Table of Contents

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	November 3rd, 2005

Lecture 1: History, Terminology and Statement of the PCP Theorem

Lecturer: Eli Ben-Sasson *Scribe: Eyal Rozenberg*

In this lecture we will formulate the PCP Theorem and discuss the current state-of-the-art in terms of its parameters. Before doing so, however, we will recall the notion of classical, error-less proof checking and introduce the (relatively new) notion of proofs "beyond reasonable doubt". As we shall see, if one is willing to settle for these proofs, the benefits are big (and surprising) savings in computational resources. After stating the PCP Theorem, we will briefly survey some of the key results leading up to it.

1.1 Deterministic proofs vs. proofs "beyond any reasonable doubt"

Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language.[1]
Let us rephrase the definition of the well-known complexity class, **NP**, in terms of proof checking:

Definition 1.1 (The class **NP**)**. NP** A language $L \subseteq \Sigma^*$ is in **NP** iff there exists a deterministic Turing machine M_L with two inputs: an *instance* x of length n and a *proof* y of length poly(n), such that the following holds:

- **Operation:** M runs in time poly(n) and outputs either accept or reject.
- **Perfect Completeness:** $x \in L \Rightarrow \exists\, y \left[M(x,y) = \mathsf{accept}\right]$.
- **Perfect Soundness:** $x \notin L \Rightarrow \forall\, y \left[M(x,y) = \mathsf{reject}\right]$.

The following is a concrete well-known example of a language in **NP** (actually, it is **NP**-complete).

Example 1.2. CIRCUIT-SAT is the language of (encodings of) *satisfiable* Boolean circuits with NOT and (fan-in two) AND gates. A circuit with n inputs is satisfiable iff there exists $y \in \{0,1\}^n$ such that $C(y) = 1$. Notice y is a *proof of satisfiability* for the circuit C.

In the above definition, the completeness and soundness demands are quite strict. Thus, a proof as defined above is similar to a classical mathematical proof. There is no room for error and a statement (of the form "$x \in L$") is true if and only if it has a proof. Consider now a legal proof as might be presented in a court. Here, one settles for a proof "beyond reasonable doubt", admitting there is a small probability that the statement being proved

[1] We usually assume $\Sigma = \{0,1\}$, and that the Turing machines' alphabet has an extra 'space' (or 'empty') character in its working alphabet.

is false (e.g. there is a small probability that the knife fell from the sky right into the deceased's chest and bounced into the hand of the suspect, his nemesis :-))
Now consider the computational setting. Suppose we are willing to settle for accepting proofs beyond reasonable doubt, i.e. with an error probability of at most ε. In other words, we introduce randomness into our verification process and acknowledge that with probability $\leq \varepsilon$ we might accept $x \notin L$ (imperfect soundness) or reject $x \in L$ (imperfect completeness). By allowing this relaxed notion of a proof (and with the use of randomness) we will save greatly on computational resources. Let us formally define the classes of languages decided by proofs beyond reasonable doubt, and before that we define the probabilistic verification process of such a proof.

Definition 1.3 (Verifier)**.** For functions $\ell, q, r, t : \mathbb{N}^+ \to \mathbb{N}^+$, a (ℓ, q, r, t)*-restricted verifier* V is a randomized Turing machine with oracle access to a proof Π; On input x of length n, V receives a proof of length $|\Pi| \leq \ell(n)$, flips at most $r(n)$ coins, works in time $\leq t(n)$, queries the proof at $q(n)$ locations at most (each query is answered by a single element of Σ) and outputs accept or reject.
The output of V on input x with proof Π and coin flips R is denoted by $V^{\Pi}[x; R]$.

When we specify our completeness and soundness demands from a (ℓ, q, r, t)-restricted verifier, we have fully defined a complexity class:

Definition 1.4 (PCP classes)**.** Let ℓ, q, r, t be as in Definition 1.3 and let $c, s : \mathbb{N}^+ \to [0, 1]$. The class

$$\mathbf{PCP}\begin{pmatrix} \mathsf{length} & = & \ell(n) \\ \mathsf{randomness} & = & r(n) \\ \mathsf{query} & = & q(n) \\ \mathsf{time} & = & t(n) \\ \mathsf{completeness} & = & c(n) \\ \mathsf{soundness} & = & s(n) \end{pmatrix}$$

is the class of languages L for which there exists a (ℓ, q, r, t)-restricted verifier V satisfying:

- **Completeness:** $x \in L \Rightarrow \exists\, \Pi \left[\mathbf{Pr}_R\left[V^{\Pi}[x; R] = \mathsf{accept}\right] \geq c(n)\right]$
- **Soundness:** $x \notin L \Rightarrow \forall\, \Pi \left[\mathbf{Pr}_R\left[V^{\Pi}[x; R] = \mathsf{accept}\right] \leq 1 - s(n)\right]$

The probability above is taken over R chosen uniformly at random from $\{0, 1\}^{r(n)}$.

Whenever we do not specify c and s, they are implicitly assumed to be 1 (perfect completeness) and 1/2 respectively.

Observation 1.5. For every verifier V, $\mathsf{time} \geq \mathsf{query}$, and we may assume without loss of generality that $\mathsf{length} \leq \mathsf{query} \cdot 2^{\mathsf{randomness}} \leq \mathsf{time} \cdot 2^{\mathsf{randomness}}$.

Observation 1.6. It follows immediately from our definition of **NP** above that

$$\mathbf{NP} = \mathbf{PCP}\begin{pmatrix} \mathsf{length} & = & \mathrm{poly}(n) \\ \mathsf{randomness} & = & 0 \\ \mathsf{query} & = & \mathrm{poly}(n) \\ \mathsf{time} & = & \mathrm{poly}(n) \\ \mathsf{completeness} & = & 1 \\ \mathsf{soundness} & = & 1 \end{pmatrix}$$

1.2 Statement of (two variants of) the PCP Theorem

There are several different parameters in the definition of a PCP class (and in the course of our lectures we shall see several more). In general, we wish to show that a language L is contained in a PCP-class with minimal length, running-time, randomness and query complexity and with maximal completeness and soundness. There are several different formulation of PCPs that optimize different sets of parameters, and we now state two such results (several other exist in the literature and are useful for different goals). The first is useful for efficient proof verification (in terms of running time and communication complexity) and the second is useful for obtaining hardness of approximation results. Most of our course will focus on proving these two results.

Theorem 1.7 (The PCP Theorem)**.** *The following hold:*

"Short (quasi-linear) PCPs":

$$\mathbf{NTIME}\left(f(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = f(n)\cdot \mathrm{polylog}(f(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}\left(1\right) \\ \mathsf{query} & = \mathcal{O}\left(1\right) \\ \mathsf{time} & = f(n)\cdot \mathrm{polylog}(f(n)) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = \frac{1}{2} \end{array}\right)$$

"Sound, query-parsimonious, PCPs" :

$$\mathbf{NTIME}\left(f(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = \mathrm{poly}(f(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}\left(1\right) \\ \mathsf{query} & = 3 \text{ (over alphabet } \{0,1\}) \\ \mathsf{time} & = \mathrm{poly}(f(n)) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = \frac{1}{2} - \varepsilon \end{array}\right)$$

For any reasonable function $f(n)$.

Notice the soundness can be increased to any arbitrary constant $s < 1$ by (sequential or expander-based) repetition. However, this increases the running time and query complexity by a factor equal to the number of repetitions. A major part of our effort will be devoted to obtaining and analyzing resource-efficient 'soundness boost' methods.

1.3 A (brief and incomplete) history of the PCP Theorem

There have been many counter-intuitive discoveries along the path leading to the PCP Theorem. One surprise is that when settling for random verification "beyond reasonable doubt", one can make do with so little computational resources (like a constant number of queries into the proof). Another surprise is that the PCP-Theorem (especially in the second part of Theorem 1.7) implies numerous tight inapproximability results.
We now briefly discuss some of the main surprises along the way, omitting many other crucial contributions (for more information, see [46, 1]).

1.3.1 Initial results leading up to the proof of the PCP Theorem

The notion of proofs "beyond reasonable doubt" in its computational setting was introduced by Goldwasser, Micali and Rakoff [27] in the 1980s. In their work a verifier (similar to the one in Definition 1.3) *interacts* with another machine (called a *prover*). Such a proof scheme is called *interactive*.
The interaction in the proof was first removed in the work of Fortnow, Rompel and Sipser [22], who replaced the interactive *prover* (Turing machine) with a non-interactive *proof* (string of bits). They developed this notion as part of the study of multi-prover interactive proof systems and referred to them as 'transparent' or 'holographic' proofs.
Later in the 1980s, Babai Fortnow and Lund [6] proved that

$$\mathbf{NEXPTIME} = \mathbf{NTIME}\left(2^{n^{\mathcal{O}(1)}}\right) = \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{length} & = & \mathrm{poly}(n) \\ \mathsf{randomness} & = & \mathrm{poly}(n) \\ \mathsf{query} & = & \mathrm{poly}(n) \\ \mathsf{time} & = & \mathrm{poly}(n) \end{array}\right)$$

Notice the big and surprising computational saving in the Theorem above. Instead of exponential time (required by a "classical" deterministic proof) we can verify proofs beyond reasonable doubt in polynomial time!
A short while later, Babai, Fortnow, Levin and Szegedy [5] performed a 'scale-down' of the principle of this proof to show that if $L \in \mathbf{NP}$ is "encoded", i.e. contains only strings that are encoded by a good error correcting code (i.e. one that is efficiently computable and has large distance), then

$$L \in \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{length} & = & \mathrm{poly}(n) \\ \mathsf{randomness} & = & \mathrm{poly}(\log(n)) \\ \mathsf{time} & = & \mathrm{poly}(\log(n)) \end{array}\right)$$

Notice the verifier is so efficient, it does not even have time to read more than a negligible part of the statement being proved! (for this to work we need to assume statements are encoded).
At about the same time, Feige, Goldasser, Lovasz, Safra and Szegedy [20] showed that **PCP** systems for **NP**– specifically, **PCP** s with a low number of queries and high soundness – imply it is hard to approximate the solution to certain combinatorial optimization problems such as MAX-CLIQUE.
As a consequence of this surprising connection, much of the attention in PCP research was given to minimizing the query complexity while maximizing soundness.
Shortly after, Arora and Safra [4] formally defined the notion of PCP's, with randomness and query complexity as main parameters. They also introduced the notion of proof-composition of PCP's (a technique crucial for reducing query complexity without reducing the soundness too much). Applying these new definitions and tools they showed:

$$\mathbf{NP} \subseteq \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{randomness} & = & \mathcal{O}(\log(n)) \\ \mathsf{query} & = & o(\log(n)) \end{array}\right)$$

Observation 1.8. The reverse of the above inclusion is relatively easy to derive: With logarithmic randomness and polynomial running time, a verifier can brute-force its way through all possible randomness sequences. The same reasoning shows the running time and proof

length are at most polynomial. What Arora and Safra had achieved was therefore a non-trivial *characterization* of **NP** in **PCP** terms.

Building on this result, Arora, Lund, Motwani, Sudan and Szegedy [3] reduced the query complexity to a constant (yet another big surprise), resulting in the following characterization of **NP**, known as the 'basic PCP Theorem':

$$\mathbf{NP} = \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{length} & = & \mathrm{poly}(n) \\ \mathsf{randomness} & = & \mathcal{O}\left(\log(n)\right) \\ \mathsf{query} & = & \mathcal{O}\left(1\right) \\ \mathsf{time} & = & \mathrm{poly}(n) \end{array}\right)$$

1.3.2 Improvement of the basic PCP Theorem and its main parameters

- length: Spielman and Polischuk [39] reduced the proof length and showed that

$$\mathrm{SAT} \in \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{randomness} & = & (1+\varepsilon)\log(n) \\ \mathsf{query} & = & \mathcal{O}\left(1\right) \end{array}\right)$$

  which implies that an almost-linear proofs suffice for constant-query PCP verifiers. Later improvements by Harsha and Sudan [29]; Goldreich and Sudan [26]; Ben-Sasson, Sudan, Vadhan, Wigderson [13]; Ben-Sasson, Goldreich, Harsha, Sudan, Vadhan [9]; Ben-Sasson, Sudan [11] and Dinur [17] led to short PCPs (the first part of Theorem 1.7).

- soundness: The 'Parallel Repetition Theorem' of Raz [40] regarding multi-prover interactive proofs has been crucial to obtaining PCPs with high soundness and small query complexity. The recent 'Gap Amplification Theorem' of Dinur [17] is an important resource-efficient soundness amplification technique (useful in a different setting of parameters than that of Raz's parallel repetition) and is crucial to obtaining quasi-linear PCPs with constant query complexity.

- query: A major tool in reducing the query complexity (while leaving the soundness relatively high) is the use of the 'long code', introduced by [8]. The second part of the PCP Theorem 1.7 above was proved by Håstad [30]. The 'long code'-based proof, besides comprising interesting techniques of its own, relies heavily on Raz' Parallel Repetition Theorem.

  > ER: *of course a long-code-based proof relies on the properties of the long code... removed that bit*

1.3.3 Recent Trends

- **Inapproximability results:** Since the appearance of the PCP Theorem (and most notably Håstad's result) there has been a steady output of various inapproximability results using the PCP Theorem (see e.g. the surveys [47, 2]).

- **Revisiting the proof of the PCP Theorem:** In attempts to understand the minimal necessary ingredients in the proof of the PCP Theorem, several simplifications arose [18, 9, 11], most notably the recent surprising proof via gap amplification [17].

- **Khot's 'Unique Games Conjecture':** A stronger version of the PCP Theorem (the complexity of which is at the moment unresolved) [31]. If the conjecture is true (and the associated problem is indeed hard to decide), then many new tight inapproximability results follow (See [32] for a survey).

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	November 14th, 2005

Lecture 2: Testability and decodability of the Hadamard codes

Lecturer: Eli Ben-Sasson *Scribe: Anat Paskin*

In this lecture, we state and present the bulk part of the proof of the following quite surprising (at the time) weaker **PCP** theorem version:

Theorem 2.1.

$$NP \subseteq \mathbf{PCP}\left(\begin{array}{ll} \text{length} & = 2^{\mathcal{O}(n^2)} \\ \text{randomness} & = \mathcal{O}\left(n^2\right) \\ \text{query} & = 16 \\ \text{time} & = \mathcal{O}\left(n^2\right) \\ \text{completeness} & = 1 \\ \text{soundness} & = 1/200 \end{array}\right)$$

Although the proof of this theorem is quite simple, it already contains several key ideas which lead to the proof of the **PCP** theorem. The proof is based on demonstrating that the **NP**-complete language CIRCUIT-SAT (see Example 1.2 for definition) belongs to this class. The main idea of the proof is to encode **PCP** witnesses using an error correcting code (ECC) with large distance between codewords. This code will have very nice properties of local testability, and locally decodability. On a high level, local testability means that there exists an efficient algorithm for testing for being a codeword which reads very few of its bits, and accepts in case the string is a valid codeword, and rejects with high probability in case it's 'far' from a codeword. Local decodability means there exists an efficient randomized 'decoder' that correctly decodes (with high probability) certain properties of a message from its 'slightly' corrupted encoding. The main result of this lecture is locally testable and decodable. This will allow us to efficiently and reliably read linear combinations of bits of a given proof and in the next lecture encode a witness for satisfiability in a manner that will allow proof verification with constant query complexity, perfect completeness and constant soundness (proving Theorem 2.1). Details follow.

2.1 Definitions and notation

In this section, we review several basic concepts used throughout the proof.

2.1.1 Error correcting codes

Definition 2.2 (Hamming distance)**.** Let F be a field, $n \in \mathbb{N}^+$, and $x, y \in F^n$. The *Hamming distance* between x and y is

$$\Delta(x,y) = |\{i \in [k] \mid x_i \neq y_i\}|$$

The *relative Hamming distance* between x and y is

$$\Delta^*(x,y) = \frac{\Delta(x,y)}{n}$$

Note. The fields we shall use in our constructions are all finite. Specifically, $F = \mathbb{F}_2 = \{0, 1\}$ throughout the proof of Theorem 2.1.

Definition 2.3 (Error-Correcting Code)**.** Let F be a field and $k, n, d \in \mathbb{N}^+$. An *error correcting code* $C : F^k \to F^n$ $[n, k, d]_F$-ECC is an injective mapping satisfying

$$\forall x, x' \in F^k \ , \ x \neq x' \Rightarrow \Delta\big(C(x), C(x')\big) \geq d$$

n is the *block length* of the code; k/n is the code's *rate* and d is the code's *distance* (d/n is the code's *relative distance*). If, additionally, C is a linear mapping, it is said to be a $[n, k, d]_F$-LECC (*linear error correcting code*).

The 'quality' of an ECC is measured both by high rate and high distance (two contradicting parameters). In what follows, we consider linear codes with very poor (small) rate and very good distance. Indeed, the distance will later on be instrumental in proving Theorem 2.1. We now extend the notion of Hamming distance from elements to sets:

Definition 2.4. The Hamming distance distance between an element $w \in F^n$ and a set $S \subseteq F^n$ is

$$\Delta(w, S) = \begin{cases} \min_{y \in S} \Delta(w, y) & S \neq \varnothing \\ 1 & \text{otherwise} \end{cases}$$

The Hamming distance between an element $w \in F^n$ and an ECC C is

$$\Delta(w, C) = \Delta\Big(w, \Big\{ C(x) \ \Big| \ x \in F^k \Big\}\Big)$$

Definition 2.5 (Hadamard code)**.** For $a \in F^k$, let $\ell_a : F^k \to F$ be the linear function

$$\ell_a(x) = \langle x \ , \ a \rangle = \sum_{i=1}^{k} a_i x_i$$

Let $\mathcal{L}_k$ denote the space of k-dimensional linear functions over F; we note that

$$\mathcal{L}_k = \Big\{ \ell_a \ : \ F^k \to F \ \Big| \ a \in F^k \Big\}$$

The k-dimensional Hadamard code, $H_k : F^k \to F^{2^k}$ is the function

$$H_k(x) = (\ell_a(x))_{a \in F^k}$$

that is, the sequence of values of the application of all linear functions to x, in some fixed order (e.g, lexicographic order of the a's).

Observation 2.6. The Hadamard code is a $[2^k, k, 2^{k-1}]_F$ LECC (in fact, the distance between any pair of codewords is *exactly* 2^{k-1}). The code has, therefore, a relative distance of $\frac{1}{2}$. The code's rate, however, is rather poor (inverse-exponential in k).

2.1.2 Locally testable codes

Another concept we will need is locally testable codes. A code C is locally testable if there exists an efficient tester, that reads few bits of a supposed codeword, and distinguishes between two cases: The string is a codeword in which case the tester always accepts, or the string is 'far' (by some problem-related measure) from being a codeword, in which case the tester rejects with constant probability. We will measure the distance from C using the metric $\Delta^*(\cdot,\cdot)$. Next, we formally define localy testable codes. Notice the similarity of this definition to that of a **PCP** language class.
From now on, we sometimes use the term **PPT** as a shorthand for probabilistical polytime algorithms.

Definition 2.7 (Local Tester)**.** Let $\mathcal{C} = \left\{C_k : F^k \to F^{n(k)} \;\middle|\; k \in \mathbb{N}^+\right\}$ denote a family of codes, and let $n(k), r(k), q(k), t(k) : \mathbb{N}^+ \to \mathbb{N}^+$ and $s(\delta, k) : [0,1] \times \mathbb{N}^+ \to [0,1]$. A randomized Turing machine T is a $(t(k), r(k), q(k), s(\delta,k))$-tester for $\mathcal{C}$ if, on input 1^k and oracle access to a string $w \in F^{n(k)}$, it satisfies the following:

- **Operation**: T^w reads its randomness of length [2] $r(k)$, makes at most $q(k)$ queries to w (each query reads an element of F), and terminates within at most $t(k)$ steps with a result in $\{\mathsf{accept}, \mathsf{reject}\}$. Denote the action of T on oracle w and randomness R by $T^w\left[1^k; R\right]$.
- **Prefect Completeness**: $w \in \mathrm{Img}(C_k) \Rightarrow \mathbf{Pr}_R\left[T^w\left[1^k; R\right] = \mathsf{accept}\right] = 1$
- **Soundness**: $w \notin \mathrm{Img}(C_k) \Rightarrow \mathbf{Pr}_R\left[T^w\left[1^k; R\right] = \mathsf{accept}\right] \leq 1 - s(\Delta^*(w, C_k), k)$

Definition 2.8. A $(t(k), r(k), q(k), s(\delta,k))$-LTC *locally-testable code* is a code $C : F^k \to F^n$ which has an $(t(k), r(k), q(k), s(\delta,k))$-local-tester.

2.2 The Hadamard code is locally testable

The main result (in terms of technical complexity) we prove in this lecture is the following theorem:

Theorem 2.9 ([36])**.** *The Hadamard codes $\{H_k \mid k \in \mathbb{N}^+\}$ are locally testable with parameters $(O(k), 2k, 3, s(\delta) \geq \min(\delta/2, 2/9))$.*

In fact, the theorem as stated here is a special case of a more general result proven by Blum, Luby, and Rubinfeld [36] for testing group homomorphism. As to further optimizing the parameters of this construction, the query complexity of this result was shown to be tight in [10]. Some parameters, however have been improved in subsequent work. Bellare et al.[7] improve s for (our) special case of the Hadamard code to $s(\delta) \geq \delta$. It has been shown in [12] that r may also be reduced to $r(k) = (1 + o(1))k$, and this randomness efficient testing was generalized in [45] to the original homomorphism testing setting of [36].
Note that any H_k-codeword may be naturally viewed as a function $f : F^k \to F$. In fact, $\mathrm{Img}(H_k)$ is precisely the set of linear functions $\left\{\ell_a : F^k \to F \;\middle|\; a \in F^k\right\}$, where $H_k(x)$ represents the 'table' of values of the function $\chi_x(a) = \ell_a(x) = \ell_x(a)$. This may be restated as follows

[2] The tester as described here is non-adaptive...

Observation 2.10. a function $f : F^k \to F$ is a codeword of H_k iff every 2 elements $a, b \in F^k$ satisfy $f(a) + f(b) = f(a + b)$.

Proof. Proof of Theorem 2.9 A local tester T_H for the Hadamard will utilize Observation 2.10: it will test a function f for being close to some linear function. Given input 1^k and oracle access to f, the tester will perform the following:

1. Select $a, b \in F^k$ randomly and independently.
2. Query the oracle for $f(a)$ $f(b)$ and $f(a + b)$.
3. Output accept if $f(a) + f(b) = f(a + b)$, reject otherwise.

T_H obviously satisfies the query, randomness and running time requirements of the theorem are obviously satisfied; by Observation 2.10, the test has perfect completeness. Lemma Lemma 2.11, proven below, is the contrapositive of the soundness requirement, so T_H satisfies this requirement as well. □

Lemma 2.11. *If T_H rejects a function $f : F^k \to F$ with probability $\varepsilon < 2/9$, then f is within a relative Hamming distance of 2ε from some codeword of H_k.*

Proof of Lemma 2.11. Assume $\mathbf{Pr}_R\left[T_H^f[1^k; R] = \mathsf{reject}\right] = \varepsilon < 2/9$. We have to show $\Delta^*(f, w) \leq 2\varepsilon$ for some Hadamard codeword w, and here is a candidate:

Definition 2.12 (Majority function)**.** For an oracle $f : F^k \to F$, define an auxiliary function $\phi : F^k \to F$ by $\phi_a = maj_{b \in F^k}(f_{a+b} - f_a)$.

Intuitively, ϕ represents the most common vote for the value of a (which may disagree with f_a). It turns out ϕ is a Hadamard codeword (we will prove this separately in Lemma 2.13, below). How far is f from ϕ? To answer this question, let

$$\text{Bad} = \left\{ a \in F^k \;\middle|\; \phi(a) \neq f(a) \right\}$$

Now we have on one hand

$$\begin{aligned}
\Delta^*(f, \phi) = \mathop{\mathbf{Pr}}_{a}[a \in \text{Bad}] \in tertextandontheotherhand,\varepsilon \qquad &= \mathop{\mathbf{Pr}}_{a,b}\left[T_H^f\left[1^k; a, b\right] = \mathsf{reject}\right] \\
&\geq \mathop{\mathbf{Pr}}_{a,b}\left[T_H^f\left[1^k; a, b\right] = \mathsf{reject} \middle| a \in \text{Bad}\right] \cdot \mathop{\mathbf{Pr}}_{a}[a \in \text{Bad}] \\
&\geq \frac{1}{2} \cdot \mathop{\mathbf{Pr}}_{a}[a \in \text{Bad}]
\end{aligned}$$

so $\Delta^*(f, \phi) \leq 2\varepsilon$, that is, f is indeed close enough to ϕ, and consequently to H_k. □

Lemma 2.13. $\phi \in \text{Img}(H_k)$.

Proof. First, we show that for all a, ϕ_a is 'voted' by a 'vast' majority of b', that is, the following claim is true:

Claim 14. *For all $a \in F^k$ we have $\Pr_{b \in F^k}(f_{a+b} - f_b = \phi_a) > 2/3$.*

Claim 14. Consider the following mental experiment: Fix any $a \in F^k$, and select $b, c \in F^k$ at random. We observe the following relations

$$Pr_{b,c}[f_{a+b} - f_b = f_{a+c} - f_c] = Pr_{b,c}[f_{a+b} + f_c = f_{a+c} + f_b] \quad (1)$$
$$Pr_{b,c}[f_{a+b} + f_c \neq f_{a+b+c}] < 2/9 \quad (2)$$
$$Pr_{b,c}[f_{a+c} + f_b \neq f_{a+b+c}] < 2/9 \quad (3)$$

Equation 1 follows by simple rearranging of the equation describing the event. The last pair of inequalities follows from the assumption $\varepsilon < 2/9$, since the pair $(a+b, c)$, in this experiment is distributed identically to a pair of random independent variables in F^k, so $Pr_{b,c}[f_{a+b} + f_c \neq f_{a+b+c}]$ is exactly the rejection probability of T^f. By the union bound, it follows from equations 2, 3, that

$$Pr_{b,c}[f_{a+b} - f_b = f_{a+c} - f_c] > 5/9 \quad (4)$$

Let $p = \Pr_b[f_{a+b} - f_b = \phi(a)] = \Pr_c[f_{a+c} - f_c = \phi(a)]$. Then $Pr_{b,c}[f_{a+b} - f_b = f_{a+c} - f_c] = p^2 + (1-p)^2$. Equation 4 implies $p^2 + (1-p)^2 > 5/9$ and by assumption on ϕ we have $p \geq 1/2$. Simple analysis shows $p > 2/3$ and this completes the proof of Claim 14. □

To complete the proof of Lemma 2.13, fix another arbitrary $a' \in F^k$. By Claim 14,

$$Pr_b[\phi(a) \neq f(b) - f(b-a)] < 1/3$$
$$Pr_b[\phi(a') \neq f(a+b') - f(b)] < 1/3$$
$$Pr_b[\phi(a+a') \neq f(a'+b) - f(b-a)] < 1/3$$

The first inequality holds since $b - a$ is distributed uniformly over F^k; the other two are justified by a similar argument. Now, by the union bound, there exists a b not satisfying any of the three inequalities above. b satisfies $\phi(a') + \phi(a) = \phi(a+a') = f(a'+b) - f(b-a)$; since a, a' are arbitrary elements of F^k, it also holds that $\phi(a') + \phi(a) = \phi(a+a')$ for every $a, a' \in F^k$. From this equality we can conclude by Observation 2.10 that indeed $\phi \in \mathrm{Img}(()H_k)$. □

This completes the proof of the Hadamard code's local testability.

2.3 What next?

Next time, we will formally define 'local decodability', and observe that the Hadamard code is locally decodable as well as locally testable. As mentioned in the introduction, local testability and local decodability properties combined will allow us to read bits of the Hadamard encoding with 'relative confidence'. In the next lecture we will complete the proof by showing that the Hadamard encoding of a CIRCUIT-SAT witness provides enough information to verify it in $O(1)$ queries with high soundness. Shortening the proof from exponential to polynomial will be our next goal.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	November 18th, 2005

Lecture 3: Hadamard local decodability and the language QCSP

Lecturer: Eli Ben-Sasson *Scribe: Dan Baum*

In this lecture we will continue to discuss properties of the Hadamard code. We will prove the local decodability of this code and use it to test various properties of the encoded message, using a constant number of queries into a (possibly corrupted) codeword. Ultimately, we will show that one can test the property of whether a message encodes a satisfying assignment to a given circuit, thus proving Theorem 2.1.

