Umans
Complexity Theory Lectures

Lecture 16: The PCP Theorem

PCP

- **PCP[r(n),q(n)]**: set of languages L with p.p.t. verifier V that has (r, q)-restricted access to a string "proof"
  - V tosses O(r(n)) coins
  - V accesses proof in O(q(n)) locations
  - (completeness) $x \in L \Rightarrow \exists$ proof such that
    Pr[V(x, proof) accepts] = 1
  - (soundness) $x \notin L \Rightarrow \forall$ proof*
    Pr[V(x, proof*) accepts] ≤ ½

PCP

- Two observations:
  - **PCP[1, poly n] = NP**
    proof?
  - **PCP[log n, 1]** $\subset$ **NP**
    proof?

The PCP Theorem (AS, ALMSS):
PCP[log n, 1] = NP.

The PCP Theorem

- Two major components:
  - **NP** $\subset$ **PCP[n^2, 1]** ("inner verifier")
    - we will prove from scratch (linearity test on PS)
  - **NP** $\subset$ **PCP[log n, polylog n]** ("outer verifier")
    - we will prove this from scratch, assuming low-degree test, and self-correction of low-degree polynomials

The inner verifier

Theorem: **NP ⊂ PCP[n^2, 1]**

Proof (first steps):

1. Quadratic Equations is NP-hard
2. PCP for QE:

   proof = *all quadratic functions* of a soln. x

   verification = check that a random linear combination of equations is satisfied by x

(if prover keeps promise to supply all quadratic fns of x)

Quadratic Equations

- quadratic equation over F_2:

$$\sum_{i<j} a_{i,j} X_i X_j + \sum_i b_i X_i + c = 0$$

- language QUADRATIC EQUATIONS (QE)

  = { systems of quadratic equations over F_2 that have a solution (assignment to the X variables) }

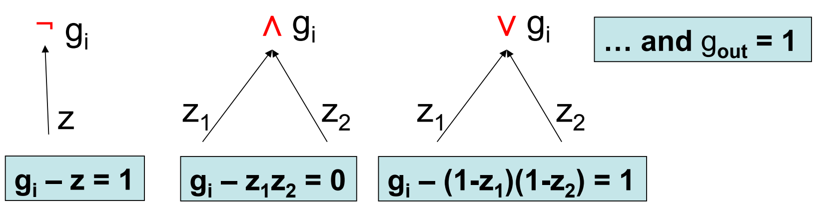
Quadratic Equations

Lemma: QE is NP-complete.

Proof: clearly in NP; reduce from CIRCUIT SAT

– circuit C an instance of CIRCUIT SAT

– QE variables = variables + gate variables



Quadratic Functions Code

- intended proof:
  - F the field with 2 elements
  - given $x \in F^n$, a solution to instance of QE
  - f_x: $F^n \rightarrow F_2$ all linear functions of x

$$f_x(a) = \sum_i a_i x_i$$

  - g_x: $F^{n \times n} \rightarrow F_2$ **includes** all quadratic fns of x

$$g_x(A) = \sum_{i,j} A[i,j] x_i x_j$$

  - KEY: can evaluate any quadratic function of x with a single evaluation of f_x and g_x

PCP for QE

If prover keeps promise to supply all quadratic fns of x, a solution of QE instance…

- Verifier's action:

–query a *random linear combination* R of the equations of the QE instance

–Completeness: obvious

–Soundness: x fails to satisfy some equation; imagine picking coeff. for this one last

Pr[x satisfies R] = 1/2

PCP for QE

$x \in F^n$ solves	$f_x(a) = \sum_i a_i x_i$	Had(x)
	$g_x(A) = \sum_{i,j} A[i,j]x_i x_j$	Had(x x)

To "enforce promise", verifier needs to perform:

- linearity test: verify f, g are (close to) linear
- self-correction: access the *linear* f', g' that are close to f, g
  [so f' = Had(u) and g' = Had(V)]
- consistency check: verify V = u u

PCP for QE

$x \in F^n$ solves	$f_x(a) = \sum_i a_i x_i$	Had(x)
	$g_x(A) = \sum_{i,j} A[i,j]x_i x_j$	Had(x x)

- Linearity test: given access to $h:F^m \rightarrow F$
  - pick random a,b; check if h(a) + h(b) = h(a+b); repeat O(1) times
  - do this for functions f and g supplied by prover

Theorem [BLR]: h linear⇒prob. success = 1; prob. success ≥ 1 – δ, ∃ linear h' s.t.

$$\Pr_a [h'(a) = h(a)] > 1 - O(\delta)$$

PCP for QE

x 2 F^n soln	$f_x(a) = \sum_i a_i x_i$	Had(x)
	$g_x(A) = \sum_{i,j} A[i,j]x_i x_j$	Had(x - x)

- Self-correction:
  - given access to $h:F^m \rightarrow F$ close to linear h'; i.e.,

$$\Pr_a [h'(a) = h(a)] \geq 1 - O(\delta)$$

  - to access h'(a), pick random b and compute h(b) + h(a+b)
  - with prob. at least 1 – 2O(δ) ≥ h(b) = h'(b) and h(a+b) = h'(a+b);
  - hence we compute h'(a)

