Randomize Methods in Computational Complexity

Summer term 2009 - University of Bonn, Germany

Contents

Chapter 1

Introduction

1.1 Last semester

This course will study some *computational* problems and their complexity. We will assume some basic knowledge of computation: Turing machines, P, NP, EXP, ... The lecture notes of the winter terme 2008/2009 are available at www.klausuren-skripte-protokolle.de. Recommended text-book (online): "Complexity Theory: A Modern Approach" by Arora and Barak.

1.2 Formalizing Problems and Difficulty

We call a problem computable if there is a turing-machine that solves it (in a finite number of steps). Such problems are also called decidable or recursive in literature. Famous examples:

1. Given a quadratic polynomial $f \in \mathbb{Z}[x_1, \dots, x_n]$. The problem of deciding the existance of a $\mathbb{Z}$-root $x \in \mathbb{Z}^n$ with $f(x) = 0$.

2. Given several quadratic polynomials $f_1, \dots, f_m \in \mathbb{Z}[x_1, \dots, x_n]$. Decide whether there is a $\mathbb{Z}$-root $x \in \mathbb{Z}^n$ with $f_1(x) = \dots = f_m(x) = 0$. This Problem is uncomputable and called "Hilbert's 10th problem".

We will formalize a problem L as subset of $\{0,1\}^*$ (which we will call a language) or as a function: $\{0,1\}^* \to \{0,1\}^*$.

The problem of "adding two integeres" can be seen as a function:

$$\begin{array}{rccc} + : & \{0,1\}^* & \to & \{0,1\}^* \\ & (m,n) & \mapsto & (m+n) \end{array}$$

or as a language

$$L_+ := \{(m, n, m+n) \mid m, n \in \mathbb{Z}\}$$

or

$$L'_+ := \{(m, n, i) \mid m, n \in \mathbb{Z}, i\text{th bit of } m+n \text{ is } 1\}$$

We view a decision problem as a language and a functional problem as a funcion $\{0,1\}^* \to \{0,1\}^*$.

Definition 1.2.1. *For a problem L, the step-by-step procedure of it's turing-machine (if one exists) is called an algorithm for L. The number of steps "#steps" of the turing-machine to stop is called time complexity of L. The number of used cells is called space complexity of L. A complexity class is a collection of problems, for e.g.*

1. *Let $T : \mathbb{N} \to \mathbb{N}$. Then*

$$\mathrm{DTime}(T(n)) := \{L \subseteq \{0,1\}^* \mid \exists \text{ turing-machine that can check } x \in L \text{ in time } O(T(|x|))\}$$

2. *The class of polynomial-time problems*

$$P := \bigcup_{c \in \mathbb{N}} \mathrm{DTime}(n^c)$$

3.

$$E := \bigcup_{c \in \mathbb{N}} \mathrm{DTime}(2^{cn})$$

4. *The class of exponential-time problems*

$$EXP := \bigcup_{c \in \mathbb{N}} \mathrm{DTime}(2^{n^c})$$

5. *The class of sub-exponential-time problems*

$$SUBEXP := \bigcap_{c \in \mathbb{N}} \mathrm{DTime}(2^{n^c})$$

   e.g. $2^{(\log(n))^d}$ for a constant d.

This course will focus on *randomized methods* i.e. the algorithms are allowed to "flip coins and proceed". The only condition being that the output should be correct with a guaranteed probability.

Definition 1.2.2.

$$BPP := \{L \subseteq \{0,1\}^* \mid \exists \text{ a polynomial-time randomized turing-machine that solves } L \text{ with high probability } \}$$

TODO: check spelling

Remark 1.2.3. *The following statements are obvious:*

- *$P \subseteq BPP$ (Open: $P = BPP$).*
- *$P \subsetneq E \subsetneq EXP$*
- *$P \subsetneq SUBEXP \subsetneq E \subsetneq EXP$*

Definition 1.2.4. *For a non-deterministic turing-machine we define* $\mathrm{Ntime}(T(n)), NE, NEXP, SUBN$ *analogous as in 1. It holds that $NP \subsetneq SUBNEXP \subsetneq NE \subsetneq NEXP$.*

There are many open questions:

- $P = BPP$

- $P = NP$
- $BPP = NP$

So it seems that changing the resource leads to open questions. We will focus on $P = BPP$.

Finally we define space complexity and space complexity classes analogous:

Definition 1.2.5. *Let $S : \mathbb{N} \to \mathbb{N}$:*

$$\text{Space} := \{L \subseteq \{0,1\}^* \mid \exists \text{ a } O(S(n))\text{-space turing-machine solving } L\}$$

and define complexiy classes

$$PSPACE := \bigcup_{c\in\mathbb{N}} \text{Space}(n^c)$$

$$EXPSPACE := \bigcup_{c\in\mathbb{N}} \text{Space}(2^{n^c})$$

Practical efficiency corresponds to P and BPP. For example it is an open question if $IntegerFactoring \in BPP$.

Definition 1.2.6. *An arithmetic circuit is a rooted tree with inputs as leaves; $+$, $*$, & as internal nodes and constants (from $\mathbb{F}$) at internal edges. The leafes correspond the x_is.*

A classical example in BPP not kwnown to be in P:

Definition 1.2.7. *Polynomial-Identity-Testing (PIT): Given an arithmetic circuit $C \in \mathbb{F}[x_1, \ldots, x_n]$. Test if $C = 0$. The corresponding language is $PIT := \{C \mid C = 0\}$.*

Theorem 1.2.8.

$$PIT \in BPP$$

Proof. Let $C(x_1, \ldots, x_n)$ be the given circuit with constants from a field $\mathbb{F}$. Let s be the size of the circuit description $|C|$. The total degree d of the output polynomial of C is $\le 2^s$. Now we unterscheiden two cases: **TODO:** translate

Case $|\mathbb{F}| > 2^{s+1}$: In this case the algorithm is simple: Pick a random $a \in \mathbb{F}^n$. If $C(a) = 0$ then output 0 else 1.

The algorithm can be implemented to take $p(s)$-many $\mathbb{F}$-operatations (where p is a polynomial). THe correctness of the algorithm: If $C = 0$ then

$$\text{Prob}_{a_1,\ldots,a_n\in\mathbb{F}}[\text{answer is correct}] = 1$$

but if $C \ne 0$ then

$$p := \text{Prob}_{a_1,\ldots,a_n\in\mathbb{F}}[\text{answer is correct}] < 1$$

□

Lemma 1.2.9.

$$p > \left(1 - \frac{d}{|\mathbb{F}|}\right)$$

Chapter 2

Circuits

2.1 Polynomial-Identity-Testing is important

Theorem 2.1.1.

$$PIT \in P \Rightarrow (NEXP \nsubseteq P/Poly) \vee (per \notin AlgP/Poly)$$

Lemma 2.1.2.

$$(PIT \in P) \;\&\; (per \in AlgP/Poly) \Rightarrow P^{per} \subseteq NP$$

Proof. □ **TODO:** copy lecture notes

Idea for 2.1.1: If we also assume that $NEXP \subseteq P/Poly$ then we will be able to show $NEXP \subseteq P^{per}$.

First we recall certain notations from the last course:

Definition 2.1.3.

- *$\Sigma_2^P = NP^{NP}$ or equivalently $L \in \Sigma_2^P$ iff $\exists$ polynomial-time turing-machine N s.t. $\forall x \in \{0,1\}^*$, $x \in L$ iff $\exists y_1 \forall y_2 \, [N(x, y_1, y_2) = 1]$.*
- *Similarly $\Sigma_3^P, \Sigma_4^P, \ldots$ and finally $PH = \bigcup_{i\geq 1} \Sigma_i^P$.*
- *$\Pi_1^P = co - \Sigma_1^P = coNP$, $\Pi_2^P = co - \Sigma_2^P = co - (NP^{NP})$ etc.*

Instead of $\exists$ and $\forall$ we can also use $\exists$ and M (most) quantifiers, where "$M_{y\in\{0,1\}^l}[P(y) = 1]$" means that for $\frac{2}{3}$ of the 2^l strings y it holds that $P(y) = 1$.

Definition 2.1.4. *We say $\underline{L \in AM[k]}$ if $\exists$ polynomial-time turing-machine N s.t. $\forall x \in \{0,1\}^*$ it holds that $x \in L$ iff $M_{y_1} \exists y_2 \ldots Q y_k \, [N(x, y_1, \ldots, y_k) = 1]$ where Q is either $\exists$ or M, depending only on the parity of k (we want them to alternate: $M\exists M\exists M\exists$). The sizes of all quantified variables is polynomial bounded. $AM[k]$ is analogous to Π_k^P.*

We say $L \in MA[k]$ if $\exists$ polynomial-time turing-machine N s.t. $\forall x \in \{0,1\}^*$ it holds that $x \in L$ iff $\exists y_1 M_{y_2} \ldots Q y_k [N(x, y_1, \ldots, y_k) = 1]$. $MA[k]$ is analogous to Σ_k^P.

We write $AM[2]$ and $MA[2]$ as AM and MA (which are not the union of all $AM[k]$).

An interpretation of $AM[k]$ and $MA[k]$: A is Arthur (randomized polynomial-time machine) and M is Merlin (all powerful machine).

We say $L \in AM[k]$ if for a given input string $x \in \{0,1\}^*$ Merlin can convince Arthur that $x \in L$ using k interactions.

$$\begin{tikzcd}
\text{Merlin} & \text{Arthur} \\
{} & x \arrow[l, "{(x,y_1)}"'] \\
{} \arrow[r, "y_2"] & {} \\
{} & {} \arrow[l, "y_2"'] \\
{} \arrow[r, "y_3"] & {} \\
{} & {} \arrow[l, "y_4"'] \\
\cdots & \cdots \\
{} \arrow[r, "y_k"] & {}
\end{tikzcd}$$

Arthur accepts x iff $N(x, y_1, \dots, y_k) = 1$

Definition 2.1.5.

$$IP := \bigcup_{c \in \mathbb{N}} AM[n^c]$$

Exercise 2.1.6. $IP \subseteq PSPACE$

Theorem 2.1.7.

$$PSPACE = IP$$

Proof. for $PSPACE \subseteq IP$ [Shamir '90]

We will give an interactive protocol for

$$TQBF := \left\{ QBF\phi := \exists x_1, \dots \forall x_n \tilde{\phi}(x_1, \dots, x_n) \mid \phi \text{ is true} \right\}$$

which is a $PSPACE$-complete problem. Arithmetize

$$\phi := Q_1 x_1 \dots Q_n x_n \tilde{\phi}(x_1, \dots, x_n)$$

where $Q_i \in \{\forall, \exists\}$.

1. arithmetize $\tilde{\phi}(x_1, \dots, x_n)$ and get $P_{\tilde{\phi}} \in \mathbb{Z}[x_1, \dots, x_n]$ by
   - $\bar{x_1} \mapsto (1 - x_1)$
   - $x_1 \vee x_2 \mapsto 1 - (1 - x_1)(1 - x_2)$
   - $x_1 \wedge x_2 \mapsto x_1 \cdot x_2$

   so we convert boolean formulas to elements of $\mathbb{Z}[x_1, \dots, x_n]$.

2. arithmetize quantifiers:

$$\tilde{Q}_i = \begin{cases} \Sigma & \text{if } Q_i = \exists \\ \Pi & \text{if } Q_i = \forall \end{cases}$$

3. arithmetize ϕ as:

$$\tilde{Q}_1 x_1 \in \{0,1\}\, \tilde{Q}_2 x_2 \in \{0,1\} \dots \tilde{Q}_n x_n \in \{0,1\} \left[P_{\tilde{\phi}}(x_1, \dots, x_n) \right] \in \mathbb{Z}$$

Exercise 2.1.8. *This value is > 0 iff ϕ is true.*

Now Merlin tries to convince Arthur that $\tilde{Q}_1 \dots \tilde{Q}_n P_{\tilde{\phi}}(x_1, \dots, x_n) = K$ (where $K \in \mathbb{Z}^{>0}$). The protocol is as follows:

1. $\boxed{\text{Arthur}}$ If $n = 1$: Arthur checks himself whether $\tilde{Q}_1 x_1 \in \{0,1\}\, P_{\tilde{\phi}}(x_1) = K$ and accepts ϕ iff it is true.

2. $\boxed{\text{Merlin}}$ sends $s(x_1) \in \mathbb{Z}[x_1]$ to Arthur, as a candidate for

$$\tilde{Q}_2 x_2 \in \{0,1\} \dots \tilde{Q}_n x_n \in \{0,1\}\, P_{\tilde{\phi}}(x_1, \dots, x_n)$$

3. $\boxed{\text{Arthur}}$ Picks a random $a \in \mathbb{Z}$ and recursively verify whether

$$\tilde{Q}_2 \dots \tilde{Q}_n P_{\tilde{\phi}}(a, x_2, \dots, x_n) = s(a)$$

Exercise 2.1.9. *If $\phi = T$ then Merlin can easily convince Arthur. If $\phi = F$ then no matter what Merlin does Arthur will detect an error with high probability.*

□

Now we are ready to state a second lemma:

Lemma 2.1.10.
$$EXP \subseteq P/Poly \Rightarrow EXP = MA$$

Lemma 2.1.11.

$$NEXP \subseteq P/Poly \Rightarrow NEXP = EXP$$

Proof. for 2.1.1. For contradiction sake, assume: $PIT \in P$

1. $per \in AlgP/Poly$

2. $NEXP \subseteq P/Poly$

By assumption 2 and 2.1.10 and 2.1.11: $NEXP = MA$. Recall Toda's theorem $PH \subseteq P^{per}$ which implies that $NEXP \subseteq P^{per}$.

By assumption 1 and 2.1.2: $P^{per} \subseteq NP$. But $NEXP \subseteq NP$ contradicts the nondeterministic time hierarchy.

