Error-correcting Codes and Computational Complexity

Notes from a series of lectures by

Madhu Sudan
MIT

The 2006 Barbados Workshop on Computational Complexity
February 26th – March 5th, 2006

Organizer:
Denis Thérien

Scribes:
Eric Allender, Arkadev Chattopadhyay, Ricard Gavaldà, Navin Goyal, Valentine Kabanets, Bruce Kapron, Valerie King, Michal Koucký, Benoit Larose, François Lemieux, Pierre McKenzie, Mark Mercer, Toniann Pitassi, Pavel Pudlák, Dana Randall, Mike Saks, Pascal Tesson, Peter Winkler.

Table of Contents

Lecture 1. Introduction	
Lecturer: Madhu Sudan	Scribes: Benoit Larose and Dana Randall

1.1 Introduction

Coding theory dates back to Hamming [25] and Shannon [41] who offered two alternative views on communication with noise. The idea is to encode a message so that even if it is transmitted over a noisy channel, the received message can still be decoded to recover the original message (see Figure 1.1). Shannon and Hamming offered two different formalizations to study when recovery is possible.

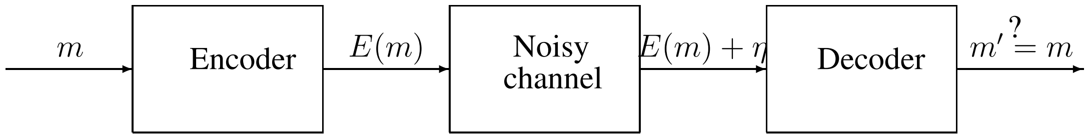

Figure 1.1: The general scheme for encoding/decoding a message m that becomes corrupted with noise η.

In 1948, Shannon [41] defined a probabilistic model where noise is modeled as a stochastic process, the binary symmetric channel. Over the binary alphabet $\{0,1\}$, each bit is unchanged with probability $1-p$ and is flipped with probability p, where p is some small probability. Each bit is acted on independently by this channel. He proposed encoding and decoding schemes under this model. Alternatively, Hamming [25] offered an adversarial model of noise. An adversary is allowed to define the message, is allowed to flip t bits of the message, and gets to choose where these t errors occur. Offering a third perspective, Elias [12] generalized the Hamming viewpoint to list decoding in 1957. In this model a list of up to L possible messages are output by the decoder, where the goal is to have the original message m in the list. This model allows us to connect Shannon's and Hamming's by relating the worst case and average case models for errors.

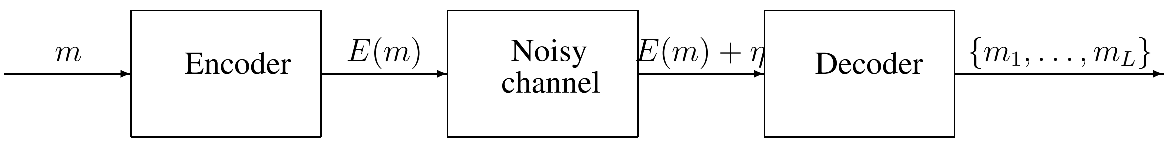

Figure 1.2: The general scheme for encoding/decoding in the list-decoding setting.

Coding theory and complexity theory share a rich history. Complexity naturally enters a study of coding theory because you would like to come up with efficient coders and decoders. But even beyond this, we see many ideas central to complexity being anticipated in coding theory. In 1960, Peterson explicitly mentions that an algorithm is favorable because in runs in polynomial time. In the coding theory literature we see the first notion of RP, namely Berlekamp's 1972 algorithm for factoring polynomials over a finite field. And it is also in the information theory literature that we find Diffie-Hellman's first cryptosystems and Lovasz' first semi-definite program. Moreover, the coding theory fundamentals have played a crucial role in complexity: construction of encodings, limitation proofs, and decoding algorithms.

1.2 Hamming's Perspective: Adversarial Errors

When we talk about encoding and decoding in the context of transmission over a noisy channel, we have some underlying alphabet Σ. For instance, we can take $\Sigma = \{0, 1\}$, or any finite set, or if q is a prime power it is often useful to take $\Sigma = \mathbb{F}_q$ (i.e., the finite field with q elements).

For $x \in \Sigma^n$, let $wt(x)$ be the number of nonzero coordinates of x, where we assume $0 \in \Sigma$.

Hamming's setting is as follows [25]: Given a message $m \in \Sigma^k$, we first encode the message forming $E(m) \in \Sigma^n$, where $n > k$. This encoded message is transmitted and may be perturbed bby some noise vector η added by the channel, that is, the noise message received is $E(m) + \eta$. The noise η is restricted so that $wt(\eta) \leq t$, but is otherwise adversarially chosen.

Let $\Delta(x, y)$ equal the number of coordinates i such that $x_i \neq y_i$ be the *Hamming distance*. Suppose $C = \{E(m) | m \in \Sigma^k\}$ is the set of codewords. Then we let

$$\Delta(C) = \min_{x \neq y \in C}\{\Delta(x, y)\}.$$

Hamming observed that if $\Delta(x, y) \leq 2t$ for any two codewords $x \neq y$, then we cannot possibly uniquely correct errors. Moreover, he showed the converse in the following theorem.

Theorem 1. *C corrects t errors uniquely iff $\Delta(C) \geq 2t + 1$.*

Definition 1. *C is an $[n, k, d]_q$ code if it is the image of $E : \Sigma^k \to \Sigma^n$ and $\Delta(C) = d$.*

One can easily show that Δ is a metric. Hamming observed that if codewords can be uniquely recovered, then balls of radius $(d - 1)/2$ in the metric space must be disjoint. This observation tells us

$$(\#\text{codewords}) \times (\text{volume of a ball}) \leq q^n.$$

If we define Vol(n, r, q) to be the volume of a ball of $\mathbb{F}_q^n$ with radius r, we have

$$\text{Vol}(n, r, q) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i \approx q^{H_q(\frac{r}{n}) \cdot n},$$

where the entropy $H_q(p) = -[p \log_q p + (1 - p) \log_q(1 - p)] + p \log_q(q - 1)$. Notice that $H_2(p)$ is zero when $p = 0$, is 1 when $p = 1 - 1/q$, and is monotone increasing in between.

Lemma 1. *(The Hamming Bound) For any $[n, k, d]_q$ code,*

$$k + \log_q \mathrm{Vol}(n, \frac{d-1}{2}, q) \leq n.$$

To have an asymptotic version, define the *relative distance* of the code as

$$\delta(C) = \frac{\Delta(C)}{n}$$

and the *rate* as

$$R(C) = \frac{k}{n}.$$

Then $\delta(C)$ and $R(C)$ are related by

$$R + H_q(\frac{\delta}{2}) \leq 1.$$

Hamming constructed codes based on these observations.

- By using a bit for parity check, we have that there exists an $[n, n-1, 2]_q$ code.
- There exists an $[n, n - \log_q(1 + (q-1)n), 3]_q$ code if q is a prime power. (Note that this matches the lower bound that states $k + \log_q \mathrm{Vol}(n, \frac{d-1}{2}, q) \leq n$.)
- It follows that there exists an $[n+1, n - \log_q(1+n), 4]_2$ code. This relies on a simple general rule that arises by adding a parity check bit. Thus, the existence of a $[n, k, 2d-1]_2$ code always implies the existence of a $[n+1, k, 2d]_2$ code.

1.3 Shannon's Perspective: Probabilistic Errors

In Shannon's model [41], a q-ary symmetric channel is used to transmit letters, where each letter has a probability p of changing, independently of all other letters sent. Thus, if $\sigma \in \Sigma$ is transmitted, then σ is received with probability $1-p$ and each $\sigma' \neq \sigma$ is received with probability $p/(q-1)$.

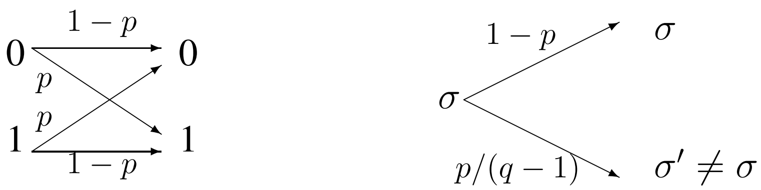

Figure 1.3: The binary and q-ary symmetric channels.

The encoding function is $E : \Sigma^k \to \Sigma^n$ and the decoding function is $D : \Sigma^n \to \Sigma^k$. The noise representing the bits that are changed in the q-ary symmetric channel is represented by "$\eta \in qSC$". Then the probability of error that we wish to bound is

$$\Pr_{\substack{m \in_R \Sigma^k, \\ \eta \in qSC}} [D(E(m) + \eta) \neq m].$$

Let the rate R be defined as k/n. Shannon's theorem shows that there is a sharp transition as R increases.

Theorem 2. *(1) If $R < 1 - H_q(p)$, then* $\Pr[\text{decoding error}] \to 0$ *exponentially fast in n.*
(2) If $R > 1 - H_q(p)$, then $\Pr[\text{decoding error}] \to 1$ *exponentially fast in n.*

Shannon's proof of part (1) is non-constructive. It is based on using a random coding function $E : \Sigma^k \to \Sigma^n$ that maps the possible messages (also assumed to be random) to random encodings. The decoding function is the best possible that finds the closest message, assuming that m and η are random. This part was made constructive in 1966 by the work of Forney [13], which we will review in the next lectures.

Proof. To show the coding theorem, we define the encoding $E : \Sigma^k \to \Sigma^n$ at random. To define the decoding function $D : \Sigma^n \to \Sigma^k$, we let $\bar{r} \in \Sigma^n$ be a received message. If there exists a unique $m \in \Sigma^k$ such that $\Delta(\bar{r}, E(m)) \leq (p + \epsilon)n$, then output m; else output "error."

We need to bound the probability that an error occurs. There are two error events we need to be concerned about:

- <u>E1:</u> The first type of error is that many errors occur. We we want to bound $\Pr[\#errors \geq (p + \epsilon)n]$. By standard tail inequalities we can show that $\Pr(\text{E1}) = \exp(-\epsilon^2 n)$.

- <u>E2:</u> The second type of error is that there is an $m' \neq m$ whose encoded version r' falls in the ball of radius $(p + \epsilon)n$ around the encoded version of m, r, in which case there may be an incorrect decoding. The probability that any message falls within the ball of radius $(p + \epsilon)n$ around the received message r is $\text{Vol}(n, q, (p + \epsilon)n)/q^n$. Taking the union bound, we find that
$$\Pr[\exists\, m' \neq m \text{ s.t. } \Delta(E(m'), r) \leq (p + \epsilon)n] \;\leq\; q^k \frac{\text{Vol}(n, q, (p + \epsilon)n)}{q^n}.$$
This quantity goes to 0 when
$$k + \log_q \text{Vol}(n, q, (p + \epsilon)n)) < n$$
which is to say
$$R + H_q(p) < 1.$$

Thus, $\Pr[\text{error}] \leq \Pr[\text{E1}] + \Pr[\text{E2}] \to 0$ when $R < 1 - H_q(p)$.

To prove part (2) we will show that the probability of correctly decoding tends to 0 when $R > 1 - H_q(p)$. Again, there are two events whose probability we have to bound.

- E1': The first event in this case if there are *fewer* than $(p-\epsilon)n$ errors. Again simple tail bounds show that $\Pr[\text{E1'}] < \exp(-\epsilon^2 n)$.

- E2': The second event is that, given that more than $(p-\epsilon)n$ errors occurred (i.e., $wt(\eta) > (p-\epsilon)n$), correct decoding is possible. We show that this event has tiny probability. For any $x \in q^n$ that lies outside of the ball of radius $(p-\epsilon)n$ around $E(m)$, the probability of receiving x is bounded by

$$\Pr[\text{receiving } x] \leq \frac{1}{\binom{n}{(p-\epsilon)n} \cdot (q-1)^{(p-\epsilon)n}} \approx\leq \frac{1}{\binom{n}{pn} \cdot (q-1)^{pn}},$$

where the first inequality is derived by symmetry. For any integer t, the number of "error patterns" with exactly t errors is $\binom{n}{t}.(q-1)^t$ and all these errors are equally likely. So the probability of any particular error pattern is at most $\frac{1}{\binom{n}{t}\cdot(q-1)^t}$. Setting $t = (p-\epsilon)n$ yields the first inequality above. Now summing over the choices for x (there are at most q^n of them), and recalling that the messages were randomly chosen in Σ^k, the probability E2' that we can correctly decode is

$$\Pr[\text{E2'}] \leq \frac{q^n}{\binom{n}{pn}(q-1)^{pn}} \cdot \frac{1}{q^k}.$$

Taking the logarithm, we find

$$\log_q \Pr[\text{E2'}] \leq n(1 - \frac{k}{n} - H_q(p)),$$

so $\Pr[\text{E2'}] < \exp\left(n(1 - H_q(p) - R)\right)$.

Putting the pieces together, we find $\Pr[\text{correct decoding}] \leq \Pr[\text{E1'}] + \Pr[\text{E2'}] \to 0$ when $R > 1 - H_q(p)$. □

1.4 Elias' Perspective: List Decoding

Elias [12] introduced the idea of a (p, L) list-decodable code, where you should be able to correct a fraction of p adversarial errors with a list of size at most L that contains the correct message. In addition, we would like that $L = poly(n)$. Thus $E : \Sigma^k \to \Sigma^n$ as before, but $D : \Sigma^n \to \binom{\Sigma^k}{L}$ such that for all m, for all η with $wt(\eta) \leq pn$, we have $m \in D[E(m) + \eta]$. This translates to the requirement that for every ball B of radius pn, we have $|B \cap C| \leq L$, rather than the stronger $|B \cap C| = 0$.

It is easy to see that any Hamming code correcting t errors is a $(t/n, 1)$ list-decodable code. There is not a straightforward reduction from Shannon codes to the list decoding model, because a nice list decodable model might, for instance, completely ignore the first bit. Assuming that the list-decoding does not have such contrived properties then there is such a reduction, as indicated by the dotted line in Figure 1.4. The reduction is as follows: we shall assume that $\Sigma = \mathbb{F}_q$. Let $C \subseteq \Sigma^n$ be a (p, L)-code; to obtain a Shannon code with it, we will in fact build a Hamming code.

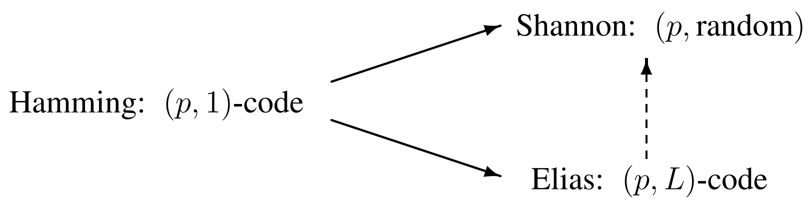

Figure 1.4: The relationship between the Shannon, Hamming, and Elias models.

Intuitively we want to pick out one member from each list; notice that the code C' we build is not explicit. Define $C' \subseteq \Sigma^n$ as

$$C' = C \cap S$$

where $S \leq \Sigma^n$ is a randomly chosen linear subspace of dimension $n - \log L$. We have that

$$|S| = |\Sigma^{n-\log L}| = \frac{\Sigma^n}{L}$$

hence C' contains ≈ 1 codeword from each list.

1.5 Existential and Universal Bounds on Hamming Codes

In this section we investigate the existence of codes meeting certain conditions on parameters and limitations on the performance of codes.

1.5.1 The Gilbert-Varshamov Bound: the Random (Greedy) Code

The Gilbert-Varshamov bound asserts the existence of codes of rate R and relative distance satisfying

$$R + H_q(\delta) \geq 1.$$

Intuitively, the construction proceeds as follows: pick a word in Σ^n at random; throw away the ball of radius d around it; pick the next codeword, and repeat the process. Formally: consider the graph G with vertices $V(G) = \Sigma^n$ where $xy \in E(G)$ is an edge of G if $\Delta(x, y) < d$. Then the code described above is simply a maximal independent set in G. Let $\alpha(G)$ denote the size of such a set and let D denote the degree of any vertex in G. A simple counting argument yields

$$|V(G)| \leq \alpha(G)(D + 1).$$

We can translate this as follows: using the fact that $D = \mathrm{Vol}(n, q, d-1) - 1$, $|V(G)| = q^n$ and $\alpha(G) = q^k$, we have obtained a $[n, k, d]$-code with $k = \lfloor \log_q \alpha(G) \rfloor$ satisfying the *Gilbert-Varshamov bound*

$$q^n \leq q^k \,\mathrm{Vol}(n, q, d-1).$$

Some improvements are possible. Letting $N = |\Sigma^n|$, Jiang and Vardy [27] have proved that if

$$\frac{\#\text{ of triangles in } G}{|V(G)|} \leq D^{2-\epsilon}$$

then

$$\alpha(G) \geq \frac{N}{D} \log D,$$

whence

$$N \leq \frac{\alpha(G) D}{\log D}.$$

Translating, we obtain

$$q^n \leq \frac{q^k \mathrm{Vol}(n, q, d-1)}{\mathcal{O}(d)}.$$

We have the following results:

- (G-V bound) there exists a code C such that $q^n \leq q^k \mathrm{Vol}(n, q, d-1)$;
- (Hamming bound) for every code C we have $q^n \geq q^k \mathrm{Vol}(n, q, \frac{d-1}{2})$;

The asymptotic versions of these are as follows:

- (G-V bound) $R + H_q(\delta) \geq 1$;
- (Hamming bound) $R + H_q(\delta/2) \leq 1$.

