P vs. NP: Geometric Approaches

Spring 2016

Contents

1 Relativization Barrier **7**
- 1.1 Diagonalization . . . 7
- 1.2 Question on Baker/Gill/Solovay . . . 9

2 Fourier Bases **11**
- 2.1 Fourier basis . . . 11
- 2.2 Fourier transform . . . 12

3 Ladner's Theorem **13**
- 3.1 Ladner's Theorem . . . 13
  - 3.1.1 Hierarchy . . . 14
- 3.2 Further Discussion . . . 15
- 3.3 Randomized Approximation Hierarchy . . . 15
- 3.4 Further Discussion . . . 18

4 Interactive Proof Systems **19**
- 4.1 Interactive Protocols . . . 19
- 4.2 Arithmetization . . . 21
- 4.3 Degree Reduction . . . 23
- 4.4 IP = PSPACE . . . 23

5 Probabilistically Checkable Proofs **27**
- 5.1 Introduction . . . 27
- 5.2 An Indirect Proof of PCP Theorem . . . 29
- 5.3 Spectral Properties of d-regular Graphs . . . 30
- 5.4 Expander Graphs . . . 32
- 5.5 Constraint Satisfaction Problems . . . 33
- 5.6 Dinur's Proof of the Inapproximability of GAP3SAT$_{1,s}$ . . . 33

6 Dinur's Proof of PCP Theorem **35**
- 6.1 The PCP Theorem by Gap Amplification . . . 35
  - 6.1.1 Sparsification (Degree Reduction) . . . 37
  - 6.1.2 Expanderizing . . . 38
  - 6.1.3 Amplification . . . 39
  - 6.1.4 Alphabet Reduction . . . 41

6.2 Appendix . . . 43
6.2.1 Background . . . 43
6.2.2 Random Walks . . . 44

7 Strict NP and the Approximibility of NP-Complete Problems 45
7.1 Circuits and Logic . . . 45
7.2 Strict NP . . . 46
7.3 L-Reduction . . . 48

8 Unique Games Conjecture 51
8.1 Introduction . . . 51
8.2 Integrality Gaps and Approximation . . . 51
8.2.1 Goemans and Williamson: Bound on SDP/ALG . . . 53
8.2.2 Feige and Schechtman: Tightness of SDP/OPT . . . 53
8.2.3 Karloff: Tightness of OPT/ALG . . . 55
8.3 The Unique Games Conjecture . . . 55
8.3.1 Raghavendra's Result: Optimality of SDP . . . 57
8.4 An Unconditional Result: Metric Space Embeddings . . . 58
8.4.1 Metric Spaces . . . 58
8.4.2 SparsestCut and BalancedSeparator . . . 59
8.4.3 Refutation of $(l_2^2, l_1, O(1))$ . . . 60
8.5 Refuting UGC . . . 64
8.5.1 Sums of Squares . . . 65
8.5.2 Relation to UGC . . . 66

9 Inner PCP 69
9.1 Introduction . . . 69
9.2 Pre-processing Φ before Modularization . . . 70
9.3 Modularization . . . 71
9.3.1 Construction of the Inner PCP(6-query AT) . . . 72
9.3.2 Conversion from the 6-query AT to a 2-query AT . . . 75
9.3.3 2-query AT is stronger PCP . . . 76
9.4 Composition . . . 76

10 Circuit Size Lower Bounds 77
10.1 Introduction . . . 77
10.1.1 Circuit Review, Formalism, and Notation . . . 77
10.2 Hastad's Lemma . . . 78
10.3 Results Following from Hastad's Lemma . . . 82
10.3.1 Lower bound for Depth of PARITY Circuits . . . 82
10.4 Razborov-Smolensky Theorem . . . 83
10.4.1 Definitions . . . 83
10.4.2 Independence of mod_2 Circuits to mod_3 Circuits . . . 83

11 Hardness vs. Randomness **87**
11.1 Boolean function analysis . . . 87
11.2 Bound on high coefficients . . . 89
11.3 Learning Constant Depth Circuits . . . 92
11.4 Pseudorandom bits . . . 93

12 Communication Complexity **95**
12.1 Introduction . . . 95
12.2 Communication Complexity . . . 96
12.2.1 Deterministic Communication Complexity . . . 96
12.2.2 Probabilistic Communication Complexity . . . 97
12.3 Threshold Circuit Complexity Lower Bounds . . . 100
12.4 Margin Lower Bounds . . . 101
12.5 Dvoretzky's Theorem . . . 102

13 Proof Barriers **105**
13.1 Relativization barrier . . . 105
13.2 Natural proofs barrier . . . 105
13.2.1 Notation and definitions . . . 105
13.2.2 Natural combinatorial properties and its generalization . . . 106
13.2.3 Inherent limitations of natural proofs (Natural proof barrier) . . . 107
13.2.4 Natural proof barrier for the discrete logarithm problem . . . 108
13.2.5 Naturalization . . . 109
13.3 Algebrizition (Algebraic relativization) barrier . . . 110
13.3.1 Notation and definitions . . . 110
13.3.2 Algebrization example . . . 111
13.3.3 Non-algebrizing techniques needed for P vs. NP (Algebrization barrier) 112
13.3.4 The integers case . . . 112
13.3.5 k-Algebrization . . . 113
13.3.6 Open problems . . . 113

14 Hardness Amplification **115**
14.1 Introduction . . . 115
14.2 Other Directions . . . 119

15 Unification of Pseudorandom Objects **121**
15.1 The Framework . . . 121
15.2 List-Decodable Codes . . . 122
15.3 Averaging Samplers . . . 123
15.4 Expander Graphs . . . 124
15.5 Randomness Extractors . . . 125
15.6 Hardness Amplifiers . . . 127
15.7 Pseudorandom Generators . . . 129

Chapter 1

Relativization Barrier

Lecture(s): 2
Date(s): 1/7
Lecturer: John Corring
Scribe: Alan Kuhnle

1.1 Diagonalization

Technique dates from Cantor's diagonalization method of proving the real numbers are uncountable. Also employed to show the Halting problem is undecideable.

More examples, in the proof of the Time and Space hierarchy theorems.

Theorem 1 (Time, space hierarchy theorems) *Let f, g be time-constructible functions s.t. $f(n)\log n = o(g(n))$. Then*

$$DTIME(f) \subsetneq DTIME(g)$$

Let f, g be space-constructible functions s.t. $f(n) = o(g(n))$. Then

$$SPACE(f) \subsetneq SPACE(g)$$

Proof: [Proof (Space)] We need $L \in DS(O(f))$ not in $DS(o(f))$.

$$L = \{(\langle M\rangle, x) : M \text{ does not accept } (\langle M\rangle, x) \text{ in } \leq f(|(\langle M\rangle, x)| \text{ space. } \}$$

L differs from $L_M \in DS(o(f))$ at $(\langle M\rangle, x)$. ■

Notice in this proof, we could replace all machines M by oracle machines M^A for some oracle A. Thus, in some sense a diagonalization proof "relativizes" from regular machines to oracle machines.

Definition 2 (Oracle machines) *An* oracle *Turing machine is a TM M that has a special read/write tape we call M's* oracle tape *and three special states $q_{query}, q_{yes}, q_{no}$. To execute M, we specify in addition to the input a language O that is used as the* oracle *for M. M queries the oracle by moving into state q_{query} after writing z on the oracle tape. In one computational step M then moves to q_{yes} if $z \in O$ and q_{no} if $z \notin O$.*

Let P^O be the set of languages decidable in polynomial time by an oracle TM with oracle O, NP^O be defined analagously. Then

Claim 3 *(i)* $\overline{SAT} \in P^{SAT}$

(ii) $O \in P$ *implies* $P^O \subset P$.

(iii) Let

$$EXPCOM = \{\langle M, x, 1^n\rangle : M(x) = 1 \text{ in } O(2^n)\}.$$

Then $P^{EXPCOM} = NP^{EXPCOM} = EXP$.

Proof:

(iii) Any exponential computation can be performed in one call with an oracle to $EXPCOM$, so $EXP \subset P^{EXPCOM}$. Furthermore, any nondeterministic Turing machine M with an oracle to $EXPCOM$ may be simulated in exponential time. So

$$EXP \subseteq P^{EXPCOM} \subseteq NP^{EXPCOM} \subseteq EXP.$$

■

Theorem 4 (Baker, Gill, Solovay) *There exist oracles* A, B *such that* $P^A = NP^A$ *and* $P^B \neq NP^B$.

Proof: We need an oracle B such that $P^B \neq NP^B$.

Let $U_B = \{1^n : B \text{ accepts an element in } \{0,1\}^n$. We will construct B such that $U_B \in NP^B$, but $U_B \notin P^B$. Clearly $U_B \in NP^B$, as we can nondeterministically choose all strings of length n and query B to determine if $1^n \in U_B$. For each i, let M_i be the oracle TM represented by the binary expansion of i.

B will be constructed in stages. At stage i, we choose B so that M_i, when run for $2^n/10$ steps does not have $L_{M_i} = U_B$, where n is chosen larger than the length of any string in B before stage i.

Add srings to B as follows at stage i: run M_i on input 1^n for $2^n/10$ steps. Whenever M_i makes an oracle query about a string whose status has been determined, answer consistently, otherwise, declare the string to not be in B.

If M_i halts and accepts, declare all strings of length n are not in B. Otherwise, choose a string of length n that M_i has not queried, and add it to B. ■

For more on relativization see

- Fortnow, Lance. "The role of relativization in complexity theory." Bulletin of the EATCS 52 (1994): 229-243.
- Arora, Sanjeev, and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.
- T. P. Baker, J. Gill, R. Solovay. Relativizations of the P =? NP Question. SIAM Journal on Computing, 4(4): 431-442 (1975)

1.2 Question on Baker/Gill/Solovay

This question was posted on Piazza after the lecture:

Why doesn't the proof $U_B \notin P^B$ not go through if you replace P^B by NP^B?

Answer: In the proof, we rely on the fact that M_i cannot query all strings of length n to the oracle, since it runs in time $2^n/10$ and there are 2^n such strings. However, if M_i were allowed to be nondeterministic, it could query all such strings in a single step.

Chapter 2

Fourier Bases

Lecture(s): 3
Date(s): 1/7
Lecturer: Meera Sitharam
Scribe: Alan Kuhnle Let X be a vector space of multilinear polynomials of degree at most some d. That is,

$$X = \{\phi : \{0,1\}^n \to \mathbf{R} : \phi \text{ polynomial of degree at most } d\}$$

,

$$\phi(x) = \sum_{\alpha \in \{0,1\}^n} a_\alpha x^\alpha,$$

where $x^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$, and $\sum_i \alpha_i \leq d$.

A common technique to show two complexity classes C, and D, with $C \subseteq D$, are not equal, is to show, for some $\epsilon > 0$ the following statement is true for $B = C$ but not for $B = D$:

$$\forall f \in B \exists \phi \in X(distance(\phi, f) < \epsilon).$$

The functions in the classes C and D are generally assumed to be Boolean, i.e., from $\{0,1\}^n \to \{0,1\}$, but in fact, the structure imposed on C, D and space X namely, the domain (vertices of the hypercube as a subset of $\mathbf{R}^n$, or simply GF_2^n) range ($\mathbf{R}$ or GF_2), the chosen norm, epsilon etc. are ALL a matter of choice. They can be chosen in any way that facilitate the only goal, namely to show that C differs from D.

2.1 Fourier basis

We can identify $2 = \{0,1\}$. Let $\alpha \in 2^n$. Define $\chi_\alpha : 2^n \to \{-1,1\} \subset \mathbf{R}$ by

$$\chi_\alpha(\beta) = (-1)^{\langle \alpha, \beta \rangle_2}.$$

Here, $\langle \alpha, \beta \rangle_2 = \sum_{i=1}^n \alpha(i)\beta(i) \mod 2$. We define an inner product on X by

$$[f, g] = \frac{1}{2^n} \sum_{\beta \in 2^n} f(\beta)g(\beta).$$

Now, it is an exercise to check that if $\alpha_i, \alpha_j \in 2^n$,

$$[\chi_{\alpha_i}, \chi_{\alpha_j}] = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

Remark: Any boolean function can be represented by multilnear polynomials of total degree at most n (where n is the number of variables). In a later lecture, this fact will be shown by reducing this Fourier basis to multilinear monomials.

2.2 Fourier transform

Let $\Omega_n = \{f : 2^n \to \mathbf{R}\}$. We wish to define the Fourier transform of $f \in \Omega_n$,

$$f(\beta) = \sum_{\alpha} \hat{f}(\alpha) x_\alpha(\beta),$$

where $\hat{f} : 2^n \to \mathbf{R}$ by

$$\hat{f}(\alpha) = [f, \chi_\alpha] = \frac{1}{2^n} \sum_{\beta} f(\beta)\chi_\alpha(\beta).$$

Open question: for what functions $2^n \to \mathbf{R}$ do the Fourier coefficients have constant modulus?

Chapter 3

Ladner's Theorem

Lecture(s): 4–6
Date(s): 1/12, 1/14
Lecturer: Alan Kuhnle
Scribe: John Corring

3.1 Ladner's Theorem

The goal of this lecture was to prove **Ladner's Theorem** [39], a statement about the existence of problems strictly between P and NP. Ladner's Theorem is conditional on the inequality of P and NP, and as one might suspect uses this distinction to construct a diagonalizing machine. Let

$$SAT_H = \{\psi 01^{n^{H(n)}} | \psi \in SAT \quad \wedge \quad n = |\psi|\}.$$

Claim 5 *1. If $\exists M \in \mathbb{N} : H(i) < M \quad \forall i \in \mathbb{N}$ then $SAT_H \in NP-complete$. I.E. if H is bounded then SAT_H is $NP-$complete.*

2. Assume $P \neq NP$. If H is unbounded ($\forall k \in \mathbb{N}, \exists m \in \mathbb{N} : H(m) > k$) then $SAT_H \notin NP-$complete.

Proof: (1): For the first claim, note that if H is bounded then $H(n) \in O(1)$ and so we can add in the final $1^{|\psi|^{O(1)}}$ and test acceptance for $\psi 01^{n^{H(n)}}$ in SAT_H.

(2): Let $H(n) \uparrow \infty$ as $n \uparrow \infty$. Assume SAT_H is $NP-$complete. Then there exists an $f \in P$ reducing $SAT \leq SAT_H$. Now, for ϕ a circuit being tested for SAT

$$f : SAT \to SAT_H \tag{3.1}$$

$$\phi \to \psi 01^{n^{|\psi|}}, \tag{3.2}$$

and as $|f(\phi)| < O(|\phi|^k)$ there exists an n large enough so that $|\phi| > n \implies |psi| < |phi|$, as otherwise $|\psi 01^{n^{|\psi|}}| \in O(n^n)$. In fact, $|\psi| = o(|\phi|)$, as otherwise the reducing transformation is non-polynomial in $|\phi|$. Therefore, if we had a polynomial time reduction of SAT to SAT_H then we would also have a polynomial time algorithm for solving SAT. ■

To use SAT_H to show the existence of a language in $NP \cap P^C \cap NP - complete^C$ we need to first define and H and then show that it is time-computable. Let

$$H(n) = \begin{cases} \text{least } i < \log\log n : M_i \text{ decides } x \in SAT_H \text{ within } i|x|^i \text{ steps} & \forall |x| < \log n \\ \log\log n \text{ if no such } i \text{ exists} \end{cases} \quad (3.3)$$

Claim 6 *H is a well-defined polynomial time computable function.*

Proof: Choose an enumeration of Turing machines M_i and proceed inductively: By the definition H only requires checking membership of strings up to length $\log n$, and since this cannot interfere with membership of strings of length greater than n (i.e. strings with $1^{n^{H(n)}}$ at the end) in SAT_H it follows that condition (1) does not pose a contradiction with SAT_H and H is thus well defined.

To compute $H(n)$ we simply compute $H(k)$ for k up to $\log(n)$ and simulate at most $\log\log n$ machines for every input up to $\log\log n$ length for at most $\log\log n(\log n)^{\log\log n}$ time. The product of these factors is polynomial in n. ■

Now we have a well defined language SAT_H that references a well defined polynomial-time computable function H. Ladner's Theorem consists of a diagonalization out of P that accesses condition (1) in the definition of H, and a proof that $H(n) \uparrow \infty$ whereby Claim 1.ii above kicks SAT_H out of $NP-$complete.

Theorem 7 (Ladner's Theorem) *If $P \neq NP$ then there is a language in $NP \cap P^C \cap NP - complete^C$.*

Proof: $\underline{SAT_H \notin P.}$ Suppose not.If M_i is a turing machine solving SAT_H in $O(n^k)$ time then for $n > 2^{2^i}$, $H(n) \leq i$. Thus for all sufficiently large input lengths SAT_H is a SAT padded with a polynomial number of 1's. So we can reduce $SAT \leq SAT_H$ by $\psi \to \psi 01^{|\psi|^i}$ which runs in $O((|\psi| + |\psi|^i)^k)$ time. $\Rightarrow\Leftarrow$

$\underline{SAT_H \notin NP - complete.}$ Since $SAT_H \notin P$ we know that there is an input x such that M_i is wrong within $i|x|^i$ time on x. So $\forall n > 2^{|x|}, H(n) > i$. So $H = i$ for at most a finite number of inputs, say at most k_i inputs. Let $m \in \mathbb{N}$. Then let $n > mk_i$ for $i = 1 \ldots m-1$. Then $H(n) > m$, so $H(n) \uparrow \infty$ as $n \uparrow \infty$.

$\underline{SAT_H \in NP.}$ Let the witness for SAT_H be the satisfying assignment to the circuit and the length of the circuit. This has length $O(n) + O(\log(n))$. Then we can check truth in $O(|\psi|)$ time and then ensure that the final bits agree with the definition of SAT_H. ■

3.1.1 Hierarchy

The proof of Ladner's Theorem suggests an ingredient in a diagonalization proof: padding a "hard problem" with a sufficiently slowly growing set of 1's. Suppose we try to repeat this ingredient with SAT replaced by SAT_H in the argument. What happens? Now, the language looks like

$$SAT_K = \{\psi 01^{K(n)} | \psi \in SAT_H\}.$$

Now if we define K similar to H but with SAT_H in place of SAT in the definition of SAT_K, then we see that

Claim 8 *$SAT_K <^P SAT_H$ and $SAT_K \notin P$.*

Proof: It is an easy exercise to check that $SAT_K \notin P$: follow the exact same steps as in the proof of Ladner's Theorem. The reduction can be done by reading backwards to find the first 0, then querying SAT_H once to check whether the prefix is in SAT_H. Finally, we use the polynomial growth of K to check the appropriate size of the suffix of 1's. The failure of the reverse reduction follows from the diagonalization argument. ■

3.2 Further Discussion

The problems above suffice for the existence of problems that are (conditional on $P \neq NP$) strictly between P and NP. However, the problems in Ladner's Hierarchy argument are not "natural":they are specifically constructed to diagonalize P. One can ask whether or not there are "natural" problems—problems that do not self-reference or reference other problems for diagonalization purposes—that fall in between P and NP. Of course, if we had an unconditional proof of the existence of such a problem then $P \neq NP$. However, there are a number of problems "in limbo" between P and NP. That is, problems that are in NP but not known to be in P or $NP-$complete. Problems like this include Graph Isomorphism, Factoring, and more. For more on this see

http://cstheory.stackexchange.com/questions/79/problems-between-p-and-npc.

Finally, note that we have executed Impaliazzo's (unpublished) proof of Ladner's theorem that uses a padded SAT language to diagonalize. Note that in Ladner's original paper [39] he drops languages from SAT to define SAT_H in a diagonalizing manner as opposed to padding SAT. This is similar to diagonalization used below, in the proof of the Randomized Approximation Hierarchy Theorem.

3.3 Randomized Approximation Hierarchy

Similar to the hierarchy resulting from Ladner's Theorem, we obtain a randomized approximation hierarchy for # P by a similar diagonalization. This results comes in the form of the following theorem by Bordewich [15]

Theorem 9 *Let $\pi \in \#P$ be a problem such that there is no FPRAS for π. Then there is a $\pi_A \in \#P$ such that*

1. *There is no FPRAS for π_A*
2. *$\pi_A <_{AP} \pi$.*

Note the strict reduction.

Definition 10 *The counting language $\pi : \{0,1\}^* \to \mathbb{Z}^+$ is in $\#P$ if $\exists M \in P$ and polynomial p: $\forall x \in \{0,1\}^*$,*

$$\pi(x) = |\{\pi(x) = y^{p(|x|)} : M(x,y) = 1\}|.$$

In this setting we can think of M as the verifier for an NP problem. The goal of this lecture is to establish a hierarchy for problems of the following variety

Definition 11 *A Randomized Approximation Scheme (RAS) for $\pi \in \#P$ is an algorithm which, given $(x, y) \in \{0,1\}^* \times 1^*$, returns $\left[(1-y^{-1})\pi(x), (1+y^{-1})\pi(x)\right]$ (a good value) with probability at least $3/4$. A Fully Polynomial RAS (FPRAS) solves a RAS in polynomial time in $|x|, |y|$. $\pi_1 \leq_{AP} \pi_2$ if there is an i such that M_i^O, given any oracle O for π_2, is a polynomial time oracle machine acting as a FPRAS machine for π_1. For a notational convenience note that $<$ means $\leq$ and not $\geq$ from a reduction standpoint.*

Note that an RAS Oracle may in general view the computation path for its caller. Two restrictions to RAS Oracles follow.

Definition 12 *A binary RAS oracle for each (x, y) returns a good value z with probability exactly $3/4$ and a bad value z' otherwise. A restricted RAS oracle is an RAS oracle whose response depends only on (x, y).*

Claim 13 $\pi_1 \leq_{AP} \pi_2 \Leftrightarrow \pi_1 \leq_{bAP} \pi_2$.

Proof: $\Rightarrow$: is trivial since every binary RAS oracle is also an RAS oracle. By $\pi_1 \leq_{AP} \pi_2$ every such oracle (for π_2) works for π_1.

$\Leftarrow$: Towards the contrapositive assume that $\pi_1 \not\leq_{AP} \pi_2$. Then $\forall i \exists O, (x', y')$ s.t. $M_i^O(x', y')$ returns good values for π_1 less than $3/4$ of the time. Now construct a new oracle O'—over all calls in M_i^O to O, choose the one value that minimizes the probability of returning a good value for M_i^O: choose the minimizing good value with probability $3/4$ and the minimizing bad value with prob $1/4$. Now O' is a bAP oracle for π_2. But clearlyn since the probability of returning a good value has strictly decreased, $M_i^{O'}$ does not reduce π_1 to π_2. So there is a bAP RAS oracle O' for π_2 that fails the universal condition for reduction and so $\pi_1 \not\leq_{bAP} \pi_2$. ■

Note before we proceed that one can use a listing of PTMs and PTOMs such that M_i runs in $O(n^i)$ for each i—where i encodes the index to machine M_i. We will use this below. Now we can begin to prove Theorem 9.

Proof: Let π be a problem such that there is no FPRAS for π. Let M be a fixed NDTM such that the witnesses have fixed size $|x|^S$ and the runtime is bounded by $|x|^T$ on all inputs x. Now we define a new problem

$$pi_a(x) = \begin{cases} 0 & \text{if } f(|x|) \text{ is even} \\ \pi(x) & \text{o/w.} \end{cases} \tag{3.4}$$

Now, provided that f is polynomially computable it follows that $\pi_A \in \#P$ as $\pi \in \#P$ and π_a involves solving π or returning zero, dependent upon a polynomial time computation $(f(|x|))$. Furthermore, the reduction $\pi_A \leq_{AP} \pi$ follows directly from the definition of π_A. This (contingent upon the polynomial computability of f) shows item 1 of the Theorem. To proceed, we need to define f.

$$f(n+1) = \begin{cases} n & \text{if } n \leq 2 & i \\ f(n) & (\log\log n)^{Tf(n)} \geq \log n & ii \\ f(n) & \text{if } f(n) \text{ is even and Condition 1 fails} & iii \\ f(n)+1 & \text{if } f(n) \text{ is even and Condition 1 holds} & iv \\ f(n) & \text{if } f(n) \text{ is odd and Condition 2 fails} & v \\ f(n)+1 & \text{if } f(n) \text{ is odd and Condition 2 holds} & vi \end{cases} \tag{3.5}$$

where Condition 1 for input n is defined as

$$(1) \Leftrightarrow \exists (x,y) \in \{0,1\}^* \times 1^* : |(x,y)| < \log\log n : \quad M_{\frac{f(n-1)}{2}}(x,y) \text{ is good for } \pi_A \text{ with probability } < 3/4, \tag{3.6}$$

and Condition 2 for input n is defined as

$$(2) \Leftrightarrow \begin{array}{l} \exists (x,y) \in \{0,1\}^* \times 1^* \wedge O' \text{ an RAS oracle for } \pi_A \text{ s. t. for some } |(x,y)| < \log\log n \\ \quad M^{O'}_{\frac{f(n-1)}{2}-1}(x,y) \text{ is good for } \pi \text{ with probability } < 3/4. \end{array} \tag{3.7}$$

Now we show the polynomial computability of f. $n \leq 2$ is $O(1)$ computable and returning n is $O(n)$, so i is polytime. Checking $(\log\log n)^{Tf(n)} > \log n$ is $Poly(\log\log n, f(n))$ as exponentiation is polynomial time in both inputs, and computing $\log n$ is also polynomial. Clearly, inequality is polynomial time in the total input, so ii is polynomial time. For $iii - vi$ we just need to check the last bit for evenness and compute the truth value of Condition 1 or 2.

For Condition (1) we must check $2^{\log\log n}$ inputs, but for each input x the size is much less than n so we already know $f(|x|)$. Thus we can compute $\pi_A(x)$ by computing $f(|x|)$ and checking up to $2^{\log\log n^S}$ possible witness for π, each in time polynomial in $\log\log n$. To determine the probability that $M_i(x,y)$ returns a good value we simply simulate it by noting that each time that M_i branches (at most $|(x,y)|^i$ times) and following each branch and computing the final probability: all (including calls to π) in $2^{\log\log n^{iT}} \in O(\log(n))$ time.

For Condition (2) we must check over all possible binary oracles for π reducing to π_A. As $\pi(x) \leq 2^{|x|^S}$ we need to check at most this many witnesses against any given oracle. To distinguish them, we note that the equivalence of binary AP and AP reductions allows to fix the Binomial distribution at each input— so we have at most $2^{2|x'|^S}$ oracles to check— where x' is the size of the oracles input. Recall that the number of oracle calls is bounded by $(\log\log n)^i$ and the size of their inputs is also bounded by $(\log\log n)^i$. Thus we have a $O(2^{\log\log n^{i(2S+1)}})$ time bound on verifying Condition 2 by checking all inputs to $M_i(\cdot,\cdot)$.

So f is polynomially computable.

Now assume that some M_i is the FPRAS for π_A. Then by Condition (2) f is constant above $2i$ and so f is bounded, and $f(n+1) = f(n)$ for n above $2i$. If $\pi \leq_{AP} \pi_A$ then there is an M_j that reduces π to π_A and similarly f is bounded by $2j+1$, $f(n+1) = f(n)$ above $2j+1$. Finally, if f is bounded then we get an FPRAS oracle for π: If f is ultimately even then $\pi_A \neq \pi$ for at most finitely many inputs and so use π_A for π. This contradicts the hypothesis that π does not have a FPRAS. If f is ultimately odd then $\pi_a = 0$ above $2j+1$

and $\pi \leq \pi_A$. This means that we can hard code π_A into an FPRAS for π in constant time, again a contradiction with the hypothesis that π does not have a FPRAS. ■

Finally, we record a Corollary to Theorem 9 that provides a hierarchy of RAS problems in $\#P$. This Corollary follows by repeating the above argument with π_{A_i} in place of π, to obtain $\pi_{A_{i+1}}$

Corollary 14 *If $NP \neq RP$ then there is a hierarchy of problem $\pi_{A_1}, \pi_{A_2}, \ldots \in \#P$ such that*

1. *π_{A_i} does not have an FPRAS*
2. *$\#SAT$ is not $AP-$reducible to π_{A_i}*
3. *For all $i, j : i < j \quad \pi_{A_j} < \pi_{A_i}$*

3.4 Further Discussion

By now, the use of the diagonalization construction for exhibiting elements in the set difference (perhaps intersect the larger set) should be relatively familiar. As mentioned in these lecture notes and lecture notes 2-3, it is a classical and useful method. For a further discussion on diagonalization in the abstract, see [52]

Chapter 4

Interactive Proof Systems

Lecture(s): 7–9
Date(s): 1/19, 1/21
Lecturer: Meera Sitharam
Scribe: Jeremy Youngquist

4.1 Interactive Protocols

This section introduces a powerful type of machine known as Interactive Protocols. We will show that the class of languages with an IP machine is equal to the class PSPACE.

Definition 15 (Interactive Protocol) *An* interactive protocol *consists of a* prover *and a* verifier. *The prover has unbounded time complexity while the verifier is polynomial time, typically in P, RP, or BPP. When given a problem, the verifier and the prover have a polynomial length conversation and the verifier accepts or rejects the input.*

If there exists a prover such that, given an $x \in L$, the verifier accepts with probability at least δ for some constant δ with $0 < \delta \leq 1$, then we say that the interactive protocol is complete. *If $\delta = 1$, then we say the problem is in* 1-sided error IP.

If for all provers such that, given an $x \notin L$, the verifier rejects with probability at most $0 \leq \delta - \epsilon < 1$ where ϵ is some constant and δ is as above, then we say that the interactive protocol is sound.

The class of problems solvable by an interactive protocol is called *IP*.

Definition 16 (The Class IP) *We say that $x \in IP$ if there exists a BPP verifier V, a prover P of arbitrary time complexity, random bits b_i generated by V, and bits p_i generated by P such that*

$$\exists p_1, \ldots, p_n \forall^* b_1, \ldots, b_n \exists p_{n+1} \ldots, p_m \forall^* b_{n+1}, \ldots, b_m \exists \ldots V(x, b_1, \ldots, p_1, \ldots)$$

where there are polynomially many quantifiers, $V(x, b_1, \ldots, p_1, \ldots)$ means that V accepted the proof, and $\forall^*$ denotes at least 2/3 of the random bits over the probability space of random bits satisfy the equation.

Similarly, $x \notin IP$ if for all polynomial verifiers V there exists a prover P such that

$$\forall p_1, \ldots, p_n \exists^* b_1, \ldots, b_n \forall p_{n+1} \ldots, p_m \exists^* b_{n+1}, \ldots, b_m \forall \ldots V(x, b_1, \ldots, p_1, \ldots)$$

where $\exists^*$ indicates that at most 1/3 of the random bits satisfy the equation.

We can represent the relationship between P, NP, BPP, and IP by the following diagram, where an arrow indicates that the class pointed to is verified by the class at which the arrow originates, a vertical arrow indicates adding interaction, and horizontal arrows indicate adding randomization:

$$\begin{array}{ccc} \text{NP} & \longrightarrow & \text{IP} \\ \uparrow & & \uparrow \\ \text{P} & \longrightarrow & \text{BPP} \end{array}$$

Definition 17 (1-sided error IP) *A language is in* 1-sided error IP *if there is an interactive protocol with completeness of $\delta = 1$ that decides it.*

Theorem 18 *Graph Non-Isomorphism (GNI) is in 1-sided error IP.*

Proof: Given an input of two graphs (G_1, G_2), we want to find an interactive protocol that accepts with probability 1 if G_1 and G_2 are not isomorphic and rejects with some constant probability if they are. We begin by having our verifier generate a random permutation σ and a random $j \in \{1, 2\}$. It then sends $\sigma(G_j)$ to the prover which returns some $j' \in \{1, 2\}$ where $G_{j'}$ is isomorphic to G_j. If $j = j'$ then the verifier accepts, i.e. claims the graphs are not isomorphic, and if $j \neq j'$ the verifier rejects.

The completness of the interactive protocol is straightforward. If G_1 and G_2 are not isomorphic, then the only graph that is isomorphic to G_j is G_j itself, so the prover returns j and the verifier accepts with probability 1. If G_1 and G_2 are isomorphic, then the prover could return either 1 or 2. If $j \neq j'$ then the verifier will reject and will be correct, else it will incorrectly accept. This is not a problem for the verifier as long as it is incorrect with some constant probability. It turns out that the prover is equally likely to return 1 or 2 in this case, so therefore the probability of it accepting incorrectly is exactly $1/2$. ■

Do the random bits of the verifier in an IP protocol need to be kept private or secret from the prover? In the proof of the above Theorem 18, this seems crucial. Consider the following class.

Definition 19 (Arthur-Merlin (AM) protocol) *AM(k) is the class of languages that can be decided in k rounds of an interactive proof where the prover has access to all random bits generated by the verifier at the time of interaction.*

While it is clear that $\text{AM}(k) \subseteq \text{IP}(k)$, which is analogously the class of languages decidable in k rounds of an interactive proof, the following theorem may be surprising.

Theorem 20 *IP(k) $\subseteq$ AM(k + 2)*

See Arora and Barak book, Theorem 8.8. This, of course, implies that we did *not* need private random bits. The protocol can be changed to one that differentiates between isomorphic and non-isomorphic graphs by noticing that in the latter case the set of graphs that are either isomorphic to G_1 or isomorphic to G_2 is twice as large (assuming we are dealing with graphs with trivial automorphism group – we count a slightly different number otherwise). Thereafter, distinguishing between isomorphic and non-isomorphic graphs reduces to finding a lower bound for the size of this set, which can be achieved using a set lower bounding public-coin protocol by Goldwasser and Sipser.

The proof of Theorem 18 can be used to show that graph isomorphism is not NP-complete (and graph non-isomorphism, GNI, is not co-NP complete) unless the polynomial time hierarchy collapses to the 2nd level due to roughly the following argument. GNI is in the class AM(2), which is basically $\mathrm{BPP}^{\mathrm{NP}}$ (and it turns out that any constant rounds of interaction can be reduced to 2). We can use GNI's assumed coNP-completeness to take a Σ_2^p-complete set Σ_2SAT (deciding truth of $\exists\forall$CNF) and reduce the $\forall$CNF (co-NP-complete part) to GNI, and then use the AM(2) protocol for GNI to massage it into a Π_2^p structure. This shows that $\Sigma_2\mathrm{SAT} \in \Pi_2^p$, resulting in the collapse of PH to the second level. (See Arora and Barak book, Theorem 8.16, proof by Boppana, Sipser, and Zachos that if co-NP$\subseteq \mathrm{BPP}^{\mathrm{NP}} = \mathrm{AM}(2)$, then PH collapses to the 2nd level.)

4.2 Arithmetization

Before we state the main theorem to be proven in this section, we show another result whose proof is similar in structure.

Definition 21 (Arithmetization of Boolean Formulas) *Given a Boolean formula $\phi(x_1, \ldots, x_n)$, its* arithmetization *is the polynomial over $\mathbb{Z}$ formed by first replacing all Boolean variables x_i with variables x_i over $\mathbb{Z}$, where an assignment of 0 corresponds to "false" and 1 to "true". Then, replace $\neg x_i$ with $(1 - x_i)$, $x_i \wedge x_j$ with $x_i x_j$, and $x_i \vee x_j$ (which is $\neg(\neg x_i \wedge \neg x_j)$) with $x_i * x_j := 1 - (1 - x_i)(1 - x_j)$.*

Lemma 22 *co-NP $\subseteq$ IP*

Proof: We will show that the co-NP-complete language TAUT, 3CNF formulas that are tautologies, is contained in IP. Let $\phi \in$ TAUT. Find Φ, the arithmetization of ϕ. Since $\phi \in$ TAUT, we know that ϕ is in conjunctive normal form with three variables per clause. For a given clause to evaluate to true, at least one of its variables must be true, and hence in Φ, the clause will evaluate to 1. For the entire formula to evaluate to true, each clause must be true, and hence each clause must evaluate to 1. Therefore, for a given truth assignment, the entire formula ϕ is satisfied if $\Phi = 1$. Now for ϕ to be in TAUT, we need it to be satisfied under *every* truth assignment.

To show that a particular ϕ is a tautology, we add a sequence of summands in front of it, one for each variable. Hence

$$\phi(x_1, x_2, \ldots, x_n) = \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} \Phi(x_1, x_2, \ldots x_n).$$

If ϕ is a tautology, this sum will evaluate to 2^n. Our prover must now convince our verifier that F_ϕ evaluates to 2^n. The basic idea is that the prover and verifier have a linear length conversation where the outermost quantifier is stripped off of F_ϕ each time. This keeps the verifier from having to do $O(2^n)$ operations, which it cannot do since it is polynomial in time.

We also note that since the verifier is bounded in polynomial time, we cannot deal with numbers which are exponential in size. Hence, we want to only deal with values from $\mathbb{Z}/\mathbb{Z}_q$ for some prime q. This puts an upper bound on the size of integers we are dealing with and since q is prime, we know that $\mathbb{Z}/\mathbb{Z}_q$ will be a field. We note that the value of F_ϕ cannot exceed $O(2^{2^n})$.

Suppose F_ϕ evaluates to A. Then we can find a prime q such that $A \neq 0 \bmod q$ if and only if ϕ is true. If ϕ is false, then $A = 0$ modulo any prime. Assume A is nonzero. If ϕ is satisfiable and A is zero modulo all polynomial length primes, then by the Chinese Remainder Theorem, it is also zero modulo their product. Since this product is $\Omega(2^{2^{n^d}})$, this contradicts the fact that A is nonzero and at most $O(2^{2^n})$. Hence we can always find a prime of the correct size.

For the protocol itself, at each stage P will send V some polynomial $P(x)$ and must convince V that $v_i = \sum_{x=0}^{1} P(x)$ where $v_i \in \mathbb{F}_q$ was computed by V in the previous round. V initializes $v_0 = 0$. At stage i, P sends V the polynomial

$$P(x) = \sum_{x_{i+1}=0}^{1} \sum_{x_{i+2}=0}^{1} \cdots \sum_{x_n=0}^{1} \Phi(r_1, r_2, \ldots, r_{i-1}, x_i, \ldots, x_n)$$

If $v_{i-1} = P(0) + P(1)$, then V chooses a random $r_i \in \mathbb{F}_q$, sets $v_i = P(r_i)$, and sends r_i to P. Else V rejects. At the end of all the rounds of interactions, if $v_n = 2^n$, then V accepts, else it rejects.

