Expander Graphs, Randomness Extractors and Error Correcting Codes

by

Aristeidis Tentes

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

Graduate Program in Logic, Algorithms and Computation

University of Athens, National Technical University of Athens, University of Patras

Committee:

May 2009

Abstract

The purpose of this thesis is to introduce three important notions of Complexity, Cryptography and generally Pseudorandomness. We survey the notions of Expander Graphs, Randomness Extractors and List Decodable Error Correcting Codes. The former are graphs that on the one hand are very sparse, namely they do not have many edges, yet on the other hand are very well connected. Randomness Extractors are function that take as input a string of imperfect randomness and output a (close to) uniformly random string. List Decodable Error Correcting Codes have seemingly little relevance with Pseudorandomness. They try to solve the problem of communication over a network that introduces higher noise than can be tolerated by simple Error Correcting Codes. The decoding procedure does not uniquely decode a codeword, but rather gives a list that includes the original message.

All three objects have their own body of research, each one with seemingly irrelevant techniques. However, it can be shown that they can be expressed in terms of the other. That is, Expander Graphs and Randomness Extractors can be expressed in the language of List Decodable Error Correcting Codes.

Acknowledgements

First of all I would like to thank my thesis advisor Stathis Zachos. He was the one that encouraged me to apply for the program and his advise has always been invaluable. I owe him a lot. Secondly, I would like to thank Aris Pagourtzis. It was him teaching the class Cryptography that inspired me and started reading papers and doing research. I also owe him for introducing him to the group of Theoretical Computer Science. That was very important to me. Then I would like to thank my teachers Yiannis Moschovakis, Kostas Dimitrakopoulos and Elias Koutsoupias for their dedication to teaching and being always helpful. Moreover, I would like to thank the third member of my committee Dimitris Fotakis.

Contents

Chapter 1

Introduction

Pseudorandomness plays an extremely important role in the theory of Computational Complexity and Cryptography. Randomized algorithms, namely algorithms which expects as input a random string, are expected to have the desired output with large probability over this random string. In many cases however, the question is if we can derandomize these algorithms and make them deterministic. The first step towards achieving that, is to reduce the length of the random string as much as possible. On the other hand, randomness is essential for Cryptography, without which Privacy and secrecy are impossible. However, quality randomness is sometimes expensive and difficult to be produced. To this end, reducing the amount of required perfect randomness is crucial. The field of Pseudorandomness focuses exactly on this goal, to use a small amount of truly random bits and produce a larger amount of bits, which behave almost as well as if they were truly random.

In this thesis we survey two important pseudorandom objects, i.e., Expander Graphs and Randomness Extractors together with List Decodable Error Correcting Codes. The motivation of the latter, had initially little to do with Pseudorandomness, however, there appears to be a surprising connection between the former two and List Decodable Error Correcting Codes. In particular, each such object can be described in the language of the other. Indeed, this connection has facilitated the construction of one object, based on previous constructions of the others.

For each case, we will survey important constructions. In the following we describe each aforementioned object and mention what is included in the main body of this thesis.

1.1 Expander Graphs

Expanders are graphs that satisfy two properties which seam contradictive. The first property, is that they are sparse, namely they have a small number of edges (linear in the number of nodes). The other property is that they are well connected. There are three ways to define the latter. The first, which is called vertex expansion captures that starting from an arbitrary node, we can reach any other node in only a few steps. Put it differently the diagonal of such graph is small, logarithmic in the number of nodes. The second is called spectral expansion and captures that a random walk mixes very quickly. The third is called edge expansion and requires that the number of outgoing edges of any set of nodes to nodes outside this set is large.

Expander graphs were first used for fault tolerant network design. However, they have numerous applications in Theoretical Computer Science. Here we first show that such graphs do exist. More specifically, a random regular graph is a good vertex expander with high probability. Then we give the definition of a bipartite expander graph and show again via the probabilistic method that a random regular bipartite graph has good expansion with good probability. Then we formally define spectral expansion and show the connection among the three expansion notions. That is, a graph that has good expansion with respect to one notion, also has good expansion with respect to the other notions. Then we present three explicit constructions of expander graphs. The first two are given by Margulis and Lubotsky-Phillips-Sarnak and the third one is the so called Zig-Zag Construction. The proof of the first two relies on Fourier Analysis and deep number theoretic theorems dealing with Ramanujan Graphs and therefore are omitted. The last construction is purely combinatorial and we present a proof.

1.2 Randomness Extractors

A Randomness Extractor ideally takes as input imperfectly random bits and outputs a shorter string that is uniformly random. It is easy to show that it is impossible to have a construction that for any even slightly imperfectly random string outputs a random string (even when we relax the requirement and require an output of close to random string). However, if the extractor is allowed to expect as input a (preferably) small random string (called seed from now on), then we can bypass this impossibility result. Randomness extractors are extremely important both in Computational Complexity and in Cryptography. For the former, we ideally want an extractor that has a seed of minimum length. For cryptography, extractors are important because they can guarantee privacy even in the presence of an adversary who can read some random bits used by honest parties. Here we first show via the probabilistic method that a random function with logarithmic seed length is a good extractor. Then, we show an explicit construction that achieves extraction with a squared logarithmic long seed.

1.3 List Decodable Error Correcting Codes

Error Correcting Codes have been developed to solve the problem of communication over a noisy network. In particular suppose that a sender wants to transmit a message to another party over a network that introduces arbitrary and unpredictable errors. An Error Correcting Code guarantees that if the number of errors introduced is not too large, then the transmitted encoded message, can be decoded to the original message. However, there are cases where the noise introduced by the network is so large that perfect error correction is impossible. However, in order to bypass this impossibility, we can encode the message in such a way that the decoding outputs a list of messages, with the guarantee that this list includes original message. In such a way we can tolerate larger noise by the network.

Here, we first present the well known Reed-Solomon Codes. Then we present the Berlekamp-Welch decoding algorithm. Then we show the list decoding properties of Reed-Solomon Codes. Then we present an improvement over the list decoding version of Reed Solomon Codes, called the Parvaresh-Vardy Codes. In the end we show that both Randomness Extractors and Expander Graphs, two seemingly irrelevant objects, can be expressed in terms of List Decodable Error Correcting Codes and vice versa.

Chapter 2

Expander Graphs

2.1 Definitions and Existence

Expander graphs are very important both in computers science and in mathematics. They have numerous applications and therefore they are studied a lot. These graphs have the nice property, that despite being sparse (they do not have many edges) they are very well connected. But how can we express this property. Firstly, we can say that every set of nodes has a lot of nodes in its neighborhood. Another way of expressing well-connectedness is requiring that between any two complementary sets there are many edges connecting them, in other words they have a large Min-Cut. We can also require that random walks converge to uniform distribution very quickly.

We will see general definitions of the first two approaches and see for what parameters these graphs are considered to be good expander. Then we will see that these approaches are almost equivalent, that is a good expander according to one definition is also a good expander according to the other definitions.

Before defining Expander graphs let us see some useful definitions, which we will use:

Definition 1.

1. A graph is d-regular, if all its vertices have degree d.
2. The neighborhood $N(S)$ of a set of vertices of a graph $G(V,E)$ is the set $\{u | \exists v \in S s.t. (u,v) \in E\}$
3. $N'(S) = N(S) \setminus S$
4. $e(S,T) = \{(u,v) \in S \times T | (u,v) \in E\}$

Vertex Expansion In our definitions we allow multigraphs and vertices with selfloops. Let us see the first definition of expansion:

Definition 2 (Vertex Expander)**.** A graph G is called (K, A)- *Vertex Expander, if for all sets S of at most K vertices* $|N(S)| \geq A|S|$.

A good expander, as we said, has to be sparse and therefore we would like that the graph has vertices with constant degree. So a good expander is a regular graph. Moreover, we would like the parameter K to be a fraction of the number of vertices of the graph, namely $K = \Omega(N)$ and the parameter A to be as close as possible to D.

Observe that a good expander graph, with $A = 1 + \Omega(1)$ has a small diameter. It is easy to see that its diameter has logarithmic size with respect to its vertices. Although we have defined good expander graphs we have not shown even if there exist such graphs. The following theorem can be proven, [Pin73, Bas81]:

Theorem 1. For all $D \geq 3$ and for all sufficiently large even N, if we choose a random D-regular graph with N vertices, then with probability at least $1/2$ it is a $(aN, D - 1.01)$-vertex expander for some fixed constant a.

Here, we will prove a variant of this theorem using the probabilistic method. We will show that there exist $(1/2N, 1 + c)$ 3-regular vertex expanders (for some fixed constant c and every sufficiently large N).

Proof. We will take three randomly selected perfect matchings of the complete graph K_N (a perfect matching is a set of edges, where no pair of edges share the same vertex and every vertex of the graph can be found in one of these edges) and view them as graphs G_1, G_2 and G_3. Then we will merge these tree graphs in the natural way. The resulting graph is obviously 3-regular although it may be a multigraph. Then we will prove that this graph has the desired properties with non zero probability (which proves the existence of this kind of Expander Graphs.

As we want to take perfect matchings we require the number of vertices to be even. The probability of the complement event, namely the event that there is a set V, with at most $1/2N$ vertices with $|N'(V)| < c|V|$, can be bounded (using the Union Bound) as follows:

$$\sum_{V:|V|\leq 1/2N} Pr[|N'(V)| < c|V|]$$

and further as

$$\sum_{V:|V|\leq 1/2N} \sum V' : |V'| = c|V| Pr[N'(V) \subset |V'|]$$

We want to count the number of additions in the above formula. The number of different sets of size i is $\binom{N}{i}$ and given a specific set of i vertices the number of sets we may take into consideration in the second sum is $\binom{N-i}{ci}$, namely all the sets of vertices, which are disjoint with the first set. Due to symmetry we may fix the sets V, V' and take some arbitrary disjoint sets V_0, V_0', say $V_0 = \{1, \ldots, i\}$ and $V_0'' = \{i, \ldots, i + ci\}$. Thus we can bound the previous with

$$\sum_{i=1}^{N/2} \binom{N}{i} \binom{N-i}{ci} Pr[N'(V_0) \subset |V_0''|]$$

This can also be bounded by

$$\sum_{i=1}^{N/2} \binom{N}{i} \binom{N-i}{ci} Pr[N(V_0) \subset |V_0'|,]$$

where $V_0' = V_0 \cup V_0''$. So it remains to bound $Pr[N(V_0) \subset |V_0'|]$. That is we want to bound the probability, with which all vertices in V_0 were matched with vertices in V_0' in every of the three matchings. This probability in a single matching is at most

$$\frac{i + ci - 1}{N-1} \frac{i + ci - 3}{N-2} \cdots \frac{i + ci - i + 1}{N - i + 1} = \prod_{j=1}^{i/2} \frac{i + ic - 2j + 1}{N - 2j + 1}$$

and doing some math we have

$$\prod_{j=1}^{i/2} \frac{i + ic - 2j + 1}{N - 2j + 1} < \prod_{j=0}^{i/2-1} \frac{i + ic - 2j}{N - 2j}$$

$$= \frac{\prod_{j=0}^{i/2-1} (i + ic)/2 - j}{\prod_{j=0}^{i/2-1} N/2 - j}$$

$$= \frac{\binom{(i+ic)/2}{i/2}}{\binom{N/2}{i/2}}$$

As the matchings were done independently we can take the cube of the above and have the following bound of the probability we want:

$$\sum_{i=1}^{N/2} \frac{\binom{N}{i}\binom{N-i}{ci}\binom{(i+ic)/2}{i/2}^3}{\binom{N/2}{i/2}^3}$$

Now we have to use the approximation of the binomial coefficients, which is $\binom{n}{an} \approx 2^{H^2(a)n}$, where $H_2(p) = p \log 1/p + (1-p)\log 1/(1-p)$ the Shannon Entropy of a random biased bit. So if we take the logarithm for some i, we have

$$\log \frac{\binom{N}{i}\binom{N-i}{ci}\binom{(i+ic)/2}{i/2}^3}{\binom{N/2}{i/2}^3}$$

$$\approx \ldots = -(H(i/N) - 2H(ci/N))N/2 + 3H(1/(1+c))\frac{i+ci}{2}$$

But for sufficiently small $c > 0$ and because $2i \leq N$, we have $H(i/N) < c \log c H(ci/N)$ and $H(\frac{1}{1+c}) \approx H(c) \approx c \log 1/c$, which gives in the summation

$$\sum_{i=1}^{N/2} 2^{-(1-c')H(i/N)N/2+2c'i},$$

for $c' \approx c \log 1/c$. In this form we can see that as c approaches to zero and N to infinity, the summation approaches to zero, which means that there is a constant c, such that for all sufficiently large N the probability that a random 3-regular (multi)graph is a $(1/2N, 1+c)$ vertex expanders is nonzero, proving thee existence of such graphs. □

Another definition, which we will consider in this thesis is the definition of bipartite vertex expander graphs. In this case we want that the neighborhood of every set of the left vertices to be large.

Definition 3. A bipartite graph G is called a (K, A)-vertex bipartite expander graph, if every set S of at most K left vertices has the property $N(S) \geq A|S|$.

There is a similar to the previous theorem about the existence of good vertex bipartite expander graphs, which is also proven using the probabilistic method.

Theorem 2. For all $D \geq 3$ and for all sufficiently large N, if we choose a random D-regular on the left bipartite graph with $2N$ vertices, then with probability at least $1/2$ it is a $(aN, D-2)$-vertex expander for some fixed constant a.

Of course we can prove a similar theorem, which is analogue to what we proved previously. That is for all constants $a < 1$ and $D > 2$, there exists a constant A such that for every sufficiently large N, if we take a random D-regular on the left bipartite graph, then with high probability it is an

(aN, A)-vertex bipartite expander graph. Moreover, we can prove that if $D > \frac{H(a)+H(aA)}{H(a)-aAH(1/A)}$, then a random D-regular graph with N vertices is as (aN, A) vertex Expander with high probability.