Two more bounds not covered in these are lectures are the Plotkin bound and (the different forms of) the Linear Programming bound. Madhu's depiction of all these upper and lower bounds appears in Figure 1.5.

1.5.2 The Elias-Bassalygo Bound and the Johnson Bound

The Elias-Bassalygo bound states that, asymptotically, if a q-ary code of rate R and relative distance δ exists then

$$R \leq 1 - H_q\left(\left(1 - \frac{1}{q}\right) \cdot \left(1 - \sqrt{1 - \frac{q}{q-1}\delta}\right)\right).$$

For simplicity we will consider only the case $q = 2$; in this case

$$R \leq 1 - H_2\left(\frac{1}{2} \cdot \left(1 - \sqrt{1 - 2\delta}\right)\right).$$

The idea of the proof is to revisit Hamming's bound (which showed that $R \leq 1 - H(\delta/2)$) and improve it. Roughly Hamming's proof notes that for every $[n, k, d]_2$ code, we have, centered at the codewords, *disjoint* balls of radius $\frac{d-1}{2}$ in Σ^n. In other words, an $[n, k, d]_2$ code is also a

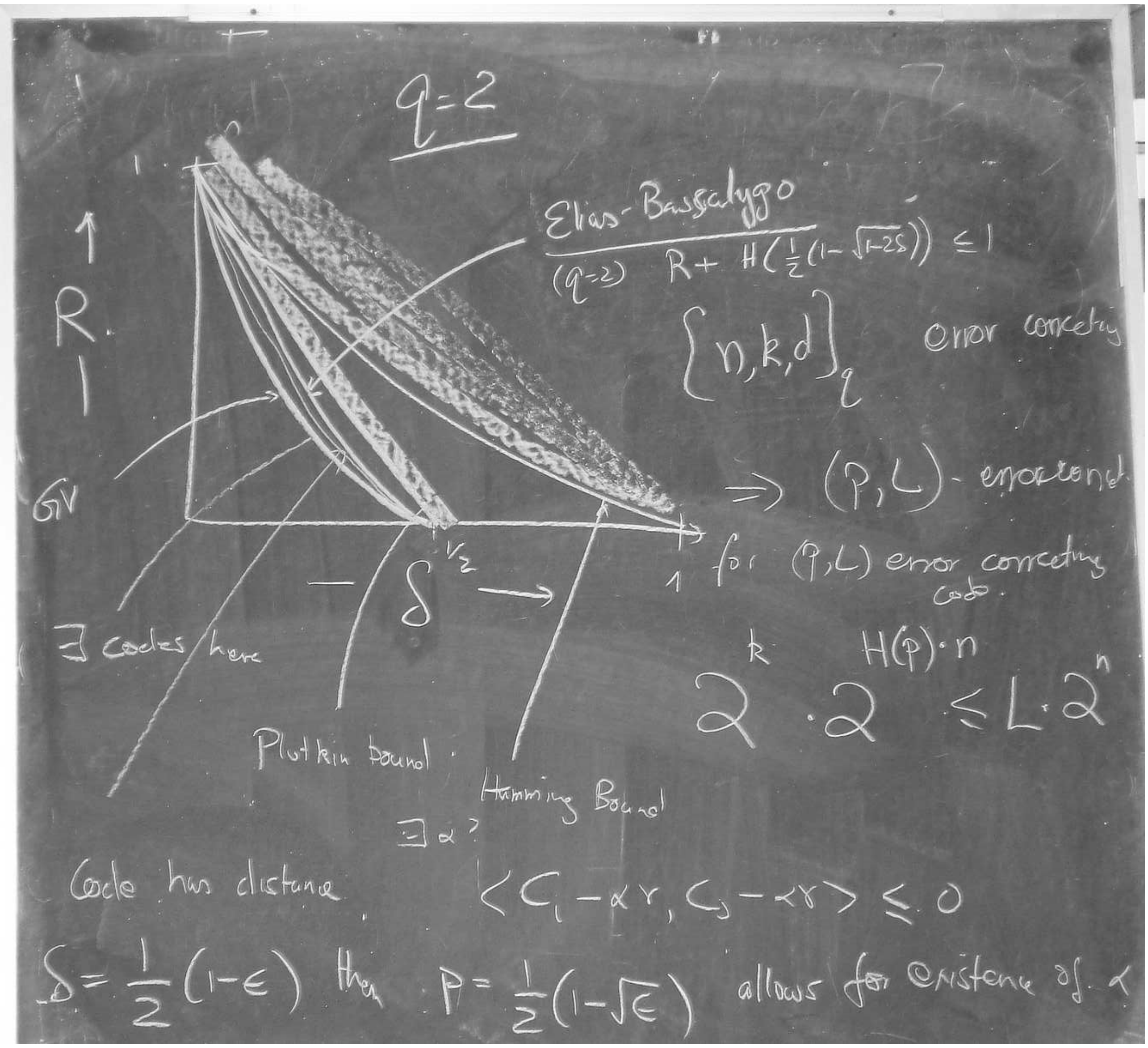

Figure 1.5: Various bounds.

$(\frac{d-1}{2} \approx \frac{\delta}{2}, 1)$-list-decodable code. Suppose we could prove that an $[n, k, d]_2$ code is also a (p, L)-code for some small L and $p > \delta/2$. Then we can create balls of radius pn around the codewords and claim that the sum of their volumes is at most $L \cdot 2^n$ and get

$$\mathrm{Vol}(n, k, pn)2^k \leq L\, 2^n.$$

This is the essence of the Elias-Bassalygo bound/proof. To apply this idea they employ what is also known as the Johnson Bound which proves that every $[n, k, d]_2$ code is also a $(p, O(n))$-list-decodable code for $p = \left(\frac{1}{2} \cdot \left(1 - \sqrt{1 - 2\delta}\right)\right)$. Below we prove the Johnson bound (and thus also the Elias-Bassalygo bound).

Let $t = pn$. Suppose that we have a word $\overline{r} \in \{0, 1\}^n$ and codewords $\overline{c_1}, \ldots, \overline{c_L} \in \{0, 1\}^n$ such that

1. for all i, $\Delta(\overline{r}, \overline{c_i}) \leq t$, and
2. for all $i \neq j$, $\Delta(\overline{c_i}, \overline{c_j}) \geq d$.

We would like to give an upper bound on the value of L.

Aside: Before we jump into the proof, lets try to get a sense of where this weird setting of p comes from. Suppose you wished to construct a received vector $\overline{r}$ and exponentially many vectors $\overline{c_1}, \ldots, \overline{c_L}$ such that the $\overline{c_i}$'s are close to $\overline{r}$ but far from each other, how would you do it? We claim that the natural approach is to set (w.l.o.g.) $\overline{r} = \overline{0}$. Now if we wish the $\overline{c_i}$'s to be within distance pn from $\overline{r}$ then a natural idea is to pick the coordinates of $\overline{c_i}$ independently to be 1 with probability p and 0 with probability $1 - p$. If we do this, we would expect we can find exponentially many codewords that lie at distance roughly t from $\overline{r}$. What about the pairwise distance? We would expect $\overline{c_i}$ and $\overline{c_j}$ to differ on roughly $2p(1 - p) \cdot n$ coordinates. To make sure that this distance is δn, we would pick p so that $2p(1 - p) = \delta$. This is exactly the setting we have above, and the proof below shows that one can't do much better than the random construction described in this paragraph.

We now prove that L can't be too large under the conditions (1) and (2) above. The proof technique consists in mapping the code to $\mathbb{R}^n$ by mapping $0 \mapsto 1$ and $1 \mapsto -1$. We then view $\overline{r}, \overline{c_1}, \ldots, \overline{c_L} \in \{-1, 1\} \subseteq \mathbb{R}^n$. Since every coordinate is ± 1 we obviously have

$$\|\overline{r}\| = \|\overline{c_1}\| = \cdots \|\overline{c_L}\| = n$$

and, for all i,

$$\|\overline{r} - \overline{c_i}\|^2 \leq 4t$$

while

$$\|\overline{c_i} - \overline{c_j}\|^2 \geq 4d$$

for all $i \neq j$. Scaling the above by n we obtain

$$\|\overline{r}\| = \|\overline{c_1}\| = \cdots \|\overline{c_L}\| = 1$$
$$\|\overline{r} - \overline{c_i}\|^2 \leq 4p$$
$$\|\overline{c_i} - \overline{c_j}\|^2 \geq 4\delta$$

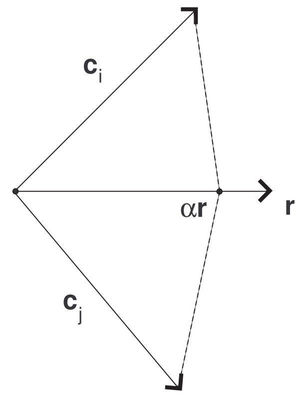

Figure 1.6: Moving the origin to $\alpha\overline{r}$.

(recall that $\delta = d/n$); we can rewrite the last two conditions in terms of the inner product, to obtain

$$\langle \overline{c_i}, \overline{c_j} \rangle \leq 1 - 2\delta$$
$$\langle \overline{r}, \overline{c_i} \rangle \geq 1 - 2p$$

for all $1 \leq i \neq j \leq L$. For a fixed angle A, one can find exponentially (in n) many vectors in $\mathbb{R}^n$ which differ by an angle of at least A, provided $A < 90°$; if $A \geq 90°$ then one can find at most n such vectors. So we try to find $0 \leq \alpha \leq 1$ such that

$$\langle \overline{c_i} - \alpha\overline{r}, \overline{c_j} - \alpha\overline{r} \rangle \leq 0.$$

I.e., we "move the origin" to $\alpha\overline{r}$ so that the resulting vectors all differ by angles greater than $90°$. It is possible to find such an α if $\delta = \frac{1}{2}(1-\epsilon)$, and then p can be chosen to be $\frac{1}{2}(1-\sqrt{\epsilon})$. This process is illustrated in Figure 1.6.

Lecture 2. Linear Codes

Lecturer: Madhu Sudan | Scribes: Mark Mercer and Pascal Tesson

2.1 Recap and the Singleton Bound

In Lecture 1 we defined the *rate* R of an $[n, k, d]_q$-code C to be the ratio k/n, and the *relative minimum distance* δ to be d/n. Ideally we would like codes with high rate (i.e. as close to 1 as possible) and high relative minimum distance. We are interested in tradeoffs between these two quantities.

The *singleton bound* provides a simple inequality.

Theorem 3. *If $C : \Sigma^k \to \Sigma^n$ is an $[n, k, d]_q$ code, then $k + d \leq n + 1$.*

Proof. For $x \in \Sigma^k$, define S_x to be the set of all words in Σ^n that agree with $C(x)$ on the first $n - d + 1$ coordinates. Observe that $S_x \cap S_{x'} = \emptyset$ for $x \neq x'$ as long as C is injective. Each set S_x is of size q^{d-1}, and there are at most q^k of them, so $q^{d-1}q^k \leq q^n$ which implies $k + d \leq n + 1$. □

Figure 1.5 gave the rate versus minimum distance tradeoffs permitted by the Singleton, Plotkin, and Gilbert-Varshamov bounds.

In particular, if $\delta = (1 - \frac{1}{q})(1 - \varepsilon)$ then $R = \tilde{\Theta}(\varepsilon^2)$; random codes will have this property. In this lecture, we want to start constructing explicit codes that attain some of these bounds.

2.2 Linear Codes: Generalities

Let us assume that $\Sigma = \mathbb{F}_q$. We can then think of Σ^k and Σ^n as vector spaces and we say that $C : \Sigma^k \to \Sigma^n$ is a *linear code* if $C \subseteq \mathbb{F}_q^n$ is a linear subspace.

There are a number of advantages to working with linear codes. First, they have a nice, simple representation: since C is linear, the transformation can be expressed by an $n \times k$ matrix G called the *generator matrix*.

In particular, if C is a linear code then there exists k linearly independent codewords $v_1, \ldots, v_k$ such that message $\overline{\lambda} = (\lambda_1, \ldots, \lambda_k) \in \mathbb{F}_q^k$ gets mapped to $\overline{\lambda}G = \sum_{i=1}^{k} \lambda_i v_i$ with $\lambda_i \in \mathbb{F}_q$. The *generator matrix* of the code is then the following $n \times k$ matrix:

$$G = \begin{bmatrix} \longleftarrow v_1 \longrightarrow \\ \longleftarrow v_2 \longrightarrow \\ \cdots \\ \longleftarrow v_k \longrightarrow \end{bmatrix}$$

The code space is therefore $\{\overline{\lambda} \cdot G : \overline{\lambda} \in \mathbb{F}_q^k\}$ Since the v_is are all linearly independent, the dimension of the code space is k. Observe that there exists $n - k$ linearly independent vectors

$y_1, \dots, y_{n-k}$ such that every v_i is orthogonal to every y_j where the inner product is defined as usual by $\langle v, y \rangle = \sum_{i=1}^{n} v^{(i)} y^{(i)}$.

We define the *parity check matrix* of the code to be the matrix H whose jth column is y_j:

$$H = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ y_1 & y_2 & \cdots & y_{n-k} \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}.$$

The generator matrix G and the parity check matrix H form dual representations of the code and one can construct H from G, and likewise, from a parity matrix H one can build an appropriate generator matrix G for the corresponding codespace.

For $z \in \Sigma^n$, the ith bit of $z \cdot H$ is $\langle z, y_i \rangle$. Then the set of codewords is exactly the set of vectors which are in the nullspace of the parity matrix. Thus for linear codes there is an easy algorithm to test whether a given $z \in \mathbb{F}_q^n$ is a codeword which consists of computing zH and checking if the result is the null vector.

2.3 Hamming-Varshamov Codes

Given a parity check matrix H we would like to understand the conditions sufficient for the corresponding code C to have distance d.

As before, define the *weight* of $x \in \Sigma^n$, or $wt(x)$, to be the number of nonzero entries in x. We claim that the minimum distance of C is equal to the weight of the lightest nonzero codeword. To see this, first observe that for a linear code, 0^k is a valid message and it has an encoding $C(x) = 0^k \cdot G = 0^n$, and so 0^n is always a codeword. Hence the weight of all nonzero codewords must be at least the distance d. Conversely by definition of distance there exist codewords $x, y \in C$ s.t. $\Delta(x, y) = d$. Since the code is linear, both $x - x$ and $x - y$ are also codewords and we have $d = \Delta(x, y) = \Delta(x - x, x - y) = \Delta(0, y) = wt(y - x)$.

The challenge of course is now to construct a parity check matrix whose null space contains only relatively heavy vectors besides 0^n. Here is an iterative strategy for building such a parity check matrix. We construct the matrix for an $[n, k, d]_q$ code row by row and denote the rows as $h_1, \dots, h_n$.

We require that every word $x \in \Sigma^n$ with $wt(x) < d$ is not in the nullspace of H. Suppose for example that x has weight $d - 1$ with its first $d - 1$ bits being non-zero. We have $xH = 0$ if $\sum_{i=1}^{d-1} x_i h_i = \vec{0}$, and this example shows that we want to construct H so that any set of up to $d - 1$ of the rows $h_1, \dots, h_n$ are linearly independent.

We can simply construct H greedily, adding new rows to H such that such linear dependencies do not occur. The number of constraints that are placed on our choice for the row h_n is about $\binom{n-1}{d-2} \cdot q^{d-2}$. If the latter is smaller than q^m where $m = n - k$ then this process will successfully provide us with the desired code.

Theorem 4. *There exist linear codes of distance d and length n with $k = n - m$ with $m \approx \log_q \binom{n-1}{d-2} q^{d-2}$.*

These codes slightly outperform the greedy codes that were outlined in Lecture 1. If $d = 3$, then we get $k \approx n - \log_q n$.

The analysis above was first given by Hamming for $d = 3$ and the corresponding codes are often known as Hamming codes (particularly for $q = 2$). The analysis was later generalized by Varshamov.

2.4 From Polynomials to Linear Codes

We now describe codes which are conceptually rather different than the ones seen thus far. They all rely on viewing messages as the coefficients of a polynomial over some finite field.

2.4.1 Univariate Polynomials: Reed-Solomon Codes

Our first example will be the *Reed-Solomon codes*. Pick some $q > n$ and let $\Sigma = \mathbb{F}_q$. Fix $\alpha_1, \ldots, \alpha_n$ to be distinct elements of $\mathbb{F}_q$. Our messages are coefficients of a polynomial over $\mathbb{F}_q$.

