CS151
Complexity Theory

Lecture 16

May 20, 2021

The PCP Theorem

- Elements of proof:
  - arithmetization of 3-SAT
    - we will do this
  - low-degree test
    - we will state but not prove this
  - self-correction of low-degree polynomials
    - we will state but not prove this
  - proof composition
    - we will describe the idea

The PCP Theorem

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

Proof Composition (idea)

NP ⊆ PCP[log n, polylog n] ("outer verifier")
NP ⊆ PCP[n^2, 1] ("inner verifier")

- **composition** of verifiers:
  - reformulate "outer" so that it uses O(log n) random bits to make **1** query to each of **3** provers
  - replies r_1, r_2, r_3 have length polylog n
  - Key: accept/reject decision computable from r_1, r_2, r_3 by small circuit C

Proof Composition (idea)

NP ⊆ PCP[log n, polylog n] ("outer verifier")
NP ⊆ PCP[n^2, 1] ("inner verifier")

- **composition** of verifiers (continued):
  - final proof contains **proof** that $C(r_1, r_2, r_3) = 1$ for inner verifier's use
  - use inner verifier to verify that $C(r_1,r_2,r_3) = 1$
  - O(log n)+polylog n randomness
  - O(1) queries
  - tricky issue: consistency

Proof Composition (idea)

- NP ⊆ PCP[log n, 1] comes from
  - repeated composition
    - PCP[log n, polylog n] with PCP[log n, polylog n] yields PCP[log n, polyloglog n]
    - PCP[log n, polyloglog n] with PCP[n^2, 1] yields PCP[log n, 1]
- details omitted…

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)

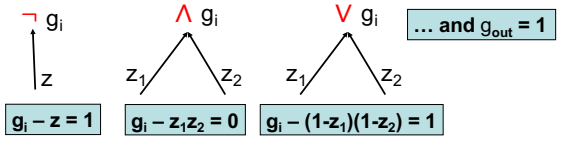
  = { systems of quadratic equations over F_2 that have a solution (assignment to the X variables) }

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: $F^n \to F_2$ all linear functions of x

    $f_x(a) = \sum_i a_i x_i$
  - g_x: $F^{n \times n} \to F_2$ **includes** all quadratic fns of x

    $g_x(A) = \sum_{i,j} A[i,j]x_ix_j$
  - 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

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

$x \in F^n$ 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)

- Linearity test: given access to $h:F^m \to F$
  - 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 ≥ $1 - \delta$ ⇒ ∃ linear h' s.t.

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

PCP for QE

$x \in F^n$ 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 $h:F^m \to F$ close to linear h'; i.e., $Pr_a [h'(a) = h(a)] \geq 1 - O(\delta)$
  - to access h'(a), pick random b; compute h(b) + h(a+b)
  - with prob. at least $1 - 2 \cdot O(\delta)$, h(b) = h'(b) and h(a+b) = h'(a+b); hence we compute h'(a)

PCP for QE

$x \in F^n$ 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)
  - pick random $a, b \in F^n$; 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

$x \in F^n$ 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 ≠ [illegible]

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

$$Pr[(uu^T)b \neq Vb]$$

$$Pr[a^T(uu^T)b \neq a^TVb] \geq ½ \cdot ½ = ¼$$

∃ i,j s.t. uu^T and V [illegible]

∃ 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. ≤ 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 $(1-\epsilon)$ 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 / loglog 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 / loglog 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 $1-\epsilon$

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

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