□

Lemma 2.1.12.

$$EXP \subseteq P/poly \Rightarrow EXP = MA$$

Proof. Recall the classical Karp-Lippton-Theorem:

$$NP \subseteq P/poly \Rightarrow \Sigma_2^P = \Pi_2^P$$

Similar ideas will be used to show that $EXP \subseteq P/poly \Rightarrow EXP = \Sigma_2^P$: Let $L \in EXP$ solved by a turing-machine N. For $x \in L$ the j-th bit in the i-th configuration of $N(x)$ can be found in exponential time. Thus $EXP \subseteq P/poly$ implies that $\exists$ small circuit C s.t. $C(x,i,j)$ is the j-th bit in the i-th step of $N(x)$. Now $x \in L$ implies that $\exists$ circuit C s.t. $\forall$ configurations i and $\forall$ bits j:

$$C(x,i,j) \to C(x,i+1,j) \text{ is a } \underline{\text{valid}} \text{ transition step of } N$$

Exercise 2.1.13. *Complete the description.*

The converse also holds. Hence $L \in \Sigma_2^P$: $EXP \subseteq \Sigma_2^P$. So $EXP = \Sigma_2^P$.

Now we show the lemma itself: We have $\Sigma_2^P \subseteq PSPACE = IP \subseteq EXP$. If we assume $EXP \subseteq P/poly$ then $EXP = IP$. Thus every problem $L \in EXP$ has an interactive protocol, like that for $TQBF$, with the prover Merline being a $PSPACE$ machine. Thus Merlin can be replaced by a polynomial sized circuit family $\{C_n\}_{n\geq 1}$. This suggests the following protocol for any $L \in EXP$: For any $x \in \{0,1\}^n$ we have

1. Merlin: Sends Arthur a circuit C supposed to be C_n
2. Arthur: Runs the interactive protocol on x using C as Merlin and accepts x iff the simulated protocoll accepts.

Exercise 2.1.14. *This means that $L \in MA$.*

□

Philosophically this is an interesting "flip" (the existance of an algorithm implies non-existance of another algorithm).

2.2 (Circuit) Lower Bounds

We believe that $NP \neq P$. One way to prove this could be to prove a stronger result: $NP \nsubseteq P/poly$. This approach was actively tried in the 70s and 80s and lower bounds for special circutits were obtained.

Definition 2.2.1.

$$AC^0 := \{L \subseteq \{0,1\}^* \mid L \textit{ has polynomial sized, constant depth boolean circuits } \}$$

$$AC^i := \{L \subseteq \{0,1\}^* \mid L \textit{ has polynomial sized, } \log^i(n) \textit{ depth boolean circuits } \}$$

$$AC := \bigcup_{i \in \mathbb{N}} AC^i$$

Definition 2.2.2.

$$\begin{aligned} NC^i := \quad & \{L \subseteq \{0,1\}^* \mid L \textit{ has polynomial sized, } \log^i(n) \textit{ depth} \\ & \textit{and bounded fanin boolean circuits } \} \end{aligned}$$

$$NC := \bigcup_{i \in \mathbb{N}} NC^i$$

Exercise 2.2.3. *Show that*

$$NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq AC^1 \subseteq NC^2 \subseteq \ldots$$

Open: $NC \neq P$

Definition 2.2.4. *The problem of parity*

$$\begin{array}{rrcl} \oplus : & \{0,1\}^n & \to & \{0,1\} \\ & (x_1, \ldots, x_n) & \to & \sum_{i=1}^n x_i \pmod 2 \end{array}$$

Remark 2.2.5. *1.* $\oplus \in NC^1$

2. $\oplus$ *can be computed by* $O(1)$*-depth,* $O(2^n)$*-sized boolean circuits.*

3. Can we do better?

Theorem 2.2.6 (Furst, Saxe & Sipser 1981)**.** $\oplus \notin AC^0$

We use "randomized restrictions" i.e. randomly picking input variables and fixing them to a random bit. We show that a depth-d circuit becomes constant if we apply random restriction on $(n - n^\epsilon)$ input variables. On the other hand $\oplus$ does not satisfy this. Now we study the effect of random restrictions on an AC^0-circuit.

Definition 2.2.7. *A k-DNF (k-CNF) is an OR-of-ANDs (AND-of-ORs) where each AND (OR) involves at most k variables.*

Lemma 2.2.8 (Switching)**.** *Suppose f is expressible as a k-DNF, let ρ be a random restriction on $t > \frac{2n}{3}$ input-variables. Then $\forall s \geq 2$:*

$$\Pr_\rho [f|_\rho \textit{ is not expressible as a s-CNF}] \leq \left[\left(1 - \frac{t}{n}\right) 10^k\right]^{\frac{s}{2}}$$

Remark 2.2.9. *It "switches" a k-DNF to a s-CNF and vice-versa.*

Proof. We first show how to use this Lemma to prove the theorem: Starting with an AC^0-circuit $C(x_1, \ldots, x_n)$ and assume w.l.o.g:

1. all fanouts are 1 (i.e. the circuit is a tree)
2. NOT-gates are only at the bottom-most level, thus we can remove them altogether by introducing $\bar{x_1}, \ldots, \bar{x_n}$ as new variables
3. AND and OR gates alternate
4. bottom level has AND gates of fanin 1.

We randomly restrict variables in C in various steps s.t. with high probabilitydepth reduces by 1 in each step. Let n_0 be the number of unfixed variables after step $i4$. We fix $(n_i - \sqrt{n_i})$ variables in the $(i+1)$-th) step So $n_i = n^{\frac{1}{2^i}}$. Let n^b be the number of gates in $C(x_1, \dots x_n)$. Let $k_i := (2^i 10b)$. We begin with a 1-DNF at the bottom. We will show that after the i-th step, with high probability are left with $(a(d-i)$-depth circuit with bottom level fanin at most k. **TODO:** hier fehlt wa

By the Switchung Lemma 2.2.8 each such $\vee$-gate will be espressable as a k_{i+1}-CNF, after the restriction with prob P:

$$\begin{array}{rcl} P & \geq & 1 - \left(\left(1 - \frac{n_i - \sqrt{n_i}}{n_i}\right) 10^{k_i}\right)^{\frac{k_{i+1}}{2}} \\ & = & 1 - \left(\frac{2^{-(i+1)}}{n^{2^{-i}}} 10^{k_i}\right) \\ & = & 1 - \left(\frac{10^{k_i}}{n^{-2^{i+1}}}\right)^{\frac{k_{i+1}}{2}} \\ & \geq & \left(1 - \frac{1}{10n^b}\right) \end{array}$$

Note that $i \leq d = const$. Thus after the $(i+1)$-th step with prob $\geq (1 - \frac{1}{10n^b})$ the last two levels compute a k_{i+1}-CNF, which we can now merge with the & layer just above it, making now the last two layers computing k_{i+1}-CNF and overall $depth = (d-i-1)$.

The second case is where two levels are $\vee$s and & s can be prove just like above using dual of the switching lemma.

After $(d-2)$ steps, we get a k_{d-2}-CNF for k_{d-2}-DNF with prob. $\geq \frac{1}{10n^b} x n^b = \frac{9}{10}$. Thus at the end we get a k_{d-2}-CNF/DNF representation of the parity of $n_{d-2} = n^{2^{-(d-2)}}$ variables. So $\exists$ a fixing of k_{d-2} variables that fixxes parity, which is a contradiction.

□

Exercise 2.2.10.

$\oplus \notin$ polynomial sized $o(\log(\log(n)))$-depth circuits

Proof. [Switching Lemma] Let f be expressible as a k-DNF on n variables. Let $t > \frac{2n}{3}$ and let R_t be the set of all restrictions to t variables. $|R_t| = \binom{n}{t} 2^t$. Let $K_{t,s}$ be those restrictions p for which $f|p$ is not a s-CNF. We will bound $\frac{|K_{t,s}|}{|R_t|}$.

Idea: We give a $1:1$ map: $K_{t,s} \to Z \times S$ where Z is the set of all restrictions on $\geq (t+s)$ variables und S is a set of size 3^{2ks}.

$$\begin{array}{rcl} \frac{|K_{t,s}|}{|R_t|} & \leq & \frac{|Z||s|}{|R_t|} \\ & \leq & \frac{\sum_{t'=t+s}^{n} \binom{n}{t'} s^{t'}}{\binom{n}{t} 2^t} 3^{2ks} \\ & \leq & \frac{n \binom{n}{t+s} 2^{t+s}}{\binom{n}{t} 2^t} 3^{2ks} \\ & < & n 2^s 3^{2ks} \frac{(n-t)(n-t-s+1}{(t+1)\cdots(t+s)} \\ & \leq & n 4^s 3^{2ks} \frac{(n-t)^s}{t^s} < \left(\frac{n-t}{t} 10^k\right)^{\frac{s}{2}} \end{array}$$

f is a disjunctin of several terms, each being & s involving k variables. Define a natural ordering on the variables and the terms (i.e. lexicografically). Now consider a restriction

$r \in K_{t,s}$ Mapping of r into $Z \times S$: Let $term_1$ be the first unfixed term in $f|r$ and let x_1 be the first unfix variable in $term_1$. All terms less then $f|P$ are fixd to false. Let R_1 be the (unique)fixing of the k variables in $term_1$ that makes it true. Note that if both $f|(X_1 = T) \circ r$ and $f|(X_1 = F) \circ r$ are expressible as $(s-1)$-CNF then $f|r$ is expressible as s-CNF.

Thus one of $f|(X_1 = T) \circ r$ and $f|(X_1 = F) \circ r)$ is inexpressible as a $(s-1)$-CNF, call that assignment L_1. Let $term_2$ be the first unfixed term in $f|L_1 \circ r$ and let x_2 be the first unfixed variable in $term_2$. Let R_2 be the fixing of $k$variables in $term_2$ making it true. [terms $< term_2$ are False in $f|L_1 r$]

Let L_2 be the fixing of x_2 s..t. $f|L_2 L_1 r$ is inexpressible as $(s-2)$-CNF. Continuing this way we get $L_1, \dots, L_s, R_1, \dots, R_s$ s.t. we define $f_i := L_i \cdots L_1 r$ and $r_0 := r$ then $\forall 1 \le i \le s$:

1. $term_i$ is the first unfixed term in $f|r_{i-1}$
2. $term_i$ is the first True term in $f|R_i r_{i-1}$
3. L_i and R_i agree with f_{i-1} on all the variables fied by r_{i-1}.

Our next aim is $r \mapsto (R_1 \cdots R_s L_s \cdots L_1 r$. Define $\tau_i := (R_i \cdots R_s r_s)$ with $\tau_{s+1} := r_s$. Notice that $term_i$ is the first True term in $f|\tau_i$. [$\tau_1 := R_1 \cdots R_s L_s \cdots L_1 r \Rightarrow f|\tau_1$ has $term_1$ as its first True term. So on for $i > 1$

Exercise 2.2.11. *Show that you can recover $term_i$ from τ_i.*

Define $z_1, \dots, z_s, w_1, \dots, w_s \in \{0, 1, *\}^k$ (here 0 and 1 mean "fixed" and $*$ means "unfixed") as: z_i describes the action of r_{i-1} on the k variables in $term_i$ and w_i the action of τ_{i+1} on the k variables in $term_i$. Finally we claim that the mapping $r \mapsto (\tau_1, z_1, \dots, z_s, w_1, \dots, w_s)$ is a $1:1$ mapping from $K_{t,s} \to Z \times \{0, 1, *\}^{2ks}$.

Exercise 2.2.12. *Prove this.*

□

We saw that AC^0 cannot compute parity in polynomial-size. What if we add parity-gates to AC^0?

Definition 2.2.13. *For $m_1, \dots, m_k \in \mathbb{Z}$*

$ACC^0[m_1, \dots, m_k] := \{L \subseteq \{0,1\}^* \mid L$ *has polynomial-size, constant depth circuits, using* $\vee$, & $, \neg,\ \pmod{m_1}, \dots,$

where

$$\begin{array}{rrcl} (\mathrm{mod}\ m): & \{0,1\}^l & \to & \{0,1\} \\ & (x_1, \dots, x_l) & \mapsto & \begin{cases} 1 & \textit{if } \sum_{i=1}^l x_i \equiv 0 \pmod{m} \\ 0 & \textit{else} \end{cases} \end{array}$$

and

$$ACC^0 := \bigcup_{k, m_1, \dots, m_k \in \mathbb{N}} ACC^0[m_1, \dots, m_k]$$

Theorem 2.2.14 (by Razborov, Smolensky, 1980s)**.** *For distinct primes p and q:* $\pmod p) \notin ACC^0[q]$.

Lower bounds for ACC^0 inspired the "method of apprixmations" first used by Razborov. This will strengthen $\oplus \notin AC^0$.

We will exhibit the proof for $p = 2, q = 3$.

Lemma 2.2.15. *For a depth d circuit $C \in ACC^0[3]$ on n inputs and size s there is a polynomial in $\mathbb{F}_3[x_1, \dots, x_n]$ of $\deg \leq (2l)^d$ which agrees with $C(x_1, \dots, x_n)$ on $\geq \left(1 - \frac{s}{2^l}\right)$ fraction of the inputs $\bar{x} \in \{0,1\}^n$.*

Proof. Thus if we fix $l = \frac{n^{\frac{d}{2}}}{2}$, we get $\frac{s}{2^l} = 0.01$ which means that $s \geq 0.01 \cdot 2^{\frac{n^{\frac{d}{2}}}{2}}$ if C computes $\pmod 2)$.

Proof of Claim 1: We construct an approximator polynomial in $\mathbb{F}_3[x_1, \dots, x_n]$ for C by induction (on the number of gates): Let g be a node in C at a height (depth) h, then we construct a polynomial $\tilde{g} \in \mathbb{F}_3[x_1, \dots, x_n]$ with $\deg(g) = (2l)^h$ s.t. $\tilde{g}(x_1, \dots, x_n) = g(x_1, \dots, x_n)$ for "most" of the 2^n inputs $(x_1, \dots, x_n) \in \{0,1\}^n$:

1. If g is a variable node x_i then $\tilde{g} = x_i$.

2. If g is a NOT gate, say $g = \neg f$ for some gate f at height $\leq (h-1)$, then by induction we have the polynomial $\tilde{f}$ with $\deg(f) \leq (2l)^{h-1}$: $\tilde{g} := (1 - \tilde{f}$. So $\deg(\tilde{g}) = \deg(\tilde{f}) \leq (2l)^{h-1}$ and its definition introduces no new errors.

