More hardness of approximation

Last time: New notion of "proof" (probabilistic proof systems)

⇒ PCP theorem

⇒ Hardness of approximation for CSPs in general,

Max-3SAT specifically:

- $\frac{15}{16}$ using first PCP thm
- $\frac{7}{8}+\varepsilon$ using Håstad's 3-bit PCP thm

Today: Another notion of "proof", hardness of approximation.

Thm (last time): For Max-3SAT, it is NP-hard to distinguish instances in which all clauses satisfiable from instances in which at most $\frac{15}{16}$ of clauses are satisfiable

- YES instance: all clauses satisfiable
- NO instance: $\leq \frac{15}{16}$ fraction of clauses satisfiable

Note: $\frac{15}{16}$ worse than $\frac{7}{8}$, but completeness 1 is nice property!

Max-3SAT: Every clause has 3 literals

Max-3SAT-5: - Every clause has 3 literals
- Every variable in 5 clauses

Standard transformation from Max-3SAT to Max-3SAT-5
- loses a constant in soundness

Thm: For Max-3SAT-5, it is NP-hard to distinguish between:
- instances in which all clauses satisfiable (YES instances)
- instances in which $\leq 1-\varepsilon$ fraction of clauses are satisfiable (NO instances)

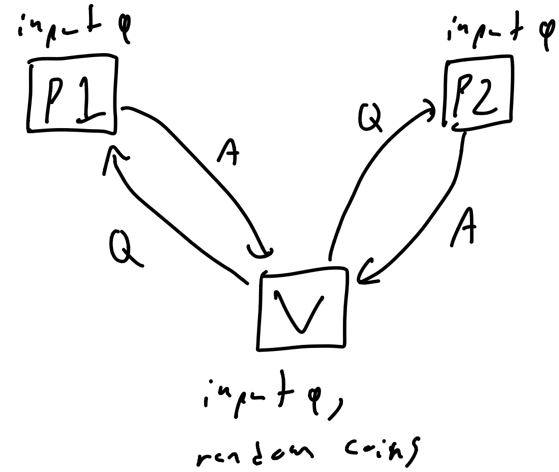
for some constant $\varepsilon > 0$

1-Round
Two-Prover proof system for language L

- Two provers, one verifier
- Verifier asks each prover a question (possibly different)
  - Provers answer. Computationally Unbounded
- Based on responses, verifier decides whether to accept (YES) or reject (NO). Must run in polytime
- Provers can decide on a strategy beforehand, but can't

communicate with each other after receiving question

- Provers trying to get verifier to accept
- Verifier trying to check if $\varphi \in L$



Trivial 2-prover proof system for 3SAT-S

- Verifier asks each prover for assignment
- Checks whether each assignment satisfies all clauses, or $\leq 1-\varepsilon$ fraction of clauses

$\Rightarrow$ if φ YES instance, provers can get verifier to accept with prob. 1 (completeness 1)

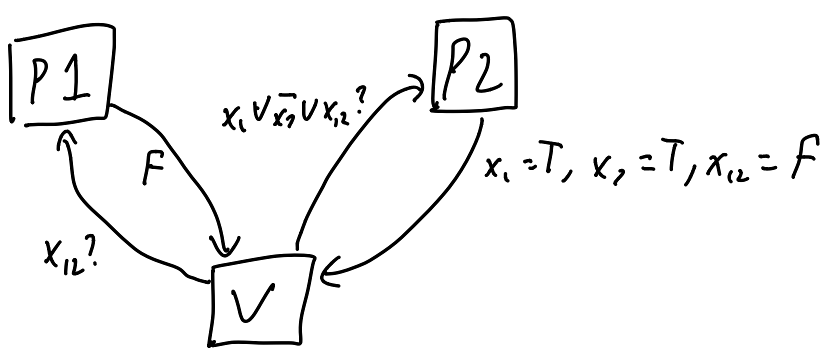
if φ NO instance, no matter what provers do, verifier accepts with prob. 0 (soundness 0).

What if we want questions, answers to be "short"?

Use randomness!