PCP for QE

x 2 F^n solves	$f_x(a) = \sum_i a_i x_i$	Had(x)
	$g_x(A) = \sum_{i,j} A[i,j]x_i x_j$	Had(x x)

- Consistency check: given access to linear functions f' = Had(u) and g' = Had(V)
  - pick random a, b ∈ F^n and check that

    $f'(a)f'(b) = g'(ab^T)$
  - completeness: if V = u u

    $f'(a)f'(b) = (\sum_i a_i u_i)(\sum_i b_i u_i) = \sum_{i,j} a_i b_j V[i,j] = g'(ab^T)$

PCP for QE

x 2 F^n soln	$f_x(a) = \sum_i a_i x_i$	Had(x)
	$g_x(A) = \sum_{i,j} A[i,j]x_i x_j$	Had(x - x)

- Consistency check: given access to linear functions f' = Had(u) and g' = Had(V)
  - soundness: claim that if V ≠ u

    $\Pr[(\sum_i a_i u_i)(\sum_i b_i u_i) = \sum_{i,j}$

    $\Pr[(uu^T)b \neq Vb]$ ¸

    $\Pr[a^T(uu^T)b \neq a^T Vb]$ ¸ $\frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}$

9 i,j s.t. uu^T and V

9 i s.t. $(uu^T)b$ and Vb differ in entry i; pick a_i last

The outer verifier

Theorem: **NP ⊂ PCP[log n, polylog n]**

Proof (first steps):

- define: Polynomial Constraint Satisfaction (PCS) problem
- prove: PCS gap problem is **NP**-hard

NP ⊂ PCP[log n, polylog n]

- MAX-k-SAT
  - given: k-CNF φ
  - output: max. # of simultaneously satisfiable clauses
- generalization: MAX-k-CSP
  - given:
    - variables $x_1, x_2, \ldots, x_n$ taking values from set S
    - k-ary constraints $C_1, C_2, \ldots, C_t$
  - output: max. # of simultaneously satisfiable constraints

NP ⊂ PCP[log n, polylog n]

- algebraic version: MAX-k-PCS
  - 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 $f:(F_q)^m \rightarrow F_q$
  - output: max. # of constraints simultaneously satisfiable by an assignment that has deg. $\leq d$

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 $f:(F_q)^m \rightarrow F_q$
  - YES: some degree d assignment satisfies all constraints
  - NO: no degree d assignment satisfies more than $(1-\varepsilon)$ fraction of constraints

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 $m = O(\log n / \log\log n)$

degree of assignments d = polylog(n)

gap ε

is **NP**-hard.

NP ⊂ PCP[log n, polylog n]

t = poly(n) k-ary constraints with k = polylog(n)

field size q = polylog(n)

$n = q^m$ variables with $m = O(\log n / \log\log n)$

degree of assignments d = polylog(n)

- check: headed in right direction
  - O(log n) random bits to pick a constraint
  - query assignment in O(polylog(n)) locations to determine if it is satisfied
  - completeness 1; soundness $1-\varepsilon$

(if prover keeps promise to supply degree d polynomial)

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. $(x_3 \vee \neg x_5 \vee x_2) \Rightarrow \psi(3,5,2,1,0,1) = 1$

NP ⊂ PCP[log n, polylog n]

- pick $H \subset F_q$ with $\{0,1\} \subset H$, $|H| = \text{polylog } n$
- pick $m = O(\log n/\log\log n)$ 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 $a(i_3)=b_3$

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

NP ⊂ PCP[log n, polylog n]

$\psi' :(F_q)^{3m+3} \to F_q$ encodes φ

$a' :(F_q)^m \to F_q$ 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: $p_{a'} :(F_q)^{3m+3} \to F_q$ 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]

$\psi' : (F_q)^{3m+3} \to F_q$ encodes φ

$a' : (F_q)^m \to F_q$ 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: $\deg(p_{a'}) \le 2(3m+3)|H|$

– 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 $p_0 : (F_q)^{3m+3} \to F_q$ of degree $\le 2(3m+3)|H|$
- $p_0(Z) = p_{a'}(Z) \quad \forall Z \in (F_q)^{3m+3}$
- $p_0(Z) = 0 \quad \forall Z \in H^{3m+3}$

NP ⊂ PCP[log n, polylog n]

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

– given: $p_0 : (F_q)^{3m+3} \to F_q$

– 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 ($\Rightarrow$) 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: $p_0 : (F_q)^{3m+3} \to F_q$

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

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

$\deg(p_1) \le \deg(p_0) + |H|$

– **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 ($\Leftarrow$) 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: $p_1 : (F_q)^{3m+3} \to F_q$

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

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

$\deg(p_2) \le \deg(p_1) + |H|$

– 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: $p_{i-1}:(F_q)^{3m+3} \to F_q$

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

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

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

– 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 ⊂ 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 $0 \leq v \leq 3m+3+1$ and $v \neq i$

– 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

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