3.1 Locally decodable codes

An error-correcting code maps messages $x \in \{0,1\}^k$ to codewords $w \in \{0,1\}^n$ such that even if a small fraction of the bits of w are changed (resulting in w'), the message x can be uniquely and efficiently recovered from w'. Suppose we want to determine whether x has a certain property (For example: Is the sum of its bits even? Is the ith bit zero?). An easy way of answering our question is to recover x from w'. As we shall see below, some codes (including the Hadamard code) allow for deciding certain such properties of x without decoding x in its entirety, and even tolerating a certain part of the encoding being erroneous.

Definition 3.1 (Locally decodable codes)**.** Let C be a $[n,k,d]$-LECC, let $G \subseteq \{g : F^k \to F\}$ be a family of functions and let q, r, t be integers. An (q,r,t)-*decoder* for G from C is a randomized algorithm with one input and oracle access to a string $w \in F^n$. On input a (description of a) function $g \in G$, the decoder tosses r coins, runs in time t, makes q queries to w (each query is answered by an element of F) and outputs an element of F. Let $D^w[g;R]$ denote the output of the decoder D on input function g, randomness string R and oracle access to w.
For $c : [0,1] \times \mathbb{N}^+ \to [0,1]$, we say G is (q,r,t,c)-*locally decodable* from C if there exists a decoder D satisfying the following completeness requirement for all $x \in F^k, w \in F^n, g \in G$:

$$\Pr_R[D^w[g;R] = g(x)] \geq c(\Delta^*(w, C(x))$$

where $\Delta^*(\cdot,\cdot)$ denotes the relative Hamming distance.

Remark. Usually in the literature a code is called locally decodable if every message bit can be locally decoded (i.e. G in the above definition is the set of projections onto single bits of the message). However, for constructing PCPs we will need to locally-decode other functions of the message, thus requiring a broader definition.

We can now formally state in which sense the Hadamard code is locally-decodable. Recall $\mathcal{L}_k$ is the space of linear functions $\ell : \{0,1\}^k \to \{0,1\}$.

Lemma 3.2. *For all $k \in \mathbb{N}^+$, every linear function in $\mathcal{L}_k$ is $(2, k, O(k), 1-2\delta)$-locally-decodable from the Hadamard code H_k.*

Proof. Let $\ell_a(x) = \sum_i a_i x_i$ be a linear function that we wish to decode from supposed codeword $w \in \text{Image}(H_k)$. Our decoder for tosses k coins to select $R \in \{0,1\}^k$, queries w_R and w_{R+a} and outputs their sum $w_R + w_{R+a}$. Notice the query complexity, randomness and running time are as claimed. Regarding completeness, the key observation is that each query of D is a random (uniformly distributed) entry of w. Thus, if $\Delta^*(w, H_k(x)) \leq \delta$, then with probability at least $1-2\delta$, our supposed codeword w agrees with the uncorrupted codeword $H_k(x)$ on both its queries. In this case (by Observation Observation 2.10 our decoder outputs $\ell_a(x)$. This completes our proof. □

3.1.1 Constraining Hadamard codes to subspaces

We now use the local testability and decodability of the Hadamard code to learn whether a message lines in an arbitrary affine space $A < \{0,1\}^k$. Recall $V \subseteq F^k$ is called a *linear space* if every $a, b \in V$ and $\alpha, \beta \in F$ satisfy $\alpha a + \beta b \in V$. Recall $A \subseteq F^k$ is an *affine space* if it is a translation of a linear subspace, i.e. there exists a vector $u \in F^k$ such that $A = V + u = \{v + u \mid v \in V\}$. The dimension of A is the dimension of V. Finally, recall a d-dimensional affine space A can also be defined using a matrix $M \in F^{(k-d)\times k}$ and a vector $u \in F^{k-d}$ by the linear constraints $A = \left\{x \in F^k \;\middle|\; Mx = u\right\}$.

Definition 3.3 (Restricting the Hadamard code to affine spaces)**.** For $A \leq F^k$ let $H_k\big|_A = \{H_k(x) \mid x \in A\}$. Similarly, let $H_k|_{a,\alpha} = \left\{x \in F^k \;\middle|\; \sum_i a_i x_i = \alpha\right\}$.

Our next result is

Lemma 3.4. *For any d-dimensional affine space $A < F^k$, the code $H_k\big|_A$ is a locally testable code with parameters $(O(k), 3k-d, 4, \min\{s_{H_k}(\delta), 1-\delta/2\})$, where s_{H_k} is the soundness function from Theorem 2.9.*

To prove this Lemma, we start with a simpler version of it that deals only with affine spaces of co-dimension 1.

Lemma 3.5. *For every $a \in F^k$ and $\alpha \in F$, the code $H_k\big|_{a,\alpha}$ is a locally testable code with parameters $(O(k), 2k, 4, \min\{s_{H_k}(\delta/2), 1-\delta\})$, where s_{H_k} is the soundness function from Theorem 2.9.*

Proof. Let T, D denote the tester and decoder from Theorem 2.9 and Lemma 3.2, respectively. A tester $T_{a,alpha}$ for $H_k\big|_{a,\alpha}$ operates as follows:

- Choose randomly $b, c \in F^k$;
- If $T^w\left[1^k; b, c\right] = \textsf{reject}$, reject. Otherwise,
- Accept iff $D^w[\ell_a; b] = \alpha$.

T satisfies the randomness and running time constraints. Regarding query complexity, T reads w_b, w_c and w_{b+c} and D reads w_b and w_{a+b} — a total of four queries. As for soundness, let $\delta = \Delta^*\left(w, H_k\big|_{a,\alpha}\right)$. There are two possibilities:

1. $\Delta^*(w, H_k) \geq \delta/2$. In this case Theorem 2.9 implies

$$\Pr_{b,c}\left[T^w\left[1^k; b, c\right] = \mathsf{reject}\right] \geq s_{H_k}(\delta/2)$$

2. $\Delta^*(w, H_k) < \delta/2 \leq \frac{1}{2}$. In this case w is $\delta/2$-close to a word $w = H_k(x')$ with $\ell_a(x') \neq \alpha$; thus Lemma 3.2 implies

$$\Pr_b[D^w[\ell_a; b] \neq \alpha] \geq 1 - \delta$$

□

ER: *maybe beef this up a bit?*

Proof of Lemma 3.4. Let $A = \left\{x \in F^k \;\middle|\; Mx = u\right\}$. The tester T^w_A (with oracle w) operates as follows:

- Select $r \in F^{k-d}, b, c \in F^k$ uniformly and independenty.
- Let $a = r^\mathsf{T} M$ and $\alpha = r^\mathsf{T} u$.
- Invoke $T_{a,\alpha}$ from Lemma 3.5 with proof w and input 1^k.

The running time, randomness,query complexity and completeness constraints are satisfied by our definition of T^w_A; it remains to analyze the soundness. Assume $\Delta^*\left(w, H_k\big|_A\right) \geq \delta$. We follow the proof of Lemma 3.5: If $\Delta^*(w, H_k) \geq \delta/2$, by Lemma 3.5 the tester will rejectwith probability at least $s_{H_k}(\delta/2)$, satisfying the constraint. Otherwise, w is $\delta/2$-close to a word $w' = H_k(x)$, for some $x \notin A$. The crucial observation is that $Mx \neq u$, so with probability exactly $\frac{1}{2}$ (over r) we have $r^\mathsf{T} Mx \neq r^\mathsf{T} u$, i.e. $\sum a_i x_i \neq \alpha$. In this case, Lemma 3.2 implies $\mathbf{Pr}_b[D^w[\ell_a; b] = \mathsf{reject}] \geq 1 - \delta$. Since the event above occurs with probability $\frac{1}{2}$, and r is independent of the random b used by D, we conclude the rejection probability in this latter case is at least $(1 - \delta)/2$. □

3.2 Quadratic constraint satisfaction problems

We shall introduce a new language QCSP and use it as an intermediate step in proving Theorem 2.1.

Definition 3.6. An instance of QCSP (Quadratic Constraint Satisfaction Problem) is a set of polynomials over $\{0, 1\}$ on n variables with total degree ≤ 2 and the additional restriction that each polynomial contains at most three monomials. An instance is in the language i.f.f there exists an assignment to the n variables causing all polynomials to vanish (evaluate to zero).

Lemma 3.7. QCSP *is* **NP***-complete.*

Proof. We reduce from CIRCUIT-SAT. Recall an instance of CIRCUIT-SAT is a circuit, i.e. a sequence ψ of n gates $(g_1, \ldots, g_n)$ such that each gate is either an input gate or is one of the following

- AND: $g_i = g_j \wedge g_k = 0$ for $j, k < i$.
- NOT: $g_i = \neg g_k$ for $k < i$.

We reduce this circuit to the following instance of QCSP over variables $y_1, \ldots, y_n$. Our set of constraints is as follows. If $g_i = g_j \wedge g_k$ is an AND gate we add the constraint $y_i - y_j \cdot y_k = 0$ and if $g_i = \neg g_k$ is a NOT gate we add the constraint $y_i - y_k + 1$. Let ϕ be the collection of all constraints arising from the gates of the circuit ψ. It is readily observable that $\phi \in \text{QCSP} \Leftrightarrow \psi \in \text{CIRCUIT-SAT}$. □

Next we provide a version of QCSP in which all constraints are degree one polynomials.

Definition 3.8 (AFFINE-QCSP)**.** An instance of AFFINE-QCSP is an affine subspace of $\{0,1\}^{n^2}$ defined by a matrix $A \in \{0,1\}^{n^2 \times n^2}$ and a vector $u \in \{0,1\}^{n^2}$. An instance is in the language iff there exists a vector $y \in \{0,1\}^n$ such that $A\big(y \times y^{\mathsf{T}}\big) = u$ (where the n^2 coordinates of $y \times y^{\mathsf{T}}$ are considered as a column vector).

Lemma 3.9. AFFINE-QCSP *is* **NP***-complete.*

Proof. We reduce from QCSP. Consider an instance ϕ of QCSP with n variables and m constraints:

$$\left\{ \sum_{i,j=1}^{n} a_{ij}^{(k)} y_i \cdot y_j + u^{(k)} \right\}_{k \in \{1,\ldots,m\}}$$

Let A be the affine space defined as the set of n^2-dimensional vectors $z = \{z_{ij}\}_{i,j \in [n]}$ over $\{0,1\}$ such that

$$\sum_{i,j=1}^{n} a_{ij}^{(k)} z_{ij} + u^{(k)} = 0$$

for all $k \in \{1, \ldots, m\}$. Note that $y \in \{0,1\}^n$ satisfies ϕ iff $z = y \times y^{\mathsf{T}} \in A$. To see this, the crucial observation is that over $\{0,1\}$ we have $y_i^2 = y_i$, so $z_{ii} = y_i$. For all other entries, by definition we have $z_{ij} = y_i y_j$. This completes our proof. □

Next lecture we will present a PCP for an instance of AFFINE-QCSP which will essentially be the Hadamard encoding of y and of $z = y \times y^{\mathsf{T}}$, where y satisfies the QCSP instance and z satisfies the corresponding AFFINE-QCSP instance. This will complete the proof of Theorem 2.1.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	November 24th, 2005

Lecture 4: Basic PCP Wrap-Up; Dinur's Gap Amplification I

Lecturer: Eli Ben-Sasson *Scribe: Eyal Rozenberg*

4.1 Theorem 2.1 Redux and Final Proof

Reminder: We set out to prove Theorem 2.1:

$$\text{CIRCUIT-SAT} \in \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = 2^{\mathcal{O}(n2)} \\ \mathsf{randomness} & = \mathcal{O}\left(n^2\right) \\ \mathsf{query} & = 13 \\ \mathsf{time} & = \mathcal{O}\left(n^2\right) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = 1/200 \end{array}\right)$$

To achieve this, we have proven the following:

- The Hadamard code H_k is both locally testable (using 3 queries), and any linear function of a (k-bit long) message is decodable using 2 queries to its H_k encoding.
- For every affine subspace $A < \{0,1\}^n$, The Hadamard code $H_k\big|_A$ is locally testable (using 4 queries).

Given circuit C with n gates, we reduce it to an instance of Affine-QCSP of dimension n^2 as shown in the previous lecture. Let $A \subset \{0,1\}^{n^2}$ be the affine space resulting from this reduction. Recall C is satisfiable iff there exists $y \in \{0,1\}^n$ such that $z = y \times y^{\mathsf{T}} \in A$. Thus, we expect as our probabilistically checkable proof the Hadamard encoding of y and of z.

Consider first an honest prover, which sends valid encodings of $y \in \{0,1\}^n$ and $z \in \{0,1\}^{n^2}$. The verifier has to check:

1. Whether or not $z \in A$ (this guarantees that the polynomials are all satisfied, i.e. that the circuit is satisfied).
2. Whether y and z are consistent with each other, that is whether $z = y \times y^{\mathsf{T}}$; this is equivalent to checking whether the matrix $M = z - y \times y^{\mathsf{T}}$ is all-zero.

Assume $z \neq y \times y^{\mathsf{T}}$, or equivalently, $M \neq 0$. We wish to reject (without making many queries). To this end, suppose we select a random linear combination of the rows of M using the random coefficient vector $r_1 \in \{0,1\}^n$. This line is non-zero with probability at least 1/2 (because $M \neq 0$). We then select another random linear combination of the results row's cells using a second, independent random coefficient vector $r_2 \in \{0,1\}^n$. If $Mr_1 \neq 0$, then ${r_2}^{\mathsf{T}} M r_1 \neq 0$ (or, equivalently, $\ell_{r_1}(y) \cdot \ell_{r_2}(y) \neq \ell_{\left(r_1 \times {r_2}^{\mathsf{T}}\right)}(z)$) with probability 1/2 . Thus with probability at least 1/4, our verifier rejects an inconsistent y, z pair.
We are finally ready to complete the proof.

Proof of Theorem 2.1. Given an input a circuit C with n gates, the verifier first reduces it to an instance of Affine-QCSP, of dimension n^2. Let $A \subseteq \{0,1\}^{n^2}$ be the resulting affine space; we recall C is satisfiable iff there exists $y \in \{0,1\}^n$ such that $z = y \times y^{\mathsf{T}} \in A$. The proof oracle is a concatenation some $Y \in \{0,1\}^{2^n}$ and $Z \in \{0,1\}^{2^{n^2}}$. The verifier conducts the following tests, accepting iff all sub-tests accept:

- Invoke the tester for the n-dimensional Hadamard code H_2 from Theorem 2.9 on oracle Y.
- Invoke the tester for $H_{n^2}\big|_A$ from Theorem 3.4 on oracle Z.
- Randomly and independently select $r_1, r_2 \in \{0,1\}^n$; Invoke the local decoder from Lemma 3.2 three times accepting iff

$$D^Y[\ell_{r_1}; R] \cdot D^Y[\ell_{r_2}; R] = D^Z\left[\ell_{r_1 \cdot r_2{}^{\mathsf{T}}}; R\right]$$

The proof length, running time, and completeness parameters all meet the theorem requirements, by definition of our verifier. The randomness used in all LD-reads and LTC-tests, as well as the selection of r_1, r_2, is $\Theta(n)$. The number of queries is:

	3	for LTC-testing that $Y \in H_n$
+	4	for LTC-testing that $Z \in H_{n^2}\big\|_A$
+	$3 \cdot 2$	for locally decoding ℓ_{r_1}, ℓ_{r_2} and $\ell_{r_1 \cdot r_2{}^{\mathsf{T}}}$
	13	in total

As usual, the more involved argument regards the **PCP** system's soundness. Suppose $C \notin$ CIRCUIT-SAT; this implies one of the following holds:

1. Y is 1/100-far from H_n.
2. Z is 1/100-far from $H_{n^2}\big|_A$.
3. Y, Z are 1/100-close to some valid y, z, but these two are inconsistent, i.e. $z \neq y \cdot y^{\mathsf{T}}$.

If either of the first or second case holds, then the relevant LTC-test rejects with probability at least 1/200. Otherwise, the third case holds. With probability at least 1/4, the choice of r_1, r_2 is such that

$$\ell_{r_1}(y) \cdot \ell_{r_2}(y) \neq \ell_{(r_1 \times r_2{}^{\mathsf{T}})}(z)$$

For such choices, local decodability implies that with probability at least $1 - 3 \cdot 2/100$ we have $D^Y[r_1] = \ell_{r_1}(y), D^Y_{r_2} = \ell_{r_2}(y)$ and $D^Z[\ell_{r_1 \cdot r_2{}^{\mathsf{T}}}] = \ell_{(r_1 \times r_2{}^{\mathsf{T}})}$, so verifier rejects. Thus in all cases the verifier rejects with probability at least 1/200, completing the proof. □

Observation 4.1. We could have inferred Y from Z, since $z = y \times y^{\mathsf{T}}$ implies that the diagonal of z is y ($y_i \cdot y_i = y_i$ in $\{0,1\}$). This would have eliminated the need for testing Y, reducing the number of queries to 10. The query complexity could be reduced further by reusing bits from the LTC-test for LD-reads.

Observation 4.2. We have proven the theorem with a soundness value which is far from being optimal, even with the methods used in our proof.

4.2 Reducing the Proof Length – Two Approaches

The question now arises, how to avoid the use of such incredibly long proofs. A possible course of action would be replacing the linear Hadamard code with different, terser codes, such as Reed-Solomon or Reed-Müller codes.

Example 4.3. Suppose we wish to encode the word $w = 010110$ of length $d = 5$. Let us choose some field $\mathbb{F}_k$ with $k > 6$, and select the polynomial $p(x) \in x[\mathbb{F}_k]$ of degree $|w| - 1$ uniquely defined by the constraints $p(i) = w_i$ for all i's: We're setting its values on $\{0, \ldots, 6\}$ and interpolating them on the rest of the elements of $\mathbb{F}_k$.

The problem with this approach is, that one must read $\Theta(d)$ (for the example, $d+2$) values of $p(x)$ to be able to test if a function is close to a codeword. I.e. the query complexity is too large. Traditionally, this problem was overcome by *composing* PCP systems: First, one would prove the existence of a PCP using short proofs (e.g. $\text{poly}(n)$) but many queries (e.g. $\sqrt{n}$). One would then apply a PCP system to the previous PCP-verifier's circuit as the input. With each repeated PCP application, the number of queries would decrease again (e.g. to $n^{1/4}$, $n^{1/8}$ etc.)
Of course, the PCP system must meet certain conditions in order to allow for composition and one had to make sure not to lose too much in terms of soundness along the way.

Remark. In [3], three compositions were used:

$$n \longrightarrow \left(\log(n)^{10}\right) \longrightarrow \log\left(\log(n)^{10}\right)^{10} = \left(10\log(\log n)\right)^{10} \longrightarrow \text{poly}(\log\log\log n)$$

At this point, even the application of Hadamard encoding does not necessitate a proof longer than $\mathcal{O}(\log(\log(n)))$.

Dinur took a different course of action. First, she uses many repeated compositions. Secondly, she begins with a low soundness value, increasing by a constant factor with further compositions, rather than beginning with a high soundness value and limiting its decrease. To take this approach she has to overcome a different problem: 'simple' soundness amplification means repeating a verification several times, resulting in the use of more randomness, and consequently increasing the proof size (and requiring consistency checking for the results of repeated verifications). Dinur's solution is an amlification of the soundness by a modest factor, but in a randomness-parsimonious manner, followed by a PCP-of-proximity (this will be defined later on) for verifying the consistency of the compositions.

4.3 Dinur's Gap Amplification

Theorem 4.4 (Irit Dinur, [17])**.** *For every sufficiently large alphabet Σ (it seems $|\Sigma| \geq 3$ suffices) there exists $\alpha(|\Sigma|) > 0$ such that for all $a \in \mathbb{N}^+$,*

$$\mathbf{PCP}\begin{pmatrix} \text{length} & = & \ell(n) \\ \text{randomness} & = & r(n) \\ \text{query} & = & 2 \\ \text{time} & = & t(n) \\ \text{completeness} & = & 1 \\ \text{soundness} & = & s(n) \end{pmatrix} \subseteq \mathbf{PCP}\begin{pmatrix} \text{length} & = & \mathcal{O}(\ell(n)) \\ \text{randomness} & = & r(n) + \mathcal{O}(1) \\ \text{query} & = & 2 \\ \text{time} & = & \mathcal{O}(t(n)) \\ \text{completeness} & = & 1 \\ \text{soundness} & = & \alpha \cdot \min\{\sqrt{a} \cdot s(n), 1/\sqrt{a}\} \end{pmatrix}$$

with the $\mathcal{O}(\cdot)$ constants dependent on $|\Sigma|$ and a.

Observation 4.5. Why the restriction on $|\Sigma|$? With only 2 queries and $|\Sigma| = 2$, the power of a **PCP** system cannot exceed **P**, as its acceptance is a 2SAT formula. If the number of queries is 3, such as in the 'basic PCP' Theorem 2.1, the constructions of the proof below would involve 3-hypergraphs rather than graphs, which would make them much more cumbersome. Thus the alphabet size must be increased.

Notice that the basic PCP Theorem is an immediate corollary.

Corollary 4.6. *There exists constant $\alpha' > 0$ and finite alphabet Σ such that*

$$\mathbf{NTIME}(t(n)) \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = t^{\mathcal{O}(1)} \\ \mathsf{randomness} & = \mathcal{O}(\log(t(n))) \\ \mathsf{query} & = 2 \\ \mathsf{time} & = t^{\mathcal{O}(1)} \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = \alpha' \end{array}\right)$$

Proof. Let $L \in \mathbf{NTIME}(t(n))$ and $\varphi \in L$ with $|\varphi| = n$. We reduce φ to a 3COL instance of size $n' = t(n')^{\mathcal{O}(1)}$ (n' is $|E(G)|$ in the graph to be colored).
Now, we can prove the colorability using a witness which is a non-encoded coloring of the vertices. If the graph is not 3-colorable, every proof induces a monochromatic edge. A verifier can select an edge randomly, read the coloring of the incident vertices from the proof and accept iff the edge is non-monochromatic. An uncolorable graph would be reject with probability at least $1/n'$. Thus

$$3\mathrm{COL} \in \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = \mathcal{O}(n') \\ \mathsf{query} & = 2 \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = 1/n' \end{array}\right)$$

with the other parameters as stated above. We now apply Theorem 4.4 to φ, $2\log_a(n')$ times, achieving PCP parameters

$$\begin{aligned} \mathsf{time} &= c^{\log_a(n')} \cdot t(n') = t^{\mathcal{O}(1)}(n) \\ \mathsf{length} &= c^{\log_a(n')} \cdot \mathcal{O}(n') = t^{\mathcal{O}(1)}(n) \\ \mathsf{randomness} &= \log(n') + \left(\log_a(n')\right) \cdot c = \mathcal{O}\left(\log(n)\right) \\ \mathsf{completeness} &= 1 \end{aligned}$$

and as for the soundness,

$$\mathsf{soundness} = \min\left\{\frac{1}{n'} \cdot \sqrt{a}^{2\log_a(n')}, \alpha/\sqrt{a}\right\} = \alpha/\sqrt{a} \qquad \stackrel{\text{def}}{=} \qquad \alpha' > 0$$

□

Question. How far can one hope to raise the soundness α? Andrej Bogdanov showed in [14] that with Dinur's methods, it is impossible to exceed $\alpha = \frac{1}{2}$ (this will part of homework assignment 2).

Before proceeding to prove Theorem 4.4, we shall describe the problem again in terms of coloring of graphs.

Definition 4.7. A *constraint graph* $\mathcal{G}$ is a triplet $(G, \Sigma, \mathcal{C})$, where $G = (V, E)$ is an undirected multigraph with self-loops allowed, Σ is a finite alphabet, and $\mathcal{C}$ are per-edge constraints:

$$\mathcal{C} = \{C_e : \Sigma \times \Sigma \to \{\mathsf{accept}, \mathsf{reject}\} \mid e \in E(G)\}$$

these constrain the possible colorings of the two incident vertices of each edge.

Definition 4.8. An *assignment* for a constraint graph $\mathcal{G}$ is a function $A : V(G) \to \Sigma$ which defines a coloring of the vertices: The *soundness of an assignment* A is

$$s(\mathcal{G}, A) = \mathop{\mathbf{Pr}}_{(u,v)\in E(G)}\left[C_{(u,v)}(A(u), A(v)) = \mathsf{reject}\right]$$

Definition 4.9. The *soundness of a constraint graph* $\mathcal{G}$ is

$$s(G) = \min_A s(\mathcal{G}, A)$$

Definition 4.10. A *promise problem* is a pair of languages $\textsc{Yes}, \textsc{No} \subseteq \Sigma^*$, satisfying $\textsc{Yes} \cap \textsc{No} = \varnothing$. If $\textsc{Yes} \cup \textsc{No} = \Sigma^*$, the pair constitutes a *decision problem*.

Definition 4.11. Let $\hat{s} : \mathbb{N}^+ \to [0, 1]$ be a soundness function. The problem GAP-CG(Σ, s) is the promise problem with

$$\begin{aligned}\textsc{Yes} &= \{\mathcal{G} = (G, \Sigma, \mathcal{C}) \mid s(G) = 0\}\\ \textsc{No} &= \{\mathcal{G} = (G, \Sigma, \mathcal{C}) \mid s(G)) \geq \hat{s}(|E(G)|)\}\end{aligned}$$

Thus the Yes graphs are those which are completely satisfiable, and the No graphs are those for which you cannot satisfy more than some fraction of the constraints.

Proposition 4.12 (restatement of Theorem 4.4)**.** *For every* Σ *with* $|\Sigma| \geq 3$ *there exists* $\alpha(|\Sigma|) > 0$ *such that for every* $a \in \mathbb{N}^+$*,*

$$\text{GAP-CG}(\Sigma, s(n)) \xrightarrow[\text{reduction}]{\text{polynomial}} \text{GAP-CG}\left(\Sigma, \alpha \cdot \min\left\{\sqrt{a} \cdot s(n), 1/\sqrt{a}\right\}\right)$$

with the $\mathcal{O}(\cdot)$ *constants dependent on* $|\Sigma|$ *and* a*.*

Proof Sketch. The proof will be comprised of three phases:

Phase I. We 'beautify' $\mathcal{G}$: We transform it into a constraint graph $\mathcal{G}'$ which is d-regular graph for some d, has self-loops, and is an expander. This makes it 'behave' similarly to a random graph in some respects. This phase decreases the soundness and increases the size.

Phase II. We apply a 'powering' to $\mathcal{G}'$, resulting in $\mathcal{G}''$ over alphabet Σ''. This phase increases the soundness, at the cost of an increased alphabet size: $|\Sigma''| > |\Sigma|$.

Phase III. We perform an alphabet-reduction phase, returning from the larger Σ'' to Σ, by composition with a PCP-of-proximity. This phases reduces the soundness again.

Accounted over all three phases, the parameters will have changed as the theorem specifies.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	December 1st, 2005

Lecture 5: Dinur's Gap Amplification II: Graph beautification

Lecturer: Eli Ben-Sasson *Scribe: Anat Paskin*

5.1 High-level proof

Recall that last time we stated a gap amplification theorem (Theorem 4.4), from which the basic **PCP** theorem follows as an immediate corollary. It was also mentioned that the construction consists of a sequence of three reductions. In this section, we precisely define and parameterize each of the reductions, and show how they fit together (which is rather straightforward). In the next sections we explicitly construct and analyze the first and third reductions. The more technically involved step, where the actual soundness amplification is done, will be analyzed in the next lecture. In the following discussion we sometimes refer to the size of E as the size of a graph $G = (E, V)$.

Definition 5.1 (Expander)**.** A (d, λ)-expander, is a d-regular undirected multigraph $G = (V, E)$ with at least one self loop per vertex, with the following property. For every set of edges $F \subseteq E$, and for all $i \geq 0$, it holds that a random walk that starts at a random edge in F, makes its $(i+1)$st step in F with probability $p \leq |F|/|E| + \lambda^i$.

We say a graph has 'meaningful' expansion according to this definition only when $\lambda < 1$.

Definition 5.2. Expander-GAP-CG$(n, \Sigma, s, (d, \lambda))$ is the restriction of the promise problem GAP-CG(n, Σ, s) to instances over (d, λ)-expander graphs.

We are now ready to present a high-level proof of Proposition 4.12 in a more detailed manner.