Edge Expansion This measure of Expansion counts the edges which connects two complementary sets of vertices. More formally:

Definition 4. A D-regular graph is called (K, ϵ)-edge Expander, if for all sets S of size at most K, $|e(S, S)| \geq \epsilon |S| D$.

We can see that there is a connection between Vertex and Edge Expansion. If $A > 1$, then a (K, A)-vertex D-regular expander graph is also a $(K, (A-1)/D)$-edge expander graph. However, this expansion measure is also closely related to the next expansion measure.

Spectral Expansion In this definition of Expansion we want to capture the idea that well connectedness for a graph means that the random walks in this graph mix very quickly (converge to the stationary distribution very fast). As we can see in the last section of this chapter, the mixing time of a regular graph is closely connected with the second eigenvalue of the normalized matrix.

Definition 5. Let $\lambda_2|$ denote the second biggest absolute eigenvalue of a regular graph G. G is a λ-Spectral Expander if $\lambda_2 \leq \lambda$.

Quite often these graphs are characterized by the spectral gap $\gamma = 1 - \lambda$. Now we will see that the characterizations spectral expander and vertex expander are almost equivalent.

Firstly we are going to see that a λ spectral expander is also a vertex expander, where the smaller the spectral expansion is the better the vertex expansion is, [Tan84, AM84].

Theorem 3. If G is a λ-spectral expander, then for all a, G is also a $(aN, \frac{1}{(1-a)\lambda^2+a})$ vertex expander.

To prove this theorem we are going to use the following definitions:

Definition 6. Let the N-dimensional vector π be a probability distribution.

1. $CP(\pi)$ is the probability that two independent samples taken according to π are equal, namely $CP(\pi) = \sum_i \pi_i^2$. (CP = collision probability)

2. The support of π is the set of indices of its nonzero entries, namely $Supp(\pi) = \{i | \pi_i > 0\}$

The following lemma shows some properties of the above defined items.

Lemma 1. For every N-dimensional vector π, which is a probability distribution we have

1. $CP(\pi) = ||\pi||^2 = ||\pi - u||^2 + \frac{1}{N}$
2. $CP(\pi) \geq \frac{1}{Supp(\pi)}$

Proof. 1. The first equality holds because of the definitions. For the second we calculate $||\pi - u||^2 = \sum_i (\pi_i - 1/N)^2 = \sum_i \pi_i^2 - \sum_i 2\pi_i/N + \sum 1/N^2 = ||\pi||^2 - 2/N + 1/N$. This means $||x||^2 = ||\pi - u||^2 + \frac{1}{N}$

2. The second part can be derived from the Cauchy-Schwartz inequality, namely

$$1 = \sum_i \pi_i \leq \sqrt{|Supp(\pi)|}\sqrt{\sum_i \pi_i^2}$$

□

The proof of the main theorem follows.

Proof. From what we have seen in the last section of this chapter we have that for all probability distributions π, $\frac{||\pi M - u||}{||\pi - u||} \leq \lambda$. Using the fact that πM is also a probability distribution and using the first part of the previous lemma we have for every probability distribution π

$$\frac{||\pi M - u||}{||\pi - u||} = \frac{CP(\pi M) - 1/N}{CP(\pi M) - 1/N} \leq \lambda.$$

Now let S be any set of at most aN vertices and π the uniform probability over the vertices of S. Now we have that

$$CP(\pi) = \frac{1}{|S|} \text{ and}$$

$$CP(\pi M) \geq \frac{1}{|Supp(\pi M)|} = \frac{1}{|N(S)|}$$

because of the second part of the lemma and the fact that $Supp(\pi M)$ is the set of all nodes reachable from S in one step. If we substitute in the above inequality, we have

$$(\frac{1}{N(S)} - \frac{1}{N}) \leq \lambda^2(\frac{1}{|S|} - \frac{1}{N})$$

$$\frac{1}{N(S)} \leq \lambda^2(\frac{1}{|S|} - \frac{1}{N}) + \frac{1}{N}$$

$$\frac{1}{N(S)} \leq \lambda^2(\frac{1}{|S|} - \frac{a}{|S|}) + \frac{1}{N} \text{ because} |N(S)| \geq |S|/a$$

$$|N(S)| \geq \frac{|S|}{(1-a)\lambda^2 + a},$$

which means that G is a $(aN, \frac{1}{(1-a)\lambda^2+a})$ vertex expander. □

□

The next theorem shows the other direction. It shows what we can say about the spectral expansion, when we have the vertex expansion.

Theorem 4. For every $\delta > 0$ and $D > 0$, there exists a $\gamma > 0$ such that if G is a D-regular $(N/2, 1+\delta)$ vertex expander, then it is also a $(1-\gamma)$ spectral expander, with $\gamma = \Omega(\frac{\delta^2}{D})$.

The next lemma shows that a good expander graph behaves very well as a random graph. That is if we select two random sets of vertices, then the number of edges connecting these two sets is very close to the expected number of edges as if the graph was selected at random, [AC88].

Lemma 2 (Expander Mixing Lemma)**.** If G is a D-regular, λ spectral expander with N vertices, then for all sets of vertices S, T, we have

$$|\frac{|e(S,T|}{ND} - \frac{|S| \cdot |T|}{N^2}| \leq \lambda\sqrt{\frac{|S| \cdot |T|}{N^2}}$$

Proof. We denote by χ_S, χ the characteristic vectors of S, T respectively. Observe that

$$|e(S,T)| = \chi_S(DM)\chi',$$

where DM is the matrix of the random walk on G multiplied by D, namely the adjacency matrix.Each entry of the vector $\chi_S(DM)$ counts the number of neighbors of the corresponding vertex in the set S. The next multiplication adds for each vertex in T the number of its neighbors in S, that is the number of edges between S and T.

As we have seen, the eigenvectors of matrix M span $\mathbb{R}^N$, which means that the vectors χ_S, χ can be written as linear combination of u and another vector orthogonal to u. Let $\chi_S^\perp, \chi^\perp$ denote these orthogonal vectors respectively. The projection of χ_S on u is given (as we know from linear algebra) by $\chi_S u$, which means that the coefficient of u in the linear decomposition of χ_S into u and $\chi_S^\perp$ is given by $\frac{\chi_S u}{||u||^2}$. However, this coefficient is equal the cardinality of S. As the same thins hold for T as well, we have

$$\chi_S = |S|u + \chi_S^\perp \text{and} \chi_T = |T|u + \chi_T^\perp.$$

Therefore we have

$$\frac{|e(S,T|}{ND} = \frac{1}{N}(|S|u + \chi_S^{\perp})M(|T|u + \chi_T^{\perp})'$$

$$= \frac{1}{N}|S||T|uMu' + \frac{1}{N}|S|uM\chi_T^{\perp'} + \frac{1}{N}|T|\chi_S^{\perp}Mu' + \frac{1}{N}\chi_S^{\perp}M\chi_S^{\perp'}$$

However, since u is an eigenvector we have $uM = u$, implying that $Mu' = u'$ (M is symmetric) and $uM\chi_T^{\perp'} = u\chi_T^{\perp'} = 0$. Therefore, from the above we have

$$\begin{aligned}\tfrac{|e(S,T)|}{ND} &= \tfrac{1}{N^2}|S||T| + \tfrac{1}{N}\chi_S^{\perp}M\chi_S^{\perp'} \\ &= \tfrac{1}{N^2}|S||T| + \tfrac{1}{N} < \chi_S^{\perp}M, \chi_S^{\perp'} > \\ &\leq \tfrac{1}{N^2}|S||T| + \tfrac{1}{N}||\chi_S^{\perp}M|| \cdot ||\chi_S^{\perp}|| \text{ by the Cauchy-Schwartz} \\ &\leq \tfrac{1}{N^2}|S||T| + \tfrac{1}{N}\lambda||\chi_S^{\perp}|| \cdot ||\chi_S^{\perp}|| \text{ by definition} \\ &\leq \tfrac{1}{N^2}|S||T| + \tfrac{1}{N}\lambda\sqrt{|S||T|} \\ &\leq \tfrac{1}{N^2}|S||T| + \lambda\sqrt{\tfrac{|S||T|}{N^2}}\end{aligned}$$

From the first line we can see that $\frac{|e(S,T|}{ND} \geq \frac{1}{N^2}|S||T|$ and therefore we can conclude

$$|\frac{|e(S,T|}{ND} - \frac{|S| \cdot |T|}{N^2}| \leq \lambda\sqrt{\frac{|S| \cdot |T|}{N^2}}\square$$

□

In fact the 'converse' also holds, [BL06]:

Lemma 3 (Converse of Expander Mixing Lemma)**.** If G is a D-regular graph with the property that for all sets of vertices S, T we have $|\frac{|e(S,T|}{ND} - \frac{|S|\cdot|T|}{N^2}| \leq \theta\sqrt{\frac{|S|\cdot|T|}{N^2}}$ for a fixed θ, then G is a λ spectral expander, with $\lambda = O(\theta \log D/\theta)$.

This means that λ is the best, up to a $\log D$ factor, of what we can have in the expander mixing lemma. The next theorem shows the connection between spectral expansion and edge expansion.

Theorem 5. Let G be an $(\frac{N}{2}, \epsilon)$-edge expander, with ϵ the highest possible. Then

$$\frac{1-\lambda}{2} \leq \epsilon \leq \sqrt{2(1-\lambda)}.$$

We are going to prove the first part:

Proof. For any set of vertices let x be an n-dimensional vector such that

$$x_i = \left\{ \begin{array}{cc} \frac{1}{|S|} & \text{if } x \in S \\ -\frac{1}{|\overline{S}|} & \text{if } x \in \overline{S} \end{array} \right\}$$

Note that $u \perp x$, since the sum of all entries is 0 and $< x, x >= \frac{1}{|S|} + \frac{1}{|\overline{S}|}$.

Thus, we have
$$\begin{array}{ll} < Ax, x > & = \sum a_{ij} x_i x_j \\ & \sum_{i,j \in S} \frac{2}{D} x_i x_j + \sum_{i,j \in \overline{S}} \frac{2}{D} x_i x_j + \sum_{i \in S, j \in \overline{S}} \frac{2}{D} x_i x_j \\ & \frac{2}{D|S|^2} \frac{D|S| - e(S,\overline{S}}{2} + \frac{2}{D|\overline{S}|^2} \frac{D|\overline{S}| - e(S,\overline{S}}{2} - \frac{2e(S,\overline{S})}{D|S|S\overline{S}|} \\ & (\frac{1}{|S|} + \frac{1}{|\overline{S}|})(1 - e(S,\overline{S})(\frac{1}{|S|} + \frac{1}{|\overline{S}|})) \end{array}$$

However, since $< Ax, x > \leq \lambda < x, x >$ we have

$$1 - e(S,\overline{S})(\frac{1}{|S|} + \frac{1}{|\overline{S}|}) \leq \lambda$$

which implies

$$\frac{e(S,\overline{S})}{D} \geq \frac{(1-\lambda)|S||\overline{S}|}{|S| + |\overline{S}|} \geq \frac{(1-\lambda)|S|}{2},$$

where the last inequality holds because $|S| \leq N/2$. $\square$

2.2 Random Walks on Expander Graphs

We know (see the last section of this chapter) that the mixing time of a random walk on a regular graph is $O(\frac{\log n}{1-\lambda})$. This is the number of steps we have to do in order to end up to an almost uniform vertex. However, in the case of expanders the spectrum λ is bounded by a constant, therefore the mixing time in expander graphs is $O(\log n)$. This means that after $O(\log n)$ steps the probability with which the random walk ends at vertex v is $\frac{1 \pm 0.01}{N}$, very close to uniform distribution.

However, if we want to choose a random vertex we only require $\log n$ random bits and if we choose a random vertex using a random walk then we require $O(D \log n)$ and if D is constant then the number of random bits required is optimal up to constant factor. So what do we gain in randomness using expander graphs? In fact, we will see that not only the last vertex of the random walk is close to uniform but also the sequence of vertices until we reach the final vertex turn out to be very close to uniform independent samples.We will see this property and see how can we use it to reduce the number of random bits required in randomized algorithms of languages in RP and BPP.

Theorem 6 (Expander Hitting Property, [Kah95])**.** Let G be a λ spectral expander and B a set of its vertices, with density $\mu = \frac{|B|}{|V|}$. Let $(V_1, \ldots, V_t)$ be the sequence of the vertices visited during a random walk of t steps starting at V_1 selected uniformly at random. Then the probability that the random walk never visited a vertex outside of B is

$$Pr[V_1, \ldots, V_t \in B] \leq (\mu + \lambda(1-\mu))^t$$

Before we prove this theorem let us see some tools we are going to use.

Definition 7. For any $N \times N$ matrix M its norm is defined to be

$$||M|| = \max_{x \in \mathbb{R}^N} \frac{||xM||}{||x||}.$$

Lemma 4. If M is symmetric then $||M|| = |\lambda_1|$, where λ_1 its largest eigenvalue. If M is the normalized vector of the random walk on a graph then its norm equals 1.

The proof of this lemma is very much alike with a proof concerning the second largest eigenvalue given in the last section of this chapter. The next lemma shows a general way of matrix decomposition.

Lemma 5. For every normalized matrix M with spectral expansion λ and every vector v, there exist matrices J, E, such that

$$vM = v((1-\lambda)J + \lambda),$$

where J is a matrix which projects onto direction u and E has the property $||vE|| \leq ||v||$.