The completeness is clear, since the variables v_i serve to count the number of satisfying assignments. $P(0)$ and $P(1)$ simulate what happens if the variable x_i is true and false, respectively. If ϕ is a tautology, then it will be satisfied in both of those cases, hence $P(0) + P(1) = 2$ and at the end of the protocol we will have that $v_n = 2^n$ and V will accept with probability 1. If ϕ is not a tautology, then for some x_i either $P(0) = 0$ or $P(1) = 0$ and hence $v_n \neq 2^n$ and V will reject with probability 1.

Recalling the definition of IP from above, we note that all $x \in$ TAUT have the form $\forall x_1, x_2, \ldots x_n \phi(x_1, x_2, \ldots x_n)$. There is one quantifier, $\forall$, which corresponds to a linear number of rounds of interaction. ■

Lemma 23 *$P^{\#P} \subseteq IP$*

Proof: We simply modify this proof of TAUT $\subseteq$ IP so that at the end V checks whether or not $v_n = k$. We have that ϕ has exactly k satisfying assignments if and only if $v_n = k$. The rest of the interactive protocol from the above proof remains the same. Hence $\mathrm{P}^{\#\mathrm{P}} \subseteq$ IP. ■

4.3 Degree Reduction

Another technique which is essential to the following proof of IP=PSPACE is degree reduction, which we will introduce via an alternate proof of the previous lemma.

Lemma 24 *co-NP* $\subseteq$ *IP*

Proof: We will show this by showing TAUT $\subseteq$ IP, as in the previous lemma, but this time when we arithmetize ϕ we will use products instead of summands. With this technique, ϕ is arithmetized to

$$F_\phi = \prod_{x_1=0}^{1} \prod_{x_2=0}^{1} \cdots \prod_{x_n=0}^{1} \Phi(x_1, x_2, \ldots, x_n)$$

This interactive protocol is identical to the previous one, except the verifier evaluates products instead of sums. The original formula is a tautology if and only if the result of the protocol is 1, and 0 otherwise.

The issue that arises is that the degree of the polynomial may double with each product, giving us a polynomial of degree 2^n. Our verifier cannot work with a polynomial of such high degree, so we must do something to fix this. We see that since our original formula had only boolean variables, the only values of the x_i that we care about are those such that $x_i^2 = x_1$. Hence, we can replace every occurrence of x_i^k with x_i for any $k > 1$. We denote this operation by $\Phi \mod (x_i^2 - x_i)$ and will use R_{x_i} to apply this operation to a polynomial, i.e. $R_{x_i}\Phi = \Phi \mod (x_i^2 = x_1)$. Hence, to keep the degree of the polynomial that the verifier must evaluate low enough, we simply reduce it at the start of each round of interaction. ■

4.4 IP = PSPACE

Using the above as an analog, we approach the main theorem of this section. The original proof was due to Shamir [53]. The proof that we present here was also influenced by [55, 33, 34].

Definition 25 (Quantified Boolean Formula (QBF)) *If* $B(x_1, x_2, \ldots, x_n)$ *is a* quantifier-free Boolean formula *and each* $Q_i \in \{\exists, \forall\}$*, then* $Q_1x_1Q_2x_2 \ldots Q_nx_nB(x_1, x_2, \ldots x_n)$ *is a* totally quantified Boolean formula*. We refer to the language consisting of these formulas as QBF.*

Theorem 26 *IP = PSPACE*

Proof: IP $\subseteq$ PSPACE: This is clear since IP can be simulated in PSPACE.

PSPACE $\subseteq$ IP: To show this, it suffices to show that some PSPACE-complete language is contained in IP. We will take the language of true quantified Boolean formulas (QBF) to be our PSPACE-complete language. Recalling the definition of IP from above and the note at the end of the proof of TAUT $\in$ IP, we note here that every $\psi \in$ QBF is of the

form $Q_1x_1Q_2x_2\ldots Q_nx_nB(x_1,x_2,\ldots,x_n)$, so we should not be surprised if it can be solved in polynomially many interactions with an interactive protocol.

Arithmetization: This section of the proof follows the first proof of TAUT $\subseteq$ IP closely. If $\psi \in$ QBF, then it is of the form $Q_1x_1Q_2x_2\cdots Q_nB(x_1,x_2,\ldots,x_n)$ and we must first arithmetize it. We first replace

$$\forall x_1 B(x_1,\ldots,x_n) \text{ with } \prod_{x_i=0}^{1} B(x_1,\ldots,x_n)$$

and

$$\exists x_1 B(x_1,\ldots,x_n) \text{ with } \coprod_{x_i=0}^{1} B(x_1,\ldots,x_n) := 1-(1-B(0,x_2,\ldots,x_n))(1-B(1,x_2,\ldots,x_n))$$

and arithmetize B into Ψ as we did earlier for $\phi \in TAUT$. With this arithmetization, our original $\psi \in$ QBF is satisfiable if and only if the resulting arithmetization evaluates to 1. Hence P will have to convince V that the polynomial evaluates to 1.

Degree Reduction: Here we follow the second proof of TAUT $\subseteq$ IP by showing how we will keep the degree low enough for the verifier to handle. The basic idea is that P and V will have a polynomially long conversation where they strip the outermost operator from the polynomial with each iteration. Let

$$F_\psi = \mathcal{O}_1 R_{x_1} \mathcal{O}_2 R_{x_1} R_{x_2} \cdots \mathcal{O}_n R_{x_1} \cdots R_{x_n} \Psi(x_1,\ldots,x_n) \mod q$$

where each $\mathcal{O}_i$ is $\prod_{x_i}$ or $\coprod_{x_i}$ and R_{x_i} is the degree reduction operation $\Psi \mod (x_i^2 = x_i)$ from above. We have that F_ψ preserves the satisfiability conditions of ψ and the R_{x_i} operations keep it computable by the verifer. V computes some value v_k each round. If $k = i + \sum_{m=0}^{i} m$ for some $i \geq 0$, then P must convince V that

$$v_k = \mathcal{O}_{i+1} \cdots \mathcal{O}_n R_{x_1} \cdots R_{x_n} \Psi_k \mod q$$

Else then $k = r + i + \sum_{m=0}^{i} m$ for some i and $1 \leq r \leq i+1$ so P must convince V that

$$v_k = R_{x_r} \cdots \mathcal{O}_{i+2} \cdots \mathcal{O}_n R_{x_1} \cdots R_{x_n} \Psi_k \mod q$$

where Ψ_k is the resulting polynomial after k operators have been stripped and q is some prime. We must take the value of Ψ modulo some prime for the same reasons we needed to in the above proof of TAUT $\subseteq$ IP.

The verifier starts with $v_0 = 1$ and $\Psi_0 = \Psi$. On step k, there are three possible cases:

Case 1: The outermost operator is $\mathcal{O}_i = \prod_{x_i}$ for some i. In this case P wants to convince V that

$$v_k = \prod_{x_i} R_{x_1} \cdots R_{x_i} \mathcal{O}_{i+1} \cdots \mathcal{O}_n R_{x_1} \cdots R_{x_n} \Psi(r_1,\ldots,r_{i-1},x_i,\ldots,x_n) \mod q. \tag{4.1}$$

We note that what actually happens is that the expression is computed symbolically first, and then evaluated at $(r_1,\ldots,r_{i-1})$. The prover sends V a degree-1 polynomial $P(x_i)$ and V checks if $v_k = \prod_{x_i} P(x_i)$. If not, then V rejects; else, V chooses a random $r_i \in \mathbb{F}_q$ and

sets $v_{k+1} = P(r_i)$. We then enter the next round. Completeness is easy to see, since if the original QBF ψ is true, then P can send

$$P(x_i) = R_{x_1} \cdots R_{x_i} \mathcal{O}_{i+1} \cdots \mathcal{O}_n R_{x_1} \cdots R_{x_n} \Psi(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n) \mod q \tag{4.2}$$

and the verifier will accept. For soundness, if Equation (4.1) is not satisfied, then P will send some polynomial other than the one described in Equation (4.2), which will only be accepted with probability $1/q$.

Case 2: The outermost operator is $\mathcal{O}_i = \coprod_{x_i}$ for some i. This case is similar to case 1.

Case 3: The outermost operator is R_{x_i} for some i. This case is similar to cases 1 and 2, however if the verifier does not reject, instead of choosing a random $r_{i+1} \in \mathbb{F}_q$, we rather choose a *new* r_i. This is because degree reduction is not eliminating any variables. Completeness and soundness are similar to the other cases with the exception that for soundness the polynomial will be accepted with probability d/q where d is the degree of the polynomial.

Determining the size of the prime q: We now need to determine what size our prime q should be in order to have a sufficiently small probability of error for soundness. Each of the n $\prod$ or $\coprod$ operators err with probability $1/q$, each of the final n R operations err with probability $3m/q$ (where m is the number of clauses), and all other R operations err with probability $2/q$. Applying union bounds give the total probability of error to be $\frac{3mn+n^2}{q}$. Hence a polynomial length q suffices to give a reasonably small probability of error. Hence, PSPACE $\subseteq$ IP. ■

An important observation is that, without loss of generality, the completeness probability c of an IP protocol can always be assumed to be 1. Since the proof that QBF is in IP gets a completeness probability of 1, and every IP can be simulated in PSPACE, this is clearly the case, i.e. the class IP can equivalently be defined to have a completeness probability of 1.

Chapter 5

Probabilistically Checkable Proofs

Lecture(s): 10–12
Date(s): 1/26, 1/28
Lecturer: Meera Sitharam
Scribe: Troy Baker

5.1 Introduction

Whereas the verifier in an interactive proof can communicate with the prover (i.e. send information) while getting chunks of the proof at a time, consider the following class where there is no communication.

Definition 27 (Probabilistically checkable proof (PCP) systems) *For some language $\mathcal{L}$, a* PCP system *is a randomized machine (the* verifier*) with constant time access to some proof generated by an oracle. By completeness, if $x \in \mathcal{L}$, then there exists some proof that makes the verifier accept. By soundness, if $x \notin \mathcal{L}$, then given any proof the verifier accepts with probability at most $1/2$.*

Definition 28 (PCP$[f(n), g(n)]$) *The class of languages recognizable by a PCP system where the verifier generates at most $f(n)$ random bits and reads at most $g(n)$ bits of the proof.*

The machinery for the PCP theorem was started in the late 1980's about the same time as circuit lower bounds, but the theorem itself was not proven until 1998 by ALMASS (Arora, Lund, Motwani, Safra, and Sudan) [7]. The proof that we will present in a later lecture is due to Dinur in 2005 [19].

Theorem 29 (PCP theorem [7]) *$NP = PCP[O(\log n), O(1)]$.*

While this theorem is a major result of computational complexity theory, several containments relating familiar classes and various PCP classes can easily be shown.

Exercise 1 *Show that $PCP[O(\log n), 0] = P$.*

Solution: The PCP machine has no proof string (i.e. looks at 0 bits of the string the prover sends) and a verifier that uses $\log n$ random bits. A polytime machine can deterministically generate $2^{\log n} = poly(n)$ bit strings of length $\log n$ and simulate the polytime running of the verifier on each bit string. If the verifier would accept on all strings, the P machine accepts; otherwise, it rejects.

In the other direction, we know that a verifier is a polytime machine. So we can simply use the P machine as our verifier (without taking advantage of random bits). ■

Exercise 2 *Show that* $PCP[O(\log n), O(1)] \subseteq NP$.

Solution: Since there are only $\log n$ random bits, at most a $poly(n)$ length proof can be indexed. Thus we can assume the proof has $poly(n)$ length. If the input x is in the language L of the PCP machine, then there exists some proof such that for all bit strings of length $\log n$ the verifier accepts. If $x \notin L$, then at most s fraction of the random bit strings cause the verifier to accept.

The NP machine does the following. Guess a proof π of $poly(n)$ length using nondeterminism. Deterministically simulate the verifier by trying all possible bit strings of length $\log n$, which can be done in polytime ($2^{\log n} = poly(n)$). Accept only if there is some π such that the verifier would accept on all bit strings. ■

Exercise 3 *Show that* $PCP[O(\log n), O(poly(n))] \subseteq NP$.

Solution: Similar to the construction of the NP machine in Exercise 2, we build a machine that nondeterministically chooses a bit string of length $poly(n)$ and a then deterministically generates all bit strings of length $\log n$ with which it simulates the verifier. The only difference is that in the simulation of the verifier, the NP machine will look at all bits in the proof string. ■

Exercise 4 *Show that* $NP \subseteq PCP[O(poly(n)), O(poly(n))]$.

Solution: If we have an NP machine, that means there is some deterministic verifier of proofs of length $poly(n)$ for the language. In our construction of the PCP machine, we simply use this as our verifier (without taking advantage of the random bits). If x is in the language L of the NP machine, then their exists some proof the prover could send that would cause the verifier to accept. If $x \notin L$, with probability 1 it will reject. ■

Exercise 5 *Show that* $PCP[O(poly(n)), O(poly(n))] \subseteq NP^{BPP}$, *where the proof is restricted to* $poly(n)$ *bits.*

Solution: The NP machine non-deterministically guesses the proof which it then gives to a BPP oracle that behaves like the verifier. The NP machine accepts if the oracle accepts and rejects otherwise. ■

Exercise 6 *Show that IP = PSPACE ⊆ PCP[$O(poly(n)), O(poly(n))$].*

Solution: Here the definition of PCP[$O(poly(n)), O(poly(n))$] is important to clarify: the prover can in fact send an exponentially long proof, since the verifier can index all of its bits using $poly(n)$ random bits; the verifier however extracts only polynomially many of the proof bits.

So what remains to be shown is that the interactive power of the IP can be simulated by the PCP protocol, in which the prover sends its proof all at one go. This is achieved by using as the PCP proof the "truthtable" that captures the IP prover's behavior, i.e., the truthtable maps the interaction and verifier random bits so far to the output that the prover communicates to the verifier in a given round. This truthtable is no more than exponentially long, and the PCP verifier can use its random bits to extract the appropriate portions of this "proof". Now using the IP's completeness and soundness, we can derive this resulting PCP protocol's completeness and soundness.

In fact, the class of multiprover interactive protocols (MIP) [8] was shown to be equivalent to PCP[$O(poly(n)), O(poly(n))$] = PCP[$O(poly(n)), O(1)$]. Additionally, it was shown that MIP = NEXP. ■

5.2 An Indirect Proof of PCP Theorem

As mentioned, showing Theorem 29 is nontrivial. The first proof [7] is very long and is similar in structure to the IP = PSPACE proof. Initially, they were able to show NP ⊆ PCP[$O(poly(n)), O(1)$]. While this is what we will demonstrate later, the same group was also able to show NP ⊆ PCP[$O(\log n), O(1)$].

Another, substantially shorter, technique was used by Dinur [19] to show Theorem 29, which worked via the following equivalence.

Definition 30 (GAP3SAT$_{c,s}$) *Given input F, c, and s, where F is a 3CNF formula with n variables and m clauses and c and s are constants between 0 and 1, the input is in GAP3SAT$_{c,s}$ if the maximum number of clauses in F that are simultaneously satisfiable is at least cm (completeness) and rejects if it is at most sm (soundness).*

Theorem 31 (PCP theorem and approximability) *PCP theorem (theorem 29) ⟺ there exists some constant $0 \leq s < 1$ such that GAP3SAT$_{1,s}$ is NP-hard.*

We will actually use the slightly weaker statement that NP ⊆ PCP[$O(\log n), O(1)$]. Then by Theorem 32 and the solution to Exercise 2 we will have proven Theorem 29.

Proof: First, we assume Theorem 29 (or rather, the weaker statement that NP ⊆ PCP[$O(\log n), O(1)$]) is true. In this case we have a (PCP[$O(\log n), O(1)$])-protocol for 3COLOR. Now we wish to find a polytime reduction R from 3COLOR to GAP3SAT$_{1,s}$.

Consider a (PCP[$O(\log n), O(1)$])-protocol for 3COLOR. For a given graph G and an instantiation i of the verifier's $\log n$ random bits, there is some specific constant number of proof bits that are accessed by the verifier. Assign each proof bit by a Boolean variable. Create a constant sized Boolean function clause C_i of those variables so that C_i is true for a given assignment α to its variables if and only if the verifier accepts when the random bits

have value i and the corresponding proof bits have value α. Now the CNF Φ is taken to be a conjunction of the C_i's. Since there are only $2^{\log n} = O(poly(n))$ instantiations i, the CNF Φ can be generated in polynomial time, hence this reduction R is a polynomial time reduction.

By the completeness of the PCP machine, if $G \in$ 3COLOR then $R(G) \in$ GAP3SAT$_{1,s}$. G being 3-colorable means there is some proof (corresponding to an assignment to the variables of $R(G)$) such that every random bit string (corresponding to clauses in $R(G)$) will cause the verifier to accept (corresponding to $R(G)$ being satisfiable). By the soundness of the PCP machine, if $G \notin$ 3COLOR then $R(G) \notin$ GAP3SAT$_{1,s}$. G not being 3-colorable means that for any proof at most a fraction s of the of the random bit strings will cause the verifier to accept (corresponding to at most a fraction s of the clauses in $R(G)$ being satisfied by any assignment). We omit a detail here: technically, the formula we make is not a 3CNF. We take it for granted that there is some method in which to massage this formula into a 3CNF while still maintaining a constant soundness that is potentially different from s.

In the other direction, we assume there exists some constant s such that GAP3SAT$_{1,s}$ is NP-hard. In this case, we have a polynomial time reduction R from any NP-hard problem to GAP3SAT$_{1,s}$. We will take R to be the reduction from 3-colorability (3COLOR), but note that this choice of NP-hard problem is arbitrary.

We now describe a (PCP$[O(\log n), O(1)]$)-protocol for 3COLOR using R. The verifier runs the reduction to get the 3CNF formula $R(G)$, with m clauses and n variables. It then generates $\log m$ random bits and uses this as an index to select a particular clause in $R(G)$, that we will call ϕ. The prover sends n bits. The verifier uses these as an assignment for $R(G)$. If ϕ is true, the verifier accepts; otherwise, it rejects.

By the completeness of GAP3SAT$_{1,s}$, if $R(G) \in$ GAP3SAT$_{1,s}$then some assignment satisfies the entire formula and, by extension, all clauses. Therefore, there is some assignment to the variables in ϕ (corresponding to a proof string) such that no matter what clause ϕ in $R(G)$ we consider (corresponding to some random bits our verifier chooses) ϕ will be true (corresponding to the verifier accepting). By the soundness of GAP3SAT$_{1,s}$, if $R(G) \notin$ GAP3SAT$_{1,s}$then no assignment satisfies more than a s of the clauses. Therefore, given any assignment (corresponding to some proof string) the probability ϕ is true is s (corresponding to the probability, over the space of random bit strings, of the verifier accepting). ■

Given the above proof, to prove Theorem 29 it suffices to show Theorem 32. Before we approach Dinur's proof of this equivalent statement we must introduce expander graphs and constraint satisfaction problems.

Theorem 32 (Inapproximability of GAP3SAT$_{1,s}$) *There exists some constant s such that GAP3SAT$_{1,s}$ is NP-hard.*

5.3 Spectral Properties of d-regular Graphs

For the purposes of this document, when we say graph we mean an undirected *multigraph*, in which there can be multiple edges between the same two vertices. The graph union of two multigraphs will have have one copy of edge (u, v) for each copy in both graphs.

There are many *spectral* properties of a graph that can be derived from the eigenvectors and eigenvalues of matrices associated with the graph. In particular, we are interested in the adjacency matrix. Given an undirected multigraph G, its adjacency matrix A_G is a $|V| \times |V|$ matrix, where element (i, j) is n, the number of edges between vertices v_i, v_j. Note that this matrix is necessarily symmetric. We call the eigenvalues of the adjacency matrix $\lambda_1, \lambda_2, \ldots, \lambda_n$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

We now introduce an important class of graphs and demonstrate some relevant properties of its eigenvalues.

Definition 33 (d-regular graph) *A graph where the degree of every vertex is exactly d.*

Observation 34 *The largest eigenvalue, λ_1, of a d-regular graph is d and the corresponding eigenvector is 1^n (the all-ones vector).*

Proof: Every row of the adjacency matrix A_G of a d-regular graph G sums to d, therefore $A_G v = dv$ where $v = 1^n$. ■

Observation 35 *The second largest eigenvalue (by magnitude) is* $\max\{\lambda_2, |\lambda_n|\}$.

Proof: Eigenvalue λ_2 must be positive, but λ_n may be a larger negative value. ■

Observation 36 *The second largest eigenvalue (by magnitude) can be written as:*

$$\lambda = \max_S \frac{|\langle A_G x, x\rangle|}{|\langle x, x\rangle|}$$

where the set $S = \{x | (x \in \mathbb{R}^n) \wedge (x \neq 0) \wedge (\langle x, 1^n\rangle = 0)\}$, i.e. the span of the eigenvectors corresponding to eigenvalues $\lambda_2, \ldots, \lambda_n$.

Proof: Note that any vector x can be written as $\sum_{i=2}^n \alpha_i v_i$ where v_i is the eigenvector corresponding to λ_i and $\{\alpha_i\}$ are some constants. Let v be the eigenvector corresponding to λ. Note that $\{v_i\}$ are orthonormal.

In one direction we have

$$\frac{|\langle A_G x, x\rangle|}{|\langle x, x\rangle|} = \frac{|\sum_{i=2}^n \alpha_i^2 \lambda_i|}{\sum_{i=2}^n \alpha_i^2} \leq \frac{\sum_{i=2}^n \alpha_i^2 |\lambda_i|}{\sum_{i=2}^n \alpha_i^2} \leq \frac{\lambda \sum_{i=2}^n \alpha_i^2}{\sum_{i=2}^n \alpha_i^2} = \lambda.$$

In the other direction, we have

$$\frac{|\langle A_G x, x\rangle|}{|\langle x, x\rangle|} \geq \frac{|\langle A_G v, v\rangle|}{|\langle v, v\rangle|} = \frac{|\langle \lambda v, v\rangle|}{|\langle v, v\rangle|} = \lambda.$$

■

Lemma 37 *If $G = (V, E_G)$ is d-regular and $H = (V, E_H)$ is d'-regular such that $G' = G \cup H$ is $(d + d')$-regular, then $\lambda_{G'} \leq \lambda_G + \lambda_H$, where λ_X is the second largest eigenvalue (by magnitude) of graph X.*

Proof: Given that G' is a union of multigraphs G and H, it must be that $A_{G'} = A_G + A_H$. By Observation 36 we have the following inequality:

$$\lambda_{G'} = \max_S \frac{|\langle A_{G'}x, x\rangle|}{|\langle x, x\rangle|} = \max_S \frac{|\langle A_G x, x\rangle + \langle A_H x, x\rangle|}{|\langle x, x\rangle|} \leq \max_S \frac{|\langle A_G x, x\rangle|}{|\langle x, x\rangle|} + \max_S \frac{|\langle A_H x, x\rangle|}{|\langle x, x\rangle|}$$

where the set $S = \{x | (x \in \mathbb{R}^n) \wedge (x \neq 0) \wedge (\langle x, 1^n\rangle = 0)\}$, i.e. the span of the eigenvectors corresponding to eigenvalues $\lambda_2, \ldots, \lambda_n$. ■

5.4 Expander Graphs

To show the inapproximability of GAP3SAT$_{1,s}$, we need to understand *expansion* and *expander graphs.*

Definition 38 (Expansion) *The expansion of graph G is*

$$\phi(G) = \min_{\{S | (S \subseteq V) \wedge (S \neq \emptyset) \wedge (|S| \leq n/2)\}} \frac{|E(S, \overline{S})|}{|S|}$$

where $E(X, Y)$ is the set of edges between vertex sets X and Y.

Intuitively, *expander graphs* are graphs with high expansion. For our purposes, we want sparse expanders, in particular, d-regular expanders. Here we state a theorem without proving it, which guarantees the existence of such a class of graphs.

Theorem 39 (d-regular expanders) *There exists constants $d > 1$, $\phi_0 > 0$, and an explicit family of d-regular graphs $\{G_n\}_{n \geq 1}$ such that $\phi(G_n) \geq \phi_0$, where n is the number of vertices.*

Definition 40 ((n, d, λ)-expander) *A d-regular graph on n vertices for which the second largest eigenvalue (by magnitude) of the adjacency matrix is equal to λ.*

We call $d - \lambda$ the *gap* of an (n, d, λ)-expander. The following are important properties of (n, d, λ)-expanders.

Lemma 41 (Expansion of (n, d, λ)-expanders) *If G is an (n, d, λ)-expander, then $(\phi(G))^2 \leq 2d(d - \lambda) \leq 2\phi(G)$*

Theorem 42 (Existence of (n, d, λ)-expander) *There exists constants $d \geq 3$ and $\lambda < d$ and an explicit family of polynomial time computable (n, d, λ)-expanders.*

For a detailed introduction to expander graphs, we refer you to [25]. The following is an explicit example of an expander on m^2 nodes discussed in the book.

Definition 43 (Margulis-Gabber-Galil Expander) *For every natural number m, consider the graph with vertex set $\mathbb{Z}^m \times \mathbb{Z}^m$ and the edge set in which every $(x, y) \in V$ is connected to the vertices $(x \pm y, y)$, $(x \pm (y + 1), y)$, $(x, y \pm x)$, and $x, y \pm (x + 1)$, where the arithmetic is modulo m.*

This graph is 8-regular with an eigenvalue bound that is (independent of m) $\lambda < 8$.

Theorem 44 *There exists an explicit construction of a family of $(m^2, 8, 7.9999)$-expanders. Furthermore, the neighbors of a vertex in these expanders can be computed in logarithmic-space.*

5.5 Constraint Satisfaction Problems

Definition 45 (Constraint satisfaction problem (CSP) over alphabet Σ) *A tuple (G, C), where $G = (V, E)$ is a graph and $C : E \to \mathcal{P}(\Sigma \times \Sigma)$ is a mapping of edges to constraints.*

An assignment *(or* evaluation*) of a CSP is a mapping $\alpha : V \to \Sigma$. An assignment* satisfies *(or is* consistent *with) a constraint $C((u, v))$, where $(u, v) \in E$, if $(\alpha(u), \alpha(v)) \in C((u, v))$.*

Definition 46 (CSP decision problem over alphabet Σ) *Given a CSP over Σ, is there an assignment that satisfies all constraints?*

Definition 47 (CSP max problem over alphabet Σ) *Given a CSP over Σ, what is the maximum number of constraints that can be simultaneously satisfied by an assignment?*

Definition 48 (CSP gap problem over alphabet Σ) *Given a CSP over Σ and constants $c, s \in [0, 1]$, is there an assignment that simultaneously satisfies at least $c|E|$ constraints? If not, then we are given a "*promise*" that no more than $s|E|$ can be satisfied.*

Example 1 (kCOLOR as a CSP) *We can formulate kCOLOR as a CSP decision problem over Σ, where $|\Sigma| = k$, in the following manner: the graph is simply the input graph and for all $e \in E$ the corresponding constraint is $C(e) = \{(a, b) | (a, b \in \Sigma) \wedge (a \neq b)\}$. Thus, a satisfying assignment of colors to each vertex is one such that for all $u, v \in V$, if $(u, v) \in E$ then $(\alpha(u), \alpha(v)) \in C(u, v)$ (i.e. no two adjacent vertices have the same color).*

As an aside, the *unique games conjecture (UGC)* makes an interesting statement about a certain class of CSPs. Recall that a *permutation* is simply a bijection of the domain with itself.

Definition 49 (Unique games CSP over alphabet Σ) *A CSP over Σ where every constraint is a permutation.*

Conjecture 50 (Unique games conjecture) *Determining membership in a unique games CSP gap problem over alphabet Σ is NP-hard.*

5.6 Dinur's Proof of the Inapproximability of GAP3SAT$_{1,s}$

Now that we have a better understanding of the methods used in Dinur's proof of Theorem 32, we sketch the proof.

Sketch of Proof: It is clear that a CSP decision problem Q is equivalent to GAP-$Q_{1,1-1/|E|}$ (as any unsatisfying assignment must violate at least one constraint), therefore

Q is NP-hard if and only if GAP-$Q_{1,1-1/|E|}$ is NP-hard. We now amplify this gap from $1-1/|E|$ to a constant s in polynomial time.

The input to GAP-$Q_{1,1-1/|E|}$ is graph G_0. At each stage of the amplification, we will transform G_i into G_{i+1} while maintaining the size of the alphabet, ending with G_{final}. If $|G_{i+1}| = poly(|G_i|)$ and we do even $O(\log n)$ stages, then $|G_{\text{final}}|$ will be too large. Therefore, we will get $|G_{i+1}| \leq m|G_i|$ where m is a constant. After $\log|E|$ stages we have, $|G_{\log|E|}| \leq m^{\log|E|}|G_i|$, which is acceptable. All of this is under the assumption the alphabet is maintained, which we will ensure.

Each stage i has 4 steps:

1. Sparsification (degree reduction): G_i is made into $G_i^{(1)}$, a d-regular graph with $\text{gap}(G_i^{(1)}) \geq \beta^{(1)}\text{gap}(G_i)$, where $w, d, \beta^{(1)}$ are constants greater than 0. This step makes use of Theorem 39. Alphabet remains the same in this step.

2. Expanding: $G_i^{(1)}$ is made into $G_i^{(2)}$, an (n, d_i, λ_i)-expander (where d_i, λ_i are constants), using something similar to Theorem 42. Alphabet and gap remain the same in this step.

3. Gap amplification: $G_i^{(2)}$ is made into $G_i^{(3)}$, which may no longer be an expander. This step amplifies the gap by some factor t, while dramatically increasing the size of the alphabet.

4. Alphabet reduction: $G_i^{(3)}$ is made into G_{i+1}, which is some constant factor larger than G_i. G_{i+1} has the same gap as $G_i^{(3)}$ but reduces the expansion properties.

■

Chapter 6

Dinur's Proof of PCP Theorem

Lecture(s): 13–18
Date(s): 2/2, 2/4, 2/9, 2/11
Lecturer: Meera Sitharam
Scribe: Rahul Prabhu

6.1 The PCP Theorem by Gap Amplification

This proof of the PCP theorem was given by Irit Dinur in her paper titled "The PCP Theorem by Gap Amplification"[18]. In this discussion, background definitions and theorems not directly related to the proof are discussed in the Appendix. We start by stating the PCP theorem.

Theorem 51 (PCP Theorem) $NP \subseteq PCP[log\ n, 1]$.

Which means that all languages in NP can be verified by a random log n bits probabilistic verifier with access to a constant number of bits of the proof. The inapproximability version of this theorem is as follows

Theorem 52 (Inapproximability version of PCP Theorem) *There are integers $q > 1$ and $|\Sigma| > 1$ such that given as input a collection C of q-ary constraints over an alphabet Σ, it is NP-hard to decide whether $gap(C) = 0$ or $gap(C) \geq \frac{1}{2}$.*

It can be shown that Theorem 51 and Theorem 52 are equivalent. This means that assuming that there is a reduction that takes an instance of any NP-language into a constraint system[1] such that the gap property holds, we can construct a verifier. This result is used to give the new proof of the PCP theorem. We first construct a reduction that takes an NP problem ($GAP3Color_{1,s}$ in our case) to a constraint system so that the gap property holds. Then we build the verifier to complete the proof. The inapproximability version of $GAP3Color_{1,s}$ is

Theorem 53 (Inapproximability of $GAP3Color_{1,s}$) *$\exists$ a universal constant s such that 3-colorability $= GAP3Color_{1,1-\frac{1}{|E|}} \leq_m^P GAP\_CSP_{1,s}$*

[1]For definition of a constraint system see Appendix 6.2.1

Overview of Reduction

The reduction makes a series of transformations on the input Graph G_0 with the objective of amplifying the gap. In the series of transformations $G_0, \ldots G_i, G_{i+1} \ldots, G_k$, for each intermediate state G_i we want the following to hold.

$$\begin{aligned} |G_i| &\leq m|G_{i+1}| \\ |G_k| &\leq m^{log(|E|)}|G_0| \end{aligned} \tag{6.1}$$

That is, after each transformation, we want the graph size to increase by no more than a constant factor and the size of the graph in the final state should be no more than a polynomial in the size of the initial graph.

Each transformation $G_i \rightarrow G_{i+1}$ is achieved by doing the following four operations in order.

1. **Sparsification(Degree Reduction)**: In this step we transform the graph in such a way that:
   (a) The resulting graph is a $d_0 + 1$ regular graph with at most constant factor size increase.
   (b) The alphabet size $|\Sigma_0|$ remains the same.
   (c) The new gap, $gap' \geq \frac{gap}{O(1)}$ i.e., the gap goes down slightly.
2. **Expanderizing** In this stage we convert the graph to a (n, d_1, λ_1) expander graph.
   - The alphabet size $|\Sigma_0|$ remains the same.
   - The gap is maintained.
3. **Amplification** Starting with G - a (n, d_1, λ_1) expander, and alphabet Σ_0, we transform it so that
   (a) Size of the graph only increases linearly. $|size'| = O(size)$.
   (b) The alphabet size grows exponentially. $\Sigma' \approx |\Sigma|^{d_1^t}$.
   (c) The gap increases by a factor of t. $gap' \geq \frac{t}{O(1)} * min(gap, \frac{1}{t})$.
4. **Alphabet Reduction** In this step we reduce the alphabet size to the original alphabet size. We do this step so that we can repeat the amplification process multiple times.
   (a) Size of the graph only increases linearly. $|size'| = O(size)$.
   (b) The alphabet size is reduced to the original size. $|\Sigma'| = |\Sigma_0|$.
   (c) The gap goes down slightly. $gap' = \frac{gap}{O(1)}$.

In Dinur's paper, the first and second stages are clubbed and are referred to as preprocessing. The following subsections describe each of these steps in more detail.

Term	Definition
G	The input graph
G'	An Intermediate graph during the transformation.
Σ	Alphabet
Σ'	Intermediate alphabet during the transformation.
E	Edge set of the graph G
E'	Edge set of the graph G'
n	number of vertices in the graph
d_1	Some fixed constant independent of the input.
λ	Second eigen value
t	$t \in N$. We raise the graph G to this power.
$cloud(u)$	A group of vertices that replace the vertex u
d_0	Some fixed constant independent of the input.
α	The best assignment of colors to the graph G
$\alpha(u)$	Plurality of $\alpha'(u)$
S^u	Those vertices where α' disagrees with $\alpha(u)$
Ω	Expansion factor of the expander graph
γ	$\frac{\lvert F\rvert}{\lvert E\rvert}$

Table 6.1: Symbol Table

6.1.1 Sparsification (Degree Reduction)

In this step we replace every vertex v of G by a "*cloud*" of degree(v) vertices. Each of those vertices has degree d_0 within the cloud. Each edge $(u, v) \in G$ is replaced by an inter-cloud edge. The resultant graph is a $d_0 + 1$ regular graph with each vertex incident on exactly one inter cloud edge and there is only a constant factor increase in the size of the graph.

$$\begin{aligned} |E'| &= \frac{d_0|V|}{2} + |E| \\ |E'| &\leq (d_0 + 1) \cdot |E|. \end{aligned} \tag{6.2}$$

Lemma 54 *After sparsification, the gap of new graph G' is $gap' \geq \frac{gap}{O(1)}$*

Proof: Let α' be the best assignment of colors for (G', C') and let the extracted assignment for the original graph G be $\alpha : V \to \Sigma$ for a given edge $(u, v) \in G$. Take $\alpha(u)$ to be color which is assigned to the maximum number of vertices in $cloud(u)$ under α'(we call this the plurality of $\alpha'(u)$). We know that α violates coloring of at least $gap * |E|$ edges in (G, C) (by definition of gap). Let S^u be those vertices where α' disagrees with $\alpha(u)$ in $cloud(u)$ (not in plurality). Let $e = (u, v)$ be one of the gap edges violated by α and let e' be its corresponding inter cloud edge in G'. Now, either e' violates α' or one of the end points of e' belongs to either S_u or S_v. Thus,

$$gap * |E| \leq \text{number of edges violated by } \alpha' + \sum_{u \in V} |S^u|$$

Now we have two cases

1. The number of edges violated by $\alpha' \geq \frac{gap}{2}|E|$.

   We know that,

$$\begin{aligned} |E'| &= |E| * (d_0 + 1) \\ gap' * |E'| &\geq \frac{gap * |E|}{2} \\ gap' * |E'| &\geq \frac{gap * |E'|}{2 * (d_0 + 1)} \\ gap' &\geq \frac{gap}{2 * (d_0 + 1)}. \\ gap' &= \frac{gap}{O(1))} \end{aligned} \tag{6.3}$$

2. For each color a let S_u^a be those vertices that are assigned a in the original assignment α. Since we have the non plurality vertices,

$$\frac{|S_u^a|}{|cloud|} \leq \frac{1}{2} \tag{6.4}$$

   Since cloud is an expander, the number of edges going out from S_u^a has to be $\Omega * S_u^a$. Where Ω is the expansion factor of the graph.

$$\begin{aligned} gap' * |E'| \geq & \sum_{u \in V} \sum_{a \in \Sigma} \frac{O(1)}{\Omega} * |S_u^a| \\ \geq & \frac{1}{O(1)} \sum_{u \in V} |S_u| \qquad \text{Since we know that } |\Sigma| \text{ is constant} \\ \geq & \frac{1}{O(1)} \frac{gap|E|}{2} \\ \geq & \frac{gap|E'|}{O(1) \cdot 2(d_0 + 1)} \\ gap' = & \frac{gap}{O(1)} \end{aligned} \tag{6.5}$$

■

6.1.2 Expanderizing

In this step we superimpose our graph (G, C) with a constraint graph $(H, \emptyset)$ by taking the union of the two graphs. Here, H is a (n, d_1, λ_0) expander.

