1

NP ⊆ PCP[log n, polylog n]

- MAX-k-PCS **gap problem**:
  - given:
    - variables $x_1, x_2, \ldots, x_n$ taking values from **field** F_q
    - $n = q^m$ for some integer m
    - k-ary constraints $C_1, C_2, \ldots, C_t$
  - assignment viewed as $f:(F_q)^m \to F_q$
  - YES: some degree d assignment satisfies all constraints
  - NO: no degree d assignment satisfies more than $(1-\epsilon)$ fraction of constraints

May 30, 2023 CS151 Lecture 17

2

NP ⊆ PCP[log n, polylog n]

Lemma: for every constant $1 > \varepsilon > 0$, the MAX-k-PCS gap problem with

t = poly(n) k-ary constraints with k = polylog(n)
field size q = polylog(n)
$n = q^m$ variables with m = O(log n / loglog n)
degree of assignments d = polylog(n)
gap ϵ

is **NP**-hard.

May 30, 2023 CS151 Lecture 17

3

NP ⊆ PCP[log n, polylog n]

- Proof of Lemma
  - reduce from 3-SAT
  - 3-CNF $\varphi(x_1, x_2, \ldots, x_n)$
  - can encode as $\psi:[n] \times [n] \times [n] \times \{0,1\}^3 \to \{0,1\}$
  - $\psi(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ iff φ contains clause $(x_{i_1}^{b_1} \vee x_{i_2}^{b_2} \vee x_{i_3}^{b_3})$
  - e.g. $(x_3 \vee \neg x_5 \vee x_2) \Rightarrow \psi(3,5,2,1,0,1) = 1$

May 30, 2023 CS151 Lecture 17

4

NP ⊆ PCP[log n, polylog n]

- pick $H \subseteq F_q$ with $\{0,1\} \subseteq H$, $|H|$ = polylog n
- pick m = O(log n/loglog n) so $|H|^m = n$
- identify $[n]$ with H^m
- $\psi:H^m \times H^m \times H^m \times H^3 \to \{0,1\}$ encodes φ
- assignment $a:H^m \to \{0,1\}$
- Key: a satisfies φ iff $\forall i_1,i_2,i_3,b_1,b_2,b_3$ $\psi(i_1,i_2,i_3,b_1,b_2,b_3) = 0$ or $a(i_1)=b_1$ or $a(i_2)=b_2$ or $a(i_3)=b_3$

May 30, 2023 CS151 Lecture 17

5

NP ⊆ PCP[log n, polylog n]

$\psi:H^m \times H^m \times H^m \times H^3 \to \{0,1\}$ encodes φ
a satisfies φ iff $\forall i_1,i_2,i_3,b_1,b_2,b_3$
$\psi(i_1,i_2,i_3,b_1,b_2,b_3) = 0$ or $a(i_1)=b_1$ or $a(i_2)=b_2$ or $a(i_3)=b_3$

- **extend** ψ to a function $\psi':(F_q)^{3m+3} \to F_q$ with degree at most $|H|$ in each variable
- can **extend** any assignment $a:H^m \to \{0,1\}$ to $a':(F_q)^m \to F_q$ with degree $|H|$ in each variable

May 30, 2023 CS151 Lecture 17

6

NP ⊆ PCP[log n, polylog n]

ψ':$(F_q)^{3m+3} \rightarrow F_q$ encodes φ

a':$(F_q)^m \rightarrow F_q$ s.a. iff $\forall(i_1,i_2,i_3,b_1,b_2,b_3) \in H^{3m+3}$

$\psi'(i_1,i_2,i_3,b_1,b_2,b_3) = 0$ or $a'(i_1)=b_1$ or $a'(i_2)=b_2$ or $a'(i_3)=b_3$