Lemma 5.3. *There exists constants $\beta = \beta(d, |\Sigma|) >, d, c_1, c_3 \geq 1$ and finite alphabet Σ_0 (independent of $|\Sigma|$), such that for all $a \in \mathbb{N}^+$ the following holds.*

Phase I. Preprocessing:

$$\text{GAP-CG}(n_0, \Sigma, \hat{s}_0) \xrightarrow{\substack{\text{polynomial} \\ \text{reduction}}} \text{Expander-GAP-CG}(n_1 = O(n_0), \Sigma, \hat{s}_1 \geq \hat{s}_0/c_1, (d, 1/2))$$

Phase II. Powering:

$$\text{Expander-GAP-CG}\left(n_1, \Sigma, \hat{s}_1(n), (d, \frac{1}{2})\right) \xrightarrow{\substack{\text{polynomial} \\ \text{reduction}}}$$

$$\text{GAP-CG}\left(n_2 \leq n_1 \cdot d^a, \Sigma^{d^a}, \beta \cdot \min(\hat{s}_2(n) \cdot a, 1/\sqrt{a})\right)$$

Phase III. Alphabet-reduction:

$$\text{GAP-CG}(n_2, \Sigma, \hat{s}_2) \xrightarrow[\text{reduction}]{\text{polynomial}} \text{GAP-CG}\left(n_3 \leq n_2 \cdot 2^{|\Sigma_0|^{d^{O(a)}}}, \Sigma_0, \hat{s}_3 \geq \hat{s}_2/c_3\right)$$

Assuming Lemma 5.3 holds, we conclude the proof of Proposition 4.12 by selecting $\sqrt{a} \geq 2c_1c_3/\beta$. Applying the reductions (one after the other, starting from the first) results in a soundness amplification by a factor of 2, and it's trivial to see that the other parameters are changed as claimed. Note that the graph resulting after applying the alphabet reduction step will be over the original alphabet is $\leq \Sigma_0$ (think of Σ_0 as a small constant).

5.2 Preprocessing Construction

We now describe the details of the preprocessing reduction. Recall that the purpose of this stage is to create a 'nicely-formed' constraint graph, suitable to serve as an input to the powering step. The reduction proceeds as follows.

Given an instance $\mathcal{G} = (G = (V, E), \Sigma, \mathcal{C})$, we reduce it to an expander graph with the proper parameters in two stages, summarized in the following two lemmas. First, we turn it into a d_0-regular graph (Lemma 5.4) and then add d_1 edges to each vertex to make it a $(d = d_0 + d_1, \frac{1}{2})$-expander (Lemma 5.7), for some constants d_0, d_1. As usual, the (somewhat) tricky part is ensuring the soundness doesn't drop by more than a constant factor in each stage.

Lemma 5.4. *There exist constants $d_0, c > 0$, such that any constraint graph $\mathcal{G} = (G = (V, E), \Sigma, \mathcal{C})$ can be efficiently transformed into a d_0-regular constraint graph $\mathcal{G}' = (G' = (V', E'), \Sigma, \mathcal{C}')$, with $|V'| = 2|E|$, and $s(G') \geq s(G) \cdot c$.*

This lemma is an 'expander-replacement' transformation, due to [38].

Proof. We first present without a proof a few additional well-known properties of expander graphs.

Definition 5.5. A graph G is a (d, e)-edge-expander, if G is d-regular and for all $S \subseteq V, |S| \leq |V|/2$, it holds that $\left|E(S, \bar{S})\right| \geq e \cdot |S|$.

We say a graph has 'meaningful' expansion according to this definition when $e > 0$. Note that definitions 5.1,5.5 both define the same object, referred in the literature as 'expander'. These definitions are related (by analyzing the second eigenvalue of the adjacency matrix of a graph) in the sense that a graph with meaningful expansion according to the first definition also has meaningful expansion according to the second definition and vice versa. We use the fact that expanders can be efficiently constructed (proving this fact is highly non-trivial [35, 23, 34, 42], but beyond the scope of this course).

Theorem 5.6. *There exists constant $d \in \mathbb{N}$, such that there exists a polytime constructible family of graphs $\{X\}_n$ (for all large enough n), where each X_n is a $(d, 2)$-edge-expander of size n.*

The construction. Given $\mathcal{G} = (G = (V, E), \Sigma, \mathcal{C})$ construct a new constraint graph $\mathcal{G}'$, by replacing each node in G by an instance $G'_v = (V'_v, E'_v))$ of the graph X_{d_v}, where d_v is the degree of v in G, and put an equality constraint on every internal edge of G'_v[3], we refer to edges of E'_v as 'internal'. Additionally, for every $e = (v, u) \in E$, draw an edge between a unique $v' \in V'_v$ and a unique $u' \in V'_u$ and place on this edge the constraint on (u, v) appearing in $\mathcal{G}$, we refer to such edges as 'external'. Finally, add a self-loop (with a constraint that always accepts), to obtain a $d + 1$-regular graph.

Analysis. The completeness of the construction is easy. This is the case, since a satisfying assignment to V, can be extended to a satisfying assignment to V' in the straightforward way. To prove the soundness property, we let $\sigma' : V' \to \Sigma$ denote the best assignment to V' (that is, one yielding an unsat value of $s(G')$). Let σ denote an assignment to V, generated by letting $\sigma(v)$ assume the most frequent value among $\sigma'(v')$, over $v' \in V'_v$. By definition, σ is rejected by $s_1(G) \geq s(G)$ of the constraints in C. Let $F \subseteq E$ denote the set of edges rejected by σ, and $F' \subseteq E'$ denote the set of edges rejected by σ' respectively. Also let S'_v denote the set of vertices in V'_v which disagree with $\sigma(v)$. Let $S' = \bigcup_{v \in V} S'_v$. Consider an external edge e' in E' corresponding to an edge $(u, v) \in F$. The key observation is that if $e' \notin F'$, then it has at least one endpoint in S'. Therefor, every $e \in F$ either corresponds (the correspondence is induced by the last observation) to an edge $e' \in F'$, or a vertex (or two vertices) $v' \in S'$. Since all external edges are disjoint in vertices, we conclude that

$$|F| \leq \left|S'\right| + \left|F'\right| \tag{5}$$

There are two possible cases:

1. If $|F'| \geq |F/2|$, then we conclude the proof with $c = 1/2 \cdot (d + 1)$.
2. Otherwise, we have $|S| \geq |F|/2$ by Equation 5. By definition, every S'_v satisfies $|S'_v| \leq |V'_v|/2$. By the properties of $\{X\}_n$, we have $\left|E'(S'_v, \bar{S'_v})\right| \geq 2 \cdot |S'_v|$. Therefor, S' induces a set of (internal) edges rejecting σ' of size at least $\frac{1}{2}\Sigma_{v \in V} 2|S'_v| = |S'| \geq |F|/2 \geq |E| \cdot s(G)/2 = \frac{|E'| \cdot s(G)}{2 \cdot (d+1)}$.

We conclude the lemma holds with $d_0 = d + 1$,$c = 1/2 \cdot (d + 1)$. □

Next, we show how to turn the graph into an expander.

Lemma 5.7. *There exists a constant d_1, such that any d_0-regular constraint graph $\mathcal{G} = (G = (V, E), \Sigma, \mathcal{C})$ (with large enough d_0, n) can be efficiently transformed into a $(d_0 + d_1)$-regular constraint graph $\mathcal{G}' = (G' = (V', E'), \Sigma, \mathcal{C}')$, with a self-loop in every vertex, where G' is a $(d_0 + d_1, \frac{1}{2})$-expander, and $s(G') \geq s(G) \cdot d_0/d_0 + d_1$.*

Proof. We only present the construction and prove the soundness property, without proving the expansion property of the new graph. Given $\mathcal{G} = (G = (V, E), \Sigma, \mathcal{C})$ over n vertices, take a (d_1, λ)-expander, for sufficiently small λ. The existence of such an expander is promised by Theorem 5.6. We put empty constraints on all the new edges (and keep the constraints of $\mathcal{G}$ on the original edges). Since the number of edges increased from d_0 to $d_0 + d + 1$, we

[3] We assume w.l.o.g. that the degree of every vertex is large enough to construct X_{d_v}

pick an original edge with probability $d_0/(d_0+d_1)$, which implies that the soundness drops by the same factor. Proving the resulting graph is indeed a $(d_0+d_1, \frac{1}{2})$-expander is not very hard, but it requires additional knowledge about expanders that is beyond the scope of this course. □

Combining the results of the last pair of lemmas, we immediately obtain a reduction from GAP-CG to Expander-GAP-CG, parameterized as specified in the preprocessing step of Lemma 5.3.

5.3 Composition

As mentioned before, we postpone the exposition and analysis of the gap-amplification stage to the next lecture, and proceed to the composition (alphabet-reducing) reduction.

The core of the construction is the concept of **PCP**'s of proximity (also known as 'assignment testers').

Definition 5.8. Let Σ denote a finite alphabet. A *pair language* $L \in \Sigma^* \times \Sigma^*$.

Example 5.9. The pair language Circuit-Val consists of all pairs (C, y) where C is a boolean circuit, y is an assignment to its input gates and $C(y) = 1$. Formally, Circuit-Val $=\{(C, y) : C(y) = 1\}$.

We will be interested in *pair languages* induced by a language $L \in \mathbf{NTIME}(t(n))$ according to the deterministic machine M_L deciding R_L (where R_L is a $poly(t(n))$ bounded relation defining L), by letting $pair - L = \{(x, y) | M_L(x, y) = \mathsf{accept}\}$. Also, let $L_x = \{y | M_L(x, y) = \mathsf{accept}\}$. We will assume w.l.o.g. x includes a description of the length of y. This assumption can be made w.l.o.g. because, if unspecified, we set the size of y to be $t(|x|)$.

Definition 5.10 (PCPP Verifier)**.** Let $\ell, r, q, t : \mathbb{N}^+ \to \mathbb{N}^+$. A $(\ell(n), r(n), q(n), t(n))$-*PCPP verifier* is a probabilistic machine V with oracle access to an *implicit* input y and a proof Π. On explicit input (x, N) with $|x| = n$, and N an integer (specifying $|y|$), verifier V runs in time $t(n)$, tosses $r(n)$ coins, makes $q(n)$ queries to y of size N and to π of size $\ell(n)$ and outputs either accept or reject.

Definition 5.11 (PCPP)**.** Let $\ell, r, q, t : \mathbb{N}^+ \to \mathbb{N}^+$ and $s : \mathbb{N}^+ \to [0, 1]$,$c : \mathbb{N}^+ \to [0, 1] \times \mathbb{N}^+$.

$$\mathbf{PCPP}\begin{pmatrix} \mathsf{length} & = & \ell(n) \\ \mathsf{randomness} & = & r(n) \\ \mathsf{query} & = & q(n) \\ \mathsf{time} & = & t(n) \\ \mathsf{completeness} & = & c(n) \\ \mathsf{soundness} & = & s(\delta, n) \end{pmatrix}$$

is the class of all pair languages L for which some (ℓ, r, q, t)-PCPP verifier V satisfies:

- Completeness: $y \in L_x \Rightarrow \exists\, \Pi \left[\mathbf{Pr}_R\left[V^{y,\Pi}[x; R] = \mathsf{accept}\right] \geq c(n)\right]$
- Soundness: $\Delta^*(y, L_x) > \delta \Rightarrow \forall\, \Pi \left[\mathbf{Pr}_R\left[V^{y,\Pi}[x; R] = \mathsf{accept}\right] \leq 1 - s(\delta, n)\right]$

We point out that our Hadamard based, exponential length PCP (Theorem 2.1) can be easily converted into a PCPP for Circuit-Val.

Theorem 5.12.

$$\textsc{Circuit-Val} \in \mathbf{PCPP}\left(\begin{array}{ll} \mathsf{length} & = 2^{n^2} \\ \mathsf{randomness} & = O(n^2) \\ \mathsf{query} & = 16 \\ \mathsf{time} & = n^{O(1)} \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = \min(1/200, 0.98\delta) \end{array}\right)$$

Proof Sketch. Given a circuit description C (in binary), we define the expected proof (y, Π) as follows. y denotes a satisfying assignment of C, and Π denotes a proof that C is satisfiable, structured as the valid oracle in the proof of Theorem 2.1. Furthermore, we expect Π to encode the assignment y, rather than some other satisfying assignment of C. To verify that both these conditions hold, we first run the verifier from Theorem 2.1, and reject upon failure. Otherwise, select an arbitrary index i, read y_i, and compare it with the decoding of y_i from Π, accept iff they are equal. It's rather straightforward from Theorem 2.1 that the above construction yields ℓ, t, r, c as required in Theorem 5.12. Note that we make 3 additional queries in order to verify that Π actually encodes y, so we make at most $13 + 3 = 16$ queries to the oracle. To see that the soundness requirement holds, consider two cases. If Π is 1/100-far from an encoding of any satisfying assignment to C, then the first test will reject with probability at least 1/200. Otherwise, Π contains a Hadamard encoding of a word, which is 1/100-close to a Hadamard encoding of a satisfying assignment of C. Since y is δ-far from a satisfying assignment of C, it's also δ-far from the (unique) assignment w encoded by Π. Therefor, with probability at least δ, the verifier will select a bit i on which y and w differ, and will recover w_i with probability at least 0.98 (independently of the bit selected). Overall, we get $s(n) \geq 1/200, 0.98 \cdot \delta$, as required.

In the next lecture we will complete the proof of the alphabet-reduction step. The idea would be encoding the assignment to each vertex (over some large alphabet Σ') in our input constraint graph by a good error correcting code, with constant rate, and constant relative distance (this step is called robustization) We then replace each constraint by a circuit (over small Σ), and run the tester above on (the encoding of) each constraint separately (composition). Since Σ is of constant size, we can afford ourselves exponentially large circuits, and using such a long **PCPP** proof. Eventually, we use a standard technique to reduce the number of queries to 2 (while reducing soundness by a constant factor), to obtain a new constraint graph with small alphabet (independent of Σ').

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	December 1st, 2005

Lecture 6: Dinur's Gap Amplification III: Alphabet reduction

Lecturer: Eli Ben-Sasson *Scribe: Dan Baum and Eli Ben-Sasson*

6.1 Introduction

In this lecture we will discuss and prove the third phase of Lemma 5.3, namely:

$$\text{GAP-CG}\left(n_2, \Sigma^{d^a}, \hat{s}_2\right) \xrightarrow[\text{reduction}]{\text{polynomial}} \text{GAP-CG}\left(n_3 \leq n_2 \cdot 2^{|\Sigma|^{d^{\mathcal{O}(a)}}}, \Sigma, \hat{s}_3 \geq \hat{s}_2/c_3\right)$$

Let $\mathcal{G}$ be an instance of $\text{GAP-CG}\left(n_2, \Sigma^{d^a}, \hat{s}_2\right)$ over graph G. Recall our verifier expects an assignment $\sigma : V \to \Sigma^{d^a}$, picks a random edge $(u, v) \in E(G)$ and applies the constraint $C_{(u,v)}$ to the pair of inputs $\sigma(u), \sigma(v) \in \Sigma^{d^a}$. We need to reduce the alphabet size, while keeping the soundness relatively high.
In order to do this we apply an efficiently encodable and decodable error correcting code $\mathcal{C} : \Sigma^{d^a} \to \Sigma^{\mathcal{O}(d^a)}$ and obtain code words over alphabet Σ. This way we have a smaller alphabet (Σ), but the number of queries is even larger than d^a. To reduce the query complexity we compose our verifier with the PCPP-verifier introduced in the previous lecture in Theorem 5.12. Recall a PCPP-verifier tests (using a small number of queries) if an input is close to an assignment that satisfies a predefined circuit. In our case, the circuit that we wish to test proximity to, is the circuit that accepts iff it's input corresponds to pair of code-words that decode to elements in Σ^{d^a} that satisfy $C_{(u,v)}$. The crucial observation is that since we encoded our inputs, testing proximity to a satisfying assignment can replace the actual testing of the assignment itself, thus reducing the query complexity. Details follow.

6.2 Composition Theorem

PCPPs (also known as Assignment Testers) were introduced by [18, 13] to facilitate composition. The idea of proof composition, introduced to the world of PCPs by [4] is as follows. Instead of having an *outer verifier* make a relatively large number of queries (in our case d^a queries), it will invoke an *inner verifier* to test that the input is *close* to satisfying a constraint (in our case $C_{(u,v)}$). This notion of a verifier invoking a different verifier is called *proof composition*. For composition to apply nicely, we require the outer verifier to be *non-adaptive*, i.e. its queries and decision predicate depend only on the input x and coin tosses R (but not on previous queries). This way, the outer verifier can easily set up the input for the inner verifier. Additionally, to ensure soundness does not deteriorate greatly, we require our inputs (to the outer verifier) to be encoded by some good error correcting code (by 'good' we mean the code has large minimal distance and rate and can be encoded and

decoded by a polynomial size circuit). We note that 'good' codes abound (but we don't go into details of constructing them). This leads to the following restricted form of PCP verifiers and languages that are a special case of Definitions 1.3, 1.4.

Definition 6.1 (Non-adaptive PCPs over coded proofs)**.** Let $\ell, r, q, t, m : \mathbb{N}^+ \to \mathbb{N}^+$ and let $\mathcal{C}$ be a $[n', k', d']_\Sigma$-ECC. A $(\ell(n), r(n), q(n), t(n), \mathcal{C}, m(n))$-*coded PCP verifier* is a probabilistic Turing machine V with oracle access to a proof Π of length $\leq \ell(n)$. On explicit input $x, |x| = n$, verifier V runs in time $t(n)$, tosses $r(n)$ coins (let R denote the string of coin-tosses), and outputs a pair:

- An *index* set $I = I_{(x;R)}$ of $q(n)$ query locations in Π. Formally: $I \in \{1, \ldots, \ell(n)\}^{q(n)}$.
- A *decision* circuit $D = D_{(x;R)}$ of size $\leq m(n)$ with $q(n)$ input blocks, each block a length n' string over alphabet Σ. The output of the circuit belongs to $\{\mathsf{accept}, \mathsf{reject}\}$ and has the further restriction that if any one of the $q(n)$ inputs is *not* a $\mathcal{C}$-codeword, then the output of D is reject.

For completeness and soundness functions $c, s : \mathbb{N}^+ \to [0, 1]$, we say a language L belongs to the class of pair languages

$$\textbf{coded-PCP}\left(\begin{array}{ll} \mathsf{length} & = \ell(n) \\ \mathsf{randomness} & = r(n) \\ \mathsf{query} & = q(n) \\ \mathsf{time} & = t(n) \\ \mathsf{code} & = \mathcal{C} \\ \mathsf{decision\text{-}size} & = m(n) \\ \mathsf{completeness} & = c(n) \\ \mathsf{soundness} & = s(\delta, n) \end{array}\right)$$

if there exists a $(\ell(n), r(n), q(n), t(n), \mathcal{C}, m(n))$-coded PCPP verifier V such that the following holds for all x, letting $\Pi\big|_I$ denote the restriction of proof Π to index set I:

- Completeness: $x \in L \Rightarrow \exists\, \Pi \left[\mathbf{Pr}_R\left[D_{(x;R)}(\Pi\big|_{I_{(x;R)}}) = \mathsf{accept}\right] \geq c(n)\right]$
- Soundness: $x \notin L \Rightarrow \forall\, \Pi \left[\mathbf{Pr}_R\left[D_{(x;R)}\right](\Pi\big|_{I_{(x;R)}}) = \mathsf{accept}\right] \leq 1 - s(n)\Big]$

With this Definition in hand, we are ready to state the composition Theorem that will be used to complete our proof (of phase III of Lemma 5.3).

Theorem 6.2 (PCPP Composition)**.** *Let*

$$L \in \textbf{coded-PCP}\left(\begin{array}{ll} \mathsf{length} & = \ell(n) \\ \mathsf{randomness} & = r(n) \\ \mathsf{query} & = q(n) \\ \mathsf{time} & = t(n) \\ \mathsf{code} & = \mathcal{C} \\ \mathsf{decision\text{-}size} & = m(n) \\ \mathsf{completeness} & = c(n) \\ \mathsf{soundness} & = s(n) \end{array}\right)$$

for some $[n', k', d']_\Sigma$*-LECC* $\mathcal{C}$ *with* $d' > n'/3$*. Suppose additionally*

$$\text{CKT-VAL} \in \mathbf{PCPP}\left(\begin{array}{ll} \mathsf{length} & = \ell'(n) \\ \mathsf{randomness} & = r'(n) \\ \mathsf{query} & = q'(n) \\ \mathsf{time} & = t'(n) \\ \mathsf{completeness} & = c'(n) \\ \mathsf{soundness} & = s'(\delta, n) \end{array}\right)$$

with monotone non-decreasing soundness function $s'(\delta, n)$*, i.e. for all* n *and* $\delta \geq \delta'$ *we have* $s'(\delta, n) \geq s'(\delta', n)$*. Then* L *belongs to*

$$\mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = \ell(n) + 2^{r(n)} \cdot \ell'(m(n)) \\ \mathsf{randomness} & = r(n) + r'(m(n)) \\ \mathsf{query} & = q'(m(n)) \\ \mathsf{time} & = t(n) + t'(m(n)) \\ \mathsf{completeness} & = c(n) \cdot c'(m(n)) \\ \mathsf{soundness} & = \frac{1}{2} s(n) \cdot s'(\frac{d'}{10q(n)n'}) \end{array}\right)$$

Among the many parameters in the previous Theorem, notice the query complexity of the composed PCP is that of the *inner* PCPP-verifier, while all other parameters (most notably, soundness) are not much worse than that of the *outer* coded PCP verifier.

Proof. We describe the operation of the composed verifier. Then we analyze it's properties, focusing on the soundness (which, as usual, is the main non-trivial part).

Operation Let V be the outer $(t, \ell, r, q, \mathcal{C}, m)$-coded PCP verifier for L. On random coins R, we are interested in computing $D(\Pi|_I)$, where D and I depend on x and R. Instead of directly doing this, we invoke the PCPP-verifier on explicit input D, implicit input $\Pi|_I$ and an auxiliary PCPP (for D).

Basic parameters Randomness, query complexity and running time are immediate. Regarding length, our composed proof is combined of the original proof Π of length ℓ, and for every coin toss R we expect a proof of length $\ell'(m(n))$, giving the length stated in the Theorem.

Completeness If $x \in L$ then there exists Π satisfying V w.p. $\geq c(n)$. So, with this probability at least, $\Pi|_I$ satisfies D and in this case there exists a PCPP proof causing the inner verifier to accept explicit input D with implicit input $\Pi|_I$ with probability $\geq c'(n)$. Thus, completeness of the composed verifier is at least $c(n) \cdot c'(n)$.

Soundness Suppose $x \notin L$. Let δ_0 be the probability that there exists $i \in I$ such that Π_i is $d'/10n'$-far from a codeword of $\mathcal{C}$. There are two cases:

- $\delta_0 \geq s(n)/2$: Recall D rejects any input that is not a concatenation of q codewords of $\mathcal{C}$. In this case, with probability $\geq \delta_0$ the distance of Π_I from an assignment accepted by D is at least $d'/10qn$ so our overall rejection probability is at least $\delta_0 \cdot s'(d'/10qn', m(n))$.

- $\delta_0 < s(n)/2$: Let Π'_i be the $\mathcal{C}$-codeword closest to Π_i (breaking ties arbitrarily). With probability $\geq s(n)$ we have $\Pi'|_I$ does not satisfy D. With probability $\geq 1 - s(n)/2$ we have Π_i is $\frac{d'}{10n'}$-close to Π'_i for all $i \in I$. We conclude that with probability $\geq s(n)/2$ we have Π_I is $d'/10n'$-close to a concatenation of q codewords of $\mathcal{C}$, yet these codewords do not satisfy D. Hence, at least $\frac{d'}{qn'}(\frac{1}{3} - \frac{1}{10})$ fraction of entries of Π_I must be changed to reach an assignment that satisfies D. So our rejection probability in this case is at least

$$\delta_0 \cdot s'(\frac{d'}{qn'}(\frac{1}{3} - \frac{1}{10}), m(n)) \geq \delta_0 \cdot s'(\frac{d'}{10qn'}, m(n))$$

The inequality above uses the monotonicity of s'. This concludes the soundness analysis and with our proof is complete.

□

6.3 Proving Phase III of Dinur's theorem

We are ready to complete the proof of phase III of Dinur's Theorem. Recall that $[n', k', d']_\Sigma$-error correcting codes $\mathcal{C}$ with $d'/n' > 1/3, n' = \mathcal{O}(k')$ that can be decoded in polynomial time (in n') are well-known. Applying such a code with $k' = d^a$ to the constraint graph $\mathcal{G}$ (over alphabet Σ^{d^a}) reduces GAP-CG$\left(n_2, \Sigma^{d^a}, \hat{s}_2\right)$ to

$$\textbf{coded-PCP}\left(\begin{array}{lcl} \mathsf{length} & = & n_2 \cdot 2^{\mathcal{O}(d^a)} \\ \mathsf{randomness} & = & \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = & 2 \\ \mathsf{time} & = & 2^{\mathcal{O}(d^a)} \\ \mathsf{code} & = & \mathcal{C} \\ \mathsf{decision\text{-}size} & = & 2^{\mathcal{O}(d^a)} \\ \mathsf{completeness} & = & 1 \\ \mathsf{soundness} & = & \hat{s}_2(n_2) \end{array}\right)$$

The reason for the running time and decision size being $2^{\mathcal{O}(d^a)}$ is that we have an arbitrary constraint over two inputs from Σ^{d^a}, so the size of a circuit deciding such a constraint is bounded by $2^{\log(|\Sigma|)\cdot d^a} = 2^{\mathcal{O}(d^a)}$ (notice the extra cost of decoding the code $\mathcal{C}$ is negligible with respect to 2^{d^a}). We now apply the Composition Theorem 6.2 to the above PCP language and the Hadamard based PCPP of Theorem 5.12. (The statement of the Theorem has query complexity sixteen. This can be reduced to two queries, over an alphabet of size 3, with a constant reduction in the soundness. For details, see Homework set 1). The main parameters of this PCPP are: Two queries; small alphabet Σ; perfect completeness; constant soundness function ($s'(\delta) \geq \delta/c'_3$ for some constant $c'_3 > 1$); We conclude that for some $c_3 > 1$ we get GAP-CG$\left(n_2, \Sigma^{d^a}, \hat{s}_2\right)$ is reduced to

$$\textbf{PCP}\left(\begin{array}{lcl} \mathsf{length} & = & n_2 + 2^{\log(n_2)} \cdot 2^{2^{\mathcal{O}(d^a)}} = \mathcal{O}(n_2) \\ \mathsf{randomness} & = & \log(n_2) + 2^{\mathcal{O}(d^a)} = \log(n_2) + \mathcal{O}(1) \\ \mathsf{query} & = & 2 \\ \mathsf{time} & = & n_2 + 2^{\mathcal{O}(d^a)} = n_2^{\mathcal{O}(1)} \\ \mathsf{completeness} & = & 1 \\ \mathsf{soundness} & = & \hat{s}_2(n_2)/c_3 \end{array}\right). \tag{6}$$

For instance, consider the soundness. The composition Theorem 6.2 gives

$$\hat{s}_3 \geq \frac{1}{2}\hat{s}_2 \cdot s'(d'/20n') \geq \frac{1}{2}\hat{s}_2 \cdot s'(1/60) \geq \hat{s}_2 \cdot 1/60c_3' \geq \hat{s}_2/c_3$$

where $c_3 = 1/120c_3'$. Since our alphabet is Σ and we have only two queries, Equation (6) is equivalent to GAP-CG$(\mathcal{O}(n_2), \Sigma, \hat{s}_3 \geq \hat{s}_2/c_3)$, and our proof of the third phase is complete.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	December 15th, 2005

Lecture 7: Dinur's Gap Amplification IV: Graph powering

Lecturer: Eli Ben-Sasson *Scribe: Eyal Rozenberg*

Today we shall focus on the second, central, phase of the proof of Proposition 4.12 – the graph powering phase. Our goal is to prove that

$$\begin{array}{c} \text{GAP-CG}\left(n_1, \Sigma, s_1(n_1), (d, \tfrac{1}{2})\text{-expansion}\right) \\ \Big\downarrow \text{polynomial reduction} \\ \text{GAP-CG}\left(d^a n_1, \Sigma^{d^a}, \beta \cdot \min\left\{\sqrt{a} \cdot s_1(n_1), 1/\sqrt{a}\right\}\right) \end{array}$$

We recall that a graph G is a (d, λ)-*expander* if it is d-regular, has self-loops at every vertex, and for every $F \subseteq E$, the probability that a random walk originating in a random edge of F has its $i+1$'th edge in F is no more than $|F|/|E| + \lambda^i$. In our case, this implies that the distribution of the $i+1$'th edge is no further than 2^{-i} from the uniform distribution.