Let $m = m_0 \ldots m_{k-1}$ be the message. We define the univariate polynomial $M(x) = \sum_{i=0}^{k-1} m_i x^i$. The Reed-Solomon encoding of m is then $(M(\alpha_1), M(\alpha_2), \ldots, M(\alpha_n))$, the value of M on the chosen α points. Note that the codewords indeed form a linear subspace of $\mathbb{F}_q^n$. Moreover, the simple fact that any univariate polynomial of degree $k-1$ has at most $k-1$ zeros implies that this code has distance $n - (k-1)$.

Observe that Reed-Solomon codes achieve optimal rate versus relative minimum distance tradeoffs with respect to the Singleton bound. However they requires us to use an alphabet of size at least n. We would prefer codes over a much smaller alphabet; it is natural to try to achieve this by generalizing Reed-Solomon codes to multivariate polynomials.

2.4.2 Multivariate Polynomials: Reed-Muller Codes

We will encode the message using a degree r polynomial, $r < q$, over $\mathbb{F}_q$ in m variables. The number of monomials in such a polynomial is $\binom{m+r}{r} \geq (\frac{r}{m})^m$. Let M be a polynomial formed by making each message letter m_i the coefficient of a degree r polynomial in m variables.

The obvious encoding scheme is to form the encoded message by giving the evaluation of M on all points and the resulting codes are known as *Reed-Muller codes* of order r.

The Schwartz-Zippel lemma gives us a lower bound on the fraction of zeroes of the polynomial M, and this implies that the relative distance δ for the Reed-Muller code is at least $1 - \frac{r}{q}$ for $r < q$. So this gives a way of encoding messages of size at least $(\frac{r}{m})^m$ to messages of size q^m with $\delta(C) \geq 1 - \frac{r}{q}$.

2.4.3 Algebraic Geometric Codes

This is not quite as good as we would have liked: simply relying on Schwartz-Zippel forces us to evaluate the polynomials at all points to get a satisfactory guarantee on the distance. However, complicated results in algebraic geometry show that, in some cases, one can pick some $S \subseteq \mathbb{F}_q^m$

very carefully so that the code obtained by evaluating the polynomials on the points of S still has large distance. These are the so-called *algebraic geometric codes* or *AG codes*. Specifically, if q is a square then we can get AG codes over $\mathbb{F}_q$ of length n, dimension k and distance $n - k - \frac{n}{\sqrt{q}-1}$. In other words, if R is the rate then the relative distance of the code is $\delta \geq 1 - R - \frac{1}{\sqrt{q}-1}$.

In comparison, the greedy codes give us $\delta \geq 1 - R - \frac{1}{\log q}$. We do not believe that it is possible to get $\delta \geq 1 - R - \frac{1}{q}$.

The AG codes are not optimal in any sense; they are just good explicit examples. They have not had much application in theoretical CS (yet?).

2.4.4 Hadamard Codes as a Special Case of Reed-Muller

In the definition of Reed-Muller codes, we are free to choose the parameters q, r and m and different settings for them lead to codes with varying properties. Consider the following interesting choice of the parameters underlying the definition of the Reed-Muller codes. Let the number of variables m be arbitrary, let the degree r be 1, let $q = 2$, and let the message length be $k = m + 1$.

In other words, we associate to the message $c = c_0 c_1 \dots c_m \in \mathbb{F}_2^m$ the polynomial

$$c_0 + \sum_{i=1}^{m} c_i x_i = f(x_1, \dots, x_m).$$

In fact it is convenient to restrict ourselves to messages where $c_0 = 0$ so that f is a linear function and the message length is simply m. Then the encoding $C : \{0,1\}^m \to \{0,1\}^{2^m}$ is the evaluation of f at all points of $\{0,1\}^m$. This is the so-called *Hadamard code*.

Observe that $f(x_1, \dots, x_m) = \langle x, c \rangle$. In other words, the letters in the encoding are the set of all inner products of a vector $x \in \{0,1\}^m$ with c. This encoding has the property that pairwise codewords agree in exactly half of the locations, which is optimal for the two-letter alphabet.

2.4.5 A Different Choice of Parameters

Now let us fix the degree of the polynomial to $q/2$ and consider different choices of m.

- $m = 1$ corresponds to the Reed-Solomon code. It achieves rate $R = \frac{1}{2}$ and relative distance $\delta = \frac{1}{2}$. However, this requires $n \leq q$.

- $m = 2$ gives a rate $R = \frac{1}{4}$ and relative distance $\delta = \frac{1}{2}$, with $n \leq q^2$.

- For general m we get a rate of $\frac{1}{2^m}$ with $\delta = \frac{1}{2}$, $n \leq q^m$

One choice of parameters which has proved useful in applications is $m \approx \frac{\log n}{\log \log n}$, field size $q \approx \log^2 n$, degree $r \approx \log n$. This gives messages of length k and codewords of length $n = k^2$.

2.5 Concatenated Codes: Forney and Justesen Constructions

We would like to construct good codes for small alphabets, particularly for the case of $q = 2$. In particular we would like to obtain good binary codes by adapting our results for $q > 2$.

Suppose we have a Q-ary code $C : \Sigma^K \to \Sigma^N$ with $Q = |\Sigma| = 2^\ell$. We can easily obtain a binary code C' from C. Assume that our binary message m is of length $K\ell$: we first break m into chunks of ℓ bits and interpret it as a string M over Σ. The code C can be applied to M and the resulting string of length N over Σ can be converted back into a bit-string of length $N\ell$. The rate and distance of the code C' is the same as C. However the relative minimum distance can go from δ to $\frac{\delta}{\ell}$ because each discrepancy over the Q-ary code may translate in a discrepancy on only one of the ℓ bits of the corresponding blocks.

We can improve on this strategy by taking each ℓ-sized segment of the encoded message $C'(m)$ and applying a good binary code $B : \{0,1\}^\ell \to \{0,1\}^{\ell'}$ to it. This general paradigm is known as *concatenated codes* and was introduced by Forney [13]. We call C the *outer code* and B the *inner code*. If d is the distance for B and D is the distance for C, then the distance for the concatenated code is dD.

For example, the random binary codes map ℓ bits to 2ℓ bits with distance $\ell/100$. The resulting concatenated code will map ℓK to $2\ell N$ bits with distance $D\ell/100$ and relative distance $\delta/100$. So by doubling the size of the encoding we can get a significant improvement in the relative minimum distance. Using this technique, we can get codes with rates that are asymptotically close to the channel capacity. Of course, the example above makes sense computationally only because $\ell \approx \log Q$ and so brute force can reasonably be used to find a satisfactory inner code.

Although Forney's construction can reasonably be considered explicit we cannot easily compute, say, the (i, j)th entry of the generator matrix. This problem was addressed by Justesen [28] who proposed using a different inner code for each block of bits representing a symbol of the outer code. The advantage of the Justesen codes is that in order for the concatenated code to have good distance properties, it suffices to insure that each of these inner codes has good distance properties with high probability. This idea enables a more explicit construction than Forney's.

2.6 BCH Codes

An alternative strategy for producing binary codes from linear codes over $\mathbb{F}_q$ with large q is to simply restrict the code to codewords which are strings in $\{0_\mathbb{F}, 1_\mathbb{F}\}$. Somewhat surprisingly, this often works quite well. The restriction of the Reed-Muller codes are called *BCH codes* after the authors Bose and Ray-Chaudhuri, and Hocquenghem, who defined these codes in 1959. BCH codes have good distance properties, for example we can use this strategy to obtain $[n, n - 5\log n, 11]_2$ codes.

Lecture 3. More Code Constructions and Pseudorandomness

Lecturer: Madhu Sudan   Scribes: Arkadev Chattopadhyay and Ricard Gavaldà

3.1 Review of Previous Code Constructions

Let us review the constructions of explicit codes that we saw in the previous lecture. All these constructions, except for the Reed-Muller code, are remarkable in at least two senses: Each one meets one of the lower bounds that we stated in Lectures 1 and 2, and they all are better than picking a code at random. The Reed-Muller code is somewhat less motivated from our viewpoint, and it does not exactly match any of the lower bounds, but we will find some applications later on.

Recall that all these codes are linear. Furthermore, they are "efficient" in the sense that we can build their generator matrix in polynomial time. We will later discuss this point in more detail.

Reed-Solomon code. Recall: Fix n elements in your field, $\alpha_1, \ldots, \alpha_n$; each message $\vec{c} = (c_0, \ldots, c_{k-1})$ defines the polynomial $M(\alpha) = \sum_i c_i \alpha^i$, of degree $k-1$. The codeword for $\vec{c}$ is then $(M(\alpha_1), \ldots, M(\alpha_n))$. This code meets exactly the singleton bound: $k + d \leq n + 1$.

Reed-Muller codes. An extension of Reed-Solomon codes using multivariate polynomials.

Hadamard code. Recall: Given a message $c = (c_0, \ldots, c_{k-1})$ of k bits, its Hadamard code is the 2^k-bit string $\langle c, x \rangle_{x \in \{0,1\}^k}$. Thus, the Hadamard code produces long codewords, but has perfect relative distance $1/2$ between codewords.

Algebraic-Geometric (or AG) codes. These achieve the impressive relation $R + \delta = 1 - \frac{1}{\sqrt{q}-1}$. The constructions become fairly complicated, and in particular it was difficult to come up with an explicit (poly-time constructible) description of their generator matrices. They are still not as explicit as e.g., Hadamard or Reed-Solomon, in the sense that building its (i,j)-th entry can be done in time polynomial in $|i| + |j|$. **BCH codes.** For a given (constant) distance d, these codes expand messages of $n - \frac{d}{2} \log n$ bits to n-bit codewords, hence almost meeting the Hamming bound. Note that Reed-Solomon code extends $n - d \log n$ bits to n bits.

Concatenated codes. We also introduced concatenated codes and discussed briefly their potential for some settings of the parameters. Recall that the general idea was to encode the message using a code with large output alphabet, then encode these large letters into $\{0, 1\}$ using another code. In the next section we look at a couple of interesting instances.

3.2 Examples of Concatenated Codes

Example 1. Reed-Solomon code concatenated with a "good" binary code

Take a code over the n-element field that maps ϵn field elements to n field elements (hence, having rate ϵ). Reed-Solomon is an example: it does the above with distance $n - \epsilon n$, i.e., every two codewords differ in a $1 - \epsilon$ fraction of places.

Then, take every $\log n$ bits (=letters in the n-ary alphabet) and encode them in $m = \Omega(\log n)$ bits, with relative distance say $\frac{1}{2} - \epsilon$. At this rate, the random code is about best, and it achieves a rate of ϵ^2, so let us take $m = (\log n)/\epsilon^2$. We can find this code by exhaustive enumeration: just check all $n^{\log n}$ possibilities. The generator matrix has size $N = 2^n$, thus it can be computed in polynomial time (in N).

Putting both codes together, we have:

- Rate: $\epsilon n \log n$ bits are expanded to $n \log n/\epsilon^2$ bits, so rate is ϵ^3.
- Distance: Observe that any two codewords from the first code must have at least $(1-\epsilon)n$ different $(\log n)$-bit blocks, and that two such different blocks, after encoding into $(\log n)/\epsilon^2$ bits, will differ in a fraction of at least $(1/2 - \epsilon)$ bits. Therefore the relative distance is

$$\frac{(1-\epsilon)n \cdot (\frac{1}{2} - \epsilon) \cdot (\log n)/\epsilon^2}{(n \log n)/\epsilon^2} = (1-\epsilon) \cdot (\frac{1}{2} - \epsilon) \cong \frac{1}{2} - \frac{3}{2}\epsilon.$$

In summary, we achieve rate $O(\epsilon^3)$ and distance $1/2 - \epsilon$, with polynomial-time encoding. A rate of about ϵ^2 is about optimal by the Linear Programming bound. The gap from ϵ^2 to $O(\epsilon^3)$ is the price we have to pay, with our current knowledge, to have an explicit construction.

Example 2. AG codes concatenated with the Hadamard code

Can we get a code of rate ϵ^2 and distance $1 - \epsilon$? Let us try using the powerful AG codes; we will not succeed, but will get a code that is more explicit and has performance slightly better than the one above.

Since AG codes have $R + \delta = 1 - \frac{1}{\sqrt{q}-1}$, to get $R = \epsilon^2$ and $\delta = 1 - \epsilon$ we pick an AG code with $q = 1/\epsilon^2$.

Concatenate this AG code with a Hadamard code. This expands $\log(1/\epsilon^2)$-bit blocks to $(1/\epsilon^2)$-bit codewords. The rate is thus $\epsilon^2 \log(1/\epsilon^2)$, and relative distance is $1/2$. What we are using here is that the number of codewords is (constant) $q = 1/\epsilon^2$, so the usual problem of the Hadamard code (long codewords) is not important.

The concatenated code thus has:

- Relative distance $\dfrac{1}{2} - \dfrac{\epsilon}{2}$.
- Rate $\dfrac{\epsilon}{2} \cdot \epsilon^2 \log \dfrac{1}{\epsilon^2} = \dfrac{\epsilon^3}{2} \log \dfrac{1}{\epsilon^2}$.

I.e., same distance as Example 1, a slightly better rate, and no exhaustive enumeration.

3.3 Graph-Theoretic Codes

The idea of using graphs with good expansion rate to build error-correcting codes goes back to Gallagher [14]. It was refined by Tanner [46, 47] and made a substantial advance with the work of Sipser and Spielman [42].

The general ideas is as follows: Take a bipartite graph G with n vertices on the left and $m = n - k$ vertices on the right. Each vertex on the left has degree c and each vertex on the right has degree d, hence $cn = dm$ by counting.

For every such graph, we can define a code as follows: A word $(c_1, \ldots, c_n) \in \{0,1\}^n$ is a code if for every vertex $v \in [m]$ on the right the following parity condition is satisfied:

$$\sum_{(i,v)\in G} c_i = 0 \bmod 2.$$

Note: Because of this condition, these codes are usually called "Low Density Parity Check Codes", in the coding theory community.

Let us argue first that this defines a linear code. Note that 1) 0^n is a codeword (certainly it satisfies all parity checks), and 2) the constraints form a system of homogeneous equations of dimension $n - k$ and n unknowns. Therefore, there are 2^k solutions, no matter what bipartite graph G we choose.

Furthermore, we will see in Lecture 5 that if G is an expander, this defines a code with good error-correcting properties. We do not know how to get rates of the form $1/2 - \epsilon$ with this strategy (which is about best for binary codes). We only know how to get codes of rate ϵ, which is not so good, but that are algorithmically *very* efficient.

A second family of graph-theoretic codes was introduced by Alon, Bruck, Naor, Naor, and Roth [1]. The goal here is the following: given a code C with small relative distance (e.g., the Justesen codes) boost it to a code C' with large relative distance. We assume for simplicity that C is a binary code. Then C' will have a large, but constant-size alphabet.

Take again a bipartite graph G, this time with the same number of vertices n on the left and right sides. Each vertex has degree d. A word $(c_1, \ldots, c_n) \in C$ is placed on the left vertices. This assigns to each vertex v on the right a d-bit string formed by taking the sequence of c_i for the vertices i to which v is connected. By taking $\{0,1\}^d$ as the output alphabet, this maps codewords in C (of length n) to strings of length n over this larger alphabet, which are the strings in C'.

Assume furthermore that G is an expander. Place on the left a codeword in C different from 0^n. Since C is a code, it should then have a constant fraction of 1s. When moving to the right, that fraction will be expanded by a factor of about d. This implies that the distance in C' is amplified by a constant factor w.r.t. that of C.

This contrasts with what we achieved with concatenated codes: Here, we go from small alphabets to large ones amplifying the distance by a constant factor. Forney showed how to move from large alphabets to small ones while loosing only a small fraction of the distance. Interleaving these two operations is at the root of Dinur's construction of PCP's, which we will survey in the final lectures.

3.4 ℓ-Wise Independence and ϵ-Biased Spaces

Randomness is considered to be an important resource in computer science. There are problems for which we have efficient randomized algorithms but no deterministic counterparts are known. Hence, an important step is to understand how much randomness is needed to solve a problem. One idea is to use a random string r of length s and then have an appropriate function G that can expand r into $G(r)$ of length m such that $G(r)$ appears "random enough" to an algorithm running in poly-time. Below, we give two notions for a set S of strings of length m to look random :

- *ℓ-wise independence*: S looks random if for every subset of ℓ co-ordinates in $[m]$, the projection of elements of S onto these ℓ co-ordinates generates the uniform distribution on $\{0,1\}^\ell$ i.e. $\Pr_{y \in_R S}[\, y_{i_1} \ldots y_{i_\ell} = b_1 \ldots b_\ell \,] = 2^{-\ell}$, for every ℓ-tuple $\{i_1, \ldots, i_\ell\} \in [m]^\ell$ and for each $b_i \in \{0,1\}$.
- *ϵ-biased sets*[1]: For every $T \subseteq [m]$, the projection of elements of S onto co-ordinates in T should generate a set that has $(1/2 \pm \epsilon)$-fraction of strings with parity 0 (or 1), i.e., for every $x \in \{0,1\}^m$ we have $\Pr_{y \in_R S}[\langle x, y\rangle = 0] \in [1/2 - \epsilon, 1/2 + \epsilon]$.