Proof. Every vector v can be written as $v^{||} + v^{\perp}$, where $v^{||}$ is collinear with u and $v^{\perp}$ orthogonal to u. Let J be the matrix which projects on u. Since u is an eigenvector of M, we have

$$vM = v^{||}M + v^{\perp}M = v^{||} + v^{\perp}M$$

$$= (1-\lambda)v^{||} + (\lambda v^{||} + v^{\perp}M) = (1-\lambda)vJ + \lambda vE = v((1-\lambda)J + \lambda E).$$

We take the norms

$$||vM|| = ||(1-\lambda)vJ + \lambda vE|| \leq ||(1-\lambda)vJ|| + ||\lambda vE||,$$

using the triangular inequality. But since J is a projection matrix, we have that $||vJ|| \leq ||v||$. So if we divide above with $||v||$, we get

$$\frac{||vM||}{||v||} \leq (1-\lambda) + \lambda\frac{||vE||}{||v||}.$$

This holds for all v and as the second term of the right side is positive we can take the maximum on both sides. Namely

$$||M|| \leq (1-\lambda) + \lambda \max_v \frac{||vE||}{||v||},$$

but $||M|| = 1$ as we can seen in the last section of this chapter. Thus,

$$\lambda \leq \lambda \max_v \frac{||vE||}{||v||}$$

and the lemma follows.

$\square$

Let us see now the proof of the theorem:

Proof. Let P be a diagonal matrix such that $P_{ii} = 1$ iff $i \in B$ and $P_{ii} = 0$ otherwise. This matrix gives us an algebraical way of expressing that the random walk does not go to a node outside of B. Observe that if a node is chosen with distribution π, then the probability with which the node is in B is given by $||\pi P||_1$. This is because the nonzero entries of the vector πP correspond to the vertices of B and give the probability of being chosen. Thus if we add all these probabilities we have the probability of choosing a vertex in B.

Generalizing the above observation we see that the probability that a random walk does not leave B in any step is given by $||uP(MP)^{t-1}||_1$. The factor uP is a vector which assigns the probabilities to the vertices of B of being chosen in the first step. If multiply by M, the resulting vector assigns the probability to each vertex of being chosen in the second step, while in the first step we chose a vertex in B. Multiplying by P, we zero out the vertices outside of B. So for each consequent step we multiply with MP. The sum of the resulting vector gives the desired probability.

Since $P^2 = P$, we can use $||uP(PMP)^t||_1$. Therefore the probability of never leaving B is

$$\begin{aligned} ||uP(PMP)^t||_1 &\leq \sqrt{\mu N}||uP(PMP)^t|| && \text{by Cauchy-Schwartz} \\ &\leq \sqrt{\mu N}||uP|| \cdot ||(PMP)||^t && \text{by definition of matrix norm} \\ &\leq \sqrt{\mu N}\sqrt{\tfrac{\mu}{N}} \cdot ||PMP||^t \end{aligned}$$

Thus we have to bound $||PMP||$ and here we are going to use the previous lemma. We are going to prove that $||PMP|| \leq \mu + \lambda(1-\mu)$. We have

$$\begin{aligned} ||PMP|| &= ||P((1-\lambda)J + \lambda)|| \\ &\leq (1-\lambda)||PJP|| + \lambda||PEP|| && \text{by triangular inequality} \\ &\leq (1-\lambda)||PJP|| + \lambda \end{aligned}$$

The last step follows from the fact that P only erases some entries of a vector and therefore never increases the norm of the vector and the previous lemma. It remains to show that $||PJP|| \leq \mu$. We want to bound $||xPJP||$. Let $y = xP$, we saw that $||y|| \leq ||x||$. As we said J is the matrix, which projects a vector on u. That is $yJ = \frac{<y,u>}{||u||^2}u$ and because B has μN vertices

$$yJ = \sum_i y_i u$$

and

$$||yJP|| = ||\sum_i y_i uP|| \leq \sqrt{\mu N}||y||\sqrt{\frac{\mu}{N}} \leq \mu||x||.$$

Now, if we substitute

$$||uP(PMP)^{t-1}||_1 \leq \mu(\mu + \lambda(1-\lambda))^{t-1} \leq (\mu + \lambda(1-\lambda))^t$$

□

The above theorem suggests that we can use expander graphs for error reduction in RP algorithms without using too many random bits. An algorithm is in RP if for any input which does not belong to the language never accepts, but for an input which does belong to the language there is a probability, say 1/2, over the random bits chosen, with which it errs. If the number of random bits required is m then if we want to reduce the error probability to 2^{-k}, we can repeat the algorithm k times and each time we choose fresh random bits. That is we have to choose km random bits. However, if we have a D-regular expander graph, with 2^m vertices, we can choose at random the first vertex, using m random bits and then performing $O(k)$ steps of the random walk, where each steps requires $\log D$ random bits. The random bits we are going to use in each repetition of the algorithm are given by the vertices of the random walk. The above theorem says that if λ is small enough there exists a constant c such that ck steps are enough to reduce the error to 2^{-k}. The set B is the set with the bad choices. The next theorem shows that we can also use expander graphs in BPP algorithms to reduce the error probability, [Gil98, Hea08].

Theorem 7. Let G be a λ spectral expander with N vertices and $f : [N] \rightarrow [0,1]$. If $V_1, \ldots, V_t$ is the sequence of a random walk on G with V_1 chosen uniformly at random, then for every $\epsilon > 0$

$$Pr[|\frac{1}{t}\sum_i V_i - E[f]| > \lambda + \epsilon] \leq 2e^{-\Omega(\epsilon^2 t)}.$$

Proof. We are going to apply a Chernoff bound to prove this result. Let $X_i = f(V_i)$ and $X = \sum_i X_i$. Like in the classic Chernoff Bounds we are going to bound the probability with which $e^{rX} = \prod_i e^{rX_i}$ is larger than $e^{rE[X]}$ and use Markov's inequality for a suitable choice of r. The problem here is that X_i's are not independent and therefore it is difficult to compute $E[e^{rX}]$.

Let P be a diagonal matrix with $P_{ii} = e^{rf(i)}$. Now using the same ideas as in the previous proof, we have

$$E[e^{rX}] = |uP(MP)^{t-1}|_1 = |u(MP)^t|_1 = \sqrt{N}||u|| \cdot ||MP||^t = ||MP||^t$$

For the first equality the arguments are very similar to those in the previous proof. The second equality simply exploits the fact that u is an eigenvector of M. The first inequality holds because of the Cauchy-Schwartz inequality and the definition of the norm of a matrix. Now we apply the matrix decomposition we saw in the previous lemma

$$||MP|| = (1-\lambda)||JP|| + \lambda||EP||.$$

Now we are going to bound the two norms on the right side. However, J is just a projection on u, reducing the norm of the projected vector. Therefore,

$$\begin{aligned} ||JP||^2 &\leq \tfrac{||uP||^2}{||u||^2} \\ &= \tfrac{\sum_i (e^{rf(i)}/N)^2}{1/N} \\ &= \tfrac{1}{N}(\sum_i (e^{2rf(i)}) \\ &\leq 1 + 2rE[f] + O(r^2) \end{aligned}$$

for $r \leq 1$ by the Taylor expansion. Therefore,

$$||JP|| \leq 1 + rE[f] + O(r^2).$$

For the other norm we have $||EP|| \leq ||P||$, because $||E|| \leq 1$, and as all the entries of the diagonal of P are 1 or e^r

$$\frac{||vEP||}{||v||} \leq \frac{||vP||}{||v||} \leq \frac{e^r||v||}{||v||} = e^r = 1 + r + O(r^2).$$

If we substitute in the decomposition we have

$$||MP|| \leq (1-\lambda)(1+2rE[f]+O(r^2)) + \lambda(1+r+O(r^2)) = 1+(\lambda+E[f])r+O(r^2)$$

and therefore

$$E[e^{rX}] \leq (1 + (\lambda + E[f])r + O(r^2))^t \leq e^{([f]+\lambda)rt+O(r^2t)}$$

and if we set $r = \frac{\epsilon}{c}$ for large enough c and apply Markov's inequality

$$Pr[X \geq ([f] + \lambda + \epsilon)t] \leq e^{-\Omega(\epsilon^2 t)}.$$

$\square$

In the case of a BPP algorithm, we want to compute the mean value of the algorithm, namely f is the algorithm. As we can see λ has to be quite small to be able to use the random walk on the expander graph. If ϵ is a constant then the number of steps of the random walk are $O(k)$ to reduce the error to 2^k and thus the number of the random bits is again $m + O(k)$.

2.3 Constructions of Expander Graphs

We have seen that expander graphs do exist, but the proof was non constructive. So the next natural problem, which arises is how to construct a family of good expander graphs explicitly. When we say explicitly we mean that the construction must have the following properties:

1. The graph can be constructed in $O(N^k)$ time, where N the number of vertices and k a constant.

2. The i-th neighbor of any vertex can be computed in time polynomial with respect to $\log N$ and $\log D$.

3. Given vertices v, w, we can decide whether they are adjacent or not in polynomial time with respect to $\log N$.

The first two constructions are based on deep number theoretic theorems and the proofs of their expansion properties are omitted. The third construction is the so called zig-zag construction.

2.3.1 Margulis' construction

These graphs are bipartite. Let $[M]$ be the set of integers between 1 and M. The set of vertices is $V = [M]^2 \cup [M]^2$ and the vertex (x, y) of the left side is connected with the vertices $(x, y), (x, x + y), (x, x + y + 1), (x + y, y)$ and $(x + y + 1, y)$ of the right side, where the arithmetic is done $\mod M$. It can be proven that this is a 5-regular expander graph and the expansion property is proven using Fourier Analysis, [Mar73].

2.3.2 Lubotsky-Phillips-Sarnak construction

Fix two distinct primes p, q such that $q \equiv 1 \pmod 4$ and $p \equiv 1 \pmod q$. The vertices of the graph are the elements of $GF(q) \cup \{\infty\}$ and two vertices z, z' are connected with an edge if

$$z' = \frac{(a_0 + ia_1)z + (a_2 + ia_3)}{(-a_2 + ia_3)z + (a_0 - ia_1},$$

for all $a_0, a_1, a_2, a_3 \in \mathbb{N}$ such that $a_0^2 + a_1^2 + a_2^2 + a_3^2 = p$, with $a_0 > 0$ and a_1, a_2, a_3 and $i^2 = -1 \bmod q$. As the number of integral solutions of $a_0^2 + a_1^2 + a_2^2 + a_3^2 = p$ is $p + 1$, the graph has degree $D = p + 1$ and can be proven that the spectral expansion is at most $\lambda \leq 2\sqrt{d-1}/D$. The proof of the spectral expansion of these graphs is based on deep number theoretic results dealing with the Ramanujan Conjecture, [Lub94].

2.3.3 Zig-zag Construction

This construction, [RVW01, RTV06, RV05], uses a simple combinatorial product. This product is called Zig-zag product and combines in some way two graphs.

Zig-zag product Let G be a (N_1, D_1, λ_1) expander graph. Each vertex has D_1 neighbors and assigns to each vertex a number between 1 and D_1. Now each vertex is going to be replaced by another graph H with D_1 vertices. The i-th vertex of graph H is connected with the i-th neighbor of the replaced vertex of G. Graph H is a (D_1, D_2, λ_2) expander graph. The resulting graph has $N_1 D_1$ vertices, is D_2 regular, each vertex assigns a number between 1 and D_2 to its neighbors and is denoted by $G \diamond H$. However, this is not the graph of the Zig-zag product. None of these edges is going to be used, but they will help us to understand how the vertices of the Zig-zag product are going to be connected. Therefore, from now on we will call the edges, which are going to be removed in the end, pseudoedges. Every vertex in $G \diamond H$ is denoted by (u, i) in a natural way, namely it is the i-th vertex of graph H, which H replaced vertex u of graph G. This vertex is going to have D_2^2 neighbors in the resulting graph and each neighbor is going to be assigned a pair $(a, b) \in D_2 \times D_2$. To find the neighbor (a, b) of vertex (u, i) we proceed as follows: take the a-th pseudoedge of i in H, which connects i, say, with i', this is vertex (u, i') in $G \diamond H$. Vertex (u, i') is connected via a pseudoedge, say, with a vertex (v, j), with $u \neq v$. Now take the b-th pseudoedge of j' in H, which is, say, j, this is vertex (v, j) and is the neighbor (a, b) of vertex

(u, i) in the resulting graph. The resulting graph is denoted by $G \circledS H$. More formally,

Definition 8 (Zig-zag Product)**.** Let G be a D_1 regular graph with N_1 vertices and G_2 a D_2 graph with D_1 vertices. The graph $G \circledS H$ is called Zig-zag product and its vertices are (u, i), with $u \in V(G)$ and $i \in V(H)$. The (a, b)-neighbor of (u, i) is (v, j) and is computed as follows:

1. Let i' be the a-th neighbor of i in H.
2. Let v be the i'-th neighbor of u in G
3. Let j' be such that u is the j'-th neighbor of v in G.
4. Let j be the b-th neighbor of j' in H.

The next theorem shows the expanding property of the Zig-zag product.

Theorem 8. Let G be a $(N_1, D_1, \lambda_1 = 1 - \gamma_1)$ expander graph and H a $(D_1, D_2, \lambda_2 = 1 - \gamma_2)$ expander graph. $G \circledS H$ is a $(N_1 D_1, D_2^2, \lambda = 1 - \gamma_1 \gamma_2^2)$ expander graph.

Proof. We want to construct the random walk matrix M of $G \circledS H$. As we can see, choosing the next step starting from a node (u, i) consists of three steps. Firstly, we choose a random neighbor of i in H, then we go to vertex v of G (this step is compulsory, we do not have the option to choose at random) and, finally, we choose a random neighbor in H from a specified vertex of H. Therefore, we can write the matrix M as a product of three matrices, the two of which are identical. If B is the matrix of the random walk in H then the matrix we are looking is $\tilde{B} = I_{N_1} \otimes B$. That is because from vertex (u, i) we have to stay in u but take a neighbor of i. The matrix which corresponds to the second step has to be a permutation, because here we do not have the ability to choose. From vertex (u, i) we go to vertex (v, j) iff v is the i-th neighbor of u and u is the j-th neighbor of v (in G). Thus $\hat{A}_{(u,i),(v,j)} = 1$ iff the above condition holds else zero. As the third step is the same as the first, we have $M = \tilde{B} \hat{A} \tilde{B}$.