Verifier on instance φ:

- Choose clause C uniformly at random
- Choose one of the three variables in C uniformly at random. Call it x_i
- Ask prover 1 for an assignment to x_i (T/F)
- Ask prover 2 for a satisfying assignment to C (7 possibilities)
- P2's answer includes assignment to x_i. Return YES if it matches P1's assignment to x_i. Otherwise return NO.



Pick $C = x_1 \vee \bar{x}_7 \vee x_{12}$
uniformly at random
Pick x_{12} u.a.r. from $\{x_1, x_7, x_{12}\}$

Lemma: If φ a YES instance (there is an assignment satisfying all clauses), then provers can get verifier to return YES with probability 1.

Pf:

Return appropriate part of satisfying assignment!

Lemma: If φ a NO instance (every assignment satisfies at most $1-\varepsilon$ fraction of clauses), then no matter what provers do,

$$\Pr[\text{Verifier returns YES}] \leq 1 - \varepsilon/3$$

Pf: Note: provers are deterministic

P1: Has some assignment, returns $x_i = T/F$ depending on assignment

$\Rightarrow$ satisfies $\leq 1-\varepsilon$ of clauses

$\Rightarrow$ with prob. $\geq \varepsilon$, we choose C not satisfied

P2: returns satisfying assignment for C

$\Rightarrow$ disagrees with P1 on at least one of the three vars

$\Rightarrow$ We choose which ver to ask P1 u.a.r from the 3

$\Rightarrow$ find disagreement with prob. $\geq \varepsilon/3$

New computational problem: Find best strategy for provers.

<u>Label Cover</u>:

Input: - Bipartite graph $G=(L,R,E)$

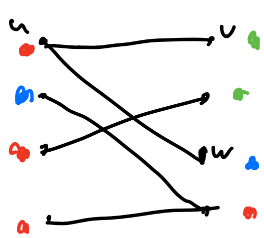
- Alphabet Σ_L - Alphabet Σ_R

- Relation $\Pi_e \subseteq \Sigma_L \times \Sigma_R$ for each $e \in E$

Feasible: Assignment $f: L \to \Sigma_L$ and $f: R \to \Sigma_R$

Objective: max fraction of edges (u,v) s.t. $(f(u), f(v)) \in \Pi_{u,v}$

<u>Ex</u>: $\Sigma_L = \Sigma_R = \{ {\color{red}\bullet}, {\color{green}\bullet}, {\color{blue}\bullet} \}$



$\Pi_{u,v} = \{ ({\color{red}\bullet}, {\color{green}\bullet}), ({\color{red}\bullet}, {\color{blue}\bullet}) \}$

$\Pi_{u,w} = \{ ({\color{blue}\bullet}, {\color{blue}\bullet}), ({\color{green}\bullet}, {\color{green}\bullet}), ({\color{green}\bullet}, {\color{blue}\bullet}), ({\color{blue}\bullet}, {\color{red}\bullet}) \}$

Informal claim: this is the problem of finding the best strategy for provers!

On input φ with n variables and $m = \frac{5}{3}n$ clauses;

L = variables (vertex for each variable)

R = clauses (vertex for each clause) $\qquad x_i$ — $x_i \cup \bar{x}_j \cup x_k$

E: add edge b/w every vertex and clause it appears in

$\Rightarrow$ left nodes have degree 5

right nodes have degree 3

$\Sigma_L = \{T, F\}$ $\qquad \Sigma_R = \{$7 satisfying assignments$\}$

$\Pi_{(u,v)}$: 7 pairs out of 14 that are consistent

Regularity $\Rightarrow$ choosing random C, random $x_i \in C$ same as choosing random edge

$\Rightarrow$ LC solution f is a strategy for provers where

Pr[verifier accepts] = fraction of edges whose relation is satisfied by f

= LC objective

Thm: There is some constant $\varepsilon > 0$ s.t. it is NP-hard to distinguish between instances of Label Cover where

- All edges can be satisfied
- At most $1-\varepsilon$ fraction of edges can be satisfied

$\Rightarrow$ NP-hard to approximate LC better than $1-\varepsilon$

Turns out LC much harder.