3.4.1 Codes and biased sets

Consider a linear code $C : \{0,1\}^m \to \{0,1\}^t$ such that the relative distance of C is between $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$. Let the t columns in the generating matrix M for C be $y_1, \ldots, y_t$. Then, for every $x \in \{0,1\}^m$, $\frac{1}{2} - \epsilon \leq \frac{wt(C(x))}{t} \leq \frac{1}{2} + \epsilon$. This implies that the set $S = \{y_1, \ldots, y_t\}$ is an ϵ-biased set of cardinality t.

Remark 1. *ϵ-biased sets imply a codespace of relative distance at least $1/2 - \epsilon$, but the reverse is not true in general.*

Remark 2. *The generator matrix for the code that concatenates AG codes with the Hadamard code produces an ϵ-biased set in the way explained above. Recall from Section 3.2 that this code achieves a rate of ϵ^3 and so the size of the biased set is m/ϵ^3.*

Naor and Naor [33] first posed the question on how small could t be such that there exists a set $S \in \{0,1\}^m$ of cardinality t, which is ϵ-biased. This number will be denoted by $t(m, \epsilon)$. Having seen the connection with linear codes, this question could be equivalently rephrased as: what is the highest rate one can achieve for a linear code such that every codeword has relative weight between $1/2 - \epsilon$ and $1/2 + \epsilon$? Using the ideas in the proof of the Gilbert-Varshamov bound one can show that there exists a linear code of rate $\Omega(\epsilon^2)$ having this property. We outline the argument: consider a random matrix M of dimension $m \times t$, where $t = \frac{cm}{\epsilon^2}$. Let the t column vectors of M be denoted by $y_1, \ldots, y_t$. Then, consider any $x \in \{0,1\}^m$. Let $X = \sum_{i=1}^{t} \langle x, y_i \rangle$, be the random variable representing the weight of the string xM. The probability that $\langle x, y_i\rangle$ has value 0 is $1/2$ for each $1 \leq i \leq t$ and so $E[X] = t/2$. Applying Chernoff bound, one concludes that $\Pr[\, |X - t/2| > \epsilon t \,] \leq 2\exp(-2\epsilon^2 t/3) = 2\exp(-2cm/3)$. Now by the union bound one sees

[1] In the literature, such sets are often called *ϵ-biased with respect to linear tests.*

the probability that there exists a x for which the weight of xM is not in $[(1/2-\epsilon)t,(1/2+\epsilon)t]$ is less than 1, for a suitable choice of c.

Alon, Goldreich, Håstad, and Peralta [2] show that the RS$\bigoplus$Hadamard code can be used to constructively generate biased sets of size m^2/ϵ^3.

3.4.2 ℓ-wise independence

We are interested here in looking for linear subspaces that are ℓ-wise independent. Let $v_1,\ldots,v_k$ be a set of k linearly independent vectors, each of dimension n. Let V be the $k\times n$ matrix whose ith row vector is v_i. The set *generated* by V is $\{y=xV \mid x\in\{0,1\}^k\}$.

Claim 1. *The set S generated by V is ℓ-wise independent iff every set of ℓ-columns of V is linearly independent.*

Proof. If there exists a set of ℓ columns $w_1,\ldots,w_\ell$ such that $w_1+\ldots+w_\ell=0$, then clearly S projected onto this set of ℓ-co-ordinates is not uniform.

Consider any submatrix U of ℓ-columns of V. The rank of U is ℓ, by the assumption on the columns of V. Thus, the null space of the linear map $\{0,1\}^k\to\{0,1\}^\ell$ induced by U has size $2^{k-\ell}$. Hence for every point $y\in\{0,1\}^\ell$ the set $I_y=\{x\in\{0,1\}^k \mid xU=y\}$ has cardinality $2^{k-\ell}$. □

Consider any code C with generator matrix G. The *dual* code of C, denoted by $C^\perp$, is the code whose parity check matrix is G^T.

Remark 3. *Claim 1 implies that set S generated by matrix V is ℓ-wise independent iff V^T is the parity check matrix of a code of distance $\ell+1$.*

3.4.3 Examples

- $\ell=2$, *pairwise independence:* Remark 3 suggests us to find a linear code of distance 3. Recall that the Hamming code achieves this distance mapping $n-\log_2(n+1)$ bits to n bits. It can be verified that the dual of this code is the Hadamard code that maps $\log_2(n+1)$ bits to n bits. Hence there exists a set $S\subset\{0,1\}^n$ with cardinality $n+1$ that is pairwise independent.

- $2t$*-wise independence:* BCH codes have a distance of $2t+1$. They map $n-t\log n$ bits to n bits and so their dual maps $t\log n$ bits to n bits. Thus we get sets of size n^t which are $2t$-wise independent.

At this point, Madhu says that we were skipping connections between coding theory, pseudo-random generators, and extractors. He suggests the following references for the interested reader and provides some very brief comments on some of them:

- N. Alon, L. Babai, and A. Itai, "A fast and simple randomized algorithm for the maximal independent set problem", *J. of Algorithms*, 7:567–583, 1986.

- L. Trevisan, "Extractors and pseudo-random generators", *J. ACM,* 48:860–879, 2001. Madhu says this is the most appropriate source to read for seeing the connection between list-decodability and extractors.

- A. Ta-Shma and D. Zuckerman, "Extractor Codes", *IEEE Transactions on Information Theory,* 50(12): 3015–3025, 2004 (also appeared in STOC 2001). Explores connection between list-decodability and extractors along the lines of Trevisan.

- A. Ta-Shma, D. Zuckerman and S. Safra, "Extractors from Reed-Muller codes", *J. Computer and System Sciences*, 2006, to appear (also appeared in FOCS 2001).

- R. Shaltiel and C. Umans, "Simple Extractors for all min-entropies and a new pseudo-random generator", *J. ACM*, 52(2):172-216, 2005 (also appeared in FOCS 2001).

Lecture 4. List Decoding Reed-Solomon Codes

Lecturer: Madhu Sudan Scribes: Eric Allender and François Lemieux

Recall that in the Reed-Solomon code the alphabet is a field $\mathbb{F}_q$ (where $q \geq n$ is a prime power) and a message $c_0 c_1 \cdots c_{k-1}$ is interpreted as the description of a polynomial $M(x) = \sum_{i=0}^{k-1} c_i x^i$. Given n distinct elements $\alpha_1, \cdots, \alpha_n \in \mathbb{F}_q$, the encoding is the vector $(M(\alpha_1), \ldots, M(\alpha_n))$. This defines a code with parameters $[n, k, n-(k-1)]_q$.

4.1 List Decoding

The problem of decoding the Reed-Solomon code can be stated as follows.

Problem: Given $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$, $y_1, \ldots, y_n \in \mathbb{F}_q$ and the parameters k and t.

Goal: Find a univariate polynomial p of degree $k-1$ such that $p(\alpha_i) = y_i$ for at least t values of $i \in \{1, 2, \ldots, n\}$.

Whenever the number of errors is $e = (n-k+1)/2$ and $t > n-e$ then the problem has at most one solution. However, for smaller values of t, there can be many polynomials p that satisfy the above conditions and the goal is now:

New Goal: Find all polynomials p of degree $k-1$ such that $p(\alpha_i) = y_i$ for at least t values of $i \in \{1, 2, \ldots, n\}$.

We say that a code is (e, l)-list decodable if whenever the number of errors in the received message $y = \langle y_1, \ldots, y_n \rangle$ is at most e, the list of codewords at distance e from y includes the original message and contains at most l elements.

Any (n, k, d)-code is $(n - \sqrt{n(n-d)}, poly(n))$ list-decodable (in [24] this fact is credited to [17]). Since Reed-Solomon codes are $[n, k, n-k+1]_q$ codes, we have $d = n-k+1$. Thus the bound from [17] says that there are only a polynomial number of codewords that can possibly map to a given received string if there are at most e errors, where $e \leq n - \sqrt{n(n-d)} = n - \sqrt{n(k+1)}$. The important question now becomes: is it possible to *construct* this list of codewords? It was shown by Guruswami and Sudan [24] that it *is* possible to construct such a list, even if the number of errors is (very) slightly *higher*: $e \leq \lceil n - \sqrt{kn} - 1 \rceil$. Thus this meets (even exceeds) the bound given by [17] but it is important to note that it is *not* known to be optimal. Guruswami's thesis discusses the fact that the bound in [17] and Johnson's bound are tight for *some* linear codes, but this bound is *not* known to be tight for Reed-Solomon Codes. (In particular, the statement in Section 2.5.2 of Trevisan's survey [48], which states that the Guruswami-Sudan list-decoding algorithm "is essentially best possible, because for smaller values of t the size of the list may be superpolynomial" appears to be inaccurate.)

The first polynomial time algorithm related to this problem is due to Peterson [37] who proved that BCH codes can be efficiently (uniquely) decoded when $t > (k+n)/2$. This result also applies to RS codes by a result of Gorenstein and Zierlet [18] who showed that BCH codes are roughly equivalent to RS codes. Faster algorithms (for unique decoding) with quadratic running time are also known, such that of Berlekamp and Massey [7, 32] and that of Berlekamp and Welch [49]. The first list-decoding algorithm was presented by Sudan [44], who gives a polynomial time algorithm for $t \geq 2\sqrt{kn}$. More recently Guruswami and Sudan [24] improved this result to $t \geq \sqrt{kn}$. Very recently Guruswami and Rudra [22] have presented list decoding algorithms for a different family of codes, that work even for $t \geq \sqrt{k}(1+\epsilon)$. In these notes, we will see the list decoding algorithm of [44] which assumes $t \geq 2\sqrt{kn}$.

Before we start presenting the algorithm of [44], it is worth discussing the factor of 2 improvement of [24]. In complexity theory, we aren't used to worrying about constant factors, and it might seem as if this is not a significant improvement. Thus, let's consider some typical parameters. It is often useful to consider the case where $k = 0.9n$ (that is, the original message is 90% as long as its encoding); this might be an acceptable overhead to pay for a good error correcting code. One can use a *unique* decoding algorithm to handle the case where the number of errors is roughly $0.05n$ (that is, if $t \geq 0.95n$). What advantages does list decoding provide for this choice of parameters? The algorithm of [44] allows us to handle only the case where $t \geq 2\sqrt{kn} = 2\sqrt{0.9}n > n$, which of course will *never* arise! In contrast, the algorithm of [24] can be used if $t \geq \sqrt{kn} = \sqrt{0.9}n \cong 0.9487n$. This means that it can be used to correct more errors than the unique decoding algorithm for this choice of parameters.

4.2 Algebraic Description of a Set of Points

We have a set of n points $(\alpha_1, y_1), \ldots (\alpha_n, y_n)$ and we want to find all polynomials of degree $k-1$ that agree on at least $t \geq 2\sqrt{kn}$ of them. Our first task will be to find a good algebraic representation of these points. For example, the polynomial p of degree n such that $p(\alpha_i) = y_i$ or the polynomial p' of degree n such that $p'(y_i) = \alpha_i$ are two different algebraic descriptions of the points.

However, these representations are not sufficient for our purpose. Instead, we will use algebraic curves in the form of bivariate polynomials $Q(x, y)$. We need Q to have small degree in x and y (say $\sqrt{n}$) and we also need that Q be 0 on the n points. Indeed, Q can have more than n zeros but this is irrelevant for the argument. The next lemma states that such a curve does exist, and can be found quickly.

Lemma 2. *There is a polynomial-time algorithm to construct a bivariate polynomial $Q(x, y)$ such that Q is not identically zero and for all $i \in [n] = \{1, 2, \ldots, n\}$ we have $Q(\alpha_i, y_i) = 0$ and the degree in x and in y is small (say $\sqrt{(n)}$).*

Proof. Our goal is to find an appropriate polynomial Q such that

$$Q(x, y) = \sum_{j=0,k=0}^{\sqrt{n}} q_{jk} x^j y^k.$$

That is, our goal is to find the coefficients q_{jk}. Note that the number of coefficients is equal to the number of pairs (j, k), which is $(\sqrt{n} + 1)^2 > n$. Hence, for each $i \in [n]$, we can think of the equation $Q(\alpha_i, y_i) = 0$ as a homogeneous equation with at least n unknowns. There are n such equations and one solution to the system can be found in polynomial time. $\square$

(In his lecture, Madhu points out that a similar argument also appears in [38].)

4.3 Bézout's Theorem (on the Plane)

Observe that the polynomial $y - M(x)$ is another polynomial that, like Q, evaluates to zero at many of the pairs (α_i, y_i) (where $M(x)$ is the degree $k - 1$ univariate polynomial representing the message). In this section, we show that $y - M(x)$ is a factor of $Q(x, y)$. (To do this, we make use of Bézout's Theorem.)

Recall that the degree of a term in a bivariate polynomial is the sum of the exponents of x and y. The degree of a bivariate polynomial is the maximum degree of its terms.

Bézout's theorem can be viewed as an analog (for multivariate polynomials) of the well-known fact that two *univariate* polynomials cannot agree in many places, unless they are identical. This fails spectacularly for multivariate polynomials. For instance, consider $Q_1(x, y) = x^2 - y^2$ and $Q_2(x, y) = x^2 + 2xy + y^2$. Since the polynomial $x + y$ divides both Q_1 and Q_2, we have that for *every* x, $Q_1(x, -x) = Q_2(x, -x) = 0$. That is, the distinct polynomials Q_1 and Q_2 agree on *infinitely many* points. Nonetheless, Bézout's theorem allows us to say something nontrivial about two multivariate polynomials that agree on many points.

Theorem 5. *Let Q_1 and Q_2 be two bivariate polynomials of degree D_1 and D_2 respectively. If Q_1 and Q_2 have more than $D_1 D_2$ common zeros, then Q_1 and Q_2 share a common factor.*

Now, for our application, let $Q_2(x, y) = y - M(x)$ and suppose that $S \subseteq [n]$ is the set of indices i such that $M(\alpha_i) = y_i$. Take any bivariate polynomial $Q(x, y)$ of degree $\sqrt{n}$ in x and degree $\sqrt{n}$ in y such that $Q(\alpha_i, y_i) = 0$ for all $i \in [n]$. Then, Q_2 and Q agree on $t = |S|$ points. Now, assume that $t > 2(k - 1)\sqrt{n} \geq deg(Q_2) \cdot \deg(Q)$. Then, Q_2 and Q have more than $\deg(Q_2) \cdot \deg(Q)$ zeros in common and thus share a common factor. Since Q_2 is irreducible, we conclude that $y - M(x)$ is a factor of Q.

4.4 Algorithm for List-Decoding RS

The discussion above gives the following algorithm that finds all polynomials with t agreements provided $t > 2(k - 1)\sqrt{n}$.

Step 1: Find Q of degree $\sqrt{n}$ in x and degree $\sqrt{n}$ in y such that $Q(\alpha_i, y_i) = 0$ for all $i \in [n]$.

Step 2: Factor Q and look for factors of the form $y - p(x)$ such that p has degree $k - 1$ and report all such p.

The most expensive part of this algorithm involves factoring polynomials. In general, factoring bivariate polynomials can be done in *probabilistic* polynomial time (where the running time depends also on the size of the underlying field). In our application, the field size is small (of size n), and the factoring algorithm can be derandomized in this case, to yield a deterministic algorithm where the dependence on the field size is insignificant.

The promised bound $t > 2\sqrt{kn}$ can be reached by finding a polynomial $Q(x, y)$ whose degree in x is $\sqrt{kn}$ and whose degree in y is $\sqrt{n/k}$. The proof of Lemma 1 can easily be adapted to these parameters. Moreover, a special case of Bézout's theorem states that if $Q_2 = y - p(x)$ has degree $k - 1$, then Q_2 is a factor of $Q_1 = Q$ whenever Q_1 and Q_2 have in common more than $\sqrt{nk} + (k - 1)\sqrt{n/k}$ zeros ([44], see also [48]).

Reed-Solomon codes can only be used with an alphabet $\Sigma = \mathbb{F}_q$ of size $q \geq n$ where n is the size of a codeword. What about binary alphabets? Good encoding over a binary alphabet can be achieved if we combine concatenation and Reed-Solomon codes. First we interpret each element of $\mathbb{F}_q$ as an element of $\{0, 1\}^m$ where $m \geq \log q$. A message $m_0 m_1 \cdots m_{k-1}$ which is seen as the polynomial $M(x) = \sum_{i=0}^{k-1} m_i x^i$ is first encoded as a vector $\langle M(\alpha_1), \ldots, M(\alpha_n)\rangle$ using RS encoding. This vector is then converted into a word $C \in \{0, 1\}^{mn}$ where each block of size m is transmitted using some good binary code having rate $1 - \epsilon$ (in particular, *not* using a Hadamard code, which has a very poor rate). This process is illustrated in Figure 4.7.