Now we can apply the Matrix Decomposition we used before. Namely, $B = \gamma_2 J + (1 - \gamma_2) E$. Applying the tensoring we have $\tilde{B} = \gamma_2 \tilde{J} + (1 - \gamma_2) \tilde{E}$, with $\tilde{J} = I_{N_1} \otimes J$ and for $\tilde{E}$ analogously. So we have

$$M = (\gamma_2 \tilde{J} + (1 - \gamma_2) \tilde{E}) \hat{A} (\gamma_2 \tilde{J} + (1 - \gamma_2) \tilde{E}),$$

which gives

$$M = \gamma_2^2 \tilde{J} \hat{A} \tilde{J} + (1 - \gamma_2)(\gamma_2 \tilde{J} \hat{A} \tilde{+} \gamma_2 \tilde{E} \hat{A} \tilde{J} + (1 - \gamma_2) \tilde{E} \hat{A} \tilde{E})$$

However, since J is a projection matrix $||xJ|| \leq ||x||$, $\hat{A}$ a permutation $||x\hat{A}|| = ||x||$ and by matrix decomposition $||xE|| \leq ||x||$, we can see that e.g.

$$\frac{||xE\hat{A}J||}{||x||} \leq \frac{||xE\hat{A}J||}{||xE||} \leq \frac{||xE\hat{A}J||}{||xE\hat{A}||} \leq \frac{||xE\hat{A}||}{||xE\hat{A}||} = 1.$$

Therefore, we have

$$||\gamma_2\tilde{J}\hat{A}\tilde{}+\gamma_2\tilde{E}\hat{A}\tilde{J}+(1-\gamma_2)\tilde{E}\hat{A}\tilde{E}|| \leq 1+\gamma_2 \text{ using triangular inequality}$$

For the other term we can see that $\tilde{J}\hat{A}\tilde{J} = A \otimes J$. This is a bit complicated to explain but take for instance the first line of $\tilde{J}$. It has entries only in coordinates corresponding to vertex, say u of G. This line is permuted by $\hat{A}$ and only coordinates which correspond to the neighbors of u have non zero entry now. After multiplying by $\tilde{J}$, we have entries in all coordinates which correspond to neighbors of u and this happens to all the lines of $\tilde{J}$, which correspond to vertex u. As the entries in J are $\frac{1}{D_1}$ we can see the equality of both sides. Thus, we have for a matrix F, with $||F|| \leq 1$ that

$$M = \gamma_2^2 A \otimes J + (1-\gamma_2^2)F.$$

Thus, we can bound the spectral expansion as follows (assuming $x \perp u$)

$$\begin{aligned} \tfrac{||xM||}{||x||} &\leq \tfrac{\gamma_2^2||xA\otimes J||+(1-\gamma_2^2)||xF||}{||x||} \\ &\gamma_2^2(1-\gamma_1)+(1-\gamma_2^2) \\ &1-\gamma_1\gamma_2^2 \end{aligned}$$

□

The expander construction In this construction we are going to use squaring, tensoring and the Zig-zag product we just saw. The construction gives a family of expander graphs $G_1, G_2 \ldots$. Let H be a (D^8, D, λ) expander graph. The family of graphs is:

$$\begin{aligned} G_1 &= H^2 \\ G_2 &= H \otimes H \\ G_t &= (G_{\lceil \frac{t-1}{2} \rceil} \otimes G_{\lfloor \frac{t-1}{2} \rfloor})^2 \circledS H \end{aligned}$$

Proposition 1. For all $t \geq 0$ we have that G_t is a $(D^{8t}, D^2, \lambda + O(\lambda))$.

Proof. It is easy to see that G_t has D^{8t} vertices by induction. The base cases hold and G_t has $D^{8\lceil \frac{t-1}{2} \rceil + 8\lfloor \frac{t-1}{2} \rfloor} D^8 = D^8$. □

2.4 Random Walks and algebraic Properties of Graphs

Let us see the following example. All card games require at least ones that the players shuffle the cards, to put it in another way to take a random ordering of the cards. However, there are 52! different orderings so it is not practical to choose one of them randomly. Therefore, a player takes the deck and shuffles it. We may assume that the shuffling is done as follows: the player takes the first card and changes it position at random. In practice if this is done a few times then the shuffling is almost unpredictably random. Now this procedure can be viewed as follows: each ordering corresponds to a vertex and there is a vertex going from one vertex to another if we can yield the corresponding one ordering from the other in the way described above. What a player does is that he begins from a specific vertex, then he chooses one of his neighbors at random and repeats for a few steps. This procedure is called random walk on a graph and as we can intuitively see it is useful when we want to choose a random element from a very big sample space. We can generalize it to every graph by choosing at the beginning a vertex according to some distribution and then from every vertex choosing one of its neighbors according to some other distribution. Here we will consider undirected regular graphs.

For each graph with n nodes, we define a $n \times n$ matrix M, called the transition matrix, which gives the for every vertex i the probability of choosing vertex j in random walk, that is $M_{ij} =$'the probability of choosing vertex i when being at vertex j'. A very common question is according to what distribution do we have to choose the first vertex such that the random walk on the graph converges to some stationary distribution and to what distribution it converges. When we say that a random walk converges to a distribution π (an n-dimensional vector which assigns a probability to each vertex and sums up to 1) we roughly mean that if we stop the random walk on this graph according to a beginning distribution π_0 and a transition matrix M after an infinite number of steps, then it stops at vertex i with probability given by π. It can be proven that if there exists a stationary distribution π, then it must satisfy $\pi = \pi M$. The intuition is clear, if π is a stationary distribution, then it should not change if we added an additional step. This additional step of the random, starting with distribution π is easy seen to be given by πM.

The adjacency matrix A of a graph G with n nodes, is a $n \times n$ matrix where $A_{ij} = 1$ iff there is an edge connecting vertices i and j and zero otherwise. If G is a d-regular graph then its normalized matrix M is defined to be its adjacency matrix, where each non zero entry is divided by d. It is easy to see that the sum of each column or row equals one. This is because each node has exactly d neighbors, by assumption, and the i'th row has a non zero entry iff the corresponding node is connected with i, namely d non zero entries. A non zero entry has value $1/d$ and as there are d such entries, the sum is 1. The same holds for columns as the graph is undirected and consequently matrix A is symmetric.

Now let us take $u = (\frac{1}{n}, \ldots, \frac{1}{n})$ and multiply it with M. Then the result is $(\frac{d}{dn}, \ldots, \frac{d}{dn}) = (\frac{1}{n}, \ldots, \frac{1}{n}) = u$, which means $\pi = \pi M$. In other words, if the random has stationary distributions then one of them is the uniform distribution. In fact this is the only stationary distribution, if G is connected. Viewing the equality $\pi = \pi M$, we can see that π corresponds to an eigenvector of M with eigenvalue 1.

Lemma 6. The multiplicity of eigenvalue 1 of the normalized matrix M equals the number of connected components.

Proof. If the graph has a connected component with $m < n$ vertices, then we can isolate these vertices and because of the same reasons as before this (sub)graph has an eigenvector $(\frac{1}{m}, \ldots, \frac{1}{m})$ with eigenvalue 1. Thus the n-dimensional vector which has entries $1/m$ in the coordinates which correspond to the vertices of this connected component is another eigenvector with eigenvalue 1, adding 1 to the multiplicity of this eigenvalue.

On the other hand suppose that there is another eigenvector with eigenvalue one. Then take the vertex with the biggest value in the (normalized) eigenvector. We can conclude that all his neighbors (which are d) must have the same value in the eigenvector, because if one of them had a smaller value then their sum divided by d would definitely be smaller than the biggest value. Continuing with the same reasoning we conclude that all the neighbors, of the neighbors, etc, (namely the vertices of the connected component) must have the same value in the eigenvector. Thus, if the graph is connected the eigenvector is the one which corresponds to the uniform distribution, it is not different, contradiction. Continuing this way we can conclude that there are as many eigenvectors as connecting components, in other words the multiplicity of the eigenvalue 1 equals the number of connected components. $\square$

Furthermore, it can be proven that any such random walk on an undirected, connected graph converges to a stationary distribution. As the vector of this stationary distribution must be the eigenvector of eigenvalue 1, we conclude that no matter which distribution we select to choose the first node, the random walk will converge to the uniform distribution.

The next very natural question, which we will consider here, is how fast does a random walk converge to the stationary distribution. How many steps of the random walk do we have to perform in order to end up to an almost uniformly random vertex.

We have to define, what we mean by saying almost uniform. There are many ways of defining the distance between two distributions. Let π be some distribution and $u = (\frac{1}{n}, \ldots, \frac{1}{n})$ the uniform distribution.

Definition 9. Let $v = (v_1, \ldots, v_n)$ be a vector. Then

$$||v||_1 = \sum_{i=1}^{n} |v_i|,$$

$$||v||_2 = ||v|| = \sqrt{\sum_{i=1}^{n} v_i^2},$$

$$||v||_\infty = \max_i |v^i|$$

The following lemma gives a connection between these norms:

Lemma 7.

$$||v||_\infty \leq ||v|| \leq ||v||_1 \leq \sqrt{n}||v||$$

Proof. That $||v||_\infty$ is the smallest norm is clear. The second inequality follows from the fact that $\sum_i v_i^2 \leq (\sum_i v_i)^2$. The last inequality follows from the Cauchy-Schwartz inequality. $\square$

As distance measure we can use any of these norms, but here we will use the l_2-norm, namely the second of the three above norms. It may not be the most natural measure, but it is the most convenient as we shall see. What we want to bound is, if we begin with a probability distribution π and perform k steps of the random walk, how far will be the distribution of last vertex from the uniform distribution. More formally, we want to bound $||\pi M^k - u||$.

Definition 10. The smallest value such that

$$||\pi M^k - u|| \leq \frac{1}{2n}$$

is called the *Mixing Time*

The following quantity gives us a measure of how much does the distance decrease in every step of the random walk, no matter what initial distribution we choose.

Definition 11. For every regular graph G, we define

$$\lambda(G) = \max_{\pi} \frac{||\pi M - u||}{||\pi - u||}$$

The following lemma shows more clearly, why this definition captures the quantity we want.

Lemma 8. For every initial distribution π and every $k \in \mathbb{N}$ we have

$$||\pi M^k - u|| \leq \lambda(G)^k ||\pi - u|| \leq \lambda(G)^k$$

Proof. From the definition of $\lambda(G)$, we have that for every initial distribution π

$$||\pi M - u|| \leq \lambda(G)||\pi - u||.$$

Observe that πM^{k-1} is also a distribution. Therefore,

$$||\pi M^k - u|| = ||\pi M^{k-1} M - u|| =\leq \lambda(G)||\pi M^{k-1} - u|| \leq \lambda(G)^2 ||\pi M^{k-2} - u||,$$

and continuing in the same way

$$||\pi M^k - u|| \leq \lambda(G)^k ||\pi - u||.$$

The last part of the lemma follows from the fact that $||\pi - u|| \leq 1$. This holds because we can show that the biggest value of $||\pi - u||$ is when e.g. $\pi = (\pi_1, \ldots, \pi_n) = (1, \ldots, 0)$. Take all π_i, with $\pi_i > 1/n$ and observe that their sum is the same with the sum of all $\pi_i < 1/n$. Using $\sum_i a_i^2 \leq (\sum_i a_i)^2$ and $\frac{(n-1)^2}{n^2} + \frac{(n-1)}{n^2} < 1$ the lemma (almost) follows. □

The following lemma is also useful, which gives an alternative equivalent definition of $\lambda(G)$.

Lemma 9.

$$\lambda(G) = \max_{x \perp u} \frac{||xM||}{||x||}$$

Proof. Let $\lambda'(G)$ denote the above value. Observe that if π is a distribution vector, then $x = \pi - u$ is orthogonal to u, because the inner product of x and u is the sum of the coordinates of x multiplied by $1/n$ and the sum of the coordinates of x is 0 (it is equal the sum of coordinates of π plus the sum of the coordinates of u, namely 1-1=0). So if we substitute x in the above ratio we have $\frac{||(\pi-u)M||}{||\pi-u||} = \frac{||\pi M - uM||}{||\pi-u||}$, but we can easily check that $uM = u$ and consequently we have for all π $\frac{||\pi M - u||}{||\pi - u||} \leq \lambda'(G)$, which implies

$\lambda(G) \leq \lambda'(G)$. On the other hand for every $x \perp u$, we can find a sufficiently small a such that $\pi = u + ax$ is a probability distribution. Note that the sum of the coordinates of x is zero, therefore the sum of the coordinates of π is 1 for every a. Thus a is the absolute inverse of the biggest absolute value of the coordinates of x (if it is bigger than 1, otherwise take 1) divided by n. Therefore, if we substitute π in the ratio of the previous definition we have $\frac{||\pi M - u||}{||\pi - u||} = \frac{||(u+ax)M - u||}{||(u+ax)-u||} = \frac{||axM + uM - u||}{||ax||}$ and again as $uM = u$ we have $\frac{||axM||}{||ax||} = \frac{||xM||}{||x||}$. Namely, for all $x \perp u$ $\frac{||xM||}{||x||} = \frac{||\pi M - u||}{||\pi - u||} \leq \lambda(G)$ and $\lambda(G) = \lambda'(G)$.

□

Now we want to give an algebraic description of $\lambda(G)$. We will see how is this measure connected to the eigenvalues. The following theorem is known from algebra, which we will not prove here.

Theorem 9. Let M be an $n \times n$ symmetric matrix and $_1, \ldots, _k$ its distinct eigenvalues. Then the subspaces of $\mathbb{R}^n$, $W_i = \{x : x$ is an eigenvector with eigenvalue $_i\}$ are orthogonal and span $\mathbb{R}^n$.