7.1 The Reduction Proper

How shall our reduction be performed? Given some constraint graph

$$\mathcal{G} = \left(G, \Sigma, \mathcal{C} = \left\{C_e : \Sigma^2 \to \{\mathsf{accept}, \mathsf{reject}\} \;\middle|\; e \in E\right\}\right)$$

we construct

$$\mathcal{G}^a = \left(G^a, \Sigma^{d^a}, \mathcal{C}^a\right)$$

The multigraph G^a has the same vertex set as G, but its adjacency matrix is $A_{G^a} = A_G^a$ — that is, for every path of length a from u to v in G, there exists an edge (u, v) in G^a (there may exist several such edges).
The set of colors now has Σ^{d^a} values; each vertex u may now be colored with d^a values in Σ, representing u's 'opinion' regarding the G-colors of the d^a vertices at the ends of length-a paths originating in u — the colors of a surface of a radius-a ball around u. Note the existence of self loops on every vertex ensure the assignment to u actually gives an opinion about every vertex at distance at most a from u.
If $\vec{\sigma} : V(G) \to \Sigma^{d^a}$ is an assignment, we denote by $\vec{\sigma}(u)_{u'}$ the opinion of vertex u regarding the color of vertex u' in G.
Each edge $(u, v) \in E(\mathcal{G}^a)$ is now constrained as follows:

- The 'opinions' of both u and v, taken individually, must be self-consistent, i.e. u may not 'believe' two paths end with a vertex in different colors, if the two paths end in the same vertex in G.

- The 'opinions' of both u and v, taken individually, must satisfy the original constraints $\mathcal{C}$: For every $e = (u', u'') \in E(G)$ for which u has an opinion on both ends, $C_e(\vec{\sigma}(u)_{u'}, \vec{\sigma}(u)_{u''}) = \mathsf{accept}$.
- u and v must agree on the G-color of every vertex reachable by a length-a path (in G) both from u and from v (the vertices in the intersection of the radius-a ball).

7.2 Validity of the Reduction

We begin by observing that n — the number of edges — has indeed increased by a factor of d^a (there are d^a length-a cycles originating in each vertex of G), and that the completeness remains 1: If $\mathcal{G}$ is has a valid Σ-coloring, one may use it to construct a valid Σ^{d^a}-coloring of $\mathcal{G}^a$, by having each vertex' opinion regarding other vertices' colors being their actual coloring by the valid coloring of $\mathcal{G}$. As usual, the difficult part is the soundness analysis, which follows immediately from the following:

Proposition 7.1. *For every $d, |\Sigma| \in \mathbb{N}^+$ there exists $\beta = \beta(d, |\Sigma|) > 0$ such, that for every $a \in \mathbb{N}^+$ and every constraint graph $\mathcal{G}$ over a $(d, 1/2)$-expander graph G,*

$$s(\mathcal{G}^a) \geq \beta \cdot \sqrt{a} \cdot \min\{s(\mathcal{G}), 1/a\}$$

To prove this, we will reconstruct an assignment to $\mathcal{G}$ from an assignment to $\mathcal{G}^a$, in a way which preserves soundness — so that the lower bound on the soundness of $\mathcal{G}$ assignments will apply to their originating $\mathcal{G}^a$ assignments.
Let $\vec{\sigma} : V(G) \to \Sigma^{d^a}$ be an assignment to $\mathcal{G}^a$. Our reconstructed assignment $\sigma : V(G) \to \Sigma$ is the assignment in which $\sigma(v)$ is the majority value of $\vec{\sigma}(u)_v$ taken over all paths of length-$\frac{1}{2}a$ starting at v (and ending at u). Note that u's are counted once for every length-$\frac{1}{2}a$ path from them to v, that is, σ is not a majority vote over vertices but over ends of paths (and some vertices may have more paths ending at them than others).
Now, we know there exists a set of edges with constraints that σ violates of size $\geq s(\mathcal{G}) \cdot |E(G)|$. Let $F \subseteq E(G)$ be a subset of these edges such that $|F| = \min\{s(\mathcal{G}), 1/a\} \cdot |E(G)|$.
To relate this set of edges to the constraints violated by $\vec{\sigma}$, we will analyze the probability of an edge of G^a not satisfying a constraint, by examining a 'middle' edge in a random walk in G, of length a.
We say the length-a path $\mathbf{e} = (u_0, \ldots, u_a)$ 'hits' F at the i'th position if $(u_{i-1}, u_i) \in F$ and $\vec{\sigma}(u_0)_{u_{i-1}} = \sigma(u_{i-1})$ and $\vec{\sigma}(u_a)_{u_i} = \sigma(u_i)$.
We consider, with foresight, only the 'middle' of the path:

$$I = \mathbb{N}^+ \cap \left[\frac{a}{2} - \sqrt{a}, \frac{a}{2} + \sqrt{a}\right]$$

and letting

$$N(\mathbf{e}) = |\{i \in I \mid \mathbf{e} \text{ hits } F \text{ at the } i\text{'th position}\}|$$

we observe that our proof can be completed by demonstrating that

$$\Pr_{\mathbf{e}}[N(\mathbf{e}) > 0] \geq \beta \cdot \sqrt{a} \cdot \frac{|F|}{|E|}$$

due to our observations regarding $\frac{|F|}{|E|}$ above. To prove this characterization of $N(\mathbf{e})$, we will avail ourselves of the following lemmata regarding its first and second moments:

Lemma 7.2 (First moment)**.** *There exists $\beta_1 > 0$ such that*

$$\mathop{\mathbf{Ex}}_{\mathbf{e}}[N(\mathbf{e})] \geq \beta_1 \cdot \sqrt{a} \cdot \frac{|F|}{|E|}$$

Lemma 7.3 (Second moment)**.** *There exists $\beta_2 > 0$ such that*

$$\mathop{\mathbf{Ex}}_{\mathbf{e}}\left[N^2(\mathbf{e})\right] \leq \beta_2 \cdot \sqrt{a} \cdot \frac{|F|}{|E|}$$

The first lemma will be proven using the existence of self-loops in G; For proof of the second lemma we will use G's expansion property.
These two lemmata will be applied using the following observation

Observation 7.4. If X is a non-negative random variable, then

$$\mathbf{Pr}[X > 0] \geq \frac{(\mathbf{Ex}[X])^2}{\mathbf{Ex}[X^2]}$$

Proof.

$$\mathbf{Ex}\left[X^2\right] - \Big(\mathbf{Ex}[X]\Big)^2 = \mathbf{Var}[X] = \mathbf{Ex}\left[\Big(X - \mathbf{Ex}[X]\Big)^2\right] \geq \mathbf{Pr}[X = 0] \cdot \Big(\mathbf{Ex}[X]\Big)^2$$

divide both hands by $\mathbf{Ex}\left[X^2\right]$:

$$1 - \frac{(\mathbf{Ex}[X])^2}{\mathbf{Ex}[X^2]} \geq \mathbf{Pr}[X = 0]$$

Noticing $\mathbf{Pr}[X > 0] = 1 - \mathbf{Pr}[X = 0]$ yields the desired result. □

Proof of Proposition 7.1. We apply the above observation to $X = N(\mathbf{e})$, using the bounds from Lemma 7.3 and Lemma 7.3 for the first and second moments respectively:

$$\mathop{\mathbf{Pr}}_{\mathbf{e}}[N(\mathbf{e}) > 0] \geq \frac{(\mathbf{Ex}[X])^2}{\mathbf{Ex}[X^2]} \geq \frac{\beta_1^2}{\beta_2} \cdot \sqrt{a} \cdot \min\left\{\frac{|F|}{|E|}, \frac{1}{a}\right\}$$

□

It now remains to prove our two moment bound lemmata.

Proof of Lemma 7.2. Let $N_i(\mathbf{e})$ be the indicator variable for (v_{i-1}, v_i) hitting F. By linearity of expectation,

$$\mathop{\mathbf{Ex}}_{\mathbf{e}}[N(\mathbf{e})] = \sum_{i \in I} \mathop{\mathbf{Ex}}_{\mathbf{e}}[N_i(\mathbf{e})]$$

Now let us consider an alternative view of the distribution of our random walk: Suppose we first choose the i'th edge $e = (u, v)$, uniformly, then choose a random head-segment

and tail-segment of the path, uniformly among the paths ending at u and beginning at v, respectively, of lengths $i-1$ and $a-i$ respectively. This indeed yields the same distribution of random walks as when starting out with the first edge, because G is d regular. A crucial observation is that the head and tail of the path are independent of each other.

We note that if $i = a/2+1$, the head segment and the tail segment are random paths of length $a/2$ from u and from v respectively. Each of them ends in a vertex u' (resp. v') colored in the majority-vote color, with probability at least $1/|\Sigma|$. When both these (independent) events occur, we conclude u is colored in $\sigma(u)$ and v in $\sigma(v)$. Thus if $(u,v) \in F$, with probability at least $1\Big/|\Sigma|^2$, the edge e will 'hit' F with our random walk:

$$\mathbf{Ex}\Big[N_{\frac{a}{2}}\Big] = \mathbf{Pr}[e \text{ hits } F] = \mathbf{Pr}[e \in F]\cdot\mathbf{Pr}\big[\vec{\sigma}\big(u'\big)_u = \sigma(u) \wedge \vec{\sigma}\big(v'\big)_v = \sigma(v)\big|e \in F\big] \geq \frac{|F|}{|E|}\cdot\frac{1}{|\Sigma|^2}$$

We would like to make a similar argument regarding N_i for all other edges with $i \in I$, which would multiply this expectation by $\mathcal{O}(\sqrt{a})$ as required. Unfortunately, the head and tail path lengths from the other edges are only *close* to $a/2$, not equal. We must thus show that the probability of having $\vec{\sigma}(u')_u = \sigma(u)$ and $\vec{\sigma}(v')_v = \sigma(v)$ is high enough even under these conditions.

In general graphs, what happens at distance $a/2$ from a vertex may be quite different than what happens at distance, say $a/2-1$ (think: bipartite graphs); fortunately for us, though, G has at least one self-loop at each vertex, so many of the possible length-a paths from a vertex correspond to shorter paths from the vertex with fewer self-loops, and the typical 'effective' distance (i.e. distance without counting self-loops) of a length a walk is $\approx (1-1/d)\cdot a \pm \sqrt{a}$. We can utilize this fact to relate the opinions of G^a neighbors of the $i = a/2$ edge with that of G^a neighbors of a different edge in I:

Fix $i \in I$. Let $X_{u,\ell} = \vec{\sigma}(u')_u$, where u is at the end of a length-ℓ random walk from u; Mark one self loop on each vertex of G (there might be several) and let $X'_{u,\ell}$ be a similar variable except that the random walk does is only over non-marked edges. When a random walk is at some vertex, it uses the marked edge with probability $1/d$, and non-marked edges with probability $p = 1-1/d$; we thus have:

$$\mathbf{Pr}[X_{u,\ell} = s] = \sum_{k=0}^{\ell} \mathbf{Pr}[B_{\ell,p} = k]\cdot\mathbf{Pr}\big[X'_{u,k} = s\big]$$

where $B_{\ell,p}$ is a binomial distribution with ℓ Bernoulli trials of success probability p.

The binomial distribution satisfies the following:

Fact 7.5. For every $0 < p < 1$ and $c > 0$ there exists $0 < \tau < 1$ such that if $|\ell_1 - \ell_2| \leq \sqrt{2\ell_1}$, then for every k satisfying $|k - p\ell_1| \leq c\cdot\sqrt{\ell_1}$,

$$\tau \leq \frac{\mathbf{Pr}[B_{\ell_1,p} = k]}{\mathbf{Pr}[B_{\ell_2,p} = k]} \leq \frac{1}{\tau}$$

(one can verify this is true by examining the binomial distribution function and bounding it from both sides). So when our ℓ_2 is close enough to ℓ_1, the binomial distributions are not very far apart; and, indeed, due to the law of large numbers, for every $\varepsilon > 0$ there exists $c > 0$ such that $\mathbf{Pr}[B_{\ell,p} \notin p\cdot\ell \pm c\sqrt{a}] < \varepsilon$. We will use such a c for $\varepsilon = 1/2|\Sigma|$.

We can now continue the earlier evaluation, setting $\ell_1 = a/2$ and $\ell_2 = i$ to obtain

$$\begin{aligned}
\mathbf{Pr}[X_{u,i} = \sigma(u)] &= \sum_{k \le i} \mathbf{Pr}[B_{i,p} = k] \cdot \mathbf{Pr}\left[X'_{u,k} = \sigma(u)\right] \\
&\ge \sum_{k \in \left[\frac{a}{2} - c\sqrt{a}, i\right]} \mathbf{Pr}[B_{i,p} = k] \cdot \mathbf{Pr}\left[X'_{u,k} = \sigma(u)\right] \\
&\ge \tau \cdot \left(\sum_{k \in \left[\frac{a}{2} - c\sqrt{a}, i\right]} \mathbf{Pr}\left[B_{\frac{a}{2},p} = k\right] \cdot \mathbf{Pr}\left[X'_{u,k} = \sigma(u)\right] \right) \\
&\ge \tau \cdot \left(\sum_{k \le \frac{a}{2}} \mathbf{Pr}\left[B_{\frac{a}{2},p} = k\right] \cdot \mathbf{Pr}\left[X'_{u,k} = \sigma(u)\right] - \frac{1}{2|\Sigma|} \right) \\
&= \tau \cdot \mathbf{Pr}\left[X_{u,\frac{a}{2}} = \sigma(u)\right] - \frac{\tau}{2|\Sigma|} \ge \frac{\tau}{|\Sigma|} - \frac{\tau}{2|\Sigma|} = \frac{\tau}{2|\Sigma|}
\end{aligned}$$

The first inequality follows because we take a partial sum. The second inequality uses Fact 7.5. The third inequality uses our choice of c such that $\mathbf{Pr}\left[B_{a/2,p} \notin p \cdot a/2 \pm c\sqrt{a}\right] < 1/2|\Sigma|$. The very last inequality follows because $\mathbf{Pr}\left[X_{u,a/2} = \sigma(u)\right] \ge 1/|\Sigma|$. Thus,

$$\mathbf{Ex}[N_i] \ge \frac{|F|}{|E|} \cdot \mathbf{Pr}[X_{u,i} = \sigma(u)] \ge \frac{|F|}{|E|} \cdot \frac{\tau}{2|\Sigma|}$$

and recalling $|I| = \Omega(\sqrt{a})$ gives,

$$\mathbf{Ex}[N] = \sum_{i \in I} \mathbf{Ex}[N_i] \ge |I| \cdot \frac{|F|}{|E|} \cdot \frac{\tau}{2|\Sigma|} \ge \beta_1 \cdot \sqrt{a} \cdot \frac{|F|}{|E|}$$

□

Proof of Lemma 7.3. The general argument for this proof is that due to the graph's expansion properties, edges on the random walk which are sufficiently far apart can't be too well-correlated with respect to hitting a relatively small set of 'bad' edges.
For $\mathbf{e} = (u_0, \ldots, u_a)$ Let $Z(\mathbf{e}) = |\{i \in I : (u_{i-1}, u_i) \in F\}|$ be the number of 'bad' edges around the 'middle' of the walk. By definition, $N(\mathbf{e}) \le Z(\mathbf{e})$, So it suffices to show a bound on $(Z(\mathbf{e}))^2$. Let Z_i be the indicator variable for the event $(u_{i-1}, u_i) \in F$; we have $Z = \sum_{i \in I} Z_i$. Now,

$$\mathop{\mathbf{Ex}}_{\mathbf{e}}\left[Z^2\right] = \sum_{i,j \in I} \mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i Z_j] = \sum_i \mathop{\mathbf{Ex}}_{\mathbf{e}}\left[Z_i^2\right] + 2\sum_{i<j} \mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i Z_j] = \sum_i \mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i] + 2\sum_{i<j} \mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i Z_j]$$

The last equality follows because Z_i is $\{0,1\}$-valued. What is $\mathbf{Ex}_e[Z_i Z_j]$? Well,

$$\mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i Z_j] = \mathbf{Pr}[Z_i Z_j = 1] = \mathbf{Pr}[Z_i = 1] \cdot \mathbf{Pr}[Z_j = 1 | Z_i = 1]$$

The first multiplicand is $|F|/|E|$. The second multiplicand is the probability that the $j-i$'th edge of a random walk starting in an edge of F is in F (to see why, construct the walk

again as the ith edge as a random edge, then add a head and a tail); at this point we can use the graph's expansion property:

$$\mathop{\mathbf{Pr}}_{\mathbf{e}}[Z_j = 1 | Z_i = 1] \leq \frac{|F|}{|E|} + 2^{-(j-i)}$$

so

$$\mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i Z_j] = \mathop{\mathbf{Pr}}_{\mathbf{e}}[Z_i = 1] \cdot \mathop{\mathbf{Pr}}_{\mathbf{e}}[Z_j = 1 | Z_i = 1] \leq \frac{|F|}{|E|} \cdot \left(\frac{|F|}{|E|} + 2^{-(j-i)} \right)$$

and substituting this into the equation for $\mathbf{Ex}_{\mathbf{e}}\left[Z^2\right]$ we get:

$$\begin{aligned}
\mathop{\mathbf{Ex}}_{\mathbf{e}}\left[Z^2\right] &= \sum_i \mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i] + 2 \sum_{i<j} \mathop{\mathbf{Ex}}_{\mathbf{e}}[Z_i Z_j] \\
&\leq |I| \cdot \frac{|F|}{|E|} + \frac{|F|}{|E|} \sum_{j>i} \left(\frac{|F|}{|E|} + 2^{-(j-i)} \right) \\
&= |I| \cdot \frac{|F|}{|E|} + 2 \binom{|I|}{2} \cdot \left(\frac{|F|}{|E|} \right)^2 + \frac{|F|}{|E|} \sum_{i \in I} \sum_{j>i} 2^{-(j-i)} \\
&\leq |I| \cdot \frac{|F|}{|E|} + 2 \binom{|I|}{2} \cdot \left(\frac{|F|}{|E|} \right)^2 + \frac{|F|}{|E|} \sum_{i \in I} 2^{-i+1}
\end{aligned}$$

(since $|F|/|E| = \mathcal{O}\left(1/\sqrt{a}\right)$ by our assumptions regarding F)

$$\begin{aligned}
&\leq |I| \cdot \frac{|F|}{|E|} + 2 \binom{|I|}{2} \cdot \frac{|F|}{|E|} \frac{1}{a} + \frac{|F|}{|E|} |I| \sum_{j=1}^{|I|} 2^{-j} \\
&\leq \frac{|F|}{|E|} \left(|I| + \mathcal{O}\left(\frac{|I|^2}{\sqrt{a}} \right) + |I| \right)
\end{aligned}$$

(since $|I| \leq 2\sqrt{a}$)

$$= \frac{|F|}{|E|} \mathcal{O}\left(\sqrt{a}\right)$$

Note we have had to make a 'sacrifice' in this last argument: We had to rely on F being small enough to counteract the squaring of $|I|$. □

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	December 22nd, 2005

Lecture 8: Quasi-linear PCPs I: Reed-Solomon codes

Lecturer: Eli Ben-Sasson *Scribe: Anat Paskin*

Over the course of the next several lectures, we will follow the proof the "short PCPs" part of Theorem 1.7:

$$\mathbf{NTIME}\left(f(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = f(n)\cdot \mathrm{polylog}(f(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}\left(1\right) \\ \mathsf{query} & = \mathcal{O}\left(1\right) \\ \mathsf{time} & = f(n)\cdot \mathrm{polylog}(f(n)) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = \frac{1}{2} \end{array}\right)$$

The proof will follow the one appearing in [11]. Its mainstay is the following result:

Theorem 8.1.

$$\mathbf{NTIME}\left(f(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = f(n)\cdot \mathrm{polylog}(f(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}\left(1\right) \\ \mathsf{query} & = \mathcal{O}\left(1\right) \\ \mathsf{time} & = \mathrm{poly}(f(n)) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = 1/\mathrm{polylog}(n) \end{array}\right)$$

We will leave the exact derivation of the "short PCPs" result from Theorem 8.1 to a later lecture, but for the sake of satisfying our curiosity: the Theorem 8.1 verifier will be transformed into an $\mathcal{O}\left(1\right)$-query verifier, paying with reduced soundness; we'll then apply Dinur's Theorem 4.4 to get the soundness back up.

Overview of proof of Theorem 8.1 Our construction will resemble the proof of the Hadamard based, exponential length PCP (Theorem 2.1), with some significant modifications. As in the proof of Theorem 2.1, the construction used in our proof is based on the ability to test if a 'received' (and possibly corrupted) word is close to a word of a 'nice' code. In the case of Hadamard codes, testing proximity is possible with three queries, as implied by the local testability of the code (Theorem 2.9). In contrast, our next construction is based on the well-known family of Reed-Solomon codes, which are *not* locally testable with constant query complexity, by a long shot. However, all we really need (even for proving Theorem 2.1) is some query-efficient method for testing proximity to the code. For this purpose, the concept of a PCP of proximity (Definition 5.11) is well-suited. The main technical part of the proof of Theorem 8.1 (to be expanded upon in this and next lecture) is a short PCP of proximity for the special family of Reed-Solomon codes, defined below. Later on we will reduce the task of checking whether $x \in L$ to a constant number (three) of Reed-Solomon proximity tests.

8.1 Reed-Solomon Codes

As said above, our short PCPs are based on a well-known family of codes, that is also heavily used in practice.

Definition 8.2 (Reed-Solomon codes)**.** Let F be a finite field. The code $\mathrm{RS}(F,S,k)$ is the $[n,k+1,n-k]_F$-LECC code given by the map $(a_0,\ldots,a_k) \to (\Sigma_{i=0}^{k}(a_i x^i))_{x\in S}$ (for some ordering of S).

Abusing notation, we will view $RS(F,S,k)$ as the image of the code, that is

$$\mathrm{RS}(F,S,k) = \{p : S \to F \mid p \text{ is an evaluation of a degree } k \text{ polynomial over } F\}$$

We will be interested in testing proximity to Reed-Solomon codes, via PCPs of proximity (PCPPs). For this purpose, we formally define the pair-language corresponding to RS-codes that will be tested by our PCPP-verifier.

Definition 8.3 (Pair-RS)**.** Pair-RS is a pair language of words of the form $\{((F,S,k),p)\}$. The explicit input is a triple (F,S,k) where F is a description of a finite fields (e.g, via an irreducible polynomial), $S \subseteq F$, $k \in \mathbb{N}^+$. The implicit input is a function $p : S \to F$. A pair $((F,S,k),p)$ belong to Pair-RS iff (F,S,k) is a description of a valid RS-code, and $p \in \mathrm{RS}(F,S,k)$.

We will construct short PCPPs for RS-codes over certain 'nice' fields, and these codes will be sufficient for proving Theorem 8.1. We denote the characteristic of a field F by $\mathrm{char}(F)$. Let $\mathbb{F}_{2^\ell}$ be a field of characteristic 2. Recall $\mathbb{F}_{2^\ell}$ may be viewed as a linear space of dimension ℓ over $\mathbb{F}_2$. We will restrict our view only to the following sub-language of Pair-RS.

Definition 8.4.

$$\textsc{Pair-Binary-RS} = \left\{ ((F,S,k),p) \in \textsc{Pair-RS} \;\middle|\; \begin{array}{l} \mathrm{char}(F) = 2 \\ S \subseteq F \text{ is a linear subspace} \end{array} \right\}$$

Our main technical result (the analog of Theorem 2.9) states the following.

Theorem 8.5. *Denoting* $n = |S|$ [4]*,*

$$\textsc{Pair-Binary-RS} \in \mathbf{PCPP} \left(\begin{array}{lcl} \mathsf{length} & = & n \cdot \mathrm{polylog}(n) \\ \mathsf{randomness} & = & \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = & \mathcal{O}(1) \\ \mathsf{time} & = & n^{\mathcal{O}(1)} \\ \mathsf{completeness} & = & 1 \\ \mathsf{soundness} & = & \delta/\mathrm{polylog}(n) \end{array} \right)$$

[4] Note that the explicit input size is $o(n)$

8.2 Bivariate low degree testing

To test that $p \in \mathrm{RS}(F, S, k)$, we need to verify that p is a polynomial of degree $\leq k$. A natural test consists of querying p on $k+1$ points, interpolating the degree k polynomial that passes through these points and then querying one more point and testing it with respect to the interpolated polynomial. Rubinfeld and Sudan [43] showed that this is indeed a good tester for the Reed-Solomon codes. However, notice the query complexity is as large as the degree of the polynomial, and in our PCP applications (say, for the language CIRCUIT-SAT) the degree will be even larger than the size of the input (circuit), so we clearly need the query complexity to be smaller than the degree. However, we can't hope to do any better than the Rubinfeld-Sudan test if we only query p and insist on perfect completeness[5], because through every $k+1$ points there passes polynomial of degree $\leq k$. Therefor, unlike in Theorem 2.9, we will need additional information, that will be stored in the proof of proximity π for p (recall that a **PCPP** verifier accepts an oracle to a pair y, π, where y is the implicit input, and π is an auxiliary proof for it). The idea is to represent p as a bivariate polynomial. This strategy is motivated by nice local testability properties of bivariate polynomials that we now discuss. Later on, we will show how to connect the univariate polynomial p to some low-degree bivariate representation of it. First, some notation.

> E.R.: *We don't need to define what bivariate polynomials are, this isn't a first-year algebra course.*

Definition 8.6. An m-variate polynomial over a ring R is a formal sum of the form

$$Q(\vec{x}) = \sum_{\vec{i}} \left(\alpha_{\vec{i}} \prod_{j} (x_j)^{i_j} \right) \quad \vec{i} \in \mathbb{N}^m \,,\, \alpha_{\vec{i}} \in R$$

and

$$\deg(Q) = \max \left\{ \sum_j i_j \;\middle|\; \alpha_{\vec{i}} \neq 0 \right\}$$

$$\deg_{x_{j_1} \ldots x_{j_k}}(Q) = \max \left\{ \sum_{j \in j_1 \ldots j_k} i_j \;\middle|\; \alpha_{\vec{i}} \neq 0 \right\}$$

are its overall degree and its degree in some specific subset of the variables.

Multivariate polynomials (specifically, bivariate ones) are instrumental in the definition following family of error correcting codes, which constitute a generalization of the Reed-Solomon codes.

Definition 8.7 (Bivariate Reed-Muller Codes)**.** Let $X, Y \subseteq F, |X| = n, |Y| = m$, $d_x, d_y \in \mathbb{N}^+$. The bivariate Reed-Muller code $RM(F, X, Y, d_x, d_y)$ is given by a map

$$(a_{i,j})_{0 \leq i \leq d_x 0 \leq j \leq d_y} \mapsto (Q_a(x, y))_{x \in X, y \in Y},$$

where $Q_a(x, y) = \Sigma_{i,j} a_{i,j} x^i y^j$

[5] In the case of for linear codes such as Reed-Solomon, non-perfect completeness implies perfect completeness, so query complexity can't be reduced in this more general case either.

It is fairly well-known (and not too hard to prove) that

Observation 8.8. $RM(F, X, Y, d_x, d_y)$ is a $[n \cdot m, (d_x+1)(d_y+1), (n-d_x)(m-d_y)]$-LECC.

For example, for $n = m, d_x = d_y = n/4 - 1$, we get a $\left[n^2, n^2/16, \approx \frac{9}{16}n^2\right]$-LECC.
Consider the task of testing proximity to the RM-code. We have oracle access to a bivariate function $f : X \times Y \to F$ and wish to minimize our queries into f, accept if $f \in RM(F, X, Y, d_x, d_y)$ and reject if f is far from this code.
Bivariate polynomials have a natural query-efficient test associated with them.

Definition 8.9 (RM Tester)**.** For F, X, Y, d_x, d_y as in Definition 8.7, bivariate function $f : X \times Y \to F$ and $x_0 \in X, y_0 \in Y$, define the following sub-tests.

- x_0**-column sub-test:** Accept iff $f|_{x=x_0} \in \mathrm{RS}(F, Y, d_y)$, where $f|_{x=x_0}$ is the restriction of f to inputs $\{(x_0, y) : y \in Y\}$.
- y_0**-row sub-test:** Accept iff $f_{y=y_0} \in \mathrm{RS}(F, X, d_x)$, where $f|_{y=y_0}$ is the restriction of f to inputs $\{(x, y_0) : x \in X\}$.

The non-adaptiveRM-tester is defined as follows. On input (F, X, Y, d_x, d_y), oracle access to $f : X \times Y \to F$ and randomness $R \in \{0,1\}^*$, the tester performs with probability half a uniformly random x_0-column sub-test and with probability half a uniformly random y_0-row sub-test. Let $T^f_{RM}[F, X, Y, d_x, d_y; R]$ denote the output of this tester.