3. If g is a $\pmod 3)$ gate, say $g = \pmod 3)(f_1, \dots, f_k)$. Then by induction we have polynomials $\tilde{f}_1, \dots, \tilde{f}_k$ each with $\deg \leq (2l)^{h-1}$. We define $\tilde{g} := (\sum_{i=1}^k \tilde{f}_i)^2$. And we see that $\deg(\tilde{g}) \leq 2 \cdot (2l)^{h-1} \leq (2l)^h$ and we introdued no new error.

4. If g is an OR gate, say $g = \bigvee_{i=1}^k f_i$. $\tilde{g} := 1 - \prod_{i=1}^k (1 - \tilde{f}_i)$ is not feasible because the degree whould explode by a factor of k. But we could only pick a random subset $S \subset [k] := \{1, \dots, k\}$ and define $\tilde{g} = \left(\sum_{i \in S} f_i\right)^2$. For any x we compute

$$p := \text{Prob}_{\emptyset \neq S \subseteq [k]} \left[f_1 \vee \ldots \vee f_k \equiv \sum_{i \in S} f_i \pmod 3 \right] \geq \frac{1}{2}$$

An x for which $f_1(x) = \ldots = f_k(x) = 0$ we know that $p = 1$.

Exercise 2.2.16. *For other xs we show that $p \geq \frac{1}{2}$.*

Hint: Lock at $\sum_{i=1}^k a_i f_i$. Picking a random subset S is equivalent to picking $a_i \in \{0,1\}$ randomly. So we get $\sum_{i=1}^k a_i f_i \equiv 0 \pmod 3$ iff $a_1 \equiv -\sum_{i=2}^k a_i f_i \pmod 3$.

This gives us the idea of picking l random subsets $S_1, \dots, S_l \subseteq [k]$ and define

$$\tilde{g}' := \bigvee_{j=1}^{l} \left(\sum_{i \in S_j} \tilde{f}_i \right)^2$$

We have by the above abservation that: For any x it holds that

$$\text{Prob}_{S_1,\dots,S_l\subseteq[k]}\left[\tilde{g}' \neq \bigvee_{i=1}^{k}\tilde{f}_i\right] \leq \frac{1}{2^l}$$

so $\exists S_1^0,\dots,S_l^0 \subseteq [k]$ with

$$\text{Prob}_{x\in\{0,1\}^n}\left[\tilde{g}' \neq \bigvee_{i=1}^{k}\tilde{f}_i\right] \leq \frac{1}{2^l}$$

Use these $S_1^0,\dots,S_l^0$ to define $\tilde{g}$:

$$\tilde{g} := OR\left(\left(\sum_{i\in S_1^0}\tilde{f}_i\right)^2,\dots,\left(\sum_{i\in S_l^0}\tilde{f}_i\right)^2\right)$$

where $OR(u_1,\dots,u_l) := 1-\prod_{i=1}^{l}(1-u_i)$. $\tilde{g}$ is of deg $\leq (2l)(2l)^{n-1} = (2l)^h$ and we introduce error on $\leq 2^{-l}$ fraction of inputs.

5. If g is an <u>AND</u> gate, say $g = \bigvee_{i=1}^{k} f_i$. By de Morgan's laws: $\neg g = \bigwedge_{i=1}^{k}\neg f_i$. Construct a polynomial for this OR using the technice in the previous case and subtract it from 1 to get $\tilde{g}$.

By applying the above 5 cases we convert a circuit $C(\bar{x})$ to a polynomial in $\mathbb{F}_3[x_1,\dots,x_n]$ of deg $\leq (2l)^d$ which disagrees with C on at mose $\frac{s}{2^l}$ fraction (where s is the size i.e. the number of nodes).

□

By replacing X^2 by X^{p-1} in the previous proof you can prove the lemma for any prime p. But it is an open question if $\pmod 5 \in ACC^0[6]$.

Lemma 2.2.17. *There is no polynomial $f \in \mathbb{F}_3[x_1,\dots,x_n]$ with $\deg(f) \leq \sqrt{n}$ which agrees with C on $\geq 0.99\cdot 2^n$ inputs.*

<u>Proof.</u> Suppose $f \in \mathbb{F}_3[x_1,\dots,x_n]$ agrees with $\pmod 2$ on $G' \subseteq \{0,1\}^n$ and $\deg(f) \leq \sqrt{n}$. Transform f to g as:

$$g(y_1,\dots,y_m) := f(y_1-1,\dots,y_m-1)+1 \pmod 3$$

Thus fs input/ouptut "0/1" will be "+1/−1" of g. $\deg(g) \leq \sqrt{n}$ and it agress with $\prod_{i=1}^{n} y_i$ on $G \subseteq +1,-1^n$ where $G' \to G$ via $0 \mapsto +1$, $1 \mapsto -1$. This already seems strange! We will show that this implies a "small" G. Let F_G be the set of functions $u : G \to \mathbb{F}_3 := \{-1,0,+1\}$.

Exercise 2.2.18. *Any $u \in F_G$ can be realizes as a multilinear polynomial in $\mathbb{F}_3[x_1,\dots,x_n]$.*

i.e.:

$$u(y_1,\dots,y_n) = \sum_{I\subseteq[n]} a_I \cdot \prod_{i\in I} y_i \quad a_I \in F_3$$

Now any monomial $\prod_{i\in I} y_i$ in u that is of deg $> \frac{n}{2}$ can be replaced by $\prod_i \in [n] y_i \cdot \prod_{i\in\bar{I}} y_i = g\cdot\prod_{i\in\bar{I}} y_i$ of degree deg $< \left(\frac{n}{2}+\sqrt{n}\right) Thusanyu \in F_G$ is a polynomial $F_3 : \sum_{I\subseteq[n],|I|<\left(\frac{n}{2}+\sqrt{n}\right)} a_I \prod_{i\in I} y_i$

Exercise 2.2.19. $\leq 0.992^n$

□

2.3 Monotone Circuits

Definition 2.3.1. *A boolean circuit is monotone if contains only AND and OR-gates and no NOT-gates.*

Remark 2.3.2. *A monotone circuit can compute only monotone functions:*

Definition 2.3.3. *1. For $x, y \in \{0,1\}^n$, $x \leq y$ if $\forall i \in [n]$, $x_i \leq y_i$.*

2. A function $f : \{0,1\}^n \to \{0,1\}$ is monotone if $f(x) < f(y)$ whenever $x < y$.

Definition 2.3.4. *Look at the functional problem*

$$Clique_{k,n} : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$$

that on a grpah $G \mapsto 1$ iff G has a k-clique (a complete graph on k vertices).

Clearly $Clique_{k,n}$ is a monotone function. Is there a "small" monotone circuit solving it? We will prove NO! (Note that Cique is NP-complete and we do expect $NP \nsubseteq P/poly$).

Definition 2.3.5. *$\forall S \subseteq [n]$ let $C_S : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ that on any input graph $G \mapsto 1$ iff $\exists$ clique on vertices S in G. Here C_S is called clique indicator of S.*

Theorem 2.3.6 (Razborov 80s)**.** *$\forall k \leq \sqrt[4]{n}$ there exists no monotone circuit of size $\leq n^{\frac{\sqrt{k}}{20}}$ solving $Clique_{k,n}$.*

Proof. The idea is to show that any monotone circuit K computing $Clique_{k,n}$ can be approximated by an OR of "few" clique indicators.

$$Clique_{k,n} \equiv \bigvee_{S \subseteq [n], |S| = n} C_S$$

Let us first show that $Clique_{k,n}$ cannot be computed by an OR of $\leq n^{\frac{n}{20}}$ clique indicators. Defin two distributions

$Y :=$ randomly choose K of size k and outputs the graph having a clique on K and no other edges.

$N :=$ randomly choose a functions $C : [n] \to [k-1]$, output the graph having an edge (u, v) iff $C(u) \neq C : V$

□

Remark 2.3.7. *$Clique_{k,n}$ is 1 on Y and 0 on N.*

Lemma 2.3.8. *If $k \le \sqrt[4]{n}$ and $S \subseteq [n]$ then either*

$$\text{Prob}_{G\in\mathbb{N}}[C_S(G) = 1] > 0.99$$

or

$$\text{Prob}_{GinY}[C_S(G)] < n^{-\frac{\sqrt{k}}{20}}$$

Proof. Define $l := \frac{\sqrt{k-1}}{10}$. If $S \le l$ then a random $f : S \to [k-1]$ is $1:1$ with probability $\ge 1 \cdot \left(1 - \frac{1}{k-1}\right) \cdots \left(1 - \frac{l}{k-1}\right) \ge e^{-\frac{1}{k-1} - \frac{2}{k-1} - \dots - \frac{l}{k-1}} \ge e^{-\frac{l^2}{k-1}} \ge e^{-\frac{1}{100}} \ge 0.99$
Case 2: $|S| > l$

$$\text{Prob}_{G\in Y}[C_S(G) = 1] = \text{Prob}_{K\subseteq[n],|K|=k}[S \subseteq k] \le \frac{\binom{n}{k-l}}{\binom{n}{k}}$$

Exercise 2.3.9.

$$\le \left(\frac{2k}{n}\right)^l < (n^{-0.7})^l < n^{\frac{-\sqrt{k}}{20}}$$

This means that an OR of $m \le n^{\frac{\sqrt{k}}{20}}$ clique indicators cannot compute $Clique_{k,n}$ because: the presence of een one indicator C_S for $|S| \le l$ will make the OR true with probability ≥ 0.99 on NO-instances. If all indicators are on Ss of size $> l$ then the $OR$is false on YES instances with probability $\ge \left(1 - \frac{1}{n^{\sqrt{k}}20}\right)^m \ge \frac{1}{e}$. Next we show how to approximate any "small" monotone circuit by an OR of "few" clique inndicators.

□

Chapter 3

Expanders

3.1 Pseudorandom Constructions: Expanders

Definition 3.1.1. *An undirected graph G is*

algebraic *an* $\underline{(n, d, \lambda)\textit{-expander}}$ *if G is d-regular, n-verrtex and the second-largest eigenvalue of A is $\leq \lambda$. ($0 \leq \lambda \leq (1 - o(1))\frac{2\sqrt{d-1}}{d}$)*

combinatorial *an* $\underline{(n, d, r)\textit{-edge-expander}}$ *if G is d-regular, n-vertex and $\forall S \subseteq V(G)$ s.t. $|s| \leq \frac{n}{2}$, $|E(S, \bar{S})| \geq r \cdot d \cdot |S|$. ($0 \leq r \leq \frac{1}{2}$)*

Lemma 3.1.2. *Let G be a (n, d, λ)-expander, it holds $\forall S \subseteq V(G)$ that:*

$$|E(S, \bar{S})| \geq \frac{(1 - \lambda)d}{n} \cdot |S| \cdot |\bar{S}|$$

<u>Proof.</u> Define a vector $\bar{x} \in \mathbb{R}^n$ as:

$$x_i = \begin{cases} |\bar{S}| & \text{if } i \in S \\ -|S| & \text{if } i \notin S \end{cases}$$

Consider

$$Z := \sum_{1 \leq i, j \leq n} A_{ij}(x_i - x_j)^2$$

and note that $Z = \frac{1}{d}|E(S, \bar{S})| \cdot n^2 \cdot 2$. On the other hand:

$$\begin{aligned} Z &= \textstyle\sum_{1 \leq i,j \leq n} A_{ij} x_i^2 + \sum_{1 \leq i,j \leq n} A_{ij} x_j^2 - 2\sum_{1 \leq i,j \leq n} A_{ij} x_i x_j \\ &= 2\textstyle\sum_{i=1}^n x_i^2 - 2\sum_{1 \leq i,j \leq n} A_{ij} x_i x_j \\ &= 2||\bar{x}||^2 - 2 < A\bar{x}, \bar{x} > \\ &\geq 2||\bar{x}||^2 - 2\lambda||\bar{x}||^2 \end{aligned}$$

Exercise 3.1.3. *1. Proof that $\sum_{1 \leq i,j \leq n} A_{ij} x_i x_j = < A\bar{x}, \bar{x} >$*

2. If $1 = \lambda_1, \ldots, \lambda_n$ are the eigenvalues of A s.t. $1 = |\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ then:

$$\lambda_2 = \max_{<\bar{v}, \bar{1}> = 0} \frac{||A\bar{v}||}{||\bar{v}||}$$

So finally we see that $\frac{2n^2}{d}|E(S,\bar{S})| \geq 2\|\bar{x}\|^2(1-\lambda)$ and $|E(s,\bar{S}) \geq \frac{d}{n^2}\|\bar{x}\|^2(1-\lambda) = \frac{d(1-\lambda)}{n^2}(|S|\cdot|\bar{S}|^2 + |\bar{S}|\cdot|S|^2) = \frac{d(1-\lambda)}{n}|S|\cdot|\bar{S}|$. □

Corrolar 3.1.4. *For $|S| \leq \frac{n}{2}$ it holds with $s = |S|$ that*

$$(1-\lambda)\frac{d}{n}s(n-s) \geq \frac{1-\lambda}{2}\cdot d\cdot s$$

Theorem 3.1.5. *If G is an (n,d,λ)-exnapder then it is an (n,d,λ)-edge-expander*

Proof. Lemma 3.1.2. □

Theorem 3.1.6. *If G is a (n,d,r)-edge-expander then G is a $(n,d,1-\epsilon)$-expander where $\epsilon = \min\{\frac{2}{d},\frac{r^2}{2}$. (G is an n-vertex, d-regular graph with all self-loops)*

Proof. Let $G = (V,E)$ be an n-vertex, d-regular graph s.t. $\forall S \subseteq V(G)$ with $|S| \leq \frac{n}{2}$ there are $\geq r\cdot d\cdot|S|$ edges between S and $V-S$. Let A be a G's randdom-walk matrix and let $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq -1$ be the eigenvalues of A.