– define: $p_{a'}$:$(F_q)^{3m+3} \rightarrow F_q$ from a' as follows

$p_{a'}(i_1,i_2,i_3,b_1,b_2,b_3) =$

$\psi'(i_1,i_2,i_3,b_1,b_2,b_3)(a'(i_1) - b_1)(a'(i_2) - b_2)(a'(i_3) - b_3)$

– a' s.a. iff $\forall\ (i_1,i_2,i_3,b_1,b_2,b_3) \in H^{3m+3}$

$p_{a'}(i_1,i_2,i_3,b_1,b_2,b_3) = 0$

NP ⊆ PCP[log n, polylog n]

ψ':$(F_q)^{3m+3} \rightarrow F_q$ encodes φ

a':$(F_q)^m \rightarrow F_q$ s.a. iff $\forall(i_1,i_2,i_3,b_1,b_2,b_3) \in H^{3m+3}$

$p_{a'}(i_1,i_2,i_3,b_1,b_2,b_3) = 0$

– note: $\deg(p_{a'}) \leq 2(3m+3)|H|$

– start using Z as shorthand for $(i_1,i_2,i_3,b_1,b_2,b_3)$

– another way to write "a' s.a." is:

- exists p_0:$(F_q)^{3m+3} \rightarrow F_q$ of degree $\leq 2(3m+3)|H|$
- $p_0(Z) = p_{a'}(Z)$ $\forall Z \in (F_q)^{3m+3}$
- $p_0(Z) = 0$ $\forall Z \in H^{3m+3}$

NP ⊆ PCP[log n, polylog n]

– Focus on "$p_0(Z) = 0\ \forall Z \in H^{3m+3}$"

– given: p_0:$(F_q)^{3m+3} \rightarrow F_q$

– define: $p_1(x_1, x_2, x_3, \ldots, x_{3m+3}) =$

$\Sigma_{h_j \in H} p_0(h_j, x_2, x_3, \ldots, x_{3m+3})x_1^j$

– **Claim**:

$p_0(Z)=0\ \forall Z \in H^{3m+3} \Leftrightarrow p_1(Z)=0\ \forall Z \in F_q x H^{3m+3-1}$

– Proof (⇒) for each $x_2, x_3, \ldots, x_{3m+3} \in H^{3m+3-1}$, resulting univariate poly in x_1 has all 0 coeffs.

NP ⊆ PCP[log n, polylog n]

– Focus on "$p_0(Z) = 0\ \forall Z \in H^{3m+3}$"

– given: p_0:$(F_q)^{3m+3} \rightarrow F_q$

$\boxed{\deg(p_1) \leq \deg(p_0) + |H|}$

– define: $p_1(x_1, x_2, x_3, \ldots, x_{3m+3}) =$

$\Sigma_{h_j \in H} p_0(h_j, x_2, x_3, \ldots, x_{3m+3})x_1^j$

– **Claim**:

$p_0(Z)=0\ \forall Z \in H^{3m+3} \Leftrightarrow p_1(Z)=0\ \forall Z \in F_q x H^{3m+3-1}$

– Proof (⇐) for each $x_2, x_3, \ldots, x_{3m+3} \in H^{3m+3-1}$, univariate poly in x_1 is $\equiv 0 \Rightarrow$ has all 0 coeffs.

NP ⊆ PCP[log n, polylog n]

– given: p_1:$(F_q)^{3m+3} \rightarrow F_q$

$\boxed{\deg(p_2) \leq \deg(p_1) + |H|}$

– define: $p_2(x_1, x_2, x_3, \ldots, x_{3m+3}) =$

$\Sigma_{h_j \in H} p_1(x_1, h_j, x_3, x_4, \ldots, x_{3m+3})x_2^j$

– Claim:

$p_1(Z)=0\ \forall Z \in F_q \times H^{3m+3-1}$

$\Leftrightarrow$

$p_2(Z)=0\ \forall Z \in (F_q)^2 \times H^{3m+3-2}$

– Proof: same.