When we say that two spaces X, Y are orthogonal, we mean that if $x \in X$ and $y \in Y$, then $x \perp y$ (equivalently their inner product is 0). Furthermore, W_i span $\mathbb{R}^n$ means $\mathbb{R}^n = W_1 + \ldots + W_k$, namely every point of $\mathbb{R}^n$ can be written as a linear combination of the eigenvectors.

As we discussed before u is an eigenvector of M and $\pi - u$ is orthogonal to u for every probability distribution π. Therefore if $v_2, \ldots, v_n$ are the other eigenvectors of M, $\pi - u$ can be written as a linear combination of these eigenvectors, namely there exist constants $c_2, \ldots, c_n$ such that $\pi = u + c_2 v_2 + \ldots + c_n v_n$. Observe that $v_i M = \lambda_i v_i$, from the definition of the eigenvectors, therefore

$$\pi M^k = u + \lambda_2^k v_2 + \ldots + \lambda_n^k v_n.$$

The next lemma gives the algebraic characterization of $\lambda(G)$ we want:

Lemma 10. Let M be the normalized matrix of a graph G. Then if λ_2 is the second largest absolute eigenvalue of M, we have $\lambda(G) = |\lambda_2|$.

Proof. We will use the second definition of $\lambda(G)$. Let $v_1 = u, v_2, \ldots, v_n$ be the eigenvectors of M with eigenvalues $\lambda_1, \ldots, \lambda_n$ respectively. Then every vector x which is orthogonal to u ($x \perp u$) can be written as a linear combination of the eigenvectors $v_2, \ldots, v_n$. Let $x = c_2 v_2 + \ldots + v_n$, then

$$\begin{aligned}
||xM||^2 &= ||\lambda_2 c_2 v_2 + \ldots + \lambda_n c_n v_n||^2 \\
&= \lambda_2^2 c_2^2 ||v_2||^2 + \ldots + \lambda_n^2 c_n^2 ||v_n||^2 && \text{(Pythagorean Theorem)} \\
&\leq \lambda_2^2 (c_2^2 ||v_2||^2 + \ldots + c_n^2 ||v_n||^2) && \text{by assumption} \\
&= \lambda_2^2 (||c_2 v_2 + \ldots + c_n v_n||^2) && \text{(Pythagorean Theorem)} \\
&= \lambda_2^2 ||x||^2
\end{aligned}$$

Moreover, equality is achieved if $x = v_2$. Therefore, $\lambda(G) = \max_{x \perp u} \frac{||xM||}{||x||} = |\lambda_2|$. □

Using this lemma we can bound the mixing time of a random walk, as we know now, that each step of the random walk decrease the l_2-distance on the vertices to the uniform distance by a factor of at least $\lambda = \lambda_2$. We want $\lambda^k < \frac{1}{2n}$, namely $k = O(\frac{\log n}{\log \frac{1}{\lambda}})$. However, this is roughly $O(\frac{\log n}{1-\lambda})$.

Definition 12. The value $\gamma(G) = 1 - \lambda$ is called Spectral Gap.

As we can see the smaller the second eigenvalue, or equally the bigger the spectral gap, the shorter the mixing time of the random walk. Now we are going to give a bound of λ.

Theorem 10. The spectral gap of a non-bipartite, d-regular graph on n vertices is at least $\frac{1}{dn^2}$.

Proof. Here we will prove it only for the case where all eigenvalues are positive.We have seen in the previous proof that $\lambda = \max_{x \perp u} \frac{||Ax||}{||x||} = \frac{||Av_2||}{||v_2||}$, where v_2 the eigenvector with eigenvalue λ_2.

$$\begin{aligned}
\lambda &= \max_{x \perp u, ||x||=1} ||Ax|| \\
&= \max_{x \perp u, ||x||=1} < Ax, x > \\
&= \max_{x \perp u, ||x||=1} \frac{1}{2d} \sum_{(i,j) \in E} 2 x_i x_j \\
&= \max_{x \perp u, ||x||=1} \frac{1}{2d} \sum_{(i,j) \in E} (x_i^2 + x_j^2 - (x_i - x_j)^2) \\
&= \max_{x \perp u, ||x||=1} \frac{1}{2d} (\sum_{(i,j) \in E} (x_i^2 + x_j^2) - \sum_{(i,j) \in E} (x_i - x_j)^2) \\
&= \max_{x \perp u, ||x||=1} \frac{1}{2d} (2d - \sum_{(i,j) \in E} (x_i - x_j)^2) \quad \text{because} ||x|| = 1
\end{aligned}$$

We know that $||x|| = 1$, which means that there exists a x_i with $x_i^2 \geq \frac{1}{n}$ and as it is orthogonal to u there exists at least one x_j with different sign to x_i.

In other words there exist i, j with $|x_i - x_j| \geq \frac{1}{sqrtn}$. However, there exists a path from vertex i to j of length at most D (the diameter of the graph). Call this path $i_1, i_2, \ldots, i_D = j$ Thus we can take from the triangular inequality

$$\sum_{k=1}^{D-1} |x_{i_k} - x_{i_{k+1}}| \geq |x_i - x_j| \geq \frac{1}{\sqrt{n}}$$

Now we have

$$\begin{aligned} 1 - \lambda \quad &= \tfrac{1}{d} \textstyle\sum_{(i,j) \in E} (x_i - x_j)^2) \\ &\geq \tfrac{1}{d} \textstyle\sum_{k=1}^{D-1} |x_{i_k} - x_{i_{k+1}}|^2 \\ &\geq \tfrac{1}{dD} (\textstyle\sum_{k=1}^{D-1} |x_{i_k} - x_{i_{k+1}}|)^2 \\ &\geq \tfrac{1}{dDn} \end{aligned}$$

As the diameter is at most n the result follows. □

Now we have a more explicit bound on the mixing time. Combining the theorems we have proven we get:

Theorem 11. Let G be a d-regular, non bipartite, connected graph with n vertices. The mixing time of a random walk in it is $O(dn^2 \log n)$.

Lemma 11. A d-regular graph G is bipartite iff -1 is an eigenvalue of M

Proof. Let G be a bipartite graph. Then we can split its vertices into to groups of left and right vertices. Then we can take a vector v, which has entries 1 to the coordinates corresponding to the left vertices and -1 to the other vertices. It is easy to see that if we multiply this vector with M then the resulting vector equals $-v$. This means that -1 is an eigenvalue.

On the other hand let M have an eigenvalue equal -1 and let v be its corresponding eigenvector. Then v must have both positive and negative values. Let us take the smallest negative entry of v. With similar arguments as in a previous proof we can conclude that all the entries of v which correspond to the neighbors of this entry must have the same value but with negative sign. Thus, we can conclude, if we continue in the same way that all vertices, which have the same sign in v do not share any edge and as there are vertices with different signs the graph is bipartite. □

2.4.1 Squaring and Tensoring

We are going to define two operators on graphs and see what spectral properties will the resulting graphs have.

Definition 13. Let G be a d-regular multi graph with n vertices, A its adjacency matrix. The square of G is denoted by G^2 and its adjacency matrix is given by A^2.

Proposition 2. If $\gamma = 1 - \lambda$ is the spectral gap of G then $\gamma' = 1 - \lambda^2$ is the spectral gap of G^2.

The proof of the above proposition is easy, because the second largest eigenvalue of A^2 is λ^2.

Definition 14. Let G_1 be a d_1-regular multi graph with n_1 vertices and G_2 a d_2-regular multi graph with n_2 vertices. Their tensor product, denoted by $G_1 \otimes G_2$, is a $d_1 d_2$-regular graph with vertex set $V_1 \times V_2$, where V_1, V_2 the vertex sets of G_1, G_2 accordingly. The (i_1, i_2)-th neighbor of vertex (x_1, x_2) is (y_1, y_2), where y_1 is the i_1-th neighbor of x_1 and analogously for y_2.

Let U, V be two matrices, then their tensor product is defined to be

$$U \otimes V = \begin{bmatrix} u_{11}V & u_{12}V & \cdots \\ u_{21}V & u_{22}V & \\ \vdots & & \ddots \end{bmatrix}.$$

We can see that if M_1, M_2 are the random walk matrices of G_1, G_2 accordingly then the random walk matrix of $G_1 \otimes G_2$ is $M_1 \otimes M_2$. It can be shown that the eigenvalues of $M_1 \otimes M_2$ are the pairwise products of the eigenvalues of M_1, M_2. Consequently, the largest eigenvalue of $M_1 \otimes M_2$ is 1 and the second largest is $\max\{\lambda_1, \lambda_2\}$, where λ_1, λ_2 are the second largest eigenvalues of M_1, M_2 accordingly.

Chapter 3

Randomness Extractors

Randomness Extractors are functions, which take as input sequences of bits of weak sources, namely sources which produce random but not necessarily independent bits, and output another sequence of almost unbiased and independent bits. The initial motivation was: given such Randomness Extractors simulate randomized algorithms using weak sources.

The first attempt of obtaining unbiased and independent random bits from a source which produces independent but biased bits, with unknown bias, was made by von Neuman in [vN63]. The idea was simple: if the source outputs 1 with probability p, then we can take every two bits and assign 1 if the outcome is 10 (which happens with probability $p(1-p)$) and assign 0 if the outcome is 01 (which happens with the same probability $(1-p)p$). Intuitively we can see that we cannot obtain as many truly random bits as the output of the source. This is the price we pay for trying to convert the bits of a weak source to truly random. In this survey we will see some constructions, which try to optimize the number of truly random bits and compare them with respect to some other parameters.

3.1 Definitions

First of all we need to characterize the weak sources in some way. The first objective is to have a measure, which is general and is also useful for our purpose. Such a measure is of course Shanon Entropy, however it is not very convenient. Therefore another measure is used, called MinEntropy, [CG88], defined below:

Definition 15. Let X be a random variable.

1. **Shannon Entropy:** Its Shannon Entropy is $H(X) = E[\log \frac{1}{Pr(X=x)}]$.

2. **Renyi Entropy:** Its Renyi Entropy is $H_2(X) = \log\left[\frac{1}{E[Pr(X=x)]}\right] = \log\frac{1}{CP(X)}$

3. **MinEntropy:** Its MinEntropy is $H_\infty(X) = \min_x \log\frac{1}{Pr(X=x)}$.

4. **k-Source:** X is a k-Source if $H_\infty(X) \geq k$.

These Entropy measures satisfy

$$\text{For all } X \quad H_\infty(X) \leq H_2(X) \leq H(X)$$

To understand the difference between these measures of entropy note that while Shanon Entropy roughly gives the number of independent bits we can extract on average (having many samples of X), the MinEntropy gives the smallest number of independent bits we can get from any sample. Suppose we have a source X which outputs with probability 0.99 a sequence of n 1's and with probability 0.1 a sequence of n truly random bits (i.e. independent and unbiased). Then $H(X) \geq 0.01n$, while $H_\infty(X) < 1$, that is although in average we can get $0.01n$ bits in most cases we cannot get not even one truly random bit. Therefore, the stronger measure of MinEntropy is used. Here we will focus on general sources, that is sources, for which the only assumption is their MinEntropy and nothing else. The next definition shows which sources we take into consideration:

Definition 16. X is called a k-source if $H_\infty \geq k$.

In other words a k-source is a source for which the most probable outcome comes with probability at most 2^{-k}, namely for all x, $Pr[X = x] \leq 2^{-k}$. The parameter k typically is

1. $k = polylog(n)$
2. $k = n^c$ for an $c \in [0, 1]$
3. $k = \delta n$ for a $\delta \in [0, 1]$
4. $k = n - O(1)$

Two examples of k-sources are

1. $Bit - fixing$ sources. These sources have n bits, where k of them are completely random and independent.
2. $Flat$ k-sources. A uniform distribution on a subset $S \subset \{0, 1\}^n$, with $|S| = 2^k$

The next lemma shows that we only have to consider $flat$ k-sources, [CG88].

Lemma 12. Every k-source X, with $2^k \in \mathbb{N}$, is a convex combination of flat k-sources.

This lemma says that if X is a k-source then we can find random variables X_i, which are flat k-sources and a probability distribution π on these flat sources, such that taking a sample from X is the same as if choosing an X_i according to π and then taking a sample from the chosen X_i.

Proof. Every finite random variable X on N values can be written in the form of an N dimensional vector, where each coordinate corresponds to a value of X and the value of the vector in each coordinates is equal the probability, which corresponds to this value of X. The vectors of all k-sources X have the properties that $X(i) \in [0,1]$, $\sum_i X(i) = 1$ and $X(i) \leq 2^{-k}$. However, these conditions are linear, therefore these vectors form a polytope, namely it is the polytope of the intersection of the hypercube $[0,1]^N$ and the hyperplane $\sum_i X(i) = 1$. This is a convex polytope, therefore any point of this polytope can be written as a convex combination of the vertices of the polytope. The vertices of this polytope are exactly the k-flat sources. □

Before we formally define Randomness Extractors we have to define what their objective is. As we said their purpose is to output a sequence which is almost uniformly random. This means that the statistical difference of the output and a uniformly random variable of the same length is very small.

Definition 17 (Statistical Difference:)**.**

1. Let X, Y be two random Variables taking values in S. Their Statistical Difference is defined to be

$$\Delta(X,Y) = 1/2 \sum_{x \in S} |Pr(X = x) - Pr(Y = x)|$$

or equivalently (can be shown)

$$\Delta(X,Y) = \max_{T \subseteq S} |Pr[X \in T] - Pr[Y \in T]|.$$

2. Let U_n denote the random variable of n uniformly distributed unbiased bits. If $\Delta(X, U_n) \leq \epsilon$ then we say that X is ϵ-close to uniform.

There is one hurdle in constructing Extractors. One can easily show that for every boolean function and for every k there is a k-source (of a random variable with $n > k$ bits) from which it is impossible to extract even a single random bit.