1. $|E_{G'}| = \frac{n(d_0+d_1)}{2}$ and $|E| = \frac{nd_0}{2}$. Therefore $\frac{|E'|}{|E|} = O(1)$. There is only a constant factor increase in the size.

2. $\Sigma' = \Sigma_0$. There is no increase in the size of the alphabet.

3. $gap' \geq \frac{gap}{O(1)}$. Since we have superimposed a graph with null constraints, the gap is maintained.

4. The resultant graph is an expander. We refer to the theorem on the union of a regular graph and an expander in the notes of Lecture 7-9 for this.

Note here that d_1 is a constant and doesn't depend on the input. Refer to the notes of Lecture 7-9 for the theorem which talks about the existence of such expanders for any constant.

6.1.3 Amplification

This step amplifies the gap by raising the graph to the power of some constant $t \in N$. Now we describe the construction of the new constraint graph $(G'(V', E'), C')$.

- V' is the same as V.
- E' - Pick a random vertex a, do a post-RW[2] from a ending on b. This generates a weighted edge (a, b) in E'.
- C' - Instead of directly defining the constraint, we describe a procedure that tests whether a constraint is valid or not. Here is the procedure used to test whether a given assignment $\alpha' : V' \to \Sigma'$ is valid in C'.
  1. If the number of steps in the post-RW is greater than $B = (10log|\Sigma|)t$ then it is valid.
  2. Otherwise for each step (u, v) along the post-RW path the distance
  $$distG(a, u) \leq t$$
  $$distG(b, v) \leq t$$
  if $(\alpha'(a)_u, \alpha'(b)_v)$ violates the old constraints $C(u, v)$ then not valid.
- The alphabet $\Sigma' = \Sigma^{d^t}$.

After these transformations, we have two problems

1. The edges are weighted. To fix this, we replace each weighted edge by w parallel edges where w is an integer proportional the weight of the edge.

2. There are edges between all possible pairs. To fix this, we throw away paths that are of length greater than B. By throwing away, we mean we make those constraints always satisfied. Now, the transformed graph has a constant degree and its size has only increased linearly.

[2]post-RW is an post stopping Random Walk. For a discussion on Random Walks refer to Appendix 6.2.2.

Effect on gap:
Completeness: if $gap = 0$ then $gap' = 0$.
Soundness: if $gap \neq 0$ we want $gap' = \frac{t}{O(1)} gap$.
For the soundness analysis, we first give a proof sketch and later flesh out the details.
Proof Sketch
From the best assignment α' in G' extract a corresponding assignment α in G as follows.

Extraction:

- Pick $v \in V$, and all w's such that $|(v, w)| \approx t$.
- To define $\alpha(v)$ given a failing assignment α', do a pre-RW from $v \to w$ under the condition that path length is less than t. This gives a probability distribution on vertices w.
- Take the plurality of all $\alpha'(w)$ to get $\alpha(v)$.

We know this assignment of α to G violates a set of edges $F \subseteq E$.

For a path $a \to b$ in G passing through an edge $(u, v) \in F$, we want to say that there is a good chance that $\alpha'(a) = \alpha(u)$ and $\alpha'(b) = \alpha(v)$. This is of course only true if a and b are at a distance of about t. Given some edge $e'(a, b) \in G'$, we can say that

$$\begin{aligned}
gap' &= prob[e' \text{ violates } \alpha'] \\
&\geq \frac{1}{O(1)} P_{e'}[e' \text{ passes through an edge in F}] \\
&\geq \frac{1}{O(1)}(1 - P_{e'}[e' \text{ misses F}]) \\
&= \frac{1}{O(1)}(1 - \frac{|F|}{|E|} \cdot f(\Omega, t)) \qquad \text{Using Lemma 55} \\
&= \frac{1}{O(1)}(1 - (1 - t\gamma)) \\
&= \frac{t}{O(1)}\gamma
\end{aligned} \tag{6.6}$$

Fleshing out the details of the proof

Now we discuss why the probability of an edge missing F is proportional to $\frac{|F|}{|E|}$. Let F be the set of original failing constraints in extracted α'. Throw away enough of F such that

$$\begin{aligned}
\frac{|F|}{|E|} &= min(gap, \frac{1}{t}) \qquad \text{Let's call this ratio } \gamma \\
\frac{|F|}{|E|} &= \gamma
\end{aligned} \tag{6.7}$$

A failing step: If $a \to b$ is a random path chosen by validity test, a step $u \to v$ is a failing step if it satisfies all of

1. $(u, v) \in F$
2. $dist_G(a, u) \leq t$

3. $dist_G(v, b) \leq t$
4. $\alpha'(a)_u = \alpha(u)$
5. $\alpha'(b)_v = \alpha(v)$

A $failing^*$ step: A step $u \to v$ is a $failing^*$ step if

1. if it is failing.
2. number of steps in overall $a \to b$ walk $\leq$ B.

We define the following random variables based on a post-RW corresponding to (G', C') from $u \to v$
N - Number of failing steps
N^* - Number of $failing^*$ steps
N_F - Number of steps that are in F

$$N^* \leq N \leq N_F$$

FACT:

$$gap' = \text{prob (validity test } c' \text{ rejects } \alpha') \geq prob(N^* > 0)$$

From the Second moment method, we have

$$gap' \geq \frac{E[N^*]^2}{E[(N^*)^2]}$$

Lemma 55

1. $E[N^*] \geq \frac{1}{8|\Sigma|^2}\frac{|F|}{|E|}$ *This is because of the way we extracted α from α'*
2. $E[(N^*)^2] \leq O(1)t\frac{|F|}{|E|}$ *This is because of the expansion property of G (See lemma 64).*

Using these two lemma get the probability of an edge missing F as proportional to $\frac{|F|}{|E|}$.

6.1.4 Alphabet Reduction

The amplification step increases the gap but it leads to a blow up in the size of the alphabet. In order to be able to repeat amplification, we need to reduce the alphabet down to the original size. This is done using two key ideas viz. modularization and composition.

We take the constraint graph G and Booleanize each edge e to get a corresponding boolean formula Φ. Note that booleanizing the constraints reduces the alphabet size from d^t to 2. Now, each of these clauses is fed to a q-query assignment tester to get a set of constraints Ψ. Constraints in Ψ have q variables and an alphabet size of 2. Now, we convert the constraints of the q-query assignment tester to constraints of a 2-query assignment tester κ. The 6-query assignment tester can be converted to a 2-query assignment tester, by simply augmenting Y with a new set of variables Y'. This allows us to easily plug in the constraints

back to the original graph by treating the two variables as the label assignments of 2 vertices at the ends of an edge.

Note that due to the decoupling in the composition step, it is possible that although there is no assignment of labels a^* that satisfies more than s fraction of constraints Φ simultaneously, in fact, each of the Φ's could be satisfiable, and therefore the collection of Ψ's could be simultaneously satisfiable, since they have been decoupled.

This is taken care of by the fact that, the "inner" PCP reduction (the 2-query AT) has much stronger completeness and soundness than the usual PCP reduction. Though it does not need to be a polynomial time reduction since it's domain is effectively constant sized inputs (namely constraints Φ on single edges).

The 2-query AT is a stronger than usual PCP reduction in the sense of completeness and soundness: For ANY assignment a^* to the Boolean variables of Φ:

- If a^* satisfies Φ, then there is an assignment b to the auxiliary variables, such that $\Psi(a, b)$ is satisfied; and
- If a^* is δ-far from any satisfying assignment of Φ, then for EVERY assignment b^* to the auxiliary variables at least δ fraction of constraints in Ψ are NOT satisfied by (a^*, b^*).

To see why the above is much stronger than the usual PCP reduction, observe that the usual PCP reduction would merely ensure this:

- If there is an assignment a that satisfies Φ, then there is an assignment b such that $\Psi(a, b)$ is satisfied; and
- If all assignments a satisfy at most s-fraction of the clauses of Φ, then all assignments b are such that (a^*, b^*) satisfies at most s'-fraction of constraints in $\Psi(a^*, b^*)$.

The inner PCP reduction also ensures that the output constraints have only 2 variables each. These 2 variables are taken from the Boolean variables of Φ as well as a set of auxiliary variables b over the original small alphabet Σ_0. Since there are only 2 variables, they can be treated as the labels of 2 vertices at the ends of an edge, thus the set of constraints can be converted to an inner Graph-CSP.

Both of the above properties of the inner PCP are crucial for the composition of the outer graph-CSP with the inner CSP's that have been obtained independently in a modularized manner for each edge.

Theorem 56 *If we have a 2 query assignment tester $AT(\gamma > 0, \Sigma_0)$ then $\exists \beta > 0$ depending only on AT and a poly(G) such that any constraint graph (G, C) can be transformed in time polynomial in G into a constraint graph (G', C') such that*

1. *size(G') = O(1) size(G)*
2. *if gap(G) = 0 then gap(G') = 0*
3. *if $gap(G) \neq 0$ $gap(G') \geq \beta \cdot gap(G)$*

Proof:

1. We first prove that the size of the new graph is linear in the size of the input graph. For this we make the following observations

   (a) In the first step of the transformation, we convert every edge of the Graph G into a Boolean formula. Since, the size of the formula depends on $|\Sigma|$ and not the input graph, the size of the output graph is still linear in the input graph.

   (b) The"inner" PCP reduction (the 2-query AT) ensures that the output constraints have only 2 variables each. In addition to this, it puts a constant bound on the output depending only on Σ. Thus the complexity of the AT itself doesn't affect the size of G.

2. If $gap(G) = 0$ then $gap' = 0$.

3. For this we need to prove that every assignment for G' violates at least $\beta \cdot gap$ fraction of G''s constraints. Now we pick the best assignment α' of colors to G'. Let F be the edges where the assignment of colors disagrees. Now by the definition of gap, we have $gap' = \frac{|F|}{|E|}$. Now we need to show that β constraints of the graph G_F are falsified by α', the best assignment of colors in G'. We know that the constraint graph G_F is generated by the assignment tester AT on input Φ from the graph G. Therefore to find the gap, we must analyze the gap in the Φ.

$$\begin{aligned} gap(G') &= \frac{1}{|E|}\sum_{e\in E} gap(G_F) \\ &= \frac{1}{|E|}\sum_{e\in F} gap(G_F) \\ &\geq \beta\frac{|F|}{|E|} \\ &\geq \beta gap(G) \end{aligned} \tag{6.8}$$

■

In the paper, a concrete 2-query assignment tester is constructed, which uses a generalization of the ideas of linearity (homomorphism) testing, self-correction of Hadamard codes, so-called long-code, tools common to locally decodable and list-decodable codes. The analysis of the completeness and soundness of certain associated constant-query tests uses the tools of Fourier analysis of Boolean functions. This, and the composition steps will be discussed in a later set of notes.

6.2 Appendix

6.2.1 Background

In this section we give definitions, theorems and tools that will be used in the proof of the PCP theorem by gap amplification.

Definition 57 (Constraint Satisfaction Problem (CSP)) *A CSP is (G,C) where* $C = \{C_e : e \in E(G)\}$ *where* $C_e \subseteq \Sigma \times \Sigma$. *Where* Σ *is not part of the input.*

Definition 58 (Decision CSP) *A decision problem on CSP is to find an assignment of colors in* Σ *to vertices such that* $\forall e \in E$, *when* $e = (u,v)$, $\alpha(u), \alpha(v) \in C_e$.

Definition 59 (Maximization CSP) *The maximization prolem on CSP is to maximize the set of edges on which* $\alpha(u) \in C_e$ *where* $e \in (u, v)$.

Definition 60 (GAP CSP) *The GAP version of CSP is*

$$CSP \in GAP\_CSP_{c,s} \implies \left\{ \begin{array}{ll} \geq & c \text{ fraction of the constraints are satisfiable.} \\ & or \\ < & s \text{ fraction of the constraints are satisfiable.} \end{array} \right\}$$

6.2.2 Random Walks

In this section we define two ways of doing a random walk and some related properties.

Definition 61 (post Stopping Random Walk(post-RW)) *In a regular graph* $G = (V, E)$

1. *Pick a random vertex* $u \in V$.
2. *Take a step along a random edge.*
3. *Stop with probability* $\frac{1}{t}$.
4. *Otherwise go to step 2.*

Definition 62 (pre Stopping Random Walk(pre-RW)) *In a regular graph* $G = (V, E)$

1. *Pick a random vertex* $u \in V$.
2. *Stop with probability* $\frac{1}{t}$.
3. *Otherwise take a step along a random edge.*
4. *Go to step 2.*

Lemma 63 relates the two kinds of random walks described.

Lemma 63 *For some constant* $k \geq 1$ *if (u, v) is a fixed edge in a regular graph G. If we do a post-RW conditioned on walking the edge (u, v) exactly k times then the following are true.*

1. *Distribution on final vertex is the same as if we did a pre-RW starting from u.*
2. *Distribution on the starting vertex is the same as if we did a pre-RW starting from u.*
3. *(1) and (2) are independent of each other.*

Lemma 64 *If G is a* (n, d, λ) *expander and* $B \subseteq V(G)$ *is a set such that* $\frac{|B|}{|V|} = \beta$ *then probability that a q step walk starting from a to vertex v leaves B is*

$$\leq (\sqrt{\beta^2 + (\frac{\lambda}{d})^2(1-\beta)^2})^t$$

Chapter 7

Strict NP and the Approximibility of NP-Complete Problems

Lecture(s): 19–22
Date(s): 2/16, 2/18, 2/23
Lecturer: Serdar Ayaz
Scribe: Kiara Hall

7.1 Circuits and Logic

Throughout the paper, we will assume that $P \neq NP$.

The following theorems will show equivalencies between complexity classes and objects in mathematical logic from [20]. In particular we will talk about models of first order, and second order logic. Formally, a model is an $\mathcal{L}$-structure that satisfies a set of first order sentences of $\mathcal{L}$. We write $\mathcal{M} \models \phi$ for $\phi \in Sent(\mathcal{L})$ if M contains elements that satisfy ϕ. Note, that the main difference between first and second order logic is that first order quantifiers can range over M while second order quantifiers can range over $\mathcal{P}(M)$. More accurately, second order quantifiers range over structures of the model.

Example 2 *Roughly, we can consider the how the equation $x + x = 1$ can be satisfied in $\mathbb{Q}$ using $x = \frac{1}{2}$. However $\mathbb{Z}$ can't satisfy this sentence as $\frac{1}{2} \notin \mathbb{Z}$.*

$$\mathbb{Q} \models \exists x(x + x = 1)$$
$$\mathbb{Z} \not\models \exists x(x + x = 1)$$

To use a second order quantifier, consider the ϕ to be the sentence $(2|x \wedge x \in S)$. Then the subset $2\mathbb{Z}$ of $\mathbb{Z}$ would satisfy this ϕ so,

$$\mathbb{Z} \models \exists S \forall x(2|x \wedge x \in S)$$

That is a little formal, we can also consider models as graphs. Let $G = (V, E)$ and let ϕ be the first order sentence for transitivity. Then,

$$G \models (\forall a, b, c, ((aEb \wedge bEc) \rightarrow aEc)) \iff G\text{'s connected compents are cliques}$$

Definition 65 *AC^0 is the class of languages recognized by a family of Boolean Circuts C with a universal depth, $O(log(n))$, regardless of the input size.*

Theorem 66 *$AC^0 = FO$*

That is to say,

$$S \in AC^0 \iff \exists \text{ a first order sentence } \phi \text{ s.t. } S = \{G : G \models \phi\}$$

Theorem 67 *$P = FO + LFP$*

That is,

$$S \in P \iff \exists \text{ a first order sentence } \phi \in \text{FO+LFP s.t. } S = \{G : G \models \phi\}$$

Theorem 68 (Fagin's Theorem) *$NP = SO\ \exists$*

That is,

$$\begin{aligned} S \in NP &\iff \exists \text{ a first order } \phi \text{ with a second order } \exists \text{ s.t. } S = \{G : G \models \phi\} \\ &\iff \exists S \forall x \exists y \psi(x, y, G, S) \end{aligned}$$

7.2 Strict NP

Fagin's theorem is important in descriptive complexity, and illustrates that every problem in NP can be expressed without actually using computation. Knowing this, we can manipulate this form and express different classes of problems. This will be the focus of Papadimitiou and Yannakakis' results [46].

Example 3 *SAT can be written in this form:*
To write this properly, we have to define two relations. We say $P(x, c)$ is true if the variable x appears positively in the clause c. We say $N(x, c)$ is true if the variable x appears negatively in the clause c. Say, if $c = (x \vee y \vee \overline{z})$, we have that $P(x, c)$, $P(y, c)$ and $N(z, c)$ hold. So, SAT can be written as follows,

$$\exists T \forall c \exists x [(P(x, c) \wedge x \in T) \vee (N(x, c) \wedge x \notin T)]$$

We can write it without using the relation P. Instead, we can use a relation $R(x, c)$. We say $R(x, c)$ is true if the variable x appears in the clause c. Now we can write SAT as follows,

$$\exists T \forall c \exists x [(R(x, c) \wedge \neg N(x, c) \wedge x \in T) \vee (R(x, c) \wedge N(x, c) \wedge x \notin T)]$$

Notice that SAT can be expressed in two different ways, but the format does not change, only the relations. In other words, the ϕ is not unique.

Definition 69 *Strict NP is the class of problems which can be expressed as $\exists S \forall x \phi(x, G, S)$. Here, ϕ is a quantifier free, S is a structure dependent on the model G and $x \in G$.*

Example 4 *3SAT is a Strict NP problem:*

$$\begin{aligned}\exists T \forall (x_1, x_2, x_3)[&(C_0((x_1, x_2, x_3)) \rightarrow x_1 \in T \vee x_2 \in T \vee x_3 \in T) \\ &\wedge (C_1((x_1, x_2, x_3)) \rightarrow x_1 \in T \vee x_2 \in T \vee x_3 \notin T) \\ &\wedge (C_2((x_1, x_2, x_3)) \rightarrow x_1 \in T \vee x_2 \notin T \vee x_3 \notin T) \\ &\wedge (C_3((x_1, x_2, x_3)) \rightarrow x_1 \notin T \vee x_2 \notin T \vee x_3 \notin T)]\end{aligned}$$

Definition 70 *For each predicate $\Pi \in NP$ of the form $\exists S \forall x \exists y \psi(x, y, G, S)$, we define max$\Pi$, the maximization version of Π, as*

$$\max \Pi = \max_S |\{x : \exists y \psi(x, y, G, S)\}|$$

MAX NP is the class of all such maximization problems. For each predicate $\Pi' \in SNP$ of the form $\exists S \forall x \phi(x, G, S)$, we define max$\Pi'$ as

$$\max \Pi' = \max_S |\{x : \phi(x, G, S)\}|$$

MAX SNP is the class of all maximization problems for $\Pi' \in SNP$.

Note that the maximization is not dependent on the size of S.

Definition 71 *Let OPT be the value of the optimal solution. An algorithm achieves an approximation ratio $\alpha \geq 1$ for a maximization problem if for every instance, it produces a solution of cost $\geq OPT/\alpha$. For a minimization problem, achieving a ratio α involves finding a solution cost $\leq \alpha OPT$.*

Definition 72 *A fully polynomial time approximation scheme, FPTAS, is an algorithm that, given input x and error bound $\epsilon > 0$, computes a solution of cost at most $(1 + \epsilon)OPT$ in time that is polynomial in $|x|$ and $1/\epsilon$. A polynomial time approximation scheme, PTAS, is an algorithm that, for any fixed $\epsilon > 0$ can achieve an approximation ratio $1 + \epsilon$ in time that is polynomial in $|x|$ but may not be polynomial in $1/\epsilon$.*

Notice that these approximation algorithms are defined specifically for optimization problems. Further, not all optimization problems have an approximation. For example, KNAPSACK has FPTAS while TSP is not approximable.

Proposition 73 *MAX3SAT is approximable within a constant factor in polynomial time.*

Proof: Consider a formula ϕ in MAX3SAT. Without loss of generalization, this formula has exactly 3 literals per clause and does not repeat a literal. There are $2^3 = 8$ independent random assignments to the 3 variables in any clause. Let R denote the set of these assignments. The probability that the clause is satisfied is $\frac{7}{8}$, since it is only unsatisfied when all of the literals are false,

$$E[(x_1, x_2, x_3) = 1 \mid x_1, x_2, x_3 \in R] = \frac{7}{8}$$
$$E[(x_1, x_2, x_3) = 0 \mid x_1, x_2, x_3 \in R] = \frac{1}{8}$$

Now let m denote the number of clauses in ϕ. The expected number of satisfied clauses in ϕ is the sum of the expectations for each of these clauses. Let y be the tuple of literals for ϕ, then we see

$$E[\text{Number of satisfied clauses of } \phi \text{ given } y \mid y \subseteq R] = \sum_{i=0}^{m} E[(x_1, x_2, x_3) = 1 \mid x_1, x_2, x_3 \in R]$$
$$= \frac{7}{8}m$$

This gives us the following simple $\frac{7}{8}$ deterministic approximation algorithm, also known as the Karloff Zwick algorithm [32].

Take the first variable x_1. Consider the expected number of clauses that are satisfied when x_1 is set to 0 and when it is set to 1. Let a_1 denote the expected number of satisfied clauses when $x_1 = 0$ and b_1 denote the expected number of satisfied clauses when $x_1 = 1$. By the above, these two average to $\frac{7}{8}m$,

$$\frac{a_1 + b_1}{2} = \frac{7}{8}m$$

Thus either $a_1 \geq \frac{7}{8}m$ or $b_1 \geq \frac{7}{8}m$. If $a_1 \geq \frac{7}{8}m$ then set $x_1 = 0$. If $b_1 \geq \frac{7}{8}m$, then set $x_1 = 1$.

Now consider x_2 using the same method, except now we have the added restriction of x_1's new assignment. Again we will see that either $a_2 \geq \frac{7}{8}k$ or $b_2 \geq \frac{7}{8}k$ for some constant $k \leq m$, and as such we can assign the appropriate value for x_2. Continue this process for $x_3 \ldots x_n$, each time either increasng or maintaining the expected number of clauses that are satisfied. In the end, we have an assignment for all literals, and the expected value for the number of clauses we have satisfied with this assignment is at least $\frac{7}{8}m$. Computing these expectations will take exponential time. So to make the algorithm polynomial time, note that we don't need to pick the variable assignments independently at random. It is sufficient to pick the assignments of the 3 variables in a clause independently at random, which will happen if the assignments are 3-wise independent. It is not hard to show that there is a space of n^3 assignments so that if we pick one uniformly at random, then for each variable, the probability of being set to 1 is exactly half, and the assignment to the variables is 3-wise independent. Using this set of assignments to compute the expectations, our algorithm becomes an n^3m time algorithm. ■

7.3 L-Reduction

Now, it seems natural that if a problem has a FPTAS or PTAS, a reduction of that problem should preserve that property. However, most problem reductions do not create or preserve the gap in the cost function seen in PTAS. This is the motivation behind creating the L-reduction.

Definition 74 *Let Π and Π' be two optimization (maximization or minimization) problems. We say that Π L-reduces to Π' if there are two polynomial-time algorithms f, g and constants $\alpha, \beta > 0$ such that for each instance I of Π we have*

- *Algorithm f produces an instance $I' = f(I)$ of Π' such that the optima of I and I', $OPT(I)$ and $OPT(I')$ respectively, satisfy $OPT(I') \leq \alpha OPT(I)$.*

- *Given any solution of I' with cost c', algorithm g produces a solution of I with cost c such that $|c - OPT(I)| \leq \beta |c' - OPT(I')|$.*

The L-reduction was created specifically for maximization problems, and does not preserve the gap for minimization problems.

Proposition 75 *1. L-reductions compose.*

2. If Π L-reduces to Π' and there is a PTAS for Π' with worst case error ϵ, then there is a PTAS for Π with worst case error $\alpha\beta\epsilon$.

Theorem 76 *MAX3SAT is MAXSNP-complete.*

Proof: As 3SAT is a SNP problem, naturally MAX3SAT is in MAXSNP. It only remains to be shown that it is MAXSNP-hard.

Consider a problem in MAXSNP, then it is in the form $\exists S \forall x \phi(x, G, S)$. Similar to the proof for proposition 0.13, we can enumerate all the possible values of x. We can ignore the values fro which ϕ is unsatisfiable, and write the first order part as a conjunction:

$$\phi_1 \wedge \cdots \wedge \phi_m$$

Here, each conjuct is for a value of x that satisfies ϕ. For each ϕ_i we can introduce auxiliary variables and construct a set C_i that contains causes of at most three literals. We can do so by viewing ϕ as a (bounded) circuit and then add an element g for every gate of ϕ as follows,

$$\begin{aligned}
&\neg \text{ gate with input } a: \text{ include the clauses } g \vee a, \overline{g} \vee \overline{a} \\
&\wedge \text{ gate with input } a, b: \text{ include the clauses } a \vee \overline{g}, b \vee \overline{g}, g \vee \overline{a} \vee \overline{b} \\
&\vee \text{ gate with input } a, b: \text{ include the clauses } \overline{a} \vee g, \overline{b} \vee g, \overline{g} \vee a \vee b
\end{aligned}$$

Lastly, if h is an output gate we include a clause for h. Now we have added these clauses to our set C_i, the set itself has the property that any truth assignment τ to the variables of ϕ_i (the inputs to the circuit), can be extended to the auxilary variables so that it satisfies all the clauses, except possibly for the clause h that corresponds to the output gate. We treat the output gate seperately, since τ may not satisify ϕ_i.

Now, by construction our instance of MAX3SAT contains all theses sets of clauses $C_1, \ldots, C_m$ for all values of x. Now we must show that this transformation f is a L-reduction. Since the ϕ_i's are of bounded size, so are the C_is and the total number of clauses is at most $k_1 m$ for some constant k_1. If there are $OPT(I)$ values of x satisfying ϕ in the optimal solution for the instance I of the MAXSNP problem, then the optimal solution for the MAX3SAT translation of the problem for instance $f(I)$ satisfies $OPT(f(I)) = \sum_i (|C_i| - 1) + OPT(I)$ clauses. We know that $OPT(I)$ is at least a constant fraction of

the size of the problem, m. So $OPT(I) \geq \frac{m}{k_2}$ for some constant k_2, thus $m \leq k_2 OPT(I)$. Finally, let $\alpha = (k_1 - 1)k_2 + 1$ and $\beta = 1$. Then we see,

$$\begin{aligned} OPT(f(I)) &= \sum_i (|C_i| - 1) + OPT(I) \\ &\leq \sum_i (k_1 - 1) + OPT(I) \\ &= mk_1 - m + OPT(I) \\ &\leq k_2 OPT(I) k_1 - k_2 OPT(I) + OPT(I) \\ &= (k_2 k_1 - k_2 + 1) OPT(I) \\ &= (k_2(k_1 - 1) + 1) OPT(I) \\ \therefore OPT(f(I)) &\leq \alpha OPT(I) \end{aligned}$$

The other condition follows automatically and we see that f is an L-reduction. ■

MAXSNP, MAXNP and L-reductions are only defined for maximization problems. We can expand on some of these ideas to include minimization problems as well.

Definition 77 *An AP-reduction or an approximation preserving reduction, is defined such that if problem* Π *is AP-reducible to a problem* Γ*, and* Γ *is approximable up to a factore* $1+\alpha$*, then wecan conclude that* Π *is approximable up to a factor* $1 + O(\alpha)$*. The class APX, is the class of all NP optimization problems that allow polynomial-time approximation algorithms.*

These are a generalization of MAXSNP to include minimization problems as well. Notice that if $\Gamma \in$APX, and Π is AP-reducible to Γ, then $\Pi \in$APX as well.

Theorem 78 *PTAS is contained in MAXSNP.*

The gap amplification in Dinur's proof of the PCP theorem tells us there is a universal constant S so that MAX3SAT problems do not have a better than s factor approximation algorithm. Recall Theorem 31, which leads to the conclusion that a $MAXSNP$-complete problem does not have a $PTAS$, or $FPTAS$, unless $P = NP$.

Chapter 8

Unique Games Conjecture

Lecture(s): 23–27
Date(s): 2/25, 3/8, 3/10
Lecturer: Joel Willoughby
Scribe: Joel Willoughby

8.1 Introduction

For these notes, we introduce a popular recent conjecture known as the Unique Games Conjecture or UGC [36]. As we will show, though just a conjecture (like $P \neq NP$), it has attracted a lot of attention in the literature and has generated many results in seeking evidence both for it and against it (often in the same paper!). Perhaps the most exciting result of this nature comes from the relation of UG to a rather large class of approximation algorithms. We will present a result from Raghavendra [49] which guarantees the best possible poly-time approximation for such problems contingent on the UGC.

Furthermore, research into UGC has enriched other areas that have long been considered difficult to attack. We will look at one such instance, so-called l_2^2-embeddability.

We assume prior knowledge of Semi-Definite Programming or SDP as it is a popular tool in the space we will be working in. We suggest looking into [47] for a primer.

8.2 Integrality Gaps and Approximation

To build up to the statement of UGC and the results surrounding it, we first provide an introduction to approximation algorithms – more specifically to a general approximation strategy known as Semidefinite Programming. We will use the Max-Cut problem as a running theme in this section. The Max-Cut of a graph $G = (V, E)$ is a partition of V into $V_1 \cup V_2$ such that $|\{(i, j) \in E | i \in V_1, j \in V_2\}|$ is maximized. It is well known that finding Max-Cut is NP-hard. Instead, we turn our attention to coming up with an approximation to Max-Cut. The work in this problem was originally done by Goemans and Williamson [24].

Max-Cut can be formulated as an Integer Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in E} \frac{1 - v_i v_j}{2} \\ \text{subject to} \quad & v_i \in \{-1, 1\} \end{aligned} \tag{8.1}$$

It is clear that we can immediately get a Max-Cut for a graph G from a solution to this problem. However, this is still NP-hard to solve. Our problem is that we are enforcing v_i to have integer solutions. If instead, we allow a little some "slack" or uncertainty in our v_i's (for instance, if we allow $v_i \in [-1, 1]$), the optimization problem becomes easier, with a loss of accuracy in our original Max-Cut problem. This is called a *relaxation*. In our case, we are interested in the Semi-Definite relaxation of Max-Cut:

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in E} \frac{1 - \langle v^i, v^j \rangle}{2} \\ \text{subject to} \quad & \langle v^i, v^i \rangle = 1 \end{aligned} \tag{8.2}$$

where $\langle x, y \rangle$ is just the standard inner product of 2 vectors. Now, we are allowing our v_i's to be unit vectors in some dimension. Note that if we add in the constraint that all vectors lie in the same direction, we get our original integer formulation back (those pointing in the positive direction are $+1$, those negative are -1).

Let `OPT` be the solution to the actual Max-Cut problem (and so the solution to (8.1)) and `SDP` be the solution to (8.2). It is clear that $\texttt{SDP} \geq \texttt{OPT}$. In practice we want `SDP` and `OPT` to be as close as possible so we don't lose too much by our relaxation.

Definition 79 (Integrality Gap) *Using the definitions of* $\texttt{SDP}, \texttt{OPT}$ *above, the integrality gap for an SDP relaxation of an IP is:*

$$IG = \texttt{SDP}/\texttt{OPT}$$

From our discussion so far, in order for a relaxation to be useful, IG should be close to 1. To put it more formally:

Observation 80 (Integrality Gap and Approximation) *If a relaxation to a problem has and IG of* α*, an algorithm using that relaxation is at best an* α*-approximation*

There is a further problem to be addressed here. The solution to (8.2) gives us an approximate *value* for the Max-Cut. It is not immediately obvious how to turn that solution to an actual cut on the graph. To answer this, we look at the original Goemans and Williamson algorithm which states that given vectors $\{v_i\}$ that maximize (8.2), to get a cut for G, simply choose a random hyperplane through the origin. If n is the normal vector to the hyperplane, partition vectors based on whether $\langle n, v_i \rangle > 0$ or $\langle n, v_i \rangle < 0$. Let the value gotten from the cut acquired in the manner be denoted `ALG`. Then, it is clear that:

$$\texttt{SDP} \geq \texttt{OPT} \geq \texttt{ALG} \tag{8.3}$$

In their original paper, Goemans and Williamson [24] bounded $\texttt{SDP}/\texttt{ALG}$. We present a summary of their proof in Section 8.2.1. This doesn't give us the whole story, however,

as there are graphs for which the relaxation meets this bound (and hence the algorithm performs well) and also those for which the integrality gap is 1 and the algorithm performs poorly. To measure the tightness of the SDP relaxation, we want to bound SDP/OPT. This was done explicitly in [21] and we summarize it in Section 8.2.2. To measure the tightness of the GW algorithm, we want to bound OPT/ALG. This was done explicitly in [31] and we summarize it in Section 8.2.3.

Observation 81 *The following results on the tightness of the GW algorithm for* Max-Cut only *hold for the GW algorithm. It has not been ruled out that there is no better algorithm for* Max-Cut. *Indeed as we will see in Section 8.3.1, this is one of the more tantalizing results contingent on UGC.*

8.2.1 Goemans and Williamson: Bound on SDP/ALG

In this section, we show the following:

Theorem 82 (SDP/ALG Bound [24])

$$\frac{\texttt{SDP}}{\texttt{ALG}} \leq \frac{1}{\alpha_{GW}}$$

where $\alpha_{GW} \approx 0.878$

To do this, we note how much a single edge $e = (i, j)$ can contribute to SDP and to ALG. Suppose the SDP solution has vectors v_i and v_j "assigned" to i and j. Then, the contribution of e to SDP is $1/2 - \langle v_i, v_j \rangle/2 = 1/2 - \cos\theta/2$ where θ is the angle between v_i and v_j.

Furthermore, since the algorithm takes a random hyperplane and separates e when v_i and v_j are on opposite sides of the hyperplane, the expected contribution of a single edge to ALG is just the probability v_i and v_j are separated. This is θ/π.

Then, we consider the quantity $\frac{1/2-(\cos\theta)/2}{\theta/\pi} = \frac{\pi-\pi\cos\theta}{2\theta}$. We note that this attains a maximum of $1/\alpha_{GW}$ when $\theta \approx 2.33$. Thus, the upper bound of SDP/ALG follows when we consider the case that each edge attains this bound.

8.2.2 Feige and Schechtman: Tightness of SDP/OPT

In the next couple of sections, we show the tightness of Theorem 82. To do this, we describe graphs for which $\texttt{SDP}/\texttt{OPT} \approx \alpha_{GW}^{-1}$ as well as graphs for which $\texttt{OPT}/\texttt{ALG} \approx \alpha_{GW}^{-1}$.

In this section, we present a graph that meets the SDP/OPT bound. More formally:

Theorem 83 (Tightness of SDP/OPT [21]) *For every* $\epsilon > 0$, $\exists G$ *such that*

$$\texttt{SDP}/\texttt{OPT} \geq \alpha_{GW}^{-1} - \epsilon$$

Combining this with Theorem 82, we get as a corollary that on such graphs G as in the Theorem, the GW algorithm actually gives us an optimal cut (i.e. $\texttt{OPT} = \texttt{ALG}$)!

We will show this result by constructing such a G. Note that we only give a general overview here. For a more complete argument, we refer the reader to the original paper [21]. The graph we construct comes from the SDP relaxation itself. We start by taking a hypersphere S in n dimensions (we will not discuss the actual number of dimensions necessary). Then, we pick sufficiently many points randomly from S. By "sufficiently many", we mean enough that the distribution of points on S is dense and uniform. It is a helpful thought experiment to think of the points we pick as *all* points on the hypersphere. The points we choose will represent our vertices.

For the edges of our graph, we find the maximum angle θ^* such that:

$$\frac{\pi - \pi\cos\theta^*}{2\theta^*} \geq \alpha_{GW}^{-1} - \epsilon \tag{8.4}$$

Then, we simply put an edge between two point p_1 and p_2 when the angle between them is at most θ^*. Immediately from construction, since the points on S as chosen make a feasible solution to the SDP relaxation (8.2), we see that $\texttt{SDP} \geq |E|(1/2 - (\cos\theta^*)/2)$.

What remains to be shown is that $\texttt{OPT}$ is at most the solution given by the GW algorithm. If we had this, then we just sum over the expected values for each edge (as we did in Section 8.2.1) to get $\texttt{OPT} \leq \theta^*/\pi$ and the result follows. In other words, we want to show that the maximum cut on such a graph G coincides with a hyperplane through the origin. To do so, we take a theorem from Feige and Schectman which is actually an older result from measure theory:

Let the optimal cut on the surface of S be A and A^C. Let a be the measure of A. Intuitively, A is a piece of the sphere and a is its area, or fraction of the vertices in A. Let $\mu_\rho(A)$ denote the fraction of the edges crossing A and A^C. I.e. $\mu_\rho(A) = Pr_{(u,v)\in E}(u \in A, v \in A^C)$. Then we have the following:

Theorem 84 (Cuts on the Sphere [21]) *Fix an a between 0 and 1 and a ρ between 0 and π. Then the maximum of $\mu_\rho(A)$ where A ranges over all (measurable) subsets of S of measure a is attained for a(ny) cap of measure a.*

We do not prove this theorem here as to do so requires some measure theory out of the scope of this course.

A cap is simply gotten from a hyperplane slicing through S and taking all points of S on one side of the hyperplane. A cap passing through the origin is a hemisphere. It is not hard to see that if we have a cut corresponding to a cap not through the origin, a larger cut can be obtained by simply taking a parallel hyperplane through the origin. This is due to the fact that the measure on a sphere is concentrated around the equators.

Hence, the maximum cut for the graph we have described is a hyperplane. This means that $\texttt{OPT} = \texttt{ALG}$. Thus, we get our bound of $\texttt{SDP}/\texttt{OPT} \geq \alpha_{GW}^{-1} - \epsilon$.