It is not hard to see that f is an RM-codeword iff the RM-tester accepts f with probability one. But things get even better. It turns out that if f is far from the RM-code, then a random row/column will also be far from the relevant RS-code. This seemingly self-evident claim is actually non-trivial at all (for a proof see [39]) and will be crucial in our construction.
To formally state our main claim about the testability of bivariate RM-codes, we generalize the notion of local testability of codes, and claim that RM codes are locally testable according to this stronger notion.

Definition 8.10 (robust-LTC)**.** A $(t(n), r(n), q(n), s(\delta, n))$-restricted, non-adaptivetester, is a PPT T with oracle access to a purported codeword $w \in F$, such that on input 1^n (where $n = |w|$), T tosses $r(n)$ coins, and outputs a set of indices $I \subseteq [n]^q$ and a truth table $S \subseteq F^q$. (I is the set of indices of queries and S is the set of views that are to be accepted by the verifier). Let $I(n, R), S(n, R)$ denote the output of T on input 1^n and randomness R, and let $w|_{I(n,R)}$ denote the restriction of w to the index set I. For a soundness function $s : [0,1] \times \mathbb{N}^+ \to [0,1]$, a family of $[n, k, d]$-ECC-codes is called an $(t(n), r(n), q(n), s(\delta, n))$-rLTC, if there exists a $(t(n), r(n), q(n), s(\delta, n))$-restricted verifier with the following properties:

- Completeness: If $w \in C$, then $\mathbf{Pr}_R\left[w|_{I(n,R)} \in S(n, R)\right] = 1$.
- Robust Soundness: $\forall\, n, w \in F^n \left[\mathbf{Ex}_R\left[\Delta^*\left(w|_{I(n,R)}, S(n, R)\right)\right] \geq s(\Delta^*(w, \mathrm{Img}(C)), n)\right]$.

To get an intuitive feeling for the previous definition, think of the RM-tester from Definition 8.9. In this case I is either the x_0-column or the y_0-row, and S is either $\mathrm{RS}(F, Y, d_y)$ (in the column case) or $\mathrm{RS}(F, X, d_x)$ (in the row case).

One difference between the above definition and the standard one (Definition 2.8) is that the tester here is non-adaptive. This difference however is not very significant, since all LTC's we've seen so far are linear codes, and without loss of generality, every tester for a linear code is non-adaptive. The significant difference is in the soundness notion used. Note that the soundness of Definition 2.8 is similar to that of the robust soundness above, but for the distance measure we use. In particular, the standard (non-robust) soundness uses (on the left hand side) the *discrete* distance measure Δ^u defined by

$$\Delta^u(w,S) = \begin{cases} 1 & w \in S \\ 0 & \text{otherwise} \end{cases}$$

The main technical result we use is a fundamental Theorem by Polischuk and Speilman [39] on local testability of bivariate RM codes. (Its proof relies on algebraic techniques beyond the scope of this course, hence omitted.)

Theorem 8.11 (The bivariate RM-code is a robust-LTC [39])**.** *Let $X, Y \subseteq F$ with $|X| = n$, $|Y| = m$, and let $d_x \leq n/4, d_y \leq m/8$. The Reed-Muller code $RM(F, X, Y, d_x, d_y)$ is an $(\text{poly}(n+m), \lceil \log(\max\{n,m\}) \rceil, \max\{n,m\}, \alpha \cdot \delta)$-rLTC for some global constant α (independent of X, Y, d_x, d_y). Furthermore, these parameters are achieved specifically by the RM-tester of Definition 8.9.*

The robust soundness of the RM-tester guarantees that when f is far from the RM-code, not only does a random test reject, rather the distance of a random row/column is far from being a RS-codeword (i.e. a low-degree univariate polynomial). As we will later see, this more 'robust' soundness property will allow us reduce query complexity. This is because we can now apply recursion. Indeed, suppose we could somehow view the univariate function $p : S \to F$ of supposed degree k as a bivariate polynomial $f : X \times Y \to F$, for $|X|, |Y| \approx \sqrt{|S|}$ of supposed degree $\sqrt{k}$ in both variables (we shall do something like this later on). Then, to test that p is close to a low-degree polynomial, we only need to test that f restricted to a random row/column is close to a small degree univariate polynomial over a smaller domain. For this purpose we can apply recursion and use a PCPP with even smaller query complexity. Our next step is to show how to associate a bivariate polynomial with our univariate one.

8.3 From univariate to bivariate polynomials

For simplicity, consider testing proximity to $\text{RS}(F, S, k = |S|/2)$ and denote $|S| = n$. Theorem 8.11 suggests that it could be nice to represent our univariate polynomial over S by a bivariate polynomial Q over a domain $X \times Y$, where $|X|, |Y|$ are small (ideally of size $\approx \sqrt{n}$), and the degree of Q in both variables is$\approx \sqrt{n}$. In this case we query a random row/column of $X \times Y$, reducing the query complexity to $\approx \sqrt{k}$. To achieve (approximately) this, we start from the following observation.

Claim 12. *For every pair of univariate polynomials $P(z), q(z) \in F[z]$, there exists a unique bivariate polynomial $Q(x,y) \in F[x,y]$, such that*

- $\deg_x(Q) < \deg(q)$, $\deg_y(Q) = \lfloor \deg(P)/\deg(q) \rfloor$.

- $P(z) = Q(z, q(z))$.

Proof Sketch. Q is obtained by dividing $P(z)$ by $y - q(z)$ via a polynomial division procedure for multivariate polynomials (see [16] for a more formal discussion). Such a division represents $P(z)$ as

$$P(z) = \tilde{Q}(z, y)(y - q(z)) + Q(z, y), \quad \deg_x(Q) < \deg(q), \quad \deg_y(Q) = \left\lfloor \deg(P)/\deg(q) \right\rfloor.$$

The claim follows by letting $y = q(z)$, canceling the first term.

The previous claim shows us a way of viewing a univariate polynomial as a bivariate one, evaluated on a carefully chosen subset of points. Indeed, letting

$$Z = Z_{q,S} = \{(z, q(z)) \mid z \in S\} \tag{7}$$

the previous claim says that the evaluation of $P(z)$ on $z \in S$, is the same as the evaluation of $Q(x, y)$ on $(x, y) \in Z$.

Every polynomial q induces a unique subset $Z_{q,S} \subset S \times F$. We will use a carefully selected 'low degree' polynomial $q(z)$ and a carefully selected evaluation set S, such that the induced table $Z_{q,S}$ is very 'nicely structured'. From now on fix $F = \mathbb{F}_{2^\ell}$. Recall $\mathbb{F}_{2^\ell}$ is a ℓ-dimensional linear space over $\mathbb{F}_2$. The set S will be a linear subspace of F. The polynomial $q(z)$ will be of degree $\approx \sqrt{n}$, and will correspond naturally to a linear transformation over $\mathbb{F}_2$. More to the point, the mapping $q : F \to F$ will be a $\mathbb{F}_2$-linear map and its kernel (i.e. its set of roots) will be a linear space of size $\sqrt{|S|}$. As will be argued next lecture, in this case, the set of points $Z_{q,S}$ has a very nice structure, as implied (straightforwardly) from the linear properties of q and S. Indeed, $Z_{q,S}$ is the disjoint union of $\sqrt{|S|}$ 'rows' of the form $A \times \{q(\beta)\}$ where A is an affine space of size $\sqrt{|S|}$ and β is an element of S. Moreover, $q(S) = \{q(\beta) : \beta \in S\}$ is a linear space of size $\sqrt{|S|}$.

This structure of $Z_{q,S}$ suggests a 'proof-oracle' for the RS-codeword $p : S \to F$ corresponding to the low degree polynomial $P(z)$. We ask the prover to write down an evaluation of the polynomial $Q(x, y)$ corresponding to P (via Claim 12) on the set of points $_{Ker()}q \times q(S)$. Notice $|_{Ker()}q \times q(S)| = |S|$, i.e. the 'proof' is short (relative to the length of the codeword). We can test that the 'proof' is close to an evaluation of a bivariate polynomial, using Theorem 8.11. Notice this reduces our query complexity to $\sqrt{|S|}$ which is significantly smaller than $\deg(P)$ which we chose to be $|S|/2$.

But the query complexity can be reduced even further. Every row/column in the set $_{Ker()}q \times q(S)$ is itself a linear subspace of F (of size $\sqrt{|S|}$) so we have reduced our problem to a similar problem of smaller size. Now, the *robustness* of the bivariate Reed-Muller code (given by Theorem 8.11) allows us to apply recursion and further reduce the query complexity. Finally, we have to make verify the 'proof' is consistent with p. We will argue that this reduces once again to testing proximity to RS-codes of size close to $\sqrt{|S|}$ evaluated over linear subspaces of F. Details will be given in the next lecture.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	December 29th, 2005

Lecture 9: Quasi-linear PCPs II: PCPPs for Reed-Solomon codes

Lecturer: Eli Ben-Sasson *Scribe: Dan Buam and Eli Ben-Sasson*

In this lecture we provide a formal proof of Theorem 8.5, showing a PCPP verifier for Reed-Solomon Codes. Next lecture we shall use this verifier to prove the short PCP Theorem 8.1. We will formally prove our result only for the following sub-class of PAIR-BINARY-RS, in which the degree is an exact fraction of the size of the evaluation set. Formally, let

$$\text{PAIR-BINARY-RS}_{\frac{1}{8}} = \left\{ ((F,S,k),p) \in \text{PAIR-BINARY-RS} : k = \frac{|S|}{8} - 1 \right\}$$

As seen in H.W 3 question 1, the existence of PCPP-verifiers for this specific degree implies PCPPs for any degree. Thus, to obtain Theorem 8.5 it suffices to prove the following Theorem, which will be the focus of this lecture.

Theorem 9.1. *Let $n = |S|$, then*

$$\text{PAIR-BINARY-RS}_{\frac{1}{8}} \in \mathbf{PCPP} \left(\begin{array}{ll} \mathsf{length} & = n \cdot \text{poly}(\log(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = \mathcal{O}(1) \\ \mathsf{time} & = n^{\mathcal{O}(1)} \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = \frac{\delta}{\text{poly}(\log(n))} \end{array} \right)$$

Recall our strategy as outlined in the previous lecture. We associate with the supposed univariate polynomial $p : S \to F$ a bivariate polynomial of degree $\approx \sqrt{k}$, via Claim 12. This allows us to view p as an evaluation of a bivariate polynomial on the set of points $Z_{q,S} = \{(z, q(z)) : z \in S\}$. Our proof oracle will be an evaluation of this bivariate polynomial on a special set of points. The set of evaluation points crucially depends on the structure of $Z_{(q,S)}$, which in turn depends on the polynomial $q(z)$ which we use to 'divide' p by (see Claim 12). We now describe the carefully chosen polynomial q.

9.1 Linearized Polynomials

We introduce the notion of a *linearized* polynomial over an extension field. For more information on these polynomials (that have been well investigated in the theory of finite fields) see [33, Chapter 3, Section 4].

Definition 9.2 (linearized polynomial)**.** Let F be an extension of a field K. A polynomial $p \in F[x]$ is called *linearized* over K if $\forall \alpha, \beta \in K$ and $\forall x, y \in F$: $p(\alpha x + \beta y) = \alpha p(x) + \beta p(y)$.

In the case that will interest us, of $F = \mathbb{F}_{2^l}, K = \mathbb{F}_2$, we conclude p is linearized iff $p(x+y) = p(x)+p(y)$ for all $x, y \in F$. Furthermore, it turns out that p is linearized iff $p(x) = \sum_i a_i x^{2^i}$ (we omit the proof of this claim). We will be using linearized polynomials of a special kind.

Definition 9.3 (sub-space polynomials)**.** Let S be a $\mathbb{F}_2$-linear subspace of $\mathbb{F}_{2^\ell}$. The *sub-space* polynomial corresponding to S is,

$$q_S(x) \triangleq \prod_{\alpha \in S} (x - \alpha)$$

Later in our proof, we will need the following nice properties of sub-space polynomials.

Claim 4. *let S be a $\mathbb{F}_2$-linear sub-space of $\mathbb{F}_{2^l}$. Let S be the direct sum of linear sub-space S_0, S_1 (denoted $S = S_0 \oplus S_1$). Let $q(x) = q_{S_0}(x)$ be the sub-space polynomial corresponding to S_0. Then,*

- *$q(x)$ is linearized.*
- *$_{Ker}()q = S_0$, where $_{Ker}()q = \{\alpha \in \mathbb{F} : q(\alpha) = 0\}$.*
- *$q(S) = q(S_1)$, where $q(S) = \{q(\alpha) : \alpha \in S\}$ (and $q(S_1)$ is analogously defined).*
- *$q : S \to \mathbb{F}$ is a $|S_0|$ to 1 mapping from S to F.*
- *For each $\beta \in S_1$, the affine subspace $S_0 + \beta = \{\alpha + \beta : \alpha \in S_0\}$ is mapped to $q(\beta)$. Formally, $q(S_0 + \beta) = \{q(\beta)\}$.*

Proof. We will only prove the first bullet, as the rest follow from the fact that q is a linear transformation. The proof is by induction on $\dim S_0$.

- $\dim S_0 = 0$**:** In this case, $S_0 = \{0\}$. Therefore $q(x) = x$ is clearly a linearized polynomial (in fact, it is even a linear polynomial).
- $\dim S_0 > 0$: Write $S_0 = \text{span}(S', \alpha)$ for some S' of dimension $\dim S_0 - 1$. Let $q^*(x) = q_{S'}(x)$ be linearized (by the inductive hypothesis). The crucial observation is

  $$q_{S_0}(x) = q^*(x) \cdot q^*(x - \alpha).$$

  To see this, notice $\beta \in S_0$ iff either $\beta \in S'$ or $(\beta - \alpha) \in S'$. Using this equality we obtain:

  $$\begin{aligned} q(x+y) &= q^*(x+y) \cdot q^*(x+y-\alpha) \\ &= q^*(x) \cdot (q^*(x+\alpha) + q^*(y)) + q^*(y) \cdot (q^*(y+\alpha) + q^*(x)) \\ &= q(x) + q(y) + 2q^*(x) \cdot q^*(y) \\ &= q(x) + q(y) \end{aligned}$$

  The first equality uses the assumption that q^* is linearized and the last equality follows from $\text{char}(F) = 2$.

□

9.2 The PCPP oracle for the Reed-Solomon code

We are ready to describe the proof oracle for a purported codeword. Recall our implicit input is a function $p : S \to F$, and we wish to test if this function is close to being a polynomial of degree $< |S|/8$. For simplicity of analysis assume $\dim S$ is even (the proof also works when $\dim S$ is odd, but this is slightly messier). Let $b_1 \ldots, b_k$ be a basis for S, where k is even. Let $S_0 = \mathrm{span}\big(b_1, \ldots, b_{k/2}\big), S_1 = \mathrm{span}\big(b_{k/2+1}, \ldots, b_k\big)$ and notice $\dim S_0, \dim S_1 = \dim S/2$. Set $q(z) = q_{S_0}(z)$. From now on we view p as a bi-variate function $p : Z_{q,S} \to F$ and look at the structure of $Z_{S,q}$. Claim 4 implies that projecting the set $Z_{q,S}$ onto its second coordinate (i.e. taking the set $\{y : \exists z \in S,\ q(z) = y\}$) gives the set $q(S_1)$ which is a linear space of dimension $\dim S_1 = \dim S/2$. Furthermore, $Z_{q,S}$ is the disjoint union of $\sqrt{|S|}$ 'rows', where each row is of the form $(S_0 + \beta) \times \{q(\beta)\}$ and $\beta \in S_1$.

Definition 9.5 (RS-proof oracle)**.** The proof oracle for testing proximity to $\mathrm{RS}(F, S, |S|/8-1)$ is defined recursively as follows.

- $|S| \le 64$: The proof oracle is the empty string.
- $|S| > 64$: Let $S'_0 = \mathrm{span}\Big(b_1, \ldots, b_{\frac{k}{2}+3}\Big)$. The proof oracle is the concatenation of:
  1. One bi-variate function $f : T \to F$, where
  $$T = \bigcup_{\beta \in S_1} \mathrm{span}\big(S'_0, \beta\big) \times \{q(\beta)\}.$$
  2. For each $x_0 \in S'_0$, a proof of f's proximity to the code $\mathrm{RS}(F, q(S_1), |q(S_1)|/8-1)$.
  3. For each $y_0 \in q(S_1)$ with $y_0 = q(\beta)$, a proof of f's proximity to the code $\mathrm{RS}(F, \mathrm{span}(S'_0, \beta), |S'_0|/8 - 1)$.

To gain intuition, we point out that f is intended to be the evaluation of the bivariate polynomial $Q(x, y)$ from Claim 12 on the set of points T. Let us examine the structure of T. Notice Claim 4 implies $Z_{q,S} \subset T$. Moreover, a constant fraction of the points on each row of T is in $Z_{q,S}$. Namely, the row $\mathrm{span}(S'_0, \beta) \times \{q(\beta)\}$ includes the set of points $(S_0 + \beta) \times \{q(\beta)\} \subset Z_{q,S}$. Thus, we can define the function $f \circ p : T \to F$ to be equal to p wherever p is defined and equal to f on all other points. (Here we're viewing p as a bivariate function whose domain $Z_{q,S}$ is a subset of the domain of f). Finally, the set T includes a subset of the form $S'_0 \times q(S_1)$ that is suitable for applying the robust tester of Theorem 8.11, provided the degree of Q is small enough with respect to $|S'_0|, |q(S_1)|$. Let us examine the degree of Q and verify it is indeed small enough.

9.2.1 The Reed-Solomon PCPP Verifier

Since $\deg(p)$ is supposed to be $|S|/8 - 1$ and $\deg(q) = \sqrt{|S|}$, Claim 12 implies $\deg_y(Q)$ is supposed to be $|q(S_1)|/8 - 1$. Similarly, Claim 12 implies $\deg_x(Q)$ is supposed to be less than $\deg(q) = \sqrt{|S|} = |S'_0|/8$. We conclude the degree in both x and y is sufficiently small with respect to $|S'_0|, |q(S_1)|$ to apply Theorem 8.11. Furthermore, the RM-tester of Definition 8.9 needs to test proximity to RS-codes over linear spaces, where the degree is

one eighth the size of the linear space, i.e. we have reduced our original problem (testing proximity of p) to a similar problem of size square root the original one. With this intuition in mind we define the RS-verifier.

Definition 9.6 (RS verifier)**.** The verifier for $\mathrm{RS}(F, S, |S|/8 - 1)$ is defined recursively as follows.

- $|S| \leq 64$: Verifier queries all entries of p and accepts iff $p \in \mathrm{RS}(F, S, |S|/8 - 1)$.
- $|S| > 64$: Verifier performs one of the following sub-tests with probability one half each.
  - **Column test:** Select $x_0 \in S_0'$ uniformly, at random. Invoke the verifier for $\mathrm{RS}(F, q(S_1), |q(S_1)|/8 - 1)$ on implicit input $f|_{x=x_0}$ and the x_0-column proof oracle.
  - **Row test:** Select $\beta \in S_1$ uniformly at random and let $y_0 = q(\beta)$. Invoke verifier for $\mathrm{RS}(F, \mathrm{span}(S_0', \beta), |S_0'|/8 - 1)$ on implicit input $(f \circ p)|_{y=y_0}$ and the y_0-row proof oracle.

9.2.2 Analysis of the verifier

We conclude this lecture with a brief analysis of the main parameters of the proof oracle and verifier. As usual, soundness is the hardest part. We only sketch this part and refer the reader to [11] for a complete proof.

Proof length Examining Definition 9.5 we notice the proof oracle for a length n code is comprised of the function f whose size is $\mathcal{O}(n)$ and $\mathcal{O}(\sqrt{n})$ sub-proofs for codewords of size $\sqrt{\mathcal{O}(n)}$ each. Thus,

$$\ell(n) = \mathcal{O}(n) + \mathcal{O}\left(\sqrt{n}\right) \cdot \ell(\mathcal{O}\left(\sqrt{n}\right)) \leq n \cdot \mathrm{poly}\Big(\log(n)\Big).$$

Query complexity is constant, by construction.

Running time can be verified to be polynomial, by inspection.

Perfect Completeness is implied by Claim 12. Indeed, if $p \in \mathrm{RS}(F, S, |S|/8 - 1)$ then f can be chosen to be an evaluation of a polynomial of x-degree $|S_0'|/8 - 1$ and y-degree $|q(S_1)|/8 - 1$. The restriction of $f \circ p$ to rows and columns gives RS-codewords of the above-mentioned degrees. By induction, for each row/column there exists a proof that causes the relevant row/column sub-test to accept with probability one.

Soundness We argue informally. If p is far from being a low degree polynomial, there are two cases.

- If f is far from being an evaluation of a low-degree bi-variate polynomial, then Theorem 8.11 implies the expected distance of a random row/column of f from the relevant RS-code (of smaller size) is large. By induction, the expected rejection probability of the sub-verifiers is large.
- If f is close to an evaluation of a low degree bi-variate polynomial, this polynomial corresponds to a univariate, low-degree, polynomial $\tilde{p}$, via Claim 12. Since p is far from every low degree polynomial, it is also far from $\tilde{p}$. This means that when we look at random rows of $f \circ p$, the entries coming from p (i.e. the value on points from $Z_{q,S}$) will be inconsistent with the other entries on the row. In other words, a random row will also be far from a low-degree, univariate polynomial of size $\approx \sqrt{|S|}$. Applying induction once again implies the expected rejection probability of a random row-test will be large.

This completes our sketch of the short PCPP for Pair-Binary-RS and the proof of Theorem 8.5. Next lecture we will use this Theorem to conclude the proof of Theorem 8.1.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	January 5th, 2005

Lecture 10: Quasi-linear PCPs III

Lecturer: Eli Ben-Sasson *Scribe: Eyal Rozenberg*

Last week, we completed the proof of Theorem 8.5, stating that proximity to Reed-Solomon codes of block-length n over fields of characteristic two, can be verified using proofs of length $\tilde{\mathcal{O}}(n)$ with constant query complexity and. The proofs have perfect completeness and regarding soundness, a received word that is δ-far from the code is rejected with probability at least $\delta/\text{polylog}(n)$. Now we will use this result to prove Theorem 8.1, stating every language in $\mathbf{NTIME}(f(n))$ can be verified using proofs of length $\tilde{\mathcal{O}}(f(n))$, with soundness $1/\text{polylog}(f(n))$. From here, Dinur's result can be used to boost the soundness to half, yielding the "Short PCP" part of Theorem 1.7. To prove Theorem 8.1 we reduce $L \in \mathbf{NTIME}(f(n))$ to a pair of Reed-Solomon proximity testing problems. Moreover, if x is an instance of L of length n, the pair of Reed-Solomon codes to which we need to measure proximity are of block-length $\tilde{\mathcal{O}}(f(n))$.

10.1 A Reed-Solomon-friendly class-complete problem

Idea: We will reduce arbitrary languages to an algebraic constraint satisfaction problem. A satisfying assignment for the problem will be a low-degree univariate polynomial. Recall we have done something similar in the proof of Theorem 2.1 using Hadamard codes: In that case, the problem had been AFFINE-QCSP (Definition 3.8), satisfied by a suitable pair of Hadamard codewords. In our case we will replace it with the following problem, called *Univariate Algebraic Constraint Satisfaction Problem* (UACSP for short) that is suitable for Reed-Solomon codes.

Definition 10.1 (UACSP)**.** Let $k, d \in \mathbb{N}$. An instance of the language $\text{UACSP}_{k,d}$ is a tuple $\phi = (F, (L_1, \ldots, L_k), H, C)$, satisfying

- F is a finite field.
- Each $L_i : F \to F$ is a linear transform of the form $L_i(x) = a_i \cdot x + b_i$ with constants $a_i, b_i \in F$.
- H is a subset of F.
- $C \in F[x, y_1, \ldots, y_k]$ satisfies $\deg_x(C) \leq |H|$ and $\deg_{y_1 \ldots y_k}(C) \leq d$ (recall the degree definitions in 8.6).

The size of the instance, denoted $|\phi|$, is the length of a reasonable encoding of this tuple (i.e. non-unary). An *assignment* to ϕ is a polynomial $A \in F^{|H|}[x]$. We also define

$$B_A(x) = C(x, A(L_1(x)), \ldots, A(L_k(x)))$$

The language $\mathrm{UACSP}_{k,d}$ is defined as

$$\mathrm{UACSP}_{k,d} = \left\{ \phi \;\middle|\; \exists\, A \in F^{|H|}[x] \;\left[B_A(\cdot) \text{ vanishes on } H \right] \right\},$$

Remark. For intuition regarding Definition 10.1, consider an instance of 3-CNF, with n clauses and n variables, where each clause and each variable is indexed by an element of a field F. Furthermore, the CNF is "algebraically structured", namely the j'th clause depends on the three variables whose indices are $L_1(j), L_2(j), L_3(j)$. Think of $A(i)$ as giving the assignment to the i'th variable, and let $H = \{ j \in F \mid C_j \text{ is a clause } \}$. The polynomial $C(x, y_1, y_2, y_3)$, of degree $|H|$ in x, and constant degree in y_1, y_2, y_3 is defined to be satisfied on $j \in H$ iff the assignment given by y_1, y_2, y_3 satisfies the j'th clause. Notice that for such a polynomial C and $x_0 \in H$ we have $B_A(x_0) = 0$ iff $A(L_1(x_0)), A(L_2(x_0)), A(L_3(x_0))$ satisfies the x_0'th constraint. Thus, B_A vanishes on H iff all constraints are satisfied by the assignment A.

The next Theorem says that any language in $\mathbf{NTIME}\left(f(n)\right)$ can be reduced to an instance of $\mathrm{UACSP}_{k,d}$ of size $\tilde{\mathcal{O}}\left(f(n)\right)$. Before proving a weaker, yet simpler version of this Theorem we will show how it implies Theorem 8.1.

Theorem 10.2. *There exist $k, d \in \mathbb{N}$ such that for any reasonable function $f : \mathbb{N}^+ \to \mathbb{N}^+$ and $L \in \mathbf{NTIME}\left(f(n)\right)$, we have*

$$L \xrightarrow[\text{reduction}]{\substack{\mathrm{poly}(f(n))\text{-}\\ \text{time}}} \mathrm{UACSP}_{k,d}$$

with the reduction output satisfying the following:

- *The set H on which the B polynomial vanishes is of size at most $f(n) \cdot \mathrm{polylog}(f(n))$.*
- *The field F over which the constrains are defined has $\mathrm{char}(F) = 2$, and its size is between $100kd|H|$ and $200kd|H|$.*

Remark. The reduction in Theorem 10.2 produces output of length quasi-linear in $f(n)$.

10.2 Short PCPs with poly-logarithmic soundness

We are very close to obtaining our short PCPs; here is the next-to-last step:

Proof of Theorem 8.1. Let $L \in \mathbf{NTIME}\left(f(n)\right)$ and $\varphi \in \Sigma^*$ be an instance of L. We first apply the reduction to UACSP described in Theorem 10.2, to obtain a string ϕ. What must an honest prover do to convince a verifier that $\phi \in \mathrm{UACSP}_{k,d}$?
First, it must provide a satisfying assignment A; the relevant proof-form for such an assignment would be a full evaluation of the polynomial function $p_A : F \to F$ and an additional 'low-degreeness' proof π_A of the fact that $\deg(p_A) \leq |H|$ (a proof of length $\tilde{\mathcal{O}}\left(f(n)\right)$ — the one we use for Pair-Binary-RS).