Theorem 12. For every function $F : \{0,1\}^n \to \{0,1\}$, there exists an $(n-1)$-source X such that $F(X) = b$ for every value of X, where $b = 1$ or 0.

Proof. Function F has outcome 0 or 1. One of these values, call it b, appears at least for half of the inputs, namely for at least 2^{n-1} values. Call this set A, then if X is a random variable uniformly distributed on A, then X is an $(n-1)$-source for which $F(X) = b$.

□

As we want to construct an Extractor, which works for every k-source we have to allow to the Extractors to take as an additional input a uniformly distributed variable, called the seed, [NZ96]. Now we are ready to formally define Randomness Extractors.

Definition 18 ((k, ϵ)-Extractor:)**.** The function $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-Extractor if for every k-source X with n bits, the random variable $Ext(X, U_{2^d}$ is ϵ-close to U_{2^m}. If in addition $(U_{2^d}, Ext(X, U_{2^d})$ is ϵ-close to $U_{2^(m+d)}$ then this Extractor is called (k,ϵ)-strong Extractor.

3.2 Existence of Extractors

However, for every k-source, if we choose a random function then this function is a good extractor with high probability. This is shown by the following theorem:

Theorem 13. For every $n, m \in \mathbb{N}$ and every k-source X, if we choose a function $F : \{0,1\}^n \to \{0,1\}^m$ uniformly at random, with $m = k - 2\log\frac{1}{\epsilon} - O(1)$, then $F(X)$ is ϵ-close to U_m, with probability at least $1 - 2^{-(\epsilon^2)}$, with $K = 2^k$.

Proof. We are going to use a Chernoff bound and show that the outcome of a random function cannot differ a lot from the mean value of all functions. What we want to calculate is the probability with which for every subset $S \subseteq [2^m]$, the density of S in $[1, 2^m]$ does not differ more than ϵ from $\frac{|\{x \in X | F(x) \in S\}|}{2^k}$. But the Chernoff bound gives that for each T the above condition holds with probability at least $1 - 2^{-(\epsilon^2)}$. Applying the union bound for all subsets T, which are 2^{2^m}, this condition does not hold for at least one T, is at most $2^{2^m}2^{-(\epsilon^2)}$. However, if $m = k - 2\log\frac{1}{\epsilon} - O(1)$ this probability is less than one. □

The next theorem is a positive result on extractors and shows that if we have a small seed, then there exist Extractors, which work for every k-source.

Theorem 14. For every $n, k \in \mathbb{N}$ and every $\epsilon > 0$, there exists a (k, ϵ)-Extractor $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, with $d = \log(n-k) + 2\log\frac{1}{\epsilon} + O(1)$ and $m = k + d - 2\log\frac{1}{\epsilon} - O(1)$

Proof. Using the same method as before we can prove that for any fixed k-source the probability of failure of a uniformly random function is at most $2^{-(D\epsilon^2)}$, with $K = 2^k$,and $D = 2^d$. This is because if X is a k-source, then (X, U_d) is a $(k+d)$-source and m has the appropriate value. Now we are applying the union bound over all k-sources and show that it is less than one and thus prove the existence of the Extractor.

The k-sources (flat) are $\binom{N}{K}$, with $N = 2^n$ in total and since $\binom{N}{K} \leq (\frac{Ne}{K})^K$, we have that the probability we want is at most $(\frac{Ne}{K})^K 2^{-(D\epsilon^2)}$. But, if $D\epsilon^2 \geq 2\log\frac{Ne}{K} = O(n-k)$, in other words if $d = \log(n-k) + 2\log\frac{1}{\epsilon} + O(1)$, the above probability is less than 1 and the theorem follows. $\square$

In the same way we can prove the existence of strong Extractors, with the same parameters. Now we are going to construct strong extractors using pairwise independent hash functions. The parameters of this extractor are not optimal, but it is the first the step towards constructing good extractors, [BBR88, HILL99, IZ89].

Theorem 15 (Leftover Hash Lemma)**.** Let $\mathfrak{H} = \{h : \{0,1\}^n \mapsto \{0,1\}^m\}$ be a family of pairwise independent hash functions with $m = k - 2\log\frac{1}{\epsilon}$. Then $Ext(x, h) = (h, h(x))$ is a $(k, \frac{\epsilon}{2})$-strong Extractor.

Proof. The proof is broken into three steps. We will see that the output of Ext has low collision probability and use this fact to show that the extractor is close to uniform in the l_2 metric. Then we use a general inequality between l_2 and l_1 to show that the extractor is close to uniform in statistical distance.

Let X be a fixed k-source in $\{0,1\}^n$ and H the random variable of a uniformly random function selected from $\mathfrak{H}$.

Let us now compute the collision probability of $(H, H(X))$. That is we want to compute the probability that if we take two independent trials from H, X, call them H, X and H', X', what is the probability of $(H, H(X)) = (H', H'(X'))$. But this event happens iff $H = H'$ and either $X = X'$ or $H(X) = H'(X')$.

$$CP[(H, H(X))] = CP(H)(CP(X) + Pr[H(X) = H'(X')|X \neq X'])$$

But,

$$CP(H) = Pr[H = H'] = \frac{1}{2^d}$$

$$CP(X) = Pr[X = X'] \leq \frac{1}{2^k} \text{ because } H_\infty \geq k$$

and

$Pr[H(X) = H'(X')|X \neq X'] = \dfrac{1}{2^m}$ because of the pairwise independence.

Therefore,

$$CP[(H, H(X))] \leq \frac{1}{2^d}(\frac{1}{2^k} + \frac{1}{2^m}).$$

Since $m = k - 2\log\frac{1}{\epsilon}$, we have that $2^m = \epsilon^2 2^k$ and substituting above

$$CP[(H, H(X))] \leq \frac{1+\epsilon^2}{2^{k+d}}.$$

We proceed to the next step and calculate the distance in l_2 metric of $(H, H(X))$ and U_{d+m}. That is

$$||(H, H(X)) - U_{d+m}||^2.$$

But this is equal to $CP[(H, H(X))] - CP[U_{d+m}]$, because

$$||(H, H(X)) - U_{d+m}||^2 =$$

$$\sum_{(a,b)} Pr[(H, H(X)) = (a, b)]^2 + \sum_x 2^{-2(d+m)} - 2 \cdot 2^{d+m} \sum_{(a,b)} Pr[(H, H(X)) = (a, b)].$$

Since, $\sum_{(a,b)} Pr[(H, H(X)) = (a, b)] = 1$ and $\sum_x 2^{-2(d+m)} = 2^{-d-m}$, the claim follows. Therefore,

$$||(H, H(X)) - U_{d+m}||^2 \leq \frac{1+\epsilon^2}{2^{k+d}} - \frac{1}{2^{k+d}} = \frac{\epsilon^2}{2^{k+d}}.$$

For the third and final step we use the first definition of statistical distance

$$\Delta((H, H(X)), U_{d+m}) = 1/2|(H, H(X)) - U_{d+m}|_1$$

and the Cauchy-Schwartz inequality according to which

$$|(H, H(X)) - U_{d+m}|_1 \leq \sqrt{2^{m+d}}||(H, H(X)) - U_{d+m}||.$$

Substituting we have that

$$\Delta((H, H(X)), U_{d+m}) \leq \frac{\sqrt{2^{m+d}}}{2}\frac{\epsilon}{2^{m+d}} = \frac{\epsilon}{2}.$$

This means that $Ext(x, h) = (h, h(x))$ is a $(k, \frac{\epsilon}{2})$-strong Extractor. $\square$

The use of the Leftover Hash Lemma for extracting has a very big disadvantage. As we can see, in order to choose a random function from the family of the pairwise independent hash functions, requires $O(n)$ random bits. This means that the seed has length $O(n)$, which is very big compared to the optimal. An extractor with optimal seed length requires $O(\log n)$ random bits. However, the output length is $n + k - 2\log\frac{1}{\epsilon}$, which is optimal, namely we extract all the MinEntropy of the source. This is the reason, why the Leftover Hash Lemma is so important and therefore it is used in many constructions.

3.3 Extractor Construction

The construction we will see here is due to [VV85, CG88, Zuc96, NZ96]. Before we proceed to the construction of an Extractor, let us see a generalization of k-sources, called Block Sources. These are sources, which can be split into blocks of k-sources.

Definition 19 (Block Sources).

1. A random variable $X = (X_1, \dots, X_t)$ is called a $(k_1, \dots k_t)$-block source, if for every i and every $x_1, \dots, x_i$ $X_i|_{X_1=x_1,\dots,X_i=x_i}$ is a k_i source.
2. If $k_1 = \dots = k_t = k$, then X is called a $t \times k$-block source.

It is easy to see that a $(k_1, \dots k_t)$-block source is also a $(k_1 + \dots + k_t)$-source, by simply adding the MinEntropy of every independent block. The next lemma shows why block sources are convenient for Extractors. What we do in the case of a two block source is we use an Extractor for the first block and then we use the output of the extractor as seed for another extractor which takes as input the second block.

Lemma 13. Let

1. $Ext_i : \{0,1\}^{n_i} \times \{0,1\}^{d_i} \to \{0,1\}^{m_i}$ be an (k_i, ϵ_i)-extractor, for $i = 1, 2$, with $m_1 \geq d_2$,
2. $Ext'((x_1, x_2), y_1) = (Ext_2(x_2, y), z)$, with $(y, z) = Ext_1(x_1, y_1)$,
3. $X = (X_1, X_2)$ be a (k_1, k_2)-block source of length $\{0,1\}^{n_1} \times \{0,1\}^{n_2}$,

then $Ext'(X, U_{d_1})$ is $(\epsilon_1 + \epsilon_2)$-close to $U_{m_1+m_2-d_2}$.

Proof. We will use the triangle inequality. By assumption we know that $(Y, Z, X_2) = (Ext_1(X_1, U_{d_1}), X_2)$ is ϵ_1-close to $(U_{m_1-d_2}, U_{d_2}, X_2)$. This means that $(Ext_2(X_2, Y), Z)$ is also ϵ_1-close to $(Ext_2(X_2, U_{d_2}), U_{m_1-d_2})$. But by assumption again the latter, $(Ext_2(X_2, U_{d_2}), U_{m_1-d_2})$, is ϵ_2-close to $(U_{m_2}, U_{m_1-d_2})$. Since $|A - C| \leq |A - B| + |B - C|$ we have that $(Ext_2(X_2, Y), Z)$ is $\epsilon_1 + \epsilon_2$ close to $U_{m_1+m_2-d_2}$. $\square$

This lemma can obviously be extended to block sources of more blocks in the obvious way. The importance of the above lemma is that if we are interested in extracting Randomness from block sources we can use seed only for the first block and then use the output of the first extracting for the second extractor (which takes as input the second block) and continue in the same way.

The following lemma shows us, how we can see a k-source as a block source. Let $l < k$, then if we take the first l bits of a k-source, we can see that these bits have MinEntropy at most l and the rest bits must have the remaining $k - l$ MinEntropy. The lemma shows that this happens with high probability for a random prefix of fixed number of bits.

Lemma 14. Let (W, X) be a jointly distributed random variable of length n and MinEntropy k. Let W have length at most l, then with probability at least $1 - \epsilon$ over a random w, $X|_{W=w}$ is a $(k - l - \log \frac{1}{\epsilon})$-source, for every $\epsilon > 0$.

As a consequence we have:

Corollary 1. Let X be a random variable of length n and MinEntropy $n-\Delta$. Then, if $X = (X_1, X_2)$, with X_i with length n_i, for $i = 1, 2$, X is ϵ-close to a $(n_1 - \Delta, n_2 - \Delta - \log \frac{1}{\epsilon})$-source.

The next lemma shows that it is useful applying the same extractor, with different seed, to the same output of a k-source. The motivation is that if with one application we extract a fraction of the MinEntropy of the source, then with the second application we extract some fraction of the remaining MinEntropy of the source. This lemma will be used several times in our construction.

Lemma 15. Let $Ext_1 : \{0,1\}^n \times \{0,1\}^{d_1} \mapsto \{0,1\}^{m_1}$ be a (k_1, ϵ_1) strong extractor and $Ext_2 : \{0,1\}^n \times \{0,1\}^{d_2} \mapsto \{0,1\}^{m_2}$ be a (k_2, ϵ_2) strong extractor. Define $Ext : \{0,1\}^n \times \{0,1\}^{d_1+d_2} \mapsto \{0,1\}^{m_1+m_2}$ by

$$Ext(x, (y_1, y_2)) = Ext_1(x, y_1)||Ext_2(x, y_2).$$

Then Ext is a $(k, \epsilon_1 + \epsilon_2)$ strong Extractor.

Before we proceed to the Construction of the Extractor we are going to define the most useful tool, the Condenser. Informally speaking, a Condenser is a function that takes as input a source of low MinEntropy compared to its length and outputs a random variable with higher rate MinEntropy over length. We are not going to prove the existence of such functions now, but this follows from the next Chapter by using Error Correcting Codes.

Definition 20 (Condenser, [RSW06])**.** A (k, k', ϵ)-condenser is a function $Con : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$, such that for every distribution X of length n with $H_\infty(X) \geq k$, the distribution $Con(X, U_d)$ is ϵ -close to a distribution X' with $H_\infty(X') \geq k'$.

Note that if $m = k'$, then the Condenser is actually an Extractor. The next theorem shows the existence of good Condensers, but as we said will not be proven now. However, it plays major role in the construction we are going to see.

Theorem 16. For every $a > 0$, $n \geq k$ and $\epsilon > 0$ there exists an explicit $(k, k+d, \epsilon)$ Condenser $Con : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$, with $d = O(\log n + \log \frac{1}{\epsilon})$ and $m = (1+a)k + O(\log \frac{n}{\epsilon})$.

Using this theorem we can easily construct an Extractor, which extract half of the MinEntropy of the source and uses a seed with length an arbitrary large constant factor smaller than the output length. That is

Lemma 16. For every constant t and integers $n \geq k$, there is a (k, ϵ)-extractor $Ext : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$, with $d = \frac{k}{t} + O(\log \frac{n}{\epsilon})$.

