More hardness of approximation

Last time: New notion of "proof" (probabilistic proof systems)
- PCP theorem
- Hardness of approximation for CSPs in general
- \text{Max-3SAT} specifically:
  - $\frac{15}{16}$ using first PCP theorem
  - $\frac{2}{8}$ using Hastad's 2-bit PCP theorem

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

The last time: For \text{Max-3SAT}, it is \text{NP}-hard to distinguish instances in which all clauses satisfiable from instances in which at most $\frac{15}{16}$ of clauses are satisfiable.
- \text{YES} instance: all clauses satisfiable
- \text{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!

\text{Max-3SAT}: Every clause has 3 literals
Max-$3\text{SAT}$-$S$:
- Every clause has 3 literals
- Every variable in 5 clauses

Standard transformation from Max-$3\text{SAT}$ to Max-$3\text{SAT}$-$S$
- loses a constant in soundness

Then: For Max-$3\text{SAT}$-$S$, it is NP-hard to distinguish between:
- instances in which all clauses satisfiable (YES instances)
- instances in which $\leq \varepsilon$ fraction of clauses are satisfiable (NO instances)

for some constant $\varepsilon > 0$

One-round Two-Prover proof system for language $L$
- Two provers, one verifier. All know input $\Phi$
- Verifier asks each prover a question (possibly different)
- Provers answer. Computationally unbounded, deterministic
- 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 yes

```
Verifier

Verifier

Verifier

Verifier

Verifier

Verifier

Verifier

Verifier

Verifier

Verifier
```

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

2) if \( \varphi \) YES instance, provers can get verifier to accept with prob. 1 (completeness 1)

it \( \varphi \) 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!

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

- Pick C uniformly at random
- Pick x₁₂ u.a.r. from \{x₁, x₇, x₁₂\}
- Pick \(x_i \lor \overline{x_i} \lor x_{i2}\)
Lemma: If \( \Phi \) is a YES instance (there is an assignment satisfying all clauses), then provers can get verifier to return YES with probability 1.

Proof:

Return appropriate part of satisfying assignment!

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

\[
\Pr_{\text{verifier returns YES}} \leq 1 - \frac{\epsilon}{3}
\]

Proof: Note: provers are deterministic

P1: Has some assignment, returns \( X_i = P1 \) depending on assignment

\( \exists \) satisfies \( \leq 1-\epsilon \) of clauses

\( \exists \) with prob. \( \geq \epsilon \), we choose \( C \) not satisfied

P2: returns satisfying assignment for \( C \)

\( \exists \) disagrees with P1 on at least one of the three cars
we choose which ver to ask P1 u.a. ver from the 3
find disagreement with prob. ≥ ε/3

New computational problem: Find best strategy for prov.:

**Label Cover:**

- **Input:** Bipartite graph \( G = (L, R, E) \)
  - Alphabet \( \Sigma_L \)
  - Alphabet \( \Sigma_R \)
  - Relation \( \Pi_e \subseteq \Sigma_L \times \Sigma_R \) for each \( e \in E \)

**Feasible Assignments:** \( f : L \rightarrow \Sigma_L \) and \( f : R \rightarrow \Sigma_R \)

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

**Ex:** \( \Sigma_L = \Sigma_R = \{\circ, \bullet, \star\} \)

\[
\begin{align*}
\Pi_{u,v} &= \{(\circ, \circ), (\circ, \bullet), (\bullet, \bullet)\} \\
\Pi_{w,u} &= \{(\bullet, \circ), (\circ, \star), (\star, \bullet), (\star, \circ)\}
\end{align*}
\]
Informal claim: this is the problem of finding the best strategy for provers!

