6.841: Advanced Complexity Theory Fall 2012

Lecture 16 — November 6th, 2012 (Prasad Raghavendra)

Prof. Dana Moshkovitz *Scribe: Geng Huang*

1 Overview

In this lecture, we will begin to talk about the "**PCP** Theorem" (Probabilistically Checkable Proofs Theorem). Since the discovery of **NP**-completeness in 1972, researchers had mulled over the issue of whether we can efficiently compute approximate solutions to **NP**-hard optimization problems. They failed to design such approximate algorithms for most problems. They then tried to show that computing approximate solutions is also hard, but apart from a few isolated successes this effort also stalled. Researchers slowly began to realize that the Cook-Levin-Karp-style reductions do not suffice to prove any limits on approximation algorithms [1].

The **PCP** Theorem states that every decision problem in the NP complexity class has probabilistically checkable proofs (proofs that can be checked by a randomized algorithm) of constant query complexity and logarithmic randomness complexity. The **PCP** Theorem gave a new definition of **NP** and provided a new starting point for reductions to show hardness of approximation [1].

We know that

$$\mathbf{NP} = \{L \mid \exists \text{ a polynomial time verifier } V \text{ s.t. } x \in L \iff \exists \text{ a proof } y, |y| = \text{poly}(|x|) \text{ s.t. } V(x,y) = 1\}$$

Our question is: how much work does the polynomial time verifier needs to do?

We know that **IP = PSPACE** and **MIP = NEXP** [2] [3] [4], where **MIP** is akin to the class **IP**, but now the verifier may interact with multiple provers that cannot communicate with each other. These two results showed that the verifier doesn't have to perform that much computation relative to the amount it would have to perform by itself, without a given proof. This motivated researchers to "scale" this result down to the class **NP**.

In this lecture, we will formally define what a probabilistically checkable proof system is, and how it is intimately related to the notion of hardness of approximation.

2 The PCP Theorem

Definition 1. *Let $\alpha = 5/6$. $PCP_{\Sigma}(q(n), r(n))$ is set of languages L such that a randomized polynomial time verifier V that uses $r(n)$ random coins and queries $q(n)$ locations in the proof $y \in \Sigma^*$, and has the property that for all x,*

1. *(Completeness) $x \in L, \exists y \in \Sigma^*$ such that $Pr[V(x,y)] = 1$*
2. *(Soundness) $x \notin L, \exists y \in \Sigma^*$ such that $Pr[V(x,y)] \leq \alpha$*

Observe that the length of the proof is bounded by $q(n)2^{r(n)}$, because there are only $2^{r(n)}$ possible choices of random bits, and for each, the verifier can query at most $q(n)$ new places. For a concrete instantiation, we consider $PCP_{\{0,1\}}(O(1), O(\log n))$, and it is easy to see that the proof length for this class of PCPs is polynomially bounded in n – hence, $\mathbf{PCP}_{\{0,1\}}(O(1), O(\log n)) \subseteq \mathbf{NP}$. From converse is the hard, and more interesting, part of the PCP Theorem.

Theorem 2 (PCP Theorem)**.** $\boldsymbol{NP} = \boldsymbol{PCP}_{\{0,1\}}(O(1), (\log n))$.

2.1 A new kind of proof checking

Imagine that we want to verify the proof for $a^3+b^3+c^3 = (a+b+c)(a^2+b^2+c^2-ab-bc-ac)$. A proof of this identity is usually presented in a sequential manner. The standard verification procedure is to read through such a proof sequentially, checking that each line follows from previous ones, and that the last line is a tautology (e.g. $0 = 0$).

Suppose we wanted to have a more efficient verification procedure – one where one only exams a few locations of the proof. Intuitively, it seems impossible to have a sound proof checking procedure: suppose the given proof was completely consistent except for a single mistake or contradiction (e.g. $1 = 0$). Every other line will follow from previous ones, but since a false statement implies any statement (true or false), one can arrive at any conclusion. So if the verification procedure does not examine the offending line, the verification procedure is liable to accept an obviously incorrect proof.

In this sense, traditional proofs are not *robust*: the slightest changes can convert a correct proof to an incorrect proof, or an incorrect proof to a (nearly) correct proof. The PCP Theorem gives a way of making proofs and proof checking robust: proofs can be written in a format such that small changes to a proof will not drastically alter its correctness.

3 Hardness of Approximation

In this section, we will discuss a seemingly unrelated topic, approximation problems, hardness of approximation, and see how it is closely related to the notion of probabilistic proof checking.