Back to 2-prover proof system: how to boost soundness from $1-\varepsilon/3$ to something smaller?

(How to boost probability of catching provers in inconsistency)?

Obvious approach: repetition

Repeat k times $\Rightarrow$

$\Pr[\text{never detect inconsistency}] \leq (1-\varepsilon/3)^k$

Works great! But to maintain connection to LC, need to maintain 1 round

Idea: repeat in parallel

Verifier:

- choose k random clauses $C_1, C_2, \dots, C_k$
- From each clause C_i choose random variable x_i from C_i
- Ask prover 1 for assignment for every x_i
- Ask prover 2 for satisfying assignment for every C_i
- Return YES if consistent on all k,
  NO otherwise

Gives LC instance:

$L = [n]^k \qquad R = [m]^k \qquad \Sigma_L = [2]^k \qquad \Sigma_R = [7]^k$

$$\Pi_{(x_1,\dots,x_k),(C_1,\dots,C_k)} = \{((\alpha_1,\dots,\alpha_k),(\beta_1,\dots,\beta_k)) \in [2]^k \times [7]^k : (\alpha_i, \beta_i) \in \Pi_{(x_i, C_i)} \text{ for all } i \in [k]\}$$

Q: Is asking questions in parallel same as repetitively?

Intuition: yes. How can provers cheat by knowing questions in parallel?

Truth: No! Provers can convince verifier with prob. $> (1-\varepsilon/3)^k$

But parallel almost as good:

Raz's Parallel Repetition Lemma:

If every assignment satisfies $\leq 1-\varepsilon$ fraction of clauses, then there is some constant $c>0$ s.t. $\forall k$, no matter what provers do in k-parallel repetition,

$Pr[\text{Verifier returns YES}] \leq (1-\varepsilon)^{ck}$

Implication to Label Cover:

Thm: There is some $\varepsilon>0$ and $c>0$ s.t. $\forall k \geq 1$, unless $NP \subseteq DTIME(n^{O(k)})$, there is no polytime algorithm which can distinguish between instances of Label Cover where:

- all edges can be satisfied
- $\leq (1-\varepsilon)^{ck}$ fraction of edges can be satisfied

Note: Instead of assuming $P \neq NP$, assuming $NP \not\subseteq DTIME(n^{O(k)})$ b/c size of LC instance $\approx n^k$

For any constant k, $DTIME(n^{O(k)}) \subseteq P$

Corollary: For any constant $0 < \alpha \le 1$, unless $P = NP$ there is no polynomial time α-approximation algorithm for Label Cover

Sps set $k = \Theta(\log^{\frac{1-\varepsilon}{\varepsilon}} n)$

$\Rightarrow$ LC graph has size $= N = n^k = n^{\Theta(\log^{\frac{1-\varepsilon}{\varepsilon}} n)}$

$\Rightarrow \log N = \Theta(\log^{\frac{1-\varepsilon}{\varepsilon}} n \cdot \log n) = \Theta(\log^{\frac{1}{\varepsilon}} n)$

$\Rightarrow \log n = \Theta(\log^{\varepsilon} N)$

$\Rightarrow$ inapproximability $\approx (1-\varepsilon)^{ck}$

$$= (1-\varepsilon)^{c \cdot \log^{\frac{1-\varepsilon}{\varepsilon}} n}$$

$$\le 2^{-c' \log^{\frac{1-\varepsilon}{\varepsilon}} n}$$

$$\le 2^{-c' \frac{\log^{\frac{1}{\varepsilon}} n}{\log n}}$$

$$= 2^{-c' \cdot \frac{\log N}{\log^{\varepsilon} N}}$$

$$\le 2^{-\log^{1-\varepsilon} N}$$

Quasipolytime: time $O(n^{\mathrm{polylog}(n)})$

Thm: For any $\varepsilon > 0$, unless NP has quasipolytime algorithms, there is no polytime algorithm for Label Cover with approximation better than $2^{-\log^{1-\varepsilon} n}$