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 \(\pi\) such that \(V^\pi(x)\) accepts w.p. 1.

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

History of PCPs


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 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 $(\text{sol?}, \text{obj}, \text{opt})$, where $\text{sol?): \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ and $\text{obj): \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{R}^+$ are polytime computable, and $\text{opt} \in \{\text{max, min}\}$.

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

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 $II = (\text{sol?}, \text{obj}, \text{opt})$ and function $\alpha: \mathbb{Z}^+ \rightarrow \mathbb{R}^+$, the $(II, \alpha)$ optimization problem is that of computing a solution $y$ to $x$ satisfying

$$y/\alpha(|x|) \leq \text{opt} \leq y\alpha(|x|)$$

. (Note need $\alpha(\cdot) \geq 1$).

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

• Example: $(\text{Clique},1) = (\text{Coloring},1) = (\text{MaxSAT},1)$.
• Is \((\text{Clique},2) = (\text{Coloring}, 2) = (\text{MaxSAT},2)\)?

• Presumably not, since \((\text{MaxSAT},2)\) is in P, and \((\text{Clique},2)\) (thanks to your next problem set) is NP-hard!

• Need to study \((\Pi, \alpha)\) seperately for each \(\Pi\) and \(\alpha\).

PCP and (in)-approximability

• PCP theorem shows that \((\text{MaxSAT},16/15-\epsilon)\) is NP-hard (actually \((\text{MaxSAT}, 8/7 - \epsilon)\) 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

- $(\Pi, D)$, where $\Pi$ is a problem and $D = \{D_n\}_n$ is a distribution on $n$-bit strings.

- Can now ask: How hard is it to compute $\Pi$ 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 $\Pi$).

- 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