Exercise 3.1.7.

Show that $\lambda_n = -1$ in an bipartit graph.

Claim 1: $\lambda_n \geq (-1+\frac{2}{d})$. Look at the graph G' after removing all self-loops from G and let A' be it's random-walk matrix:

$$(d-1)A' = dA - I_n \Rightarrow A = \left(\frac{d-1}{d}A' + \frac{1}{d}I_n\right)$$

So $\lambda_n \geq \frac{d-1}{d}(-1) + \frac{1}{d} = (-1+\frac{2}{d})$.

By the definition of eigenvalue we have that $\exists \bar{u} \in \mathbb{R}^n - \{\bar{0}\}$ s.t.

$$A\bar{u} = \lambda_2\bar{u} \text{ and } \bar{u} < \bar{u}, \bar{1} >= 0$$

Thus $\bar{u}$ has positive nand negative coordinates, let $\bar{u} = \bar{v} + \bar{w}$ where $\bar{v}$ is the positive part and $\bar{w}$ is the negative part. Note that $\bar{v},\bar{w} \neq \bar{0}$. Assume that $v_i = 0\ \forall i > \frac{n}{2}$ otherwise use $-\bar{u}$. Sort the $v_1 \geq v_2 \geq \ldots v_{\frac{n}{2}}$.

Define

$$Z := \sum_{1\leq i,j\leq n} A_{ij}(v_i^2 - v_j^2)$$

we will show that

Claim 2: $Z \geq 2r\|\bar{v}\|^2$

$$Z = \sum_{1\leq i,j\leq n} A_{ij}(v_i^2 - v_j^2) = \sum_{1\leq i,j\leq n} A_{ij}\sum_{k=i}^{j-1}(v_k^2 - v_{k+1}^2)$$

Rewriting this with respect to k and using $v_k = 0$ for $k > \frac{n}{2}$ the above equals

$$\frac{2}{d}\sum_{k=1}^{\frac{n}{2}}(v_k^2 - v_{k+1}^2)|E(\{1,\ldots,k\},\{k+1,\ldots,n\})| \geq \frac{2}{d}\sum_{k=1}^{\frac{n}{2}} pdk(v_k^2 - v_{k+1}^2)$$

which is equal to

$$2r\sum_{k=1}^{\frac{n}{2}}(kv_k^2 - kv_{k+1}^2) = 2r\sum_{k=1}^{\frac{n}{2}}(kv_k^2 - (k-1)v_k^2) = 2r||\bar{v}||^2$$

<u>Claim 3:</u> $Z \leq \sqrt{8(1-\lambda_2)}\cdot||\bar{v}||^2$.
Note that

$$< A\bar{u},\bar{v} >=< \lambda_2\bar{u},\bar{v} >=< \lambda_2\bar{v},\bar{v} > + < \lambda_2\bar{w},\bar{v} >= \lambda_2||\bar{v}||^2$$

and on the other hand:

$$| < A\bar{u},\bar{v} >=< A\bar{v},\bar{v} > + < A\bar{w},\bar{v} >\leq< A\bar{v},\bar{v} >$$

which implies that $\lambda_2 \leq \frac{<A\bar{v},\bar{v}}{||\bar{v}||^2}$ and by this we get

$$(1-\lambda_2) \geq \frac{||\bar{v}||^2 - < A\bar{v},\bar{v}}{||\bar{v}||^2} = \frac{\frac{1}{2}\sum_{1\leq i,j\leq n}A_{ij}(v_i-v_j)^2}{||\bar{v}||^2} = \frac{\sum_{1\leq i,j\leq n}A_{ij}(v_i-v_j)^2\cdot\sum_{1\leq i,j\leq n}A_{ij}(v_i-v_j)^2}{||\bar{v}||^2\cdot\sum_{1\leq i,j\leq n}A_{ij}(v_i-v_j)^2}$$

The Cauchy-Schwarzt inequality implies that the numerator is $\leq (\sum_{1\leq i,j\leq n}A_{ij}(v_i-v_j)^2)^2 = Z^2$. The denominator:

$$||\bar{v}||^2\cdot\sum_{1\leq i,j\leq n}A_{ij}(v_i-v_j)^2 = 2||\bar{v}||^2\cdot(\sum_{1\leq i,j\leq n}A_{ij}v_i^2+\sum_{1\leq i,j\leq n}A_{ij}v_j^2+2\sum_{1\leq i,j\leq n}A_{ij}v_iv_j = 2||\bar{v}||^2(2||\bar{v}||^2+2\sum_{1\leq i,j\leq n}A_{ij}v_iv_j \leq 2||v||^2($$

Which implies that

$$(1-\lambda_2) \geq \frac{Z^2}{8||\bar{v}||^4}$$

And finally $Z \leq \sqrt{8(1-\lambda_2)}||\bar{v}||^2$.

The three claims together show that for a combinatorial expander in a graph (with all the self-loops) it holds that:

$$\lambda_2 \leq (1-\frac{r^2}{2}) \text{ and } \lambda_n \geq (\frac{2}{d}-1)$$

□

3.2 Error-Reduction using expanders

Recall from the last semester that if a problem $L \in TPP$ has an algorithm $M(x,y)$ for $x \in L$ that uses r random bits y and has an error-probability $< \frac{1}{3}$ then by repeating the algorithm k times: $[M(x,y_1),\dots,M(x,y_k)]$ and taking the majority-vote we get a new algorithm M' that uses $r\cdot k$ random bits and has error-probability $< \frac{1}{2^k}$ (Chernoff bound).

Now we will use expanders to reduce the used random bits to $r+o(k)$ random bits. <u>Idea:</u> Suppose wwe have an $(2^r,d,\frac{1}{10})$-expander G for a constant d (for example 3) in which the neighbours of a vertex are listable in time $p(r)$ for a polynomial p. Choose a vertex $v_1 \in V(G)$ at random and do a random walk $(v_1,\dots,v_k)$. Use the v_i as the random strings to repeat M for k times. We will show that the majority-vote is wrong probability $< 2^{-k}$. The number of used random bits here is $\leq r + k\log(d)$. This means that in the expander G on $\{0,1\}^k$ vertices, the set $B \subseteq \{0,1\}^k$ of bad vertices (i.e. on which $M(x)$ makes an error)satisfies $|B| < \frac{2^k}{3}$. We will compute the probability that all the k vertices in the random walk are in B.

Theorem 3.2.1. *Let G be an (n,d,λ)-expander and $B \subset V = [n]$ with $|B| < \beta n$. Let $X_1,\dots,X_k$ be random variables denoting the k-random-walk in G. Then:*

$$\mathrm{Prob}_{k\text{-}walk}[\forall i \in [k] : X_i \in B] < ((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}$$

(Note that $(1-\lambda)\beta + \lambda < 1$ if $\beta, \lambda < 1$).

Proof. Let B_i be the event $X_i \in B$. We are trying to bound $\mathrm{Prob}[\bigwedge_{i=1}^n B_i] = \mathrm{Prob}[B_1]\cdot\mathrm{Prob}[B_1]\cdot\mathrm{Prob}[B_2|P_1]\dots\mathrm{Prob}[B_k|\bigwedge_{i=1}^{k-1} B_i]$. We will express this in terms of a matrxi product, so we define:

1. Let A be tghe random-walk matrix of G
2. Let B be the $n\times n$-identity matrix with rows corresponding to $[n]-B$ replaced by zreo
3. $\vec{1} := (\frac{1}{n})_i$

We see that

$$\mathrm{Prob}[B_1] = |B\vec{1}|_1$$

and

$$\mathrm{Prob}[B_1]\cdot\mathrm{Prob}[B_2|B_1] = |BA\cdot B\vec{1}|_1$$

Exercise 3.2.2. *1.*

$$\mathrm{Prob}[\bigwedge_{i=1}^{n} B_i] = |(BA)^{k-1}B\vec{1}$$

2. $\forall \vec{u} \in \mathbb{R}^n$*:* $||\vec{u}|| \leq |\vec{u}|_1 \leq \sqrt{n}||\vec{u}||$.

Thus it suffices to show that

$$||(AB)^{k-1}\cdot B\vec{1}|| < \frac{((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}}{\sqrt{n}}$$

To show this we will use the notion of the spectral norm $||(BA)^{k-1}B||_s$ and show it to be small: For any matrix $C \in \mathbb{R}^{n\times n}$ the spectral norm:

$$||C||_s := \max_{u\in\mathbb{R}^{n\times n}-\{0\}} \frac{||Cu||}{||u||}$$

it is a well-behaved norm of $n\times n$-matrices:

Exercise 3.2.3. *1.* $||C+D||_s \leq ||C||_s + ||D||_s$

2. $||CD||_s \leq ||C||_s||D||_s$

3. For a random-walk matrix A: $||A||_s = 1$.

Lemma 3.2.4. *Let A be the random-walk-matrix of an (n,d,λ)-expander and $T := (1/n)_{n\times n}$. Then*

$$A = (1-\lambda)T + \lambda C$$

where $||C||_s \le 1$.

Proof. Consider $C := \frac{1}{\lambda}(A-(1-\lambda)T)$. We will show that $||C\vec{v}|| \le ||\vec{v}||$ for all $\vec{v} \in \mathbb{R}^n$. Decompose $\vec{v} = \vec{u} + \vec{w}$ where $\vec{u} = \alpha\vec{1}$ and $< \vec{w}, \vec{1} >= 0$. Now $C\vec{u} = \frac{1}{\lambda}(A\vec{u}-(1-\lambda)T\vec{u}) = \frac{1}{\lambda}(\vec{u}-(1-\lambda)\vec{u}) = \vec{u}$ and $C\vec{w} = \frac{1}{\lambda}(A\vec{w}-(1-\lambda)T\vec{w}) = \frac{1}{\lambda}A\vec{w}$. And we see that $||C\vec{v}||^2 = ||C\vec{u} + C\vec{w}||^2 \le ||C\vec{u}||^2 + ||C\vec{w}||^2 \le ||\vec{u}||^2 + \frac{1}{\lambda}||A\vec{w}||^2 \le ||\vec{u}||^2 + ||\vec{w}||^2 = ||\vec{u}+\vec{w}||^2 = ||\vec{w}||^2$. □ This C will help usin bounding $||BA||_s$:

$$\begin{aligned} BA &= (1-\lambda)BT + \lambda BC \\ \Rightarrow \quad ||BA||_s &\le (1-\lambda)||BT||_s + \lambda||BC||_s \end{aligned}$$

Exercise 3.2.5. *Show that $||BT||_s = \sqrt{\beta}$ and $||BC||_s \le 1$.*

and we see that $||BA||_s \le (1-\lambda)\sqrt{\beta}+\lambda$ and $||(BA)^{k-1}||_s \le ((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}$ and finally

$$||(BA)^{k-1}B\vec{1}|| \le ((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}||B\vec{1}|| = ((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}\sqrt{\frac{\beta}{n}}$$

□

Corrolar 3.2.6. $\forall I \subset [k]$*:*

$$\text{Prob}[\forall i \in I, X_i \in B] \le \left((1-\lambda)\sqrt{\beta}+\lambda\right)^{|I|-1}$$

Using ail estimates for binomial sums we get:

Theorem 3.2.7. *(Expander-Chernoff-Bound) $\forall i \in [k]$ let $\tilde{B}_i$ be the random variable that is 1 if $X_i \in B$ and 0 otherwise. Then $\forall \delta > 0$:*

$$\text{Prob}_{k\text{-}walk}[|\frac{\sum_{i=1}^{k}\tilde{B}_i}{k} - \beta| > \delta] < 2\left(e^{-\frac{\delta^2(1-\lambda)}{3}}\right)^k$$

Exercise 3.2.8. *Proof 3.2.6 and 3.2.7.*

3.3 Explicit Construction of Expanders

We will construct an explicit family of expander graphs, where local connectivity information can be efficienbtly retrieved. The main idea would be to to define operations on graphs: graph products. The trade-off whould be between the degree and the second largest eigenvallue.

Till now we have used only adjacency matrix , now we will define another representation of a graph:

Definition 3.3.1. *If $G$is an n-vertex, d-regular graph then we give each neighbour of each vertex a label $\in [d]$ and define a rotation map:*

$$\begin{array}{rrcl} \hat{G}: & [n]\times[d] & \to & [n]\times[d] \\ & (v,i) & \mapsto & (u,j) \end{array}$$

where u is the i-th neighbour of v and v is the j-th neighbour of u.

Remark 3.3.2. *$\hat{G}$ is a permutation on $[n]\times[d]$ and describes G.*

3.3.1 The Matrix-product

Definition 3.3.3. *For two n-vertex graphs G and G' with degrees d and d' and random-walk matrices A and A' we define a new graph GG' as the graph with the random-walk matrix $A\cdot A'$ (or equivalently $d\cdot d'\cdot A\cdot A'$).*

Remark 3.3.4.

- *$A\cdot A'$ is a symmetric stochastic matrix*
- *The matrix-product of two graphs are general graphs possibly with multiple edges and loops*
- *GG' has degree $d\cdot d'$ (counting multiple edges) and n vertices*

Lemma 3.3.5.

$$\lambda(GG')\le\lambda(G)\cdot\lambda(G')$$

Proof.

$$\lambda(GG')=\lambda(A\cdot A')=\max_{<\vec{u},\vec{1}>=0}\frac{||AA'\vec{u}||}{||\vec{u}||}=\max_{<\vec{u},\vec{1}>=0}\frac{||AA'\vec{u}||}{||A'\vec{u}||}\frac{||A'\vec{u}||}{||\vec{u}||}$$

and we know that $<A'\vec{u},\vec{1}>=0$ because A' is a random-walk matrix. So:

$$\max_{<\vec{u},\vec{1}>=0}\frac{||AA'\vec{u}||}{||A'\vec{u}||}\frac{||A'\vec{u}||}{||\vec{u}||}\le\lambda(A)\lambda(A')=\lambda(G)\lambda(G')$$

□

Theorem 3.3.6. *If G and G' are (n,d,λ) and (n,d',λ')-expanders then GG' is an $(n,dd',\lambda\lambda')$-expander.*

Thus the matrix product improves the expansion at the cost of the degree.

