Class Notes 4/28/15: Inapproximability maximization

Say want to prove problem $\Pi_1$ hard to approximate.

Gap reduction from a problem $\Pi_2$ that we already know is NP-hard (e.g., SAT)

\[ f : \text{ESAT} \rightarrow \text{ESAT} \]

$\Pi_2 \approx \text{SAT}$

Say could approximate $\Pi_1$ better than $\frac{\alpha}{\beta}$.

Say $\gamma$-approx to $\Pi_1$, i.e. $\gamma < \frac{\alpha}{\beta}$.

\[ \Rightarrow \text{on SAT instance } x, \text{ use reduction to get } F(x), \text{ run } \gamma \text{-approx on } F(x) \text{ to get ALG} \]

If $x$ is not satisfiable, then $\gamma \text{-opt}(F(x)) \leq \delta \beta < \alpha$.

If $x$ is satisfiable, then $\text{ALG} \geq \gamma \text{-opt}(F(x)) \geq \alpha \beta \delta > \beta$.

\[ \Rightarrow \text{polytime alg. for SAT.} \]

Usually easier to start w/ a problem where there's already a gap, e.g. if we are want to prove hardness for $\Pi_3$,

\[ \text{out} \leq \text{opt} \]

\[ \text{out} > \text{opt} \]

\[ \text{out} \leq \text{opt} \]

\[ \text{out} > \text{opt} \]

\[ \Gamma_1 \]

Doesn't matter what reduction does to middle!
So generic reduction technique: start w/ problem \( \Pi_2 \) where \( \Pi_2 \in \text{NP-hard to distinguish instances are partitioned into 3 groups: YES, NO, MAYBE, and is NP-hard to distinguish YES instances from NO instances. \} \)

- completeness: If \( x \in \text{YES} \), \( \text{OPT}(F(x)) \geq \alpha \)
- soundness: If \( x \in \text{NO} \), \( \text{OPT}(F(x)) \leq \beta \)

More powerful than usual reductions since don't need to worry about what happens to MAYBE.

Can get pretty far w/ this (Book 16.1, 16.2), but real improvement from PCP theorem.

**Def:** \( \text{LENP} \) if \( \exists \) polynomial \( p \) and alg. \( A \) s.t. \( x \in L \), there exists \( \text{proof} \ y \) s.t.

1. if \( x \in L \), \( \exists \) proof \( y \) s.t. \( |y| \leq p(|x|) \) and \( A(x,y) \) returns \text{YES} w/ \text{one-sided error} \( \leq p(|x|) \)
2. if \( x \notin L \), then \( A(x,y) \) returns \text{NO}.

**Def:** A probabilistic proof system is a verifier \( A \) s.t.

1. Verifier reads \( r(n) \) random bits w/ \( c(n) \)
2. Verifier reads \( r(n) \) bits of a (proof
3. If \( x \in L \), then a verifier returns \text{YES} w/ \text{wp.} \geq c(n) \) (completeness)
4. If \( x \notin L \), then a verifier returns \text{YES} w/ \text{wp.} \leq s(n) \) (soundness)

**Example:** If \( \text{LENP} \), then \( L \) has a probabilistic proof system w/ \( r(n) = 0 \), \( c(n) = 0 \), \( s(n) = 0 \).
NEP: Let \( \text{PCP}_{\epsilon_0, \delta_0} (x^n, y^n) \) be the class of languages that have a probabilistic proof system with parameters 
\( (\epsilon_0, \delta_0, n) \).

So in this setting, \( \text{NP} = \text{PCP}_{\epsilon_0, \delta_0} (O(1), \text{poly}(n)) \).

Theorem (PCP Thm): \( \text{NP} = \text{PCP}_{\frac{1}{2}, 1} (O(k), O(1)) \)

Easy direction: \( \text{PCP}_{\frac{1}{2}, 1} (O(k), O(1)) \subseteq \text{NP} \).

For each choice of \( O(k) \) random bits, the verifier checks \( O(1) \) bits of the proof.

\( \implies \) only \( 2^{O(k)} \cdot O(1) = \text{poly}(k) \) bits of proof need ever be looked at.

So \( \text{NP} \) verifier \( A \) can simply try each \( 2^{O(k)} \) possible choices of random bits, simulate \( \text{PCP} \) verifier on each one, return \( \text{YES} \) if all runs return \( \text{YES} \) & return \( \text{NO} \) otherwise.

- If \( x \in L \) then \( \text{PCP} \) proof verifier always returns \( \text{YES} \).
- If \( x \notin L \) then on \( \approx \frac{1}{2} \) choices of random bits, \( \text{PCP} \) verifier returns \( \text{NO} \) \( \implies \) \( \text{NP} \) verifier will return \( \text{NO} \).