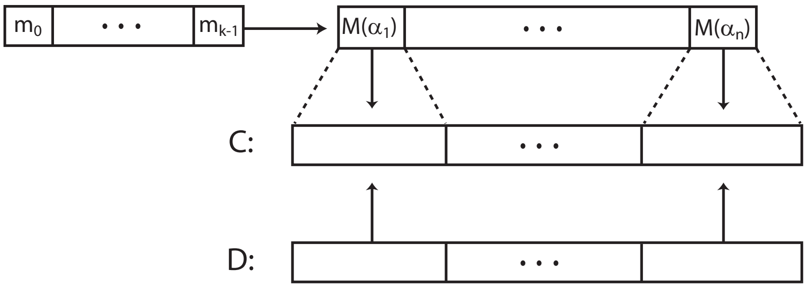

Figure 4.7: Decoding a concatenated code.

The list-decoding of the received sequence of bits D is done as follows. First, list-decode each block and pick at random a representative in each list. Then, interpret each representative as an element of the field $\mathbb{F}_q$ and finally use RS list-decoding. If the number of errors is smaller than $\frac{1}{2} - \tau$, then it is possible to construct a code of rate $poly(\tau)$ ([45, 20]) that can be list-decoded in polynomial time.

4.5 The Blum-Micali-Yao Generator and Hard-core Predicates

In this section we present a useful application of list-decodable codes to the theory of pseudorandom generators. Since our emphasis is on codes, we will not bother to give detailed definitions for some of the fundamental concepts that underlie this section: one-way functions, and pseudorandom generators. Instead, we will rely on an informal understanding of these notions.

A function f is *one-way* if it is easy to compute but hard on average to invert. That is, any efficient algorithm attempting to compute $f^{-1}(x)$ will fail for a large number of x in the range of f. (We will deal mainly with one-way permutations in these notes; that is, function f computes a permutation on the set of all n-bit strings.) A function g that maps, say, n input bits to outputs of length n^2 is a *pseudorandom generator* if the output "looks random". That is, given any statistical test T that is easy to compute, the probability that T accepts a string generated by g should be roughly the same as the probability that T accepts a string of length n^2 chosen uniformly at random; thus T cannot distinguish random strings from pseudorandom strings. Equivalently, given the first i bits of some n^2-bit pseudorandom string, no efficient algorithm should be able to predict what the next bit will be with probability significantly greater than $1/2$.

The basic idea relating pseudorandom generators to one-way permutations is generally credited to papers by Blum and Micali [9] and Yao [50]. The essential flavor of the Blum and Micali generator is illustrated in Figure 4.8. A string x of length n is presented as input. The first "pseudorandom" bit produced is simply x_0, the low-order bit of x (or perhaps the high-order bit, or some other specified bit). Then we compute $f(x)$, which we hope "looks random" relative to x (since by assumption there is no easy way to compute x given $f(x)$) and we output $f(x)_0$, the low-order bit of $f(x)$, as our second "pseudorandom bit". Then we compute $f(f(x))$ and output $f(f(x))_0$ as the third bit of output, and proceed until we have generated n^2 bits.

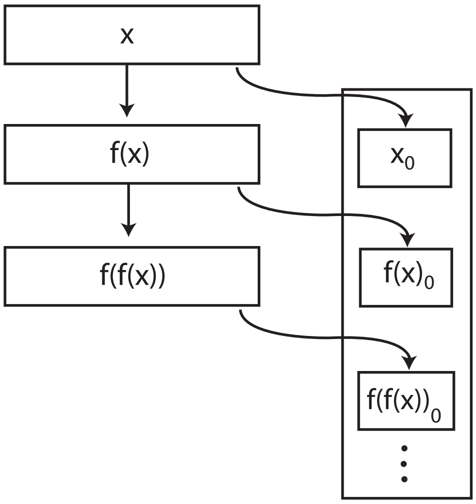

Figure 4.8: The Blum-Micali generator scheme.

Is this secure? Does the sequence produced in this way really "look random"? Blum and Micali showed that if one chooses f to be the permutation given by the Rivest-Shamir-Adleman (RSA) cryptosystem, then this scheme *is* secure, if we choose the *high-order* bit at each step as $x_0, f(x)_0, \ldots$. That is, given a test T that is able to distinguish the pseudorandom bits that are generated from truly random bits, one can use T to crack the RSA cryptosystem, which means that f was not a one-way function to begin with. On the other hand, if one uses the *low-order* bits (as actually described on the preceding paragraph), then this scheme is *not* secure.

This motivates the question: given a one-way permutation f, what bits $b_0, b_1, b_2, \ldots$ can one generate from the sequence $x, f(x), f(f(x)), \ldots$ so that the Blum-Micali-Yao construction *is* secure? One of the contributions of [9, 50] is a proof that it is sufficient to consider a generator that takes n bits of input and produces *just one more* bit of output. Namely, we consider the generator that takes x as input and produces output $f(x)b_0$, where b_0 is some easy-to-compute predicate of x; it can be shown that the sequence of bits $b_0, \ldots, b_{n^2}$ is pseudorandom if and only if every resource-bounded test T has roughly the same probability of accepting strings of the form $f(x)b_0$ as it does of accepting a string of length $n+1$ chosen uniformly at random.

Goldreich and Levin [16] considered this question, and phrased it more-or-less as follows: given f, is there a predicate Q_f so that the function that maps x to $f(x)Q_f(x)$ yields a pseudorandom generator? Ideally, one could hope to find a *single* construction that works for *any* f; that is, a predicate $Q(f, x)$ so that the sequence $Q(f, x), Q(f, f(x)), Q(f, f(f(x))), \ldots$ is pseudorandom if f is one-way. They succeeded (and the predicate Q_f that they construct is called a "hard-core predicate"), but it involves working with a slight modification of the given one-way permutation f. Rather than presenting the original construction of Goldreich and Levin (which in some sense is based on the Hadamard code) we present a related construction that is based directly on list-decodable codes.

Let C be a (binary) code mapping n-bit strings to N-bit strings. We want C to be list-decodable in polynomial time if there are no more than $(\frac{1}{2} - \epsilon/2)N$ errors, which can be achieved by a binary code with rate ϵ^8, so that $N = n/\epsilon^8 = n^{O(1)}$ if ϵ is as small as $1/n^{O(1)}$. (In the original construction of Goldreich and Levin based on the Hadamard code, N is exponential in n.) Let f be any permutation on n-bit strings. We consider a new permutation f' on the domain $\{0,1\}^n \times \{1, \ldots, N\}$ where $f'(x, j) = (f(x), j)$. (Note that f is one-way if and only if f' is one-way, too.) The predicate Q_f that we will use in generating our sequence of pseudorandom bits is $Q_f(x, j) = C(x)_j$ (where $C(x)_j$ is the j-th bit of the encoding of x). That is, our pseudorandom generator that extends input (x, j) by one bit is $(f'(x), C(x)_j) = (f(x), j, C(x)_j)$.

We need to show that if f is a one-way function, then this longer sequence of bits looks random to any resource-bounded test T. Note that since f' is a permutation, the only way that T can hope to distinguish the pseudorandom strings from the truly random strings is to detect the correlation between $f'(x, j)$ and $C(x)_j$ better than simply guessing. Thus assume that T can take $f'(x, j)$ as input and predict the value of $C(x)_j$ with probability greater than $\frac{1}{2} + \epsilon$. Then for at least an $\epsilon/2$ fraction of all strings x, T will predict the value of $C(x)_j$ (given $(f(x), j)$) for at least $\frac{1}{2} + (\epsilon/2)$ of the values j. For any of these "good" strings x, T can thus compute the value of $C_j(x)$ with fewer than $(\frac{1}{2} - \epsilon/2)N$ errors. Since C is list-decodable, we can thus compute a polynomial-size list of strings S such that $x \in S$, and by computing $f(y)$ for every string y in S, we can determine the correct value of x. That is, we have a deterministic algorithm to compute $f^{-1}(x)$ for a significant fraction of the n-bit inputs to f (namely, $2^n/n^{O(1)}$ inputs). Thus f is not one-way.

Lecture 5. Codes Constructed using Expanders

Lecturer: Madhu Sudan — Scribes: Navin Goyal and Michal Koucky

In this lecture we return to the codes constructed using expander graphs that we introduced in Lecture 3. These codes achieve good rate and relative distance, and in addition allow for linear time decoding.

5.1 Introduction

Let $B = \langle L, R, E\rangle$ be a bipartite graph with vertex sets $|L| = n$ and $|R| = m$.

Define the code $C_B \subseteq \{0,1\}^n$ associated with B as

$$C_B = \{(c_1, \ldots, c_n) \mid \forall v \in R, \sum_{(u,v)\in E} c_u = 0 \text{ (mod 2) } \}.$$

Thus, we can think of each vertex in R as specifying a linear constraint that the codeword has to satisfy. Clearly, every linear code over $\mathbb{F}_2$ can be represented in this way.

We say that the graph B is (c, d)-regular if all vertices in L have degree c and all vertices in R have degree d. For $\delta \in [0, 1]$ and $\gamma \in [0, c]$, a (c, d)-regular graph B is called a (γ, δ)-expander if $\forall S \subseteq L$ such that $|S| \leq \delta n$ we have

$$|\Gamma(S)| \geq \gamma |S|,$$

where $\Gamma(S) := \{v \in R \mid \exists u \in S \text{ such that } (u, v) \in E\}$, that is, $\Gamma(S)$ is the set of vertices in R that are adjacent to some vertex in S.

That such graphs exist can be proved using the probabilistic method but it was only recently that Capalbo et al. [10] gave explicit constructions of (c, d)-regular graphs with (γ, δ)-expansion. For the precise parameters in their construction we refer to the original paper.

Before we go on, let us give a bit of history about the graph-based codes. Gallager [14] first suggested using the adjacency matrix of a randomly chosen low-degree bipartite graph as the parity check matrix of an error-correcting code. He showed that such a code with good probability has a rate and minimum distance near the Gilbert-Varshamov bound and suggested a natural belief-propagation-like decoding procedure. However, he did not give a tight running time analysis of his algorithm.

Tanner [46] looked at the girth of the graph used to construct the code and showed that it controls the ability to correct errors, and that one could correct up to $O(\sqrt{n})$ errors.

5.2 The Sipser-Spielman Codes

Sipser and Spielman [42] looked at the expansion of the underlying graph and this enabled them to prove that good expansion gives codes with good distance and also allows rigorous guarantees about the performance of their decoding algorithm: their algorithm can correct up to linear number of errors. This is what we show next.

Theorem 6. *If $\gamma > c/2$ then $\delta(C_B) \geq \delta$, where $\delta(C_B)$ is the relative distance of code C_B and δ is the expansion parameter of B.*

We omit the proof of this theorem.

Theorem 7 (Sipser-Spielman [42]). *If B is a (c, d)-regular (γ, δ)-expander with $\gamma > 3c/4$ and $\delta > 0$ then there exists a linear time decoding algorithm for the code C_B correcting $\Omega(n)$ errors.*

Proof. Define

$$\Gamma_{\text{odd}}(S) := \{v \in R \mid \text{ number of } u \in S \text{ adjacent to } v \text{ is odd } \}.$$

The relevance of this notion comes from the fact that if for graph B it holds that for all $S \subseteq L$ and $|S| \leq \delta n$ we have $|\Gamma_{\text{odd}}(S)| > 0$, then C_B is a code of distance δn. This is because if $(c_1, \ldots, c_n)$ is a code word and $(y_1, \ldots, y_n)$ is a word and $S \subseteq L$ corresponds to the set of indices where $(c_1, \ldots, c_n)$ and $(y_1, \ldots, y_n)$ differ. Then $\Gamma_{\text{odd}}(S)$ corresponds to the set of those constraints that are not satisfied by $(y_1, \ldots, y_n)$.

However, it will be more convenient to work with a related notion:

$$\Gamma_{\text{unique}}(S) = \{v \in R \mid \text{ there is a unique vertex in } S \text{ adjacent to } v\}.$$

Clearly, $\Gamma_{\text{unique}}(S) \subseteq \Gamma_{\text{odd}}(S)$ for all $S \subseteq L$.

Lemma 3. *If B is a (c, d)-regular (γ, δ)-expander then $\forall S \subseteq L$ with $|S| \leq \delta n$ we have*

$$|\Gamma_{\textit{unique}}(S)| \geq (2\gamma - c)|S|.$$

Proof. Let $A := \Gamma_{\text{unique}}(S)$ and $B := \Gamma(S) \setminus A$. By the expansion condition for B we have

$$|A| + |B| \geq \gamma|S|.$$

Also,

$$c|S| \geq \text{number of edges touching } S \geq |A| + 2|B|.$$

The above two inequalities give $|A| \geq (2\gamma - c)|S|$, which is what we wanted to prove. ☐

We now give the *Sipser–Spielman decoding algorithm*:

- The input to the algorithm is $y_1, \ldots, y_n \in \{0, 1\}$ representing a corrupted codeword.
- While there is an $u \in L$ such that for a strict majority of neighbors v of u the constraint defined by v is unsatisfied, do flip(u).
- Output the current string and terminate.

For the analysis and proving the correctness of this algorithm we need the following three claims:

Claim 2. *If $(y_1, \ldots, y_n)$ is at relative distance $\tau < \delta/2$ from codeword $(c_1, \ldots, c_n)$ then:*

1. *The algorithm terminates in $c\tau n$ steps.*
2. *If $\gamma > 3c/4$ and S is any set of size $\leq \delta n$ then $|\Gamma_{\textit{unique}}(S)| > c|S|/2$.*
3. *The algorithm terminates with all parity conditions satisfied and outputs $(c_1, \ldots, c_n)$.*

Proof. Claim 1. We just need to observe that in each step of the algorithm the number of unsatisfied constraints decreases, and there are at most $c\tau n$ such constraints. Note that this claim does not depend on the expansion properties of B.

Claim 2. This follows readily from Lemma 3.

Claim 3. For some time step of the algorithm let U be the set of vertices in R that correspond to unsatisfied constraints; and let S be the set of indices where $(y_1, \ldots, y_n)$ differs from $(c_1, \ldots, c_n)$. By Claim 1, $|U| \leq c\tau n$.

If $|S| < \delta n$ then we have by Claim 2 that $|\Gamma_{\text{unique}}(S)| > c|S|/2$ and hence there is a vertex u in L whose strict majority of neighbors are unsatisfied and its value can be flipped to reduce the number of unsatisfied constraints.

We now show that the other case $|S| \geq \delta n$ cannot happen; in other words, it cannot be that at any intermediate stage the word the algorithm has in that stage is far away from $(c_1, \ldots, c_n)$. If it does and we have $|S| = \delta n$ then by Lemma 3 we have $|\Gamma_{\text{unique}}(S)| \geq c\delta n/2 > c\tau n$, which is not possible as with each step the number of unsatisfied constraints only decreases and initially it was $\leq c\tau n$. ☐

This completes the proof of Theorem 7. ☐

Sipser and Spielman's code [42] as described above was not constructive, as explicit expanders with the parameters above were not available at the time. As mentioned already, the work of Capalbo et al. [10] changed this situation. Sipser and Spielman [42] gave alternative algorithms that worked with weaker parameter expanders.

We should mention *Turbo Codes* here, a recently discovered class of codes that has been useful in practice. These codes are good for concrete settings but no analysis of their performance is known. In contrast, Sipser-Spielman codes do allow analysis but are not very useful in practical situations as δ is too small a constant.

5.3 Linear-time Encoding and Decoding

On the other hand, Sipser and Spielman did not say how to do the encoding for their codes. This was done by Spielman [43].

Spielman generalized the idea of expander codes to obtain codes that can be encoded and decoded in linear time. His construction goes as follows. Consider a good bipartite expander (L, R, E) with $|L| = 2|R| = k$. We will label vertices in L by our message m of length k. Each vertex in R will be labeled by the parity of the labels of its adjacent vertices in L. Instead of requiring that all these parities are zero we will store the labels of R as a part c of the codeword. Given a corrupted codeword if we know the part c correctly we can error-correct the message m as in the case of Sipser–Spielman codes.

Thus we only need to make sure that we know the part c correctly, i.e., that we can correct possible errors in that part of the codeword. Since the part c is $1/2$ the size of m we can use recursively the same type of error correction for c. To make this really work one needs to provide also an extra parity check sum c' of all the additional information. Figure 5.9 shows the scheme of the construction of the codeword.

