Today

- Capturing the power of the prover in PCPs.
- Approximability and Inapproximability.
- Average-case Hardness.

Recall PCPs

Defn: (r, q)-restricted PCP verifier is a prob. polytime machine with access to oracle that tosses $r(n)$ coins and queries the oracle $q(n)$ times to decide whether it accepts x of length n.

Defn: $\text{PCP}[r, q]$ is the class of languages L s.t. there exists a (r, q)-restricted PCP verifier with

Completeness For every $x \in L$, there exists a proof oracle π such that $V^{\pi}(x)$ accepts w.p. 1.

Soundness For every $x \notin L$, for every proof oracle π, $V^{\pi}(x)$ accepts w.p. $\leq \frac{1}{2}$.

History of PCPs

Defn: Defined explicitly by Arora & Safra 1992, based on implicit definition by Feige et al. 1991. Variant defns already defined by Fortnow et al. 1988, Babai et al. 1991.

Major results:

Babai-Fortnow-Lund (1990): $\text{NEXP} \subseteq \text{PCP}[\text{poly}, \text{poly}]$.

Arora et al. (1992): $\text{NP} = \text{PCP}[O(\log n), O(1)]$.

Hastad (1997): $\text{NP} = \text{PCP}[O(\log n), 3]$.

Optimal proof for PCP

- Let bits of proof be variables $\pi_1, \ldots, \pi_n$.
- For fixed randomness, verifier's actions give a decision tree of depth 3 on variables $\pi_1, \ldots, \pi_n$.
- Exercise: Convert depth-3 decision tree into $\ell \leq 8$ clauses such that every assignment to variables satisfies at least $\ell - 1$ clauses and satisfies all iff decision tree accepts.
- Create such block of clauses for every random string and take their conjunction.
- If $x \in L$ then formula satisfiable. If $x \notin L$ then at most $15/16$ fraction of clauses satisfied by any assignment.

- Conclude: If you can find assignment satisfying more that $15/16$ fraction of clauses in every satisfiable SAT formula, then can decide $\mathrm{PCP}[O(\log n), 3]$ and hence (by Hastad) can decide NP.

- Or equivalently, Can't approximate # satisfiable clauses by factor of $15/16$ in P unless NP=P.

Complexity and Optimization

Combinatorial optimization problems: described by a triple (sol?, obj, opt), where sol?: $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and obj: $\{0,1\}^* \times \{0,1\}^* \to \mathbb{R}^+$ are polytime computable, and opt$\in \{\max, \min\}$.

Given x, goal is to find solution y (i.e., sol?(x.y) = 1) so as to opt obj(x,y).

P and NP (and P?=NP) owe their popularity in large measure due to ability to explain solvability of optimization problems, in theory.

Gap between theory and practice

- NP-completeness is not the end of the story.
- In practice people still develop heuristics.
- Typical justification: "Heuristic comes to within 99% of optimum on 95% of all cases."
- Does this contradict NP-completeness?
- No, No! On two grounds:
  - Approximation, not exact.
  - Average-case, not worst-case.

Approximability

- Given optimization problem Π =(sol?,obj,opt) and function $\alpha : \mathbb{Z}^+ \to \mathbb{R}^+$, the (Π, α) optimization problem is that of computing a solution y to x satisfying
$$y/\alpha(|x|) \leq opt \leq y\alpha(|x|)$$
. (Note need $\alpha(\cdot) \geq 1$).

- NP-completeness usually gives negative results about $(\Pi, 1)$. But what about $(\Pi, 2)$.

- Example: (Clique,1) = (Coloring,1) = (MaxSAT,1).

- Is (Clique,2) = (Coloring, 2) = (MaxSAT,2)?
- Presumably not, since (MaxSAT,2) is in P, and (Clique,2) (thanks to your next problem set) is NP-hard!
- Need to study (Π, α) seperately for each Π and α.

PCP and (in)-approximability

- PCP theorem shows that (MaxSAT,16/15-ϵ) is NP-hard (actually (MaxSAT, 8/7 - ϵ) if you are careful).
- Shows many other such results.
- Consequence: Have good understanding of this variation.

Average-case vs. worst-case: The other objection

- NP-completeness only talks about problems on the worst-case.
- In practice, don't have to worry about the worst-case.
- Theoretical justification: Too complex for environment to compute the worst-case.
- So environment also polynomial time bounded, but maybe can toss random coins. If so, should only worry about average-case.
- But average-case on what distribution?

- Don't know, but will make this part of the problem.

Distributional problems

- (Π, D), where Π is a problem and $D = \{D_n\}_n$ is a distribution on n-bit strings.
- Can now ask: How hard is it to compute Π on distribution D?
- No different from worst-case unless D is restricted (or else, consider the distribution $D = \sum_{i=1}^{\infty} 2^{-i}$ Bad input for machine M_i to solve Π).
- Restriction on D? Make it polynomial time sampleable. Can pick $x \in \{0,1\}^n$ according to D_n in time polynomial in n. Will mix notation a bit to say D_n is the sampling circuit.

- Why not just uniform?
  - Chromatic number of most graphs = $n/\log n$ - so $\log n$ approximation trivial.
  - Clique number of most graphs = $\log n$, so $\log n$ approximation trivial.
  - Yet Clique/Chromatic number not considered easy in practice. More interesting solutions desired.

Example: Distributed Permanent

Show Lipton's reduction.

DNP and Avg-P