Since the verifier is not reading all of A, it cannot calculate B, so the prover must also include an evaluation of $p_B : F \to F$ and a proof π_B of its low-degree; to obtain a bound on this degree, we recall:

$$B_A(x) = C(\underbrace{x}_{\deg_x(C)\leq|H|}, \underbrace{\underbrace{A\ \underbrace{(L_1(x))}_{L_i \text{ are linear}}, \ldots, A(L_k(x))}_{\deg(A)\leq|H-1|}}_{\deg_{y_1,\ldots,y_k}(C)\leq d})$$

so $\deg(B_A) \leq |H| + d(|H|-1) = \tilde{\mathcal{O}}(f(n))$, and thus π_{B_A} is also of length $\tilde{\mathcal{O}}(f(n))$.
Finally, the prover must lead the verifier to conclude that $B_A\big|_H \equiv 0$. At first sight this seems very hard because to get the value of a degree D polynomial on $|H|$ points either requires reading its values on H, or performing interpolation, requiring D queries.
However, some basic linear algebra gives a query-efficient way of performing our task. Recall that, for any polynomial $P \in F[x]$ and $x_0 \in F$, $P(x_0) = 0$ iff $(x - x_0)$ divides P. Thus, B_A vanishes over H if and only if it is divisible by the product of $(x - x_0)$ for all $x_0 \in H$. Thus the verifier may be convinced that $B_A\big|_H \equiv 0$ if it is supplied with the quotient of a division of B by the polynomial $g_H(x) = \prod_{h\in H}(x - h)$:

$$q_{B_A} = \frac{B_A}{\prod_{h\in H}(x-h)}$$

The prover can provide the verifier with such a polynomial in a similar fashion as it does A: an evaluation of q_{B_A} over F, accompanied by a proof π_q of the low degree of this evaluation. In fact, having gotten q_{B_A}, the verifier does not even need to have B_A itself, since $B_A = q_{B_A} \cdot g_H$, and the verifier can calculate g_H, so q is sufficient in the proof to determine the value of B; this saves the verifier some consistency checking.
In conclusion, a PCP proof for ϕ belong to $\text{UACSP}_{k,d}$ as a tuple $(p_A, \pi_A), (p_q, \pi_q)$, where $p_A, p_q : F \to F$ and π_A, π_q are PCPP proofs for p_A, p_q being Reed-Solomon codewords — polynomials of degree $|H|, d(|H|-1)$, respectively. A PCP verifier V for deciding whether $\phi \in \text{UACSP}_{k,d}$ will operate as follows:

- Run the Reed-Solomon verifier for each of the function-proof pairs (p_A, π_A) and (p_q, π_q) w.r.t. the appropriate degree bound.
- Verify the consistency of p_q with p_A: Randomly choose $x_0 \in F$ and ensure that

$$C(x_0, p_A(L_1(x_0)), \ldots, p_A(L_k(x_0))) = B_A(x_0) = g_H(x_1) \cdot p_q(x_0)$$

- Accept if none of the checks has failed.

Let us analyze the verifier's parameter:

- length: This requirement holds, since $|F| = \tilde{\mathcal{O}}(f(n))$ and the Reed-Solomon proofs are of lengths $\tilde{\mathcal{O}}(|F|) = \tilde{\mathcal{O}}(f(n))$.
- randomness: By Theorem 8.5, a single run of the Reed-Solomon verification requires $\log(|F| \cdot \log(|F|)) + \mathcal{O}(1)$ coin tosses; we re-use the same randomness for our 2 runs.

- time: The running time is dominated by the running time of the Reed-Solomon verifications: $\mathrm{poly}(f(n))$.
- completeness: The perfect completeness is obvious given our description of what an honest verifier would provide as a proof for an input ϕ with a satisfying assignment.
- soundness: Assume $\phi \notin L$. Necessarily, $\varphi \notin \mathrm{UACSP}_{k,d}$ — that is, for every univariate polynomial $A \in F^{|H|-1}[x]$ we have $B_A\big|_H \not\equiv 0$. Now, if p_A or p_q are $1/100(kd+1)$-far from satisfying their degree bound, then one of the Reed-Solomon verifications fail, with probability at least $1/(100(kd+1)\cdot \mathrm{polylog}(\mathcal{O}(f(n)))$. If p_A, p_q are close to having a low degree, let $\tilde{A}, \tilde{q}$ be the low-degree polynomials to whose evaluation p_A, p_q are respectively close. Since $B_{\tilde{A}}\big|_H \not\equiv 0$, we have $\tilde{q}\cdot g_H \neq B_{\tilde{A}}$, because otherwise $\tilde{A}$ would satisfy the unsatisfiable constraint polynomial, contradiction. So with probability at least $1-\deg\left(B_{\tilde{A}}\right)/|F|-2/100(kd+1)$, the consistency check chooses an x_0 on which $B_{\tilde{A}}(x_0) \neq \tilde{q}(x_0)\cdot g_H(x_0)$ yet $p_A(x_0) = p_{\tilde{A}}(x_0)$ and $p_q(x_0) = p_{\tilde{q}}(x_0)$. In this case the consistency test rejects; this probability exceeds $\frac{1}{2}$. Summing up, the rejection probability is at least $1/\mathrm{polylog}(f(n))$.
- query: $\mathcal{O}(1)$ queries are necessary for each of the Reed-Solomon verifications, plus $\mathcal{O}(1)$ for each consistency check, i.e. $\mathcal{O}(1)$ in total. These queries are over alphabet F of size $\mathcal{O}(\log(f(n)))$. However, since all tests are computed by efficient circuits (of size $\mathrm{polylog}(f(n))$) we can reduce the alphabet size to constant, at the price of increasing the proof length by an additive factor of $2^{\mathsf{randomness}}\cdot\mathrm{polylog}(f(n)) = \tilde{\mathcal{O}}(f(n))$ and decreasing soundness by a multiplicative factor of $\mathrm{polylog}(f(n))$ (which still leaves it at $1/\mathrm{polylog}(f(n))$).

□

And now, at last, we have reached our destination:

Proof of Theorem 1.7 – "Short PCPs". Let $L \in \mathbf{NTIME}(f(n))$. By Theorem 8.1 we have

$$L \in \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = f(n)\cdot\mathrm{polylog}(f(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = \log(f(n)) + \mathcal{O}(1) \\ \mathsf{time} & = \mathrm{poly}(f(n)) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = 1/\mathrm{polylog}(f(n)) \end{array}\right)$$

Using techniques similar to those we had used in Homework assignment 1, Problems 2 and 3, we reduce the number of queries at some cost of reduced soundness (still bounded as $1/\mathrm{polylog}(f(n))$ though), obtaining

$$L \in \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = f(n)\cdot\mathrm{polylog}(f(n)) \\ \mathsf{randomness} & = \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = 2 \\ \mathsf{time} & = \mathrm{poly}(f(n)) \\ \mathsf{completeness} & = 1 \\ \mathsf{soundness} & = 1/\mathrm{polylog}(f(n)) \end{array}\right)$$

for some alphabet Σ of constant size (independent of n). We now employ Dinur's gap amplification method (used to prove Theorem 4.4) $\mathcal{O}(\log\log(f(n)))$-many times. Each application increases the proof size and soundness by a constant factor and since $\mathcal{O}(\log\log(f(n)))$

applications suffice to reach constant soundness $\varepsilon > 0$, the proof length increases only by a factor of $\text{polylog}(f(n))$, giving:

$$L \in \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{length} & = & f(n)\cdot \text{polylog}(f(n)) \\ \mathsf{randomness} & = & \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = & 2 \\ \mathsf{time} & = & \text{poly}(f(n)) \\ \mathsf{completeness} & = & 1 \\ \mathsf{soundness} & = & \varepsilon > 0 \end{array}\right)$$

Finally, we may employ expander-based amplification methods[6] to increase this soundness to $\frac{1}{2}$ at the cost of making more queries, but without using much more randomness. We thus reach

$$L \in \mathbf{PCP}\left(\begin{array}{lcl} \mathsf{length} & = & f(n)\cdot \text{polylog}(f(n)) \\ \mathsf{randomness} & = & \log(\mathsf{length}) + \mathcal{O}(1) \\ \mathsf{query} & = & \mathcal{O}(1) \\ \mathsf{time} & = & \text{poly}(f(n)) \\ \mathsf{completeness} & = & 1 \\ \mathsf{soundness} & = & 1/2 \end{array}\right)$$

(the number of queries is in fact $\mathcal{O}(\log(1/\varepsilon))$). □

10.3 On the completeness of UACSP

What remains to be done is describe a reduction proving Theorem 10.2; we will describe here rather, a simpler reduction, which incurs a quasi-quadratic (not quasi-linear) blowup of the input, but admits a simple proof of validity. For proof of the full-strength Theorem 10.2 consult [11].

Theorem 10.3 (Theorem 10.2, weakened)**.** *Theorem 10.2 stands with $k = 5$ and (universally fixed) integer d but with H only guaranteed to satisfy $|H| \leq \tilde{\mathcal{O}}\left(f(n)^2\right)$.*

We will prove Theorem 10.3 by reducing the following well-known language to it (see also (See also [37, Proof of Theorem 8.2]).

Definition 10.4. Let Σ be a finite alphabet. Considering sets of constraints of the form

$$\mathcal{C} = \left\{C_{i,j} : \Sigma^4 \to \{\mathsf{accept}, \mathsf{reject}\} \;\middle|\; i,j \in [f(n)]\right\}$$

and assignments $A : [f(n)]^2 \to \Sigma$, we define

$$\text{DOMINO}(n) = \left\{(1^n, \mathcal{C}) \;\middle|\; \exists\, A\left[\forall\, i,j\left[C_{i,j}\begin{pmatrix} A(i,j), A(i,j-1), \\ A(i+1,j), A(i,j+1)\end{pmatrix} = \mathsf{accept}\right]\right]\right\}$$

Theorem 10.5. *[15] The Language* DOMINO *is* **NTIME**$(f(n))$*-complete for reductions using $\mathcal{O}(\log(f(n)))$ space with quadratic blowup.*

[6] For a survey of randomness efficient sampling techniques see [25]

Proof. Apply the construction of Cooks SAT reduction for **NP** problems. The reduction constructs an $f(n) \times f(n)$ table, with each row representing a configuration at a certain point in time. The constraints each relate to some table position, and the neighborhood positions in the previous time point (previous table row); these correspond to the four parameters of the $C_{i,j}$ constraints. These are constrained so that the table describes a valid $f(n)$-time computation which begins with some input word and ends in an accepting state of the Turing Machine.
One may easily verify $x \in L$ iff the table satisfies the DOMINO constrains described above. □

Proof of Theorem 10.3. Using Theorem 10.5, it suffices to prove that DOMINO reduces to $\mathrm{UACSP}_{k,d}$ with quasi-linear blowup.
Let F be a field with $|F| > 100 \cdot f(n)^2$, let $\omega \in F$ be a generator of the multiplicative group $F \setminus \{0\}$, let Σ be identified with some $S \subseteq F$ with $|S| = |\Sigma|$, let $\mathsf{accept} = 0$ and let $\mathsf{reject} = 1$.
We can thus think of $C_{i,j}$ as a quadrivariate polynomial whose domain of interest to us is Σ^4; its degree in every one of its variables does not exceed $|\Sigma|$.
We define a mapping of $[f(n)]^2$ index pairs to elements of the multiplicative group by $(i,j) \mapsto \omega^{i \cdot f(n)+j}$. A UACSP assignment to the table cell variables will be a function with domain $H = \left\{\omega^{i \cdot f(n)+j} \;\middle|\; i,j \in [f(n)]\right\}$, and we can think of it as a low-degree interpolation of a DOMINO-assignment:

> EBS: *We need to be a bit more careful. $\tilde{A}$ has domain H (and range $\Sigma \subset F$). So extending it to domain F needs some explanation.*
>
> ER: *isn't it enough to say that it's the low-degree extension?*

$$\begin{array}{ccc} A & : & [f(n)]^2 \to \Sigma \\ & & \Big\downarrow \tilde{A}\left(\omega^{i \cdot f(n)+j}\right) = A(i,j) \\ \tilde{A} & : & F \to F \end{array}$$

The linear functions forming the constrains will be:

$$\begin{array}{lcl} L_1(x) = x & \iff & \text{table position } (i,j) \\ L_2(x) = x \cdot \omega & \iff & \text{table position } (i,j+1) \\ L_3(x) = x \cdot \omega^{f(n)} & \iff & \text{table position } (i+1,j) \\ L_4(x) = x \cdot \omega^{-1} & \iff & \text{table position } (i,j-1) \end{array}$$

We can thus expect the UACSP constraint polynomial to be

$$C_{\text{constraints}}(x, y_1, \ldots, y_4) = \sum_{i,j=1}^{f(n)} \left(\delta_{i,j}^H(x) \cdot C_{i,j}(y_1, \ldots, y_4)\right)$$

with $\delta_{i,j}^H$ the minimal degree monic polynomial satisfying:

$$\delta_{i,j}^H(x) = \begin{cases} 1 & x = \omega^{i \cdot f(n)+j} \\ 0 & x \in H \setminus \left\{\omega^{i \cdot f(n)+j}\right\} \end{cases}$$

One may verify that a polynomial $\tilde{A} \in F^{f(n)^2}[x]$ satisfies $C_{\text{constraints}}$ (that is, $B_A\big|_H \equiv 0$; see Definition 10.1) iff A satisfies the DOMINO constraints $\mathcal{C}$ and $A\big|_H \subseteq \Sigma$. To ensure this additional condition holds, it seems we would require a second constraint polynomial for checking A evaluates to Σ on H, namely we should require the following polynomial vanish on $x = A(z_0)$ for all $z_0 \in H$:

$$C_\Sigma(x) = \prod_{\sigma \in \Sigma} (x - \sigma)$$

Fortunately, it is possible to combine these two polynomials into a single one — although it is not trivial: One cannot simply multiply the polynomials, or degree-shift and multiply them. Instead, we add another unknown, y_5, another linear transform: $L_5(x) = x \cdot \omega^{-f(n)^2}$, double the size of H by letting $\hat{H} = \left\{ \omega^{i \cdot f(n)+j}, \omega^{f^2(n)+i \cdot f(n)+j} \;\middle|\; i \in \left[f(n)^2\right] \wedge j \in [f(n)] \right\}$ and redefining $\delta^{\hat{H}}_{i,j}$ using $\hat{H}$ instead of H. We now force A to evaluate to Σ on H by constraining a 'shifted copy' of H using the new variable y_5 as follows:

$$C(x, y_1, \ldots, y_5) = \sum_{i,j=1}^{n} \delta^{\hat{H}}_{i,j}(x) \cdot C_{i,j}(y_1, \ldots, y_4) + \sum_{i=n+1}^{2n} \sum_{j=1}^{n} \delta^{\hat{H}}_{i,j}(x) \cdot C_\Sigma(y_5)$$

This construction (due to [11, Section 4.1]) completes our reduction:

$$(1^n, \mathcal{C}) \Longrightarrow \Big(F, (L_1, \ldots, L_5), \hat{H}, C\Big)$$

Now, if $(1^n, \mathcal{C}) \in \text{DOMINO}(f(n))$, there exists an assignment $A : [f(n)]^2 \to \Sigma$ satisfying all $C_{i,j}$ constraints; this assignment extends to a UACSP assignment $\tilde{A}$ for which the $C_{\text{constraints}}(x, y_1, \ldots, y_4)$ indeed vanishes on all points in F corresponding to table cells (i.e. on H), as well as having $\tilde{A}\big|_H \subseteq \Sigma$; consequently, $\tilde{A}$ also makes $C(x, y_1, \ldots, y_5)$ vanish on the relevant points.

One can also show that, similarly, if $\tilde{A}$ satisfies C, i.e. $B_{\tilde{A}}\big|_{\hat{H}} = 0$ then the evaluation of $\tilde{A}$ on H is contained in Σ, i.e. $\tilde{A}$ encodes an assignment to the domino instance, and furthermore, this assignment satisfies the instance. □

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	January 12th, 2006

Lecture 11: Hardness of Approximation I: From PCPs to MIPs

Lecturer: Eli Ben-Sasson *Scribe: Anat Paskin and Eli Ben-Sasson*

11.1 Introduction

After proving the first part of Theorem 1.7 by constructing quasi-linear PCPs, we focus on the second part of the Theorem and construct over the next few lectures PCPs with high soundness and low query complexity (over the alphabet $\{0,1\}$). Such a construction is very useful for proving tight inapproximability results. In fact, the Theorem we'll prove will immediately imply that for some **NP**-hard maximization problems, the 'trivial' approximation algorithms are in fact optimal (unless $\mathbf{P} = \mathbf{NP}$)!
The main result we prove is the following Theorem due to Håstad [30]. After stating the Theorem we briefly discuss some implications of it to approximation problems. The rest of this Lecture as well as the next two ones will be devoted to sketching a proof of Håstad's Theorem.

Theorem 11.1 (Håstad [30])**.** *For all constant $\varepsilon, \eta > 0$,*

$$\mathbf{NP} \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = n^{\mathcal{O}(1)} \\ \mathsf{randomness} & = \mathcal{O}(\log(n)) \\ \mathsf{query} & = 3 \text{ over alphabet } \{0,1\} \\ \mathsf{time} & = n^{\mathcal{O}(1)} \\ \mathsf{completeness} & = 1-\varepsilon \\ \mathsf{soundness} & = 1/2-\eta \end{array}\right),$$

Moreover, the verifier is non-adaptive and accepts the answers to the three queries (denoted x_1, x_2, x_3) based on a linear constraint of the form '$x_1+x_2+x_3 = 1 \mod 2$' or '$x_1+x_2+x_3 = 0 \mod 2$'.

Remark. Note that Theorem 11.1 cannot hold with perfect completeness, that is with $\varepsilon = 0$ unless $\mathbf{P} = \mathbf{NP}$, since deciding satisfiability of a linear system is in **P**. The formulation in Theorem 1.7 does have *perfect* completeness and corresponds to a later improvement by Guruswami *et al.* [28]. However, the perfect-completeness three-query **PCP** does not use linear verifiers.

11.2 Applications of Håstad's Theorem 11.1 to Hardness of Approximation

Before stating two of the (numerous) Corollaries of Theorem 11.1, let us formally define the notion of 'hardness of approximation'.

Definition 11.2 (Maximization problem)**.** A maximization problem P is a tuple $(\mathcal{I}, f)$, where $\mathcal{I}$ is collection of instances, $W(I)$ denotes a set of 'witnesses' for each I, and

$$f : (I, W(I)) \to \mathbb{R}^+$$

is an efficiently computable function from instance/witness pairs $I \in \mathcal{I}, w \in W(I)$ to the reals. We denote $|I|$ by n. Led

$$OPT(I) = \max_{\{w \in W(I)\}} (f(I, w)).$$

We say a polynomial time algorithm A is an $\alpha(n)$-*approximator* for P, if for all $|I|$'s, $f(I, A(I)) \geq \alpha(n) \cdot OPT(I)$. A problem P is $\alpha(n)$-*hard for approximation*, if the existence of an $\alpha(n)$-approximator for P implies $\mathbf{P} = \mathbf{NP}$.

When studying hardness of approximation, the following type of promise problems are very useful.

Definition 11.3. Given a maximization problem P, we define a corresponding family of promise problems GAP-P(c, s) by

$$\textsc{Yes} = \{I \in P \mid OPT(I) \geq c\}$$
$$\textsc{No} = \{I \in P \mid OPT(I) \leq 1 - s\}$$

Remark. Minimization problems may be defined analogously. Also, other hardness assumptions (rather than NP-hardness) may be derived from **PCP** proof systems, and there are several works obtaining such results (see the surveys [2, 47] for examples and more information).

The first Corollary of Theorem 11.1 shows MAX-3LIN cannot be approximated beyond its 'trivial' approximation factor.

Definition 11.4 (MAX-3LIN)**.** An instance of MAX-3LIN is a set of variables $x_1, \ldots, x_k$, and a set of linear constraints of the form $x_{i_1} + x_{i_2} + x_{i_3} = b \mod 2$ over the aforementioned set of variables, where $b \in \{0, 1\}$. The witness set for I is the set of assignments to $x_1, \ldots, x_k$. $f(C, w)$ denotes the fraction of constraints satisfied by w.

Corollary 11.5. *For all constant $\varepsilon > 0$,* MAX-3LIN *is $\frac{1}{2} + \varepsilon$-hard for approximation.* [7]

Observation 11.6. Notice Corollary 11.5 gives a *tight* inapproximability result, because MAX-3LIN can be approximated in polynomial time to within a factor of $\frac{1}{2}$. The factor half approximation holds because the expected fraction of linear constraints satisfied by a random assignment is (by linearity of expectation) half. Furthermore, a witness w satisfying at least $\frac{1}{2}$ of the constraints can be efficiently found using conditional expectancy.

Proof of Corollary 11.5. Assume there exists a $(\frac{1}{2} + \varepsilon)$-approximator A for MAX-3LIN for some $\varepsilon > 0$. For any $\varepsilon', \eta > 0$, Theorem 11.1 implies a polynomial time reduction from $\mathrm{L} \in \mathbf{NP}$ to $\textsc{Gap-Max-3LIN}\left(1 - \varepsilon', \frac{1}{2} - \eta\right)$ (mapping $x \in L$ to YES instances, and $x \notin L$ to NO instances). Set $\varepsilon' = \eta = \varepsilon/4$. Consider the following algorithm B for deciding an **NP**-complete language L, operating on x, an instance of L:

[7] Of course, this result is only meaningful when $\varepsilon \leq \frac{1}{2}$

- Reduce x via Theorem 11.1 to an instance C of GAP-MAX-3LIN$\left(1-\varepsilon/4, \frac{1}{2}-\varepsilon/4\right)$.
- Apply the $(\frac{1}{2}+\varepsilon)$-approximator A to C.
- Output 'accept' iff $f(C, A(C)) \geq \frac{1}{2}+\varepsilon/3$, and output 'reject' otherwise.

Notice the algorithm B runs in polynomial time. We only need to analyze its correctness. If $x \in L$ then C is a YES-instance, so $OPT(C) \geq 1-\varepsilon/4$. Since A is a $(\frac{1}{2}+\varepsilon)$-approximator we have

$$f(C, A(C)) \geq (1/2+\varepsilon) \cdot (1-\varepsilon/4) \geq 1/2+\varepsilon/3,$$

so B outputs 'accept'. Similarly, if $x \notin L$ then

$$f(C, A(C)) \leq OPT(C) \leq 1/2+\varepsilon/4 < 1/2+\varepsilon/3,$$

so B outputs 'reject'. We conclude the existence of a $(\frac{1}{2}+\varepsilon)$-approximator for MAX-3LIN allows polynomial time decision of L, implying $\mathbf{P}=\mathbf{NP}$. This completes our proof. □

Another tight inapproximability result, for MAX-3SAT(defined below), follows from Corollary 11.5.

Definition 11.7 (MAX-3SAT)**.** An instance of MAX-3SAT is a set of variables $x_1, \ldots, x_k$, and a set of clauses $C = \{c\}$, where each $c \in C$ is of the form $c = x_{i_1} \vee x_{i_2} \vee x_{i_3}$ over the aforementioned set of variables. The witness set for I is the set of assignments to $x_1, \ldots, x_k$. $f(C, w)$ denotes the share of clauses in C satisfied by w. The maximization problem MAX-3SAT(5) is defined as MAX-3SAT with the additional restriction that each clause has exactly three variables and each variable appears in the clauses of C exactly five times.

Corollary 11.8. *For all $\varepsilon > 0$,* MAX-3LIN *is $7/8+\varepsilon$-hard for approximation.*

As in the case of MAX-3LIN, the existence of a 7/8-approximator for this problem can be based on linearity of expectation (and can be found deterministically by computing conditional expectancies).

Proof. First note a MAX-3LIN instance can be transformed to a MAX-3SAT one, by replacing the linear constraint $x+y+z=b$ over $\mathbb{F}_2$ with the following four clauses:

$$x+y+z=1 \mod 2 \mapsto (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (x \vee y \vee z),$$

$$x+y+z=0 \mod 2 \mapsto (\overline{x} \vee y \vee z) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee \overline{z}).$$

The key observation is that an assignment satisfying the left hand will satisfy all four clauses on the right, and an assignment not satisfying the left hand side will satisfy exactly three out of the four clauses on the right.
Let $\varepsilon > 0$, and assume contrarily to the claim there exists a $7/8+\varepsilon$-approximator A for MAX-3SAT. Consider the following algorithm B for deciding an **NP**-complete language L, operating on instance x:

- Employing the reduction outlined in Corollary 11.5, reduce x to an instance C of GAP-MAX-3LIN$\left(1-\varepsilon/4, \frac{1}{2}-\varepsilon/4\right)$.

- Reduce each linear constraint of C to four clauses, using the reduction mentioned earlier in our proof. Let C' denote the (3CNF) conjunction of all these clauses.
- Output '`accept`' iff $f(C', A(C')) \geq 7/8 + \varepsilon/3$ and otherwise output '`reject`'.

The running time of B is polynomial, so we only need analyze its correctness. If $x \in L$ then $OPT(C') \geq OPT(C) \geq 1 - \varepsilon/4$, so

$$f(C', A(C')) \geq (1 - \varepsilon/4) \cdot (7/8 + \varepsilon) \geq 7/8 + \varepsilon/3,$$

and we output '`accept`'. Similarly, if $x \notin L$ then

$$f(C', A(C')) \leq OPT(C') \leq \left(\frac{1}{2} + \varepsilon/4\right) + \frac{3}{4}\left(\frac{1}{2} - \varepsilon/4\right) \leq 7/8 + \varepsilon/16 < 7/8 + \varepsilon/3.$$

Therefor, the existence of a $(7/8 + \varepsilon)$-approximator for MAX-3SATimplies $\mathbf{P} = \mathbf{NP}$ and our proof is complete. □

11.3 Outline of the proof of Håstad's Theorem 11.1

We will prove Theorem 11.1 using the following four steps:

I. We start with a basic PCP for **NP** as stated in the first part of Theorem 1.7. This system has $\mathcal{O}(1)$ queries, perfect completeness and 'small' constant soundness $\varepsilon > 0$. As shown in Homework assignment 1, this implies it is **NP**-hard to decide Gap-Max-3SAT$(1, \varepsilon')$ for some $\varepsilon' > 0$. Now we reduce this gap problem to the more structured Gap-MAX-3SAT$(5)(1, \varepsilon'')$, in which each clause has exactly three literals and each variable has exactly five appearances in clauses. (We won't show the details of this reduction). The extra structure of our gap problem will be useful in the last step of our analysis.

II. We have reached a PCP system with three queries, perfect completeness, and small constant soundness $\varepsilon'' > 0$. We need a way to amplify soundness arbitrarily close to 1, while using a very small number of queries and constant alphabet size. Keeping the number of queries small (≤ 3) is crucial, since we can then hope to reduce the alphabet to binary by encoding the symbols with a good binary error correcting code. The only way known today of achieving arbitrary soundness while using only two queries is via parallel repetition. To apply parallel repetition we transform the PCP system into a special constraint graph over a regular bi-partite graph, called a 'label-cover' instance.

III. At this stage, given $\varepsilon'' > 0$ and arbitrary $\delta > 0$, we apply Raz' parallel repetition theorem [40] to the constraint graph from the previous stage. This gives us a PCP system with two queries (over constant size alphabet), perfect completeness and soundness $1 - \delta$. Both running time and alphabet size depend on ε'' and δ. However, for constant $\varepsilon'', \delta > 0$ the running time is polynomial and the alphabet size constant.

IV. In the final stage, we have obtained a sufficiently sound, two query, 'outer' **PCP**. We now encode each (constant size) proof symbol (corresponding to an answer to a single

query) by an error correcting code. Since the answer length is constant, we can afford using a very long code with good local testability and decodability properties. We will use the aptly called 'long code' to encode our proof symbols, and apply Fourier analysis to obtain tight estimation of the completeness and soundness of our tests.

In what follows we will describe the last three stages outlined above. The first step (reducing **NP** to GAP-MAX-3SAT(5)$(1, \varepsilon'')$) is rather straightforward and can be found in [19, Section 2].

11.4 Phase II - Reduction to Label Cover

The parallel repetition Theorem of Raz that we shall use to amplify soundness, is usually described in the literature using notions of *interactive multi-prover systems* (see [40] for definitions and more information). In such systems (historically predating PCPs) the verifier interacts with two computationally unbounded provers. The investigation of multi-prover systems is a major theme of theoretical computer science (see the survey [24] for more information). However, following [31] and favoring simplicity over historical context, we will state Raz's Theorem using a special case of it formulated in terms of bipartite constraint graphs. Recall the following definitions and notation: Constraint graph $\mathcal{G}$ (Definition 4.7), its assignment A (Definition 4.3) and soundness $s(\mathcal{G})$ (Definition 4.9) and the promise problem GAP-CG(Σ, s) (Definition 4.11).

Definition 11.9 (Label Cover)**.** Let $d, M, N \in \mathbb{N}^+$, an instance of LABEL-COVER(d, M, N) is a constraint graph

$$\mathcal{G} = (G, \Sigma, \mathcal{C} = \{C_e \mid e \in E\})$$

satisfying the following additional requirements:

- G is a bi-partite (multi)-graph over vertex set $W \cup V$ and edge set E.
- G is regular with right degree d and left degree $d|W|/|V|$.
- Σ is the union of Σ_W (left colors) and Σ_V (right colors), where $|\Sigma_W| = M, |\Sigma_V| = N$.
- The only 'legal' assignments give W values in Σ_W and give V values in Σ_V. Formally, for every edge $e = (w, v), w \in W, v \in V$ and assignment $\sigma_1, \sigma_2 \in \Sigma$,

$$\text{If } (\sigma_1 \notin \Sigma_W \text{ or } \sigma_2 \notin \Sigma_V) \text{ then } C_e(\sigma_1, \sigma_2) = \mathsf{reject}.$$

The language LABEL-COVER(d, M, N) is the set of satisfiable instances, for which $s(\mathcal{G}) = 0$. The promise problem LABEL-COVER$(d, M, N, \hat{s})$ has:

- YES $= \{\mathcal{G} \mid$ is a LABEL-COVER(d, M, N) instance and $s(\mathcal{G}) = 0\}$.
- NO $= \{\mathcal{G} \mid$ is a LABEL-COVER(d, M, N) instance and $s(\mathcal{G}) \geq \hat{s}\}$.