Note that we hid a few things under the rug that are addressed in the original proof in [21]. First, what dimension is S embedded in? n has to be sufficiently large for this to work. Second, how do we get a set of points that is sufficiently dense and uniform yet still finite? For answers to thee questions, please see [21].

8.2.3 Karloff: Tightness of OPT/ALG

To fully wrap up our discussion of Max-Cut, we now need to find a graph for which the GW algorithm performs poorly. In Section 8.2.2, we exhibited a G where the SDP relaxation had a high gap. In this section, we give a graph G for which the SDP relaxation turns out to yield the same solution as the original IP formulation, but the random hyperplane can be shown to yield a less than optimal cut. Formally:

Theorem 85 (Tightness of OPT/ALG [31]) *For every $\epsilon > 0$, $\exists G$ such that*

$$\texttt{OPT/ALG} \geq \alpha_{GW}^{-1} - \epsilon$$

Note the similarities to Theorem 83. Indeed, this Theorem actually appeared in 1996 [31], before the 2000 [21] result which sought to address the Integrality gap directly.

Like last section, we present a graph built from the vector space of the SDP solution space. Unlike last section, we show that a cut from a hyperplane performs particularly poorly for the set of points. Furthermore, the embedding we give for the graph turns out to be an optimal solution to the SDP.

Instead of embedding points randomly on a hypersphere, we make our vertices be the corners of the d dimensional hypercube embedded in the d dimensional hypersphere. So, $V = \left\{\frac{-1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right\}^d$. The edge set is formed connecting all vertices v_1 and v_2 when their hamming distance is a specific value q. The constraint on the hamming distance makes sense if you consider the value an edge contributes to SDP: $1/2 - \langle v_1, v_2\rangle/2$. Multiplying out and taking advantage of the known hamming distance, for any edge (v_i, v_j)

$$1/2 - \langle v_1, v_2\rangle/2 = q/d$$

We choose q to make $q/d \approx 1/2 - (\cos\theta^*)/2$, where $\theta^* \approx 2.33$. So we can conclude that all edges are at this critical angle and hence the GW algorithm when run on *this* feasible solution to the SDP problem has value at most θ^*/π.

It remains to show two things: (1) that optimum solution for this graph is q/d and (2) that this embedding of the graph optimizes the SDP formulation (i.e. SDP = OPT). To see (1), we take a cut corresponding to the first coordinate of V. In other words, V_1 consists of all vertices with the first coordinate positive and V_2 all with first coordinate negative. It is not hard to see that such a cut has a value of SDP.

Showing (2) is a lot more involved and indeed is the focus of much of [31]. We will elide its proof here. For further information, [21] and [5] give different takes on the proof.

8.3 The Unique Games Conjecture

Now that we have gotten our feet wet with integrality gaps and approximation algorithms, we now present one of the more popular topics of the area. As we shall see, either its proof or its refutation will mean much to the area. If true, we will have a tight bound for a number of optimization algorithms. If false, we will necessarily have an algorithm better than any currently known at solving these problems.

A little back-story on how The Unique Games Conjecture was formed. Essentially, the question was: what would it take to mimic the optimal inapproximability results of Hastad [27] for other specific problems on the RHS of the desired reduction that pose a bottleneck to Hastad's type of proof. Hastad's type of proof uses long code 2-prover 1-round PCP's and Parallel repetition to obtain a reduction from an NP-hard problem Q on the LHS to Gap versions of certain problems on the RHS (as in the proof in Troy's notes that "PCP theorem iff there is a reduction from 3-color to $gap3SAT$.")

The answer in many such specific bottle neck RHS problems was: choose Q (LHS) to be Unique Games (or unique label cover).

To begin with, we need some definitions:

Definition 86 (Constraint Satisfaction Problem (CSP)) *A tuple (G, C), where $G = (V, E)$ is a graph and $C : E \to P(\Sigma \times \Sigma)$ is a mapping of edges to constraints. An assignment (or evaluation) of a CSP is a mapping $\alpha : V \to \Sigma$. An assignment satisfies (or is consistent with) a constraint $C((u, v))$, where $(u, v) \in E$, if $(\alpha(u), \alpha(v)) \in C((u, v))$.*

Definition 87 (Unique Game (UG)) *A unique game on alphabet Σ is a constraint satisfaction problem (G, C) with the additional constraint that for every $e \in E$, $C(e)$ yields a bijection. In other words, $P(\Sigma \times \Sigma)$ can be written as a one-to-one correspondence.*

Definition 88 (Decision problem for a CSP) *Given a CSP, is there an assignment that satisfies all constraints?*

Definition 89 (Max Problem for a CSP) *Given a CSP, what is the maximum number of constraints that can be simultaneously satisfies by an assignment*

Definition 90 (Gap Problem for a CSP) *Given a CSP and constants $c, s \in [0, 1]$ determine which of the following hold (with a promise that at least one will):*

- *YES case: there is an assignment satisfying at least $c|E|$ constraints*
- *NO case: no assignment satisfies more than $s|E|$ constraints*

Note for our purposes, we will take $\Sigma = [n]$. We immediately see that UG is a subset of CSP. The added bijection constraint for UG allows us to solve the decision version for a UG in quadratic time (in n). Just run through all possible labels for some $v_1 \in V$. Since the constraints are bijections, this label fixes the labels of all vertices connected to v_1, which fixes all vertices connected to those and so on.

Recall from the PCP theorem notes that there is a universal constant s such that $GAPCSP_{1,s}$ is NP-hard via a reduction from $GAP3SAT$. This reduction involves an inner PCP that makes a constant number (3) of queries to its prover to determine membership. UG can be looked at as those problems that can be solved with such an inner PCP that makes only 2 queries. This is again due to the transitive property above. Such PCP reductions are not as powerful and hence not as difficult to simulate. For this reason, we cannot use the same proof method to prove hardness of $GAPUG$.

Even so, researchers have not been able to disprove this hardness either. This notion was captured by Khot [35] in what is now known as the Unique Games Conjecture:

Definition 91 (Unique Games Conjecture (UGC)) *For every $\epsilon, \delta > 0$, there is an n large enough such that $GAPUG_{1-\epsilon,\delta}$ on an alphabet of size n is NP-hard.*

We note a few things about the conjecture before we delve into its importance and applications. As we stated before, if $\epsilon = 0$, the problem reduces to the decision version, which we already stated was easy to solve. Furthermore, the spirit of the conjecture mostly concerns that question of whether $GAPUG$ is in P. It does not necessarily have to be NP-hard.

We will be giving a summary of some of the results that have come about from studying this problem. We will see that unlike the $P \neq NP$ conjecture, there is strong evidence for each side of UGC. In fact most papers on the topic will present arguments both for and against it.

8.3.1 Raghavendra's Result: Optimality of SDP

The first result we present is contingent on UGC and loosely speaking, it shows the optimality of an SDP approximation for *any* UG problem. This was shown by Rahavendra in [49]. Formally:

Theorem 92 (SDP Optimality [49]) *For any CSP C with optimal value* `OPT`*, there exists a "natural" SDP relaxation with optimal value* `SDP` *such that: assuming UGC, it is NP-hard to approximate C with any factor strictly less than* `SDP/OPT`*.*

We will not go into the details of the proof here and we refer the reader to his original paper [49] for details. Interestingly, this result hold not just for CSPs we have seen so far but for a larger class called *generalized* CSPs. Simply put, instead of having a set predicates that are either satisfied or not satisfied (0,1), each has a payoff function value attached to it that can take any values in $[-1, 1]$.

Raghavendra's proof converts the integrality gap for generalized CSPs to a reduction to an instance of GAP UG via a longcode-style PCP. Something similar can be found in Rahul's notes about inner PCP of Dinur, which is a Hadamard-code PCP that is also 2-prover 1-round. However it yields a particularly strong version of the reduction called an Assignment Tester. Here, we can exploit the fact that for the problems we are reducing to, i.e., the GSPs we know specifically have some integrality gap for this natural SDP relaxation. This allows for weakening of the reduction and yields the result we are looking for.

It is interesting to note that Raghavendra's result can be applied to problems even where the integrality gap is not necessarily known. In other words, if we have a problem where the natural SDP integrality gap is not known (no proof exists), we can still apply Raghavendra's result to show that for this ephemeral gap, it is NP-hard to approximate the problem better. We note that we still need a proof that such a gap exists, we just don't need to know its value. We can contrast this to such problems as Max-Cut. The Goemans and Williamson algorithm can be coaxed into this natural SDP form and hence Raghavendra's result applies directly to the Max-Cut problem. There was a proof of this for Max-Cut specifically [37], but it involved knowing the gap for Max-Cut (i.e. [31], [21]).

8.4 An Unconditional Result: Metric Space Embeddings

So far, the main tool we have seen for solving problems is a relaxation based on semi-definite programming. The gap of such relaxations comes from the fact that we must take the vectors we get form solving the SDP relaxation and turn them into some kind of discrete answer to our original optimization problem. For instance the Goemans and Williamson algorithm used a random hyperplane to assign discrete labels to vectors on the sphere. Looking at it slightly differently, we can view this as the problem of taking vectors that live in some SDP vector space (i.e. unit vectors on the sphere) and embed them into some different space (i.e. ± 1 labels). The integrality gap for certain problems can then be thought of as some sort of measure of distortion in this embedding process.

8.4.1 Metric Spaces

Here we present a formal introduction to the metric space tools as well as the conjecture that will dominate this section.

Definition 93 (Metric Space Embedding) *A metric (X_1, d_1) embeds with distortion at most Γ into (X_2, d_2) if there is a map $f : X_1 \to X_2$ such that for all $x, y \in X_1$, $d_1(x, y) \leq d_2(f(x), f(y)) \leq \Gamma d_1(x, y)$. If $\Gamma = 1$, d_1 is said to embed isometrically into d_2.*

Using this notion of metric space embeddings, we can now frame the hardness of approximating certain problems in terms of the level of distortion that arises from embedding their relaxations back into the original integer space. Khot and Vishnoi [38] take this idea and using techniques inspired from the recent work into UGC, solve the following conjecture from Goemans and Linial:

Definition 94 (Negative type metric) *A metric space (X, d) is of negative type if $(X, \sqrt{d})$ embeds isometrically into l_2. The class of negative type metrics is denoted l_2^2.*

Conjecture 95 ($(l_2^2, l_1, O(1)$ Conjecture) *Every negative type metric embeds into l_1 with constant distortion.*

Working through their dis-proof of this conjecture makes up the rest of this section. We will see that even though it uses many of the ideas we have discussed in relation to UGC, it is an unconditional result. For now we give some background on the origin of the conjecture:

Definition 96 (Cut Metric) *A cut metric δ_S on a set X where $S \subseteq X$ is defined by:*

$$\delta_S(x, y) = \begin{cases} 1, & \text{if } |\{x, y\} \cap S| = 1 \\ 2, & \text{otherwise} \end{cases}$$

The class of all such metrics forms a cone and is denoted Cut.

Fact 97 *Cut metric characterization (X, d) is l_1 embeddable iff $d \in$ Cut*

We can instantly see that Cut metrics are useful to us as many of the problems we have discussed so far can be written as optimizations over *cuts* of graphs. Hence, solutions to our optimization problems are going to be l_1 embeddable.

Fact 98 *Every l_1 metric is negative type.*

Fact 99 (l_2^2 Characterization) *A metric d on $\{1, ..., n\}$ is negative type iff the matrix $\mathbf{Q}$ with $\mathbf{Q}_{ij} = (d(i,n) + d(j,n) - d(i,j))$ is PSD.*

The previous characterization gives us a good reason to look at l_2^2 metrics. Looking back at our SDP relaxation for Max-Cut (8.2), we can modify it in the following way:

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in E} \frac{1 - ||v^i - v^j||^2}{2} \\ \text{subject to} \quad \forall i \quad & ||v^i||^2 = 1 \\ \forall i,j,k \quad & ||v^i - v^j||^2 + ||v^j - v^k||^2 \geq ||v^i - v^k||^2 \end{aligned} \tag{8.5}$$

The extra inequality constraints force the resulting vectors to be a negative type metric space. Another result of [38] shows that even with the extra constraints, graphs can be constructed with gaps arbitrarily close to α_{GW}. It is also shown in [21] that with such constraints, SDP = OPT and so all of the gap is concentrated in the rounding procedure (the random hyperplane).

Hence, we can now look at the Goemans and Williamson integrality gap as the distortion of embedding these vectors into the l_1 space of the original problem. In this case, because α_{GW} was universal, we see that the distortion is constant. It is results like this that motivated the $(l_2^2, l_1, O(1))$ conjecture.

8.4.2 SparsestCut and BalancedSeparator

To give a proof of the $(l_2^2, l_1, O(1))$ conjecture, we must first introduce a couple more cut problems.

Definition 100 (SparsestCut) *For a graph $G = (V, e)$ with weight $wt(e)$ and demand $dem(e)$ associated with each edge e, the sparsest cut is given by the nontrivial S that minimizes:*

$$\underset{S\subsetneq V}{minimize} \quad \sum_{(i,j)\in E(S,\bar{S})} \frac{wt((i,j))}{dem((i,j))} \tag{8.6}$$

We note that taking advantage of all Cut metrics l_1 embeddable, we can rewrite this problem in terms of metrics in the following way:

$$\underset{d \text{ is } l_1 \text{ embeddable}}{\text{minimize}} \quad \sum_{(i,j)\in E} d(i,j) \frac{wt((i,j))}{dem((i,j))} \tag{8.7}$$

And since all l_1 metrics are l_2^2 metrics, the following objective can be seen as a relaxation of (8.7):

$$\underset{d \text{ is } l_2^2}{\text{minimize}} \quad \sum_{(i,j)\in E} d(i,j)\frac{wt((i,j))}{dem((i,j))} \tag{8.8}$$

Note that we can efficiently get a solution to (8.8) using SDP. Then, we have the following:

Observation 101 *If every n-point l_2^2 metric embeds into l_1 with distortion $f(n)$, then* SparsestCut *can be approximated within factor $f(n)$. In particular if $(l_2^2, l_1, O(1))$ conjecture is true,* SparsestCut *has a constant factor approximation.*

The previous observation gives us a method to disprove the $(l_2^2, l_2, O(1))$ conjecture. We must exhibit an instance (or family of instances) of SparsestCut with a super-constant integrality gap. Instead of working with SparsestCut, the authors of [38] use a slightly modified version:

Definition 102 (BalancedSeparator) *For a graph $G = (V, E)$ with weight wt and demand dem associated with each edge, let D be the total demand: $\sum_e dem(e)$. Given a balance parameter B where $D/4 \leq B \leq D/2$, we want to:*

$$\begin{aligned} \underset{S\subsetneq V}{\textit{minimize}} \quad & \sum_{(i,j)\in E(S,\bar{S})} wt((i,j)) \\ \textit{subject to} \quad & \sum_{(i,j)\in E(S,\bar{S})} dem((i,j)) \geq B \end{aligned} \tag{8.9}$$

We can play the same game we did with the SparsestCut problem to see that if we can construct a super constant integrality gap for BalancedSeparator, then we can disprove the $(l_2^2, l_1, O(1))$ conjecture. Note that this does not even necessarily have to involve UG at all. If we could just prove a super constant IG for BalancedSeparator using any method, we still get the desired result. This, however, has proven to be difficult (though it has since been done, see [28] for example).

The approach used in [38] as we shall see is inspired by the techniques developed around UGC and relies on the close connection of BalancedSeparator and UG.

8.4.3 Refutation of $(l_2^2, l_1, O(1))$

The refutation described in [38] involves the following steps:

1. A PCP reduction from UG to BalancedSeparator
2. Constructing a super constant integrality gap instance for UG
3. Show this instance of UG reduces to a super constant gap for BalancedSeparator

PCP reduction

We now describe the PCP style reduction from an instance I of UG given by graph $G = (V, e)$, constraints π_e and alphabet $[N]$ to a instance of BalancedSeparator $G' = (V', E')$. The construction works by replacing each vertex $v \in V$ with a block of 2^N vertices corresponding to the boolean hypercube $\{-1, 1\}^N$. Denote this block $V'[v] = \{(v, x) : x \in \{-1, 1\}^N\}$. Then the new V' has a total of $|V| \cdot 2^N$ vertices.

For every edge $e = (i, j) \in E$, we need to in G' capture the constraint that the labels of i and j are consistent. Specifically, two points (i, x) and (j, y) will have an edge between them if x' and y have hamming distance at most ϵN. Here x' is x with indices permuted according to the permutation π_e. We note that this formulation is crucial to both the reduction as well as (we shall see) the actual integrality gap instance of UG that we will construct.

Theorem 103 (Completeness) *If* $\texttt{OPT}(I) \geq 1 - \eta$*, then* G' *has a* $(1/2, 1/2)$ *partition that cuts at most* $\eta + \epsilon$ *fraction of its edges.*

Theorem 104 (Soundness) *If* $\texttt{OPT}(I) \leq 2^{-O(1/\epsilon^2)}$*, then every* $(1/4, 3/4)$ *partition of* G' *cuts at least* $\sqrt{\epsilon}$ *fraction of its edges.*

We refer the reader to the original paper [38] for the proofs. The important thing to take away is that this reduction from UG to BalancedSeparator is gap-preserving, much like Raghavendra's result. By gap-preserving, we mean the reduction preserves the gap of the SDP relaxation given by Definitions 107 and 112. Under the assumption of UGC and an application of Raghavendra's result, we can show that this reduction is of the natural form and hence it is NP-hard to get a better integrality gap for BalancedSeparator.

However, we do not need this result (gladly, as we have seen it has proven difficult); it can be shown that the vectors that maximize the BalancedSeparator relaxation form an l_2^2 metric and hence a super constant integrality gap for this *specific* relaxation is enough to refute the $(l_2^2, l_1, O(1))$ conjecture. In other words, if we had an instance I of UG, we could run this reduction to get an instance I' of BalancedSeparator. If we could then approximate BalancedSeparator to a constant factor, we immediately get back a constant factor approximation for I.

Fact 105 (UGC refutes $(l_2^2, l_2, O(1))$ **[16])** *If UGC is true, then* SparsestCut *and* BalancedSeparator *are NP-hard to approximate within any constant factor.*

This means that one way to refute $(l_2^2, l_1, O(1))$ is to prove UGC. Of course, this has proven to be a daunting task. The power of this reduction is that now we only have to find *one* instance of UG with a super constant integrality gap. Then, when we apply this reduction, we will get an instance of BalancedSeparator that preserves this super constant gap. This can be seen as strong evidence for UGC.

Gap For UG

The construction of the super constant integrality gap instance for UG is quite involved and involves small-set expanders and a so-called noisy hypercube. We will define these later.

For now, we will discuss the main results of the formulation and then give a brief overview of how they are achieved.

Theorem 106 ([38]) *Let N be an integer and $\eta > 0$ be a parameter. Then, there is a graph $G = (V, E)$ on $2^N/N$ vertices with the following properties: each vertex u is assigned a set of orthonormal basis vectors $B[u] = \{u_1, ..., u_N\}$ of $\mathbb{R}^N$.*

1. *For every edge $e = (u, v) \in E$, $B[u]$ and $B[v]$ are almost the same up to a small perturbation. This perturbation is in the form of a permutation $\pi_e : [N] \to [N]$ such that $\langle u_{\pi_e(i)}, v_i \rangle \geq 1 - \eta$.*
2. *For any labeling $\lambda : V \to [N]$, for at least $1 - N^{-\eta}$ fraction of edges $e = (u, v) \in E$ have $\pi_e(\lambda(u)) \neq \lambda(v)$.*

Definition 107 (SDP relaxation of UG)

$$\begin{aligned} &\textit{maximize} \quad \sum_{(u,v)\in E} wt((u,v)) \frac{1}{N} \left(\sum_{i=1}^{N} \langle u_{\pi_e(i)}, v_i \rangle \right) \\ &\textit{subject to} \quad \textit{the constraints of Theorem 106} \end{aligned} \tag{8.10}$$

1 and 2 together capture an interesting concept and bear looking into. 1 says that every pair of subspaces spanned by $B[u]$ and $B[v]$ are very close together. In 2, the act of assigning a labeling λ to the graph can be seen as picking a representative vector from each basis $B[u]$. Then, given any such labeling, almost all of the edges are not assigned their "smooth" transition guaranteed by 1 In other words, locally for any edge, the subspaces are almost continuous, but it is not possible to continuously "walk" along all bases using only the permutations.

It is clear what our instance of UG will be. Take the graph given by Theorem 106 and take the permutations of part 1. to be our constraints. Then, we immediately get that $\texttt{OPT} \leq N^{-\eta}$ from 2. But our SDP relaxation is just a weighed sum over all the edges of the inner products of our basis vectors which from 1. will have value at least $1 - \eta$. Thus, the overall integrality gap for this instance will be at least $(1 - \eta)/N^{-\eta}$ which is $o(1)$.

What remains to be shown is how to construct such a G and thus how to construct such bases $B[u]$. To do this, we will first introduce the concept of label extended graphs:

Definition 108 (Label extended graphs) *Given a UG instance $G = (V, E)$ with constraints π and alphabet $[N]$, the label extended graph $G' = (V', E')$ is defined as $V' = V \times [N]$ and for every edge $e = (u, v) \in E$, $e' = ((u, \pi_e(i)), (w, i))$ will be in E'.*

Basically, to get a label extended graph, we simply replace each vertex with a cloud of vertices corresponding to all possible labels. Edges are added in accordance with the constraints given. We gain the following from these label extended graphs:

Observation 109 *For any labeling λ to the vertices of G, define a set on the label extended graph G' as follows: $S_\lambda = \{(v, \lambda(v)) | v \in V\}$. Let $val(\lambda)$ be the fraction of constraints satisfied by λ and $\Phi(S_\lambda)$ be the expansion of S_λ in G'. Then $val(\lambda) = 1 - \Phi(S_\lambda)$.*

To see the above, simply note that expansion is the probability that when starting in S_λ, a random edge will take you out of S_λ. If the edge we pick is satisfied by λ, then it follows that we will not leave S_λ. Hence, the probability of leaving S_λ and the edge not satisfying a constraint are the same.

This observation now allows us to think about the optimal UG value in terms of expanding sets. The rest of the construction then relies on finding an instance of UG that has a label extended graph with high expansion. The graph given by [38] is based on the so called Noisy Hypercube:

Definition 110 (Noisy Hypercube) *The noisy hypercube H with parameters $N, \eta > 0$ has vertex set corresponding to $\{-1, 1\}^N$ and edges gotten by starting at a vertex v and randomly flipping each bit of v with probability η. The resulting vertex w induces the edge (v, w).*

We have seen such a construction a few times this course. Indeed, this is very similar to the reduction used earlier from UG to BalancedSeparator. The rest of the construction revolves around proving certain properties of such a Noisy Hypercube H. To simplify, we take H and group its vertices into groups of size N such that the resulting graph is a label extension of some UG. In other words, the edges between the groups are permutations. From here, we use the fact for Noisy Hypercubes, the following result holds:

Theorem 111 *Let H be a noisy hypercube with parameters N, η and $S \subseteq V$ be a set of relative size $1/N$. Then $1 - \Phi(S) \leq N^{-(\eta+\eta^2)}$.*

This guarantees the second part of Theorem 106. It can also be shown that the first part holds for H constructed in this manner. So, altogether, this gives us our bases we were looking for and hence we have our super constant integrality gap instance for UG.

Gap for BalancedSeparator

Now all that is left is to combine the high integrality gap instance of UG along with the reduction to BalancedSeparator to get a high integrality gap for BalancedSeparator.

For completeness, here is the SDP relaxation for BalancedSeparator:

Definition 112 (SDP relaxation of BalancedSeparator)

$$\begin{aligned} \textit{minimize} \quad & \frac{1}{|E|} \sum_{(u,v)\in E} \frac{1}{4} ||v_u - v_v||^2 \\ \textit{subject to} \quad \forall i \in V \quad & ||v_i||^2 = 1 \\ \forall i, j, k \in V \quad & ||v_i - v_j||^2 + ||v_j - v_k||^2 \geq ||v_i - v_k||^2 \\ & \sum_{i<j} ||v_i - v_j||^2 \geq |V|^2 \end{aligned} \tag{8.11}$$

Given $\epsilon > 0$ (which is a bound for our eventual BalancedSeparator gap), we can find a $\eta > 0$ such that: we let I be an instance of UG constructed from the basis vectors guaranteed

by Theorem 106. Let I' be the BalancedSeparator instance gotten from running the PCP reduction of Section 8.4.3 on I. As shown in Section 8.4.3, I has $\texttt{OPT}(I) \leq N^{-\eta}$. Choose η such that $N^{-\eta} \leq 2^{-O(1/\epsilon^2)}$, then we can apply Theorem 104 to see that every balanced partition of G' (from I') cuts at least $\sqrt{\epsilon}$ fraction of its edges. In other words $\texttt{OPT}(I') \geq \sqrt{\epsilon}$.

We can also construct a solution to the SDP relaxation with objective value at most $O(\eta + \epsilon)$. To do so, we recall the construction of the high gap UG instance I in Theorem 106. Each vertex u had a set of basis vectors $B[u]$ attached to it that more or less let us get the high gap. We will use this when constructing our solution for BalancedSeparator. Also recall from the PCP reduction, each vertex u' of the new BalancedSeparator instance is (u, x), where $x \in \{-1, 1\}^N$.

To each vertex (u, x) of I', we attach a unit vector $V_{u,s,x}^{\otimes t}$ for $s = 4, t = 2^{240} + 1$ where

$$V_{u,s,x} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} x_i u_i^{\otimes 2s}$$

Here the notation $v^{\otimes n}$, means v tensor'd with itself n times. It can be shown that all such $V_{u,s,x}^{\otimes t}$ form a l_2^2 metric. Furthermore, we can go on to show that the value of the SDP relaxation is $O(\eta + \epsilon)$. We do this again by invoking the properties of Theorem 106. For any edge $((u, x), (v, y))$ of I', we have $\langle u_{\pi_e(i)}, v_i \rangle \geq 1 - \eta$ and that the hamming distance between x and y is at most ϵN. Putting these together, we get that:

$$\langle V_{u,s,x}^{\otimes t}, V_{v,s,y}^{\otimes t} \rangle \geq 1 - O(st(\eta + \epsilon)) = 1 - O(\eta + \epsilon)$$

This suffices to show that the objective value is $O(\eta + \epsilon)$ which means the integrality gap of the relaxation $\texttt{OPT}/\texttt{SDP} \geq \frac{\sqrt{\epsilon}}{\eta + \epsilon} \approx \epsilon^{-1/2}$. So, we choose $\epsilon \approx (\log N)^{-1/3}$, which means the integrality gap is $\approx (\log N)^{1/6}$, or super constant.

8.5 Refuting UGC

So far, we have mostly seen evidence that seems to be in favor or dependent on UGC being true. There has also been a substantial amount of work put into disproving the conjecture. We address some of those efforts in this section. Before we proceed, it is important to keep in mind that Khot's original formulation of the UGC stipulated UG be NP-hard to approximate. However, even if this were proven to be false, the question of exactly how hard UG is will likely remain open. In other words, the spirit of the conjecture is just that we cannot reasonably efficiently approximate UG. It doesn't have to be NP-hard.

Indeed, in [6], a subexponential time algorithm approximating UG is given, which we will not go into here. Though not very efficient, such an algorithm is extremely strong evidence that approximating UG is *not* NP-hard. This is due to the allure of the so-called Exponential Time Hypothesis.

Definition 113 (Exponential Time Hypothesis (ETH)) *3-SAT cannot be solved in* $2^{o(n)}$.

The ETH basically just says that NP-hard problems are at least exponential. Hence, the result of [6] places approximations of UG in the same grey area of graph isomorphism.

8.5.1 Sums of Squares

For the rest of the section, we will be presenting a lot of the work from [11], which gives a good overview of what is known about refuting the conjecture from around 2014. The main focus of this paper is the so called Sums of Squares (SoS) hierarchy. The motivation behind this is the attractiveness of results like Raghavendra's. Mainly that such a large class of CSPs share the *same* optimal algorithm. This algorithms is the natural SDP relaxation. Hence, it must be that there is something underlying all of them that leads to this result and pin-pointing whatever this is will give much stronger evidence for or against UG.

The work of [11] seeks to find a an algorithm that performs better than the natural SDP formulation. Note that doing so would refute UGC. Even so, with the approach taken, it is likely that such an algorithm will naturally give a better approximation to many still open problems and much light will be shed on what exactly ties all these problems together.

The SoS hierarchy was developed independently by several researchers, while the one we will focus on is based on Lasserre [40] and Parillo [48]. The algorithm has its origins in a very old problem: Hilbert's 17th problem. The name Sums of Squares comes from the natural notion of proving a function non-negative by expressing it as a sum of square polynomials. This resulted in the following famous result (or at least a corollary):

Theorem 114 (Positivstellensatz (Stengle '74)) *Let $P_1, ..., P_m \in \mathbb{R}[x_1, ..., x_n]$ be multivariate polynomials. Then, the system of equations given by $\mathcal{E} = \{P_1 = 0, ..., P_m = 0\}$ has no real solution iff there are polynomials $Q_1, ..., Q_m \in \mathbb{R}[x]$ and a sum of squares S, such that*

$$-1 = S + \sum Q_i P_i \tag{8.12}$$

Thus, proving *non-existence* of a common zero to a group of polynomials can be thought of as the problem of finding such $Q_1, ..., Q_n$ and S as described. Such a collection of Q's and S is a called a *SoS proof* refuting the system of equations. If all parts of the proof Q_i, S have degree at most ℓ, then it is known as a *degree ℓ SoS proof*. It is the degree that defines the SoS hierarchy, which all hinges on the following result:

Theorem 115 (Finding degree ℓ SoS proofs [48, 40]) *If a degree ℓ proof refuting $\mathcal{E} = \{P_1 = 0, ..., P_m = 0\}$ exists, it can be found in $O(mn^{O(\ell)})$ time.*

Proof: [Proof Idea] As stated above, we need to find degree ℓ $Q_1, ..., Q_m$ and S on n variables such that $-1 = S + \sum P_i Q_i$. First, note that each polynomial Q_i, S can be written using $n^{O(\ell)}$ coefficients. Then, the problem of finding a proof reduces to solving the system of linear equations in $nm^{O(\ell)}$ variables given by $-1 = S + \sum P_i Q_i$. The only constraint we need to enforce is that S is SoS.

Enforcing this constraint is actually not difficult to do. If S is SoS, then $S = \sum p_k^2$ for some polynomials $\{p_k\}$. Assume for a moment that $S = p^2$ for a single polynomial p. Assume p is written as a sum of distinct monomials $\mathbf{p} = c_1 f_1 + c_2 f_2 + ... + c_k f_k$, where f_i is just the ith monomial of p. Then

$$S = p^2 = (\sum c_i f_i)^2 = (c^t f)^2 = f^t (cc^t) f = f^t C f \tag{8.13}$$

Here C is PSD. Hence, enforcing the constraint that S is SoS comes down to solving a linear system of equations with PSD constraints on a portion of the variables, which is exactly what SDP efficiently solves. ■

With this in hand, we can derive an algorithm for the following optimization problem:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & P_0(x) \\ \text{subject to} \quad & P_1(x) = P_2(x) = ... = P_m(x) = 0 \end{aligned} \tag{8.14}$$

The algorithm is parameterized by the maximum degree ℓ of the proofs we want to search for. It works as follows: Given ℓ, output the smallest $\gamma^{(\ell)}$ such that there is *not* a degree ℓ refuting SoS proof for $\mathcal{E} = \{P_0 = \gamma^{(\ell)}, P_1 = 0, ..., P_m = 0\}$. So, start with some very small value δ that you know underestimates $P_0(x)$. Find a degree ℓ proof refuting $P_0(x) = \delta$. Increase δ and repeat until you no longer have a proof of refutation. The last value δ is is denoted $\gamma^{(\ell)}$, the degree ℓ estimate for min P_0. Because higher degree proofs are a superset of lower degree proofs, we get the following chain of inequalities:

$$\gamma^{(2)} \leq \gamma^{(4)} \leq ... \leq \min P_0(x) = \texttt{OPT}$$

In other words, as we increase the degree of the proofs, we can only get closer to the optimum value. It is from this where we derive the notion of the *hierarchy* of SoS.

8.5.2 Relation to UGC

We introduce the SoS hierarchy as it is a promising candidate for an algorithm that can perform better than the natural SDP relaxation shown by Raghavendra to be optimal assuming the UGC. Interestingly, this natural relaxation is in fact part of the SoS hierarchy. It can formulated with the right polynomials as a degree 2 estimate of those polynomials.

To motivate this fact, note that the natural SDP relaxation only uses inner products of the vectors. Hence, the entire optimization problem can be written in terms of degree 2 monomials (gotten from taking these dot products), so a refutation need only encompass these degree 2 monomials, meaning the refutation will ultimately have degree 2.

UGC can now be thought of as a question of the power of the SoS hierarchy. If we restrict ourselves to polynomials representing UGs, then UGC basically asks if any (polynomial) level of the SoS hierarchy performs strictly better than the first level (2). If so, we would effectively have an efficient algorithm that performs better than the natural SDP formulation and thus UGC would be refuted. That's not to say that UG becomes *easy* in any sense. Suppose such a result can be shown. There will naturally follow another conjecture that says that polynomially many levels of the SoS hierarchy is optimal for UG.

To illustrate the difficulty of the area, it is currently unknown whether degree 4 proofs provide any more power in the general UG setting. What is known is for certain instances thought "hard" for degree 2, we can do better at higher levels. For instance, hearkening back to our discussion on Max-Cut, we recall that [21] showed tight instances for the GW algorithm, i.e. specific graph constructions for which the integrality gap of the GW algorithm met the GW constant. Using degree 4 SoS algorithms, this gap is improved upon

for that specific graph [14]. This falls short of refuting UGC because it does not hold for all graphs. This is the truth for many areas. There is not currently any known problem instance that remains hard to solve using a constant number of levels of the SoS hierarchy. A proof that it performs better on all instances remains elusive, however.

Chapter 9

Inner PCP

Lecture(s): 28–30
Date(s): 3/15, 3/17
Lecturer: Meera Sitharam
Scribe: Rahul Prabhu

9.1 Introduction

In Dinur's proof of the PCP theorem [18], the amplification step increases the gap but it leads to a blow up in the size of the alphabet. In order to be able to repeat amplification, we need to reduce the alphabet down to the original size. This is done using two key ideas viz. modularization and composition.

We take the constraint graph G and Booleanize each edge e to get a corresponding boolean formula Φ. Note that booleanizing the constraints reduces the alphabet size from d^t to 2. Now, each of these clauses is fed to a q-query assignment tester to get a set of constraints Ψ. Constraints in Ψ have q variables and an alphabet size of 2. Now, we convert the constraints of the q-query assignment tester to constraints of a 2-query assignment tester κ. This allows us to easily plug in the constraints back to the original graph by treating the two variables as the label assignments of 2 vertices at the ends of an edge.

In this notes, the entire process of alphabet reduction is divided into two parts. The first part is the modularization also called the "Inner PCP". This step involves the conversion of the boolean formula Φ to the set of constraints Ψ using the q-query assignment tester and the subsequent conversion of q-query AT Ψ's to a bunch of 2-query AT κ's. This modularization does not affect the soundness of the PCP because the inner PCP is a stronger version of the PCP. The procedure and the analysis of the stronger PCP are discussed in section 9.3. The next step is the composition where 2-variable clauses κ (containing variables taking values in Σ_0) output by the 2-query AT are converted into edges and get "patched" in place of the original edge e whose corresponding constraint Boolean formula Φ was input to the q-query AT. This is discussed in section 9.4. But before we begin with the modularization, we need to do some processing on the input constraint Φ to make sure that the unsatisfiability value of Φ does not depend on t the amplification factor in the previous step. Section 9.2 discusses how we get rid of this dependence and why it is important.

9.2 Pre-processing Φ before Modularization

The q-query AT starts with assignments 'a' that are δ-far from being a satisfying assignment. A Booleanization 'a' of an assignment a_X that doesn't satisfy the constraint on an edge e is at least $\frac{1}{log(d^t)}$-far from a satisfying assignment for the Φ corresponding to that edge e. This δ depends on d and t, which are constant, nevertheless as pointed out later, the dependence on t is problematic, so we will provide a method to remove this dependence. The 2-query AT inner PCP proof (discussed later) shows that for all assignments b to the auxiliary variables $Y \in \{a \cup b\}$ satisfies at most a constant fraction of clauses of Ψ. This constant fraction carries over after we convert into a 2-query AT going form a Boolean (size 2 alphabet) to a size Σ_0 alphabet.

However, what we need to do for proving the composition theorem is to show that any non-satisfying assignment to the constraints of the CSP graph G' after composition must fail to satisfy at least $gap' = \beta \cdot gap(G)$ fraction of the constraints, where $gap(G)$ was the original gap in the graph G before alphabet-reduction/composition (after amplification), and β is some constant.

Here, we require β to be a constant independent of the size of the graph and depend on the 2-query AT. We also require it to be independent of t. This is because in the amplification step, we amplified the gap by t. Having β depend on t may nullify it. However, β does depend on t. More precisely it is (γ) * $(\delta = \frac{1}{log(d^t)})$. So, we have to find a way to remove the dependence on t and make $\beta = \gamma * \rho'$, where ρ' does not depend on t. This is achieved as follows

1. Define an encoding of the label assignments in Σ' for the vertices of G. Instead of just lexicographically encoding label assignments by Boolean strings of length $log(|\Sigma'| = d^t)$, we encode each label by a codeword in an error correcting code - with minimum distance ρ between codewords. Let the encoding bijection be c. Each label gets mapped to a unique codeword, and each codeword has a unique label associated with it. For all x, y distinct label assignments, $c(x)$ is ρ-far from $c(y)$. (ρ is a fraction of the total number of bits in the two strings).