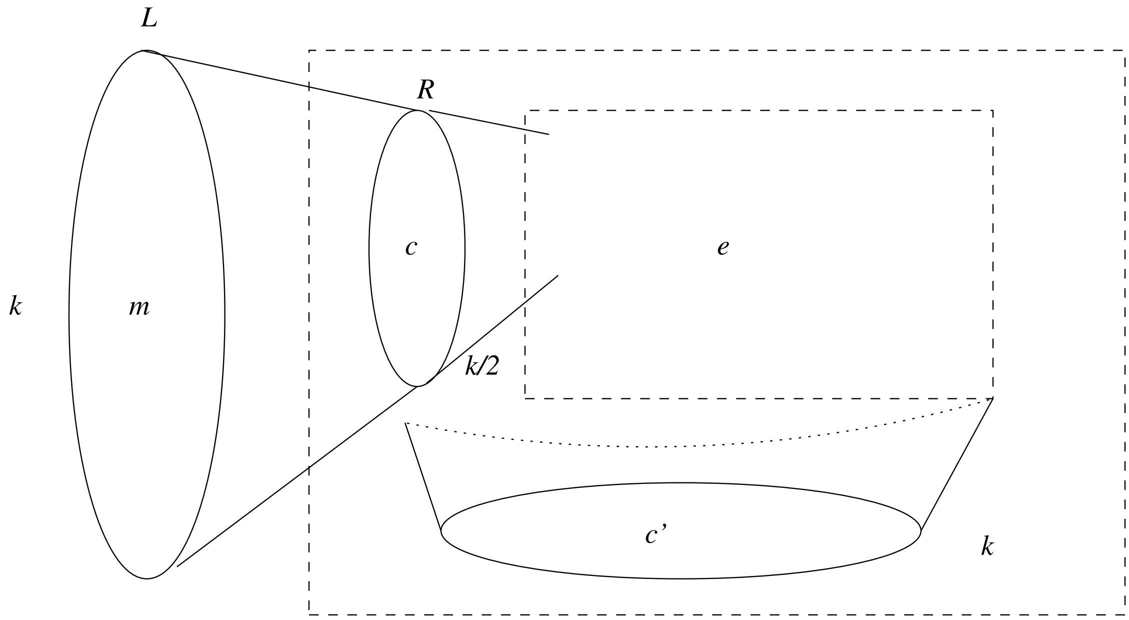

Figure 5.9: Spielman's code

Decoding proceeds by reducing the number of errors in the part e and c using information from c', then error correcting e, followed by recovering c using e and finally recovering m. If the graph used is $(c, 2c)$-bipartite $(7c/8, \delta)$-expander then the code is able to correct constant fraction of errors in the codeword.

Spielman codes have linear time encoding and decoding and are able to correct a constant fraction of errors. The fraction is rather small though. Indeed it is an open problem to construct a code with rate $(1 + \epsilon)$-close to the Shannon limit, small error probability, and encoding and decoding in time polynomial in n and $1/\epsilon^2$.

Lecture 6. Local List-Decoding of Reed-Muller Codes
Lecturer: Madhu Sudan   Scribes: Valentine Kabanets and Peter Winkler

6.1 List-Decoding of Reed-Muller Codes

This evening's lecture will be devoted to local list-decoding of Reed-Muller (RM) codes, and applications. Recall that we are given $f : \mathbb{F}_q^m \to \mathbb{F}_q$ as an oracle, and we want an oracle for a polynomial p of degree d agreeing with f in a fraction at least τ of the inputs. We also want $p(z) = \alpha$ for certain z and α so that this choice of (z, α) uniquely specifies p.

As before, our approach is to use a small-dimensional subspace of $\mathbb{F}_q^m$, e.g. the line through x and z.

We describe the algorithm $A_{z,\alpha}$ as follows. Given x, let $\ell := \{tx + (1-t)z\}$. Using the Reed-Solomon (RS) list-decoding method, we obtain a list $g_1, \ldots, g_L$ of univariate polynomials of degree d such that $\text{agree}(g_i, f|_\ell) \geq \tau/2$, for every i. If there is a *unique* j such that $g_j(z) = \alpha$, we output $g_j(x)$; otherwise, we indicate "error."

Note that as we vary over αs, we get all polynomials p that have agreement at least τ with f.

We will argue that for random x and for a good random choice of z, the algorithm described above works. Then we will use the unique decoding algorithm for RS codes from Lecture 4.

Claim 3. $\Pr($*f and p have agreement* $< \tau/2$ *on* $\ell) \leq O\left(\frac{1}{\tau q}\right)$, *where the probability is taken over uniformly random x and z.*

Proof. Use Chebyshev's inequality. □

Claim 4. $\Pr($*there exist* g_i, g_j *such that* $g_i(z) = g_j(z)) \leq \binom{L}{2} \cdot \frac{d}{q} \leq \mathrm{O}\left(\frac{d}{\tau^2 q}\right)$, *where the probability is again over uniformly random x and z.*

Proof. For the proof, consider an equivalent random experiment: we first pick a random x and a random line ℓ through x, and only then pick a random z on the line. Note that z can be chosen at the very last step of the algorithm after all univariate polynomials $g_1, \ldots, g_L$ are already determined. Since each of these polynomials is of degree at most d, the probability that a pair of them agree at a random point z is at most d/q by the Schwarz-Zippel lemma. By the union bound, we get the first inequality of the claim. To get the right inequality, note that if τ is large enough, then $L \leq 4/\tau$ (as argues in the last lecture). □

Now, if we consider the algorithm $A_{z,p(z)}$, we have

$$\Pr_{z,x}(A_{z,p(z)}(x) \neq p(x)) = \beta \to 0.$$

And by a simple averaging argument, we get

$$\Pr_z \left(\Pr_x (A_{z,p(x)}(x) \neq p(x)) \geq \sqrt{\beta} \right) \leq \sqrt{\beta}.$$

So, pick z at random and fix it.

All this works provided d/q is sufficiently small.

Let us draw some conclusions. If $d = \log^2 k$, then the number of coefficients is at least $(\frac{d}{m})^m$ which will be equal to k if $m = \log k / \log\log k$ (ignoring constant factors). Now pick $q = \log^3 k$. We then get that $\frac{d}{k} \to 0$, and

$$n = q^m = \left(\log^3 k\right)^{\log k/\log\log k} = k^3.$$

Our algorithm computes $p(x)$ in time polynomial in m, d and q which means polylog in k. Thus:

Theorem 8. *Given k and ϵ there is a code C mapping k elements to $(k/\epsilon)^{O(1)}$ elements, with alphabet size polynomial in $(\log k)/\epsilon$, which we can list-decode from ϵ-fraction agreement in time polynomial in $(\log k)/\epsilon$.*

Note that this kind of extremely efficient "local" decoding — a symbol in a message can be computed using much less than k queries — is a feature of RM codes. One can say that RM codes have a lot of "local" redundancy: a random line through x is "local" (it is a tiny portion of the codeword) and yet is has enough redundancy to recover the value of the codeword at x.

Turning to the binary case, we get

Theorem 9. *Given k and ϵ there is a binary code C mapping k bits to poly(k/ϵ) bits, with a local list-decoding algorithm correcting $\frac{1}{2} - \epsilon$ fraction of errors in time polynomial in $\log k/\epsilon$.*

Note: we cannot expect to correct from $\frac{1}{2}$ fraction of errors. Just consider a string of all zeros. It $\frac{1}{2}$-agrees with almost every string.

Proof. Use the first theorem, and encode every element in binary using a good code having $\delta > \frac{1}{2} - \epsilon$. In other words, use our concatenation technique. □

6.2 History

Hadamard decoding is from the work by Blum, Luby, and Rubinfeld [8]; Beaver and Feigenbaum [4] decoded Reed-Muller codes of degree d with error fraction $\mathrm{O}(1/d)$. Lipton [30] generalized their result to any polynomial, and noticed for example that you can use this technique to upgrade an algorithm that computes the permanent of a matrix and works "on average" (that is, on most of the q^{n^2} inputs). This works because the permanent is merely a polynomial of degree n.

Gemmell, Lipton, Rubinfeld, Sudan, and Wigderson [15] moved up from error fraction $\mathrm{O}(1/d)$ to 1/4; Sudan, Trevisan, and Vadhan [45] list-decoded Reed-Muller codes. In the 'eighties, Goldreich and Levin [16] gave an algorithm to *locally* list-decode Hadamard codes $C: \{0,1\}^n \to$

$\{0,1\}^{2^n}$ in time polynomial in n, up to error $\frac{1}{2}-\epsilon$. This was further pursued by Babai, Fortnow, Levin, and Szegedy [3].

Rackoff's proof of Goldreich-Levin follows the idea that when the field is small (so that a line or plane contains too few points to work with), one should instead use a low-dimensional subspace. One wants the number of candidates to be less than $1/\epsilon^2$; the dimension ℓ of the chosen subspace should be such that $\epsilon^2 q^\ell$ is big. For the proof of the Goldreich-Levin result, check, e.g., the survey by Trevisan [48].

6.3 A Lower Bound

A locally decodable code C: k bits $\to$ n bits is said to be "t-locally decodable" if there is an algorithm which, with ϵ-fraction corruption of a codeword, finds a specified message bit making only t oracle queries. We have seen a $k \to k^3$ case with $t = \log^3 k$. Katz and Trevisan [29] proved an embarrassingly weak lower bound: a t-locally decodable code must have $n > k^{1+1/t}$ (perhaps c/t for some constant c).

6.4 An Application: Worst-case to Average-case Hardness Amplification

Let's do an application of our binary local list decoding theorem to the problem of obtaining average-case hardness from worst-case hardness. In other words, given some $f : \{0,1\}^k \to \{0,1\}$ which is hard to compute (worst-case), can we get some $\tilde{f}$ which is hard on *random* inputs?

We say that $\tilde{f} : \{0,1\} \to \{0,1\}$ is (s,α)-*hard* if any circuit A of size s has $\Pr_x(Z(x) \neq f(x)) \geq \alpha$. Thus, "worst-case hard" means (poly$(n), 2^{-n}$)-hard. We want to amplify this to (poly$(n), \frac{1}{2}-\epsilon$)-hard.

Some motivation was supplied by Nisan and Wigderson [34], who proved that if there is an $\tilde{f}$ computable in deterministic time 2^{100n}, but $(2^{n/100}, \frac{1}{2} - 2^{-n/100})$-hard, then BPP=P.

Impagliazzo and Wigderson [26] showed that if there exists an $f : \{0,1\}^k \to \{0,1\}$ in $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ that is $(s, 2^{-k})$-hard, then there is an $\tilde{f} : \{0,1\}^n \to \{0,1\}$, also in E, which is $(s', \frac{1}{2}-\epsilon)$-hard.

To prove this, think of f as a string of length 2^k. Use our binary local list-decoding theorem to map $K = 2^k$ bits to $N = 2^n$ bits, and look at $C(f)$.

Claim 5. *If there is a circuit A of size s' computing $C(f)_x$ correctly for a fraction $\frac{1}{2}+\epsilon$ of the x's, then a list-decoding algorithm using A as its oracle computes f_y correctly for every y with high probability; the circuit for this algorithm has size polynomial in s', k and $1/\epsilon$.*

Lecture 7. Local Testability of Hadamard Codes	
Lecturer: Madhu Sudan	Scribes: Toniann Pitassi, Pierre McKenzie

7.1 Locally Testable Codes

Until now we have been talking about locally decodable codes. Given some string r with a promise that r is close to some codeword c, the goal is to show that with oracle access to r, we can obtain oracle access to c with a small number of queries.

Today we will talk about checking the promise above, i.e., about the decision problem where we are given a string r and we want to know if it is close to some codeword or not.

Definition 2. *Let C be a code of relative distance δ. Then C is (t, δ, ϵ)-locally testable if there exists a test T that makes t queries to r, and if $r \in C$, then T accepts with probability 1, and if r is $\delta/2$ far from every codeword in C, then T rejects with probability at least ϵ.*

Why are we interested in locally testable codes that are not necessarily locally decodable? First, constructions of nearly all PCPs use locally testable codes (LTC's). Secondly, both Hadamard codes and Reed-Muller codes are LTC's. While they are locally decodable, they take a polylogarithmic number of queries. In contrast, testing is much more efficient as it requires only a constant number of queries. Indeed the best LTC's known today map k bits to $k(\log k)^c$ bits and they are $(O(1), 1/2 - \epsilon, 1/2)$-locally testable.

7.2 Local Testability of Hadamard Codes

Today we will see that Hadamard codes are locally testable. The goal is as follows. Given $f : \{0,1\}^n \to \{0,1\}$ as an oracle, we want to test if f is a Hadamard codeword. I.e., we want to know if there exists $\alpha \in \{0,1\}^n$ such that for all $x \in \{0,1\}^n$, $f(x) = \langle \alpha, x \rangle$.

Given a LDC there is a natural test: pick a random point, try to decode, and accept if you succeed. The BLR test by Blum, Luby, and Rubinfeld [8] is as follows. Given oracle access to f, pick x, y at random from $\{0,1\}^n$ and accept if and only if $f(x) + f(y) = f(x + y)$. Clearly the BLR test is a 3-query test. Note that if f is a codeword, then for all x, y $f(x) + f(y) = f(x+y)$, so it follows that the BLR test accepts with probability 1 whenever f is a codeword. It is left to show that if f is far from every codeword, then the BLR test rejects with relatively high probability. This amounts to showing something like

if $\mathrm{Pr}_{x,y}[f(x) + f(y) \neq f(x + y)] < \epsilon$ then
there exists an α such that $Pr_x[f(x) \neq \langle \alpha, x \rangle] < 1/4$

or in other words

if $\Pr_{x,y}$[the BLR test accepts f] $\geq 1-\epsilon$ then
some codeword has relative distance to f smaller than $1/4$.

To get a feel for the BLR test, consider the groups G and H, and suppose that $\phi : G \to H$ is τ-far from $f : G \to H$. I.e., $Pr_x[\phi(x) \neq f(x)] = \tau$. Note that the latter probability is the relative distance from ϕ to f. Consider the following events, whose probabilities are deduced from the definition of τ and the fact that G is a group:

- $Pr[\phi(x) = f(x)] = 1 - \tau$
- $Pr[\phi(y) = f(y)] = 1 - \tau$
- $Pr[\phi(x + y) \neq f(x + y)] = \tau$.

Since these three events are pairwise independent, it follows[2] that the probability that all three events hold is greater than or equal to $\tau(1 - 2\tau)$. Unfortunately the bound is useless when τ is close to $1/2$. This comes as no big surprise: we cannot expect a very far ϕ (that could even be a homomorphism) to be able to spot the rejection of f for us.

In the Hadamard setting, the codewords consist of the homomorphisms from Z_2^n to Z_2. The following theorem extends the idea to abelian groups. We think of the codewords as the homomorphisms from G to H. The theorem says that if an arbitrary f passes the BLR test with probability greater than $7/9$, then f is close to some codeword ϕ.

Theorem 10 (Blum, Luby, Rubinfeld [8]). *Let G and H be finite abelian groups. If $f : G \to H$ satisfies $\Pr_{x,y}[f(x) + f(y) \neq f(x + y)] = \epsilon < 2/9$, then there exists a homomorphism $\phi : G \to H$ such that $\Pr_{x,y}[f(x) \neq \phi(x)] \leq 2\epsilon$.*

The theorem above was shown to be tight by Ben-Or, Coppersmith, Luby, and Rubinfeld [6]. In particular, they exhibited the function $f : Z_{3^m} \to Z_{3^{m-1}}$ where $f(x) = \lfloor (x + 1)/3 \rfloor$. This function passes the test with probability exactly 7/9, and every homomorphism agrees with f at exactly three places! Thus as soon as ϵ in the statement of the theorem is allowed to reach 2/9, there is an f whose relative distance to every codeword ϕ is exponentially close to 1, meaning that every codeword is very far from f. Hence the constant 2/9 in the statement cannot be improved.

Today we will prove a special case of the above theorem. For convenience we map the output 1 to -1, and 0 to 1, so $f : \{0, 1\}^n \to \{-1, 1\}$. We want to compute the probability, over random x and y, that $f(x)f(y)f(x + y) = 1$. A closely related value is the expectation:

$$E_{x,y}[f(x)f(y)f(x + y)] \;=\; \Pr[\text{test accepts}] - \Pr[\text{test rejects}] = 1 - 2\Pr[\text{test rejects}].$$

The next theorem immediately implies that the Hadamard code is $(3, \delta, \delta/2)$-locally testable:

Theorem 11 (Bellare, Coppersmith, Hastad, Kiwi, Sudan [5]). *If $f : \{0, 1\}^n \to \{1, -1\}$, then there exists an α such that $\Pr_x[f(x) \neq L_\alpha(x)] \leq \Pr[\text{test rejects}]$ where $L_\alpha(x) = (-1)^{\langle \alpha, x\rangle}$.*

[2] The Addendum in Section 7.3 explains this.

We will now prove this theorem. Let us first see a few examples to get some intuition.

Consider testing $f_1(x)f_2(y)f_3(x+y) = 1$. The first example is when $f_1 = f_2 = f_3 = L_\alpha$. Then $E[f_1(x)f_2(y)f_3(x+y)] = 1$ and thus the probability that the test rejects is 0.