First, let us informally review what an optimization problem is: there is an objective function $f : U \to \mathbb{R}$ over some universe U that one would like to, say, maximize. Many important problems can be easily described using this description. For example, the problem of deciding `3SAT` can be described as determining whether, given a 3SAT formula φ, all of it φ's clauses can be satisfied by an assignment to φ's variables. Here, U is the set of all variable assignments, and f counts the number of satisfied clauses by the input assignment. Usually, the optimum of an objective function f on an instance I of an optimization problem is denoted $\mathrm{OPT}(I)$.

In an *approximation* problem, on the other hand, one is willing to settle for an approximation of OPT. An optimization problem is α-*approximable* if there is an algorithm A such that, given instance I of the optimization problem, $A(I) \geq \alpha \cdot \mathrm{OPT}(I)$, assuming that the optimization problem is one of maximization. Usually, one cares about approximation algorithms that run in polynomial time, as many optimization problems are **NP**-hard to solve exactly.

As we will see by the end of this lecture, the PCP Theorem implies that there are many optimization

problems that are **NP**-hard even to approximate.

As an example, we will discuss the MAX-3SAT problem:

Definition 3. *MAX-3SAT is the problem of finding, given a 3CNF Boolean formula φ as input, an assignment that maximizes the number of satisfied clauses.*

This problem is **NP**-hard, because the corresponding decision problem, 3SAT, is **NP**-complete. We define an approximation algorithm for MAX-3SAT in the following way.

Definition 4. *For every 3CNF formula φ, the value of φ, denoted by $\mathtt{val}(\varphi)$, is the maximum fraction of clauses that can be satisfied by any assignment to φ's variables. In particular, φ is satisfiable iff $\mathtt{val}(\varphi) = 1$.*

For every $\rho \leq 1$, an algorithm A is a ρ-approximation algorithm (which may be randomized) for MAX-3SAT if for every 3CNF formula φ with m clauses, $A(\varphi)$ outputs an assignment satisfying at least $\rho \cdot \mathtt{val}(\varphi)m$ of φ's clauses.

Basically, a algorithm A is a ρ-approximation for MAX3SAT if $\forall \varphi, \mathtt{val}_A(\varphi) \geq \rho \cdot opt(\varphi)$.

3.1 $\frac{7}{8}$-approximation for MAX-3SAT

There is a very simple algorithm that can approximate MAX-3SAT to within 7/8: simply choose an assignment to the variables uniformly at random, and output that assignment. In expectation, the fraction of satisfied clauses will be 7/8's. Let x denote the randomly chosen assignment. For any clause, we have $Pr_x[(x_i \vee \bar{x}_j \vee x_k) = 1] = \frac{7}{8}$. So, we know E[number of clauses satisfied] $= \frac{7}{8} \cdot$ number of clauses $\geq \frac{7}{8} \cdot \mathrm{OPT}(\varphi)$.

3.2 $\mathtt{Gap3SAT}_{1,\alpha}$

Although the PCP Theorem implies hardness of approximation for optimization problems, it will be more convenient to reframe optimization problems as decision problems, because then we can formally show that such problems are **NP**-hard and thus imply the original approximation problem is also computationally intractable (modulo $\mathbf{P} \neq \mathbf{NP}$).

Definition 5. *$\mathtt{Gap3SAT}_{1,\alpha}$ is the promise problem where, on input φ, decide whether*

1. $\mathrm{OPT}(\varphi) = 1$, *or*
2. $\mathrm{OPT}(\varphi) < \alpha$,

promised that one is the case.

Case (1) is usually referred to as the "YES" case, or the "Completeness" case, and case (2) is usually referred to as the "NO" case, or the "Soundness" case. The following theorem demonstrates an equivalence between the optimization formulation of 3SAT and the gap formulation.

Theorem 6. *Suppose A is a polynomial time α-approximation algorithm for* `MAX3SAT`*. Then,* $\texttt{Gap3SAT}_{1,\alpha}$ *is solvable in polynomial time.*

Proof. Suppose A is an α-approximation algorithm for `MAX-3SAT`. Given $\varphi \in \texttt{Gap3SAT}_{1,\alpha}$, we run A on φ. We know that $opt(\varphi) \geq A(\varphi) \geq \alpha \cdot opt(\varphi)$. Consider the following two cases:

1. (case 1) OPT $= 1$, $A(\varphi) \geq \alpha$
2. (case 2) OPT $< \alpha$, $A(\varphi) < \alpha$.