   First we change the Boolean formula constraint Φ on edge e of the graph G (before composing with 2-query AT output Ψ) as follows: $\Phi(a, b) = 1$ if and only if there is an a' and b' (original label assignments) such that $c(a') = a, c(b') = b$ and the edge constraint holds for (a, b). This has the effect that the satisfying assignments of Φ must be built from two codewords and are at least ρ-far from each other. Now Φ becomes input to the 2-query AT.

2. Next note that this massaging of Φ does not affect completeness. If there is a satisfying assignment for G, i.e, $gap(G) = 0$, then clearly there is a satisfying assignment for the intermediate graph that has Φ instead of the original constraints on the edges which implies a satisfying assignment for the composed graph G' (after composition, i.e, after patching in the outputs Ψ of the 2-query AT's on inputs Φ for each edge e of G).

   However, note that for the non-satisfying label assignments of G, each violated edge e has an endpoint whose label maps via c to a string that is ρ-far from any string corresponding to satisfying assignment of Φ.

This is what will help us with the control of the gap factor β on the soundness side and make it independent of t below.

3. To prove the required soundness and gap, start with a G' that does not have a satisfying assignment. Due to completeness, this means G does not have a satisfying assignment either, and every assignment for G fails to satisfy $gap(G)$ fraction of constraints.

   Take an arbitrary assignment σ' for G', and extract an assignment σ for G as follows: for each edge e of G, take the Booleanized X-part corresponding to the assignment σ': each such X-part contains 2 parts, i.e, the Booleanized label assignments of the 2 endpoints u, v of the edge $e = (u, v)$ corresponding to Φ. Call these parts $\sigma'(u)$ and $\sigma'(v)$ respectively (these are arguments to both Φ and Ψ). Now σ is chosen such that for each vertex w the encoding $c(\sigma(w) = z)$ is the closest codeword to $\sigma'(w)$.

   We know σ violates $gap(G)$ of the constraints/edges of G. Let $e = (u, v)$ be any one of these violated constraint edges. We want to know how far $\sigma'(u), \sigma'(v)$ is to a satisfying assignment of Φ (recall that these variables are maintained by the 2-query assignment tester in generating the clauses of Ψ). Let $\sigma''(u), \sigma''(v)$ be the closest satisfying assignment of Φ.

   By the definition of Φ, both $\sigma''(u), \sigma''(v)$ must correspond to codewords s and t, and since σ violates the constraint on e, either the codeword $c(\sigma(u)) \neq s$ or $c(\sigma(v)) \neq t$. Let us assume the former without loss of generality. Then,

$$\begin{aligned} \rho &\leq \text{distance between } c(\sigma(u)) \textit{ and } s \\ &\leq \text{distance between } c(\sigma(u)) \text{ and } \sigma'(u) + \text{distance between } \sigma'(u) \text{ and } s \\ &\leq 2 \cdot \text{distance between } s \text{ and } \sigma'(u) \end{aligned} \tag{9.1}$$

   (the last inequality is holds because $c(\sigma(u))$ is the codeword closest to $\sigma'(u)$).

   This means at least $\rho/2$ positions of $\sigma'(u)$ must be changed to obtain a satisfying assignment $s =$ the closest satisfying assignment $\alpha''(u)$. Thus $\sigma'(u), \sigma'(v)$ is at least $\rho/4$ (as a fraction) far from any satisfying assignment to Φ.

   For every one of the $gap(G)$ fraction of edges $e = (u, v)$ violated by $\sigma \in G$, the assignment $\sigma'(u)\sigma'(v)$ is $\rho/4$-far from a satisfying assignment of Φ, and hence violates some $\gamma\rho/4$ fraction of clauses of Ψ (where γ is the constant depending on the 2-query AT), thus σ' violates at least $\beta gap(G)$ fraction of edges in G', where $\beta = \gamma\rho/4 = \gamma\rho'$ which is independent of t.

9.3 Modularization

We start by defining a q-query assignment tester.

Definition 116 (q-Query Assignment Tester) *The assignment tester is a reduction algorithm $qAT(\Phi)^X \to \Psi^{\{X \cup Y\}}$, that takes as input a Boolean function Φ of boolean variables and outputs a system of constraints Ψ over X and a set Y of auxiliary variables such that*

- *Variables in Y take values in Σ_0.*
- *Each conjunctive clause in Ψ uses at most q variables.*
- *$\forall$ a if $\Phi(a) = 1$, $\exists$ b for Y such that $\Psi(a, b) = 1$.*
- *If $\Phi(a)$ is δ far from being satisfied, then for every b, $\Psi(a, b)$ is β-far from being satisfied. Where $\beta = \gamma\delta$ and γ is some constant depending on inner workings of q-query AT.*

9.3.1 Construction of the Inner PCP(6-query AT)

Now, we describe the construction of the assignment tester. The goal is, given Φ^X, a conjunction of clauses and 'a' assigned to X, give a PCP that uses only 6 queries and has completeness and soundness as follows:

- $\Phi(a) = 1 \implies \exists\ b$ such that verifier accepts.
- $\Phi(a)$ is δ-far from $1 \implies \forall\ b$, verifier accepts with probability $\gamma\delta$.

Such a PCP yields a fomula $\Psi^{\{X \cup Y\}}$ where all clauses of the formula have at most q variables and the following hold

- $\Phi(a) = 1 \implies \exists\ b$ such that $\Psi(a, b) = 1$.
- $\Phi(a)$ is δ-far from $1 \implies \forall\ b, \Psi(a, b)$, violates at least δ clauses.

To be able to convert each clause in Φ to a clause in Ψ with a constant number of queries, we arithmetize the boolean formula to a system of quadratic equations. Assuming, z_i and z_j are the inputs to the gate and z_k is the output, we convert each gate in the following way,

- We replace each AND gate with the equation $z_i z_j - z_k = 0$.
- We replace each OR gate with the equation $z_i + z_j + z_i z_j - z_k = 0$.
- We replace each NOT gate with the equation $z_i + z_k + 1 = 0$.
- We replace each OUT gate with the equation $z_k + z_i - 1 = 0$.

Any circuit with input variables in X and gates Y has a corresponding quadratic equation mod 2.

$$\mathcal{P} = P_1, P_2, \ldots, P_m$$

over $\{X \cup Y\}$ where, $|X \cup Y| = N$. Given an edge e, an assignment to $\{X \cup Y\}$ is to check whether $\mathcal{P} = 0$. The naive way of solving this would require knowing what e was assigned to, which would require N queries. To reduce the number of queries(sizes of clauses in Ψ) we pick $r \in \{0, 1\}^n$ random equations in $\mathcal{P}$ and check if

$$\sum_{i=1}^{m} r_i Pi(a) mod\ 2 = 0$$

We show that even if some P_i non-zero with probability $\frac{1}{2}$ their linear combination is non zero. This test has completeness = 1 and soundness = $\frac{1}{2}$. It is easy to see the correctness is true since if all P_i are zero, their sum is zero as well. The soundness follows from the following lemma.

Lemma 117 *If all P_i are not zero then*

$$Prob[\sum_{i=1}^{m} P_i(a) \neq 0] = \frac{1}{2}$$

Instead of testing $P_i(a)$ for all a, we test it for some m random P_i.

$$Prob(a) = \sum_{i=1}^{m} r_i P_i(a) \mod 2$$

$$Prob(a) = S_0 + \sum_{i=1}^{N} s_i a_i + \sum_{1 \leq i \leq j \leq N} t_{ij} a_i a_j$$

Working of the 6-query AT

Input:

ϕ is a set of quadratic constraints over X and Y

X is the inputs to the circuit computing ϕ

Y contains the output of the gates and variables corresponding to entries in L and Q

$a \in \{0,1\}^{N=|X|}$ a supposedly satisfying assignment for ϕ.

$L_a(s) - a^{th}$ entry of the s^{th} code word of the Hadamard code.

Table L: $F_2^N \to F_2^N$ such that $L_a(s) = \frac{1-<-1>^{s,a}}{2} = \sum_i a_i s_i \mod 2$

Table Q: $F_2^N \to F_2$ such that $Q_a(t) = \sum_{i,j} a_i a_j t_{ij}$.

Output:

Ψ a set of clauses in $\{X \cup Y\}$

The procedure:

Step 1 Run BLR test on L
Run BLR test on Q.

Step 2 Pick a random s and s' and check self-correct(L, s') = self-correct$(Q, s \otimes s')$.

Step 3 Pick a random vector $r \in F_2^m$ and compute co-efficients s_i such that

$$Pr(a) = \sum_{i=1}^{m} r_i P_i(a) = s_0 + \sum_i s_i(a) + \sum_{i,j} t_{ij} a_i a_j$$

Check if s_0+ self-correct(L, s)+self-correct$(Q, T) = 0$.

Step 4 Pick a random $i \in \{1, \ldots, |X|\}$ with $e_i = 0$. Check self-correct$(L, e_i) = a_i$.

Suppose we are given f that is δ-close to linear function L_s, i.e., L_s and f disagree on δ fraction entries then,

- If $\delta < \frac{1}{4}$ then $\exists$ a unique codeword L that is δ-close to f.
- The self correct procedure(shown in Algorithm 1) would compute $L(x)$ for any x with high accuracy (Just knowing f and x is sufficient since x could be exactly where L and f differ).

self-correct(f,x)
select y randomly in $\{0,1\}^n$
return f(x+y) - f(y)

Algorithm 1: The self-correct Algorithm

To construct Ψ, we pick 3 variable constraints over variables indexed by L and we pick constraints 3 variable constraints over variables indexed by Q.

Lemma 118 *If f is δ close to a linear function L for $\delta < \frac{1}{4}$ then self-correct(f, x) computes $L(x)$ with probability at least $1 - 2\delta$.*

Correctness Analysis

In this section C_i is the discussion on the correctness of step i.

C1 - Correctness of the first step means that L is a linear function over $F_2^N = \psi_c$ for some vector $c \in F_2^n$.

C2 - Q is a linear function over $F_2^N = \chi$ for some matrix $\mathcal{C} \in F_2^N$.

C3 - Q and L are referring to the same vector C i.e., $C_{ij} = c_i c_j$.

C4 - C is referring to assignment a.

Observation 1: In C3 if we are given 2 vectors $s, s' \in F^N$ and $L = L_C$ and $Q = Q_C$ then notice that

$$L_C(s)L_C(s') = \sum_i (c_i s_i) \sum_i (c_i s'_i) = \sum_i \sum_j c_i c_j (s_i, s'_i) = Q_C(s \otimes s')$$

If Q and L satisfy the test C3 then with constant probability they are referring to the same assignment C.
Observation 2: To test C4, it is sufficient to test if co-efficient of s_i in $L(s)$ is a_i. We know that $L_C(e_i)$ should be a_i. self-correct(c, e_i) will output $L(e_i)$ reliably then check if this is a_i.

Soundness Analysis

Lemma 119 *If L and Q are δ-far Step 1 will reject with probability greater than δ. Refer to the BLR soundness for this.*

Lemma 120 *Given M(non-zero matrix) for random choice of vectors s and s', $sMs' = 0$ with probability at most $\frac{3}{4}$.*

Proof: Let M_{ij} be the ij^{th} column of M. We have,

$$s^T Ms' + (s^T + e_i)M_{s'} + sM(s' + Ej) + (s^T + e_i)M(s' + e_j) = e_i M e_j = M_{ij} \neq 0.$$

One of the terms must be non-zero. Since s and s' have been chosen randomly, with probability $\frac{1}{4}$, $s^T Ms' \neq 0$. ■

Lemma 121 *If L is δ_2-close to a linear function L_C and Q is δ_2-close to Q_C such that $C \neq c_i c_j$, then step 2 rejects with probability at least $\frac{1}{4} - 6\delta_2$.*

Lemma 122 *If L is δ_3-close to a linear function L_C and Q is δ_2-close to QF_C and for some i we ahve $P_i(c) \neq 0$ then step 3 will reject with probability at least $\frac{1}{2} - 4\delta_3$.*

Theorem 123 *Do steps 1 through 4 with probability $\frac{1}{4}$ each. Let a_X denote assignment a restricted to X of the boolean circuit corresponding to ϕ then,*

Correctness *$L = L_a$ and $Q = QF_a$ and $P_j(a) = 0 \ \forall \ 1 \leq j \leq m \implies$ test accepts with probability 1.*

Soundness *If $\delta \leq \frac{1}{28}$, If assignment a_x is δ-far from any satisfying assignment of ϕ^X, then test rejects with probability at least $\frac{\delta}{8}$ regardless of the contents of the tables L and Q.*

This theorem gives the soundness condition required for Ψ.

9.3.2 Conversion from the 6-query AT to a 2-query AT

In this section we describe the conversion of a 6-query size-2-alphabet AT to a 2-query Σ_0-alphabet AT. The 6-query assignment tester can be converted to a 2-query assignment tester, by simply augmenting Y with a new set of variables Y'. Y has alphabet $\{0, 1\}$, the new augmented variables will have alphabet $0, 1^q$. We call the newly introduced variables z_{ψ_i}. Given ψ_i, a 6-query clause with $|\Sigma'| = 2$, we want to convert it to a bunch of 2-query constraints.

To do this, for each ψ_i, and each variable u_j in ψ_i, we construct constraints $\kappa_{ij}(z_{\psi_i}, u_j)$ that are satisfied if and only if the assignment to $z_{\psi_i} \in \{0, 1\}^q$ namely, the entries of the assignment (a, b_1) relevant to ψ_i satisfies ψ_i and the assignment to u_j, namely, the entry of the assignment b_2 that correspond to the variable u_j coincides with the value assigned to u_j within (a, b_1). Note that every ψ_i that has at most 6 variables over the binary alphabet gives 6 clauses κ_{ij} each with two variables over an alphabet of size 2^q.

The new constraints $\kappa_{ij}(z_{\psi_i}, u_j)$ have the following correctness and soundness properties.

- **Correctness**: If $\phi(a) = 1$ then, $\exists\ b_1\ \forall_i, \psi_i(a, b_1) = 1$ and $\exists\ b_2\ \forall i, j\ \kappa_{ij}(a, b_1, b_2) = 1$.
- **Soundness**: If $\phi(a)$ is δ-far from being satisfied, $\forall\ b_1,\ \psi(a, b_1)$ violates at least β fraction of the clauses as already shown. Since ψ_i being violated implies at least one of the 6 κ_{ij} is violated it follows that $\kappa(a, b_1, b_2)$ violates at least $\frac{\beta}{6}$ fraction of the clauses.

The alphabet size for κ is $\Sigma_0 = 2^q$ which is a constant. This allows us to repeat the amplification step several times without it leading to a blow up in the input alphabet.

9.3.3 2-query AT is stronger PCP

Note that due to the decoupling in the composition step, it is possible that although there is no assignment of labels a^* that satisfies more than s fraction of constraints Φ simultaneously, in fact, each of the Φ's could be satisfiable, and therefore the collection of Ψ's could be simultaneously satisfiable, since they have been decoupled.

This is taken care of by the fact that, the "inner" PCP reduction (the 2-query AT) has much stronger completeness and soundness than the usual PCP reduction. Though it does not need to be a polynomial time reduction since it's domain is effectively constant sized inputs (namely constraints Φ on single edges).

The 2-query AT is a stronger than usual PCP reduction in the sense of completeness and soundness: For ANY assignment a^* to the Boolean variables of Φ:

- If a^* satisfies Φ, then there is an assignment b to the auxiliary variables, such that $\Psi(a, b)$ is satisfied; and
- If a^* is δ-far from any satisfying assignment of Φ, then for EVERY assignment b^* to the auxiliary variables at least δ fraction of constraints in Ψ are NOT satisfied by (a^*, b^*).

To see why the above is much stronger than the usual PCP reduction, observe that the usual PCP reduction would merely ensure this:

- If there is an assignment a that satisfies Φ, then there is an assignment b such that $\Psi(a, b)$ is satisfied; and
- If all assignments a satisfy at most s-fraction of the clauses of Φ, then all assignments b are such that (a^*, b^*) satisfies at most s'-fraction of constraints in $\Psi(a^*, b^*)$.

9.4 Composition

Let $G = ((V, E), \Sigma, C)$ be a constraint graph and let AT be an assignment tester. The constraint graph $G \circ AT = ((V', E'), \Sigma_0, C')$ is got by replacing every edge $e \in E$ in G with the set of edges that we get by running the AT on the boolean formula Φ corresponding to that edge. As you can see, the number of edges and hence the size of the graph increases by a large amount. However, this is still a constant factor increase in the size of the graph.

Chapter 10

Circuit Size Lower Bounds

Lecture(s): 31–33
Date(s): 3/22, 3/24
Lecturer: Jeremy Youngquist
Scribe: John Corring

10.1 Introduction

The first lecture reminds us of the formal definition of circuits and introduces a parameterization for classes of circuits as well as some formalism that will shorten lengthy descriptions downstream. I'll provide some intuition about this formalism as I introduce it, to avoid formal nonsense in the proofs. Hastad's "Switching Lemma" (aka The theorem) [26] is about how circuits change under *random restrictions:* setting some variables to constants. In the simplest case, it says that restrictions of CNFs (Conjunctive Normal Form—circuits made up of OR clauses modulated by ANDs) *probably* yield simple DNFs (Disjunctive Normal Form—vice versa). Emphasizing the italics: the theorem is a probabilistic existence result and so there is no "algorithm" given to construct "nice" restrictions. It will only tell us that there *exist* restrictions that allow this convenient reparameterization. The power of Hastad's (Switching) lemma is that no matter what circuit you provide, it is possible to simplify the circuit with a random restriction.

Hastad's work represents the culmination of several years of effort to show lower bounds on the depth of circuits computing the Parity function: the mod_p function on binary inputs. To dig deeper into this literature see [58, 13]

10.1.1 Circuit Review, Formalism, and Notation

The circuits we'll be discussing are logical circuits, with bounded depths and input sizes. Since circuits are written recursively using a base of logical operations (we'll use AND, OR, NOT) we can represent them by DAG's (directed acyclic graphs). The only cycles in the undirected version arise from the possibility of putting a variable in multiple clauses. This can also be done by adding an auxiliary variable and adding a condition to make the auxiliary behave like the original variable. So we can talk about circuits in terms of trees. Furthermore, it is always possible to transform a circuit into an alternating sequence of

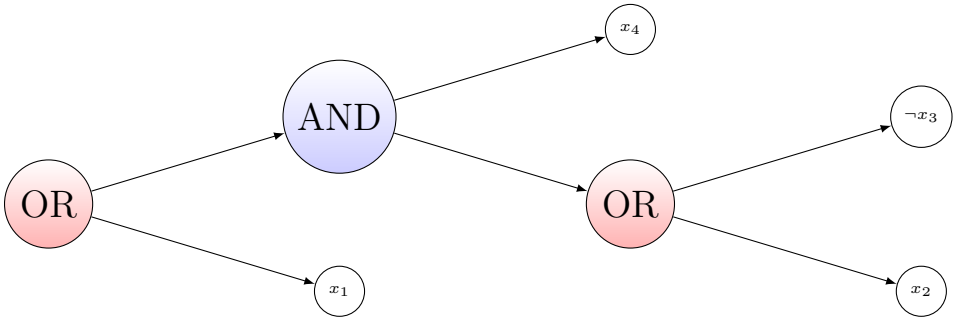

Figure 10.1: A Σ_3^2 circuit. $m = \{x_1 \to 1\}$ is a minterm of the circuit. The largest minterm is of size two: $m = \{x_4 \to 1, x_2 \to 1, x_3 \to 0\}$ has minimal size 2.

AND and OR, and move all negations to the variables (by expanding the recursion). As circuit C is called a Σ_i^S circuit if it has a total of at most S gates organized into at most i levels, with an OR gate at the output.

The **fanin** is the maximum degree of a node in the circuit. Since the fanin parameter will be important below, we will consider the more specific class of $\Sigma_i^{S,t}$ circuits that have at most $i+1$ levels, with an OR output, and fanin at most t at the input. We call a circuit C $t-$**open** if it is a $\Sigma_1^{S,t}$ circuit for some S—i.e. $t-$open is equivalent to a DNF with at most t variables per clause. The symbol for a circuit ending in an AND gate is Π (with parameters having the same meaning as above) and $t-$closed circuit is a CNF with at most t variables per clause.

Now we will get our hands a little bit dirty with circuits. As mentioned above, Hastad's lemma is a theorem about **restrictions**, or assignments of variables to the inputs of circuits. Given a circuit C a restriction ρ is a mapping from $\text{input}(C) = \{x_1, \ldots, x_n\}$ to $\{0, 1, *\}^n$. $*$ represents "free-variable". We can denote the induced circuit $C|_\rho$, $\rho^* C$, or $C \circ \rho$. Since we will be drawing random restrictions, let $\mathcal{R}_p$ be the distribution over restrictions. $\mathcal{R}_p$ models iid assignment of $\{0, 1, *\}$ over each variable in the input, with the probability of $*$ appearing in any given variable being p. The probability of $0, 1$ are then set to be even: $p(0) = p(1) = \frac{1-p}{2}$.

A **minterm** of a circuit C is a minimal assignment m of $\{0, 1\}$ to variables in $\text{input}(C)$ (so it is a (possibly partial) restriction) which forces $m^* C$ to be a tautology. What this (minimal as opposed to minimum) means is that given an assignment σ that makes C true, the minterm associated with σ is the sub-assignment of σ that has the fewest set variables while also making C true. The largest minterm of C is denoted by $\text{mint}(C)$.

Of course, to do complexity theory we need a complexity class. The class that will emerge after out discussion of Hastad's lemma will be AC^0, the set of constant depth circuits with polynomially many gates in terms of the input size. We will see more about this later.

10.2 Hastad's Lemma

Now that the notation and definitions have been set up let's dive in.

Theorem 124 (Hastad's Switching Lemma) *Let f be a $t-$closed circuit and $\rho \sim \mathcal{R}_p$. Then*

$$P[\rho^* f \text{ is not } s\text{-open}] \leq \alpha^s, \tag{10.1}$$

where $\alpha = 5pt$.

In class, we discussed a simplified version first. In this simplified version we assumed that the function f was a "read-once circuit"—a circuit in which each of the variables is featured in at most one clause. We spent quite a bit of time on the proof of this simplified version. I'll bypass the simplified proof since we have already seen the structure of the proof (in both forms). If you want to review that proof, check [13] pages 14-15. Recall that the idea was to proceed by induction: assuming the theorem for a fixed number of clauses n and showing that if we add a clause (read once, so with entirely new variables) then the theorem holds. The *key obstacle* that we must overcome in extending that inductive proof idea to the general case is that the variables that occur in the new clause may have appeared somewhere else in the circuit. To keep the notes readable I will decompose the proof into a sequence of claims that will allows us to plug in some of the bounds and equalities needed to proof the Switching lemma. Additionally, since we will technically prove a different theorem, I'll point out the equivalence between the aforementioned Theorem and the one we will prove.

Theorem 125 (Hastad's Lemma v2) *Let C be a $t-$closed circuit and $\rho \sim \mathcal{R}_p$, and suppose that F is an arbitrary function. Then*

$$P[\rho^* C \text{ is not } s\text{-open, given } \rho^* F \equiv 1] \leq \alpha^s, \tag{10.2}$$

where $\alpha = \gamma pt$, where $\gamma = \frac{4}{\ln(1+\sqrt{5})}$.

Remark If Theorem 125 holds then for $F \equiv 1$ we obtain Hastad's Lemma. Also recall again that $\rho^* C$ being $s-$open means that the restricted circuit is a DNF with fanin $\leq s$.

Proof:

First some auxiliary claims. The proof proceeds below the line break.

Claim 126 *For arbitrary events A, B,*

$$P(A|B) \leq P(A) \Leftrightarrow P(B|A) \leq P(B). \tag{10.3}$$

Proof: $\Rightarrow$: If A or B is null then both sides are $0 \leq 0$, true. If neither is null then by Bayes rule $P(A,B) = P(A|B)P(B) = P(B|A)P(A)$. By taking ratios

$$P(A|B) \leq P(A) \Rightarrow \frac{P(B|A)}{P(B)} = \frac{P(A|B)}{P(A)} \leq 1. \tag{10.4}$$

$\Leftarrow$: The implication goes both ways by the same argument. ■

Claim 127 *Let T be the set of variables provided to a disjunction C_1, and $Y \subset T$. Then*

$$P[\rho(Y) \equiv * \mid \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1] \leq (2p)^{|Y|}, \tag{10.5}$$

where $\rho(Y) = *$ denotes the event that ρ assigns all free variables to Y.

The proof of this claim consists of several probabilistic bounds and can be found in Boppana's survey [13] on pages 18-20.

Now for the proof. Note that we can show that $\rho^* C$ is $s-$open if we can bound the number of minterms of $\rho^* C$: any circuit can be written as an OR of all of its minterms, just by the definition of minterm. So we will aim to show that this probability (that $\rho^* C$ has s or fewer minterms) is sufficiently high—actually we will show that the probability of the complement set is low.

The proof proceeds by induction on the number of clauses at the last level. The inductive base is a trivial implication: if there are no clauses then the theorem is true vacuously. The inductive hypothesis is that the theorem holds for an arbitrary $t-$closed circuit with n clauses at the last level.

First we decompose the probability with respect to the possible outcomes of restrictions on the first level of ORs:

$$\begin{aligned} P[\min(\rho^* C) \leq s \mid \rho^* F \equiv 1] = P[\rho^* C_1 \equiv 1 \wedge F \equiv 1](A) \\ + P[\rho^* C_1 \not\equiv 1 \wedge F \equiv 1](B). \end{aligned} \tag{10.6}$$

where

$$(A) = P[\min(\rho^* C) \geq s \mid \rho^* C_1 \equiv 1 \wedge \rho^* F \equiv 1] \tag{10.7}$$

$$(B) = P[\min(\rho^* C) \geq s \mid \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1]. \tag{10.8}$$

For (A), clearly if $\rho^* C_1 \equiv 1$ then it does not contribute to the minterms in $C \setminus C_1$, so induction applies by taking $F' = C_1 \wedge F$ as the arbitrary function and $C \setminus C_1$ to be the circuit. So we can bound $(A) \leq \alpha^s$.

To bound be we will analyze the collection of a minterms according to which variables in C_1 then contain. Let $Y \subset \text{var}(C_1)$ be a subset of the variables in C_1. Denote by $\text{mint}^Y(C)$ the size of the largest minterm of C which sets all of Y and does not affect $\text{var}(C_1) \setminus Y$. Let $\sigma \in \{0,1\}^{|Y|}$ be an assignment for Y. Let $\text{mint}^{\sigma \to Y}(C)$ be the size of the largest minterm of C which sets all of the variables in Y to σ and does not affect $\text{var}(C_1) \setminus Y$. So

$$\text{mint}^Y(C) = \max_{\sigma \in \{0,1\}^{|Y|}} \text{mint}^{\sigma \to Y}(C). \tag{10.9}$$

If there are no minterms for C that fully assign Y then $\text{mint}^Y(C) = 0$. Let $\rho(Y) = *$ denote the event that ρ assigns free variables to all of Y.

Using the notation and definitions from above we can split the events over outcomes on

different subsets of variables $Y \subset \mathrm{var}(C_1)$ and employ a union-bound approach (again)

$$(B) = P_\rho[\mathrm{mint}(\rho^* C) \geq s \,|\rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1] \tag{10.10}$$

$$\leq \sum_{Y \subset \mathrm{var}(C_1)} P_\rho[\mathrm{mint}(\rho^* C) \geq s \,|\rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1] \tag{10.11}$$

$$= \sum_{Y \subset \mathrm{var}(C_1)} \{P_\rho[\rho(Y) = * \,|\rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1] \tag{10.12}$$

$$P_\rho[\mathrm{mint}^Y(\rho^* C) \geq s \,|\rho(Y) = * \wedge \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1]\} \,. \tag{10.13}$$

Now let

$$(D) = P_\rho[\mathrm{mint}^Y(\rho^* C) \geq s | \rho(Y) = * \wedge \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1], \tag{10.14}$$

$$\leq \sum_{\sigma \in \{\{0,1\}^{|Y|} \setminus 0^{|Y|}} P_\rho[\mathrm{mint}^{\sigma \to Y}(\rho^* C) \geq s | \rho(Y) = * \wedge \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1]. \tag{10.15}$$

We now assume that none of the inputs to T are negated (we can move the negations onto other clauses if necessary). So $\sigma = 0^{|Y|}$ can be excluded from the above probability. Now, if we decompose the variables as $\mathrm{var}(C) = T \oplus \mathrm{var}(C) \setminus T$. Consider $\rho = \rho_1 \oplus \rho_2$, then we can bound the probability by taking the maximizing ρ_1 over only the restrictions that assign 0 or $*$ to variables in T, only $*$ to variables in Y. To invoke the inductive hypothesis on the remainder of the circuit, we need to deal with these free variables.

Dealing with the free variables in $\mathrm{var}(C)$**:** The summation deals with part of them, and we will deal with the variables in $W = T \setminus Y$ by taking a maximum over assignments to W. We must permit the probability to be as big as possible over assignments to these remaining variables:

$$P[\mathrm{mint}^{\sigma \to Y}(\rho^* C) \geq s | \rho(Y) = * \wedge \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1] \leq \max_{\tau \in \{0,1\}^{|W|}} P_{\rho_2}[X] \tag{10.16}$$

$$\text{where } X = \{\mathrm{mint}(\rho_2^* \rho_1^* \tau^* \sigma^a st(C \setminus C_1)) \geq s - |Y| \,|\rho(Y) = * \wedge \rho^* C_1 \not\equiv 1 \wedge \rho^* F \equiv 1\} \,. \tag{10.17}$$

Now, $\rho(Y) = *$ does not depend on ρ_2, and neither does $C_1 \not\equiv 1$, so

$$\max_{\tau \in \{0,1\}^{|W|}} P_{\rho_2}[X] = P_{\rho_2}[\mathrm{mint}(\rho_2^* \rho_2^* \tau^* \sigma^* (C \setminus C_1)) \geq s - |Y| \,|\rho_2^* \rho_1^* F \equiv 1]. \tag{10.18}$$

Now, the inductive hypothesis may be invoked and we get a bound of $\alpha^{s-|Y|}$. So

$$(D) \leq \sum_{\sigma \in \{0,1\}^{|Y|} \setminus 0^Y} \alpha^{s-|Y|} \tag{10.19}$$

$$= (2^{|Y|} - 1)\alpha^{s-|Y|}, \tag{10.20}$$

$$(B) \leq \sum_{Y \subset T} (2p)^{|Y|}(2^{|Y|} - 1)\alpha^{s-|Y|} \tag{10.21}$$

$$\leq \alpha^s \left(\sum_{t \geq k \geq 0} \binom{t}{k} (4/\alpha)^k - \sum_{t \geq k \geq 0} \binom{t}{k} (2/\alpha)^k \right) \text{ where } t = |T| \tag{10.22}$$

$$= \alpha^s((1 + 4/\gamma t)^t - (1 + 2/\gamma t)^t) \tag{10.23}$$

$$\leq \alpha^s. \tag{10.24}$$

I'll walk through these one at a time. The bound for (D) simply follows by counting the number of summands in the equation. To get the first bound for (B) on line (10.21), invoke the Claim from above. To go from line (10.21) to (10.22), factor out the α^s leaving an $\alpha^{|Y|}$ within each summand. $\alpha^{|Y|} = \alpha^k$ in the new index k, which is used to enumerate over all subsets and that is where the binomial coefficients come from. Distributing across the $2^{|Y|} - 1$ is where the difference comes from. Then the equality (10.23) is just the binomial series, a standard result. Finally, γ is chosen to make the inequality in line (10.24) hold. ■

By using an AND bound we obtain the following theorem by applying Hastad's lemma repeatedly

Corollary 128 *For all $p, d : 0 \leq k \leq d - 1$, if $C \in \sigma_d^{S,t}$ then for $\rho \sim \mathcal{R}_{p^k}$*

$$P[\rho^* C \notin \sum_k^{S,t}] \leq S(\gamma p t)^t \tag{10.25}$$

where γ is as above.

Again, this is proved exactly as outlined and can be easily checked to follow from Hastad's Theorem. Sometimes this Corollary is treated as Hastad's lemma, and it is proved directly (by repeating the argument of the lemma above and the proof of the corollary).

10.3 Results Following from Hastad's Lemma

Hastad's Lemma leads to some important lower bounds in circuit complexity. We'll cover how Hastad's lemma proves the fundamental lower bound of $PARITY \notin AC^0$ below.

10.3.1 Lower bound for Depth of PARITY Circuits

Theorem 129 *The function*

$$PARITY : (x_1, \ldots, x_n) \mapsto \sum_i x_i \mod 2 \tag{10.26}$$

is not in AC^0.

Proof: We will show that $PARITY$ is not in $\sigma_d^{poly(n),poly(n)}$ for any fixed d (n is the input size). First a lemma

Lemma 130 *If $f \in \sigma_d^{S,t}$ for $t \geq \log S$ then there is a restriction ρ such that at least $\frac{n}{3(10t)^{d-1}} - t$ free variables remain and $\rho^* f$ is constant.*

Proof:

(of lemma)
By the Corollary, if $p = \frac{1}{10t}$ and $\rho \sim \mathcal{R}_{p^{d-1}}$ then

$$P_\rho[\rho^* f \notin \sigma_1^{S,t} \leq (\gamma/10)^t. \tag{10.27}$$

One can check that

$$\gamma/10 + P[\rho \text{ has fewer than } np^{d-1}/3 \text{ stars }] < 1. \tag{10.28}$$

This ensures that the output will be $\Sigma_1^{S,q}$ where $q \leq t$. By setting the at most t inputs for some clause to remain false (say) we can ensure that the function is a constant. ■

Invoking the lemma, we see that if $S < 2^{n^{1/d}}$ then there is some restriction with a nonzero number of stars that sets $PARITY$ to a constant, but we can change parity with just one bit. ■

10.4 Razborov-Smolensky Theorem

The Razborov-Smolensky Theorem[4] shows the "independence" between modular arithmetic for relatively prime integers. It is proved with a slightly different circuit manipulation trick which relies on approximation of circuits by polynomial functions.

10.4.1 Definitions

First, let's define a complexity class useful for the result. Let $ACC^0(q) = AC^0(\wedge, \vee, \neg, \text{mod}_q)$. So ACC^0 has access to the standard logical operations and mod_q. A **b-approximator** is a polynomial of degree b from $\{-1, 0, 1\}$ to $\{0, 1\}$. The proof uses $b-$approximators as approximations to circuits and keeps track of the errors. A key thing to keep in mind is that ACC^0 has circuits with polynomially many gates in the input.

10.4.2 Independence of mod_2 Circuits to mod_3 Circuits

First, the proof shows that $\sqrt{n}-$approximators can estimate $ACC^0(3)$ well. Then we prove that this space of approximators disagrees with mod_2 on a sizeable set of inputs. After we develop the appropriate bounds we can count the number of gates in the circuit C for mod_2 by dimensional analysis; if we assume that $\text{mod}_2 \in ACC^0(3)$ then we are lead to the conclusion that any $ACC^0(3)$ circuit for mod_2 has exponentially many gates—a contradiction with the definition of ACC^0.

Theorem 131 $\mathrm{mod}_2 \notin ACC^0(3)$.

Proof: First we build up the $ACC^0(3)$ class by approximators. We can approximate $NOT(x_1)$ by $1 - x_1$, and $\mathrm{mod}_3(x_1, x_2, \ldots, x_n) = (\sum_i x_i)^2$. Note that NOT has $b = 1$, mod_3 has $b = 2$. $AND(x_1, x_2, \ldots, x_n)$ can be approximated by $x_1 x_2 \ldots x_n$ with no error, with $b = n$. $OR = \neg AND(\neg\cdot, \neg\cdot)$ so has the same degree. We need to reduce the degree, which we do by introducing error in the computation of OR: let $X_1, \ldots, X_p \subset \{x_1, \ldots, x_n\}$ be subsets of cardinality $L \ll n$, and define $g_i = OR(X_i)$. Then g_i can be computed by an $L-$approximator, and replacing the input set $(x_1, \ldots, x_n)$ with $(g_1, \ldots, g_L)$ we get a $L-$approximator for OR which disagrees with OR on at most 2^{n-L} inputs. The last part follows by noting that on any one of the 2^n inputs $p[g_i = 0 | OR(x_1, \ldots) = 1] \leq 1/2$, so $p[OR(g_1, \ldots, g_L) = 0 | OR(x_1, \ldots, x_n)] \leq 2^{-L}$.

Lemma 132 *Every circuit C of depth d has a $(2L)^d-$approximator which disagrees with C on at most $size(C)2^{n-L}$ inputs.*

Proof: This follows from using the $2L$-approximators for OR each of the d levels. For each gate, we introduce at most 2^{n-L} new error assignments. Note that this coarse upper bound is only used to bound $size(C)$ below. ■

If we let $L = \frac{1}{2} n^{\frac{1}{2d}}$ in the previous lemma, then we get a $\sqrt{n}$-approximator. So now we have an approximation of the $ACC^0(3)$ circuits from $\sqrt{n}-$approximators with disagreements on at most $2^{n-1/2n^{\frac{1}{2d}}}$ inputs. This concludes the first part of the proof.

The second part of the proof is a bit more requires an observation about how the arithmetic of $\mathbb{F}_2$ simplifies polynomials. The effect of working over this small range is that if we have a polynomial over some variables Y we can rewrite monomials

$$\prod_{\lambda \in \Lambda} y_\lambda = \prod_{\lambda \notin \Lambda} y_\lambda \prod_{y \in Y} y \tag{10.29}$$

$$= a \prod_{\lambda \notin \Lambda} y_\lambda \tag{10.30}$$

where a is just the product of all the variables in Y—which is $\mathrm{mod}_2(Y)$—which has a $\sqrt{n}$ approximator over some subset of inputs (we'll call them G below). This will allow us to cap the degree of monomials over mod_2 by $\sqrt{n} + n/2$. The $\sqrt{n}$ comes in via the approximator and the $n/2$ comes in via the above argument (by moving to the complement set of variables if necessary). This cap will leave us just enough room to show that G cannot be all of the input space (for sufficiently large n).