NP ∈ PCP[log n, polylog n]

– given: p_{i-1}:$(F_q)^{3m+3} \rightarrow F_q$

$\boxed{\deg(p_i) \leq \deg(p_{i-1}) + |H|}$

– define: $p_i(x_1, x_2, x_3, \ldots, x_{3m+3}) =$

$\Sigma_{h_j \in H} p_{i-1}(x_1, x_2, \ldots, x_{i-1}, h_j, x_{i+1}, x_{i+2}, \ldots, x_{3m+3})x_i^j$

– Claim:

$p_{i-1}(Z)=0\ \forall Z \in (F_q)^{i-1} \times H^{3m+3-(i-1)}$

$\Leftrightarrow$

$p_i(Z)=0\ \forall Z \in (F_q)^i \times H^{3m+3-i}$

– Proof: same.

NP $\in$ PCP[log n, polylog n]

– define degree $3m+3+2$ poly. $\delta_i: F_q \to F_q$ so that
  - $\delta_i(v) = 1$ if $v = i$
  - $\delta_i(v) = 0$ if $0 \le v \le 3m+3+1$ and $v \ne i$

– define $Q: F_q \times (F_q)^{3m+3} \to F_q$ by:

$$Q(v, Z) = \Sigma_{i=0\ldots3m+3}\delta_i(v)p_i(Z) + \delta_{3m+3+1}(v)a'(Z)$$

– note: degree of Q is at most

$$3(3m+3)|H| + 3m + 3 + 2 < 10m|H|$$

NP $\subseteq$ PCP[log n, polylog n]

- Recall: MAX-k-PCS **gap problem**:
  – given:
    - variables $x_1, x_2, \ldots, x_n$ taking values from **field** F_q
    - $n = q^m$ for some integer m
    - k-ary constraints $C_1, C_2, \ldots, C_t$

  – assignment viewed as $f:(F_q)^m \to F_q$

  – YES: some degree d assignment satisfies all constraints

  – NO: no degree d assignment satisfies more than $(1-\epsilon)$ fraction of constraints

NP $\subseteq$ PCP[log n, polylog n]

– Instance of MAX-k-PCS **gap problem**:
  - set $d = 10m|H|$
  - given assignment $Q: F_q \times (F_q)^{3m+3} \to F_q$
  - expect it to be formed in the way we have described from an assignment $a: H^m \to \{0,1\}$ to φ
  - note

    to access $a'(Z)$, evaluate $Q(3m+3+1, Z)$

    $p_{a'}(Z)$ formed from a' and ψ' (formed from φ)

    to access $p_i(Z)$, evaluate $Q(i, Z)$

NP $\subseteq$ PCP[log n, polylog n]

– Instance of MAX-k-PCS **gap problem**:
  - set $d = 10m|H|$
  - given assignment $Q: F_q \times (F_q)^{3m+3} \to F_q$
  - expect it to be formed in the way we have described from an assignment $a: H^m \to \{0,1\}$ to φ
  - **constraints**: $\forall Z \in (F_q)^{3m+3}$

$(C_{0,Z})$: $p_0(Z) = p_{a'}(Z)$

$0 < i \le 3m+2$ $(C_{i,Z})$: $p_i(z_1, z_2, \ldots, z_i, z_{i+1}, \ldots, z_{3m+3}) = \Sigma_{h_j \in H}\, p_{i-1}(z_1, z_2, \ldots, z_{i-1}, h_j, z_{i+1}, \ldots, z_k)z_i^j$

$(C_{3m+3,Z})$: $p_{3m+3}(Z) = 0$

NP $\subseteq$ PCP[log n, polylog n]

- given $Q: F_q \times (F_q)^{3m+3} \to F_q$ of degree $d = 10m|H|$
- **constraints**: $\forall Z \in (F_q)^{3m+3}$