3.3.2 The Tensor-product

Definition 3.3.7. *For two graphs G and G' of n and n' vertices, d and d' degress and random-walk matrices A and A' the graph $G\otimes G'$ is the graph with the random-walk matrix $A\otimes A'$ (with adjacency matrix $dd'A\otimes A'$).*

$$A\otimes A':=\begin{pmatrix} A_{1,1}A' & \cdots & A_{1,n}A' \\ \vdots & & \vdots \\ A_{n,1}A' & \cdots & A_{n,n}A' \end{pmatrix}_{nn'\times nn'}$$

Exercise 3.3.8. *1. $A \otimes A'$ is a symmetric stochastic matrix.*

2. $G \otimes G'$ has nn' vertices and degree dd'

Lemma 3.3.9.

$$\lambda(G \otimes G') \leq \max\{\lambda(G), \lambda(G')\}$$

Proof. If $1 = \lambda_a \geq \ldots \geq \lambda_n \geq -1$ are the eigentvalues of A and $1 = \lambda'_1 \geq \ldots \geq \lambda'_{n'} \geq -1$ are the eigenvalues of A' then all the eigenvalues of $A \otimes A'$ are $L = \{\lambda_i \lambda'_j \mid i \in [n], j \in [n']\}$ (exercise). Thus the largest in L apart from 1 are of the form $1 \cdot \lambda'_j$ and $\lambda i \cdot 1$ for all $i \in [n]$ and $j \in [n']$. So the largest of them is exactle $\max\{\lambda(G), \lambda(G')\}$. □

Theorem 3.3.10. *If G and G' are (n, d, λ) and (n', d', λ') expanders then $G \otimes G'$ is an $(nn', dd', \max\{\lambda, \lambda'\})$-expannder.*

Thus the tensor product increases the number of vertices and the degree while preserving the expansion.

3.3.3 The Replacement-product

This product is easier to see as a walk rather than a matrix operation. Suppose we have an expander G with high degree D then, in order to walk using fewer random bits, we can use another expander G' with D vertices and degree $d << D$. This motivates the following product:

Definition 3.3.11. *Let G and G' be graphs with vertices n and D, degrees D and d and random-walk matrices A and A'. The new graph $\underline{G \odot G'}$ is a graph on $n \cdot D$ vertices s.t.*

1. *$\forall u \in V(G), G \odot G'$ has a copy of G'*
2. *If $\{u, v\} \in E(G)$ then we place d parallel edges between $\{u, i\}$ and $\{v, j\}$ in $G \odot G'$ where $\hat{G}(u, i) = (v, j)$.*

Remark 3.3.12. *$G \odot G'$ has nD vertices and degree $2d$.*

From the definition the following matrix seems a candidate for the random-walk matrix of $G \odot G'$:

$$\begin{pmatrix} A' & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A' \end{pmatrix}_{nD \times nD} + \hat{A}$$

where $\hat{A}$ has 1 at the entry $((u, i), (v, j))$ iff $\hat{G}(u, i) = (v, i)$ for all $u, v \in [n]$ and $i, j \in [D]$. After normalization it defines a new matrix product:

$$A \odot A' := \frac{1}{2}(I_n \otimes A' + \hat{A})$$

Exercise 3.3.13. *Show that $A \odot A'$ is the random walk matrix of $G \odot G'$.*

Lemma 3.3.14. *If for $\epsilon, \delta > 0$, $\lambda(G) \le 1-\epsilon$ and $\lambda(G') \le 1-\delta$ then:*

$$\lambda(G \odot G') \le 1 - \frac{\epsilon\delta^2}{24}$$

<u>Proof.</u> We will show that $\lambda((A \odot A')^3) \le (1 - \frac{\epsilon\delta^2}{8})$ which implies that $(\lambda(A \odot A'))^3 \le (1 - \frac{\epsilon\delta^2}{8}$ and $\lambda(A \odot A') \le (1 - \frac{\epsilon\delta^2}{24})$.

So let $B := (A \odot A')^3$ it holds that

$$B = \left(\frac{1}{2}(I_n \otimes A') + \frac{\hat{A}}{2}\right)^3$$

and since we have seen that $\exists C$ with $\|C\|_s \le 1$ s.t. $A' = (1-\delta)C + \delta J_D$ where $J_D = \frac{I_D}{D}$ we write

$$B = \left(\frac{\hat{A}}{2} + \frac{1-\delta}{2}(I_n \otimes C) + \frac{\delta}{2}(I_n \otimes J_D)\right)^3$$

and we see that in this expansion $\hat{A}$, $I_n \otimes C$ and $I_n \otimes J_D$ have spectral norms ≤ 1.

$$B = \left[\left(\frac{1}{2} + \frac{1-\delta}{2} + \frac{\delta}{2}\right)^3 - \frac{1}{2}\frac{\delta}{2}\frac{\delta}{2}\right]C' + \frac{\delta}{2}(I_n \otimes J_D)\frac{\hat{A}}{2}(I_n \otimes J_D)\frac{\delta}{2}$$

where C' has spectral norm ≤ 1.

$$B = \left(1 - \frac{\delta^2}{8}\right)C' + \frac{\delta^2}{8}(I_n \otimes J_D)\hat{A}(I_n \otimes J_D)$$

Exercise 3.3.15. *Show that $(I_n \otimes J_D)\hat{A}(I_n \otimes J_D) = A \otimes J_d$.*

$$B = \left(1 - \frac{\delta^2}{8}\right)C' + \frac{\delta^2}{8}(A \otimes J_D)$$

with $\|C'\|_s \le 1$. And finally:

$$\lambda(B) \le \left(1 - \frac{\delta^2}{8}\right)\lambda(C') + \frac{\delta^2}{8}\lambda(A \otimes J_D) \le \left(1 - \frac{\delta^2}{8}\right)\cdot 1 + \frac{\delta^2}{8}\cdot \max\{\lambda(A), \lambda(J_D)\}$$

where $\lambda(J_D) = 0$, so we get

$$\lambda(B) \le 1 - \frac{\delta^2}{8} + \frac{\delta^2}{8}(1-\epsilon)$$

□

Theorem 3.3.16. *If G and G' are $(n, D, 1-\epsilon)$ and $(D, d, 1-\delta)$-expanders then $G \odot G'$ is an $\left(nD, 2d, 1 - \frac{\epsilon\delta^2}{24}\right)$-expander*

Thus when $D >> d$ the Replacement-product significantly reduces the degree while almost preserving expansion.

3.4 The Construction

Using this three products we now give strongly explicit construction of an (n, d, λ)-expanders G, i.e. for $(u, i) \in E(G)$ the i-th neighbour of u can be computed in $O(p(\log(n)))$ time (where p is a polynomial).

Theorem 3.4.1. *(Reingold, Vadhan, Wigdesson 2000) $\exists$ a strongly explicit (d, λ)-expander family for some $d \in \mathbb{N}$, $0 < \lambda < 1$.*

Proof. We will recursively construct a family of graphs $\{G_k\}_{k \geq 1}$ s.t. G_k has at most c^k vertices (for some constant c). Let H be a $((2d)^{100}, d, 0.01)$-expander. It can be found by brute-force for a constant d large enough (the existance of such a d and H is garantueed be probabilistic reasons). Instead of brute-force, number theory constructions can be used to find H. Also find a $((2d)^{100}, 2d, 0.5)$-expander G_1. Now set $G_2 = G_1$. For odd $k > 2$ define

$$G_k := \left(G_{\frac{k-1}{2}} \otimes G_{\frac{k-1}{2}}\right)^{50} \odot H$$

and for even $k > 3$ we set $G_k = G_{k-1}$.

Claim: For every odd k the graph G_k is a $((2d)^{100k}, 2d, 1 - \frac{1}{50})$-expander.

This is clear for $k = 1$. By induction the number of vertices in G_k equals

$$\left((2d)^{100\frac{k-1}{2}}\right)^2 \cdot (2d)^{100} = (2d)^{100k}$$

and the degree of $\left(G_{\frac{k-1}{2}} \otimes G_{\frac{k-1}{2}}\right)^{50}$ is $(2d)^{2\cdot 50}$. Replacement with H is well defined and so the degree of G_k is $2d$. Finally

$$\lambda\left(\left(G_{\frac{k-1}{2}} \otimes G_{\frac{k-1}{2}}\right)^{50}\right) \leq \left(1 - \frac{1}{50}\right)^{50} < e^{-1} < \frac{1}{2}$$

and $\lambda(H) \leq 0.01 = 1 - 0.99$, which implies that $\lambda(G_k) \leq 1 - \frac{0.5 \cdot 0.99^2}{24} < 1 - \frac{1}{50}$. The algorithm for finding all the neighbours of a vertex in G_k is also recursive: Say listing a row in $G_{\frac{k-1}{2}}$ takes time $T\left(\frac{k-1}{2}\right)$. Using this listing algorithm we can list a row in $G_{\frac{k-1}{2}} \otimes G_{\frac{k-1}{2}}$ in time $2T\left(\frac{k-1}{2}\right) + p(d, k)$ (as usual p is a polynomial) and uses $(2d)^2$ space. Listing all the needed $(2d)^2$ rows will take $(4d^2) \cdot 2T\left(\frac{k-1}{2}\right)$ in one matrix. Powering by 50 takes $50 \cdot (4d^4) \cdot 2T\left(\frac{k-1}{2}\right) + poly(d, k)$ time. Listing a row in G_k takes

$$T(k) \leq (400d^2)T\left(\frac{k-1}{2}\right) + poly(d, k)$$

so

$$T(k) \leq p\left(d^{\log(k)}\right) = d^{O(\log(k)} = k^{O(\log(k)}$$

the number of vertices in G_k is $(2d)^{100k} =: n$ vertices:

$$T(k) \leq (\log(n))^{O(\log(d))} = p(\log(n))$$

□

3.5 Derandomization of UDC

Now we are ready to derandomize undirected connectivity's LogSpace-algorithm.

Theorem 3.5.1. *(Reingold 2005) Undirected connectivity* $\in$ LogSpace

Proof. Given an n-vertex graph G and $s,t \in V(G)$. Idea: apply graph products on G to get $\tilde{G}$ s.t. the connected component of s in $\tilde{G}$ becomes an expander. Recall that in an expander the shortest path are $O(\log(n))$ long. Thus, we can now check all the paths of length $O(\log(n))$ in $\tilde{G}$ starting from s in LogSpace. Let d be a large enough constant s.t $\exists$ a $\left(d^{50}, \frac{d}{2}, 0.01\right)$-expander H. Compute it and keep this H. We can make G a d^{50}-regular graph.

Exercise 3.5.2. *Show that this is possible.*

For this exposition assume G is the connected component of s. $G_0 := G$ and for $k \geq 1$ define:

$$G_k := (G_{k-1} \odot H)^{50}$$

Claim: For all k it holds that G_k is a $\left(d^{50k}n, d^{50}, \max\{1 - \frac{1}{10}, 1 - \frac{2^k}{12n^2}\}\right)$-expander.

For $k = 0$ the claim holds because any connected graph has $\lambda < \left(1 - \frac{1}{12n^2}\right)$ (see last semester or the book). For $k \geq 1$ it holds that

$$\begin{aligned} |V(G_k)| &= d^{50(k-1)}nd^{50} = d^{50k}n \\ \deg(G_k) &= d^{50} \end{aligned}$$

If $\lambda G_{k-1} \leq 1 - \epsilon$ for $\epsilon = \min\left\{\frac{1}{10}, \frac{2^{k-1}}{12n^2}\right\}$ then $\lambda(G_{k-1} \odot H) \leq 1 - \frac{\epsilon \cdot 0.99^2}{24}$ and $\lambda(G_k) \leq \left(1 - \frac{\epsilon \cdot 0.99^2}{24}\right)^{50} < (1 - 2\epsilon)$ thus we double ϵ with each iteration. So for $l = 2\log(n)$ we get that G_l is an expander with $\lambda < 1 - \frac{1}{10}$. So for any given graph G we consider the strongly explicit expander G_l and test all parts of length l from s.

Exercise 3.5.3. *Show that a row in G_l can be listed in* LogSpace *using recursion. And finally check whether $(\tilde{s}, \tilde{t})$ have an l-path in G_l.*

$\square$

Chapter 4

Pseudorandom generators

4.1 Introduction and Definition

Expanders help in derandomizing the algorithm for undirected connectivity. Pseudorandom generators (PRGs) are objects to derandomize more general randomized algorithms.

Definition 4.1.1. *A distribution R on $\{0,1\}^m$ is called $\underline{(S,\epsilon)\text{-pseudorandom}}$ if for all circuits C of size $\leq S$:*

$$\left|\Pr_{x\in R}[C(x)=1] - \Pr_{x\in U_m}[C(x)=1]\right| < \epsilon$$

where $\underline{U_m}$ is the uniform distribution on $\{0,1\}^m$. The difference measures how well C can distinguish R from U_m.

Definition 4.1.2. *If $S : \mathbb{N} \to \mathbb{N}$ then a 2^n-time computable $G : \{0,1\}^* \to \{0,1\}^*$ is called an $\underline{S(l)\text{-PSG}}$ if $\forall\ l \in \mathbb{N}$ it holds that $G : \{0,1\}^l \to \{0,1\}^{S(l)}$ and $G(U_l)$ is $(S(l)^3, 0.1)$-pseudorandom. The constants 3 and 0.1 are chosen for convenience.*

4.2 Derandomization and PRG

Lemma 4.2.1. *Let $l, S : \mathbb{N} \to \mathbb{N}$ be easily computable functions. If $\exists S(l)$-PRG then*

$$\mathrm{BPTime}(S(l(n))) \subseteq \mathrm{DTime}(2^{O(l(n))} \cdot S(l(n)))$$

This gives the following conditional derandomizations:

Corrolar 4.2.2.