Lemma 133 *For sufficiently large n, any $\sqrt{n}-$approximator m for mod_2 differs from mod_2 on at least $\frac{1}{50} 2^n$ inputs.*

Proof: Let $G = \{x \in \{-1, 1\}^n | m(x) = \mathrm{mod}_2(x)\}$. If F^G is the set of functions from $G \to \{-1, 0, 1\}$ then $|F^G| = 3^{|G|}$; we will estimate $|G|$ by approximating each $f \in F^G$ by a polynomial of degree at most $\sqrt{n} + n/2$[1], and counting the distinct such polynomials.

[1] See the observation on the previous page.

Now observe that each of the terms in the approximators are actually monomials. There are $\alpha = \sum_{i=0}^{n/2+\sqrt{n}} \binom{n}{i}$ multilinear monomials (as in $\mathbb{F}_2 = (\{-1,1\},\times)$, $x^2 = 1$) of degree at most $n/2+\sqrt{n}$. By estimating $|F^G| = 3^\alpha \leq 3^{\frac{49}{50}2^n}$, we get that $|G| \leq \frac{49}{50}2^n$. So there are at least $\frac{1}{50}2^n$ inputs on which m and mod_2 differ. ■

Now, recall that 2^{n-L} is an upper bound on the error per gate. Since the total error is at least $\frac{1}{50}2^n$ and the error per gate is at most 2^{n-L} we get the bound

$$size(C) \geq \frac{1}{50}2^L = \frac{1}{50}2^{\frac{1}{2}n^{1/2d}}. \tag{10.31}$$

■

Chapter 11

Hardness vs. Randomness

Lecture(s): 34–36
Date(s): 3/29, 3/31
Lecturer: Zihua Wei
Scribe: Alan Kuhnle

11.1 Boolean function analysis

We consider boolean functions

$$f : \{-1, 1\}^n \to \{-1, 1\}.$$

An example is the max_2 function defined as follows:

$$max_2(+1, +1) = +1$$

$$max_2(-1, +1) = +1$$

$$max_2(+1, -1) = +1$$

$$max_2(-1, -1) = -1$$

Define an indicator polynonial as follows:

$$1_a(x) = (\frac{1 + a_1x_1}{2})(\frac{1 + a_2x_2}{2}) \dots (\frac{1 + a_nx_n}{2})$$

Then we can represent f in terms of indicator polynomials:

$$f(x) = \sum_{a \in \{-1,1\}^n} f(a)1_a(x)$$

Multilinear and fourier expansion Since the indicator polynomials are multilinear when expanded out, the interpolation always produces a multilinear polynomial. For example,

$$max_2 = \frac{1}{2} + \frac{1}{2}x_1 + \frac{1}{2}x_2 - \frac{1}{2}x_1x_2$$

Let function χ_S be defined as follows:

$$\chi_S(x) = \prod_{i \in S} x_i$$

Theorem 134 (Fourier expansion) *Every function $f : \{1,1\}^n \to R$ can be uniquely expressed as a multilinear polynomial,*

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x)$$

This expression is called the Fourier expansion of f , and the real number $\hat{f}(S)$ is called the Fourier coefficient of f on S.

Parity basis We can define an inner product as follows:

$$< g, f >= 2^{-n} \sum_{x \in \{-1,1\}^n} f(x)g(x) = \mathbf{E}[f(x)g(x)]$$

Observe the following,

- for every A,B: $\chi_A\chi_B = \chi_{A \triangle B}$, where $A \triangle B$ is the symmetric difference.
- the family $\{\chi_S\}$ for all $S \subset \{1 \dots n\}$ forms an orthonormal basis, that is $< \chi_A, \chi_B >= 0$ if $A \neq B$

By the fourier representation of f, and the above observations, we have the following theorem:

Theorem 135 (Parseval's theorem) *For any $f : \{-1,1\}^n \to R$*

$$< f, f >= \boldsymbol{E}[f(x)^2] = \sum_{S \subseteq [n]} \hat{f}(S)^2$$

Next, consider a version of the Hastad Switching Lemma:

Theorem 136 (Hastad Switching Lemma)

$$\frac{1}{2^n}\left| \sum_{|x|even} f(x) - \sum_{|x|odd} f(x) \right| = |\hat{f}(1^n)| \leq M2^{-n^{1/d}}$$

This easily implies that the parity function of n bits cannot be computed by AC^0 circuits. Also readily extended to the general result by treating LHS as a fraction of place where function and parity are the same.

A reference for this section is [41].

11.2 Bound on high coefficients

Lemma 137 (Main lemma) *Let f be a Boolean function on n variables computable by a Boolean circuit of depth d and size M, and let t be any integer. Then*

$$\sum_{S \subset 1 \ldots n, |S| > t} \hat{f}(S)^2 \leq 2M2^{-t^{1/d}/20}$$

where $\hat{f}(S)$ denotes the Fourier Transform off at S.

Therefore, an AC^0 Boolean function has almost all of its power spectrum on the low-order coefficients!

Lemma 138 (HASTAD) *Let f be given by a CNF formula where each clause has size at most t, and choose a random restriction ρ with parameter p (i.e., $Pr[\rho(x_i) = *] = p$). With probability of at least $1 - (5pt)^s$, f_ρ can be expressed as a DNF formula each clause of which has size of at most s, and the clauses all accept disjoint sets of inputs.*

Corollary 139 *If f is given by a CNF (or DNF) of bottom fanin at most t, and ρ is chosen at random with $Pr[*] = p$, then*

$$Pr[deg(f_\rho) > s] < (5pt)^s$$

Repeated application of Hastad's lemma yields the following lemma:

Lemma 140 *Let f be a Boolean function computed by a circuit of size M and depth d. Then*

$$Pr[deg(f_\rho) > s] \leq M2^{-s}$$

where ρ is a random restriction with

$$Pr[*] = \frac{1}{10^d s^{d-1}}$$

Proof: We view the restriction ρ as obtained by first having a random restriction with $Pr[*] = 1/10$, and then $d1$ consecutive restrictions each with $Pr[*] = 1/(10s)$.

From 1-DNF to s-CNF

- The original fanin is at least 2s

  In this case, the probability that the gate was not eliminated by ρ, that is, that no input to this gate got assigned a 0 (assuming without loss of generality that the bottom level is an AND level) is at most $0.55^{2s} < 2^{-s}$;

- The original fanin is at most 2s

  In this case, the probability that at least s inputs got assigned a * is at most $\binom{2s}{s} 0.1^s < 2^{-s}$. Thus, the probability of failure at this stage is at most $m_1 2^{-s}$, where $m1$ is the number of gates at the bottom level.

From s-CNF to s-DNF similar. ■

Lemma 141 *Let f be a Boolean function and S an arbitrary subset of the variables. Then, for any subset of the variables A*

$$\hat{f}(A) = 2^{-|S^c|} \sum_{R \in \{0,1\}^{|S^c|}} \chi_{A \cap S^c}(R) \hat{f}_{S^c \leftarrow R}(A \cap S)$$

Proof:

$$\hat{f}(A) = E_x[f \chi_A]$$

$$2^{-|S^c|} \sum_{R_1 \in \{0,1\}^{|S^c|}} \chi_{A \cap S^c}(R_1) \Big[2^{-|S|} \sum_{R_2 \in \{0,1\}^{|S|}} \chi_{A \cap S}(R_2) f_{S^c \leftarrow R1}(R_2) \Big]$$

■

Lemma 142 *Let f be a Boolean function and S an arbitray subset. For any $B \subset S$:*

$$\sum_{C \subset S^c} \hat{f}(B \cup C)^2 = 2^{-|S^c|} \sum_{R \in \{0,1\}^{|S^c|}} \hat{f}_{S^c \leftarrow R}(B)^2$$

Proof: By Lemma 141, we have

$$LHS = 2^{-2|S^c|} \sum_{C \subset S^c} \Big(\sum_{R \in \{0,1\}^{|S^c|}} \chi_C(R) \hat{f}_{S^c \leftarrow R}(B) \Big)^2$$

$$2^{-|S^c|} \sum_{R_1 \in \{0,1\}^{|S^c|}} \sum_{R_2 \in \{0,1\}^{|S^c|}} \hat{f}_{S^c \leftarrow R_1}(B) \hat{f}_{S^c \leftarrow R_2}(B) \Big[2^{-|S^c|} \sum_{C \subset S^c} \chi_C(R_1 \oplus R_2) \Big]$$

■

Lemma 143 *Let f be a Boolean function, S an arbitrary subset, and k an integer*

$$\sum_{A, |A \cap S| > k} \hat{f}(A)^2 \leq Pr[deg(f_{S^c \leftarrow R}) > k]$$

where R is a $0-1$ assignment to variables in S^c chosen at random.

Proof:

- If $deg(f_{S^c \leftarrow R}) \leq k$: for any A such that $|A \cap S| > k$, the corresponding Fourier coefficient is zero
- On the other hand : $f_{S^c \leftarrow R}$ is a boolean function, value of $\sum_{|B|>k} \hat{f}_{S^c \leftarrow R}(B)^2$ is bounded by one

Therefore, sufficient to show :

$$\sum_{A,|A\cap S|>k} \hat{f}(A)^2 = E_R\Big[\sum_{|B|>k} \hat{f}_{S^c\leftarrow R}(B)^2\Big]$$

■

Lemma 144 *Let f be a Boolean function, t an integer, and $0 < p < 1$*

$$\sum_{|A|>t} \hat{f}(A)^2 \leq 2E_s(\sum_{|A\cap S|>pt/2} \hat{f}(A)^2)$$

where S is a subset chosen at random such that each Latiable appears in its independently with probability p, and $pt > 8$.

Proof: **The Chernoff Bounds** Let $X = \sum_{i=1}^n X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = E(X) = \sum_{i=1}^n p_i$. Then
Upper tail: $P(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2}{2+\delta}\mu}$ for all $\delta < 0$
Lower tail: $P(X \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}$ for all $0 < \delta < 1$

Using Chernoff bounds, the probability that $|A \cap S| > pt/2$ is at least $1 - e^{-tp/8}$. In our case, $tp > 8$; therefore, we can assume that the probability of $|A \cap S| > pt/2$ is at least $1/2$. Each set A contributes $\hat{f}(A)^2$ to at least half of the sets S, and the lemma follows. ■ We

are finally ready to prove the Main Lemma:
Proof: [Proof of Main Lemma] Fix $p = 1/(10t^{(d-1)/d})$, and $s = pt/2 = t^{1/d}/20$. By Lemma 144,

$$\sum_{|A|>t} \hat{f}(S)^2 \leq 2E_s(\sum_{|A\cap S|>pt/2} \hat{f}(A)^2)$$

where S is chosen at random such that each variable appears in it with probability p. Using Lemma 143, this is bounded from above by

$$2E_s Pr\Big[deg(f_{S^c\leftarrow R}) > \frac{pt}{2} = s\Big]$$

Consider now the distribution of the restriction $S^c \leftarrow R$ induced by first choosing S at random such that each variable appears in it with probability p, and then choosing a random 01 assignment R to the bits in S^c. This is exactly the same distribution as choosing a restriction p at random with $Pr[*] = p$. Since by our choice of p and s, $p \leq 1/(10^d s^{d-1})$, Lemma 140 applies and the above quantity is bounded by

$$2M2^{-s} = 2M2^{-t^{1/d}/20}$$

■ For more details on the material in this section, see [41].

11.3 Learning Constant Depth Circuits

Learning concepts Boolean functions as concepts. The standard scenario is the following: A class of concepts is fixed and known to the learner who is trying to identify a specific member in the class that is unknown to him. To this end, the learner observes pairs of input/output of the concept. Based on these observations, the learner wishes to find some concept that is close to the unknown concept. One key consequence of the LMN theorem is the smooth size-depth tradeoff in the parity lower bound of Hastad. Precisely that for any AC^0 function f computed by a circuit of size M and depth d,

Theorem 145 *Discrepancy of a function f of n bits with parity function of any set S of t input bits (both treated as functions whose range is $\{-1,+1\}$) is*

$$1/2^n|\sum_x f(x)\chi_S(x)| \leq M2^{-(t^{1/d})}$$

We consider a learning model with two phases: learning and prediction. In the learning phase the algorithm is presented randomly chosen inputs x, along with $f(x)$. During the prediction phase the algorithm is only presented random inputs x, and must output a prediction $\widetilde{f}(x)$. For a given distribution D, a alogrithm is called as (ϵ, δ, D) prediction algorithm if

$$Pr_D[\widetilde{f} \quad \text{disagree with f on more than} \quad \epsilon \quad \text{fraction of inputs}] \leq \delta$$

Learning algorithm

- LEARNING PHASE The algorithm observes f on m randomly chosen sample points $x_1, \ldots, x_m$, where $m = 4(2n^k/\epsilon)ln(2n^k/\delta)$ and $k = (20log(2m/\epsilon))^d$. Its approximation for the Sth coefficient of f is

  $$\alpha_S = \frac{\sum_{i=1}^m f(x_i)\chi_S(x_i)}{m}$$

  For all $|S| < k$, and $\alpha_S = 0$, for all $|S| > k$.

- PREDICTION PHASE The predicted value of f on input x is

  $$\widetilde{f}(x) = sign\Big(\sum_{|S|\leq k} \alpha_S\chi_S(x)\Big)$$

Theorem 146 *The above algorithm is an (ϵ, δ, D) learning algorithm for circuits of depth d and size M, where U is the uniform distribution.*

We need the following lemmas:

Lemma 147 *With high probability all low-order coefficients are approximated well.*

$$Pr\Big[For\ some S, |S| \leq k, |\alpha_S - \hat{f}(S)| > \sqrt{\frac{\epsilon}{2n^k}}\Big] \leq \delta$$

Proof: For a subset S, consider the random variable $Y_s = f(x)\chi_S(x)$. The expected value of Y_s is, by definition, $\hat{f}(S)$. The algorithm estimates this expected value by averaging over m samples. The lemma follows through a standard application of Chernoff bounds. ■

Lemma 148 *Let f be a Boolean function, and g an arbitrary function such that* $\sum S(\hat{f}(S) - \hat{g}(S))^2 \leq \epsilon$, *then* $Pr[f(x) \neq sign(g(x))] < \epsilon$.

Proof: Since f is a Boolean function, $f(x) \neq sign(g(x))$ implies that $|f(x) - g(x)| \geq 1$. Note $\|f - g\|^2 = E_x(f(x) - g(x))^2$; thus, the probability that $|f(x) - g(x)| \geq 1$ does not exceed $\|f - g\|^2$. Finally, by Parseval's equality, $\|f - g\|^2 = \sum_S (\hat{f}(S) - \hat{g}(S))^2 \leq \epsilon$ ■

Proof: [Proof of Theorem 146] Consider the function $g = \sum_{|S| \leq k} \alpha_S \chi_S$. If $|S| > k$, then $\hat{g}(S) = O$. But f has a circuit of depth d and size M, and an application of the Main Lemma yields:

$$\sum_{|S|>k} (\hat{f}(S) - \hat{g}(S))^2 = \sum_{|S|>k} \hat{f}(S)^2 \leq \frac{\epsilon}{2}$$

By Lemma 8, with probability of at least $1 - \delta$, there holds $(\hat{g}(S) - \hat{f}(S))^2 < \epsilon/2n^k$ for every set $|S| \leq k$. Whenever this is the case g satisfies the conditions of Lemma 9, so that sign(g) disagrees with f on no more than an ϵ fraction of the inputs. ■

For more details on the material in this section, see [41].

11.4 Pseudorandom bits

The Nisan-Wigderson generator introduced in [44] is an important piece that is used in many subsequent lectures. The assumption under which we construct a generator is the existence of a "hard" function. By "hard" we mean not only that the function cannot be computed by small circuits but also that it cannot be approximated by small circuits:

Definition 149 (Pseudorandom generator) $G = \{G_n : \{0,1\}^{l(n)} \to \{0,1\}^n\}$, *denoted by* $G : l \to n$, *is called a* pseudorandom generator *if for any circuit C of size n:*

$$|Pr(C(y) = 1) - Pr(C(G(x) = 1)| < 1/n.$$

Definition 150 *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a boolean function. We say that f is* (ε, S)-hard *if for any circuit C of size S,*

$$|Pr\left(C(x) = f(x)\right) - 1/2| < \varepsilon/2.$$

Definition 151 *Let* $f : \{0,1\}^* \to \{0,1\}$ *be a boolean function, and let* f_m *be the restriction of f to strings of length m. The* Hardness *of f at m,* $H_f(m)$ *is defined to be the maximum integer* h_m *such that* f_m *is* $(1/h_m, h_m)$*-hard.*

Definition 152 *A collection of sets* $\{S_1, \ldots, S_n\}$, *where* $S_i \subset \{1, \ldots, l\}$ *is called a* (k, m)-design *if:*

- *For all i:*
$$|S_i| = m$$
- *For all $i \neq j$,*
$$|S_i \cap S_j| \leq k.$$

Definition 153 *Let A be a $n \times l$ $0-1$ matrix, let f be a boolean function, and let $x = (x_1 \dots x_l)$ be a boolean string. Denote by $f_A(x)$ the n bit vector of bits computed by applying the function f to the subsets of the $x's$ denoted by the n different rows of A.*

Lemma 154 *Let m, nl be integers; let f be a boolean function; $f : \{0,1\}^m \to \{0,1\}$, such that $H_f(m) \geq n^2$; and let A be a boolen $n \times l$ matrix which is a $(\log n, m)$ design. Then $G : l \to n$ given by $G(x) = f_A(x)$ is a pseudorandom generator.*

Theorem 155 *For every funtion s, $m \leq s(m) \leq 2^m$, the following are equivalent:*

- *For some $c > 0$, $EXPTIME$ cannot be approximated by circuits of size $s(m^c)$.*
- *For some $c > 0$, there exists a Pseudo-random generator $\{G_m : \{0,1\}^m \to \{0,1\}^{s(m^c)}\}$ that runs in time exponential in m and its output looks random to any circuit of size $s(m^c)$.*

Explicitly construct a family of functions (pseudo-random bit generators) that convert a polylogarithmic number of truly random bits to n bits that appear random to any family of circuits of polynomial size and depth d. The functions we construct are computable by a uniform family of circuits of polynomial size and constant depth. This allows us to simulate randomized constant depth polynomial size circuits in $DSPACE(polylog)$ and in $DTIME(2^{polylog})$.

This result follows from the generalized parity lower bound Theorem 145 and the NW generator for the case of AC^0.

Theorem 156 *For any integer d, there exists a family of functions: $\{f_n : \{0,1\}^l \to \{0,1\}^n\}$, where $l = O((logn)^{2d+6})$ such that:*

(1) $\{f_n\}$ can be computed by a log-space uniform family of circuits of polynomial size and $d+4$ depth

(2) For any family $\{C_n\}$ of circuits of polynomial size and depth d, for any polynomial $p(n)$, and for all large enough n:

$$|Pr(Cn(y) = 1) - Pr(Cn(fn(x)) = 1)| \leq \frac{1}{p(n)}$$

where y is chosen uniformly in $0, 1^n$, and x is chosen uniformly in $\{0,1\}^l$.

Chapter 12

Communication Complexity

Lecture(s): 37–39
Date(s): 4/5, 4/7
Lecturer: Rahul Prabhu
Scribe: Jeremy Youngquist

12.1 Introduction

The focus of these lecture notes will be on sign rank complexity of a matrix M. Letting M be an $m \times n$ matrix, we define the sign rank in four equivalent ways:

1. The sign rank complexity of M is $\min_{\tilde{M}} rank(\tilde{M})$ where $\tilde{M}$ is a matrix such that $\forall i, j$ $sign(\tilde{M}_{i,j}) = sign(M_{i,j})$

2. The sign rank of M is the minimum k such that there exists a $k \times m$ matrix B and a $n \times k$ matrix X such that $\forall i, j$ $sign((XB)_{i,j}) = sign(M_{i,j})$. We say that B *sign-realizes* the matrix M.

3. Notice that in the previous definition the rows of B span a k-dimensional subspace of an m-dimensional space. Then the sign rank complexity of M is the minimum k such that there exists a k-dimensional subspace which realizes M; call this the *subspace-realization* of M.

4. We can think of the columns of B as the normals v_j to a set of hyperplanes in $\mathbb{R}^k$ and the rows of X as vectors u_i in $\mathbb{R}^k$ such that $M_{i,j} = \langle u_i, v_j \rangle$. Then this collection of hyperplanes is called a *k-dimensional half-space realization* of M.

A main result in these notes is a theorem by Forster [22] in which he proves a lower bound in the dimension of half-space realizations. His main intent was to prove a lower bound in so-called margin complexity in machine learning, but his result reaches much further. Not only did he obtain his margin complexity result, but his theorem has corollaries in probabilistic two-party communication complexity and circuit complexity.

In communication complexity, his result gives a lower bound on the number of bits which must be exchanged between two parties who are working together to compute the value of

some distributed function $f(x, y) : X \times Y \to \{0, 1\}$ when the parties choose which bits to send to the other in a probabilistic way. In circuit complexity, his result gives a lower bound on the number of gates of a depth-2 threshold circuit with a top gate of unbounded weight and bottom gates of bounded weight. Previously the result for a bounded top gate was known and the problem with unbounded weight input gates is still open.

After stating and proving Forster's result and discussing the applications and corollaries to it, we turn our attention to Dvoretzky's theorem. This theorem is related to problems with subspace realizations and states that, given a convex body of high enough dimension, we can always find an "almost-spherical" cross section of it. Without any further ado, let's begin our first section.

12.2 Communication Complexity

The general set up, found in [50], is that we have two players, Alice and Bob, finite sets X and Y, and a distributed function $f(x, y) : X \times Y \to \{0, 1\}$. Alice is given some $x \in X$ and Bob is given some $y \in Y$ and they are given the task of computing $f(x, y)$. Alice and Bob can communicate with each other, sending elements of $\{0, 1\}$ to one another and each has unbounded computational complexity. Initially Alice sends some $a_1 \in \{0, 1\}$ to Bob. Bob sees the value of a_1 and, based on it, sends Alice some $b_1 \in \{0, 1\}$. They continue to communicate in this way until one of them can compute the value of $f(x, y)$. The number of bits exchanged between them is called the *communication complexity* and will be the focus of this section.

There are two main types of ways that the bits a_i and b_i are generated. The first is the most obvious: Alice and Bob *deterministically* generate their respective bits given all previous bits. In the second method of generating bits, the a_i and b_i are generated *probabilistically*. Given the previous bits exchanged, they generate their next bit from a probability distribution. If one party generates their bits deterministically and the other probabilistically, then this is called a *one way probabilisitic communication*. If both parties are probabilistic, then it is called a *two way probabilistic communication*.

Throughout the notes it will be useful to be able to identify a boolean distributed function $f(x, y) : \{0, 1\}^n \times \{0, 1\}^m \to \{0, 1\}$ with the $2^n \times 2^m$ matrix M_f where $m_{x,y} := (2f(x, y) - 1)$. Many communication bounds are best represented using the rank of this matrix.

12.2.1 Deterministic Communication Complexity

When we have a deterministic communication protocol, we immediately have that

$$C(f) \leq \lceil \log_2 |X| \rceil + 1$$

where $f(x, y) : X \times Y \to \{0, 1\}$ and $C(f)$ is the communication complexity of computing $f(x, y)$. The protocol is simple: Alice encodes her input x as a binary string of length $\lceil \log_2 |X| \rceil$ using some injective encoding $g : X \to \{0, 1\}^{\lceil \log_2 |X| \rceil}$ and sends $g(x)$ to Bob. Now Bob decodes Alice's message and is now in possession of both x and y and therefore can compute $f(x, y)$, which he does, and then sends it to Alice. Hence the communication takes $\lceil \log_2 |X| \rceil + 1$ bits [50].

Definition 157 (Rank) *Let $f(x,y) : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a boolean function represented by a matrix M_f. Then $rank(f)$ is equal to the rank of M_f.*

We also have a lower bound on the communication complexity: given any $f(x,y) : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ we have that

$$C(f) \geq \log\ rank(f)$$

With this result, we now know that

$$\log\ rank(f) \leq C(f) \leq \lceil \log_2 |X| \rceil + 1$$

A natural question that arises is if we can do any better than this or not. The infamous *log rank* conjecture asks this very question and has been an open problem for decades.

Conjecture 158 (Log-Rank) *For deterministic communication protocols there exists a universal constant $C > 0$ such that for every boolean function f*

$$C(f) \leq C \cdot (\log\ rank(f))^C$$

If true, this implies that the communication complexity of a function is polynomially related to the log of the rank of the matrix which computes it.

12.2.2 Probabilistic Communication Complexity

We now turn our attention to probabilistic communication protocols. An early result states that the complexity of any probabilistic communication protocol which computes a function $f(x,y) : \{0,1\}^n x \{0,1\}^n \to \{0,1\}$ with error probability bounded by $\frac{1}{2} - \frac{1}{s}$ is at least

$$\frac{1}{4}(n - \log_2 ||M_f|| - \log_2 \sqrt{s} - 2)$$

for all $s \in \mathbb{N}$ where $||M_f||$ is the operator norm of the matrix M_f. This result was later improved to remove the condition the the error is bounded to get a lower bound of

$$n - \log_2 ||M_f||$$

We will prove this second, improved result.

Definition 159 (Hadamard Matrix) *A Hadamard matrix is a square matrix made up entirely of $+1$ and -1 in which any two rows or two columns differ on exactly half of their entries (i.e. when interpreted as vectors, their rows and columns are orthogonal).*

For two finite sets X and Y we denote by $\mathbb{R}^{X \times Y}$ the set of real matrices with rows indexed by the elements of X and columns indexed by elements of Y. We denote the group of non-singular matrices $A \in \mathbb{R}^{n \times n}$ by $GL(n)$. We also define the sign function $sign(x)$ in the obvious way: $sign(x)$ is 1 if $x > 0$, is 0 if $x = 0$, and is -1 if $x < 0$. We denote by S^{k-1} the unit sphere of dimension $k - 1$.

Definition 160 (Homogeneous Half Spaces) *Given a k-dimensional space, a* half space *is a $(k-1)$-dimensional plane which separates the space into two regions. A half-space which passes through the origin is called a* homogeneous half-space.

We can use homogeneous half-spaces to represent matrices in $\mathbb{R}^{n\times m}$. A matrix $M \in \mathbb{R}^{X\times Y}$ with no zero entries, $|X| = n$ and $|Y| = m$ can be realized by a k-dimensional linear arrangement of homogeneous half spaces if there are vectors $u_i, v_j \in R^k$ for $i \in [n], j \in [m]$ such that $sign(M_{ij}) = sign\langle u_i, v_j\rangle$. We then say that k is the dimension of the arrangement.

Equivalently, given a matrix $M \in \mathbb{R}^{n\times m}$, we can represent it by homogeneous half-spaces by finding points $u_i \in \mathbb{R}^k$ for every row $i \in [n]$ of M and half-spaces $H_j \subseteq \mathbb{R}^k$ for every column $j \in [m]$ of M such that u_i lies in the half-space H_j if and only if $sign(M_{ij}) = 1$.

Theorem 161 *Let $X \subseteq \mathbb{R}^k$, $|X| \geq k$, such that all subsets of X with at most k elements are linearly independent. Then there is a nonsingular linear mapping $A \in GL(k)$ such that*

$$M(A \cdot X \cdot N) = \sum_{x\in X} \frac{1}{||Ax||^2}(Ax)(Ax)^T = \frac{|X|}{k} I_k$$

Where $A \in GL(k)$ is a nonsingular linear mapping, $N(X)$ normalizes the vectors in X ($N(X) = \frac{x}{||x||}$ for all $x \in X$), and

$$M(X) := \sum_{x\in X} xx^T \in \mathbb{R}^{k\times k}$$

If X is $k \times n$ matrix, then A is a $k \times k$ nonsingular linear transformation and N is an $n \times n$ matrix which normalizes $A \cdot X$. This theorem is the technical crux of the main theorem which is to follow since it allows us to find a "nice" realization of a set of vectors such that the vectors are balanced.

Definition 162 (Niceness) *A $k \times m$ matrix B is said to be* nice *if its rows form an orthogonal basis with the basis vectors all having l_2 norm equal to $\sqrt{m/k}$ and its columns all have an l_2 norm at most 1. We also say that a k-dimensional realization B of M is a* nice subspace *if the infinity-norm of all unit vectors in the subspace is at most $\sqrt{k/m}$.*

In order to prove the important Theorem 161 we need the following two lemmas:

Lemma 163 *Let $X \subseteq S^{k-1}$, $|X| \geq k$, such that all subsets of X with at most k elements are linearly independent. Then either $M(X) = \frac{|X|}{k} I_k$ or there is some nonsingular linear mapping $A \in GL(k)$ such that the smallest eigenvalue of*

$$M(A \cdot X \cdot N) = \sum_{x\in X} \frac{1}{||Ax||^2}(Ax)(Ax)^T$$

is strictly larger than the smallest eigenvalue of $M(X)$.

This lemma allows us to construct a sequence of A_i and N_i such that the sequence $\{(A_1 \cdot X \cdot N_1), (A_2 \cdot (A_1 \cdot X \cdot N_1) \cdot N_2), ...\}$ has strictly increasing smallest eigenvalues. The matrix A in the statement of the theorem will be $A = A_n \cdot A_{n-1} \cdots A_1$.

Lemma 164 *Let $X \subseteq \mathbb{R}^k$, $|X| \geq k$, such that all subsets of X with at most k elements are linearly independent. Then for every $\epsilon \geq 0$ the subset*

$$\mathcal{A} := \{A \in GL(k) : ||A|| = 1, M(N(A(X))) \geq (1+\epsilon)I_k\}$$

of $\mathbb{R}^{k \times k}$ is compact.

This lemma assures us that if we have a convergent sequence of nonsingular linear mappings that satisfy certain criteria, then the matrix to which they converge is also a nonsingular linear map, by the definition of compact. We will use Lemma 163 to construct a convergent sequence of such matrices, and then use Lemma 164 to show that the matrix to which it converges is also a nonsingular linear map.

Proof: We will now prove Theorem 161. Let A be a linear mapping that maps k arbitrarily chosen subsets of X to the canonical unit vectors of $\mathbb{R}^k$. If $|X| = k$ then we are done.

Assume $|X| > k$. Then $M(A \cdot X \cdot N) \geq I_k$, that is, the smallest eigenvalue of $M(A \cdot X \cdot N)$ is at least 1. By Lemma 163 we have that if $M(A \cdot X \cdot N) \neq \frac{|X|}{k} I_k$ then we can modify A such that the smallest eigenvalue of $M(A \cdot X \cdot N)$ increases. Hence, we can find some A' and $\epsilon > 0$ such that the smallest eigenvalue of $M(A' \cdot X \cdot N)$ is $1 + \epsilon$.

By Lemma 164 we have that the set of all $A \in GL(k)$, $||A|| = 1$ for which the smallest eigenvalue of $M(A \cdot X \cdot N)$ is at least $1+\epsilon$ is compact. The smallest eigenvalue of a symmetric positive semi-definite matrix is equal to the smallest singular value, and the singular values depend continuously on the matrix. Hence, the smallest eigenvalue of $M(A \cdot X \cdot N)$ is a continuous function of $A \in GL(k)$ and there exists some $A \in GL(k)$ for which the smallest eigenvalue of $M(A \cdot X \cdot N)$ is maximal.

For this A, it is the case that $M(A \cdot X \cdot N) = \frac{|X|}{k} I_k$ because if not the we could apply Lemma 163 and increase the smallest eigenvalue of $M(A \cdot X \cdot N)$, contradicting the fact that the smallest eigenvalue of $M(A \cdot X \cdot N)$ is maximal. Hence, such a "nice" realization of the vectors exists. ■

We are now in a position to state the following two theorems:

Theorem 165 *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a distributed function. If k is the smallest dimension of an arrangement of homogeneous half-spaces that realize M_f, then*

$$\lceil \log_2 k \rceil \leq C(f) \leq \lceil \log_2 k \rceil + 1$$

The proof of the above theorem uses covering rectangles and we did not prove it in class. The following theorem is the main theorem of this section and it is due to [22].

Theorem 166 (Main Theorem) *If a matrix $M \in \{1,-1\}^{\{X \times Y\}}$ can be realized by an arrangement of homogeneous half spaces in $\mathbb{R}^k$, then*

$$k \geq \frac{\sqrt{|X||Y|}}{||M||}$$

Proof: We will prove the second of these two theorems.

Assume there are vectors $u_i, v_j \in \mathbb{R}^k$ such that $sign\langle u_i, v_j\rangle = M_{ij}$, i.e. M can be realized by an arrangement of homogeneous half spaces in $\mathbb{R}^k$. Note that normalizing u_i and v_j will not effect the sign of their inner product, so we can assume they lie on the $k-1$ dimensional unit sphere S^{k-1}. If $A \in GL(k)$ is a nonsingular linear transformation, then replacing u_i, v_j with $(A^T)^{-1}u_i, Av_j$ also does not effect the sign of the inner product. By Theorem 161 we can then find a linear mapping which transforms a given set of v_j's into a set of "nice" v_j's such that $\sum_j v_j v_j^T = \frac{m}{k} I_k$, i.e. the vectors are nicely balanced. If we have such "nice" v_j's, then we also have that

$$\sum_j |\langle u_i, v_j\rangle| \geq \sum_j \langle u_i, v_j\rangle^2 = u_i^T (\sum_j v_j v_j^T) u_i = \frac{m}{k}$$

This means that for all j, $|\langle u_i, v_j\rangle|$ are on average $\frac{1}{k}$, implying that the u_i cannot lie arbitrarily close to the homogeneous half-space which has normal v_j.

We then have that the upper bound in terms of the operator norm $||M||$ is

$$\sum_i \left(\sum_j |\langle u_i, v_j\rangle| \right)^2 \leq m||M||^2$$

From these two bounds it follows that

$$n\left(\frac{m}{k}\right)^2 \leq \sum_i \left(\sum_j |\langle u_i, v_j\rangle| \right)^2 \leq m||M||^2$$

from which the desired lower bound immediately follows. ■

From these two theorems we get the following two corollaries:

Corollary 167 *For every distributed function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ the communication complexity is at least*

$$C(f) \geq n - log_2||M_f||.$$

Corollary 168 *The communication complexity of the distributed function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ for which M_f is the Hadamard matrix H_n is at least $\frac{n}{2}$. The Hadamard matrix H_n can only be realized by an arrangement of homogeneous half spaces in $\mathbb{R}^k$ if $k \geq 2^{\frac{n}{2}}$.*

12.3 Threshold Circuit Complexity Lower Bounds

The results of the previous section have been successfully applied to prove lower bounds on the size of depth-2 threshold circuits in [23].

Definition 169 (Linear Threshold Gate) *A* linear threshold gate *is a boolean gate that outputs 1 if the weighted sum of it's inputs is greater than some threshold t and 0 otherwise.*

Theorem 170 *Let (f_n) be a family of distributed boolean functions $f_n : \{0,1\}^n \times \{0,1\}^n \to \{-1,+1\}$. Suppose (f_n) is computed by depth-2 threshold circuits in which the top gate is a linear threshold gate with unrestricted weights. Then if the bottom level has s linear threshold gates using integer weights of absolute value at most W, then $s = \Omega\left(\frac{d(f_n)}{n \cdot W}\right)$ where $d(f)$ is the minimal dimension of f.*

The proof of this theorem is based on Theorem 161, along with the following two lemmas:

Lemma 171 *Let $G : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a threshold function where for $x, y \in \{0,1\}^n$ such that $G(x,y) = 1$ if and only if*

$$\sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} \beta_i y_i \geq \mu$$

for weights α_i, β_i, $\mu \in \mathbb{Z}$. Then G, when viewed as a matrix, has rank at most $min\{\sum_{i=1}^{n} |\alpha_i|, \sum_{i=1}^{n} |\beta_i|\} + 1$.