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

$L$ = variables (vertex for each variable) 
$R$ = clauses (vertex for each clause)

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

$= \text{left nodes have degree 5}$

$\Rightarrow \text{right nodes have degree 3}$

$\Sigma_L = \{ T, F \}$

$\Sigma_R = \{ 7 \text{ satisfying assignments} \}$

$\Pi_{(\varphi)}$ = 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

$P[\text{verifier accepts}] = \text{fraction of edges whose relation is satisfied by } f$

$= \text{LC objective}$
There is some constant \( \varepsilon > 0 \) s.t. it is \( \text{NP} \)-hard to distinguish between instances of Label Cover where
- All edges can be satisfied
- At most \( 1 - \varepsilon \) fraction of edges can be satisfied

\( \Rightarrow \text{NP} \)-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 \( \Rightarrow \)

\( p(\text{never detect inconsistency}) \leq (1 - \varepsilon/3)^K \)

Works great! But to maintain connection to LC, need to maintain 1 round.
Idea: repeat in parallel

Verifier:
- Choose \( k \) random clauses \( C_1, C_2, \ldots, C_k \)
- From each clause \( C_i \) choose random variable \( x_i \) from \( C_i \)
- Ask prover \( L \) for assignment for every \( x_i \)
- Ask prover \( L \) for satisfying assignment for every \( C_i \)
- Return YES if consistent on all \( k \)
  NO otherwise

Gives \( \mathcal{L}C \) instance:
\[
\mathcal{L} = \mathcal{L}n^k \quad \mathcal{R} = \mathcal{L}m^k \quad \mathcal{E}_L = \mathcal{L}2^k \quad \mathcal{E}_R = \mathcal{L}7^k
\]

\[\Pi(x_1, \ldots, x_k), (x_1, \ldots, x_k) = \begin{cases} (x_1, \ldots, x_k), (P_1, \ldots, P_k) \in \mathcal{L}2^k \times \mathcal{L}7^k: \\
(x_i, P_i) \in \Pi(x_i, C_i) \text{ for all } i \in \mathcal{C}[k] \end{cases}\]

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 \( > (1 - \varepsilon^2)^k \)
But parallel almost as good:

**Raz's Parallel Repetition Lemma:**

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

\[
p[\text{Verifier returns YES}] \leq (1 - \varepsilon)^c k (1 - e^{-c})^k
\]

**Implication to Label Cover:**

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

- all edges can be satisfied
- \(\leq (1 - \varepsilon)^c k\) fraction of edges can be satisfied

**Note:** Instead of assuming \(P \neq NP\), assuming \(NP \not\subseteq \text{DTIME}(n^{O(k)})\)

- size of LC instance \(= n^k\)

For any constant \(k\), \(\text{DTIME}(n^{O(k)}) \subseteq P\)
Corollary: For any constant $0 < \alpha \leq 1$, unless $P = NP$, there is no polynomial time $\alpha$-approximation algorithm for Label Cover.

Set $k = \Theta \left( \log \frac{1-\epsilon}{\epsilon} n \right)$.

1. Let graph $G$ have size $N = n^k = \Theta(\log \frac{1-\epsilon}{\epsilon} n)$.

2. $\log N = \Theta \left( \log \frac{1-\epsilon}{\epsilon} n \cdot \log n \right) = \Theta \left( \log \frac{1}{\epsilon} n \right)$.

3. $\log n = \Theta \left( \log^\epsilon N \right)$.

4. Inapproximability $\approx \left( 1-\epsilon \right)^{ck}$.

\[
\begin{align*}
&= \left( 1-\epsilon \right)^{c \cdot \log \frac{1-\epsilon}{\epsilon} n} \\
&= \left( 1-\epsilon \right)^{-c \cdot \log \frac{1-\epsilon}{\epsilon} n} \\
&= 2^{-c \cdot \log \frac{1-\epsilon}{\epsilon} n} \\
&= 2^{-c \cdot \frac{\log N}{\log^\epsilon N}} \\
&= 2^{-c \log_{\log^\epsilon N} N} \\
&\leq 2^{-\log_{\log^\epsilon N} N}
\end{align*}
\]

Quasipolytime: time $O(n \log^{polylog(n)})$. 
Thm: For any $\varepsilon > 0$, unless $NP$ has quasi-polytime algorithms, there is no polytime algorithm for Label Cover with approximation better than $2^{-\log^c n}$.

Uniq Game:

**Claim:** It is $NP$-hard to distinguish

$1 - \varepsilon$ vs. $\varepsilon$ in Unique Games over $\mathbb{F}_2$. 

\[
\text{uniqueness} \quad \text{partition}
\]