1. *If $\exists\ 2^{\epsilon l}$-PRG (for some $\epsilon > 0$) then $BPP = P$.*
2. *If $\exists\ 2^{l^\epsilon}$-PRG (for some $\epsilon > 0$) then $BPP \subseteq \mathrm{DTime}(2^{p(\log(n))}) =: \mathrm{QuasiP}$ where p is a polynomial.*
3. *If $\forall c > 1$, $\exists l^c$-PRG then $BPP \subseteq SUBEXP := \bigcap_{\epsilon>0} \mathrm{DTime}(2^{n^\epsilon})$.*

Exercise 4.2.3. *Proof 4.2.2.*

Proof. (of 4.2.1) a language is $L \in \mathrm{BPTime}(S(l(n)))$ if $\exists$ an algorithm A that uses $m := S(l(n))$ random bits r on an input $x \in \{0,1\}^n$ and runs for time $O(S(l(n)))$ s.t.

$$\Pr_{r \in \{0,1\}^m}[A(x,r) = L(x)] \geq \frac{2}{3}$$

The derandomization idea is to use an $S(l)$-PRG G to produce every possible value for r:

On input $x \in \{0,1\}^n$ out deterministic alogirhtm B will go over all $z \in \{0,1\}^{l(n)}$ strings, compute $A(x, G(z))$ and output the majority vote. We claim that

$$\Pr_{z \in \{0,1\}^{l(n)}}[A(x,G(z)) = L(x)] \geq \frac{2}{3} - 0.1 > \frac{1}{2}$$

thus B is a correct algorithm for L. Suppose the contrary:

$$\left| \Pr_{z \in \{0,1\}^{l(n)}}[A(x,G(z)) = L(x)] - \Pr_{r \in \{0,1\}^m}[A(x,r) = L(x)] \right| > 0.1$$

Now define a circuit C that on input $y \in \{0,1\}^m$ output 1 iff $A(x,G(z)) = L(x)$ (C can compute $A(x,y)$ and the value $L(x)$ is wired-in). Since A is $O(S(l(n)))$-time algorithm C is of size $O(S^2(l(n)))$. But then C can distinguish the uniform-distribution U_m from $G(U_{l(n)})$ which contradicts G being an $S(l)$-PRG. Thus B is a deterministic algorithm solving L correctly in time $O(2^{l(n)} \cdot 2^{l(n)} \cdot S(l(n)))$ and $L \in \mathrm{DTime}(2^{2l(n)} \cdot S(l(n)))$. □

How do we construct a PRG? Miraculously using hard functions! (Hardness vs. Randomness).

4.3 Hardness and PRG

We define two types of circuit hardness of boolean functions:

Definition 4.3.1. *(Worst-case hardness) Let $f : \{0,1\}^* \to \{0,1\}$ then $\underline{H_{wrs}(f)}$ is the largest $S(n)$ s.t. $\forall$ circuits C on n bits of size $\leq S(n)$ it holds that*

$$\Pr_{x \in \{0,1\}^n}[C(x) = f(x)] < 1$$

(which means that C cannot compute f).

Definition 4.3.2. *(Average-case hardness) Let $f : \{0,1\}^* \to \{0,1\}$ then $\underline{H_{avg}(f)}$ is the largest $S(n)$ s.t. $\forall$ circuits C on n bits of size $\leq S(n)$ it holds that*

$$\Pr_{x \in \{0,1\}^n}[C(x) = f(x)] < \frac{1}{2} + \frac{1}{S(n)}$$

Exercise 4.3.3. *Shwo that for any function $f : \{0,1\}^n \to \{0,1\}$ it holds that $H_{avg}(f) \leq H_{wrs}(f) \leq O\left(\frac{2^n}{n}\right)$*

Conjecture natural examples:

1. $H_{wrs}(3SAT) \geq 2^{\Omega(n)}$
2. $H_{avg}IntFactoring \geq n^{\omega(1)}$

Later we will build worst-case hard function from average-case hard function. Now we show that hard-on-average functions give a PRG.

Theorem 4.3.4. *(Nisan, Wigderson '88) If* $\exists f \in E$ *(so solvable in* $2^{O}(n)$*) with* $H_{avg}(f) \geq S(n)$ *then* $\exists S'(l)$*-PRG where* $S'(l) = S(n)^{\delta}$ *for some* $\delta > 0$ *(*l *will be like* n *in the construction).*

Corrolar 4.3.5. $\exists f \in E$ *with* $H_{avg}(f) \geq 2^{\epsilon n}$ *then* $BPP = P$*.*

Corrolar 4.3.6. $\exists f \in E$ *with* $H_{avg}(f) \geq 2^{n^{\epsilon}}$ *then* $BPP \subseteq \text{QuasiP}$*.*

Corrolar 4.3.7. $\exists f \in E$ *with* $H_{avg}(f) \geq n^{\omega(1)}$ *then* $BPP \subseteq SUBEXP$*.*

Chapter 5

Error-Correction

5.1 Concatenated Code

Walsh-Hadamard code has a large code length, while Reed-Solomon code has a nonbinary alphabet.

Definition 5.1.1. *Let $\mathbb{F}$ be a finite field, $RS : \mathbb{F}^n \to \mathbb{F}^n$ and $WH : \{0,1\}^{\log(|\mathbb{F}|)} \to \{0,1\}^{|\mathbb{F}|}$ then the concatenated code is $WH \circ RS : \{0,1\}^{n\log(|\mathbb{F}|)} \to \{0,1\}^{m|\mathbb{F}|}$ by*

1. *View RS as a code from $\{0,1\}^{n\log(|\mathbb{F}|)}$ to $\mathbb{F}^m$ and WH as code from $\mathbb{F}$ to $\{0,1\}^{|\mathbb{F}|}$ (using a natural boolean representation of field elements)*

2. *$\forall x \in \{0,1\}^{n\log(|F|)}, WH \circ RS(x) =< WH(RS(x)_1), \dots, WH(RS(x)_m) >$.*

Remark 5.1.2. *$WH \circ RS$ is computable in $\operatorname{poly}(|\mathbb{F}|)$ time*

Lemma 5.1.3. *Let $\delta_1 = \left(1 - \frac{n}{m}\right)$ be the distance of RS and $\delta_2 = \frac{1}{2}$ be the distance of WH. Then $WH \circ RS$ has distance $\delta_1\delta_2$.*

Proof. Let $x, y \in \{0,1\}^{n\log(|\mathbb{F}|)}$ with $x \neq y$.

$$\Delta(RS(x), RS(y)) \geq \delta_1$$

If $x', y' \in \mathbb{F}$ in the i-th place of $RS(x)$ and $RS(y)$ are different then $\Delta(WH(x'), WH(y')) \geq \delta_2$. Overall we get $\Delta(WH(RS(x)), WH(RS(y))) \geq \delta_1\delta_2$. □

Recall that $\forall k \geq 2\ \exists p \in [k, 2k) \cap \mathbb{P}$ which gives us a field $\mathbb{F} := \mathbb{F}_p$. Thus $WH \circ RS$ gives us an ECC.

$$E : \{0,1\}^{k\log(k)} \to \{0,1\}^{10k \cdot 2k}$$

with distance $\frac{1}{2}\left(1 - \frac{n}{m}\right) > \frac{1}{2}\left(1 - \frac{k}{10k}\right) > 0.4$. So for all $n \in \mathbb{N}$ there exists a polynomial-time computable (in n) ECC $E : \{0,1\}^n \to \{0,1\}^{20n^2}$ of distance 0.4 (and unique decoding up to 20% of errors).

5.2 Efficient Decoding

We need an efficient way to decode the message x from $E(x)$, even if we have a corrupted $E(x)$. Let us now decode Redd-Solomon:

Theorem 5.2.1. *Given a list $(a_1, b_1), \dots, (a_m, b_m) \in \mathbb{F}^2$ for which exists a polynomial $G(x)$ of degree d (with $m > d$) satisfying $G(a_i) = b_i$ for t of the pairs where $t > m\left(\frac{1}{2} + \frac{d}{2m}\right)$. Then there exists a polynomial time algorithm that reconstructs G.*

Remark 5.2.2. *RS-code: $\delta > \left(1 - \frac{n-1}{m}\right), d = (n-1)$; so even if $RS(x)$is corrupted in $\frac{1}{2}\left(1 - \frac{n}{m}\right)$ places x can be reconstructed by the above algorithm.*

Proof. This algorithm is called Berlekamp-Welsh decoding (1986). Without error we whould have tried to compute a degree d polynomial $C(x)$ s.t. $C(a_i) = b_i \quad \forall i \in [m]$.

1. Find a degree $\left(\frac{m+d}{2}\right)$ polynomial $C(x)$ and a degree $\left(\frac{m-d}{2}\right)$ polynomial $E(x)$ s.t. $C(a_i) = b_i \cdot E(a_i)$ for all $i \in [m]$. View these as m linear equations in $\frac{m+d}{2} + 1 + \frac{m-d}{2} + 1 = (m+2)$ variables. Note that this system has at least one solution: Pick $E(a_i) = 0$ whenever $G(a_i) \neq b_i$. Put $C(x) = G(x) \cdot E(x)$.
2. Output $C(x)/E(x)$. Consider $T(x) := C(x) - E(x)G(x)$. It holds that $\deg(T) \leq \frac{m+d}{2} < t$. Whenever $G(a_i) = b_i$: $T(a_i) = C(a_i) - E(a_i)b_i = 0$ and $T(x) = 0 \Rightarrow \frac{C(x)}{E(x)} = G(x)$.

□

5.3 Decoding Concatenated Code

Theorem 5.3.1. *For $WH \circ RS : \{0,1\}^{n \log(|\mathbb{F}|)} \to \{0,1\}^{m|\mathbb{F}|}$ there exists a $\mathrm{poly}(|\mathbb{F}|)$-time decoder that uniquely corrects by $\frac{1}{4}\left(\frac{1}{2} - \frac{n}{2m}\right)$ errors.*

Proof. Given a corrupted y "close to" $< WH(RS(x)_1), \dots, WH(RS(x)_m) >$. Note that

$$\left|\left\{ i \in [m] \mid WH(RS(x)_i) \text{ has } \geq \frac{|\mathbb{F}|}{4} \text{ errors} \right\}\right| < m\left(\frac{1}{2} - \frac{n}{m}\right)$$

Thus WH-decoding of the blocks in y gives $< \tilde{y}_1, \dots, \tilde{y}_m >$ then $\tilde{y}_i = RS(x)_i$ for $\geq m\left(\frac{1}{2} - \frac{n}{2m}\right)$ of the is. RS-decoding of $< \tilde{y}_1, \dots, \tilde{y}_m >$ gives back x uniquely. □

5.4 Local Decoding

Definition 5.4.1. *Let $E : \{0,1\}^n \to \{0,1\}^n$ be an error-correcting code and $r \in (0,1)$. A local decoder for E handling r errors is an algorithm that, given $j \in [n]$, oracle access to $y \in \{0,1\}^m$ s.t. $\Delta(y, E(x)) < r$, outputs x_j with probability $\geq \frac{2}{3}$ and in $\mathrm{poly}(\log(m))$-time.*

5.4.1 Local decoding for Walsh-Hadamard-code

Theorem 5.4.2. *For every $r < \frac{1}{4}$, the Walsh-Hadamard code has a local decoder handling r errors.*

Proof. Input: $j \in [n]$, orace $f : \{0,1\}^n \to \{0,1\}, x \mapsto WH(x) \mapsto y$ (where $x \in \{0,1\}^n$ and $WH(x) \in \{0,1\}^{2^n}$) s.t.

$$\text{Prob}_z[f(z) \neq x \odot z] \leq r$$

Output: a bit (goal: x_j).
Algorithm:

- $e_j \in \{0,1\}^n$ is all zero except 1 at j-th place.
- randomly pick $z \in \{0,1\}^n$
- ouptut $f(z) + f(z + e_j) \pmod 2$.

Analysis:
Time: poly(n) = poly(log(m)).
Probability:

$$\text{Prob}_z\left[f(z) = x \odot z \wedge f(z+e_j) = x \odot (z+e_j)\right] \geq 1 - 2r > \frac{1}{2}$$

so

$$\text{Prob}_z\left[f(z) + f(z+e_j) = x \odot (z+z+e_j) = x_j \pmod 2\right] \geq 1 - 2r > \frac{1}{2}$$

By repeating algorithm twice we get error probability $< \frac{1}{4}$. □

5.4.2 Local decoder for RM Code

Recall that $RM : \mathbb{F}^{\binom{l+d}{d}} \to \mathbb{F}^{|\mathbb{F}|^l}$ for a finite field $\mathbb{F}$. For local decoding we will redefine RM to be mapping $\binom{l+d}{d}$ evaluations of a polynomial to $|\mathbb{F}|^l$ evaluations.