Lemma 172 *Let $f_n : \{0,1\}^n \times \{0,1\}^n \to \{-1,+1\}$ be a Boolean function computed by a depth-2 threshold circuit C with a top gate using unrestricted weights and each of the bottom gates using weights of absolute value at most W. Then there is a real matrix F such that $sign(F_{x,y}) = f(x,y)$ for all $x, y \in \{0,1\}^n$ and $rank(F) = O(s \cdot n \cdot W)$ where s is the number of bottom gates.*

Proof: The proof of Theorem 170 is straightforward. By Lemma 172, if a depth-2 threshold circuit computes $f_n(s,y)$, then there is a matrx F_n such that $sign(F_{x,y}) = f(x,y)$ and $rank(F) = O(s \cdot n \cdot W)$. We also have that, since there is always a $rank(F_n)$-dimensional linear arrangement for F_n, which is also a linear arrangement for f_n, that $rank(F_n) \geq d(f_n)$. Comparing the upper and lower bounds gives $s \cdot n \cdot W = \Omega(d(f_n))$. ■

12.4 Margin Lower Bounds

We can also apply the techniques that are in the proof to learning theory, allowing us to prove stronger general bounds on margins of maximal margin classifiers, as found in [17]. This is the area the Forster initially set out to prove results in when he proved Theorem 166

Definition 173 (Margin) *We say that the matrix $M \in \mathbb{R}^{X \times Y}$ with no zero entries can be realized by an arrangement of homogeneous half spaces with* margin γ *if there are vectors u_x, v_y for $x \in X$, $y \in Y$ that lie in the unit ball of $\mathbb{R}^k$ such that $sign(M_{x,y}) = sign\langle u_x, v_y \rangle$ and $|\langle u_x, v_y \rangle| \geq \gamma$ for all $x \in X$, $y \in Y$.*

Definition 174 (Maximum Margin Classifier) *A* maximal margin classifier *is a learning algorithm which computes the hyperplane that separates a sample with the largest margin and uses this hyperplane to classify new instances.*

Definition 175 (Concept Class) *In the context of machine learning, given X, a set of possible instances, a* concept *is a function $c : X \to \{0, 1\}$. We then form C, a set of these functions. A* concept class *is then the pair (X, C). We can also define a concept class as the matrix*

$$M_{X,C} \in \{-1, 1\}^{n \times m}$$

where each entry is

$$(M_{X,C})_{x,c} = \begin{cases} 1 \ if \ x \in C \\ -1 \ otherwise \end{cases}$$

Given a matrix which is realizable by an arrangement of homogeneous half spaces, we might wonder how the the margin is effected by the size of the spaces in which the matrix is realized. The next theorem places an upper bound on the size of the margin of such a realization, but first we need the following lemma:

Lemma 176 *For finite sets X, Y of S^{k-1} let*

$$M := (sign\langle u_x, v_y\rangle)_{x \in X, y \in Y} \in \mathbb{R}^{X \times Y}.$$

Then

$$\sum_{y \in Y} \left(\sum_{x \in X} |\langle u_x, v_y\rangle| \right)^2 \leq |X| \|M\|^2$$

We are now in a position to state and prove the main result of this section:

Theorem 177 *If a matrix $M \in \{0, 1\}^{X \times Y}$ can be realized by an arrangement of homogeneous half spaces with margin γ, then*

$$\gamma \leq \frac{||M||}{\sqrt{|X||Y|}}$$

Proof: Without loss of generality, let X, $Y \subseteq S^{k-1}$, $M = (sign\langle u_x, v_y\rangle)_{x \in X, y \in Y}$ and $|\langle u_x, v_y\rangle| \geq \gamma$ for all $x \in X$, $y \in Y$. From the previous lemma it follows that $|Y||(|X|_\gamma)^2 \leq |X|\|M\|^2$. ■

12.5 Dvoretzky's Theorem

We will now revisit the crux of Theorem 166, Theorem 161. The "nice" realizations that we found in Theorem 161 are highly related to ϵ-Euclidean spaces, which Dvoretzky's theorem tells us we can always find in high-dimensional convex bodies. We first start with some preliminaries. Letting S^{k-1} be the $k-1$-dimensional unit ball, we have the following lemma:

Lemma 178 *Given $M \in \mathbb{R}^{n \times m}$ and $u_i, v_j \in S^{k-1}$ such that $sign\langle u_i, v_j\rangle = sign(M_{ij})$ for all i, j, construct the $k \times m$ matrix B such that the v_j's are the columns of B and let $w_i = \sqrt{k/m} \cdot u_i \cdot B$. Then the rows of the matrix B form a basis for the k-dimensional subspace that realizes M.*

Proof: Note that the vectors $w_i = u_i B$ are just the u_i expressed in basis B, and hence are nothing but the u_i living in the ambient m-dimensional space B. Denote the j^{th} coordinate of w_i by $w_i(j)$. Then $w_i(j) = \sqrt{k/m}\langle u_i, B_j\rangle$. Since by construction the columns B_j of the matrix are just the v_j, we have that

$$sign(w_i(j)) = sign\left(\sqrt{\frac{k}{m}}\langle u_i, B_j\rangle\right) = sign\left(\sqrt{\frac{k}{m}}\langle u_i, v_j\rangle\right) = sign(M_{ij})$$

for all i, j. Hence B satisfies the definition of a subspace realizing the matrix M. ■

Definition 179 (Almost Spherical) *A k-dimensional realization B of M is said to be almost-spherical if the one-norm of all unit vectors in B is at least $\sqrt{m/k}$.*

The previous definition only gives a lower bound on the the one-norm of the unit vectors of B, which raises the question of why such a matrix B is called "almost spherical". It turns out that if we are given a lower bound of the size of the vectors in such a matrix, we can also find an upper bound.

Definition 180 (Nice Subspace) *A k-dimensional ralization B of M is a* nice subspace *if the infinity norm of all unit vectors in the subspace is at most $\sqrt{k/m}$.*

If B is a subspace of $\mathbb{R}^m$, we have the following theorems:

Theorem 181 *If B is a nice subspace, then B is almost spherical.*

Theorem 182 *B is realized by a nice matrix if and only if it is a nice subspace.*

The previous two theorems give us the following corollary:

Corollary 183 *If B is realized by a nice matrix, then B is almost spherical.*

Now that we have built up some of the machinery and intuition related to Dvoretzky's theorem, we can finally explicitly state it as follows:

Theorem 184 (Dvoretzky's Theorem) *For every $k \geq 2$ and every $\epsilon > 0$ there exists an $N = N(k, \epsilon)$ such that every normed space (X, p) with $dim(X) \geq N$ contains a k-dimensional subspace that is ϵ-Euclidean.*

Definition 185 (ϵ-Euclidean) *A normed space (X, p) is ϵ-Euclidean if there exists an inner-product norm, say $|\cdot|$ and a constant C such that $\forall x \in X$*

$$C(1-\epsilon)|x| \leq p(x) \leq C|x|$$

There are many (≥ 10) proofs of Dvoretzky's theorem, almost all of which rely on the probabilistic method. As a result, they show that such an ϵ-Euclidean subspace exists, but they fail to explicitly give one. Derandomizing these proofs to give an explicit ϵ-Euclidean subspace is still an open problem.

Chapter 13

Proof Barriers

Lecture(s): 40–42
Date(s): 4/12, 4/14
Lecturer: Kiara Hall
Scribe: Serdar Ayaz

13.1 Relativization barrier

The first of the proof barriers to be used in complexity classes is relativization barrier. It is first introduced by Baker, Gill and Solovay [10]. Baker-Gill-Solovay theorem shows that relativization using oracles cannot be used to prove $P \stackrel{?}{=} NP$ problem. Here is the formal definition of the theorem:

Theorem 186 (Baker-Gill-Solovay) *There exist oracles A and B such that $P^A = NP^A$ and $P^B \neq NP^B$.*

Discovery of this problem have directed the researchers to use other methods than Turing machines to describe complexity classes. One of the most used complexity class definitions are based on Boolean circuit complexity. Since we have already covered this theorem previously this semester, I do not include the details of the theorem.

13.2 Natural proofs barrier

Razborov and Rudich have introduced a new framework on proving lower bounds based on Boolean circiut complexity [51]. They call this framework as *natural proofs* and show that several different previous proofs can be *naturalized*, i.e., can be expressed using natural proof framework. Later they show some limitations on natural proofs on some problems including $P \stackrel{?}{=} NP$ problem, which is called *natural proof barrier*.

13.2.1 Notation and definitions

The set of Boolean functions with n variables is denoted as F_n. A Boolean function $f_n \in F_n$ can be described as the *truth table* of the n variables. Note that the size of a truth table is

2^n whereas the size of F_n is 2^{2^n}. Bold characters show a random object in a set, e.g. $\boldsymbol{f_n}$ is a random element of F_n.

Here are some definition of complexity classes based on Boolean circuits:

Definition 187 ($AC^0[m]$) *The class of functions computable by circuits that allow MOD_m gates in addition to AND and OR gates with a depth bounded by a polynomial of the input size.*

Definition 188 (TC^0) *The class of functions computable by circuits with bounded depth that allow threshold gates in addition to AND, OR and NOT gates.*

Definition 189 ($P/poly$) *The class of functions computable in polynomial time with unbounded-depth circuits and a polynomial size advice function.*

13.2.2 Natural combinatorial properties and its generalization

Definition 190 (Combinatorial properties) *A set of Boolean functions $C_n \subseteq F_n$ is called combinatorial property. A function f_n has the property C_n iff $f_n \in C_n$. The function notation is also used to denote a property, i.e., $C_n(f_n) = 1$ means $f_n \in C_n$.*

Definition 191 (Natural combinatorial properties) *A combinatorial property C_n is* natural *if it contains has a subset $C_n^* \subseteq C_n$ has constructivity and largeness properties:*

Constructivity: The predicate is $f_n \overset{?}{\in} C_n^*$ is computable in polynomial time with respect to the truth table of f_n, which has a size of 2^n.

Largeness: $|C_n^*| \geq 2^{-O(n)} \cdot |F_n|$.

The authors use "there exists (no) natural proof" as equivalent to "there exists (no) natural combinatorial property. In addition to constructivity and largeness, a combinatorial property may have one more property:

Definition 192 (Usefulness) *A combinatorial property C_n is useful against $P/poly$ if the circuit size of any sequence of functions in C_n is super-polynomial, i.e, for any constant k, the circuit size of $f_n \in C_n$ is in $\omega(n^k)$. If C_n is a combinatorial property that is useful against $P/poly$, C_n is said to be natural against $P/poly$.*

Here, usefulness is defined against $P/poly$, because a natural combinatorial property that C_n that is useful against $P/poly$ implies that any function g_n that is in C_n does not have a polynomial-sized circuit. It also implies that the g_n is hard to compute.

The reason of defining largeness property is to make $\boldsymbol{f_n}$ having a considerable chance to be in C_n.

The natural combinatorial properties defined above is the default version. It can be generalized with three two complexity classes (Γ and Λ) and a density function (δ_n):

Definition 193 (Generalized natural combinatorial properties) *A combinatorial property C_n is Γ-natural against Λ with density δ_n if it has a subset $C_n^* \subseteq C_n$ that has properties:*

Constructivity: The predicate is $f_n \overset{?}{\in} C_n^*$ is computable in Γ.

Largeness: $|C_n^*| \geq \delta_n \cdot |F_n|$.

Usefulness: If $f_n \in C_n$ happens infinitely often, $\{f_n\}$ cannot be computable in Λ.

13.2.3 Inherent limitations of natural proofs (Natural proof barrier)

Here is the general strategy for proving $P \neq NP$ before the natural proof barrier is presented:

- Formulate a mathematical notion of "discrepancy" of the values of a Boolean function. (Here it is a combinatorial property C_n with functions of "high" discrepancy defined as natural).
- Show that polynomial-sized circuits cannot compute functions with "high" discrepancy by induction. (Show that C_n is useful against $P/poly$).
- Show that a function in NP has "high" discrepancy. (Show that SAT is in C_n).

The authors have proven that this strategy does not work by showing any combinatorial property that is natural against $P/poly$ gives a statistical test that breaks all the polynomial time pseudo-random generator.

Definition 194 (Hardness of a pseudo-random generator) *For a pseudo-random generator $G_k : \{0,1\}^k \to \{0,1\}^{2k}$ the hardness $H(G_k)$ is the minimal S for which there exists a circuit C of which size not greater than S such that*

$$|\boldsymbol{P}[C(G_k(\boldsymbol{x})) = 1] - \boldsymbol{P}[C(\boldsymbol{y}) = 1]| \geq 1/S$$

where $\boldsymbol{x}$ is a random string from $\{0,1\}^k$ and $\boldsymbol{y}$ is a random string from $\{0,1\}^{2k}$.

Claim 195 (Pseudo-random generator conjecture) *There exist pseudo-random generators of hardness 2^{n^ϵ} for some $\epsilon > 0$.*

Natural proofs are described as "self defeating" since their associated functions can be used against them. More specifically, if C_n is $P/poly$-natural against $P/poly$, there is an associated algorithm that distinguishes a super-polynomial function $f_n \in C_n$ from a pseudo-random function in $P/poly$. This algorithm can be used to break pseudo-random generators by a constructive statistical test described in detail in [51].

Nisan's [43] pseudo-random generator that is secure against AC^0-attack yields the following theorem:

Theorem 196 *For any integer d, there exists a family $G_{n,s} \subseteq F_n$ where s is a seed of size polynomial in n such that $G_{n,s} \in AC^0[2]$ and $G_{n,s}$ looks random for $2^{O(n)}$-size, d-depth circuit family $C_n : F_n \to \{0,1\}$,*

$$|\boldsymbol{P}[C_n(\boldsymbol{f_n}) = 1] - \boldsymbol{P}[C_n(G_{n,\boldsymbol{s}}) = 1]| < 2^{-\omega(n)} \tag{13.1}$$

Here $\boldsymbol{s}$ is a random seed of the appropriate size.

Theorem 197 *There is no lower bound proof which is AC^0-natural against $AC^0[2]$.*

Proof: Suppose that such a lower bound proof exists and C_n is associated combinatorial property that is AC^0-natural against $AC^0[2]$. Let d be the depth of a size $2^{O(n)}$ circuit to compute C_n. Let $G_{n,s}$ be the generator which is pseudo-random against d-depth $2^{O(s)}$-sized circuits from Theorem 196. From the definition of a proof natural against $AC^0[2]$ for sufficiently large n, we have $C_n(G_{n,s}) = 0$. From this and (13.1), we have $\boldsymbol{P}[C_n(\boldsymbol{f_n}) = 1] < 2^{-\omega(n)}$ which contradicts the largeness condition. ■

Corollary 198 *Theorem 197 can be generalized as follows: A complexity class Λ that contains pseudo-random function generators that are sufficiently secure against a Γ-attack implies there is no Γ-natural proof against Λ.*

13.2.4 Natural proof barrier for the discrete logarithm problem

Here Razborov and Rudich prove that natural proofs cannot show that the discrete logarithm problem requires exponential-sized circuits. They use the fact that discrete logarithm problem is hard on average iff it is hard on the worst case.

For a prime p and a generator g for the finite field $\mathbb{Z}_p^*$, the predicate $B_{p,g}(x)$ on $\mathbb{Z}_p^*$ is defined as 1 if $\log_g x \leq (p-1)/2$ and 0 otherwise. Blum and Micali [12] has shown that $B_{p,g}(x)$ is the hard bit of discrete logarithm problem. $B_{p,g}(x)$ is padded to be a Boolean function in $\lceil \log(p) \rceil$ variables.

Definition 199 ($SIZE$) *Let $t(n)$ be an integer-valued function. $SIZE(t(n))$ is the complexity class consisting of all the functions with circuit size $O(t(n))$.*

Definition 200 (Half-exponential functions) *Let $t(n)$ be an integer-valued function. Let $t^{-1}(n)$ is defined as $\max\{x|t(x) \leq n\}$. It is said that $t(n)$ is half-exponential if it is non-decreasing and*

$$t^{-1}(n^C) \leq o(\log t(n)) \tag{13.2}$$

for every $C > 0$.

Theorem 201 *Let $t(n)$ be an arbitrary half-exponential function. Then there is no combinatorial propery C_n useful against $SIZE(t(n))$ and satisfying $P/poly$-constructivity and largeness conditions such that $\bigcup_{n\in\omega} C_n$ contains infinitely many functions of the form $B_{p,g}(x)$.*

Proof: Assume the contrary, let $\{B_{p_v,g_v}\}$ be the infinite sequence contained in $\bigcup_{n\in\omega} C_n$ such that $\lceil \log(p_1) \rceil < \lceil \log(p_2) \rceil < ...$ Let k_v be defined as $\lceil \log(p_v) \rceil$. Half-exponentialness and usefulness yield that, there exists a subsequence of $\{B_{p_v,g_v}\}$ where all the functions have a circuit size at least $t(k_v)$. Without loss of generality, assume this subsequence is the sequence itself.

Let $G_v : \{0,1\}^{2k_v} \to \{0,1\}^{4k_v}$ be the pseudo-random generator based on $\{B_{p_v,g_v}\}$. Using a proof in [12], it can be shown that the circuit size of $\{B_{p_v,g_v}\}$ is polynomial in $H(G_v)+k_v$. Therefore we have:

$$t(k_v) \leq (H(G_v) + k_v)^{O(1)}. \tag{13.3}$$

Now we construct a pseudo-random function generator $f_v : \{0,1\}^{2k_v} \to F_{n_v}$ using G_v. Let us define n_v as $t^{-1}(k_v^{C+1})+1$ where $C > 0$ is a constant that makes $f_v(x)(y)$ computable by $(k_v + n_v)^C$-sized circuits for almost all v.

From half-exponentialness, $k_v > (C+1)\log k_v$ and from the definition of k^{-1}, $k_v \leq k^{-1}(k_v) \leq k^{-1}(k_v^{C+1})$.. From (13.2), $k^{-1}(k_v^{C+1}) \leq \log t(k_v)$. Therefore we have $k_v^{C+1} < t(k_v)$. So $n_v = t^{-1}(k_v^{C+1}) + 1 \leq t^{-1}(t(k_v)) + 1 \leq k_v$. Then the circuit size of f_v is $(k_v + n_v)^C \leq (2k_v)^C \leq k_v^{C+1} \leq t(n_v)$ for almost all v. Then we apply the usefulness condition and find that for almost all v, the image of the generator f_v has the empty intersection with C_n. This gives us,

$$H(G_v) \leq 2^{O(n_v)} \tag{13.4}$$

Then by the largeness condition, we would have that $C_n \neq \emptyset$ for almost all n, therefore we have:

$$t(n) \leq 2^n \tag{13.5}$$

From the definition $n_v = t^{-1}(k_v^{C+1}) + 1$.
From (13.2), we have $t^{-1}(k_v^{C+1}) + 1 \leq o(\log t(k_v))$.
From (13.3), we have $o(\log t(k_v)) \leq o(\log(H(G_v) + k_v))$.
From (13.5), we have $o(\log(H(G_v) + k_v)) \leq o(\log H(G_v) + \log k_v)$.
From (13.4), we have $o(\log H(G_v)) \leq o(O(n_v)) \leq o(n_v)$
From (13.2), we have $o(\log k_v) \leq o(n_v)$.
These yield $n_v \leq o(n_v)$ which is a contradiction. ■

13.2.5 Naturalization

As it is mentioned before, several different proofs can be shown on a natural proof scheme, which is called *naturalization*. Here we present Smolensky's proof on a natural circuit lower bound [56].

Theorem 202 (Smolensky) *Parity does not have small $AC^0[3]$ circuits.*

First, we need to define the polynomial approximation of a Boolean function. Boolean value TRUE corresponds to -1 and FALSE corresponds to 1. Let f be a Boolean function and let p be a polynomial over $\mathbb{Z}^3$ where they have the same variable names. The smaller the number of assignments A such that $p(A) \neq f(A)$ the better p approximates f. Since the variables are all in {-1, 1} the polynomials are multi-linear. So we do not need to consider higher powers than 1.

Proof: The original proof has two main parts:

1. Any f computed by "small" $AC^0[3]$ circuits can be "reasonably" approximated by a "low" degree polynomial over $\mathbb{Z}^3$.

2. The parity function in n variables cannot be "reasonably" approximated by a "low" degree polynomial over $\mathbb{Z}^3$.

The first part of the proof is skipped. Suppose that p is a polynomial in $\mathbb{Z}^3$ with a low degree d and agrees with the parity function on all but a small number of assignments. Let W be the set of assignments f and p differ. Let $w = |W|$ and $N = 2^n$. Assume that n is odd and l_1 and l_2 are polynomials with degrees less than $n/2$. Since we are dealing with multi-linear polynomials, any polynomial q over $\mathbb{Z}^3$ can be represented as $pl_1 + l_2$. Then we have $deg(q) = \max(deg(pl_1), deg(l_2)) \geq d + (n-1)/2$. A polynomial with degree $d + (n-1)/2$ cannot represent all the polynomials over $\mathbb{Z}^3$ on {-1, 1}$^n \setminus W$ which has 3^{N-w} functions. ■

To naturalize this proof, let us start with choosing C_n as the set of functions that cannot be approximated by a low degree polynomial over $\mathbb{Z}^3$. First part of the proof shows that C_n is useful against $AC^0[3]$.

Let L be the vector space of all polynomials with degree less than $n/2$ and T be the compliment of L. Both of them have dimensions $N/2$.

Then we need to choose C_n^* that satisfies largeness and constructivity properties. Let us define C_n^* as the functions such that every polynomial such that $dim(\overline{f}_n L + L) \geq 3N/4$ where $\overline{f}_n$ is the multi-linear polynomial representing f_n. Since we use $\overline{f}_n$ we have constructivity. We also have largeness. If $C_n^*(f_n) = 0$, we have:

$$\begin{aligned} dim((r\overline{f}_n L + L)/L) &= dim((rL + \overline{f}_n L)/\overline{f}_n L) \\ &\geq dim((rL + \overline{f}_n L + L)/(\overline{f}_n L + L)) \\ &= dim((T + L)/(\overline{f}_n L + L)) \\ &= dim(T + L) - dim(\overline{f}_n L + L) \\ &\geq N - 3N/4 \\ &= N/4 \end{aligned}$$

So $|C_n| \geq \delta_n |F_n|$ if $\delta_n = 1/4$.

13.3 Algebrizition (Algebraic relativization) barrier

Relativization and natural proof barriers were capable of proving most of the lower bound proofs until some researchers come up with a combination of arithmetization (same as algebrization) and diagonalization. Relativization does not work on these proofs since arithmerization does not relativize. Natural proofs do not work either because diagonalization do not naturalize. To draw a limit for these proofs, Aaronson and Widgerson came up with a new barrier: Algebraic relativization or algebrization [3]. Here, we present their algebrization proofs and the limitations of this proof technique.

13.3.1 Notation and definitions

Here we use similar definitions with the previous section. However we need to add some new definitions described in [3].

Definition 203 (mdeg) *Given a multivariate polynomial $p(x_1, ..., x_n)$, the multidegree of p, or mdeg(p), is the maximum degree of any x_i.*

Definition 204 (Extension oracle over a finite field) *Let $A_m : \{0,1\}^m \to \{0,1\}$ be a Boolean function, and let $\mathbb{F}$ be a finite field. Then an extension of A_m over $\mathbb{F}$ is a polynomial $\widetilde{A}_{m,\mathbb{F}} : \mathbb{F}^m \to \mathbb{F}$ such that $\widetilde{A}_{m,\mathbb{F}(x)} = A_m(x)$ whenever $x \in \{0,1\}^m$. Also, given an oracle $A = (A_m)$, an extension $\widetilde{A}$ of A is a collection of polynomials $\widetilde{A}_{m,\mathbb{F}} : \mathbb{F}^m \to \mathbb{F}$ one for each positive integer m and finite field $\mathbb{F}$, such that:*

1. *$\widetilde{A}_{m,\mathbb{F}}$ is an extension of A_m for all $m, \mathbb{F}$, and*
2. *there exists a constant c such that $mdeg(\widetilde{A}_{m,\mathbb{F}}) \leq c$ for all $m, \mathbb{F}$.*

Then given a complexity class $\mathcal{C}$, $\mathcal{C}^{\widetilde{A}}$ means the class of languages decidable by a $\mathcal{C}$ machine that can query $\widetilde{A}_{m,\mathbb{F}}$ for any integer m and finite field $\mathbb{F}$. $\mathcal{C}^{\widetilde{A}[poly]}$ means the class of languages decidable by a $\mathcal{C}$ machine that, on inputs of length n, can query $\widetilde{A}_{m,\mathbb{F}}$ for any integer $m = O(poly(n))$ and finite field with $|\mathbb{F}| = 2^{O(m)}$.

There are two different definitions for inclusion and separation of complexity classes:

Definition 205 (Algebrization) *The complexity class inclusion $\mathcal{C} \subseteq \mathcal{D}$ algebrizes if $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ for all oracles A and all finite field extensions $\widetilde{A}$ of A. Likewise, $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ does not algebrize, or that proving $\mathcal{C} \subseteq \mathcal{D}$ would require non-algebrizing techniques, if there exist A, $\widetilde{A}$ such that $\mathcal{C}^A \not\subseteq \mathcal{D}^{\widetilde{A}}$.*

The separation $\mathcal{C} \not\subseteq \mathcal{D}$ algebrizes if $\mathcal{C}^{\widetilde{A}} \not\subseteq \mathcal{D}^A$ for all $A, \widetilde{A}$. Likewise, $\mathcal{C} \not\subseteq \mathcal{D}$ does not algebrize, or that proving $\mathcal{C} \not\subseteq \mathcal{D}$ would require non-algebrizing techniques, if there exist A, $\widetilde{A}$ such that $\mathcal{C}^{\widetilde{A}} \subseteq \mathcal{D}^A$.

Note that only one side of the relations can reach the extension oracles.

13.3.2 Algebrization example

In this subsection, it is proven that the interactive proof results of Lund et al. [42] and Shamir [54] algebrize.

Theorem 206 *For all A, $\widetilde{A}$, we have $PSPACE^{A[poly]} \subseteq IP^{\widetilde{A}}$.*

Sketch of Proof: We can generalize the $\#P$ protocol in [42] to the $PSPACE$ protocol in [54]. We give an interactive proof protocol for true quantified boolean formula (TQBF), but we have to arithmetize quantifiers. when given a boolean function f , we extend this to a polynomial on some $p(x_1, ..., x_n)$ over some finite field $\mathbb{F}$. We lose quantifiers roughly in the following way

$$\forall x \exists y \text{ such that } f(x,y) \text{ is true} \iff \prod_i \sum_j p(i,j) \tag{13.6}$$

The prover now needs to prevent the degrees of the relevant polynomials from doubling at each iteration. To achieve this, we have to add steps to the degree reduction. The only step of the the protocol that is relevant for algebrization is the last one. In this step, the verifier must evaluate the arithmetized polynomial directly, that is when the verifier checks that $p(r_1, ..., r_N)$ is equal to the value claimed by the prover for some $r_1, ..., r_N \in \mathbb{F}$. The verifier can do this by using an algebraic oracle. ■

13.3.3 Non-algebrizing techniques needed for P vs. NP (Algebrization barrier)

Here the authors show that we need to use different techniques than algebrization to prove $P \stackrel{?}{=} NP$. Next two theorems show there are extension oracles that can used for both separation and inclusion of P and NP.

Theorem 207 *There exists A and $\widetilde{A}$ such that $NP^{\widetilde{A}} \subseteq P^A$.*

Proof: Let A be any PSPACE-complete language, and $\widetilde{A}$ be the unique multilinear extension of A. As a multilinear extension of language in $PSPACE$, $\widetilde{A}$ is a language within $PSPACE$ as well. So by an argument in Baker, Gill, and Solovay [10], we have $NP^{\widetilde{A}} = NP^{PSPACE} = PSPACE = P^A$ ■

Theorem 208 *There exist A and $\widetilde{A}$ such that $NP^A \not\subset P^{\widetilde{A}}$. Furthermore, the language L that achieves the separation simply corresponds to deciding, on inputs of length n, whether there exists a $w \in \{0,1\}^n$ with $A_n(w) = 1$.*

13.3.4 The integers case

For the sake of simplicity, algebrization techniques are defined over (finite) fields. We can use low-degree extensions over $\mathbb{Z}$ considering some complications. Unlike fields, multiplication does not have an inverse over integers, therefore we cannot use methods like Gaussian elimination. Also we need to consider the size of the input and output of extension oracles where the *size* of an integer vector $v = (v_1, ..., v_n)$ is defined as:

$$\text{size}(v) := \sum_{i=1}^{n} \lceil \lg(|v_i| + 2) \rceil$$

Definition 209 (Extension oracle over the integers) *Let $A_m : \{0,1\}^m \to \{0,1\}$ be a Boolean function. Then an extension of A_m over $\mathbb{Z}$ is a polynomial $\widehat{A}_m : \mathbb{Z}^m \to \mathbb{Z}$ such that $\widehat{A}_{m(x)} = A_m(x)$ whenever $x \in \{0,1\}^m$. Also, given an oracle $A = (A_m)$, an extension $\widehat{A}$ of A is a collection of polynomials $\widehat{A}_m : \mathbb{Z}^m \to \mathbb{Z}$ one for each $m \in \mathbb{Z}^+$, such that:*

1. *$\widehat{A}_m$ is an extension of A_m for all m,*
2. *there exists a constant c such that $mdeg(\widehat{A}_m) \leq c$ for all m, and*
3. *there exists a polynomial p such that $size(\widehat{A}_m(x)) \leq p(m + size(x))$ for all $m \in \mathbb{Z}^m$.*

Then given a complexity class $\mathcal{C}$, by $\mathcal{C}^{\widehat{A}}$ ($\mathcal{C}^{\widehat{A}[poly]}$) means the class of languages decidable by a $\mathcal{C}$ machine that, on inputs of length n, can query $\widehat{A}_m$ for any $m (m = O(poly(n)))$ respectively.

13.3.5 k-Algebrization

Relativization is transitive over inclusion, i.e., if $\mathcal{C} \subseteq \mathcal{D}$ and $\mathcal{E} \subseteq \mathcal{E}$ relativize, then $\mathcal{C} \subseteq \mathcal{E}$ also relativizes. But the authors doubts whether algebrization is transitive. However the transitivity over extension holds. Given an oracle A, we can create a double-extension $\widetilde{\widetilde{A}}$ of A by:

1. Taking a low degree extension $\widetilde{A}$ of A.
2. Letting f be a Boolean oracle such that $f(x,i)$ is the i^{th} bit in the binary representation of $\widetilde{A}(x)$.
3. Taking a low-degree extension $\widetilde{\widetilde{A}}$ of f.

We can further iterate this. Then it follows as:

Proposition 210 *For all complexity classes $\mathcal{C}, \mathcal{D}, \mathcal{E}$, if $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ and $\mathcal{D}^A \subseteq \mathcal{E}^{\widetilde{A}}$ for all A, $\widetilde{A}$, then $\mathcal{C}^A \subseteq \mathcal{E}^{\widetilde{\widetilde{A}}}$ for all A, $\widetilde{\widetilde{A}}$.*

Theorem 211 *Any proof of $P \neq NP$ will require techniques that are not merely non-algebrizing, but non-k-algebrizing for every constant k.*

Proof: The proof is similar to the proof of Theorem 207. All we need to see is $A^{\widetilde{\widetilde{A}}}$ is also $PSPACE$-complete which is shown in Babai et al. [9]. ■

13.3.6 Open problems

Arithmetization is one of the most powerful ideas in complexity theory. Yet it is fundamentally unable to resolve many of the barrier problems in the field, such as P versus NP, derandomization of RP, and circuit lower bounds for $NEXP$. Here the authors stated some directions for the future work:

1. Find non-algebrizing techniques.
2. Find ways to exploit the structure of polynomials produced by arithmetization.
3. Find open problems that can still be solved with algebrizing techniques.
4. Prove algebraic oracle separations.
5. Understand algebrization better.

Chapter 14

Hardness Amplification

Lecture(s): 43–45
Date(s): 4/19, 4/22
Lecturer: Meera Sitharam
Scribe: Joel Willoughby

14.1 Introduction

In these notes, we present some of the classical results that relate existence of "hard" problems to pseudo-randomness. Specifically, a long sought after result in complexity theory is a complete relation between the P and NP gap and the P and BPP gap. To put it simply:

Conjecture 212 (BOLD Conjecture) *$P \neq NP$ iff $P = BPP$.*

Here we mainly concern ourselves with work towards the $\rightarrow$ direction (Conjecture 213) There has been some work in the converse direction (namely that circuit size lower bounds are necessary for derandomization), which we briefly mention at the end.

Conjecture 213 *$P \neq NP \rightarrow P = BPP$.*

In general, the idea is to take the existence of hard problems for P and turn them into pseudo-random generators that fool sets in P, i.e., they look random to sets in P, and can be used in lieu of randomness in BPP algorithms, thus derandomizing them. See Zihua's notes for more information on how this exactly can be done. Unfortunately, as we shall discuss later, such a result is currently out of our grasp. Instead we will narrow our focus to what has been shown in this direction. The main result we will look at is a rather well-known result from Impagliazzo and Wigderson:

Definition 214 (Circuit Complexity) *The circuit complexity of a boolean function f, denoted $CC(f)$ is the size of the smallest circuit that can compute f.*

Theorem 215 ([30]) *If $\exists f \in \mathbf{E}$ such that $CC(f) \geq 2^{\phi n}$, $0 < \phi < 1$, then $BPP = P$.*

We will discuss an even simpler theorem which is nearly the same:

Theorem 216 *If* $\exists f \in \mathbf{E}$ *such that* $CC(f) \geq 2^{n^\phi}$, $0 < \phi < 1$, *then* $BPP = DTIME(2^{\text{polylog}(n)})$.

Most of the rest of the paper will be dedicated to outlining the proof of this result. To begin with, we need some definitions.

Definition 217 (Worst Case Hardness) *A function* $f : \{0,1\}^n \to \{-1,1\}$ *is worst-case hard for size* S *if* $\forall C : \{0,1\}^n \to \{-1,1\}$ *with* $size(C) < S$, $E_x[f(x)C(x)] < 1$.

Definition 218 (Hardness) *A function* $f : \{0,1\}^n \to \{-1,1\}$ *is* $1-\delta$ *hard for size* S *if* $\forall C : \{0,1\}^n \to \{-1,1\}$ *with* $size(C) < S$, $E_x[f(x)C(x)] < 1-\delta$.

The proof around Theorem 216 involves the following:

- We have a function f that is worst-case hard for subexponential circuits
- Using some theory from error correcting codes, we turn this f into an f^* such that f^* is $1 - 100/n$ hard for subexponential circuits
- Using a nice result called Yao's XOR Lemma, we turn f^* into an f' such that f' is 2^{-n^ϕ} hard for circuits of size 2^{n^ϕ}.
- Finally, we apply the Nisan Wigderson generator to get a pseudo-random function that fools all BPP algorithms. This generator runs in time $2^{\text{polylog}(n)}$.

So, to begin with, we need to take a worst-case hard function f for subexponential circuits and turn it into an f^* as above.

Theorem 219 (Worst Case to $1 - 100/n$ Hardness) *Given a worst case hard function* f *for subexponential circuits,* $\exists f^*$ *that is* $(1 - 100/n)$ *hard for the same circuits.*

Proof: [Rough Sketch] The proof of this theorem comes from the realm of error-correcting codes. We will not go into the detail here, but merely highlight the main points. Refer to Troy's notes for more on error-correcting codes.

Instead of thinking of f as a function, think of it as a vector indexed by its inputs. I.e. $f \in \{0,1\}^{2^n}$. Then, f being worst case hard for size s means that for every bit vector $C \in \{0,1\}^{2^n}$ such that C can be computed by a circuit of size at most s, the hamming distance of f and C is at least 1. The proof idea for this theorem is to think of all such f's now as some codewords that we are trying to encode.

It can be shown using multi-linear extension and a Hadamard code, we can encode the vectors f into longer vectors f^*. The important thing to note from this construction is that $|f^*|$ is polynomial in $|f|$, so this encoding does not blow up our function too much. It can be shown that for any 2 valid decodable words f^* and g^*, the hamming distance between them is at least $1/100n * |f^*|$.

Thus, if we assume that there is a circuit C of sufficiently small size that can match an f^* with at least $1 - 1/100n$ accuracy, we can then recover f simply by decoding f^*. Hence, it must be that no such C exists and so f^* is $1 - 1/100n$ hard.

One final thing to note here is that even if we were able to construct such a C as above, decoding usually takes time linear in $|f^*|$. But, linear in $|f^*|$ is exponential in n, so it seems this does not work. However, we can get around this barrier by noting that we do not need to decode the entire vector f^* at any one time, we simply need to find out what $f(x)$ is for a given input x. I.e., we need only *partially* decode f^* for any given input. There is a nice theory of *local decoding* procedures which solve this exact problem. Hence, we can avoid this hiccup. ■

Now, we can assume we have a function f that is $1 - 100/n$ hard for subexponential circuits. We use the following to turn it into an f' that can be used with the NW-generator:

Theorem 220 (Yao's XOR Lemma) *If a function $f : \{0,1\}^n \to \{-1,1\}$ is $1-\delta$ hard for size s, let $f' : \{0,1\}^{nk} \to \{-1,1\}$ such that $f'(x_1x_2...x_k) = f(x_1)f(x_2)\cdots f(x_k)$. Then for every $\epsilon > 2(1-\delta)^k$, f' is 2ϵ hard for size $s' = \frac{\epsilon^2 s}{100 \log 1/\delta\epsilon}$.*

Specifically, we set $\delta = 1/100n$, $\epsilon = 2^{-n^\phi/20}$, and $s = 2^{n^\phi}$. Then, we get an f' that is $2^{-n^\phi/20}$ hard for circuits of size $\approx 2^{n^\epsilon}$, which as we shall see will allow us to de-randomize BPP.

The proof of Theorem 220 is pretty straightforward once we have another result:

Theorem 221 (Hardcore Set Lemma) *Suppose f is $(1-\delta)$ hard for size s, given $0 < \epsilon < 1$, there is a distribution H with density $\geq \delta$ such that f is ϵ hard on H for size $s' = \frac{\epsilon^2 s}{100 \log 1/\delta\epsilon}$.*

By "hard on H", we mean that $E_{x \leftarrow H}[f(x)C(x)] < 1 - \delta$. This theorem essentially implies that every hard function has a smallish set of inputs, a *hardcore set*, for which the function is extremely hard. Note that we can make ϵ as small as we'd like. Those inputs not drawn from H need not be hard at all, i.e., many circuits of size $< s$ may be able to compute them.

Proof: [Of Theorem 221] The proof of this theorem is generally done with von-Neumman's Min-Max theorem which deals with so-called *2 player symmetric zero-sum games*. Briefly, we can think of a zero sum game in the following manner. The game is played by two opponents p and q. The game is represented as an $N \times N$ matrix A with positive and negative entries. p, on their turn, chooses a row of the matrix, then q chooses a column. p seeks to maximize the entry chosen, q chooses to minimize it (think p wins iff $A_{p,q} > 0$). We assume that p and q have a distribution that they pull from for their moves, represented by probability vectors $\hat{p} = \{p_1, ..., p_N\}$ such that $\sum p_i = 1, p_i \geq 0$. The Min-Max theorem states roughly that under these assumptions:

$$\underset{p \leftarrow \hat{p}}{\text{Min}} \underset{q \leftarrow \hat{q}}{\text{Max}} A = \underset{q \leftarrow \hat{q}}{\text{Max}} \underset{p \leftarrow \hat{p}}{\text{Min}} A \tag{14.1}$$

In other words, it doesn't matter if p goes first or q goes first. We can apply this game idea to our Hardcore set. The theorem basically says that $\exists H$ such that $\forall C$, when $x \leftarrow H$, $f(x)$ and $C(x)$ differ with probability at least δ. The Min-Max theorem tells us that we can instead show $\forall C$ such that $\exists H$, when $x \leftarrow H$, $f(x)$ and $C(x)$ differ with probability at least δ. In other words, we don't need to find one single H to prove the theorem, we need

only find an H for each C, which seems more manageable. There are a few differences. One is that there are infinitely many possible C's. It can be shown that Min-Max works for this case easily enough. The other problem is that the Min-Max theorem applies to *distributions* of strategies. This is alright in the case of H, as a distribution of distributions makes sense.