W.l.o.g. an assignment to an instance of Label-Cover$(d, M, N, \hat{s})$ can be thought of as a pair of assignments $A_W : W \to \Sigma_W, A_V : V \to \Sigma_V$, because forcing this restriction does not reduce the perfect completeness (an assignment satisfying all constraints must be of this form) and the soundness can only increase. Thus, from here on we assume all assignments are of this form. The following Theorem is a straightforward corollary of the **NP**-hardness of MAX-3SAT(5)$(1, \varepsilon'')$.

Theorem 11.10. *There exists constant $\varepsilon''' > 0$ such that it is* **NP***-hard to decide* Gap-Label-Cover$(d = 5, M = 7, N = 2, s = \varepsilon''')$.

Proof. We start with MAX-3SAT(5)$(1, \varepsilon'')$. Let ϕ be an instance of this promise language, with m clauses $\phi_1, \ldots, \phi_m$ and $n = 3m/5$ variables $x_1, \ldots, x_n$. We construct the following regular bi-partite multi-graph. Let W be the set of clauses, V be the set of variables, and connect ϕ_i to x_j iff x_j appears in ϕ_i. Notice G is a regular (multi) graph with right degree 5, left degree 3, $|W| = m$ and $|V| = n$. The left alphabet is $\Sigma_W = \{\sigma_1, \ldots, \sigma_7\}$ and the right one is $\Sigma_V = \{0, 1\}$. We next describe the edge constraints. Let $e = (\phi_i, x_j) \in E$ where x_j appears in ϕ_i. Notice there are exactly seven assignments to the variables of ϕ_i that satisfy ϕ_i. We associate them arbitrarily with Σ_W. For $\sigma \in \Sigma_W, b \in \{0, 1\}$, we define

$$C_e(a, b) = \begin{cases} \mathsf{accept} & \text{the restriction of } \sigma \text{ to variable } x_j \text{ equals } b \\ \mathsf{reject} & \text{otherwise} \end{cases}$$

Having defined our reduction, we are left with arguing completeness and soundness. As usual, completeness is easy. If ϕ is satisfiable we set A_V to equal a satisfying assignment, and A_W will be the corresponding (satisfying) assignment to all clauses.
Regarding soundness, suppose ϕ is a No-instance of MAX-3SAT(5)$(1, \varepsilon'')$. Notice any assignment $A_V : V \to \{0, 1\}$ corresponds to an assignment to ϕ. Thus, it falsifies at least ε'' clauses. Thus, with probability ε'' we select on the left hand side a vertex corresponding to a clause ϕ_i unsatisfied by A_V. But the assignment to ϕ_i by definition satisfies ϕ_i, so it must be inconsistent with at least one variable under the assignment A_V. We conclude the rejection probability is at least $\varepsilon''' = \varepsilon''/3 > 0$. □

11.5 Phase III - Soundness amplification via parallel repetition

To boost soundness, we define a graph powering operation on Label-Cover instances. For $G = (W, V, E)$ a bi-partite graph and integer k, let $G^{\otimes k}$ be the bi-partite graph with left vertex set $W^k = \underbrace{W \times \ldots \times W}_{k \text{ times}}$, right vertex set V^k and edge set

$$E^{\otimes k} = \left\{ (\bar{w}, \bar{v}) \in W^k \times V^k \;\middle|\; (w_i, v_i) \in E(G) \text{ for } i = 1, \ldots, k \right\}$$

Definition 11.11 (Repetition powering)**.** Let $\mathcal{G} = \{G, \Sigma_V \cup \Sigma_W, \mathcal{C}\}$ be a constraint graph instance of Label-Cover(d, M, N) and let $k \geq 1$. The k-powering of $\mathcal{G}$ is defined to be

$$\mathcal{G}^{\otimes k} = \left\{ G^{\otimes k}, \Sigma_W^k \cup \Sigma_V^k, \mathcal{C}^{\otimes k} \right\}$$

Where the constraint for each edge $(\bar{w},\bar{v}) \in E\left(\mathcal{G}^{\otimes k}\right)$ is

$$C_e\big((\sigma_1,\ldots,\sigma_k),(\sigma'_1,\ldots,\sigma'_k)\big) = \mathsf{accept} \Leftrightarrow \bigwedge_{i=1}^{k} \left(C_{(w_i,v_i)}(\sigma_i,\sigma'_i) = \mathsf{accept}\right)$$

The following is a special case of the celebrated Parallel Repetition Theorem of Raz. The general statement (appearing in [41]) applies to any two-prover, one round, interactive proof (of which our Label-Cover instances are but a special case).

Theorem 11.12 ([41])**.** *For any integers N, M, d there exists a constant $\alpha > 0$ depending only on N, M such that the following holds. For any instance $\mathcal{G}$ of* Label-Cover(d, M, N) *and all integers k,*

$$s\left(\mathcal{G}^{\otimes k}\right) \geq 1 - (1 - s(\mathcal{G}))^{\alpha k}.$$

Remark. A different soundness amplification technique, very similar to parallel repetition, was shown by Feige and Kilian [21]. Their result can also be applied to obtain inapproximability results and in certain settings outperforms parallel repetition. We refer the reader to [21] for more information and a comparison of the two soundness amplification techniques. In this course we use the simpler-to-state Theorem 11.12, that also has better parameters in the setting we're interested in (constant alphabet size and constant initial soundness).

Applying Theorem 11.12 to Theorem 11.10 boost the soundness arbitrarily close to one.

Theorem 11.13. *For any constant $\varepsilon > 0$ there exists an integer k depending only on ε such that it is* **NP***-hard to decide* Gap-Label-Cover$\left(d = 5^k, M = 7^k, N = 2^k, s = 1 - \varepsilon\right)$.

Proof. We start with an instance $\mathcal{G}$ of the **NP**-hard language Gap-Label-Cover$(5, 7, 2, \varepsilon''')$ from Theorem 11.10. Let $\alpha = \alpha(M, N)$ be the constant from Raz's Theorem 11.12. Take k to be the minimal integer such that $(1 - \varepsilon''')^{\alpha k} < \varepsilon$ and notice k depends only on ε, because ε''', α are fixed. Consider the polynomial time reduction sending $\mathcal{G}$ to $\mathcal{G}^{\otimes k}$. By Definition 11.11, $\mathcal{G}^{\otimes k}$ is an instance of Label-Cover$\left(d^k, 7^k, 2^k\right)$. Completeness is easy to argue, because

$$\mathcal{G} \in \text{Label-Cover}(5, 7, 2) \Rightarrow \mathcal{G}^{\otimes k} \in \text{Label-Cover}\left(5^k, 7^k, 2^k\right).$$

Regarding soundness, assume $\mathcal{G}$ is a No-instance, i.e. $s(\mathcal{G}) \geq \varepsilon'''$. Theorem 11.12 implies $s(\mathcal{G}^{\otimes k}) \geq 1 - \varepsilon$. This completes our proof. □

A comparison of Raz's Theorem 11.12 with Dinur's Gap amplification (Proposition 7.1) is due. On one hand, Raz's Theorem implies for any pair of constants $0 < s < s' < 1$, we can boost the soundness from s to s' by a constant number of parallel repetitions. And it is precisely the ability to boost soundness arbitrarily close to one that will be crucial for proving Håstad's Theorem 11.1 (as well as for proving many other inapproximability results). Recall Dinur's gap amplification fails to boost soundness beyond $s = \frac{1}{2}$ (as argued in Homework assignment 2). On the other hand, Raz's soundness amplification comes with a large price-tag in terms of size, because parallel repetition gives $\left|\mathcal{G}^{\otimes k}\right| \approx |\mathcal{G}|^k$ whereas Dinur's graph powering of a d-regular graph gives $\left|\mathcal{G}^k\right| \approx d^k \cdot |\mathcal{G}|$ (and typically $d = \mathcal{O}(1)$). Achieving the best of the two worlds, i.e. boosting soundness arbitrarily close to 1 with only a small increase in problem size, is a major open problem in this field.

At first sight, Raz's Theorem 11.12 looks easy to prove. If $s(\mathcal{G}) = s$, then for any assignment to G, selecting k edges and repeatedly querying both end points will cause the acceptance probability to be at most $(1-s)^k$. In other words, the soundness of k-wise *sequential* repetition is at least $1-(1-s)^k$. How can the *parallel* version do any better? Actually, in the early days of PCP research it was believed the parallel repetition bound is equivalent to the sequential one, i.e. the parameter α appearing in the statement of Theorem equals one. However, as counter-examples emerged (one such example is given in Homework assignment 3), it became evident that proving the parallel version would be harder than the (trivial) sequential case. Indeed, Raz's proof of Theorem 11.12 is uses some heavy information-theoretic machinery (beyond the scope of this course). Finding a simpler/different proof of this fundamental Theorem remains an important challenge.
For the rest of the lecture we will discuss some of the problems of the soundness analysis of parallel repetition. As mentioned above, parallel repetition provides somewhat weaker soundness than expected at first glance (because α is typically smaller than 1). Consider a 2-wise (parallel or sequential) repetition over an instance $\mathcal{G}$. For simplicity assume $G = (W, V, E)$ where $|W| = n$ and the vertices of W are indexed by $\{1, \ldots, n\}$. In this scenario, verifier selects two edges $(w_1, v_1), (w_2, v_2) \in E$. In the sequential case, verifier makes four queries: w_1, v_1, w_2, v_2. In the parallel case, the verifier makes only two queries: (w_1, w_2) and (v_1, v_2). In both cases we have

$$\mathbf{Pr}[\mathsf{accept\ both\ trials}] = \mathbf{Pr}[\mathsf{accept\ first\ trial}] \cdot \mathbf{Pr}[\mathsf{accept\ second\ trial} | \mathsf{accept\ first\ trial}]$$

However, in the case of parallel repetition the answers of the provers in the first trial may depend on the queries for the second trial (w_2, v_2), so success in the first trial may give additional information about the pair in the second trial, thus raising the conditional success probability there.
For concreteness, suppose the first prover answers w_1 with (say) the parity of the index w_2.[8] Now, the second prover receives v_1, v_2, thus has some initial information about the possible values for w_2 (it is a neighbor of v_2 in G). The second prover also knows her answer to the first query. The additional knowledge of success in the first trial gives her additional information about the identity of w_2 – it is a vertex whose index-parity (when given as an answer by the first prover in the first round) causes the first round to succeed. This information may increase the success probability of the second prover in the second round! If the above example seems somewhat contrived, Homework 3 shows an explicit example where the two-wise parallel repetition (of a general **MIP** game) does not increase the soundness at all! (and the reason is precisely the passage of information as outlined above).

What next? With this we end our (far too brief) discussion of soundness amplification via parallel repetition. In the next couple of lectures we will start with a two-query **PCP** system with perfect completeness and arbitrarily large soundness over a large constant sized alphabet, and see how to reduce the alphabet to $\{0,1\}$, using three queries and $\mathsf{soundness} \approx \frac{1}{2}, \mathsf{completeness} \approx 1$.

[8] Notice we're only assuming $|\Sigma_W| \geq 2$. If Σ_W is larger than two, then the first prover can 'pass even more information' about w_2 in her first-round answer.

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	January 12th, 2006

Lecture 12: Revisiting the BLR Linearity test with Fourier analysis

Lecturer: Eli Ben-Sasson *Scribe: Dan Baum and Eli Ben-Sasson*

To complete the proof of Håstad's Theorem 11.1 we need to introduce tools from analysis of Boolean functions using Fourier representations. As a warm-up, we will use these tools to improve the soundness parameter of the BLR linearity test (Theorem 2.9). Next lecture we will employ our new toolbox to complete the proof of Håstad's Theorem 11.1 (by proving Phase VI from section 11.3).

12.1 Fourier representation of Boolean functions

Let $\mathcal{F}_n$ be the set of functions $f : \{0,1\}^n \to \mathbb{R}$ whose domain is Boolean and range is the real numbers. f can be represented as a vector of 2^n real numbers

$$(f(00\ldots00), f(00\ldots01), \ldots f(11\ldots11)).$$

In other words, $\mathcal{F}_n$ is a 2^n-dimensional vector space over $\mathbb{R}$. The *standard* basis for this space is the set of *delta* functions $\{\delta_\alpha(x) \mid \alpha \in \{0,1\}^n\}$ where

$$\delta_\alpha = \left(0, 0\ldots0, \underbrace{1}_{\alpha}, 0, \ldots, 0\right).$$

We now describe a different basis for $\mathcal{F}_n$ called the *Fourier* basis, whose basis-elements are the *characters* F_2^n described below, where F_2 is the two-element finite field.

Definition 12.1 (Character)**.** For $\alpha \in \{0,1\}^n$, the *α-character* is the function

$$\chi_\alpha : \{0,1\}^n \to \{1,-1\}$$
$$x \xmapsto{\chi_\alpha} (-1)^{\sum_{i=1}^n \alpha_i \cdot x_i} = \prod_{i=1}^n (-1)^{\alpha_i x_i}$$

Remark. Characters are studied in 'representation theory', the theory dealing with *representations* of groups as matrices. In general, a character is a homomorphism from a group G to the complex unit circle (consisting of complex numbers with absolute value one). In Definition 12.1, G is the additive group $\{F_2^n, +\}$ and a character maps this group to $\{1,-1\} \subset \mathbb{C}$. For more information see e.g. [33, Chapter 5], [44].

A useful property of characters, whose proof follows by inspection, is summarized below. Throughout this lecture, the bit-wise xor operation is denoted by $\oplus$.

Claim 2. *For all $\alpha, \beta, x, y \in \{0,1\}^n$,*

$$\chi_\alpha(x) \cdot \chi_\beta(x) = \chi_{\alpha\oplus\beta}(x) \quad \textit{and} \quad \chi_\alpha(x) \cdot \chi_\alpha(y) = \chi_\alpha(x \oplus y).$$

Notice there are 2^n distinct characters (one for each $\alpha \in \{0,1\}^n$), thus we have the right number of elements to form a basis for $\mathcal{F}_n$. Do the characters form one? To answer this let us introduce an *inner product* over the 2^n-dimensional space $\mathcal{F}_n$ given by

$$\langle f\,,\,g\rangle \stackrel{\text{def}}{=} 2^{-n}\sum_{x\in\{0,1\}^n} f(x)\cdot g(x) = \mathop{\mathbf{Ex}}_{x}[f(x)g(x)]. \tag{8}$$

Recall an inner-product defines a *norm* on vectors, given by $\|v\| \stackrel{\text{def}}{=} \sqrt{\langle v\,,\,v\rangle}$. Recall a set of vectors V is *ortho-normal* (with respect to an inner product) if $\|v\| = 1$ (normality) and $\langle u\,,\,v\rangle = 0$ for all $u \neq v \in V$ (orthogonality). Recall a ortho-normal set of N vectors residing in a N-dimensional vector space over $\mathbb{R}$ forms a *ortho-normal basis*.

Claim 3. *The characters form an ortho-normal basis for $\mathcal{F}_n$.*

Proof. For $\alpha,\beta \in \{0,1\}^n$ and $\gamma = \alpha \oplus \beta$, Claim 2 and Equation (8) imply

$$\langle \chi_\alpha\,,\,\chi_\beta\rangle = \mathop{\mathbf{Ex}}_{x}[\chi_\gamma(x)].$$

If $\alpha = \beta$ then $\gamma = \vec{0}$ and $\mathbf{Ex}_x\left[\chi_{\vec{0}}\right] = 1$, because $\chi_{\vec{0}}(x) = 1$ for all x. We conclude the characters are *normal*. To prove they are *orthogonal* let $\alpha \neq \beta$ and assume wlog $\gamma_1 = 1$. Then

$$\langle \chi_\alpha\,,\,\chi_\beta\rangle = \mathop{\mathbf{Ex}}_{x}[\chi_\gamma(x)] = \mathop{\mathbf{Ex}}_{x}\left[\prod_{i=1}^{n}(-1)^{\gamma_i x_i}\right]$$

(by independence of the x_is)

$$= \prod_{i=1}^{n}\mathop{\mathbf{Ex}}_{x_i}[(-1)^{\gamma_i x_i}]$$

$$= \mathop{\mathbf{Ex}}_{x_1}[(-1)^{x_1}]\cdot\prod_{i=2}^{n}\mathop{\mathbf{Ex}}_{x_i}[(-1)^{\gamma_i x_i}]$$

(notice $\mathbf{Ex}_{x_1}[(-1)^{x_1}] = \frac{1}{2}(1+(-1)) = 0$, so)

$$= 0$$

We conclude the characters form an ortho-normal set of cardinality 2^n, i.e. they form an orhto-normal basis for $\mathcal{F}_n$. □

Claim 3 implies every function in (the vector-space) $\mathcal{F}_n$ can be represented as a (unique) linear combination of characters, i.e.

$$f(x) = \sum_{\alpha}\hat{f}_\alpha\cdot\chi_\alpha(x),$$

where $\hat{f}_\alpha \in \mathbb{R}$ is called the *α-coefficient* of f under the Fourier basis representation. Moreover, $\hat{f}_\alpha$ is computed by taking the inner product of f and χ_α, i.e. it is the *projection* of (the vector) f on (the basis vector) χ_α,

$$\hat{f}_\alpha = \langle f \, , \, \chi_\alpha \rangle. \tag{9}$$

We conclude with a very useful property to be used later on, saying that representing f under the Fourier basis does not change it's norm

$$\textbf{Parseval's Equality: } \text{For all } f \in \mathcal{F}_n, \sum_{\alpha \in \{0,1\}^n} \left(\hat{f}\alpha\right)^2 = \|f\|^2 \tag{10}$$

Notice Parseval's Equality implies that if the range of f is $\{1, -1\}$ (as we'll soon assume it to be), then $\|f\| = 1$ so the sum of squares of it's Fourier coefficients is one.

12.2 Fourier analysis of the BLR linearity test

We now use the Fourier representation to analyze the Blum-Luby-Rubinfeld (BLR) test of the Hadamard code. Recall the definitions of the n-dimensional Hadamard code H_n (Definition 2.5); Hamming distance $\Delta(\cdot,\cdot)$ and relative Hamming distance $\Delta^*(\cdot,\cdot)$ (Definition 2.2); and the BLR tester T_H (appearing in the proof of Theorem 2.9).
We are given oracle access to a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and wish to distinguish between the case $f \in H_n$ (or $\Delta^*(f, H_n) = 0$) and the case $\Delta^*(f, H_n)$ is large. The BLR test addresses this problem by randomly and independently selecting $x, y \in \{0,1\}^n$ and accepting iff

$$f(x) \oplus f(y) \oplus f(x \oplus y) = 0.$$

Let us change the range of f from $\{0,1\}$ to $\{1,-1\}$. Formally, let $\tilde{f} : \{0,1\}^n \to \{1,-1\}$ be the function defined by $\tilde{f}(x) \stackrel{\text{def}}{=} (-1)^{f(x)}$. The reason for this transformation is given by the following claim.

Observation 12.4. For $\alpha \in \{0,1\}^n$ let $\ell_\alpha : F_2^n \to F_2$ be the linear function as in Definition 2.5. Identifying the two-element field F_2 with $\{0,1\}$ we have $\tilde{\ell_\alpha} \equiv \chi_\alpha$, i.e. for all $x \in \{0,1\}^n$, $\chi_\alpha(x) = (-1)^{\ell_\alpha(x)}$.

Consider the possible benefit from our new view. We're interested in estimating the distance of f to H_n. Now, our problem is transformed to that of measuring the distance of the *vector* $\tilde{f}$ to the ortho-normal basis of characters. And to solve this problem, we only need to know the projection of f onto the various characters (as shown by the following claim).

Claim 5. *For $f, g : \{0,1\}^n \to \{0,1\}$ and $\tilde{f}, \tilde{g}$ as defined above,*

$$\Delta^*(f,g) = 1/2\left(1 - \left\langle \tilde{f} \, , \, \tilde{g} \right\rangle\right)$$

Proof. If $f(x) = g(x)$ then $\tilde{f}(x)\tilde{g}(x) = 1$ whereas $f(x) \neq g(x)$ implies $\tilde{f}(x)\tilde{g}(x) = -1$. Thus,

$$\Delta^*(f,g) = \mathop{\mathbf{Ex}}_x[f(x) \neq g(x)] = \mathop{\mathbf{Ex}}_x\left[\frac{1 - \tilde{f}(x)\tilde{g}(x)}{2}\right] = 1/2\left(1 - \left\langle \tilde{f} \, , \, \tilde{g} \right\rangle\right).$$

□

We are ready to prove the main result of this lecture, namely the improved-soundness version of Theorem 2.9. The idea of our proof is to express the success probability of the BLR tester on oracle function f in terms of the Fourier coefficients of $\tilde{f}$. In particular, we'll show that a large success probability implies the existence of a large Fourier coefficient $\hat{f}_\alpha$. Using Claim 5 we will conclude f is close to the linear function ℓ_α.

Theorem 12.6. *The Hadamard codes* $\{H_n \mid n \in \mathbb{N}^+\}$ *are* $(\mathsf{time} = \mathcal{O}(n), \mathsf{randomness} = 2n, \mathsf{query} = 3, \mathsf{soundness} = s(\delta) \geq \delta)$*-LTC's, with the BLR linearity tester achieving these parameters.*

Remark. Recall Theorem 2.9 claimed $s(\delta) \geq \min(\delta/2, 2/9)$, which is weaker than our new statement. We point out the original proof of Theorem 2.9 does have an advantage over our new one. Namely, the 'combinatorial' proof in Lecture 2 can be applied to the problem of Homomorphism testing over any group. In contrast, the new proof below works only for homomorphism testing over groups that can be represented using the Fourier basis, and it turns out that groups having this property are necessarily Abelian.

Proof. The running time, randomness, query complexity and completeness were argued in the proof of Theorem 2.9. We focus on the soundness. Note

$$
\begin{aligned}
& \mathbf{Pr}[\mathsf{accept}] \\
= \; & \mathop{\mathbf{Pr}}_{x,y}[f(x) \oplus f(y) = f(x \oplus y)] \\
= \; & \mathop{\mathbf{Ex}}_{x,y}\left[\frac{1}{2}\Big(1 + \tilde{f}(x)\tilde{f}(y)\tilde{f}(x \oplus y)\Big)\right] = \frac{1}{2}\left(1 + \mathop{\mathbf{Ex}}_{x,y}\Big[\tilde{f}(x)\tilde{f}(y)\tilde{f}(x \oplus y)\Big]\right)
\end{aligned}
$$

(take the Fourier representation of $\tilde{f}$ and use $\hat{f}_\alpha$ to denote the α-coefficient of $\tilde{f}$)

$$
= \frac{1}{2}\left(1 + \mathop{\mathbf{Ex}}_{x,y}\left[\left(\sum_{\alpha \in \{0,1\}^n} \hat{f}_\alpha \chi_\alpha(x)\right) \cdot \left(\sum_{\beta \in \{0,1\}^n} \hat{f}_\beta \chi_\beta(y)\right) \cdot \left(\sum_{\gamma \in \{0,1\}^n} \hat{f}_\gamma \chi_\gamma(x \oplus y)\right)\right]\right)
$$

(linearity of expectation allows us to change order of summation)

$$
= \frac{1}{2}\left(1 + \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \cdot \mathop{\mathbf{Ex}}_{x,y}[\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(x \oplus y)]\right)
$$

(Claim 2 implies $\chi_\gamma(x \oplus y) = \chi_\gamma(x)\chi_\gamma(y)$ and $\chi_\alpha(x)\chi_\gamma(x) = \chi_{\alpha \oplus \gamma}(x)$ and $\chi_\beta(y)\chi_\gamma(y) = \chi_{\beta \oplus \gamma}(y)$)

$$
= \frac{1}{2}\left(1 + \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \cdot \mathop{\mathbf{Ex}}_{x,y}[\chi_{\alpha \oplus \gamma}(x)\chi_{\beta \oplus \gamma}(y)]\right)
$$

(independence of x, y allows us to multiply expectations)

$$= \frac{1}{2}\left(1 + \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \cdot \mathop{\mathbf{Ex}}_x[\chi_{\alpha\oplus\gamma}(x)] \cdot \mathop{\mathbf{Ex}}_y[\chi_{\beta\oplus\gamma}(y)]\right)$$

Claim 3 implies $\mathbf{Ex}_x[\chi_{\alpha\oplus\gamma}(x)] = 0$ unless $\alpha = \gamma$. Similarly $\mathbf{Ex}_y[\chi_{\beta\oplus\gamma}(y)] = 0$ unless $\alpha = \gamma$. Thus, we only sum over $\alpha = \beta = \gamma$. For this case ($\alpha \oplus \gamma = \beta \oplus \gamma = \vec{0}$), Claim 3 implies $\mathbf{Ex}_x\left[\chi_{\vec{0}}\right] = 1$, so

$$\mathbf{Pr}[\mathsf{accept}] = 1/2\left(1 + \sum_\alpha \hat{f}_\alpha^3\right)$$

Assume $\rho \le \mathbf{Pr}[\mathsf{accept}]$, so by our previous calculation

$$\rho \le 1/2\left(1 + \sum_\alpha \hat{f}_\alpha^3\right),$$

or

$$2\rho - 1 \le \sum_\alpha \hat{f}_\alpha^3.$$

Pareval's Equality (10) implies $\sum_\alpha \hat{f}_\alpha^2 = 1$. Notice $\hat{f}_\alpha^2 \in [0,1]$, so think of $\hat{f}_\alpha^2$ as defining a probability D over $\{0,1\}^n$. Taking this view, the previous inequality says the expectation of $\hat{f}_\alpha$ according to D is at least $2\rho - 1$. This implies there exists α such that

$$\hat{f}_\alpha \ge 2\rho - 1. \tag{11}$$

We are almost done. Informally, a large value of $\hat{f}_\alpha$ means the projection of $\tilde{f}$ onto χ_α is large. But this large projection implies large proximity of f to ℓ_α. Formally, Claim 5 gives

$$\Delta^*(f, \ell_\alpha) = \frac{1}{2}\left(1 - \left\langle \tilde{f}\,,\, \chi_\alpha \right\rangle\right)$$

Equation (9) implies

$$= \frac{1}{2}\left(1 - \hat{f}_\alpha\right)$$

Equation (11) implies

$$\le \frac{1}{2}(1 - (2\rho - 1)) \le 1 - \rho.$$

We have proved

$$\mathbf{Pr}[\mathsf{accept}] \ge \rho \Rightarrow \Delta^*(f, H_n) \le 1 - \rho.$$

Set $\delta = 1 - \rho$ and reverse the implication above,

$$\Delta^*(f, H_n) > \delta \Rightarrow \mathbf{Pr}[\mathsf{accept}] < 1 - \delta,$$

or in other words, the BLR-soundness function satisfies $s(\delta) \ge \delta$ as claimed. Our proof is complete □

Technion **IIC: 236610**	From error-correcting codes to hardness of approximation in light of the PCP Theorem	January 26th, 2006

Lecture 13: Hardness of Approximation III: Testing for Dictatorship

Lecturer: Eli Ben-Sasson *Scribe: Eyal Rozenberg and Eli Ben-Sasson*

13.1 Overview

In this lecture we complete the proof of Håstad's Theorem 11.1 by completing Phase IV of section 11.3. Our starting point is the gap-problem from section 11.5 restated below.

Theorem 13.1. *Theorem 11.13 For any constant $\varepsilon > 0$ there exists an integer k depending only on ε such that it is* **NP***-hard to decide* Gap-Label-Cover$\big(d = 5^k, M = 7^k, N = 2^k, s = 1 - \varepsilon\big)$.