Another useful example to consider is when $f_1 = L_\alpha$, $f_2 = L_\beta$ and $f_3 = L_\gamma$, where now α, β, and γ are not all equal. Suppose that $\alpha \neq \gamma$. Then we have

$$E_{x,y}[L_\alpha(x)L_\beta(y)L_\gamma(x+y)] = E_{x,y}[L_\alpha(x)L_\gamma(x)L_\beta(y)L_\gamma(y)]$$

where the last equality holds because $L_\gamma(x+y) = L_\gamma(x)L_\gamma(y)$. Now this equals

$$E_x[L_\alpha(x)L_\gamma(x)]E_y[L_\beta(y)L_\gamma(y)].$$

This equals 0 since $E_x[L_\alpha(x)L_\gamma(x)] = 0$. Thus the probability that the test rejects in this case is 1/2.

We will use harmonic analysis to analyze the general case. Let f, g be functions from $\{0,1\}^n$ to the real numbers. Define $\langle f, g\rangle = E_x[f(x)g(x)]$. Then $\langle L_\alpha, L_\alpha\rangle = 1$, and for $\alpha \neq \beta$, $\langle L_\alpha, L_\beta\rangle = 0$. Thus the L_α's form an orthonormal basis for the functions f from $\{0,1\}^n$ to the real numbers. Thus for any f, we can write $f(x) = \sum_\alpha \hat{f}_\alpha L_\alpha(x)$, where $\hat{f}_\alpha = \langle f, L_\alpha\rangle$.

Recall that Parseval's identity states that $\sum \hat{f}_\alpha^2 = \langle f, f\rangle = 1$.

We will write $E_{x,y}[f(x)f(y)f(x+y)]$ via its Fourier expansion.

$$\begin{aligned}
E_{x,y}[f(x)f(y)f(x+y)] &= E_{x,y}\left[\sum_\alpha \hat{f}_\alpha L_\alpha(x) \cdot \sum_\beta \hat{f}_\beta L_\beta(y) \cdot \sum_\gamma \hat{f}_\gamma L_\gamma(x+y)\right] \\
&= \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma E_{x,y}[L_\alpha(x)L_\beta(y)L_\gamma(x+y)] \\
&= \sum_\alpha \hat{f}_\alpha^3 \\
&\leq \sum_\alpha (\max_{\alpha_0}\{\hat{f}_{\alpha_0}\})(\hat{f}_\alpha)^2 \\
&= \max_{\alpha_0}\{\hat{f}_{\alpha_0}\}\,.
\end{aligned}$$

Thus we know that A holds, where A is the following statement:

$$\Pr[\text{test rejects}] = (1 - E_{x,y}[f(x)f(y)f(x+y)])/2 \geq (1 - \max_{\alpha_0}\{\hat{f}_{\alpha_0}\})/2.$$

Secondly, by definition of the Fourier coefficients, we have the following (call it B):

$$\hat{f}_\alpha = \langle f, L_\alpha\rangle = E_x[f(x)L_\alpha(x)] = 1 - 2\Pr_x[f(x) \neq L_\alpha(x)].$$

Combining A and B we get

$$\Pr[\text{test rejects}] \geq [1 - (1 - 2\Pr[f(x) \neq L_{\alpha_0}(x)])]/2 = \Pr_x[f(x) \neq L_{\alpha_0}(x)]$$

as desired, proving the theorem.

We will later consider nonlinear codes. Instead of Hadamard, i.e.,

$$\alpha \longrightarrow \{L(\alpha) \mid L \text{ is a linear function in } n \text{ variables}\}$$

encoding α as the 2^n-bit result of evaluating every linear function at α, we will consider

$$\alpha \longrightarrow \{Q(\alpha) \mid Q \text{ is a degree-2 polynomial in } n \text{ variables}\},$$

thus encoding α as the 2^{n^2}-bit result of evaluating every quadratic function. We will show in the next lecture that such an encoding is locally testable as well. This will have further complexity-theoretic consequences.

7.3 Addendum

Claim 6. *Let A, B and C be pairwise independent events such that $\Pr(A) = \Pr(B) = 1 - \tau$ and $\Pr(C) = \tau$. Then $\Pr(A \wedge B \wedge C) \geq \tau - 2\tau^2$.*

Proof. One way to see this is to define 8 variables corresponding to the probabilities of the 8 possible combinations of occurrences of A, B and C. One can then set up 7 equations on these variables, such as

$$\Pr(A \wedge B \wedge C) + \Pr(A \wedge B \wedge \overline{C}) = (1 - \tau)(1 - \tau),$$

which expresses the pairwise independence of A and B, or

$$\Pr(A \wedge B \wedge C) + \Pr(A \wedge B \wedge \overline{C}) + \Pr(A \wedge \overline{B} \wedge C) + \Pr(A \wedge \overline{B} \wedge \overline{C}) = \Pr(A) = (1 - \tau).$$

With an obvious convention for naming variables, Mathematica handles these 7 equations as depicted in Figure 7.10. In particular, one reads off the following equality:

$$\Pr(\overline{A} \wedge \overline{B} \wedge C) = \Pr(A \wedge B \wedge C) - \tau + 2\tau^2.$$

Since the lhs is nonnegative, it follows that $\Pr(A \wedge B \wedge C) \geq \tau - 2\tau^2$ as claimed.

Madhu provided the following alternate proof after the workshop:

$$\begin{aligned}
\Pr(A \wedge B \wedge C) &= \Pr(A \wedge B|C) \cdot \Pr(C) \\
&= \big(1 - \Pr(\overline{A} \vee \overline{B}|C)\big) \cdot \Pr(C) \\
&\geq \big(1 - \Pr(\overline{A}|C) - \Pr(\overline{B}|C)\big) \cdot \Pr(C) \\
&= \Pr(C) - \Pr(\overline{A} \wedge C) - \Pr(\overline{B} \wedge C) \\
&= \tau - \tau^2 - \tau^2,
\end{aligned}$$

where the inequality above is the union bound, and the last equalities use pairwise independence of A and C and B and C respectively. (In particular, we don't need A and B to be independent for this.)

$\square$

```
In[74]:= Solve[{ 1 - t == A,
           1 - t == B,
           ABC + ABc == A B,
           ABC + AbC == A t ,
           ABC + aBC == B t ,
           ABc + ABC + AbC + Abc == A,
           ABc + ABC + aBC + aBc == B,
           ABC + aBC + AbC + abC == t,
           ABC + aBC + AbC + ABc + abC + aBc + Abc + abc == 1
           }, {A, B, ABC, abc, ABc, AbC, aBC, Abc, aBc, abC}]

        Solve::svars : Equations may not give solutions for all "solve" variables. More…

Out[74]= {{abc → -ABC + t - t^2, Abc → ABC, aBc → ABC, abC → ABC - t + 2 t^2,
           ABc → 1 - ABC - 2 t + t^2, AbC → -ABC + t - t^2, aBC → -ABC + t - t^2, A → 1 - t, B → 1 - t}}
```

Figure 7.10: Mathematica program to verify Claim 6.

Lecture 8. Probabilistically Checkable Proofs

Lecturer: Madhu Sudan | Scribes: Pavel Pudlák, Mike Saks

8.1 NP and PCPs

We can view NP as defined by proof systems as follows:

$$L \in NP \text{ provided that there exists a polynomial time verifier } V$$

such that

1. $x \in L$ if and only if there exists y, $|y| = |x|^{O(1)}$ s.t. $V(x, y)$ accepts;
2. $x \notin L$ if and only if for all y $V(x, y)$ rejects.

Definition 3. *$L \in PCP[r, q]$ provided that there exists a* probabilistic *polynomial time verifier V such that*

1. *$x \in L$ if and only if there exists y, $|y| = |x|^{O(1)}$ s.t. $V(x, y)$ accepts with probability 1;*
2. *$x \notin L$ if and only if for all y $V(x, y)$ rejects with probability $\geq 1/2$.*

V is allowed to read at most q bits from the proof and use $r(|x|)$ random bits.

Theorem 12 (The PCP Theorem)**.** $NP = PCP[O(\log n), O(1)]$

Today we only prove a weaker result:

$$NP \subseteq PCP[poly(n), O(1)].$$

One can prove that the later class is in fact equal to $NEXP$.

We need only to show that such a proof system exists for one NP-complete problem. We shall use the problem of the solvability of quadratic equations in the two-element field.

8.2 A PCP for Systems of Quadratic Equations over $\mathbb{F}_2$

We study the following problem:

Input: a collection of quadratic polynomials $P_1, \ldots, P_m$ in variables $x_1, \ldots, x_n$.
Goal: Is there an assignment $a_1, \ldots, a_n \in \mathbb{F}_2$ to the variables such that $\forall j$ $P_j(\bar{a}) = 0$?

Notice that if all of the P_j were linear, then we could use the following proof system: The prover provides a sequence of length 2^n which is supposed to be the Hadamard encoding of some $a_1, \ldots, a_n$ that satisfies the constraints. Thus the proof π is indexed by linear functions L and the value of $\pi[L]$ is supposed to be $L(a_1, \ldots, a_n)$. The verification of proposed proof π is the test for membership in the Hadamard code given in Lecture 7: Choose two linear functions L_1 and L_2 at random and accept if and only if $\pi[L_1] + \pi[L_2] = \pi[L_1 + L_2]$.

For the given problem where the P_j can be quadratic the prover is supposed to choose a satisfying assignment $\bar{a} = a_1, \ldots, a_n$ and provide the string

$$Quad(\bar{a}) = \{Q(\bar{a})\}_Q,$$

where Q ranges over all degree 2 polynomials

$$Q(\bar{x}) = q_0 + \sum q_i x_i + \sum q_{ij} x_i x_j$$

over F_2. The length of this proof is $2^{1+n+\binom{n}{2}}$, thus $O(n^2)$ bits suffice to specify a constant number of positions in the string.

We write $\pi[Q]$ for the value of a proof π on Q. Call the proof *honest* if $\pi = Quad(\bar{a})$ for a satisfying assignment $\bar{a}$. We want to construct a small number of tests, each checking only a small number of bits that satisfies:

1. for any honest proof π, all of the tests accept,
2. If there is no satisfying assignment, then for some constant $\gamma > 0$, for any proof π, the probability that all tests accept is at most $1 - \gamma$.

By repeating each test a constant number of times, we can replace γ by $1/2$ as needed for the definition of a PCP.

Call a proof *coherent* if it is equal to $Quad(\bar{a})$ for some (not necessarily satisfying) assignment $\bar{a}$. Assuming that the proof is coherent, we can use the following test to verify that it is honest. Pick random $r_1, \ldots, r_m \in \{0, 1\}$ and check that

$$\pi[\sum r_j P_j] = 0.$$

This test always accepts an honest proof. On the other hand, if $P(\bar{a}) \neq 0$ for some j, then

$$Pr_{\bar{r}}[\sum r_j P_j(\bar{a}) \neq 0] \geq \frac{1}{2}.$$

Notice that if π is corrupted (ie., $\pi[Q] \neq Q(\bar{a})$) at a few places, then this test may be fooled into accepting with higher probability.

Next let's weaken the assumption that π is coherent to the assumption that π is "ε-close" to being coherent, which means for some $\bar{a}$, $Pr_Q[\pi[Q] \neq Q(\bar{a})] \leq \varepsilon$.

We use the following test:

Test 1. Pick $r_1, \ldots, r_m \in \{0, 1\}$ at random and pick Q at random and verify that

$$\pi[Q + \sum r_j P_j(\bar{a})] - \pi[Q] = 0.$$

If there exists j such that $P_j(\bar{a}) \neq 0$, then w.p. $\geq \frac{1}{2}(1 - 2\varepsilon)$ Verifier rejects.

So if π is ε-close to being coherent but there is no satisfying assignment, this test will detect this.

Next we want a test that will accept π if it is coherent, but reject it with positive probability if it is not ϵ-close to being coherent. This is accomplished by the combination of three tests.

A proof is said to be *linear*, if $\pi[Q_1] + \pi[Q_2] = \pi[Q_1 + Q_2]$ for all quadratic polynomials Q_1, Q_2. Notice that any coherent proof is linear.

Test 2. Pick Q_1 and Q_2 at random and verify

$$\pi[Q_1] + \pi[Q_2] = \pi[Q_1 + Q_2].$$

This is a linearity test. As we saw previously, if the proof is not ε-close from being linear, then this proof will detect this with some positive probability depending on ε.

So assume the proof is ε-close to being linear, As in the discussion of the linearity test, we can conclude that there is a unique vector $\bar{b}$ indexed by the set $J = \{0, 1, \ldots, n\} \cup \{(i, j) : 1 \leq i < j \leq n\}$ such that for all but an ε-fraction of quadratic polynomials Q, $\pi[Q]$ is equal to the dot product of the coefficient vector of Q with $\bar{b}$. For a coherent proof, the vector $\bar{b}$ should satisfy (i) $b_0 = 0$ and for all $i < j$, (ii) $b_{i,j} = b_i b_j$.

Property (i) is checked by

Test 3. Pick Q at random and verify

$$\pi[Q + 1] - \pi[Q] = 1.$$

This test will reject w.p. $1 - 2\epsilon$ if $b_\emptyset \neq 0$.
Property (ii) is tested by

Test 4. Pick *linear* functions L_1 and L_2 at random and pick quadratic functions Q_1, Q_2, Q_3 at random and verify

$$(\pi[Q_1 + L_1] - \pi[Q_1])(\pi[Q_2 + L_2] - \pi[Q_2]) = \pi[Q_3 + L_1 L_2] - \pi[Q_3].$$

The idea is that one can test if the $J \times J$ matrix (b_{ij}) is the product of vectors $(a_1, \ldots, a_n)$ and $(a_1, \ldots, a_n)^\top$ by taking random vectors $\vec{\alpha}$ and $\vec{\beta}$ indexed by J and verifying

$$\vec{\beta}(b_{ij})\vec{\alpha}^\top = (\vec{\beta}(a_1, \ldots, a_n)^\top)((a_1, \ldots, a_n)\vec{\alpha}^\top).$$

The three tests together give 13 queries that will accept any coherent proof, and reject any proof that is ε-far from coherent with some positive probability depending on ε.

8.3 Overview of Dinur's Proof of the PCP Theorem

We shall use the generalized k-coloring problem, defined below. We start with 3-colorings of graphs. If G is not 3-colorable, then the verifier rejects with probability $\geq \frac{1}{m}$, where m is the number of edges of the graph. The idea of the proof is to amplify this probability to get a positive constant. This is obtained by iterating the following lemma.

Lemma 4. *There is a linear time reduction $G \mapsto G'$ s.t. $\forall k\ \exists \epsilon_0 > 0\ \forall \epsilon > 0$,*

1. *if G is k-colorable, then so is G';*
2. *if in every k-coloring of G, at least an ϵ fraction of edges are not satisfied, then in every k-coloring of G', at least a $\min\{2\epsilon, \epsilon_0\}$ fraction of edges are not satisfied.*

Iterating this lemma $O(\log n)$ times the probability of rejecting is amplified to a positive constant and the size of the graphs increases only polynomially.

The proof of this lemma is split into two parts, the following two lemmas.

Lemma 5. *$\forall c,\ \forall k\ \exists K\ \exists$ a linear time reduction $G \mapsto G'$ s.t. $\exists \varepsilon'_0 > 0\ \forall \epsilon > 0$,*

1. *if G is k-colorable, then G' is K colorable;*
2. *if in every k-coloring of G, an ϵ fraction of edges are not satisfied, then in every K-coloring of G', a $\min\{c\epsilon, \epsilon'_0\}$ fraction of edges are not satisfied.*

Lemma 6. *$\forall k\ \exists c'\ \forall K\ \exists$ a linear time reduction $G \mapsto G'$ s.t. $\forall \epsilon' > 0$,*

1. *if G is K-colorable, then G' is k colorable;*
2. *if in every K-coloring of G ϵ' fraction of edges are not satisfied, then in every k-coloring of G', an ϵ'/c' fraction of edges are not satisfied.*

Note how critical the placement of the quantifiers is in these lemmas, especially in Lemma 6: there we fix c', and *then* any K, and surprisingly still loose only a c'-fraction of the distance.

Lecture 9. Dinur's Proof of the PCP Theorem

Lecturer: Madhu Sudan | Scribes: B. Kapron & V. King

9.1 Review of Last Lecture

Dinur's paper [11] considers a general problem:
GENERALIZED GRAPH k-COLOURING

> Input: Graph G; edge constraints
> Question: Is there a k colouring of the vertices to satisfy the edge constraints?

We define the *gap* of a graph as the fraction of edges not colourable in any assignment. As stated in the previous lecture, the PCP theorem follows by repeatedly applying the following two lemmas in sequence.

Lemma 7. *$\forall c\ \forall k\ \exists K$ such that there is a linear time reduction mapping any graph G which is k-colourable with gap ϵ to a graph G' which is K colourable with a gap equal to* $\min\{c\epsilon, \epsilon'_o\}$.