Key: all low-degree polys

$(C_{0,Z})$: $p_0(Z) = p_{a'}(Z)$

$(C_{i,Z})$: $p_i(z_1, z_2, \ldots, z_i, z_{i+1}, \ldots, z_{3m+3}) = \Sigma_{h_j \in H}\, p_{i-1}(z_1, z_2, \ldots, z_{i-1}, h_j, z_{i+1}, \ldots, z_k)z_i^j$

$(C_{3m+3,Z})$: $p_{3m+3}(Z) = 0$

– **Schwartz-Zippel**: if any one of these sets of constraints is violated *at all* then at least a **(1 – 12m|H|/q)** fraction in the set are violated

NP $\subseteq$ PCP[log n, polylog n]

- Proof of Lemma (summary):
  – reducing 3-SAT to MAX-k-PCS **gap problem**

  – $\varphi(x_1, x_2, \ldots, x_n)$ instance of 3-SAT

  – set $m = O(\log n/\log\log n)$

  – $H \subseteq F_q$ such that $|H|^m = n$ ($|H| = \text{polylog } n$, $q \approx |H|^3$)

  – generate $|F_q|^{3m+3} = \text{poly}(n)$ constraints:

$$C_Z = \wedge_{i=0\ldots3m+3+1} C_{i,Z}$$

  – each refers to assignment poly Q and φ (via $p_{a'}$)

  – all polys degree $d = O(m|H|) = \text{polylog } n$

  – either all are satisfied or at most $d/q = o(1) \ll \varepsilon$

NP ⊆ PCP[log n, polylog n]

- O(log n) random bits to pick a constraint
- query assignment in O(polylog(n)) locations to determine if constraint is satisfied
  - completeness 1
  - soundness ($1-\epsilon$) if prover keeps promise to supply degree d polynomial
- prover can cheat by not supplying proof in expected form

NP ⊆ PCP[log n, polylog n]

- Low-degree testing:
  - want: randomized procedure that is given d, oracle access to $f:(F_q)^m \rightarrow F_q$
    - runs in poly(m, d) time
    - always accepts if $\deg(f) \leq d$
    - rejects with high probability if $\deg(f) > d$
  - too much to ask. Why?

NP ⊆ PCP[log n, polylog n]

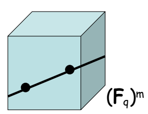
Definition: functions f, g are δ-close if

$\Pr_x[f(x) \neq g(x)] \leq \delta$

Lemma: ∃δ > 0 and a randomized procedure that is given d, oracle access to $f:(F_q)^m \rightarrow F_q$
- runs in poly(m, d) time
- uses $O(m \log |F_q|)$ random bits
- always accepts if $\deg(f) \leq d$
- rejects with high probability if f is not δ-close to any g with $\deg(g) \leq d$

NP ⊆ PCP[log n, polylog n]

- idea of proof:
  - restrict to random line L
  - check if it is low degree



  - always accepts if $\deg(f) \leq d$
  - other direction more complex

NP ⊆ PCP[log n, polylog n]

- can only force prover to supply function f that is close to a low-degree polynomial
- how to bridge the gap?
- recall low-degree polynomials form an error correcting code (Reed-Muller)
- view "close" function as corrupted codeword

NP ⊆ PCP[log n, polylog n]

- Self-correction:
  - want: randomized procedure that is given x, oracle access to $f:(F_q)^m \rightarrow (F_q)$ that is δ-close to a (unique) degree d polynomial g
    - runs in poly(m, d) time
    - uses $O(m \log |F_q|)$ random bits
    - with high probability outputs g(x)

NP ⊆ PCP[log n, polylog n]

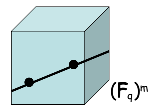
Lemma: ∃ a randomized procedure that is given x, oracle access to $f:(F_q)^m \rightarrow (F_q)$ that is δ-close to a (unique) degree d polynomial g

- runs in poly(m, d) time
- uses $O(m \log |F_q|)$ random bits
- outputs g(x) with high probability

25

NP ⊆ PCP[log n, polylog n]

