{T}}\mathbf{y} \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\|$.

Proof Consider the polynomial

$$\begin{aligned} P(x) &= (a_1 x + b_1)^2 + \ldots + (a_n x + b)^2 \\ &= (a_1^2 + \ldots + a_n^2)x^2 + 2(a_1 b_1 + \ldots + a_n b_n)x + (b_1^2 + \ldots + b_n^2) \end{aligned} \tag{6.1}$$

Clearly $P(x)$ is nonnegative for every $x \in \mathbb{R}$, thus it can have at most one real root. Since $P(x)$ is a degree 2 polynomial this translates into $\Delta \leq 0$. Hence

$$\frac{\Delta}{4} = (a_1 b_1 + \ldots + a_n b_n)^2 - (a_1^2 + \ldots + a_n^2)(b_1^2 + \ldots + b_n^2) \leq 0$$

and the inequality follows.

Equality occurs if the squares in (6.1) can be set simultaneously to zero which is possible iff

$$\frac{a_1}{b_1} = \ldots = \frac{a_n}{b_n}$$

■

Theorem 6.2 (Chebyshev's Inequality). *Let X be a random variable with expected value μ and variance σ^2. Then for any real number $k > 0$,*

$$\Pr[|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

Proof Define

$$g(x) = \begin{cases} 1, & \text{if } |X-\mu| \geq k\sigma \\ 0, & \text{otherwise} \end{cases}$$

Note that $0 \leq g(x) \leq \dfrac{(X-\mu)^2}{k\sigma}$. Then

$$\Pr[|X-\mu| \geq k\sigma) = \mathbb{E}[g(x)] \leq \mathbb{E}[\frac{(X-\mu)^2}{k\sigma}] = \frac{1}{k^2\sigma^2}\mathbb{E}[(X-\mu)^2]$$

The desired inequality follows since

$$\mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2 - 2\mu X + \mu^2] = \mathbb{E}[X^2] - \mu^2 = \sigma^2$$ ∎

Theorem 6.3. *Let X be a random variable which assumes nonnegative values. Then*

$$\Pr[X > 0] \geq \frac{(\mathbb{E}[X])^2}{\mathbb{E}[X^2]}$$

Proof Define

$$g(x) = \begin{cases} 1, & \text{if } X > 0 \\ 0, & \text{otherwise} \end{cases}$$

Then using Cauchy-Schwarz inequality we have that

$$\mathbb{E}[X] = \mathbb{E}[X \cdot g(X)] \leq \sqrt{\mathbb{E}[X^2]} \cdot \sqrt{\mathbb{E}[g(X)^2]}$$

The inequality follows since $\mathbb{E}[g(X)^2] = \Pr[X > 0]$. ∎

6.2 Rayleigh Quotient

Theorem 6.4 (Rayleigh Quotient). *Let $\mathbf{A}$ be a symmetric real matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ respectively. Then*

1. $\lambda_1 = \max\limits_{\mathbf{x} \in \mathbb{R}^n} \dfrac{\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|^2}$.

2. *For $1 < i < n$,* $\lambda_i = \max\limits_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \mathbf{x} \perp \mathbf{e}_1, \ldots, \mathbf{e}_{i-1}}} \dfrac{\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|^2}$.

Proof Let $\mathbf{x} \in \mathbb{R}^n$. Since $\mathbf{e}_i$, $1 \leq i \leq n$ form an orthonormal basis in $\mathbb{R}^n$, $\mathbf{x}$ can be uniquely represented as a linear combination of them. Namely, let

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{e}_i$$

for some scalars c_i. Then

$$\begin{aligned}\|\mathbf{x}\|^2 &= \sum_{i=1}^{n} c_i \mathbf{e}_i^{\mathrm{T}} \sum_{j=1}^{n} c_j \mathbf{e}_j \\ &= \sum_{1 \le i,j \le n} c_i c_j \mathbf{e}_i^{\mathrm{T}} \mathbf{e}_j\end{aligned}$$

Recalling that $\mathbf{e}_i^{\mathrm{T}} \mathbf{e}_j = 1$ iff $i = j$, we obtain

$$\|\mathbf{x}\|^2 = \sum_{i=1}^{n} c_i^2$$

Similarly, using that $\mathbf{A}\mathbf{e}_j = \lambda \mathbf{e}_j$,

$$\begin{aligned}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} &= \left(\sum_{i=1}^{n} c_i \mathbf{e}_i^{\mathrm{T}} \right) \left(\sum_{j=1}^{n} c_j \mathbf{A}\mathbf{e}_j \right) \\ &= \left(\sum_{i=1}^{n} c_i \mathbf{e}_i^{\mathrm{T}} \right) \left(\sum_{j=1}^{n} c_j \lambda_j \mathbf{e}_j \right) \\ &= \sum_{1 \le i,j \le n} c_i c_j \lambda_j \mathbf{e}_i^{\mathrm{T}} \mathbf{e}_j \\ &= \sum_{i=1}^{n} c_i^2 \lambda_i\end{aligned}$$

Assuming that $\mathbf{x} \perp \mathbf{e}_i$ for some $1 \le i \le n$, we get that $\mathbf{x}^{\mathrm{T}} \mathbf{e}_i = 0$, which translates into $c_i = 0$. Thus,

$$\max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \mathbf{x} \perp \mathbf{e}_1, \ldots, \mathbf{e}_{i-1}}} \frac{\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|^2} = \frac{\displaystyle\sum_{j=i}^{n} c_j^2 \lambda_j}{\displaystyle\sum_{j=i}^{n} c_i^2} \le \frac{\lambda_i \displaystyle\sum_{j=i}^{n} c_j^2}{\displaystyle\sum_{j=i}^{n} c_i^2} = \lambda_i^2$$

It is trivial to see that equality can be obtained by letting $c_j > 0$ iff $j = i$. Hence, the second part of the theorem follows. With slight modifications to the above argument we can clearly prove the first part also. ∎