However, the notion of a distribution of *circuits* is not obvious. What is suggested is that given a distribution of circuits, we can construct a representative circuit by taking the majority of enough circuits from the distribution and build a hard set H from the places where the majority differs from f. We do this in such a manner that the resulting circuit has size at most s, meaning we have a disagreement with f on at least δ fraction of inputs. Then, we can use a Chernoff bound to show that f is hard on that H for the entire distribution of circuits. ■

With the Harcore Lemma in hand, we can move on to the proof of Yao's XOR Lemma:
Proof: [Of Theorem 220] Let H be the δ-density distribution guaranteed by the Harcore Lemma using $\epsilon/4$. From this, we know that $\forall C$ with $size(C) < s'$,

$$E_{x \leftarrow H}[f(x)C(x)] < \epsilon/2 \tag{14.2}$$

Now, we want to take this core hardness of H in f and turn it into a uniform hardness of f'. To do so, let us define the uniform distribution of $\{x_1...x_k\}$ as follows: let $U_n = \gamma H + (1-\gamma)Y$ with suitable choices of $0 \leq \gamma \leq 1$ and complementary distribution Y such that U_n is the uniform distribution. Then, let U_n^k we think of is the direct product of k U_n distributions. We can write as the convex combination $U_n^k = \alpha_0 Z_0 + \alpha_1 Z_1 + ... + \alpha_m Z_m$, where each Z_i is the concatenation or direct product of some number of H and Y distributions. Wlog, let Z_0 be made up entirely of Y, i.e. $Z_0 = Y^k$ and $\alpha_0 = (1-\delta)^k$.

Then, we want ideally that $E_{x' \leftarrow U_n^k}[f'(x)C'(x)] < \epsilon$, so assume the contrary:

$$E_{x \leftarrow U_n^k}[f'(x)C'(x)] = \sum_{i=0}^{m} \alpha_i E_{x \leftarrow Z_i}[f'(x)C'(x)] \geq \epsilon \tag{14.3}$$

Since $\epsilon > 2(1-\delta)^k$,

$$\sum_{i=0}^{m} \alpha_i E_{x \leftarrow Z_i}[f'(x)C'(x)] \geq \epsilon - (1-\delta)^k \geq \epsilon/2$$

Then for some $i > 0$, we have

$$E_{x \leftarrow Z_i}[f'(x)C'(x)] \geq \epsilon/2$$

From here we note that Z_i is a direct sum of k independent distributions of Y and H, at least one of which is H. Wlog, assume $Z_i^1 = H$ Then, it is not hard to see that there is a string $z \in \{0,1\}^{n(k-1)}$ such that we can "hard code" z in all of Z_i except for Z_i^1. Then, with that z hardcoded, the Z_i^1 part of f' still meets this bound. In other words, given $g : \{0,1\}^{nk} \to \{-1,1\}$: $g_z(x_1) = g(x_1 z_1 ... z_{k-1})$.

$$E_{x \leftarrow Z_i^1}[f'_z(x)C'_z(x)] = E_{x \leftarrow H}[f(x)C(x)] \geq \epsilon/2$$

This contradicts (14.2) above, so our assumption (14.3) was wrong, proving the lemma. ■

Now, we can move on to the final step in our construction which is an invocation of the Nisan Wigderson generator:

Theorem 222 (NW Generator [45]) *For every function s, $\ell < s(\ell) < 2^\ell$, if $\exists f \in \mathbf{E}$ such that f is $(s(\ell^c)^{-1})$ hard for circuits of size $s(\ell^c)$, then there is a pseudorandom generator $G : \ell \to s(\ell^c)$ that runs in time $2^{O(\ell)}$.*

This is the final piece of the puzzle. We note that from Yao's XOR Lemma, we can get a function f that is 2^{-n^ϕ} hard for circuits of size 2^{n^ϕ}, then as an application of the NW generator: there is a pseudorandom generator $G : \log^c n \to n$ that runs in time $2^{\log^c n}$. Hence, $BPP = DTIME(2^{\text{polylog}(n)})$.

14.2 Other Directions

There have also been some results for the converse direction of Conjecture 213, namely that circuit size lower bounds are required for derandomization, [29] and more recently [2].

There is a somewhat different approach to the 2 conjectures via a representative problem in BPP (there are no complete problems or universal sets for BPP since Cook's Theorem and usual diagonalization fail). This representative problem is the so-called *Blackbox derandomization* of Polynomial Identity Testing, for which Schwartz-Zippel gives a randomized polynomial time algorithm. This problem is shown to be equivalent to the problem of testing whether a given arithmetic circuit computes the permanent polynomial (see [1]), and is thus shown to be closely linked (in both directions) to proving arithmetic circuit size lower bounds, in fact for constant depth arithmetic circuits, since permanent has a natural constant depth arithmetic circuit via its self-reducibility.

There has been an explosion of work in this area recently with multiple useful results such as:

1. Size lower bounds for depth-3 arithmetic circuits gives size lower bounds for arbitrary depth circuits, called the Chasm at Depth 3.

2. The apparent closeness in arithmetic circuit complexity of permanent and determinantal polynomial identity testing.

3. As would be expected due to the permanent-determinant relationship, there are connections to the Geometric complexity theory approach of Mulmuley, where the link between derandomization of Noether's normalization lemma called the "GCT chasm" and blackbox derandomization of polynomial identity testing, and via the above discussion to constant depth arithmetic circuit size lower bounds.

Chapter 15

Unification of Pseudorandom Objects

Lecture(s): 46–48
Date(s): 4/26, 4/28
Lecturer: Troy Baker
Scribe: Troy Baker

15.1 The Framework

A diverse range of pseudorandom objects can all be expressed in a similar way. These include list-decodable codes, samplers, expander graphs, randomness extractors, hardness amplifiers, and pseudorandom generators. Understanding how they are linked is useful in translating the results of one field into another. In this section, the framework underlying all of these objects will be established. These lectures are based on Ref. [57].

First, some convenient notation is introduced. The set $[X]$ is taken to be $\{1, \ldots, X\}$. The variable x is taken to be $\lceil \log X \rceil$, and it is often useful to think of $[X]$ as the set of bit strings of length x. The distribution U_S is the uniform distribution over set S and $x \xleftarrow{\mathrm{R}} S$ means s is sampled uniformly from set S.

The unifying framework was motivated by list-decodable codes. The following definition adopts this terminology, hence the name LIST.

Definition 223 (LIST) *Given function* $\Gamma : [N] \times [D] \to [M]$*, a set* $T \subseteq [M]$*, and an* agreement parameter $\varepsilon \in [0, 1]$*:*

$$LIST_\Gamma(T, \varepsilon) = \left\{x \in [N] : \Pr[\Gamma(x, U_{[D]}) \in T] > \varepsilon\right\}$$

$$LIST_\Gamma(T, 1) = \left\{x \in [N] : \Pr[\Gamma(x, U_{[D]}) \in T] = 1\right\}$$

Let $\mathcal{C}$ be a class of subsets of $[M]$. Defining a pseudorandom object in the framework will be a matter of choosing an appropriate Γ, $\mathcal{C}$, ε, and K such that the following condition successfully characterizes the object: "For every subset $T \in \mathcal{C}$, we have $|\mathrm{LIST}_\Gamma(T, \varepsilon)| \leq K$." The parameters ε and $K \in [0, N]$ are "quality" parameters of the object, and in the framework a "better" object would minimize both.

15.2 List-Decodable Codes

Given a *message word* of length n, some function converts this into a *codeword* of length d. This function, called an *encoding*, uniquely defines a *code*. The codeword is sent over some noisy channel and becomes a *corrupted word* also of length d, differing from the codeword on some fraction of the bits. On a given channel, an *error-correcting code* is a code from which a corrupted word can be used to determine a unique codeword and, through an inversion of the encoding, a unique message, too. If the corruption is too great on the channel, instead the goal is a *list-decodable code* from which a corrupted word can be used to determine a list (of length at most K) of possible codewords; K is known as the *list-size*.

Formally, the encoding is a function $\text{Enc} : \{0,1\}^n \to [q]^D$, for some alphabet $[q]$; often the codeword alphabet will be binary as well, but to be general $[q]$ is used. Therefore, the possible corrupted words are string in $[q]^D$. List-decodability is defined below. A corrupted word is expected to agree with most codewords on $1/q$ of the positions; therefore the definition demands some additive constant error ε, otherwise there could be no meaningful decoding.

Definition 224 *A code is* (ε, K) list-decodable *if, for every corrupted word* $r \in [q]^D$*, there are at most* K *messages* $x \in \{0,1\}^n$ *such that* $Enc(x)$ *and* r *agree in greater than* $1/q + \varepsilon$ *of the positions.*

To characterize some function Enc as (ε, K) list-decodable in the framework, an appropriate Γ, $\mathcal{C}$, ε, and K must be chosen, as per Definition 223. Let $\Gamma : [N] \times [D] \to [D] \times [q]$ be defined by

$$\Gamma(x, y) = (y, \text{Enc}(x)_y), \tag{15.1}$$

where $\text{Enc}(x)_y$ is the y'th symbol in the codeword $\text{Enc}(x)$. Given some word $r \in [q]^D$, let $T_r = \{(y, r_y) : y \in [D]\}$; the class $\mathcal{C}$ is any T of such form, for all r. The framework parameter ε is $1/q + \varepsilon$. The framework parameter K is K. Using these parameters, the following proposition is made in the form of the condition expressed in Section 15.1.

Proposition 225 *Let* $Enc : \{0,1\}^n \to [q]^D$ *correspond to* $\Gamma : [N] \times [D] \to [D] \times [q]$ *as defined in Eq. 15.1. Then Enc is* (ε, K) *list-decodable if and only if:*

$$\forall r \in [q]^D \qquad |LIST_\Gamma(T_r, 1/q + \varepsilon)| \leq K,$$

where $T_r = \{(y, r_y) : y \in [D]\}$.

Proof: Consider Definition 223:

$$\begin{aligned} x \in \text{LIST}_\Gamma(T_r, 1/q + \varepsilon) &\iff \Pr[\Gamma(x, U_{[D]}) \in T_r] > 1/q + \varepsilon \\ &\iff \Pr_{y \xleftarrow{R} [D]}[(y, \text{Enc}(x)_y) \in T_r] > 1/q + \varepsilon \\ &\iff \Pr_{y \xleftarrow{R} [D]}[\text{Enc}(x)_y = r_y] > 1/q + \varepsilon \end{aligned}$$

This shows that for every r and x, x is in the list iff $\text{Enc}(x)$ agrees with r in greater than $1/q + \varepsilon$ of the positions. Therefore, Enc is (ε, K) list-decodable iff list-size is K. ■

In this context, the meaning of LIST is exactly as expected; $\mathrm{LIST}_\Gamma(T_r, 1/q+\varepsilon)$ is the set of possible decodings of r, i.e. the set of words whose encoding matches r at more than $1/q+\varepsilon$ of the positions. The maximum size of $\mathrm{LIST}_\Gamma(T_r, 1/q+\varepsilon)$ over r is precisely the definition of list-size.

15.3 Averaging Samplers

Suppose you are given a boolean function $T : [M] \to \{0,1\}$ for which you wish to compute the average. The Chernoff Bound shows that we need $O(\log(1/\delta)/\varepsilon^2)$ independent random samples from $[M]$ to find $\mathrm{E}[T] \pm \varepsilon$ with probability $\geq 1-\delta$. Instead, we use a *sampler*, which takes an input seed drawn at random and selects *samples* from $[M]$ and uses them to approximate the average value of T. Alternatively, a sampler can be thought of as producing samples such that the correct fraction of samples land in any subset of the universe with high probability. A formal definition is provided below.

Definition 226 *Let sampler Smp for domain $[M]$ take a seed $x \in [N]$ and output a D-tuple of elements in $[M]$. Smp is a (δ,ε)* averaging sampler *if, for every function $T : [M] \to \{0,1\}$, we have*

$$\Pr_{x \xleftarrow{R} [N]} [\sigma(T,x) \leq \mu(T) + \varepsilon] \geq 1-\delta,$$

where $\mu(T) = E[T(U_{[M]})]$ and $\sigma(T,x) = \frac{1}{D}\sum_{i=1}^{D} T(Smp(x)_i)$.

I.e. σ is the approximated average of T using the sampler and μ is the expected value of T. Although it only bounds from above, it can easily be adjusted to bound from below by applying the definition to the complement of T, increasing the error δ by a factor of 2.

Let Smp be such a function $\mathrm{Smp} : [N] \to ([M] \times [M] \times \cdots \times [M])$, where there are D such outputs from $[M]$. For the characterization of (δ,ε) averaging samplers in the framework, let $\Gamma : [N] \times [D] \to [M]$ be defined by:

$$\Gamma(x,y) = \mathrm{Smp}(x)_y,$$

where $\mathrm{Smp}(x)_y$ is the y'th sample output by the sampler. T will be thought of as the characteristic function of a subset of $[M]$; thus, the expected value is $\mu(T) = |T|/M$. The definitions of $\mathcal{C}$, ε, and K are clear in the following proposition.

Proposition 227 *Smp is (δ,ε) averaging sampler if and only if:*

$$\forall T \subseteq [M] \qquad |LIST_\Gamma(T, \mu(T)+\varepsilon)| \leq K,$$

where $\mu(T) = |T|/M$ and $K = \delta N$.

Proof:

$$\begin{aligned} x \in \mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon) &\iff \Pr[\Gamma(x, U_{[D]}) \in T] > \mu(T)+\varepsilon \\ &\iff \Pr_{y \xleftarrow{R} [D]} [\mathrm{Smp}(x)_y \in T] > \mu(T)+\varepsilon \\ &\iff \sigma(T,x) > \mu(T)+\varepsilon \end{aligned}$$

In this context, $\text{LIST}_\Gamma(T, \mu(T)+\varepsilon)$ is the set of "bad" seeds whose average output on T is greater than $\mu(T)+\varepsilon$. The probability of a bad seed is at most δ iff $|\text{LIST}_\Gamma(T, \mu(T)+\varepsilon)| \leq \delta N$. ■

15.4 Expander Graphs

Expander graphs are multigraphs with the seemingly contradictory properties of sparsity and high connectivity. In the general setting considered here, they are bipartite graphs with a regular left-sets of vertices and high *vertex expansion*, i.e. every subset of not-too-many left vertices has many neighbors in the right-set.

In this section, multigraph G is taken to be a left-regular bipartite multigraph with left-set $[N]$, right-set $[M]$, and left degree D. A formal definition of expanders is presented below.

Definition 228 *Graph G is an $(= K, A)$* expander *if every subset S of the left-set with size at least K has at least AK neighbors in the right-set. G is a (K, A)* expander *if it is a $(= K', A)$ expander for every $K' \leq K$.*

For the characterization of (K, A) expander in the framework, $\Gamma : [N] \times [D] \to [M]$ is defined by:

$$\Gamma(x, y) = \text{Adj}(x)_y, \tag{15.2}$$

where $\text{Adj}(x)_y$ is the y'th vertex in the adjacency list of x. The definitions of $\mathcal{C}$, ε, and K are clear in the following proposition.

Proposition 229 *Let G correspond to $\Gamma : [N] \times [D] \to [M]$ as defined by Equation 15.2. G is an $(= K, A)$ expander if and only if:*

$$\forall T \subseteq [M] \text{ s.t. } |T| < AK \qquad |LIST_\Gamma(T, 1)| < K.$$

Thus, G is a (K, A) expander if and only if:

$$\forall T \subseteq [M] \text{ s.t. } |T| < AK \qquad |LIST_\Gamma(T, 1)| < |T|/A.$$

Proof:

$$\begin{aligned} x \in \text{LIST}_\Gamma(T, 1) &\iff \Pr[\Gamma(x, U_{[D]}) \in T] = 1 \\ &\iff \Pr_{y \xleftarrow{R} [D]}[\text{Adj}(x)_y \in T] = 1 \\ &\iff \text{Adj}(x) \subseteq T \end{aligned}$$

■

In this context, T is a subset of the right-set, i.e. $T \subseteq [M]$. $\text{LIST}_\Gamma(T, 1)$ is the subset of left vertices whose neighbors are a subset of T.

15.5 Randomness Extractors

Natural sources, such as temperature, quartz vibrations, weather, etc., can be used to generate long strings of biased and correlated bits with some amount of inherent randomness. A *randomness extractor* aims to generate a string of near-uniformly random bits using such strings, using a small *seed* of truly random bits. The following definitions characterize the randomness of arbitrary bit strings.

Definition 230 *The* min-entropy *of a random variable X is*

$$H_\infty(X) = \min_{x \in Supp(X)} \log(1/\Pr[X = x]).$$

Definition 231 *X is a k-*source *if $H_\infty(X) \geq k$. Equivalently, X is a k-source if $\Pr[X = x] \leq 2^{-k}$ for all x.*

A k-source can be thought of as having "k bits of randomness." As one would expect, the above definition implies that an RV uniformly distributed over $[K] = [2^k]$ is a k-source.

The following definitions provide a way to compare the distributions of two RVs. The ultimate goal of randomness extraction is to find a bit string that is not-too-far from a uniform distribution of bits.

Definition 232 *The* statistical difference *between RVs X and Y (in universe $[M]$) is:*

$$\begin{aligned}\Delta(X, Y) &= \max_{T \subseteq [M]} |\Pr[X \in T] - \Pr[Y \in T]| \\ &= \max_{T \subseteq [M]} \Pr[X \in T] - \Pr[Y \in T]\end{aligned}$$

Definition 233 *X and Y are ε-*close *if $\Delta(X, Y) \leq \varepsilon$. Otherwise, they are ε-*far*.*

The absolute value in the definition of entropy is irrelevant because the definition maximizes over all $T \subseteq M$, and the value for T is the negation of $\bar{T}$.

Suppose there is RV X, which is an n-source with universe $[N]$. Ideally, an extractor would be a function of the form $\text{Ext} : [N] \to [M]$, s.t. $\text{Ext}(X)$ is ε-close to $U_{[M]}$. However, this is impossible; consider the case of m being 0 or 1. Without loss of generality, let $m = 1$, then a uniform distribution on the set of preimages $\text{Ext}^{-1}(1)$ is an $(n-1)$-source, but the output of the extractor is constant and thus extracted no randomness. Instead, extractors are provided with a d-bit *seed*, being defined as a function $\text{Ext} : [N] \times [D] \to [M]$.

Despite needing a truly random source for the seed, extractors are quite useful. In practices, the seed can be logarithmic in the length of the output string. This means the need for a seed in polynomial settings can be eliminated entirely by simply enumerating all D possibilities, so long as the extractor algorithm is at most exponential.

Definition 234 *$Ext : [N] \times [D] \to [M]$ is a (k, ε)* extractor *if, for every k-source X taking values in $[N]$, $Ext(X, U_{[D]})$ is ε-close to $U_{[M]}$.*

Ideally, given some k-source and some seed of d bits, an extractor captures all k bits of randomness from the source and all d bits of randomness from the seed, such that the uniform output is of length $m \approx k + d$. In practice, most algorithms aim for $m \approx k$.

Let $\text{Ext} : [N] \times [D] \to [M]$. For the characterization of (k, ε) extractors in the framework, $\Gamma : [N] \times [D] \to [M]$ is defined by:

$$\Gamma(x, y) = \text{Ext}(x, y) \tag{15.3}$$

The definitions of $\mathcal{C}$, ε, and K are clear in the following proposition. Unlike the previous sections, the characterization is a little more hairy, being slightly different in each direction.

Proposition 235 *1. If Ext is a (k, ε) extractor, then:*

$$\forall T \subseteq [M] \qquad |LIST_\Gamma(T, \mu(T) + \varepsilon)| < K, \tag{15.4}$$

where $\mu(T) = |T|/M$.

2. Conversely, if Eq. 15.4 holds, then Ext is a $(k + \log(1/\varepsilon), 2\varepsilon)$ extractor.

Proof: Let Y be an RV from a uniform distribution over $[M]$. Note that $\Pr[Y \in T] = |T|/M = \mu(T)$.

$$\begin{aligned} x \in \text{LIST}_\Gamma(T, \mu(T) + \varepsilon) &\iff \Pr[\Gamma(x, U_{[D]}) \in T] > \mu(T) + \varepsilon \\ &\iff \Pr[\text{Ext}(x, U_{[D]}) \in T] > \mu(T) + \varepsilon \\ &\iff \Pr[\text{Ext}(x, U_{[D]}) \in T] - \mu(T) > \varepsilon \\ &\iff \Pr[\text{Ext}(x, U_{[D]}) \in T] - \Pr[U_{[M]} \in T] > \varepsilon \end{aligned}$$

For part one, we show the contrapositive. Suppose $|\text{LIST}_\Gamma(T, \mu(T) + \varepsilon)| \geq 2^k$. Let RV X be drawn from a uniform distribution over $\text{LIST}_\Gamma(T, \mu(T)+\varepsilon)$. Note that X is a k-source.

$$\begin{aligned} x \in \text{LIST}_\Gamma(T, \mu(T) + \varepsilon) &\iff \Pr[\text{Ext}(X, U_{[D]}) \in T] - \Pr[U_{[M]} \in T] > \varepsilon \\ &\implies \Delta(\text{Ext}(X, U_{[D]}), U_{[M]}) > \varepsilon \\ &\implies \text{Ext}(X, U_{[D]}) \text{ is } \varepsilon\text{-far from } U_{[M]} \\ &\implies \text{Ext is not a (k,}\varepsilon\text{) extractor} \end{aligned}$$

For part two, we prove directly. Suppose, for all T, $|\text{LIST}_\Gamma(T, \mu(T) + \varepsilon)| < 2^k$. Let RV X be any $(k + \log(1/\varepsilon))$-source taking values from $[N]$.

By definition of k-source:

$$\Pr[X] \leq 2^{-(k+\log(1/\varepsilon))} = \varepsilon 2^{-k}$$

and therefore

$$\begin{aligned} \Pr[X \in \text{LIST}_\Gamma(T, \mu(T) + \varepsilon)] &\leq |\text{LIST}_\Gamma(T, \mu(T) + \varepsilon)| \cdot \varepsilon 2^{-k} \\ &\leq 2^k \cdot \varepsilon 2^{-k} = \varepsilon \end{aligned}$$

Furthermore, we have shown that:

$$x \in \mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon) \iff \Pr[\mathrm{Ext}(x, U_{[D]}) \in T] > \mu(T)+\varepsilon$$
$$x \notin \mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon) \iff \Pr[\mathrm{Ext}(x, U_{[D]}) \in T] \leq \mu(T)+\varepsilon$$

which gives

$$\Pr[\mathrm{Ext}(X, U_{[D]}) \in T \mid X \notin \mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon)] \leq \mu(T)+\varepsilon$$

Putting the above together:

$$\begin{aligned}
&\Pr[\mathrm{Ext}(X, U_{[D]}) \in T] \\
&\leq \Pr[X \in \mathrm{LIST}_\Gamma(\cdot,\cdot)] + \Pr[\mathrm{Ext}(X, U_{[D]}) \in T \mid X \notin \mathrm{LIST}_\Gamma(\cdot,\cdot)] \\
&\leq \varepsilon + (\mu(T)+\varepsilon) \\
&= \mu(T) + 2\varepsilon
\end{aligned}$$

■

In this context, T is a subset of extracted strings, i.e. $T \subseteq [M]$. $\mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon)$ is the set of biased sources of length n for which the extractor, given a uniformly distributed seed, does not produce a uniform distribution over subset T.

15.6 Hardness Amplifiers

One of the most impressive results of this unification of pseudorandomness is the characterization of hardness amplifiers, which relates complexity-theoretic objects (intractability) and the information-theoretic objects discussed previously. *Hardness amplifiers* convert worst-case hard boolean functions into average-case hard functions.

Definition 236 *A function $f : [n] \to [q]$ is* (s, δ) hard *if, for every boolean circuit C of size s, we have:*

$$\Pr[C(U_{[N]}) \neq f(U_{[N]})] \geq \delta$$

Note that, f is not (s, δ) hard if there exists some C such that $\Pr[C(U_{[N]}) = f(U_{[N]})] > 1 - \delta$.

Hardness amplification aims to increase δ. A *worst-case hard* function has $\delta = 0$. The best amplification would be to $\delta = 1 - 1/q - \varepsilon$; a constant circuit can achieve $\delta = 1 - 1/q$.

More formally, given a function $f : [n] \to \{0,1\}$ that is $(s, 0)$ hard, a hardness amplifier converts f to function $f' : [n'] \to [q]$ that is $(s', 1 - 1/q - \varepsilon)$ hard (for constant q and small ε.) Usually, $s' < s$, $n' > n$, and the complexity of f increases.

Roughly, the process of finding an amplifier will have two steps. First, find a universal black box Amp such that given any oracle f, Amp^f solves f'. This review will not discuss any explicit construction of such a black box amplifier. Second, prove f' is average-case hard when f is worst-case hard by finding a reduction from r, any algorithm that computes f' well on average, into an algorithm computing f in the worst-case.

A formal characterization is below. An *advice string* is used to assist in the construction of a circuit for variable sized input. This allows for the discussion of algorithms despite the use of circuits in the previous definitions.

Definition 237 *Let $Amp^f : [D] \to [q]$ be an algorithm with oracle $f : [n] \to \{0,1\}$. Amp is a (t, k, ε)* black-box worst-case-to-average-case hardness amplifier *if there is an oracle algorithm Red (the* reduction*) running in time t s.t., for every function $r : [D] \to [q]$ s.t.*

$$\Pr[r(U_{[D]}) = Amp^f(U_{[D]})] > 1/q + \varepsilon,$$

there is an advice string $z \in 2^k$ s.t.

$$\forall i \in [n] \qquad Red^r(i, z) = f(i).$$

Proposition 238 *If Amp is a (t, k, ε) black-box hardness amplifier and f is $(s, 0)$ hard, then Amp^f is $(s/\tilde{O}(t), 1 - 1/q + \varepsilon)$ hard.*

I.e. $\mathrm{Amp}^f : [D] \to [q]$ is the $f' : [n'] \to [q]$ function of $(s', 1 - 1/q + \varepsilon)$ hardness we were looking for.

Let $\mathrm{Amp}^f : [D] \to [q]$ be defined for every oracle $f : [n] \to \{0,1\}$. For the characterization of (t, k, ε) black-box worst-case-to-average-case hardness amplifiers in the framework, $\Gamma : [N] \times [D] \to [D] \times [q]$ is defined by:

$$\Gamma(f, y) = (y, \mathrm{Amp}^f(y)). \tag{15.5}$$

Think of $[N]$ as containing all 2^n boolean function on $[n]$. The definitions of $\mathcal{C}$, ε, and K are clear in the following proposition.

Proposition 239 *Amp is an (∞, k, ε) black-box hardness amplifier if and only if*

$$\forall r : [D] \to [q] \qquad |LIST_\Gamma(T_r, 1/q + \varepsilon)| \leq 2^k,$$

where $T_r = \{(y, r(y)) : y \in [D]\}$.

Proof: In the forward direction, suppose Amp is an (∞, k, ε) black-box hardness amplifier, with reduction Red.

$$\begin{aligned} f \in \mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon) &\iff \Pr[\Gamma(f, U_{[D]}) \in T_r] > 1/q + \varepsilon \\ &\iff \Pr_{y \xleftarrow{R} [D]}[(y, \mathrm{Amp}^f(y)) \in T_r] > 1/q + \varepsilon \\ &\iff \Pr_{y \xleftarrow{R} [D]}[\mathrm{Amp}^f(y) = r(y)] > 1/q + \varepsilon \end{aligned}$$

The above means there exists $\mathrm{Red}^r(\cdot, z)$ for some $z \in 2^k$ that solves f in the worst case. There are at most 2^k choices for z, so at most 2^k such functions f. Therefore, $|\mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon)| \leq 2^k$.

In the other direction, suppose, for every function r, there are at most 2^k functions f in $\mathrm{LIST}_\Gamma(T_r, 1/q+\varepsilon)$. Sort them lexicographically (repeating the last if necessary) as $f_{r,1}, \ldots, f_{r,2^k}$. We define a reduction $\mathrm{Red}^r(i,z) = f_{r,z}(i)$. Since the time complexity is unbounded, this can be done via brute force. Therefore, Amp is an (∞, k, ε) black-box hardness amplifier. ■

To understand the meaning of $\mathrm{LIST}_\Gamma(T_r, 1/q+\varepsilon)$ in this context, consider the following:

$$f \in \mathrm{LIST}_\Gamma(T_r, 1/q+\varepsilon) \iff \Pr_{y \xleftarrow{R} [D]}[\mathrm{Amp}^f(y) = r(y)] > 1/q + \varepsilon$$
$$\implies \mathrm{Amp}^f(y) \text{ is } \textit{not } (s, 1-1/q-\varepsilon) \text{ hard.}$$

Notice that unbounded time complexity hardness amplifiers are *equivalent* to list-decodable codes. Bounding time complexity makes them stronger, but that will not be covered here.

15.7 Pseudorandom Generators

A *pseudorandom generator (PRG)* takes a short *seed* of truly random bits and generates a much longer string of bits that "looks random." PRGs are often used in derandomization and cryptography.

In the derandomization setting, "looking random" is generally from the perspective of the algorithm to be derandomized. If the algorithm is polytime, there should be no polytime algorithm can differentiate the given pseudorandom string from a truly random string. In this setting, it is desirable for to have d in $O(\log m)$, so every seed can be enumerated.

In the cryptography setting, "looking random" is more difficult because the algorithm that is attempting to determine if the string is random can have a much longer running time. This distinguishing algorithm is an *adversary*, and may run much longer than the generator. In this setting, a vanishing error parameter is important, that shrinks in some polynomial factor of the output length.

The definitions below present a way to measure the similarity of two distributions. The following is a computational analogue of statistical difference.

Definition 240 *RVs X and Y are* (s, ε) indistinguishable *if, for every boolean circuit T of size s, we have:*

$$\Pr[T(X) = 1] - \Pr[T(Y) = 1] \leq \varepsilon.$$

Definition 241 *A function $G : [D] \to [M]$ is an* (s, ε) pseudorandom generator *if $G(U_{[D]})$ is (s, ε) indistinguishable from $U_{[M]}$.*

A formal characterization is below. The following is quite similar to hardness amplification. As was found there, an *advice string* is used to allow for an algorithmic characterization despite the use of circuits in the previous definitions.

Definition 242 *Let $G^f : [D] \to [M]$ be an algorithm with oracle $f : [n] \to \{0,1\}$. G is a (t, k, ε)* black-box PRG construction *if there is an oracle algorithm Red (the* reduction*) running in time t s.t., for every boolean function $T : [M] \to \{0,1\}$ s.t.*

$$\Pr[T(G^f(U_{[D]})) = 1] - \Pr[T(U_{[M]}) = 1] > \varepsilon,$$

there is an advice string $z \in 2^k$ *s.t.*

$$\forall i \in [n] \qquad Red^f(i, z) = f(i).$$

Similar to black-box hardness amplifiers, a construction conforming to the definition suffices to create a PRG from an f of high complexity.

Proposition 243 *If* G *is a* (t, k, ε) *black-box PRG construction and* f *is* $(s, 0)$ *hard, then* G^f *is an* $(s/\tilde{O}(t), \varepsilon)$ *pseudorandom generator.*

Let $G^f : [D] \to [M]$ be defined for every oracle $f : [n] \to \{0, 1\}$. For the characterization of (t, k, ε) black-box worst-case-to-average-case hardness amplifiers in the framework, $\Gamma : [N] \times [D] \to [M]$ is defined by:

$$\Gamma(f, y) = (y, G^f(y)).$$

Think of $[N]$ as containing all 2^n boolean function on $[n]$. The definitions of $\mathcal{C}$, ε, and K are clear in the following proposition.

Proposition 244 G^f *is an* (∞, k, ε) *black-box PRG construction if and only if*

$$\forall T \subseteq [M] \qquad |LIST_\Gamma(T, \mu(T) + \varepsilon)| \leq 2^k,$$

where $\mu(T) = |T|/M$.

Proof: Similar to the proof of Proposition 239 ■

To understand the meaning of $\mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon)$ in this context, consider the following:

$$\begin{aligned}
f \in \mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon) &\iff \Pr[\Gamma(f, U_{[D]}) \in T] > \mu(T) + \varepsilon \\
&\iff \Pr[G^f(U_{[D]}) \in T] - \mu(T) > \varepsilon \\
&\iff \Pr[G^f(U_{[D]}) \in T] - \Pr[U_{[M]} \in T] > \varepsilon \\
&\iff \Pr[T(G^f(U_{[D]})) = 1] - \Pr[T(U_{[M]}) = 1] > \varepsilon \\
&\implies G^f \text{ is } \textit{not } (s, \varepsilon) \text{ indistinguishable from } U_{[M]}
\end{aligned}$$

Notice that unbounded time complexity PRGs are *equivalent* to averaging samplers and randomness extractors. Furthermore, realize that efficient reduction is equivalent to efficient "local decoding."

Bibliography

[1] Derandomization vs. circuit lower bounds. http://www.cs.sfu.ca/~kabanets/talks/Barriers.pdf. Slides: Accessed: April 29, 2016.

[2] Scott Aaronson and Dieter van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011.

[3] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. Technical report, MIT Theory of Computing Colloquium, 2007.

[4] Noga Alon and Ravi B Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[5] Noga Alon and Benny Sudakov. Bipartite subgraphs and the smallest eigenvalue. *Comb. Probab. Comput.*, 9(1):1–12, January 2000.

[6] S. Arora, B. Barak, and D. Steurer. Subexponential algorithms for unique games and related problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 563–572, Oct 2010.

[7] 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, May 1998.

[8] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, SFCS '90, pages 16–25 vol.1, Washington, DC, USA, 1990. IEEE Computer Society.

[9] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *computational complexity*, 1(1):3–40.

[10] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[11] Boaz Barak and David Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. *CoRR*, abs/1404.5236, 2014.

[12] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, November 1984.

[13] Ravi B Boppana and Michael Sipser. *The complexity of finite functions*, volume 99. 1989.

[14] Boaz Borak. Sum of squares upper bounds, lower bounds, and open questions. 2014.

[15] Magnus Bordewich. *The complexity of counting and randomised approximation.*

[16] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *computational complexity*, 15(2):94–114.

[17] Pranav Dandekar. Algebraic-geometric methods for complexity lower bounds. Master's thesis, University of Florida, 2004.

[18] Irit Dinur. The PCP theorem by gap amplification. In *Proc. 38th ACM Symp. on Theory of Computing*, pages 241–250, 2006.

[19] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), June 2007.

[20] R. Fagin. Generalized first-order spectra, and polynomial. time recognizable sets. *SIAM-AMS Proceedings*, 7:43–73, 1974.

[21] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for max cut. Technical report, Algorithms, 2000.

[22] Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, December 2002.

[23] Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science: 21st Conference Bangalore, India, December 13–15, 2001 Proceedings*, chapter Relations Between Communication Complexity, Linear Arrangements, and Computational Complexity, pages 171–182. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[24] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995.

[25] Oded Goldreich. *Studies in Complexity and Cryptography: Miscellanea on the Interplay between Randomness and Computation*, chapter Basic Facts about Expander Graphs, pages 451–464. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[26] Johan Håstad. Computational limitations of small-depth circuits. 1987.

[27] Johan Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 1–10, New York, NY, USA, 1997. ACM.

[28] Hamed Hatami, Avner Magen, and Evangelos Markakis. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007. Proceedings*, chapter Integrality Gaps of Semidefinite Programs for Vertex Cover and Relations to 1 Embeddability of Negative Type Metrics, pages 164–179. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[29] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672 – 694, 2002. Special Issue on Complexity 2001.

[30] Russell Impagliazzo and Avi Wigderson. P = bpp if e requires exponential circuits: Derandomizing the xor lemma. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 220–229, New York, NY, USA, 1997. ACM.

[31] Howard Karloff. How good is the goemans-williamson max cut algorithm? In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 427–434, New York, NY, USA, 1996. ACM.

[32] Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for max 3sat? In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pages 406–, Washington, DC, USA, 1997. IEEE Computer Society.

[33] Jonathan Katz. Notes on complexity theory, lecture 18, oct 2011.

[34] Jonathan Katz. Notes on complexity theory, lecture 19, nov 2011.

[35] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. ACM.

[36] Subhash Khot. *On the unique games conjecture*, pages 99–121. 2010.

[37] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, April 2007.

[38] Subhash Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. *CoRR*, abs/1305.4581, 2013.

[39] Richard E Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171, 1975.

[40] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11:796–817, 2001.

[41] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.

[42] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, October 1992.

[43] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70.

[44] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

[45] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149 – 167, 1994.

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

[47] A. Pablo Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320.

[48] Pablo A. Parrilo. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. Technical report, 2000.

[49] Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp. In *In Proc. 40 th ACM STOC*, pages 245–254, 2008.

[50] Alexander A. Razborov. *An Invitation to Mathematics: From Competitions to Research*, chapter Communication Complexity, pages 97–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[51] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24 – 35, 1997.

[52] Uwe Schöning. A uniform approach to obtain diagonal sets in complexity classes. *Theoretical Computer Science*, 18(1):95–103, 1982.

[53] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, October 1992.

[54] Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, October 1992.

[55] A. Shen. IP = PSPACE: Simplified proof. *J. ACM*, 39(4):878–880, October 1992.

[56] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 77–82, New York, NY, USA, 1987. ACM.

[57] S. Vadhan. The unified theory of pseudorandomness: Guest column. *SIGACT News*, 38(3):39–54, September 2007.

[58] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *— 26th Annual Symposium on Foundations of Computer Science*, pages 1–10. IEEE, 1985.