CS221: Computational Complexity Prof. Salil Vadhan

Lecture 32: The **PCP** Theorem and Approximating Optimization Problems

12/11 *Scribe: Arthur Rudolph*

Contents

1 The PCP Theorem

We say that $L \in \mathbf{PCP}_{c,s}[r(n), q(n)]$ if there is a probabilistic polynomial time verifier V such that:

1. $x \in L \Rightarrow \exists \pi \text{ s.t.} \Pr[V^{\pi}(x) \text{ accepts}] \geq c$
2. $x \notin L \Rightarrow \forall \pi^* \text{ s.t.} \Pr[V^{\pi^*}(x) \text{ accepts}] \leq s$
3. $\forall x, \pi V^{\pi}(x)$ tosses at most $O(r(|x|))$ coins, and reads at most $q(|x|)$ bits from π

(We'll assume the reads are nonadaptive; we can convert q adaptive reads to 2^q nonadaptive reads by thinking about all possible responses and using the one that's "right"). We require $1 \geq c > s > 0$ constants. If it is not specified, assume that $c = 1$.

Theorem 1 (PCP Theorem (Arora-Safra, ALMSS '92))

$$\mathbf{NP} = \mathbf{PCP}(\log n, O(1))$$

That is, any mathematical proof can be encoded in such a way that they can be verified by reading only a *constant* number of (randomly chosen) bits! (Actually, we'll see that we can make the constant as small as three.)

One direction of the PCP theorem is easy:

Proposition 2 $\mathbf{PCP}(\log n, \mathrm{poly}(n)) \subseteq \mathbf{NP}$

Proof: Let $r = \log n$ and $q = \mathrm{poly}(n)$. There are at most $2^r q$ positions that V ever reads; with r, q as above, $2^r q = \mathrm{poly}(n)$. The witness that $x \in L$ is thus the **PCP** proof that makes V accept w.p. $\geq s$. We verify this witness by enumerating over all $2^r = \mathrm{poly}(n)$ sequences of coin tosses. ■

We'll sketch some of the ideas in the proof of the other direction, but only after spending a couple of lectures on the connection between the PCP Theorem and approximating optimization problems.

2 Hardness of Approximation

Recall that an **NP** optimization problems (from Problem Set 1) is a poly-time computable objective function.

$$\text{Obj} : \underbrace{\Sigma^*}_{\text{input}} \times \underbrace{\Sigma^*}_{\text{sol'n}} \to \mathbb{R}$$

The goal is to minimize (or maximize) this objective function.

MAXSAT**:** Given φ, find an assignment which satisfies as many clauses as possible.

MAXTSP**:** Given distances matrix, find the shortest travelling-salesman tour.

These problems are **NP**-hard. What about approximating them? That is, finding a solution s.t. the objective function is within some factor α of the minimum/maximum. This is the same as the notion of an *α-approximation algorithm* that we used when discussing approximate counting. With optimization problems, one can look at the function version, i.e. approximating the *value* of the optimum, or the search version, i.e. finding a solution that is approximately optimal. Most of what we'll say will apply to both versions.) We begin with some simple examples of approximation algorithms for optimization problems (more ones can be found in an algorithms class, e.g. CS223).

Proposition 3 $\exists$ *poly-time 2-approximation of* MAX3SAT

Proof: Either the all-1's or all-0's assignment satisfies at least half of the clauses, which is certainly within a factor of 2 of the optimum. ■

Actually there is an 8/7-approximation for MAX3SAT: it is fairly simple if all the clauses of size 3 (because a random assignment satisfies 7/8 of the clauses on average) and very sophisticated if the clauses can also be of size 1 and 2 (the algorithm uses semidefinite programming).

Can we get a better approximation factor than 8/7? How about arbitrarily close to 1? The **NP**-completeness of the exact version of MAX3SAT does not imply any sort of hardness of approximation, and obtaining such results was a longstanding open problem in complexity theory. That is, until the suprising discovery was made that approximation of optimization problems is intimately related to PCPs:

Theorem 4 *The* **PCP** *theorem holds iff* $\exists\, \varepsilon > 0$ *and a poly-time reduction mapping*

$$\begin{aligned} &\textit{3-CNF } \varphi \mapsto \textit{3-CNF } \varphi' \textit{ s.t.} \\ &\varphi \textit{ is satisfiable} \Rightarrow \varphi' \textit{ is satisfiable} \\ &\varphi \textit{ is unsatisfiable} \Rightarrow \textit{no ass't satisfies} \geq (1-\varepsilon) \textit{ fraction of clauses in } \varphi' \end{aligned}$$

(We call such a reduction an "amplifying reduction").