- idea of proof:
  - restrict to random line L passing through x
  - query points along line
  - apply error correction



26

NP ⊆ PCP[log n, polylog n]

- Putting it all together:
  - given L ∈ NP and an instance x, verifier computes reduction to MAX-k-PCS **gap problem**
  - prover supplies proof in form

    $f:(F_q)^m \rightarrow (F_q)$

    (plus some other info used for low-degree testing)
  - verifier runs low-degree test
    - rejects if f not close to some low degree function g
  - verifier picks random constraint C_i; checks if sat. by g
    - uses self-correction to get values of g from f
  - accept if C_i satisfied; otherwise reject

27

New topic: relativization and natural proofs

28

Approaches to open problems

- Almost all major open problems we have seen entail proving **lower bounds**
  - **P ≠ NP**
  - **L ≠ P**
  - **P ≠ PSPACE**
  - **NC** proper
  - **BPP ≠ EXP**
  - **PH** proper
  - **EXP ⊊ P/poly**
  - **P = BPP ***
  - **NP = AM ***

• we know circuit lower bounds imply derandomization

• more difficult (and recent): derandomization implies circuit lower bounds!

29

Approaches to open problems

- two natural approaches
  - simulation + diagonalization (uniform)
  - circuit lower bounds (non-uniform)
- no success for either approach as applied to date

Why?

30

Approaches to open problems

in a precise, formal sense these approaches are **too powerful !**

- if they could be used to resolve major open problems, a side effect would be:
  - proving something that is false, or
  - proving something that is believed to be false

May 30, 2023 CS151 Lecture 17

31

Relativization

- Many proofs and techniques we have seen **relativize**:
  - they hold after replacing all TMs with oracle TMs that have access to an oracle A
  - e.g. $\mathbf{L}^A \subseteq \mathbf{P}^A$ for all oracles A
  - e.g. $\mathbf{P}^A \neq \mathbf{EXP}^A$ for all oracles A

May 30, 2023 CS151 Lecture 17

32

Relativization

- Idea: design an oracle A relative to which some statement is *false*
  - implies there can be no **relativizing proof** of that statement
  - e.g. design A for which $\mathbf{P}^A = \mathbf{NP}^A$
- Better: also design an oracle B relative to which statement is *true*
  - e.g. also design B for which $\mathbf{P}^B \neq \mathbf{NP}^B$
  - implies no **relativizing proof** can resolve truth of the statement either way !

May 30, 2023 CS151 Lecture 17

33

Relativization

- Oracles are known that falsify almost every major conjecture concerning complexity classes
  - for these conjectures, non-relativizing proofs are required
  - almost all known proofs in Complexity relativize (sometimes after some reformulation)
  - notable exceptions:
    - The PCP Theorem
    - IP = PSPACE
    - most circuit lower bounds (more on these later)

May 30, 2023 CS151 Lecture 17

34

Oracles for **P** vs. **NP**

- Goal:
  - oracle A for which $\mathbf{P}^A = \mathbf{NP}^A$
  - oracle B for which $\mathbf{P}^B \neq \mathbf{NP}^B$
- conclusion: resolving

  P vs. NP

  requires a non-relativizing proof

May 30, 2023 CS151 Lecture 17

35

Oracles for **P** vs. **NP**

- for $\mathbf{P}^A = \mathbf{NP}^A$ need A to be powerful
  - warning: intend to make **P** more powerful, but also make **NP** more powerful.
  - e.g. A = SAT doesn't work
  - however A = QSAT works:

    $\mathbf{PSPACE} \subseteq \mathbf{P}^{QSAT} \subseteq \mathbf{NP}^{QSAT} \subseteq \mathbf{NPSPACE}$

    and we know $\mathbf{NPSPACE} = \mathbf{PSPACE}$

May 30, 2023 CS151 Lecture 17

36

Oracles for **P** vs. **NP**

Theorem: there exists an oracle B for which $P^B \neq NP^B$.