Lemma 8. *$\forall k\ \exists c'\ \forall K\ \exists$ a linear time reduction mapping any instance G' of* GENERALIZED GRAPH K-COLOURING *to an instance G" of* GENERALIZED GRAPH k-COLOURING *such that if G' has a gap of ϵ', then G" has a gap of ϵ'/c'.*

9.2 Generalized Hypergraph Colouring

We introduce:

GENERALIZED (k, l) COLOURING OF HYPERGRAPHS

> Input: hypergraph in which every edge contains l vertices; constraints on edges
> Goal: k-colour the vertices so as to satisfy the maximum number of constraints.

Remark. This problem is equivalent to MAX CSP, maximum constraint satisfaction, from folklore and appearing in [35].

Lemma 9. *$\forall k_1, k_2, l_1, l_2 \geq 2$ with $\max\{k_1, l_1\} \geq 3$ and $\max\{k_2, l_2\} \geq 3$ such that given any instance G of a (k_1, l_1)-colouring problem there is a linear time reduction to an instance G' of a (k_2, l_2)-colouring problem such that if G has gap ϵ, G' has gap ϵ/c.*

So instead of Lemma 2, it suffices to replace Lemma 2 by:

Lemma 10. *$\exists c', k, l \forall K$ there is a linear time reduction from an instance G' of* GENERALIZED K-COLOURING *to an instance G" of* GENERALIZED (k, l)-COLOURING *such that if G' has a gap of ϵ', then G'' has a gap of ϵ'/c'.*

Let's try to use a "generic" reduction. Suppose we have $\chi : \{v_1, \ldots, v_n\} \to \{1, \ldots, K\}$. We can map this to a Boolean CSP on variables v_{ij}, $1 \le i \le n$, $1 \le j \le k$, and introduce 2-colouring χ' defined by

$$\chi'(v_{ij}) = \begin{cases} 1 & \text{if } \chi(v_i) = j; \\ 0 & \text{otherwise.} \end{cases}$$

The problem in doing this is that the encoding $1 \to 1000\ldots0$, $2 \to 0100\ldots0$, $3 \to 0010\ldots0$, $\ldots$ is not very robust – it is hard to check violations of edge constraints by sampling.

Suppose instead we create gadgets for, e.g., v_1, v_2 and edge (v_1, v_2) so that if (v_1, v_2) is not satisfied, then several edges in the gadget are not satisfied, and sampling to detect violation of constraints will work. We will do this using an error-correcting code.

To simplify, we will assume that $K = 2^t$, so colours are t bit strings ($\chi(v_1) \in \{0,1\}^t$). Then constraints have the form $c_{12} : \{0,1\}^{2t} \to \{0,1\}$. Without loss of generality, we may assume that c_{12} is a degree 2 polynomial over $\mathbb{F}_2$.

We define vertex and edge gadgets. Vertex v_1 is mapped to vertices $v_{11}, v_{12}, \ldots v_{1m}$. These vertices are coloured either 0 or 1. We want that $\chi(v_1, q) = q(\chi(v_1))$, where q is a quadratic function on t bits ("vertex gadgets") and $\chi(v_{12}, Q) = Q(\chi(v_1), \chi(v_2))$, where Q is a quadratic function on $2t$ bits ("edge gadgets"). Edges must be of constant size l, independent of t. These simulate PCP's to verify that there is an assignment satisfying the constraints (hyperedge constraints correspond to PCP tests.) Figure 9.11 illustrates the construction.

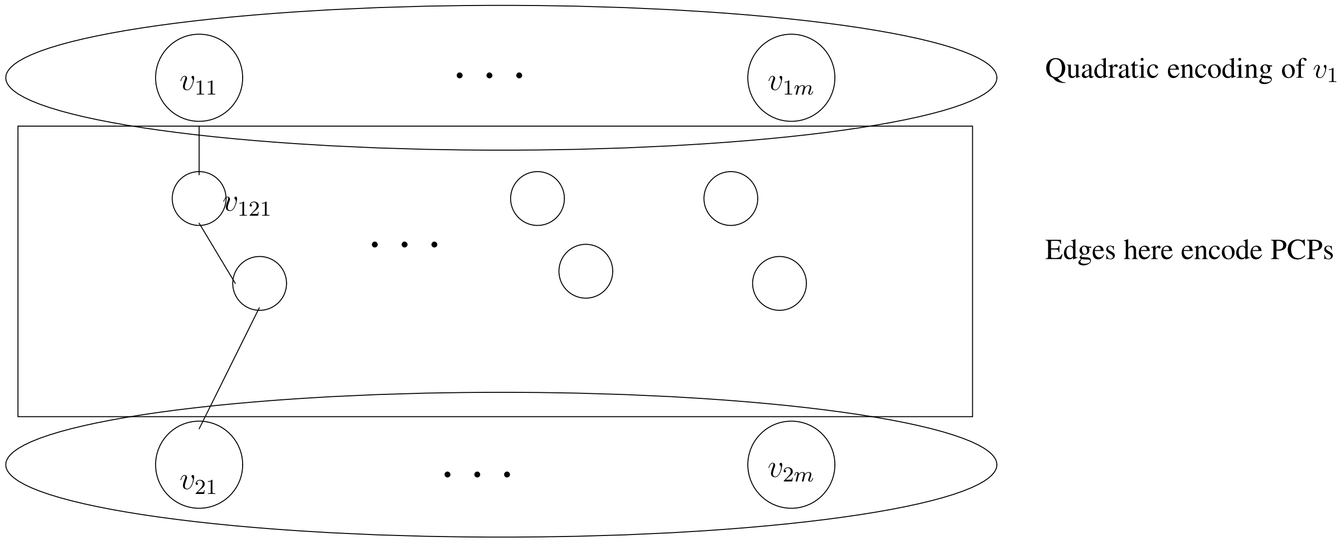

Figure 9.11: Connecting the gadgets.

We need to enforce consistency by checking Q's against q's, as shown in Figure 9.12. We also have to verify that the colouring of constraint nodes satisfy the constraint, and that the colouring is consistent with node colourings.

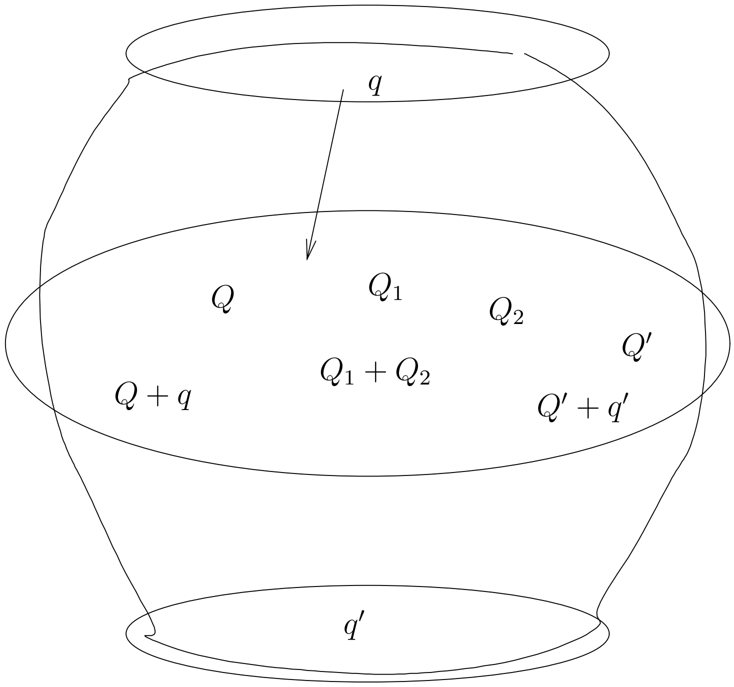

Figure 9.12: Checking consistency.

The gadgets function as follows: if the fraction of $v_1 - v_2$ edges that are satisfied is $> 1 - \frac{1}{c'}$, then the values assigned to the gadgets for v_1 and v_2 are 90% close to encodings of colours a_1 and a_2 such that $c_{12}(a_1, a_2)$ is satisfied.

If we now go to each node encoding and decode such that the assignments satisfy a $1 - \frac{\epsilon}{c'}$ of all the constraints, it follows that on an ϵ fraction, $1 - \frac{1}{c'}$ are satisfied, and so we can satisfy a $1 - \frac{1}{\epsilon}$ fraction of the original constraints.

9.3 The New Part in Dinur's Proof

The new idea in Dinur's proof is the way of performing Gap Amplification:

$$\begin{array}{rcl} k\text{-colouring} & \longrightarrow & K\text{-colouring} \\ \text{colourable} & \longrightarrow & \text{colourable} \\ \epsilon\text{-uncolourable} & \longrightarrow & \min\{c\epsilon, \epsilon_0\}\text{-uncolourable} \end{array}$$

By picking a random edge and checking its colour, we will be wrong $1 - \epsilon$ of the time. We would like to boost our chance of success through repeated sampling. We will use a *parallel repetition*, where nodes are represented using pair of nodes and edges are represented using pairs of edges.

Ran Raz [39] has shown how such a scheme can work on bipartite graphs to reduce the error. However, there is still a problem with quadratic blowup in the size of the graph. The solution here is not just to use pairs, but to collect sets of vertices, based on the result of a random walk on an expander. Usually, when this technique is used the expander is not related to the original graph, but this is not the case here. The expander is added to the original graph, and the random walk is on the union of the edges in the two graphs.

We form a new graph as follows: for every vertex v of G, there is a corresponding vertex $\tilde{v}$ in G', where $\tilde{v}$ is the ball of radius t around v. For a constant size reduction, we must have constant size balls. We colour as follows. Assuming that G has constant degree d, if $\chi(v) \in \{1, \ldots, k\}$, then $\chi(\tilde{v}) \in \{1, \ldots, k\}^{d^t}$. The edges of G' are *all* walks of length $l \in_R [\frac{9}{10} \ldots t]$. A constraint on w of length l from $\tilde{v}_1$ to $\tilde{v}_2$ enforces that any assignment on the two balls agrees on their intersection, and that every edge constraint in the union of the two balls is satisfied.

We have that if ϵ edges are not satisfied in G, then $\min(\Omega(\epsilon\sqrt{t}), \frac{1}{t})$ will be unsatisfied in G'.

It remains to describe how we colour G. Namely, we set $\chi(v_1) = \text{Plurality}\{\chi(\tilde{v}_2)_{v_1}\}$ where v_2 is a node reached by taking a random walk of length $\frac{l}{2}$ starting at v_1, and c_{v_1} denotes the projection of c onto v_1.

9.4 Course summary, topics not covered

Bounds

Significant bounds covered were the singleton bound, Hamming bound and Elias bound. We did not cover Delsarte and [40]. Also the LP bound, which is powerful, but hard to explain. Alon gives an LP-like bound for ϵ-biased spaces. See Sudan's crash course notes for more details. We would like to see a clean proof that the ϵ^2 given by Chernoff bounds holds at the limit where $\delta = \frac{1}{2}$. We know that $R + \delta \geq 1 - \frac{1}{\sqrt{q}}$ (algebraic geometric codes) and that $R + \delta \leq 1 - \frac{1}{q}$ (Plotkin bound).

Codes

We covered polynomial-based codes in detail, and also looked at expander-based codes. We did not consider the graph-based codes of [1]. Also BCH codes, which give us

$$\frac{d}{2}(\log n)^{1-o(1)} \underset{\text{(Hamming)}}{\leq} n - k \underset{\text{(BCH)}}{\leq} \frac{d}{2}\log n$$

for fixed d as $n \to \infty$, for $q = 2$. What happens for $q > 2$? We have

$$\frac{d}{2}\log_q n \leq n - k \leq \frac{(q-1)d}{q}\log_q n,$$

which is not so tight. There seems to be a strong connection to the "Rodl Nibble", producing a constant weight code of large density. In general, we would like to know if there is a simpler way than AG codes to get $1 - \frac{1}{\sqrt{q}}$?

Algorithms for decoding

Expander-based:

- Luby *et al.* construction [31] – works extremely well for random errors.
- Guruswami and Indyk's [21] – Expander based list decoding. They give a linear time list decoding algorithm for many codes including [1]

See Guruswami's survey [19].

Algebraic list decoding:
Results are given by [36] and by [23]. The latter achieve channel capacity. A message $p(x)$ is mapped to $p(\alpha_1)\ldots p(\alpha_n)$. This is broken into blocks of size c:

$$\underbrace{\{p_{\alpha_1}\ldots p_{\alpha_c}\}}_{\text{(First letter of new alphabet)}}, \ldots, \{p(\alpha_{n-c+1})\ldots p(\alpha_n)\}.$$

It is still open whether there are binary codes with $R = \epsilon^2$ with correct fraction $\frac{1}{2} - \epsilon$ of errors.

Bibliography

[1] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good low-rate error -correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–282, 1992.

[2] N. Alon, O. Goldreich, J. H. stad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[3] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, 1991.

[4] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the Seventh Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48, Berlin, 1990. Springer Verlag.

[5] M. Bellare, D. Coppersmith, J. Håstad, M. A. Kiwi, and M. Sudan. Linearity testing in characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, 1996.

[6] M. Ben-Or, D. Coppersmith, M. Luby, and R. Rubinfeld. Non-abelian homomorphism testing, and distributions close to their self-convolutions. In *APPROX-RANDOM*, volume 3122 of *Lecture Notes in Computer Science*, pages 273–285. Springer Verlag, 2004.

[7] E. R. Berlekamp. *Chapter 7 in Algorithmic Coding Theory*. New-York: McGraw-Hill, 1968.

[8] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[9] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal of Computing*, 13(4):850–864, 1984.

[10] M. R. Capalbo, O. Reingold, S. P. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 659–668, 2002.

[11] I. Dinur. The PCP theorem by gap amplification. *Electronic Colloquium on Computational Complexity (ECCC)*, 05(046), 2005.

[12] P. Elias. List decoding for noisy channels. *Wescon Convention Record, Part 2, Institute of Radio Engineers (now IEEE)*, pages 94–104, 1957.

[13] G. D. Forney, Jr. *Concatenated Codes*. MIT Press, Cambridge, MA., 1966.

[14] R. Gallager. *Low density parity-check codes*. MIT Press, Cambridge, MA., 1963.

[15] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 32–42, 1991.

[16] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[17] O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000. Preliminary version in FOCS'95.

[18] D. Gorenstein and N. Zierler. A class of error-correcting codes in p^m symbols. *J. Soc. Ind. Appl. Math.*, 9:207–214, 1961.

[19] V. Guruswami. Error-correcting codes and expander graphs. *SIGACT News*, 35(3):25–41, 2004.

[20] V. Guruswami, J. Håstad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1034, 2002.

[21] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the Forty-Second Annual IEEE Symposium on Foundations of Computer Science*, pages 658–667, 2001.

[22] V. Guruswami and A. Rudra. Explicit capacity-achieving list-decodable codes. Technical report, Electronic Colloquium on Computational Complexity, 2005.

[23] V. Guruswami and A. Rudra. Explicity capacity-achieving list-decodable codes. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, 2006.

[24] V. Guruswami and M. Sudan. Reed-solomon and algebraic- geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[25] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.

[26] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[27] T. Jiang and A. Vardy. Asymptotic improvement of the gilbert-varshamov bound on the size of binary codes. *IEEE Transactions on Information Theory*, 50(8):1655–1664, 2004.

[28] J. Justesen. A class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18:652–656, 1972.

[29] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 80–86, 2000.

[30] R. Lipton. New directions in testing. In J. Feigenbaum and M. Merrit, editors, *Distributed Computing and Cryptography*, pages 191–202. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 2, AMS, 1991.

[31] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47:569–584, 2001.

[32] J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15:122–127, 1969.

[33] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[34] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[35] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[36] F. Parvaresh and A. Vardy. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *Proceedings of the Forty-Sixthth Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005.

[37] W. W. Peterson. Encoding and error-correction procedures for bose-chaudhuri codes. *IRE Transactions on Information Theory*, IT-60:459–470, 1960.

[38] A. Polishchuk and D. A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 194–203, 1994.

[39] R. Raz. A parallel repetition theorem. *SIAM J. on Computing*, 27(3):763–803, 1998.

[40] H. R. R.J. McEliece, E.R. Rodemich and L. Welch. New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *IEEE Transactions on Information Theory*, 23:157–166, 1977.

[41] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 (July and October):379–423 and 623–656, 1948.

[42] M. Sipser and D. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[43] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1732, 1996.

[44] M. Sudan. Decoding of reed-solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[45] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[46] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[47] R. Tanner. Explcit concentrators from generalized n-gons. *SIAM Journal on Algebraic Discrete Methods*, 5(3):287–293, 1984.

[48] L. Trevisan. Some applications of coding theory in computational complexity. *Survey Paper in Quaderni di Matematica*, 13:347–424, 2004.

[49] L. R. Welch and E. R. Berlekamp. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.

[50] A. C.-C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.