Proof. □

Now, we have all the tools we need to construct an explicit extractor which uses a seed of logarithmic length and extracts any fraction of the MinEntropy of the source.

For any n, ϵ we will see by induction on i how to construct an (k, ϵ_i)-extractor for sources with MinEntropy $k \leq 2^i 8d$. Our extractor $Ext : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$ will have $d = c \log \frac{n}{\epsilon_0}$ and $\epsilon_0 = \frac{\epsilon}{n^c}$.

- For $i = 0$, namely $k \leq 8d$, we can use the previous lemma. If we set $t = 9$, then we can have an extractor with seed $d = O(\log \frac{n}{\epsilon_0})$, which extracts half of the MinEntropy.

- Suppose that we can construct such an extractor for every $i < j$. We are going to prove that we can also construct such an extractor for $i = j$, namely for sources with MinEntropy $k \leq 2^i 8d$. We apply the following recursive procedure:

1. We use the Condenser of the previous theorem to convert the k-source X to a source that is ϵ_0-close to a k source of length $(1+a)k+O(\log\frac{n}{\epsilon_0})$ for sufficiently small a, which will be determines later.
2. The output of the Condenser is split into two equally long halves $X' = (X_1, X_2)$. As we have seen X' is $2\epsilon_0$-close to a $2 \times k'$ block source, with $k' = k - \frac{1}{2}((1+a)k + O(\log\frac{n}{\epsilon_0})) = \frac{k}{2} - \frac{ak}{2} - O(\log\frac{n}{\epsilon_0})$.
3. The selection of a, c must be such that $k' \geq 2d$, since we can assume that $k > 8d$.
4. As $k' > 2d$ we can extract from the first block d bits using the previous lemma, with $t = 16$.
5. Using the induction hypothesis we can use these d bits to extract $\frac{k'}{2}$ bits from the second block.

To put everything together we have the following theorem:

Theorem 17. For every constant $a > 0$, all integers $n \geq k$ and every $\epsilon > 0$, we can explicitly construct a (k, ϵ) Extractor $Ext : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$, with $d = O(\log\frac{n}{\epsilon})$ and $m = (1-a)k$

Moreover if we apply the above extractor several times with different seed we can extract all the MinEntropy, that is $k + d - O(\log\frac{1}{\epsilon})$. The total times of extracting will be $O(\log k)$ and thus we have:

Theorem 18. For all integers $n \geq k$ and every $\epsilon > 0$, we can explicitly construct a (k, ϵ) Extractor $Ext : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$, with $d = O(\log k \log\frac{n}{\epsilon})$ and $m = k + d - O(\log\frac{1}{\epsilon})$

Chapter 4

Error Correcting Codes

The purpose of Error Correcting codes, [Sha48, Ham50] is the secure transmission of a message over a noisy channel. There is a sender who wants to send a message to a receiver, but the problem is that the channel they want to use for their communication has a non zero probability of changing the message. What the sender can do is to add some redundancy to the message, so that the receiver can use this redundancy to recover the original message.

For instance suppose that there is a binary channel, which flips the bit sent by the sender with a probability at most p (say $\frac{1}{8}$, surely less than $\frac{1}{2}$). Then suppose that the sender wants to send a bit. A very easy solution would be that the sender sends the same message several times and the receiver computes the majority of the received bits as the original message. Applying a very simple Chernoff bound, we can see that if the sender sends the bit n times then with a high probability $(1 - e^{-O(n)})$ the receiver computes the correct message.

More generally the sender uses a function $C : \Sigma^k \mapsto \Sigma^n$ with $n \geq k$ to encode a message $x \in \Sigma^k$. The receiver receives a y, which is possibly different than $C(x)$, because of the noisy channel and uses the decoding function D to compute x.

Definition 21 (Hamming Distance)**.** The Hamming Distance of two strings a, b denoted by $H(a, b)$ is the number of different entries of these strings.

For example $H(10100, 00110) = 2$, because these two strings differ in the first and the fourth entry. Now let us see the definition of a code.

Definition 22. An $[n, k, d]_q$ code is a function $C : \Sigma^k \mapsto \Sigma^n$ such that

1. Σ is the alphabet of the messages, with $|\Sigma| = q$.

2. k the *Block Length is the length of the messages to be encoded.*

3. *n the Information Length is the length of the encoded messages, i.e. the codewords.*

4. *For every x, x' it holds that $H(x, x') \geq d$.*

If we make the assumption that the noisy channel corrupts the message at at most e entries, then the code admits an error correcting function if and only if the errors are not too many, that is iff for every encoded message with e corrupted entries there does not exist a different encoded message with Hamming Distance less or equal than e. Thus we have the following proposition:

Proposition 3. An $[n, k, d]_q$ code is an error correcting code if and only if $d \geq 2e + 1$.

Of course we want to have explicit Error Correcting Codes, that is Codes, where the encoding and decoding are made in polynomial time. However, we are going to talk about such an Error correcting Code later. When constructing Error Correcting Codes we want to optimize the following parameters:

1. Maximize the number of errors the Code is able to correct

2. Minimize the rate of the Information Length over the Block length.

3. Minimize the alphabet size $|\Sigma| = q$

4. Minimize the running time of Encoding and Decoding.

The next lemma, called Singleton Bound, shows an impossibility result regarding k, d and n.

Lemma 17 (Singleton Bound)**.** For every code C, $[n, k, d]_q$, we have that

$$k \leq n - d + 1$$

Proof. Suppose we project the codewords to the first $n - d + 1$ coordinates. That is we take $C' : \Sigma^k \mapsto \Sigma^{n-d+1}$ the projection of $C : \Sigma^k \mapsto \Sigma^n$. Suppose that C' is not injective, namely there exist x, x', such that $C'(x) = C'(x')$. Then $C(x)$ and $C(x')$ can differ at most at $d-1$ coordinates, a contradiction. Therefore, C' must be an injection and thus $k \leq n - d + 1$. □

The next lemma, which uses the probabilistic method, shows that there exists very good binary codes, [Gil52].

Lemma 18 (Gilbert Bound)**.** For every $d \leq \frac{n}{2}$ there exists an $[n, k, d]_2$ code such that

$$k \geq n(1 - H_2(\frac{d}{n})) - (\log n)$$

Proof. To prove this lemma we are going to see that we can greedily construct an $[n, k, d]_2$ code with the above property. Let $x_1, \dots, x_{2^k}$ be an enumeration of all the the words, which we want to encode. Let $C(x_1)$ be the string with n zeros and then $C(x_i)$ is chosen to be the next string, according to the lexicographic order, such that it has distance at least d from all codewords $C(x_1), \dots, C(x_{i-1})$. It remains to show that we can always choose such a codeword.

Suppose that $c_1, \dots, c_i$ are random codewords. Let c be another codeword chosen uniformly at random. Then the probability that this codeword has Hamming distance less than d from c_j is

$$Pr[H(c, c_j) < d] = \frac{1}{2^n} \sum_{i=0}^{d-1} \binom{n}{i}.$$

However, using the Strirling's formula we can see that

$$\binom{n}{k} = (\sqrt{\frac{n}{k(n-k)}}(\frac{n}{k})^k(\frac{n}{n-k})^{n-k})$$

and if we take the logarithm

$$\log \binom{n}{k} = nH(\frac{k}{n}) + (\log n),$$

where the function H is the binary entropy. Now, if we apply this to the above probability we have

$$Pr[H(c, c_j) < d] \leq \frac{1}{2^n} 2^{nH(\frac{k}{n}) + (\log n)},$$

because $d \leq \frac{n}{2}$. Applying the union bound over all c_j, we have

$$Pr[\exists j : H(c, c_j) < d] \leq \frac{1}{2^n} 2^{k + nH(\frac{k}{n}) + (\log n)}$$

which has to be smaller than 1 and the lemma follows. □

4.1 Reed-Solomon Codes

Now we are going to see a very popular construction of Error Correcting Codes called Reed-Solomon codes, [RS60]. In this construction we are going to use some properties of univariate polynomials over a finite field.

Given a prime number q and $n \leq q$ and $k \leq n$, the Reed-Solomon Code $RS_{q,n,k}$ is constructed as follows:

1. Generate the field $\mathbb{F}_q$.
2. Choose n distinct elements $(a_1, \ldots, a_n)$ of $\mathbb{F}_q$.
3. The representation of the message to be encoded is a sequence of k elements $(a_1, \ldots, a_n)$ of $\mathbb{F}_q$.
4. Define the polynomial
$$C(x) = \sum_{i=0}^{k-1} c_j x^j.$$
5. The encoding of the message is the evaluation of $C(x)$ at the n elements of $\mathbb{F}_q$,
$$< C(a_1), \ldots, C(a_n) >$$

The reason why $n \leq q$ must hold is that the elements chosen in the second step must be all distinct. The properties of the Reed-Solomon Codes are based on the following algebraic fact:

Theorem 19. Let p_1, p_2 be two polynomials of $\mathbb{F}_q[X]$ of degree at most $k-1$. Then there exist at most $k-1$ distinct elements a such that $p_1(a) = p_2(a)$.

We are not going to prove this fact, as it needs many other facts of Algebra, however the same holds for polynomials over the reals, so this theorem goes along our intuition.

Theorem 20. The Reed-Solomon code $RS_{q,n,k}$ is an $[n, k, n-k+1]_q$ code.

Proof. The first two parameters, Block length and information length, are easily seen to be true because of the construction described above. For the third parameter we can see that it is exact. By the Singleton bound we have that $d \geq n-k+1$. On the other hand suppose that we have two polynomials p_1, p_2 of degree at most $k-1$. Then in a previous theorem we saw that these polynomials agree on at most $k-1$ points of $\mathbb{F}_q$ and thus disagree on at least $n-k+1$ points of a set of n points. It follows that the Reed-Solomon code has distance $n-k+1$. □

4.2 The Berlekamp-Welch Algorithm

Now we are going to see how we can run an Error Correcting algorithm for the Reed-Solomon Algorithm, which is efficient, namely it runs in polynomial time, with respect to n, [Pet60, Ber68]. This algorithm is called the Berlekamp-Welch Algorithm, but is not the fastest known.

Our assumption is that there is a channel which corrupts at most $e < \frac{n-k+1}{2}$ symbols of the transmitted message. The block length is k, the message length n and the elements of $\mathbb{F}_q$ used for the polynomial evaluation are $(a_1, \ldots, a_n)$.

- **Input:** n elements $(y_1, \ldots, y_n)$ of $\mathbb{F}_q$
- **Output:** A polynomial p of degree at most $k-1$ such that for at most e distinct i $p(a_i) \neq y_i$.

1. If there is a polynomial p such that $p(a_i) = y_i$ then output p

   else

2. Find polynomials $E(x), N(x)$ such that

   (a) $E(x) \neq 0$

   (b) $E(x)$ has degree at most e and $N(x)$ at most $e + k - 1$

   (c) $N(a_i) = E(a_i)y_i$ for every $i \leq n$.

   (d) Output $\frac{N(x)}{E(x)}$.

Let I denote the set of the indices of the "bad" y_i, namely $I = \{i|p(x_i) \neq y_i\}$. If $I = \emptyset$ then p can be found efficiently by interpolating. If $I \neq \emptyset$ then we will see that we can compute p in time $O(n^3)$.

If we write $E(x) = \sum_{i=0}^{e} b_i x^i$ and $N(x) = \sum_{i=0}^{e+k-1} c_i x^i$ then taking the set of equations $N(a_i) = E(a_i)y_i$ for every $i \leq n$ and supposing that there exists a solution, then this solution can be found using Gauss elimination and in time $O(n^3)$.

Let $E(x) = \prod_{i \in I}(x - a_i)$ and $N(x) = E(x)p(x)$. We observe that $E(x)$ has degree at most e and $N(x)$ at most $e + k - 1$ as required. We can see that the condition $N(a_i) = E(a_i)y_i$ is satisfied for every $i \leq n$, because

1. if $i \in I$ then $E(a_i) = 0$ and $N(a_i) = E(a_i)p(a_i) = 0$ and
2. if $i \neq I$ then $p(a_i) = y_i$ and thus $N(a_i) = E(a_i)p(a_i) = E(a_i)y_i$.

So we can see that there exists a solution. But we have to prove that if we find a solution then it is the same solution, which means that if $\frac{N'(x)}{E'(x)}$ is the solution we found, then $\frac{N'(x)}{E'(x)} = \frac{N(x)}{E(x)}$, i.e. $N'(x)E(x) = N(x)E'(x)$. However, for all $i \leq n$ we have

$$N'(a_i)E(a_i) = y_iE(a_i)E'(a_i) = E(a_i)N'(a_i)$$

and since both $N'(x)E(x)$ and $N(x)E'(x)$ have degree at most $2e+k-1$ and by assumption $e < \frac{n-k+1}{2}$ we it suffices to check equality at the n points $a_1, \ldots, a_n$. Thus the algorithm is correct.

4.3 List Decoding

The aspect of Error Correcting Codes, which is interesting for Pseudorandomness is List Decoding. The idea is that we would like to be able to decode in case more corruptions take place during the transmission of the message, [Eli57, Woz58]. As we have seen we are able to deterministically decode if the number of corruptions are strictly less than half of the minimum of the minimum distance between any two codewords. But what if the number of corruptions are more than this threshold. What we can do is to output a small list which includes all probable codewords. We also would like this algorithm (which outputs the list) to run in polynomial time of course. The problem is the following:

List Decoding

1. **Input:** A received word c and an error bound e.
2. **Output:** A list of all codewords $c_1, \ldots, c_m$, whose hamming distance from c is at most e.

Definition 23. An $[n,k,d]_q$ code is called (e,L)-error correcting if up to e errors can be corrected with a list of size at most L.