To distinguish, we simply output whether $A(\varphi) \geq \alpha$ or not. □

The following theorem expresses the equivalence between hardness of approximation and the PCP Theorem.

Theorem 7. $\texttt{Gap3SAT}_{1,\alpha}$ *is* ***NP****-complete for some* $\alpha < 1 \iff$ *The* ***PCP*** *Theorem is true.*

Proof. ($\Longrightarrow$) Suppose $\texttt{Gap3SAT}_{1,\alpha}$ is **NP**-hard for some $\alpha < 1$. This implies that $\forall L \in \mathbf{NP}$, $\exists$ a polynomial reduction to $\texttt{Gap3SAT}_{1,\alpha}$ such that all x gets mapped to an input φ_x where

1. $x \in L \Rightarrow \text{OPT}(\varphi_x) = 1$ (Yes Case)
2. $x \notin L \Rightarrow \text{OPT}(\varphi_x) < \alpha$ (No Case).

The PCP system for L will be as follows: on input x, the verifier will perform a polynomial-time transformation from x to φ_x. Then, the verifier will perform the following check on the given proof y:

1. The verifier samples a random clause C of φ_x.
2. The verifier then tests if y satisfies the clause C. If so, then accept. Otherwise, reject.

Suppose $x \in L$. Then $\text{OPT}(\varphi_x) = 1$, so $\exists y$ such that $Pr[V(x, y) = 1] = 1$. Suppose that $x \notin L$. Then $\text{OPT}(\varphi_x) < \alpha$, so $\forall y$, $Pr[V(x, y) = 1] < \alpha$, by definition of $\texttt{Gap3SAT}_{1,\alpha}$. This shows that $L \in \mathbf{PCP}_{\{0,1\}}(O(1), O(\log n))$. Although the definition of **PCP** above required $\alpha < 5/6$, any constant suffices. We have shown that any problem in **NP** admits a PCP system.

($\Longleftarrow$) Here, we assume that the **PCP** Theorem is true, and want to show that $\texttt{Gap3SAT}_{1,\alpha}$ is **NP**-hard for some $\alpha < 1$. For simplicity, let us assume that the PCP system for `3-coloring` performs 3 queries. If it performs some larger number of queries q, then we would actually prove a hardness result for `Gap`q`SAT` instead. We know that `3-coloring` $\in \mathbf{NP}$, so `3-coloring` $\in \mathbf{PCP}_{\{0,1\}}\{O(1), O(\log n)\}$ by assumption.

We show how to reduce `3-coloring` to `Gap3SAT`. There is some associated proof string for the PCP system for `3-coloring`, call it $y \in \{0,1\}^{\text{poly}\{n\}}$. Suppose an input to the `3-coloring` problem were x. Then, for every choice of randomness in $\{0,1\}^{r(n)}$, the PCP verifier V will query 3 bits of the proof y and perform a test, accepting or rejecting based on the values of those 3 bits. Suppose for some setting of randomness z, the verifier queries bits $y_{z_1}, y_{z_2}, y_{z_3}$, and accepts or rejects based on

a predicate $C_z(y_{z_1}, y_{z_2}, y_{z_3})$. Observe that C_z can be expressed as a constant-sized 3SAT formula in terms of y_{z_1}, y_{z_2}, and y_{z_3}. Let the conjunction of all these 3SAT formulas be $\varphi(x)$.

Suppose $x \in L$. Then by definition, all of the checks of the verifier V pass, so $\mathrm{OPT}(\varphi(x)) = 1$. Suppose $x \notin L$. Then, less than α fraction of the checks pass, and since each clause of $\varphi(x)$ corresponds to a check, $\mathrm{OPT}(\varphi(x)) < \alpha$. Thus, we have successfully reduced `3-coloring` to $\texttt{Gap3SAT}_{1,\alpha}$. □

Next time, we will start the proof of the PCP Theorem.

References

[1] S. Arora and B. Barak, *Computational Complexity - A Modern Approach*, Cambridge University Press, ISBN 978-0-521-42426-4.

[2] L. Babai, L. Fortnow, L.A.Levin, and M. Szegedy *Checking computations in polylogarithmic time*, In *STOC*, pages 21-32. ACM, 1991.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy *Proof of verification and the hardness of approximation problems*, In *J. ACM*, 45(3):501-555, 1998. Prelim version FOCS '92.

[4] S. Arora and S. Safra *Probabilistic checking of proofs: A new characterization of NP*, In *J. ACM*, 45(1):70-122, 1998. Prelim version FOCS '92.