Real difficulty is proving \( \text{NP} = \text{PCP}_{\frac{1}{2}, 1} (O(k), O(1)) \).

Why is this useful? for proving hardness of approximation?

Intuition: Let \( T \) be arbitrary \( \text{NP} \)-complete problem (e.g., 3SAT).

Consider problem \( T \).

Then by PCP Thm: \( A \) \( \text{NP} \) verifier \( \neq O(k) \) random bits, \( O(1) \) query \( \text{poly}(k) \) bits.

\[ \text{poly}(k) = \text{poly}(\log n) \]
Let $T'$ be the problem of designing a proof to maximize probability that verifying says YES.

By def, hard to approximate better than $\frac{1}{2}$.

What kind of problem? A CSP!

For each of the poly$(n)$ choices of random bits, verifier computed a (deterministic) function

$f(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the first bits queried.

$\exists$ poly$(n)$ constraints, each $a(x, y)$ is $1$ if $x \oplus y = 0$ s.t.

satisfied constraints.

If have soundness $s(x)$, completeness $c(x)$, hard to approx better than $\frac{f(n)}{c(n)}$.

By rewriting arbitrary $f(x)$ as 3CNF formula, gives $\frac{1}{2}$-hardness for Max-3SAT.

Can do better by restricting these functions through different versions of PCP

Def: let $\text{odd}(x, y, z) = 1$ if $x + y + z$ odd, $\text{even}(x, y, z) = 1$ if $x + y + z$ even.

Then $\text{H falsify}$: For any constant $\delta, \epsilon > 0$

$NP \subseteq \text{PCP}_{\delta, \epsilon} \text{ poly}(\log n, 1)$ and verifier restricted to odd and even $f(x)$

Then: $\exists$ poly-time algorithm that is $\text{NP}$-hard to approximate Max-3SAT better than $\frac{7}{8} + \epsilon$
PE: Stat w/ arbitrary NP-complete problem $\text{SAT}$.

By Hatsett, I verified $\text{NP}$-completeness 1-2 randomness $\mathcal{O}(\log)$ random bits and making 3 queries using odd even tests (b).

Let $N = 2^m$ be the number of random strings used.

Each of $N$ random strings gives three bits and an odd (even) test $x_i$.

We will construct $\text{Max-3SAT}$ instance out of these $N$ tests.

For each even test $x_i$, create 4 clauses:

- $x_i \lor \overline{x}_j \lor \overline{x}_k$
- $\overline{x}_i \lor x_j \lor x_k$  \quad \text{if odd}(x_i, x_j, x_k) \text{ satisfied, all 4 clauses satisfied}
- $x_i \lor \overline{x}_j \lor \overline{x}_k$  \quad \text{if odd}(x_i, x_j, x_k) \text{ not satisfied}
- $\overline{x}_i \lor x_j \lor \overline{x}_k$  \quad \text{then exactly 3 clauses satisfied}

For each even $(x_i, x_j, x_k)$ constraint, 4 clauses:

- $x_i \lor x_j \lor x_k$
- $\overline{x}_i \lor \overline{x}_j \lor \overline{x}_k$  \quad \text{if even}(x_i, x_j, x_k) \text{ satisfied, all 4 clauses satisfied}
- $x_i \lor \overline{x}_j \lor \overline{x}_k$  \quad \text{if exactly 3 satisfied}
- $\overline{x}_i \lor x_j \lor \overline{x}_k$  \quad \text{otherwise}

Consider instance $q$. If $q \in \text{SAT}$ (Yes instance), then

3 proof s.t. verifier accepts w.p. $\ge 1 - \varepsilon$ (Hatsett).

$\Rightarrow$ there is a way of satisfying $\ge 4(1 - 2\varepsilon)N + 2\varepsilon N$

$\ge (4 - 5\varepsilon)N$ clauses of $T(q)$

If $q \notin \text{SAT}$ (No instance), then $\forall$ proofs, verifier accepts w.p. $\le \frac{1}{2} + \delta$

Then in any assignment for $T(q)$, can only satisfy all

4 clauses for $\le (\frac{1}{2} + \delta)N$ constraints

$\Rightarrow$
any assignment for $f(q)$ satisfies $\leq 4(\frac{1}{2} + \epsilon)N + 3(\frac{1}{2} - \epsilon)N = \frac{7}{2} + \epsilon)N$ classes

The trace of $\frac{7 + \epsilon}{4 - \epsilon}N = \frac{7}{6} + \epsilon'$