Definition 24. Let $LIST(r,\epsilon) = \{c | c \text{ agrees with } r \text{ at a fraction of at least } \epsilon \text{ places }\}$.

The next proposition is the analogue of the proposition for the error Correcting codes.

Proposition 4. An $[n,k,d]_q$ code is an (e,L)-error correcting code iff for every codeword c, $|LIST(c, 1-\frac{e}{n})| \leq L$.

The following bound shows when we are sure that an $[n,k,d]_q$ Code is List decodeable with a list of polynomial length, specifically linear, list size with respect to the Information Length n.

Theorem 21 (Johnson Bound:)**.** An $[n, k, d]_q$ code is an (e, L)-error correcting code provided that

$$\frac{e}{n} \leq \frac{q-1}{q}(1 - \sqrt{1 - \frac{q}{q-1}\frac{d}{n}}).$$

If $q = 2$, $\epsilon = \frac{e}{n}, \delta = \frac{d}{n}$ then we have

$$\epsilon \leq \frac{1}{2}(1 - \sqrt{1 - 2\delta})$$

Proof. We are going to see the proof only for the case $q = 2$, i.e. of binary codes. $\square$

4.3.1 List Decoding of Reed-Solomon Codes

Now we will see how can we list decode the Reed-Solomon Code in polynomial time. The following theorem is proven by Sudan, [Sud97, GS99].

Theorem 22. The Reed-Solomon code $RS_{q,n,k}$ is an $(e, \sqrt{\frac{n}{k}})$-error correcting code with a polynomial time decoding algorithm if $t > 2\sqrt{kn}$, where $t = n - e$.

Proof.

- **Input:** n pair elements $(a_1, y_1), \dots, (a_n, y_n)$ of $\mathbb{F}_q^2$
- **Output:** A list of all polynomials p of degree at most $k - 1$, such that $p(a_i) = y_i$ for at least t values of i.

The algorithm consists of the following two steps:

1. Find a bivariate polynomial $Q(x, y)$ such that
   - Q has degree at most $d_x - 1$ in x and $d_y - 1$ in y.
   - $Q(a_i, y_i) = 0$ for all $i \leq n$.
   - Q is not the zero polynomial.
2. Factor Q. For every factor of the form $(y - p(x))$ output p.

Note that the polynomial Q can be written as

$$Q(x, y) = \sum_{i=0}^{d_x-1} \sum_{j=0} d_y - 1 c_{ij} x^i y^j,$$

which means that Q is defined by its $d_x d_y$ coefficients. Thus if y_i are more than $d_x d_y$, we can always find a non zero solution using Gauss Elimination.

This holds because each y_i imposes a linear constraint on the coefficients of Q.

Now, we are going to prove that for every polynomial p with degree at most $k-1$, which satisfies that for at least t distinct a_i, it holds $p(a_i) = y_i$, then $Q(x, p(x)) = 0$. However, $Q(x, p(x))$ is a univariate polynomial with degree at most $(d_x - 1) + (k-1)(d_y - 1)$. Moreover, for each a_i, with $p(a_i) = y_i$, we have that $Q(a_i, p(a_i)) = Q(a_i, y_i) = 0$ by the construction of Q, and there are at least t such a_i by assumption. So, provided that t is greater than the degree of Q we can argue that $Q(x, p(x)) = 0$, i.e. it must hold that $t > (d_x - 1) + (k-1)(d_y - 1)$. To conclude, we know that if is a root of a polynomial in x, then $(x -)$ divides this polynomial. Therefore, if we view Q as a polynomial in y then $(y - p(x))$ must divide Q.

The parameters we have seen that must hold are the following:

1. $d_x d_y > n$
2. $t > d_x + k d_y$

Our parameters are optimized, while satisfying the above conditions if we set $d_x = \sqrt{kn}$ and $d_y = \sqrt{\frac{n}{k}}$. Then $t = 2\sqrt{kn}$ and there are at most $\sqrt{\frac{n}{k}}$ polynomials p, which are in the list. The theorem follows. □

4.3.2 Parvaresh-Vardy Codes

Now we are going to see an improvement of the previous algorithm proposed by Parvaresh and Vardy, [PV05]. The main idea is to generalize the previous algorithm in some way. That is we let the polynomial Q be a multivariate polynomial, not just bivariate. For example, let $Q(x, z_1, \ldots, z_m)$ be an $(m+1)$-variate polynomial such that $Q(a_i, y_i, \ldots, y_i) = 0$ for all $i \leq n$. We can observe that before the degree of the bivariate Q should be roughly $\sqrt{q}$ in each variable, but if there are more variables the degree of each variable must be roughly $q^{\frac{1}{m}}$.

The actual idea of Parvaresh and Vardy is to let Q be an $(m+1)$-variate polynomial $Q(x, z_1, \ldots, z_m)$ of degree d_x in x and $d_z = h - 1$ in each z_i, such that $Q(a_i, y_i^h, y_i^{h^2}, \ldots, y_i^{h^{m-1}}) = 0$ for all $i \leq n$.

The Code is a little changed from the Reed-Solomon codes. The messages are still viewed as polynomials of degree $k-1$ over $\mathbb{F}_q$. The two differences are:

1. $\Sigma = \mathbb{F}_q^m$

2. Let f be the message to be encoded, then its encoding is for every a_i

$$[f_0(a_i), f_1(a_i), \dots, f_{m-1}(a_i)],$$

with $f_i(x) = f_i(x)^{h^i} \mod E(x)$, where $E(x)$ is a fixed irreducible polynomial of degree k over $\mathbb{F}_q$.

Let r be the received message. The list decoding algorithm consists of the following two steps:

1. Find an $(m+1)$-variate polynomial $Q(x, z_0, \dots, z_{m-1})$ such that
   - Q has degree at most d_x in x and $h-1$ in each z_i.
   - $Q(a_i, r|_i) = 0$ for all $i \leq n$. ($r|_i$ is the 'projection' of r in the entry corresponding to a_i, which is an (m)-dimensional vector)
   - Q is not the zero polynomial.
2. Factor $Q^*(z) = Q(x, z, z^h, \dots, z^{h^{m-1}}) \mod E(x)$. For every factor of the form $(z - p(x))$ output p.

Theorem 23. For appropriate parameters of h, m the Parvaresh-Vardy list decoding algorithm that list-decodes up to distance $\delta = 1 - 2\sqrt{\frac{k}{n}}$.

Proof. 1. In order to be able to find such a polynomial the number of the coefficients of the variables must be greater than the constraints. Namely, reasoning in the same way as before it must hold that

$$d_x h^m > n.$$

Moreover we may assume that Q is not divisible by $E(x)$ since we can divide out all factors of E. Note that this will not affect our conditions since E is irreducible and thus has no roots.

2. Reasoning in the same way as before we can see that each polynomial f, which belongs to the list is a solution of Q, namely $Q(x, f_0(x), \dots, f_{m-1}(x)) = 0$. However this is true provided that

$$t > d_x + (h-1)(k-1)m.$$

Now if we take both sides modulo $E(x)$ we have that

$$Q(x, f^{h^0}(x), \dots, f^{h^{m-1}}(x)) \mod E(x) = 0.$$

Thus we can define the polynomial

$$Q^*(x, z, z^h, \dots, z^{m-1}) \mod E(x)$$

and argue that each f in the list is a root of Q^* over the field $\mathbb{F}_q[Y]/E(Y)$. By factoring Q^* we can take all the polynomials in the list.

The two conditions which must be satisfied are

$$d_x h^m > n$$

and

$$t > d_x + (h-1)(k-1)m.$$

If we set

$$d_x = t - khm$$

and

$$\epsilon > \frac{1}{h^m} + \frac{khm}{q}$$

then both conditions can be satisfied. If we set $h = 2$ and $m = O(\log \frac{1}{\epsilon})$ the theorem follows. □

4.4 Unified View

The next proposition shows, what we mentioned in the introduction, that both Expander Graphs and Randomness Extractors can be expressed in the language of List Decodable Error Correcting Codes.

Proposition 5. 1. Let K be a positive integer, then $\Gamma : [N] \times [D] \mapsto [M]$ is a (K, A) vertex $N \times M$ bipartite expander if and only if for every set $T \subset [D] \times [M]$ of size at most KA it holds that $|LIST_\Gamma(T, 1)| < K$.

2. Let $\Gamma = Ext : [N] \times [D] \mapsto [M]$, $K = 2^k$ and $0 \leq \epsilon \leq 1$.

   (a) Let Ext be a (k, ϵ) extractor. Then for every $f : [M] \mapsto [0, 1]$ it holds $|List_\Gamma(f, \mu(f) + \epsilon)| < K$.

   (b) If for every $T \subset [M]$ it holds that $|List_\Gamma(T, \mu(T) + \epsilon)| \leq K$, then Ext is a $(k + log(1/\epsilon), 2\epsilon)$ extractor.

Proof. 1. Suppose that Γ is not a (K, A) expander. This happens iff there exists set $S \subset [N]$ of size at least K and $|N(S)| < KA$. The latter happens iff $T \subset [M]$ with $|LIST(T, 1)| \geq K$ and $|T| < KA$.

2. (a) Suppose that $|List_\Gamma(f, \mu(f) + \epsilon)| \geq K$ and let X be the uniform distribution over $List_\Gamma(f, \mu(f) + \epsilon)$. Then X is a k source and $E[f(Ext(X, U_{[D]}))] > \mu(f) + \epsilon = E[f(U_{[M]})] + \epsilon$. This contradicts that Ext is a (k, ϵ) Extractor.

   (b) Let X be a $(k+log(1/\epsilon))$-source and $T \subset [M]$. Then $Pr[Ext(X, U_{[D]}) \in T] \leq Pr[X \in LIST(T, \mu(T) + \epsilon)] + Pr[Ext(X, U_{[D]}) \in T | X \notin LIST(T, \mu(T) + \epsilon)]$. However, the latter can be upper bounded by $|LIST(T, \mu(T) + \epsilon)| 2^{-k-log(1/\epsilon)} + \mu(T) + \epsilon = \mu(T) + 2\epsilon$.

$\square$

Bibliography

[AC88] Noga Alon and Fan R. K. Chung. Explicit construction of linear sized tolerant networks. In *Discrete Mathematics*, pages 15–19, 1988.

[AM84] Noga Alon and V. D. Milman. Eigenvalues, expanders and superconcentrators (extended abstract). In *FOCS*, pages 320–322, 1984.

[Bas81] L.A. Bassalygo. Asymptotically optimal switching circuits. In *Problems of Information Transmission,*, 1981.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. In *SIAM J. Comput.*, pages 210–229, 1988.

[Ber68] E. R. Berlekamp. Algebraic coding theory. In *McGraw-Hill Book Co., New York*, 1968.

[BL06] Yonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap*. In *Combinatorica*, pages 495–519, 2006.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *SIAM J. Comput.*, pages 230–261, 1988.

[Eli57] P. Elias. List decoding for noisy channels. In *Massachusetts Institute of Technology, Cambridge, Mass.*, 1957.

[Gil52] E. Gilbert. A comparison of signaling alphabets. In *Bell Systems Technical Journal,*, 1952.

[Gil98] David Gillman. A chernoff bound for random walks on expander graphs. In *SIAM J. Comput.*, pages 1203–1220, 1998.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. In *IEEE Transactions on Information Theory*, pages 1757–1767, 1999.

[Ham50] R.W. Hamming. Error detecting and error correcting codes. In *The Bell System Technical Journal*, 1950.

[Hea08] Alexander Healy. Randomness-efficient sampling within nc. In *Computational Complexity*, pages 3–37, 2008.

[HILL99] Johan Hstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. In *SIAM J. Comput.*, pages 1364–1396, 1999.

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *FOCS*, pages 248–253, 1989.

[Kah95] Nabil Kahale. Eigenvalues and expansion of regular graphs. In *J. ACM*, pages 1091–1106, 1995.

[Lub94] A. Lubotzky. Discrete groups, expanding graphs and invariant measures,. In *Progress in Mathematics*, 1994.

[Mar73] G.A. Margulis. Explicit constructions of expanders. In *Problemy Peredaci Informacii,*, 1973.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. In *J. Comput. Syst. Sci.*, pages 43–52, 1996.

[Pet60] W. W. Peterson. Encoding and error-correction procedures for the bose-chaudhuri codes. In *IRE Transactions on Information Theory,*, 1960.

[Pin73] Mark S. Pinsker. On the complexity of a concentrator. In *7th International Teletraffic Conference*, 1973.

[PV05] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *FOCS*, pages 285–294, 2005.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. In *Journal of the Society of Industrial and Applied Mathematics*, 1960.

[RSW06] Omer Reingold, Ronen Shaltiel, and Avi Wigderson. Extracting randomness via repeated condensing. In *SIAM J. Comput.*, pages 1185–1209, 2006.

[RTV06] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Pseudorandom walks on regular digraphs and the rl vs. l problem. In *STOC*, pages 457–466, 2006.

[RV05] Eyal Rozenman and Salil P. Vadhan. Derandomized squaring of graphs. In *APPROX-RANDOM*, pages 436–447, 2005.

[RVW01] Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2001.

[Sha48] C.E. Shannon. A mathematical theory of communication. In *The Bell System Technical Journal,*, 1948.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. In *J. Complexity*, pages 180–193, 1997.

[Tan84] M. R. Tanner. Explicit concentrators from generalized n-gons. In *SIAM Journal on Algebraic Discrete Methods,*, 1984.

[vN63] J von Neuman. Various techniques used in conjunction with random digits. In *Collected works. Vol. V:Design of computers, theory of automata and numerical analysis.*, 1963.

[VV85] Umesh V. Vazirani and Vijay V. Vazirani. Random polynomial time is equal to slightly-random polynomial time. In *FOCS*, pages 417–428, 1985.

[Woz58] J. Wozencraft. List decoding. In *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 1958.

[Zuc96] David Zuckerman. Simulating bpp using a general weak random source. In *Algorithmica*, pages 367–391, 1996.