- Proof:
  - define

    $L = \{1^k : \exists\, x \in B \text{ s.t. } |x| = k\}$
  - we will show $L \in NP^B - P^B$.
  - easy: $L \in NP^B$ (no matter what B is)

May 30, 2023 CS151 Lecture 17

37

Oracles for **P** vs. **NP**

- design B by diagonalizing against all "P^B machines"
- $M_1, M_2, M_3, \ldots$ is an enumeration of deterministic OTMs
- each machine appears infinitely often
- B_i will be those strings of length $\leq i$ in B
- we build B_i after simulating machine M_i

May 30, 2023 CS151 Lecture 17

38

Oracles for **P** vs. **NP**

$L = \{1^k : \exists\, x \in B \text{ s.t. } |x| = k\}$

- Proof (continued):
  - maintain "exceptions" X that must not go in B
  - initially $X = \{\,\}$, $B_0 = \{\,\}$

  Stage i:
  - simulate $M_i(1^i)$ for $i^{\log i}$ steps
  - when M_i makes an oracle query q:
    - if $|q| < i$, answer using B_{i-1}
    - if $|q| \geq i$, answer "no"; add q to X
  - if simulated M_i accepts 1^i then $B_i = B_{i-1}$
  - if simulated M_i rejects 1^i, $B_i = B_{i-1} \cup \{x \in \{0,1\}^i : x \notin X\}$

May 30, 2023 CS151 Lecture 17

39

Oracles for **P** vs. **NP**

$L = \{1^k : \exists\, x \in B \text{ s.t. } |x| = k\}$

- Proof (continued):
  - if M_i accepts, we ensure no strings of length i in B
  - therefore $1^i \notin L$, and so M_i does not decide L
  - if M_i rejects, we ensure *some* string of length i in B
  - Why?

    $B_i = B_{i-1} \cup \{x \in \{0,1\}^i : x \notin X\}$

    and $|X|$ is at most $\Sigma_{j \leq i}\, j^{\log j} << 2^i$
  - therefore $1^i \in L$, and so M_i does not decide L
  - Conclude: $L \notin P^B$

May 30, 2023 CS151 Lecture 17

40

Circuit lower bounds

- Relativizing techniques are out…
- but most circuit lower bound techniques do not relativize
- exponential circuit lower bounds known for weak models:
  - e.g. constant-depth poly-size circuits
- But, utter failure (so far) for more general models. Why?

May 30, 2023 CS151 Lecture 17

41

Natural Proofs

- Razborov and Rudich defined the following "natural" format for circuit lower bounds:
  - identify property $\underline{\mathbf{P}}$ of functions $f:\{0,1\}^* \rightarrow \{0,1\}$
  - $\underline{\mathbf{P}} = \cup_n \underline{\mathbf{P}}_n$ is a natural property if:
    - (useful) $\forall n\ f_n \in \underline{\mathbf{P}}_n$ implies f does not have poly-size circuits [i.e. $f_n \in \underline{\mathbf{P}}_n$ implies ckt size $\geq s(n) >> poly(n)$]
    - (constructive) can decide "$f_n \in \underline{\mathbf{P}}_n$?" in poly time given the *truth table* of f_n
    - (large) at least $(½)^{O(n)}$ fraction of all 2^{2^n} functions on n bits are in $\underline{\mathbf{P}}_n$
  - show some function family $g = \{g_n\}$ is in $\underline{\mathbf{P}}_n$

May 30, 2023 CS151 Lecture 17

42

Natural Proofs

- *all known circuit lower bound techniques* are natural for a suitably parameterized version of the definition

Theorem (RR): if there is a 2^{n^ϵ}-OWF, then there is no natural property **P**.

- – factoring believed to be 2^{n^ϵ}-OWF
- – general version also rules out natural properties useful for proving many other separations, under similar cryptographic assumptions

43