CS151
Complexity Theory

Lecture 16

May 25, 2004

Outline

- approximation algorithms
- Probabilistically Checkable Proofs
- elements of the proof of the PCP Theorem

Optimization Problems

- many hard problems (especially **NP**-hard) are optimization problems
  - e.g. find *shortest* TSP tour
  - e.g. find *smallest* vertex cover
  - e.g. find *largest* clique

  - may be minimization or maximization problem
  - "opt" = value of optimal solution

Approximation Algorithms

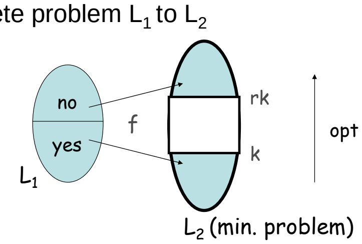
- often happy with approximately optimal solution
  - warning: lots of heuristics
  - we want approximation algorithm with guaranteed approximation ratio of r
  - meaning: on every input x, output is guaranteed to have value

    at most r*opt for minimization

    at least opt/r for maximization

Approximation Algorithms

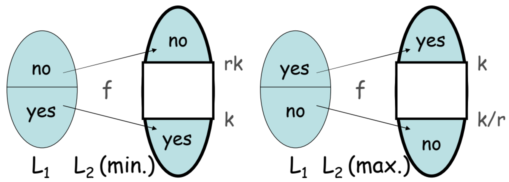
- "gap-producing" reduction from **NP**-complete problem L_1 to L_2



Gap producing reductions

- r-gap-producing reduction:
  - f computable in poly time
  - $x \in L_1 \Rightarrow \text{opt}(f(x)) \leq k$
  - $x \notin L_1 \Rightarrow \text{opt}(f(x)) > rk$
  - for max. problems use "$\geq$ k" and "< k/r"
- Note: target problem is not a language
  - **promise problem** (yes $\cup$ no *not* all strings)
  - "promise": instances always from (yes $\cup$ no)

Gap producing reductions

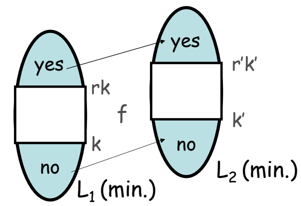


- Main purpose:
  - r-approximation algorithm for L_2 distinguishes between f(yes) and f(no); can use to decide L_1
  - "**NP**-hard to approximate to within r"

Gap preserving reductions

- gap-producing reduction difficult (more later)
- but **gap-preserving** reductions easier

Warning: many reductions **not** gap-preserving



Gap preserving reductions

- Example **gap-preserving** reduction:
  - reduce MAX-k-SAT with gap ε
  - to MAX-3-SAT with gap ε' constants
  - "MAX-k-SAT is NP-hard to approx. within ε ⇒ MAX-3-SAT is NP-hard to approx. within ε' "
- **MAXSNP** (PY) – a class of problems reducible to each other in this way
  - PTAS for **MAXSNP**-complete problem iff PTAS for all problems in **MAXSNP**

MAX-k-SAT

- Missing link: first gap-producing reduction
  - history's guide

    it should have something to do with SAT
- Definition: MAX-k-SAT with gap **ε**
  - instance: k-CNF φ
  - YES: some assignment satisfies **all** clauses
  - NO: no assignment satisfies more than (1 – ε) fraction of clauses

Proof systems viewpoint

- k-SAT **NP**-hard ⇒ for any language L ∈ **NP** proof system of form:
  - given x, compute reduction to k-SAT: φ_x
  - expected proof is satisfying assignment for φ_x
  - verifier picks random clause ("local test") and checks that it is satisfied by the assignment

    $x \in L \Rightarrow \Pr[\text{verifier accepts}] = 1$

    $x \notin L \Rightarrow \Pr[\text{verifier accepts}] < 1$

Proof systems viewpoint

- MAX-k-SAT with gap **ε** **NP**-hard ⇒ for any language L ∈ **NP** proof system of form:
  - given x, compute reduction to MAX-k-SAT: φ_x
  - expected proof is satisfying assignment for φ_x
  - verifier picks random clause ("local test") and checks that it is satisfied by the assignment

    $x \in L \Rightarrow \Pr[\text{verifier accepts}] = 1$

    $x \notin L \Rightarrow \Pr[\text{verifier accepts}] \leq (1 - \varepsilon)$
  - can repeat O(1/**ε**) times for error < ½