The above gap-problem has a natural verifier with perfect completeness and high-soundness. Namely, verifier picks a random edge $e = (w, v)$ and accepts iff the edge-constraint C_e is satisfied by the assignment to w, v.
In order to complete our proof, we need to reduce the alphabet size from $\max\{M, N\}$ to 2 and construct a three-query linear test for the new problem that will have large completeness and soundness.
To reduce the alphabet size we will ask our prover to encode each symbol in $[M], [N]$ (i.e. value of her assignment) by a special code (over the binary alphabet) called the *long code*, which we describe in the next section.
Following that, we will use a modified version of the (three query) BLR linearity test to test proximity to the long-code. However, this test is not sufficient for our purposes. We are given encodings of assignments to vertices w, v and need to test that the *un-encoded* assignments satisfy a constraint. The final part of our proof will be to analyze a test that makes two queries to the encoding of w and one query to the encoding of v that achieves our goal.

13.2 The long code

To define the long-code, we start by defining a simple, yet important family of functions, that correspond to codewords of the long-code. Recall $\ell_\alpha : F_2^n \to F_2$ is the linear function given in Definition 2.5. We identify the two-element field F_2 with $\{0, 1\}$.

Definition 13.2 (Dictatorships)**.** A function $f : \{0,1\}^N \to \{0,1\}$ is a *dictatorship* if there exists $i \in [N]$ such that the value of $f(x)$ is *dictated* by x_i (the ith bit of x). In other words, $f(x) = \ell_{e_i}(x)$ where

$$e_i = \left(\underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{N-i} \right).$$

Remark. The function $f(x) = 1 - \ell_{e_i}(x)$ is also a dictatorship (also known as an anti-dictatorship) because x_i dictates the value of $f(x)$. However, in what follows we will only need the dictatorship functions where $f(x)$ agrees with x_i.

We now define the code that will eventually be used for encoding assignments to a constraint graph instance of Gap-Label-Cover$\big(d = 5^k, M = 7^k, N = 2^k, s = 1 - \varepsilon\big)$. The code was introduced by Bellare, Goldreich and Sudan [8].

Definition 13.3 (Long code)**.** The *long code* with N messages is the mapping

$$\mathcal{D}_N : [N] \to \{0,1\}^{2^N}$$

which sends $i \in [N]$ to the truth-table of the dictatorship ℓ_{e_i}. Formally, using H_N to denote the N-dimensional Hadamard code (Definition 2.5),

$$\mathcal{D}_N(i) = H_N(e_i) = (\ell_{e_i}(x))_{x \in F_2^N}$$

The 'long code' is indeed 'long', because it sends a message of length $k = \lceil \log(N) \rceil$ (encoding $i \in [N]$) to a codeword of length $2^N = 2^{2^k}$.

13.3 Testing the long code with noisy queries

We now define a three-query linear test for the long-code and analyze it using Fourier representations. By 'linear test' we mean a test that makes three queries and accepts iff their xor is zero. We start with some intuition.

Suppose we wish to test if $f : \{0,1\}^N \to \{0,1\}$ is (close to) a word of the long-code, and have at our disposal a distribution over (three-query) linear tests.

The codewords of the long-code $\mathcal{D}_N$ correspond to a subset of the Hadamard code H_N. However, the long-code is not linear, because although both ℓ_{e_i}, ℓ_{e_j} are codewords, this is not true of $\ell_{e_i \oplus e_j}$. If we expect *perfect completeness* from our test, we're bound to get into trouble. The set of words passing a set of linear tests with perfect completeness must be a linear space. But as we just said, the long-code is not a linear code. Thus, we will need to settle for a (linear) test that has *imperfect* completeness.

Suppose $f \equiv \ell_i$ is a word of the long-code. Clearly, f passes the BLR linearity test with perfect completeness, but so does any linear function. We need a test that will distinguish f from some ℓ_α where α has large support.

The solution is to *perturb* one of the three BLR-queries by flipping each bit independently with probability 1/100. I.e. instead of querying $f(x \oplus y)$ (in addition to querying $f(x), f(y)$ as in the standard BLR test), we ask for $f(x \oplus y \oplus \mu)$ where $\mu \in \{0,1\}^n$ is 'noise', to be thought of as random and having small support. Notice that as long as $\mu_i = 0$ (which happens with probability 0.99) we will still have $f(x) \oplus f(y) = f(x \oplus y \oplus \mu)$. Now consider ℓ_α where α has large support. Notice ℓ_α is 'more likely' to be hit by μ in which case the perturbed BLR test will reject. Details follow.

Definition 13.4. [Noise] For noise parameter $\rho \in [0,1]$ let $\mathcal{N}_\rho^N$ be the following distribution over $\{0,1\}^N$. To select $\mu \in \{0,1\}^N$ according to $\mathcal{N}_\rho^N$, set

$$\mu_i = \begin{cases} 0 & \text{w.p. } 1-\rho \\ 1 & \text{w.p. } \rho \end{cases}$$

for each $i \in [N]$ independently at random.

The following long-code test is a modification on the BLR linearity test, and was introduced by [30].

Definition 13.5 (Håstad's long-code test)**.** For noise parameter $\rho \in [0,1]$ and oracle access to $f : \{0,1\}^N \to \{0,1\}$ the long-code tester $T^f_{\mathcal{D}_N,\rho}$ proceeds as follows.

- Select $x, y \in \{0,1\}^N$ uniformly and independently at random.
- Select $\mu \in \{0,1\}^N$ according to distribution $\mathcal{N}^N_\rho$.
- Accept iff $f(x) \oplus f(y) = f(x \oplus y \oplus \mu)$.

Notice the long-code test is a three-query, linear test. To analyze its completeness and soundness we need the following definition of an 'almost-dictatorship'.

Definition 13.6 (Juntas)**.** Let $\delta \in [0,1]$ and $h \in \mathbb{N}^+$. We say the Boolean function $f : \{0,1\}^N \to \{0,1\}$ is a (δ, h)-*junta* if at least one $\alpha \in \{0,1\}^N$ with $|\alpha| \leq h$ satisfies $\hat{f}_\alpha \geq \delta$ where $\hat{f}_\alpha$ is the α-coefficient of the Fourier representation of the function $\tilde{f}(x) \stackrel{\text{def}}{=} (-1)^{f(x)}$.

Indeed, a Junta is 'almost' a dictatorship because it can readily verified that a dictatorship is a $(1,1)$-Junta. We are ready to analyze Håstad's long-code test.

Proposition 13.7. *For all $N \in \mathbb{N}^+$, every function $f : \{0,1\}^N \to \{0,1\}$ and every $\rho, \eta > 0$, Håstad's (three-query, linear) long-code test satisfies*

- **Completeness:** *If f is a dictatorship,* $\mathbf{Pr}\left[T^f_{\mathcal{D}_N,\rho} = \mathsf{accept}\right] \geq 1 - \rho$.
- **Soundness:** *If* $\mathbf{Pr}\left[T^f_{\mathcal{D}_N,\rho} = \mathsf{accept}\right] \geq 1/2 + \eta$, *then f is a* $\left(2\eta, \frac{1}{\rho} \cdot \ln\left(\frac{1}{\eta}\right)\right)$*-junta.*

The probability in both cases above is over the coin tosses of the long-code tester.

Proof. As usual, completeness is easy. If $f = \ell_{e_i}$ is a dictatorship, the acceptance probability is the probability of not flipping the dictator bit, i.e. $\mathbf{Pr}[\mu_i = 0] = 1 - \rho$.
Regarding soundness, we use Fourier analysis to show that large acceptance probability implies f is a junta. We follow the method of proof of Theorem 12.6. In particular, we look at $\tilde{f}(x) = (-1)^{f(x)}$ and use $\hat{f}_\alpha$ to denote the α-Fourier coefficient of $\tilde{f}$.

$$\begin{aligned}
\mathbf{Pr}[\mathsf{accept}] &= \mathop{\mathbf{Ex}}_{x,y,\mu}\left[\frac{1}{2}\Big(1 + \tilde{f}(x) \cdot \tilde{f}(y)) \cdot \tilde{f}(x \oplus y \oplus \mu)\Big)\right] \\
&= \frac{1}{2} + \frac{1}{2}\mathop{\mathbf{Ex}}_{x,y,\mu}\left[\tilde{f}(x) \cdot \tilde{f}(y)) \cdot \tilde{f}(x \oplus y \oplus \mu)\right] \\
&= \frac{1}{2} + \frac{1}{2}\mathop{\mathbf{Ex}}_{x,y,\mu}\left[\left(\sum_\alpha \hat{f}_\alpha \chi_\alpha(x)\right)\left(\sum_\beta \hat{f}_\beta \chi_\beta(y)\right)\left(\sum_\gamma \hat{f}_\gamma \chi_\gamma(x \oplus y \oplus \mu)\right)\right] \\
&= \frac{1}{2} + \frac{1}{2}\mathop{\mathbf{Ex}}_{x,y,\mu}\left[\sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \cdot \chi_{\alpha\oplus\gamma}(x) \cdot \chi_{\beta\oplus\gamma}(y) \cdot \chi_\gamma(\mu)\right]
\end{aligned}$$

(by linearity of expectation and independence of x, y, μ)

$$= \frac{1}{2} + \frac{1}{2} \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \cdot \mathop{\mathbf{Ex}}_{x}\left[\chi_{\alpha\oplus\gamma}(x)\right] \cdot \mathop{\mathbf{Ex}}_{y}\left[\chi_{\beta\oplus\gamma}(y)\right] \cdot \mathop{\mathbf{Ex}}_{\mu}\left[\chi_\gamma(\mu)\right]$$

(since $\forall \varsigma \neq \bar{0}\ \left[\mathbf{Ex}\left[\chi_\varsigma\right] = 0\right]$, and $\mathbf{Ex}[\chi_{\bar{0}}] = 1$)

$$= \frac{1}{2} + \frac{1}{2} \sum_{\gamma} \hat{f}_\gamma^3 \cdot 1 \cdot 1 \cdot \mathop{\mathbf{Ex}}_{\mu}\left[\chi_\gamma(\mu)\right]$$

(by independence of the μ_i's)

$$= \frac{1}{2} + \frac{1}{2} \sum_{\gamma} \hat{f}_\gamma^3 \cdot \prod_i \mathop{\mathbf{Ex}}_{\mu_i}[(-1)^{\gamma_i \mu_i}]$$

(since $\gamma_i = 0 \Rightarrow \mathbf{Ex}_{\mu_i}[(-1)^{\gamma_i\mu_i}] = 1$ and otherwise the value is 1 w.p. $1-\rho$ and -1 w.p. ρ)

$$= \frac{1}{2} + \frac{1}{2} \sum_{\gamma} \hat{f}_\gamma^3 \cdot (1-2\rho)^{|\gamma|}$$

now, under our assumption

$$\frac{1}{2} + \eta \leq \frac{1}{2} + \frac{1}{2} \sum_{\gamma} \hat{f}_\gamma^3 \cdot (1-2\rho)^{|\gamma|}$$

$$2\eta \leq \sum_{\gamma} \hat{f}_\gamma^3 \cdot (1-2\rho)^{|\gamma|} = \sum_{\gamma} \hat{f}_\gamma^2 \cdot \hat{f}_\gamma \cdot (1-2\rho)^{|\gamma|}$$

Parseval's Equality (10) implies $\left\{ \hat{f}_\gamma^2 \,\middle|\, \gamma \in \{0,1\}^N \right\}$ defines a distribution over $\{0,1\}^N$, so Markov's inequality assures us that there must exist some γ_0 with $2\eta \leq \hat{f}_{\gamma_0} \cdot (1-2\rho)^{|\gamma_0|}$. Since $(1-2\rho), \hat{f}_{\gamma_0} \leq 1$ we have $\hat{f}_{\gamma_0} \geq 2\eta$ and $(1-2\rho)^{|\gamma_0|} \geq 2\eta$. Now, since $1-x \leq \exp(-x)$, we may conclude that

$$\exp(-2\rho|\gamma_0|) \geq 2\eta$$

$$-2\rho|\gamma_0| \geq \ln(2\eta)$$

$$|\gamma_0| \leq \frac{1}{\rho} \cdot \ln\left(\frac{1}{\eta}\right) - \ln(2) = \mathcal{O}\left(\frac{1}{\rho} \cdot \ln\left(\frac{1}{\eta}\right)\right)$$

as required. □

13.4 Completing the proof of Theorem 11.1

Let us restate the Theorem we're going to prove.

Theorem 13.8 (Theorem 11.1)**.** *For all constant $\varepsilon, \eta > 0$,*

$$\mathbf{NP} \subseteq \mathbf{PCP}\left(\begin{array}{ll} \mathsf{length} & = n^{\mathcal{O}(1)} \\ \mathsf{randomness} & = \mathcal{O}(\log(n)) \\ \mathsf{query} & = 3 \text{ over alphabet } \{0,1\} \\ \mathsf{time} & = n^{\mathcal{O}(1)} \\ \mathsf{completeness} & = 1-\varepsilon \\ \mathsf{soundness} & = 1/2-\eta \end{array}\right),$$

Moreover, the verifier is non-adaptive and accepts the answers to the three queries (denoted x_1, x_2, x_3) based on a linear constraint of the form '$x_1+x_2+x_3 = 1 \mod 2$' or '$x_1+x_2+x_3 = 0 \mod 2$'.

Proof. Let $\psi, |\psi| = n$ be an instance of an **NP**-languange L. Our verifier starts by reducing ψ to an instance of Gap-Label-Cover$\big(d = 5^k, M = 7^k, N = 2^k, s = 1-\rho\big)$, where $\rho = \rho(\varepsilon, \eta)$ will be defined later. Notice the reduction runs in polynomial time, because η, ε are fixed constants. Let

$$\mathcal{G} = (G, \Sigma, \mathcal{C} = \{C_e \mid e \in E\})$$

be the constraint graph as in Definition 11.9 that ψ is reduced to. Recall $\Sigma = \Sigma_W \cup \Sigma_V$ where $|\Sigma_W| = M, |\Sigma_V| = N$, so wlog set $\Sigma_W = [M], \Sigma_V = [N]$. Recall an assignment to $\mathcal{G}$ labels each left vertex by $i \in [M]$ and each right vertex by $j \in [N]$.

Projection property Let us examine a useful property of our edge constraints. Let (w, v) be an edge. Recall (from the proof of Theorem 11.13) that the left-vertex w corresponds to k clauses $\phi_1, \ldots, \phi_k$ from a 3-CNF and the right-vertex v corresponds to a subset of k variables $z_1, \ldots, z_k$ of $\phi_1, \ldots, \phi_k$ (one variable per clause). Recall an assignment to w gives an assignment to all literals in $\phi_1, \ldots, \phi_k$ that satisfies these clauses. Furthermore, once this assignment is given, there is one and only one assignment to v that will satisfy the edge constraint over (w, v), namely the *projection* of the assignment to $\phi_1, \ldots, \phi_k$ to variables $z_1, \ldots, z_k$. Thus, we can view the constraint $C_{(w,v)}$ as a mapping

$$\pi_{(w,v)} : [M] \to [N], \tag{12}$$

where for $i \in [M]$, we define $\pi_{(w,v)}(i)$ to be the unique $j \in [N]$ such that

$$C_{(w,v)}(i, \pi_{(w,v)}(i)) = \mathsf{accept}.$$

Proof Oracle As a proof, the verifier expects to see for every $w \in W$ a word of 2^M bits, which is supposed to be the long-code encoding of the assignment to w. Similarly, for each $v \in V$ verifier expects a word of 2^N bits (the supposed long-code encoding of the assignment to v). Formally, the proof oracle is collection of purported long-code words, one per vertex,

$$\Pi = \Big\{ g_w : \{0,1\}^M \to \{0,1\} \;\Big|\; w \in W \Big\} \cup \Big\{ f_v : \{0,1\}^N \to \{0,1\} \;\Big|\; v \in V \Big\}.$$

Notice the size of our proof oracle is polynomial in $|\mathcal{G}|$ which is polynomial in the size of the original input instance ψ.

Verifier operation First, verifier selects a uniformly random edge $(w, v) \in G$. Let $g = g_w$ and $f = f_v$. Let $\pi = \pi_{(w,v)}$ be the constraint associated with this edge as per Equation (12). For $x \in \{0,1\}^N$ let $S_x \subseteq [N]$ be the set whose indicating vector is x, i.e. $S = \{j \in [N] \mid x_j = 1\}$. Let $\pi^{-1}(S_x) = \{i \in [M] \mid \pi(i) \in S_x\}$ be the pre-image of S_x under π. Finally, let $\pi^{-1}(x) \in \{0,1\}^M$ be indicating vector of the set $\pi^{-1}(S_x)$ defined by $\left(\pi^{-1}(x)\right)_i = x_{\pi(i)}$ for $i \in [M]$. Verifier runs the following sub-test, denoted $T^{f,g}_{\text{Håstad}}[\pi, M, N, \varepsilon]$.

1. Draw $x \in \{0,1\}^N$, $y \in \{0,1\}^M$ independently and uniformly.
2. Draw $\mu \in \{0,1\}^M$ according to noisy distribution $\mathcal{N}^{\varepsilon}_M$.
3. Let $z = y \oplus \pi^{-1}(x) \oplus \mu$.
4. If $f(x) \oplus g(y) \oplus g(z) = 0$, accept; otherwise reject.

Basic properties Notice the test above is linear, makes three queries over the binary alphabet and requires $\mathcal{O}(n)$ random bits and $n^{\mathcal{O}(1)}$ running time. We only need to verify completeness (easy) and soundness (less easy).

Completeness Suppose $\psi \in L$. By Theorem 11.13 there exists an assignment

$$A_W : W \to [M], A_V : V \to [N]$$

satisfying all constraints, i.e.

$$\pi_{(w,v)}(A_W(w)) = A_V(v) \text{ for all edges } (w, v) \in E.$$

As a proof oracle, take the long-code encoding of such an assignment. Formally,

$$\Pi = \{g_w = \mathcal{D}_M(A_W(w)) \mid w \in W\} \cup \{f_v = \mathcal{D}_N(A_V(v)) \mid v \in V\}.$$

Consider the success probability of the tester above when applied to random edge (w, v), where $A_W(w) = i$, $A_V(v) = j$. By construction $f = f_v = \ell_{e_j}$ and $g = g_w = \ell_{e_i}$ so $f(x) = x_j, g(y) = y_i$ and $g(z) = y_i \oplus \left(\pi^{-1}(x)\right)_i \oplus \mu_i$. The crucial observation is that $\left(\pi^{-1}(x)\right)_i = x_j$ because $\pi(i) = j$. We conclude

$$\mathbf{Pr}[f(x) \oplus g(y) \oplus g(z) = 0] = \mathbf{Pr}[x_j \oplus y_i \oplus (y_i \oplus x_j \oplus \mu_i) = 0] = \mathbf{Pr}[\mu_i = 0] = 1 - \varepsilon.$$

Notice the acceptance probability above applies regardless of our selection of the edge (w, v), by the perfect completeness of Theorem 11.13. This proves completeness $\geq 1 - \varepsilon$ as claimed.

Soundness Suppose $\psi \notin L$. Theorem 11.13 implies any assignment cannot satisfy more than a ρ-fraction of edge constraints. The main claim used in analyzing the soundness is Proposition 13.9 below. It's proof uses Fourier analysis techniques discussed in the past two lectures and is part of Homework assignment 4.

Proposition 13.9. *Let $f, g, \pi, \vec{\pi}$ be as defined above. Let $\hat{f}_\beta, \hat{g}_\alpha$ denote respectively the α, β Fourier coefficient of the functions $\tilde{f}(x) \stackrel{\text{def}}{=} (-1)^{f(x)}$ and $\tilde{g}(x) \stackrel{\text{def}}{=} (-1)^{g(x)}$. For every $\epsilon, \eta > 0$,*

$$\text{If } \mathbf{Pr}\left[T^{f,g}_{\text{Håstad}}[\pi, M, N, \varepsilon] = \mathsf{accept}\right] \geq \frac{1+\eta}{2}, \text{ Then } \sum_{\substack{\alpha\in\{0,1\}^M,\ \beta\in\{0,1\}^N \\ S_\beta\subseteq\pi(S_\alpha) \\ |\alpha|,|\beta|\leq\mathcal{O}\left(\frac{1}{\varepsilon}\cdot\ln\left(\frac{1}{\eta}\right)\right)}} \hat{g}_\alpha^2 \hat{f}_\beta^2 \geq \eta^2.$$

where S_β, S_α are the respective subsets of $[N], [M]$ whose indicator vectors are β, α, and $\pi(S_\alpha) = \{\pi(i) \mid i \in S_\alpha\}$.

Let us 'decode' an assignment A for $\mathcal{G}$ from any proof Π. Later we will claim that high acceptance probability of Π implies high acceptance probability of A (contradiction).
Repeat the following 'decoding' process for each $w \in W$, where $g = g_w$ is the purported long-code of the assignment to w. Parseval's Equality (10) says $\sum_\alpha \hat{g}_\alpha^2 = 1$, implying $\left\{\hat{g}_\alpha^2 \,\middle|\, \alpha \in \{0,1\}^M\right\}$ is a distribution over $\{0,1\}^M$. Sample a random $\alpha \in \{0,1\}^M$ according to this distribution. Let $S_\alpha \subseteq [M]$ be the set whose indicator vector is α. Sample a random $i \in S_\alpha$ and set $A_W(w) = i$.
Similarly, repeat the same process for every $v \in V$, with $f = f_v$ the purported long-code of the assignment to v. Namely, pick random $S_\beta \subset [N]$ according to the distribution $\left\{\hat{f}_\beta \,\middle|\, \beta \in \{0,1\}^N\right\}$, and set $A_V(v) = j$ for a randomly chosen $j \in S_\beta$. Notice we have obtained an assignment to $\mathcal{G}$.
Assume, by way of contradiction,

$$\mathbf{Pr}\left[V^\Pi[\psi] = \mathsf{accept}\right] \geq 1/2 + \eta \tag{13}$$

where V denotes our PCP verifier and the probability is over her random coin-tosses. Our main claim is

$$\mathbf{Pr}_{(w,v)\in E(G)}\left[\pi_{(w,v)}(A_W(w) = A_V(v))\right] > \rho \tag{14}$$

where A is the assignment obtained from Π be the 'decoding' process. Notice A satisfies more than a ρ-fraction of constraints, contradiction. Thus, from here on we focus on proving Inequality (13) implies Inequality (14).
Consider a verifier for $\mathcal{G}$ which selects a random edge and reads the assignment to both vertices. Inequality (13) implies that with probability at least $\eta/2$, the verifier V selects a 'good edge' (w, v) such that the 'If' (left hand) side of Proposition 13.9 holds, so (by this Proposition) the 'Then' (right hand) side also holds. Viewing Fourier coefficients as defining a distribution, the right hand side of Proposition 13.9 says that with probability at least η^2, the decoding process has chosen[9] 'good α and β' such that $S_\beta \subseteq \pi(S_\alpha)$ and $|\alpha|, |\beta| = \mathcal{O}\left(\frac{1}{\varepsilon} \cdot \ln\left(\frac{1}{\eta}\right)\right)$. If this holds, then with probability at least $(|\alpha||\beta|)^{-1}$ the decoding process has also chosen 'good assignments' $A_W(w) = i$ and $A_V(v) = j$ such that $\pi(i) = j$, in

[9] We stress the decoding process does not depend on the edge (w, v) or on the queries of the verifier V. Rather, a value is assigned to w and v beforehand.

which case the constraint $C_{(w,v)}$ is satisfied. The combined probability for all of the above occurring is at least:

$$\left(\underbrace{\eta/2}_{\text{good edge}} \cdot \underbrace{\eta^2}_{\text{good } \alpha \text{ and } \beta} \right) \cdot \underbrace{c}_{\text{the } \mathcal{O}(\cdot) \text{ constant}} \cdot \underbrace{\frac{\varepsilon^2}{\ln(1/\eta)^2}}_{\text{this is } (|\alpha||\beta|)^{-1}} = \underbrace{\kappa}_{\text{some constant depending only on } \varepsilon, \eta}$$

We conclude the existence of an assignment for $\mathcal{G}$ satisfying a κ-fraction of constraints. Thus, setting $\rho < \kappa = \kappa(\varepsilon, \eta)$ in Theorem 11.13, we show our assumption (Inequality (13)) implies a contradiction. We conclude

$$\forall \Pi, \mathbf{Pr}\left[V^{\Pi}[\psi] = \mathsf{accept}\right] \leq 1/2 + \eta.$$

This completes our proof of Theorem 11.1, modulo the proof of Proposition 13.9 which is part of Homework assignment 4. □

References

[1] Sanjeev Arora. *Probabilistic checking of proofs and the hardness of approximation problems*. PhD thesis, UC Berkeley, 1994.

[2] Sanjeev Arora and Carsten Lund. Hardness of approximations. Technical Report TR-504-95, Princeton University, Computer Science Department, December 1995.

[3] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[4] Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[5] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, New York, NY, USA, 1991. ACM Press.

[6] Laszlo Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. Technical report, University of Chicago, Chicago, IL, USA, 1990.

[7] M. Bellare, D. Coppersmith, J. Hastad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 432, Washington, DC, USA, 1995. IEEE Computer Society.

[8] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability — towards tight results. *SIAM Journal on Computing*, 27(3):804–915, June 1998.

[9] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing (STOC-04)*, pages 1–10, New York, June 13–15 2004. ACM Press.

[10] Eli Ben-Sasson, Oded Goldreich, and Madhu Sudan. Bounds on 2-query codeword testing. In *RANDOM-APPROX 2003*, pages 216–227, 2003.

[11] Eli Ben-Sasson and Madhu Sudan. Short PCPs with poly-log rate and query complexity. In *STOC*, pages 266–275, 2005.

[12] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short pcps via epsilon-biased sets. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 612–621, New York, NY, USA, 2003. ACM Press.

[13] Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *STOC*, pages 612–621, 2003.

[14] Andrej Bogdanov. Gap amplification fails below 1/2. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2005.

[15] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, STOC'71 (Shaker Heights, OH, May 3-5, 1971)*, pages 151–158, New York, 1971. ACM, ACM Press.

[16] David A. Cox, John B. Little, and Don O'Shea. *Ideals, Varieties, and Algorithms*. LNCS. Springer-Verlag, Berlin, 1992.

[17] Dinur. The PCP theorem by gap amplification. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2005.

[18] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP-theorem. In *FOCS*, pages 155–164, 2004.

[19] Uriel Feige. A threshold of $\ln(n)$ for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[20] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[21] Uriel Feige and Joe Kilian. Two-prover protocols — low error at affordable rates. *SIAM J. Comput.*, 30(1):324–346, 2000.

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

[23] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, June 1981.

[24] Oded Goldreich. Probabilistic proof systems—A survey. In *14th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1200 of *lncs*, pages 595–611, Lübeck, Germany, 27 February–March 1 1997. Springer.

[25] Oded Goldreich. A sample of samplers - A computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

[26] Oded Goldreich and Madhu Sudan. Locally testable codes and pcps of almost-linear length. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 13–22, Washington, DC, USA, 2002. IEEE Computer Society.

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

[28] Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A tight characterization of **NP** with 3 query **PCP**s. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS-98)*, pages 8–17, Los Alamitos, CA, 1998. IEEE Computer Society.

[29] Prahladh Harsha and Madhu Sudan. Small PCPs with query complexity. In Afonso Ferreira and Horst Reichel, editors, *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science, STACS'2001 (Dresden, Germany, February 15-17, 2001)*, volume 2010 of *LNCS*, pages 327–338. Springer-Verlag, Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan-Paris-Singapore-Tokyo, 2001.

[30] Johan Håstad. Some optimal inapproximability results. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1997. ACM Press.

[31] Subhash Khot. On the power of unique 2-prover 1-round games. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, New York, NY, USA, 2002. ACM Press.

[32] Subhash Khot. On the unique games conjecture. In *FOCS*, page 3, 2005.

[33] Rudolf Lidl and Harald Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1997.

[34] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[35] G. A. Margulis. Explicit constructions of expanders. *Problemy Peredači Informacii*, 9(4):71–80, 1973.

[36] M.Blum, M.Rubi, and R.Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Journal of Computer and System Sciences*, pages 549–595, 1993.

[37] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.

[38] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[39] Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 194–203, New York, NY, USA, 1994. ACM Press.

[40] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998.

[41] Ran Raz. A parallel repetition thorem. In *SIAM journal of computing*, volume 27, pages 763–803, 1998.

[42] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 3, Washington, DC, USA, 2000. IEEE Computer Society.

[43] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, April 1996.

[44] J. P. Serre. *Linear representations of finite groups*, volume 42 of *Graduate texts in Mathematics*. Springer, 1977.

[45] Amir Shpilka and Avi Wigderson. Derandomizing homomorphism testing in general groups. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 427–435, New York, NY, USA, 2004. ACM Press.

[46] Madhu Sudan. Probabilistically checkable proofs. Lecture notes, July 2000.

[47] Luca Trevisan. Inapproximability of combinatorial optimization problems. *CoRR*, cs.CC/0409043, 2004.