Theorem 5.4.3. *There exists a* poly$(|\mathbb{F}|, l, d)$*-time algorithm s.t. given $x \in \mathbb{F}^l$ and an oracle access to $f : \mathbb{F}^l \to \mathbb{F}$ (that agrees with some l-variate d-degree polynomial p on $\geq 1 - \underbrace{\frac{1}{6}\left(1 - \frac{d}{|\mathbb{F}|}\right)}_{r}$ evaluations) outputting $P(x)$ with probability $\geq \frac{2}{3}$.*

Proof. Idea: Pick a ranndom line L_x throigh x, evaluate f at each point of the line and use RS decoder.
Algorithm:

1. Pick a random $z \in \mathbb{F}^l$ and
$$L_x := \{x + tz \mid t \in \mathbb{F}\}$$
2. Query f on the while of L_x i.e. collect pairs: $\{(t, f(x+tz)) \mid t \in \mathbb{F}\}$

3. Run RS decoder on this set of pairs to find a univeriate Q s.t. $Q(t) = f(x+tz)$ for the largest number of t's.

4. Output $Q(0)$.

Time: poly$(|\mathbb{F}|, l, d)$-time
Analysis: The polynomial that RS decoder tries to reconstruct is $P(x + tz) := \tilde{Q}(t)$ which has degree $\leq d$ in variable t. Thus we only need to shat the probability of $\geq \frac{2}{3}$.

$$\frac{\#\{t \in \mathbb{F} \mid \tilde{Q}(t) \neq f(x+tz)\}}{\#\mathbb{F}} < \frac{1}{2}\left(1 - \frac{d}{|\mathbb{F}|}\right)$$

For that we compute the expected value:

$$E_z\left[\#\{t \in \mathbb{F} \mid \tilde{Q}(t) \neq f(x+tz)\}\right] = \sum_{t \in \mathbb{F}} \text{Prob}_z[P(x+tz) \neq f(x+tz)] \leq \sum_{t \in \mathbb{F}} r = r|\mathbb{F}|$$

Thus by Markov-Inequality:

$$\text{Prob}_z[\#\{t \in \mathbb{F} \mid \tilde{Q}(t) \neq f(x+tz)\} \geq \frac{1}{2}\left(1 - \frac{d}{|\mathbb{F}|}\right)|\mathbb{F}|] \leq \frac{r|\mathbb{F}|}{\frac{1}{2}\left(1 - \frac{d}{|\mathbb{F}|}\right)|\mathbb{F}|} = \frac{1}{3}$$

This means with $\text{Prob}_z \geq \frac{2}{3}$ step (3) produces $Q(t) = \tilde{Q}(t) = P(x+tz) \Rightarrow Q(0) = P(x)$. □

5.4.3 Local decoder for Conactenated-Code

Theorem 5.4.4. *Let $E_1 : \{0,1\}^n \to \Sigma^m$ (resp. $E_2 : \Sigma \to \{0,1\}^k$) be two error-correcting codes (where Σ is a set of arbitrary symbols) with local decoders using q_1 (resp. q_2) queries and handling r_1 (resp. r_2) errors. Then there exists an $O(q_1 q_2 \log(q_1)\log(q_2)\log(|\Sigma|))$-query local decoder for the error-correcting code $E := E_2 \circ E_1 : \{0,1\}^n \to \{0,1\}^{km}$, handling $r_1 \cdot r_2$ errors.*

Proof.
Input: An $i \in [n]$ and oracle access to $y \in \{0,1\}^{mk}$ s.t. $\Delta(y, E(x)) < r_1 r_2$.
Output: a bit (goal: x_i)
Algorithm:

1. View y as m blocks each of k-bits

   (Note that $E(x) =< E_2(E_1(x)_1), \dots, E_2(E_1(x)_m) >$)

2. To find the j-th symbol of $E_1(x)$ (i.e. $E_1(x)_j$) we call the local decoder of E_2 on "j-th block of y", $\log(|\Sigma|)$ times (to compute the full $E_1(x)_j \in \Sigma$). To get a decent probability we first boost E_2's local decoder by repeating $\log(q_1)$ times. Thus finding $E_1(x)_j$ takes $O(q_2 \log(|\Sigma|)\log(q_1))$-queries and has error-probability $\leq \frac{1}{10q_1}$ (since $\frac{\log(|\Sigma|)}{3^{\log(q_1)}} \leq \frac{1}{10q_1}$). Note that at most r_1 of the m blocks in y can be at distance $> r_2$ from the corresponding "correct" $E_1(x)_j$ block. Thus local decoder of E_1 can be used querying q_1

of the blocks in y. With probability $\geq 1 - \frac{1}{10q_1} \cdot q_1 = 0.9$ the q_1 answers to the local decoder of E_1 are all correct. So E_1 outputs x_j with probability $\geq 0.9 - \frac{1}{3} > 0.6$.

□

5.4.4 Local List Decoding

Out final goal is to show that starting from a worst-case hard boolean function f and a locally decodable code E, $E(\hat{f})$ is the true table of an average-case hard function g (where $\hat{f}$ is the true table of f on $\{0,1\}^n$). But for average-case hardness of g (i.e. g cannot be computed upto $\frac{1}{2} + \frac{1}{s}$ inputs) we whould need E that is locally decodable upto $\left(\frac{1}{2} - \delta\right)$-errors.

None of out error-correcting codes are "uniquely decodable" at $\left(\frac{1}{2} - \delta\right)$-error. So we relax "unique decodability".

Theorem 5.4.5. *(Johnson's bound, 1962) If $E : \{0,1\}^n \to \{0,1\}^m$ is an error-correcting code with distance $\geq \left(\frac{1}{2} - \epsilon\right)$ then $\forall x \in \{0,1\}^m$ and $\delta \geq \sqrt{\epsilon}$ there exist at most $\frac{1}{2}\delta^2$ codewords $y_1, \ldots, y_l \in \{0,1\}^m$ s.t.*

$$\Delta(x, y_i) \leq \frac{1}{2} - \delta \quad \forall i$$

Proof. Let us define l vectors $z_1, \ldots, z_l \in \{-1,1\}^m$ s.t.

$$z_{i,k} := \begin{cases} +1 & \text{if } x_k = y_{i,k} \\ -1, \text{else} & \end{cases}$$

Now since $\Delta(x, y_i) \leq \left(\frac{1}{2} - \delta\right)$ one can calculate

$$\sum_{k=1}^{m} z_{i,k} \geq 1\left(\frac{1}{2} + \delta\right)m - 1\left(\frac{1}{2} - \delta\right)m = 2\delta m$$

and since $\Delta(y_i, y_j) \geq \left(\frac{1}{2} - \epsilon\right)$ we get

$$< z_i, z_j >= \sum_{k=1}^{m} z_{ik} z_{jk} \leq \left(\frac{1}{2} + \epsilon\right)m - \left(\frac{1}{2} - \epsilon\right)m = 2\epsilon m \leq 2\delta^2 m$$

Define $w := \sum_{k=1}^{m} z_k$. By the preceeding two equation we get

$$\begin{aligned} < w, w > &= \textstyle\sum_{i=1}^{l} < z_i, z_i > + \sum_{i \neq j \in [l]} < z_i, z_j > \\ &\leq \textstyle\sum_{i=1}^{l} m + \sum_{i \neq j \in [l]} 2\delta^2 m \\ &= (lm + 2\delta^2 m l^2) \end{aligned}$$

and $(< w, w >= \sum_{k=1} m w_k^2)$

$$\sum_{k=1}^{m} w_k = \sum_{1 \leq i \leq l, 1 \leq k \leq m} z_{i,k} \geq 2\delta m l$$

Cauchy-Schwartz yields:

$$\sum_{k=1}^{m} w_k^2 \geq \frac{\left(\sum w_k\right)^2}{m} \geq 4\delta^2 l^2 m$$

Combiniing the two bounds we get

$$\begin{array}{rrcl} & 4\delta^2 l^2 m & \leq & < w, w > \leq (lm + 2\delta^2 m l^2) \\ \Rightarrow & 2\delta^2 l^2 m & \leq & lm \\ \Rightarrow & l & \leq & \frac{1}{2\delta^2} \end{array}$$

□

Thus there are $\frac{1}{2\epsilon}$ many codewords $\left(\frac{1}{2} - \sqrt{\epsilon}\right)$-close to a string, if the code is of distance $\left(\frac{1}{2} - \epsilon\right)$. These can be found efficiently in the case of RS code:

5.4.5 List Decoding of RS

Theorem 5.4.6. *(Sudan 1996) there exists a polynomial-time-algorithm that given*

$$\left\{(a_i, b_i) \in \mathbb{F}^2 \mid 1 \leq i \leq m\right\}$$

with $m > d$ returns the list of <u>*all*</u> *polynomials $G(x)$ with* $\deg(G) = d$ *s.t.*

$$\#\left\{i \in [m] \mid G(a_i) = b_i\right\} > 2\sqrt{dm}$$

Remark 5.4.7. *The RS-decoder "uniquely" found a G for which the number of fitting pairs (a_i, b_i) was $\frac{m+d}{2}$ while now the number of fitting pair is just $2\sqrt{dm}$.*

<u>Proof.</u> Find a nonzero $Q(x,y)$ with $\deg_x(Q) \leq \sqrt{dm}$ and $\deg_y(Q) \leq \sqrt{\frac{m}{d}}$ s.t. $Q(a_i, b_i) = 0$ for all $i \in [m]$ (view this as m linear equations in the coefficients of Q as unknowns, which are $\left(1 + \sqrt{dm}\right)\left(1 + \sqrt{\frac{m}{d}}\right) > m$ at number. So the system has a nonzero solution). Now factor $Q(x,y)$ completely (using any efficient polynomial-factoring algorithm). For all factors of the form $(y - P(x)) \mid Q(x,y)$, $\deg(P(x)) = d$ and $\#\left\{i \in [m] \mid P(a_i) = b_i\right\} > 2\sqrt{dm}$ output $P(x)$.

If a degree d polynomal $G(x)$ fits $t > 2\sqrt{dm}$ points (a_i, b_i) then consider $Q(x, G(x))$ has $\deg_x \leq \sqrt{dm} + d\sqrt{\frac{m}{d}} < t$ and at least t points a_i it is zero. This implies that $Q(x, G(x)) = 0$ and $y - G(x) \mid Q(x,y)$. □

5.4.6 Local List Decoding

Definition 5.4.8. *Let $E : \{0,1\}^n \to \{0,1\}^m$ be an error-correcting code and $\epsilon := \left(\frac{1}{2} - \rho\right)$ for $\rho \in \left(0, \frac{1}{2}\right)$. An algorithm Dis a* <u>*local list decoder for E handling ρ errors*</u> *if $\forall\ x \in \{0,1\}^n, y \in \{0,1\}^m$ s.t. $\Delta(E(x), y) \leq \rho$: $\exists i_0 \in \left\{1, \ldots, \mathrm{poly}\left(\frac{n}{\epsilon}\right)\right\}$ s.t. on input $\langle i_0, j,$ oracle f for $y\rangle$ s.t. D runs for for* $\mathrm{poly}\left(\log(m), \frac{1}{\epsilon}\right)$ *time and outputs x_j with probability $\geq \frac{2}{3}$.*

i_0 is called <u>*advice*</u>*. interpret the defintions as: D outputting the j-th bit of the i_0-th element in the list: $\left\{x \in \{0,1\}^m \mid \Delta(E(x), y) \leq \rho\right\}$.*

5.4.7 Local List Decoder for WH

Theorem 5.4.9 (Goldreich-Levin)**.** *Let $WH : \{0,1\}^n \to \{0,1\}^{2^n}$ and f be an oracle s.t. for some $x \in \{0,1\}^n$:*

$$\Pr_{z \in \{0,1\}^m}\left[f(z) = WH(x)_z\right] \geq \frac{1}{2} + \frac{\epsilon}{2}$$

Then the list of messages $\{x \mid \Delta(\hat{f}, WH(x)) \leq \frac{1}{2} - \frac{\epsilon}{2}\}$ *can be computed in* $\text{poly}\left(\frac{n}{\epsilon}\right)$ *time by a probabilistic algorithm.*

Proof. Fix $m = \frac{200n}{\epsilon}, k = \log(m+1)$. Randomly pick $s_1, \ldots, s_k \in \{0,1\}^n$ and $\sigma_1, \ldots, \sigma_k \in \{0,1\}$. Our hope is that $\forall i \in [k] : \sigma_i = x \odot s_i$. Creating a lot of pairwise independent strings. $\forall \phi \neq T \subseteq [k]$ define $s_T := \oplus_{ß \in T} s_i$ and $\rho_T := \oplus_{i \in T} \sigma_i$. Hope: $\forall T \subseteq k, \rho_t = x \odot s_T$. Compute for alle $j \in [n], x_j := \text{maj}_T\{\rho_T \oplus f(r_T + e_j)\}$. Output $x_1 \cdots x_n$.

Analysis: Let us look at a fixed $x \neq 0^n$ in the list.

$$\Pr[\forall i \in [k], \sigma_i = x \odot s_i] = 2^{-k} \geq \frac{1}{2m}$$

We now assume in the arguments that $\forall i : \sigma_i = x \odot s_i$. Define $\forall T \subseteq [k]$ a random variable

$$Z_T := \begin{cases} 1 & \text{if } \rho_T = x \odot r_T \text{ and } f(r_t + e_j) = x \odot (r_t + e_j) \\ 0 & \text{else} \end{cases}$$

we now can calculate the expectation

$$E[Z_T] = \Pr_{r_T}[f(r_T + e_j) = x \odot (r_T + e_j)] \geq \frac{1}{2} + \frac{\epsilon}{2}$$

by linearity of expectation-value we get

$$E\left[\sum_T Z_T\right] \geq (2^k - 1)(1 + \epsilon) = m(1 + \epsilon)$$

By pairwise independence of the Z_Ts we can use "Chebychev's inequality":

$$\Pr[|R - E(R)| > \delta \cdot \text{Var}(R)] \leq \frac{1}{\delta^2}$$

to calculate

$$\Pr\left[\sum_T Z_T \leq \frac{m}{2}\right] = \Pr\left[\left|\sum Z_T - E\left(\sum Z_T\right)\right| \geq \frac{m\epsilon}{2}\right] < \frac{1}{m\epsilon^2} < \frac{1}{10n}$$

using $\text{Var}\left(\sum_T Z_T\right) \geq \sqrt{\frac{m}{4}}$.

Exercise 5.4.10. *Proof the above.*

So we finally get

$$\Pr[\text{the last step gives } X] \geq \left(1 - \frac{1}{10}\right)$$

Thus

$$\Pr[x_1 \cdots x_n = X] \geq \frac{1}{2m}\left(1 - \frac{1}{10}\right) = \frac{\epsilon^2}{400n}\left(1 - \frac{1}{10}\right)$$

This means that repeating this algorithm for example $\frac{400n}{\epsilon^2}$ times we expect to get the full list (of messages, see above) with constant probability $\geq \frac{2}{3}$. □

5.4.8 Local list decoder for RM

Recall that RM maps $\binom{l+d}{d}$ evaluations of a d-degree, l-variate polynomial $P(\vec{x})$ to $|\mathbb{F}|^l$ evaluations. The task of local list decoder is to output $P(\vec{x})$ for a given $\vec{x} \in \mathbb{F}^l$ and oracle access to a corrupted codeword.