Proof systems viewpoint

- can think of reduction showing k-SAT NP-hard as designing a proof system for **NP** in which:
  - verifier only performs local tests
- can think of reduction showing MAX-k-SAT with gap **ε** NP-hard as designing a proof system for **NP** in which:
  - verifier only performs local tests
  - invalidity of proof* evident all over: "holographic proof" and an ε fraction of tests notice such invalidity

PCP

- Probabilistically Checkable Proof (PCP) permits novel way of verifying proof:
  - pick random local test
  - query proof in specified k locations
  - accept iff passes test
- fancy name for a NP-hardness reduction

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] $\leq$ ½

PCP

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

<u>The PCP Theorem</u> (AS, ALMSS):

PCP[log n, 1] = **NP.**

PCP

<u>Corollary</u>: MAX-k-SAT is **NP**-hard to approximate to within some constant ε.

- using PCP[log n, 1] protocol for, say, VC
- enumerate all $2^{O(\log n)}$ = poly(n) sets of queries
- construct a k-CNF φ_i for verifier's test on each
  - note: k-CNF since function on only k bits
- "YES" VC instance $\Rightarrow$ all clauses satisfiable
- "NO" VC instance $\Rightarrow$ every assignment fails to satisfy at least ½ of the $\varphi_i \Rightarrow$ fails to satisfy an $\varepsilon = (½)2^{-k}$ fraction of clauses.

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^3, 1]** (“inner verifier”)
    - we will not prove

Proof Composition (idea)

NP ⊂ PCP[log n, polylog n] (“outer verifier”)
NP ⊂ PCP[n^3, 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^3, 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^3, 1] yields PCP[log n, 1]
- many details omitted…

The outer verifier

<u>Theorem</u>: 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 $f:(F_q)^m \to F_q$
  - 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 $f:(F_q)^m \to F_q$
  - 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 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 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
  - log n random bits to pick a constraint
  - query assignment in 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 $\{0,1\} \subset H$, $|H|$ = 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]

ψ:$H^m \times H^m \times H^m \times H^3 \to \{0,1\}$ 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 ψ':$(F_q)^{3m+3} \to F_q$ 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]

ψ':$(F_q)^{3m+3} \to F_q$ 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]

ψ':$(F_q)^{3m+3} \to F_q$ 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: $p_1(x_1, x_2, x_3, \ldots, x_{3m+3}) =$

  $\Sigma_{h_j \in H} p_0(h_j, x_2, x_3, \ldots, x_{3m+3}) x_1^j$
- **<u>Claim</u>**:

  $p_0(Z)=0\ \forall Z \in H^{3m+3} \Leftrightarrow p_1(Z)=0\ \forall Z \in F_q \times H^{3m+3-1}$
- 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) ≤ deg(p_0) + |H|
- Claim:

  $p_0(Z)=0\ \forall Z \in H^{3m+3} \Leftrightarrow p_1(Z)=0\ \forall Z \in F_q \times H^{3m+3-1}$
- 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: $p_2(x_1, x_2, x_3, \ldots, x_{3m+3}) =$

  $\Sigma_{h_j \in H} p_2(x_1, h_j, x_3, x_4, \ldots, x_{3m+3}) x_2^j$

  deg(p_2) ≤ 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: $p_{i-1}:(F_q)^{3m+3} \rightarrow F_q$

– define: $p_i(x_1, x_2, x_3, \ldots, x_{3m+3}) = \Sigma_{h_j \in H} p_2(x_1, x_2, \ldots, x_{i-1}, h_j, x_{i+1}, x_{i+2}, \ldots, x_{3m+3}) x_i^j$

deg(p_i) ≤ 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. $\delta_i:F_q \rightarrow F_q$ so that
- $\delta_i(v) = 1$ if $v = i$
- $\delta_i(v) = 0$ if $0 \le v \le 3m+3+1$ and $v \neq i$

– define $Q:F_q \times (F_q)^{3m+3} \rightarrow F_q$ 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 ⊂ 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 $f:(F_q)^m \rightarrow F_q$

  – YES: some degree d assignment satisfies all constraints

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

NP ⊂ PCP[log n, polylog n]

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

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

- given $Q:F_q \times (F_q)^{3m+3} \rightarrow F_q$ 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 ⊂ 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 $m = O(\log n/\log\log n)$
  - $H \subset F_q$ 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 $d/q = o(1) << \varepsilon$

NP ⊂ PCP[log n, polylog n]

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