Proof: We prove the $\Rightarrow$ direction, we'll do the reverse next time. Let V be a $\mathbf{PCP}_{1,1/2}(\log n, q)$ verifier for SAT. (We can bring the soundness down to 1/2 by repetition, albeit while increasing the number of queries substantially.) On coin tosses r the verifier reads $i_1, \ldots, i_q$ from proof π, accepts according to $f_R(\pi_{i_1}, \ldots, \pi_{i_q})$ (this function also depends on φ). Construct φ':

- Start with variables $\pi_1, \ldots, \pi_m$ representing the bits of the **PCP** proof.
- φ is sat. $\Rightarrow \exists$ ass't to π_i's s.t.$\forall R f_R(\pi_{i_1}, \ldots, \pi_{i_q}) = 1$.
- φ is unsat. $\Rightarrow \forall$ ass't to π_i's $f_R(\pi_{i_1}, \ldots, \pi_{i_q}) = 1$ for at most $1/2$ of the R's (by soundness).

How do we make this into a 3CNF? Each f_R can be computed by a boolean circuit of constant size (since q is constant), so we can apply Cook-Levin:

- $f_R \to \varphi_R(\pi_{i_1}, \ldots, \pi_{i_q}, z_1, \ldots, z_t)$ where φ_R is a 3-CNF using $t = t(q)$ auxiliary variables. (We use different aux. vars for each φ_R.)
- $f_R(\bar{\pi}) = 1 \Rightarrow \exists \bar{z}$ s.t.$\varphi_R(\bar{\pi}, \bar{z}) = 1$.
- $f_R(\bar{\pi}) = 0 \Rightarrow \forall \bar{z}, \varphi_R(\bar{\pi}, \bar{z})$ has at least one clause not satisfied.

Say each φ_R has $c = c(q)$ clauses. Define $\varphi' = \wedge_R \varphi_R$. Then

$$
\begin{aligned}
\varphi \text{ sat.} \quad &\Rightarrow \quad \exists \text{ ass't satisfying all } \varphi_R\text{'s} \\
\varphi \text{ unsat.} \quad &\Rightarrow \quad \forall \text{ ass'ts, at least } 1/2 \cdot 1/c \text{ fraction of clauses are not satisfied}
\end{aligned}
$$

■

Corollary 5 *Assuming the PCP theorem and* $\mathbf{P} \neq \mathbf{NP}$*, there is an* $\varepsilon > 0$ *such that there is no polynomial-time* $(1+\varepsilon)$*-approximation algorithm for* Max3SAT.

What is the value of ε in this hardness of approximation result? Note that, in the above proof, it depends on the number of clauses in φ_R, which in turn depends exponentially on the number of queries made by the PCP. More generally, if we hadn't first reduced the soundness to 1/2, it involves some tradeoff between the soundness and the number of queries. Using the original PCP theorem, we either get a soundness that is very close to 1 or a very large, so the resulting ε is extremely small (like 1/10000). But we see that if we can sufficiently reduce the the number of queries and the error probabilities, we prove hardness for larger approximation factors. This is true not just for Max3SAT but also other optimization problems. Thus, much work in the past decade has been devoted to proving stronger, even optimal, forms of the PCP theorem. (Since there are 4 parameters in the definition of **PCP**, there are many possible optimal **PCP** theorems.) For example:

Theorem 6 (Håstad) *Every* $L \in \mathbf{NP}$ *has a* **PCP** *in which* V *reads only 3 bits from the proof and accepts according to the exclusive-or of those bits (or its negation). Here* $c = 1-\varepsilon, s = 1/2+\varepsilon$.

We'll see next time that this implies that it's **NP**-hard to approximate Max3SAT within $8/7 - \varepsilon$.