Theorem 5.4.11. *RM has a local list decoder handling* $\left(1 - 10\sqrt{\frac{d}{|\mathbb{F}|}}\right)$ *errors.*

Proof. We first give an algorithm D that computes $P(x)$ correctly on 0.9 of the xs with high probability in poly$(|F|, l) - time$ given $\langle f, i_0, x\rangle$ (f is the oracle and i_0 is an advice). The local decoder of RM can then be used to make D work for all $x \in \mathbb{F}^l$. □

5.4.9 Local List Decoder for RM

Input:

1. Oracle f s.t. $\Pr_{x\in\mathbb{F}^l}[f(x) = P(x)] > 10\sqrt{\frac{d}{|\mathbb{F}|}}$ for some l variate and d-degree unknown polynomial $P(x)$.
2. An $x \in \mathbb{F}^l$
3. An advice index $i_0 = (x_0, y_0) \in \mathbb{F}^l \times \mathbb{F}$ (this will help us to uniquely identify P)

Output: $y \in \mathbb{F}$ (goal: $y = P(x)$ with probability $\frac{2}{3}$ for 0.9 of the xs)
Algorithm:

1. Let L_{x,x_0} be a random quadratic curve passing through the points x and x_0 defined as: Pick a random $r \in \mathbb{F}$ and consider the degree 2 univariat $q : \mathbb{F} \to \mathbb{F}^l$ s.t. $q(0) = x$ and $q(r) = x_0$. Then $L_{x,x_0} := \{q(t) \mid t \in \mathbb{F}\}$.
2. Query f on all points of L_{x,x_0} to obtain $S := \{(t, f(q(t))) \mid t \in \mathbb{F}\} \subseteq \mathbb{F} \times \mathbb{F}$.
3. Run Sudan's RS list decoder to obtain $(2d)$-degree polynomials $g_1, \ldots g_k$ each fitting $\geq 8\sqrt{\frac{8}{|\mathbb{F}|}}$ pairs in S (recall $k \leq \sqrt{\frac{|\mathbb{F}|}{2d}}$).
4. If there exists an unique i with $g_i(r) = y_0$ then output $g_i(0)$ else goto step 1.

Analysis: We will now show that for all inputs f and x and for all polynomials P, if x_0 is chosen at random and $y_0 = P(x_0)$ then the output is $P(x)$ with probability ≥ 0.98. This implies: for all inputs f and suitable (x_0, y_0) there exists a unique P s.t. for 0.9 of the xs the algorithm outputs $P(x)$ with probability $\geq \frac{2}{3}$.

Note that for all inputs f and x and for all polynomials P where P and f agree on the random 2-degree-curve q on $\geq 8\sqrt{\frac{d}{|\mathbb{F}|}}$ points with probability ≥ 0.99.

Exercise 5.4.12. *Show the above.*

Thus step 3 gives a list containing $(q(t))P$ say g_1 with probability ≥ 0.99.

Next $\Pr_r\left[\exists\text{unique } i \in [k], g_i(r) = y_0\right] = 1 - \Pr_r\left[\exists i \neq 1, g_i(r) = y_0 = g_1(r)\right] \geq 1 - \sqrt{\frac{|\mathbb{F}|}{2d}\frac{2d}{|\mathbb{F}|}} = 1 - \sqrt{\frac{2d}{|\mathbb{F}|}} > 0.99$ thus with probability 0.99 there exists an uniqze i at step 4, $g_i(t) = P(q(t)) \Rightarrow g_i(0) = P(q(0)) = P(x)$. Thus with probability $\geq 0.99^2 > 0.98$ the algorithm outputs correctly.

5.4.10 Local List Decoder for Concatenated Code

Theorem 5.4.13. *If $E_1\, \{0,1\}^n \to \Sigma^m$ and $E_2 : \Sigma \to \{0,1\}^k$ are two codes that are locally list decodable then so is $E_2 \circ E_1 : \{0,1\}^n \to \{0,1\}^{km}$.*

Proof. As seen before in local decoding: Run the local list decoder of E_1 and answer its queries using the local list decoder of E_2. Assume that the local list decoder of E_1 takes an index in the set I_1 and can handle $1 - \epsilon_1$ errors. The local list decoder of E_2 takes an index in the set I_2 and handles $\left(\frac{1}{2} - \epsilon_2\right)$ errors. The local list decoder of $E_2 \circ E_1$ will take an index $(i_1, i_2) \in I_1 \times I_2$ and runs E_1's decoder with i_1 and E_2's decoder with i_2.

Claim: It can handle $(1 - \epsilon_1|I_2|)\left(\frac{1}{2} - \epsilon_2\right)$ errors.

Say y given with $\Delta(y, E_2 \circ E_1(x)) < (1 - \epsilon_1|I_2|)\left(\frac{1}{2} - \epsilon_2\right)$. So $\exists \leq (1 - \epsilon_1|I_2|)$ blocks in y each having $\geq \left(\frac{1}{2} - \epsilon_2\right)$ disagreements with the corresponding block of $E_2 \circ E_1(x)$. There exists more than $\epsilon_1|I_2|$ blocks in y each having $\geq \left(\frac{1}{2} + \epsilon_2\right)$ agreements with the correspondig block in $E_2 \circ E_1(x)$. Thus $\exists i_2 \in I_2$ s.t. witth i_2 output of local list decoder of E_2 agrees with $E_1(x)$ on ϵ_1 symbols. $\Rightarrow$ $\exists i_1 \in I_1$ s.t. with (i_1, i_2) and $\forall\ j \in [n]$ the combined local list decoder outputs x_j with high probability. □

5.5 Hardness Amplification

Now we apply local list decoding to transform worst-case to average-case hardness.

Theorem 5.5.1. *Let $f \in E := \mathrm{DTime}\left(2^{O(n)}\right)$ be s.t. $H_{wrs}(f) \geq S(n)$ for some $S : \mathbb{N} \to \mathbb{N}$ (say $S(n) \geq n^{1000}$). Then $\exists\ g \in E$ s.t. $H_{avg}(g) \geq S\left(\frac{n}{c}\right)^{\frac{1}{c}}$ for some constant $c > 0$.*

Proof. We treat $f|_{\{0,1\}^n}$ as a string $f' \in \{0,1\}^N$ for all n with $N := 2^n$. We encode f' using a suitable error-correcting code $E : \{0,1\}^N \to \{0,1\}^{N^c}$. g is defined to be the following function: $\{0,1\}^{cn} \to \{0,1\}$, $x \mapsto E(f')_x$ (which means nothing more than "$E(f')$ is the true-table of g"). For g to satisfy the theorem we need an error-correcting code E s.t.

1. $\forall x \in \{0,1\}^N$, $E(x)$ is computable in poly(N).
2. E has a local list decoder that runs in poly$(\log(N))$ time and handles $\left(\frac{1}{2} - \frac{1}{S(n)}\right)$ errors.

If $\exists$ circut D approximating g on $\geq \left(\frac{1}{2} + \frac{1}{S(n)}\right)$ inputs then view D as a corrupt oracle for the true codeword $E(f')$. Then the local list decoder of E can now use D and reconstruct bits in f' which basically means that it can compute f: a small circut solves f which is a contradiction to the worstcase-hardness of f. □

Theorem 5.5.2. *If $f \in E := \text{DTime}(2^{O(n)})$ s.t. $H_{wrs}(f) \geq S(n)$ for some $S : \mathbb{N} \to \mathbb{N}$ then $\exists g \in E$ s.t. $H_{avg}(g) \geq S\left(\frac{n}{c}\right)^{\frac{1}{c}}$ for some constant $c > 0$.*

Proof. Let $f' \in \{0,1\}^N$ be the true-table of $f|_{\{0,1\}^n}$ for $N := 2^n$. g is defined by $g' := E(f') \in \{0,1\}^{N^c}$ (where E can be $WH \circ RM$-Code). So g can be viewed as a function $\{0,1\}^{cn} \to \{0,1\}$. For g to satisfy the theorem statement, we need an error-correcting code E s.t.

1. $\forall x \in \{0,1\}^N$ it holds that $E(x)$ can be computed in poly(N) time.
2. E has a local list decoder, that runs in poly$(\log(N))$ time and handles $\frac{1}{2} - \frac{1}{S(n)}$ of the errors.

We achieve this by concatenating RM-Code with WH-Code.

Parameters for RM-Code: $|\mathbb{F}| = S(n)^{\frac{1}{100}}$, $d = S(n)^{\frac{1}{200}}, l = \frac{300\log(N)}{\log(S(n))}$.

Exercise 5.5.3. *Proof that $\binom{d+l}{l} > \left(\frac{d}{l}\right)^l > N$ assuming $N = 2^n \geq S(n) \geq n^{1000}$.*

So this RM-Code can be used to encode $f' \in \{0,1\}^N$. And since $|\mathbb{F}|^l < N^3$ this encoding can be done in poly(N) time. Distance of this RM-Code $\geq 1 - \frac{d}{|\mathbb{F}|} \geq 1 - \frac{1}{S(n)^{\frac{1}{200}}}$. On this RM-Code we will apply WH-Code, $WH : \mathbb{F} \to \{0,1\}^{|\mathbb{F}|}$. Its local list decoder runs in poly$(\log(|\mathbb{F}|))$ time and handles $\frac{1}{2} - \epsilon$ errors (for any $\epsilon > 0$). The concatenation $WH \circ RM$ code has a local list decoder, that runs in poly$(\log(N) \cdot \log(|\mathbb{F}|))$ = poly(n) time, handling $\left(1 - \sqrt{\frac{d}{|\mathbb{F}|}} \cdot \frac{\log|\mathbb{F}|}{\epsilon^2}\right)\left(\frac{1}{2} - \epsilon\right)$ errors. This is approximately $\left(1 - \frac{1}{S(n)^{\frac{1}{400}}} \frac{\log(S(n))}{\epsilon^2}\right) \cdot \left(\frac{1}{2} - \epsilon\right)$. We fix $\epsilon = S(n)^{-\frac{1}{801}}$. So the above product is $\geq \left(\frac{1}{2} - \frac{1}{S(n)^{\frac{1}{802}}}\right)$.

$f' \in \{0,1\}^N \overset{WH \circ RM}{\to} g' \in \{0,1\}^{N^3}$ so $g : \{0,1\}^{3n} \to \{0,1\}$. Suppos that G is a circuit of size $\leq S(n)^{\frac{1}{802}}$ that computes $g|_{\{0,1\}^{3n}}$ on more than $\left(\frac{1}{2} + \frac{1}{S(n)^{\frac{1}{802}}}\right)$ fraction of inputs. This implies that we can use the local list decoder of $WH \circ RM$ using G as an oracle, to get a circuit F of size $\left(\text{poly}(n) + S(n)^{\frac{1}{802}}\right) < S(n)$ computing $f'|x = f(x)$ $\forall x \in \{0,1\}^n$. This contradicts $H_{wrs}(f) \geq S(n)$.

So $H_{avg}(g) \geq S\left(\frac{n}{3}\right)^{\frac{1}{802}} > S\left(\frac{n}{802}\right)^{\frac{1}{802}}$. □

Chapter 6

Ausblick

6.1 Extractors

We constructed expanders (inconditionally) in chapter 3 and PRGs in 4 (conditionall), another related pseudorandom object is an extractor. They "extract randomness" from a "weakly random" source.

Definition 6.1.1. *The min entropy of a random variable X, denoted by $H_\infty(X)$, is the larges $k \in \mathbb{R}$ s.t. $\Pr[X = x] \le 2^{-k}$ for all $x \in range(X)$.*

If X is a distribution on $\{0,1\}^n$ with $H_\infty(X) \ge k$ then it is called an (n,k)-source. If $k < n$ we call it a weak random source.

Exercise 6.1.2.

1. *If X is a distribution on $\{0,1\}^n$ then $H_\infty(X) \le n$.*
2. *$H_\infty(X) = n$ iff $X \approx U_n$ (the uniform distribution).*
3. *Generally a probabilistic algorithm that uses k random bits requires access to a prob. distribution X with $H_\infty(X) \ge k$.*
4. *If X is the uniform distribution over $S \subset \{0,1\}^n$ of size 2^k then $H_\infty(X) = k$ (this distribution "essentially captures" any distribution with min entropy k.*

Definition 6.1.3. *If X, Y are probability-distributions over Ω then the statistical distance*

$$\Delta(X,Y) := \max_{f:\Omega\to\{0,1\}} (E[f(x)] - E[f(y)])$$

Exercise 6.1.4. *It is also $\frac{1}{2}|\vec{x} - \vec{y}|_1$ where $\vec{x}, \vec{y} \in \mathbb{R}^{|\Omega|}$ describe X and Y respectively.*

Definition 6.1.5. *We call two distributions X and Y to be ϵ-close if*

$$\Delta(X,Y) \le \epsilon$$

Extractors transform an (n,k)-source into an almost uniform distribution, formally:

Definition 6.1.6. *A function*

$$\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

is called a <u>*(k,ϵ)-extractor*</u> *if for any (n,k)-source X it holds that*

$$\Delta(\mathrm{Ext}(X,U_d),U_m) \le \epsilon$$

Remark 6.1.7.

1. *$\{0,1\}^d$ is called* <u>*seed of the extractor function*</u> *and one of the aims is to achieve $m = k$ using d as small as possible.*

2. <u>*Existence:*</u> *Probabilistic method proves for $d = \log(n) + 2\log\left(\frac{1}{\epsilon}\right) + O(1)$ and $m = k$.*

3. <u>*Explicit constructions:*</u> *using expanders or PRG ideas.*