CMSC 39600: PCPs, codes and inapproximability 25 Sep, 2007

Lecture 1: The PCP Theorem – Introduction and two views

Lecturer: Prahladh Harsha *Scribe: Joshua A. Grochow*

In today's lecture, we will introduce the PCP Theorem from two different points of view – (1) hardness of approximation, and (2) proof verification and then prove the equivalence of both the versions of the PCP Theorem.

1.1 Hardness of Approximation

One of the goals of classical computational complexity – perhaps even *the* goal – is to gain a better understanding of the difficulty of various *computational problems*, such as:

- $MAXCLIQUE$: Given a graph G, output the vertices in its largest clique.
- 3CNF satisfiability: Given a 3CNF Boolean formula, output a satisfying assignment if one exists. A related problem is that of $MAX3SAT$: Given a 3CNF Boolean formula, output an assignment which satisfies the maximum number of clauses.
- 3-COLOR: Given a graph G, color the vertices with 3 colors such that no edge is monochromatic if such a coloring exists.
- SET COVERING PROBLEM: Given a collection of sets $S_1, S_2, \ldots, S_m$ that cover a universe $U = \{1, 2, \ldots, n\}$, find the smallest sub-collection of sets $S_{i_1}, S_{i_2}, \ldots$ that also cover the universe
- ...

Rather than studying these problems directly, it is often convenient to study polynomially equivalent *decision problems*, i.e. computational problems whose output is always a simple "yes" or "no." For example, the decision problem equivalents of the above problems are:

- $CLIQUE = \{\langle G, k\rangle | \text{graph } G \text{ has a clique of size } \geq k\}$
- $3SAT = \{\varphi | \varphi \text{ is satisfiable}\}$
- SET-COVER $= \{\langle U; \{S_1, \ldots, S_m\}, k\rangle | \exists 1 \leq i_1 \leq i_2 \leq \ldots, i_k \leq m, \text{ such that } \bigcup_{j=1}^{m} S_{i_j} = U\}$
- ...

It is more convenient to work with the decision equivalents as they are simpler though they are polynomially equivalent to the original computational problems. The theory of NP-completeness discusses the hardness of these computational problems by studying the hardness of the equivalent decision problems.

Similarly, when we study approximation algorithms for computational problems, it is often useful to study a restricted set of polynomially equivalent problems. First, what is an approximation algorithm?

Definition 1.1. *An algorithm A is said to be an α-*approximation algorithm *for a (maximization) computational problem Π and some $0 < \alpha < 1$ if for all instances x of the problem Π, $\alpha \cdot OPT(x) \leq A(x) \leq OPT(x)$, where $OPT(x)$ denotes the optimum value associated with the instance x. (For the case of minimization problems, the inequalities get reversed, i.e., $OPT(x) \leq A(x)\alpha^{-1} \cdot OPT(x)$)*

For example, an α-approximation algorithm for the clique problem would output (the size of) a clique that was at least α times as large as the largest clique in the input graph. For another example, note that a random assignment will satisfy 7/8 of the clauses in any 3CNF formula, so a randomized 7/8-approximation to $MAX3SAT$ can simply output a random assignment. (this particular result can be derandomized using the method of conditional expectations.)

We would like to study the hardness of these approximation problems. As in the case of computational problems, it would be nice if we could capture the hardness of the approximation problems via decision problems. The analogue of decision problems for approximation algorithms are known as *gap problems,* and these are exactly the problems that are used to study the hardness of approximation. Whereas a decision problem specifies a set of "yes" instances – and thus implicitly specifies that all other instances are "no" instances – a gap problem explicitly specifies both the "yes" instances YES and the "no" instances NO. Obviously we require $YES \cap NO = \emptyset$, but – unlike in decision problems – we do not require that $YES \cup NO$ covers all instances of the problem.

For example, gap-3SAT$_\alpha$ (for $\alpha \leq 1$) is the gap problem whose (YES, NO) are defined as follows:

$$\begin{aligned} YES &= \{\langle\varphi, k\rangle | \text{there is an assignment satisfying } \geq k \text{ clauses of } \varphi\} \\ NO &= \{\langle\varphi, k\rangle | \text{every assignment satisfies } \leq \alpha k \text{ clauses of } \varphi\} \end{aligned}$$

where φ is a 3CNF formula and k any positive integer.

Any instance of the problem which is not specified as either YES or NO is a "don't care" instance: that is, an algorithm solves a gap problem (YES, NO) if it outputs "yes" on all $x \in YES$ and "no" on all $x \in NO$, and we don't care what the algorithm says on other instances x. Thus, if an algorithm solves a gap problem and outputs "yes" on input x, all we can conclude in general is that $x \notin NO$, since x might be a "don't care" instance.

As promised, approximation problems are polynomially equivalent to gap problems. We show this in the case of $MAX3SAT$ below.

Proposition 1.2. *α-approximating $MAX3SAT$ is polynomially equivalent to solving gap-3SAT$_\alpha$.*

Proof. ($\Rightarrow$) Suppose there is an α-approximation algorithm A to $MAX3SAT$. Then, consider the following algorithm B for gap-3SAT$_\alpha$.

```
$B$ : "On input $\langle\varphi, k\rangle$
    1. Run A on $\varphi$ and let $k' = A(\varphi)$.
    2. Accept iff $k' \geq \alpha k$.                    "
```

B solves gap-3SAT$_\alpha$: For if $k' \geq \alpha k$, then there must be some assignment satisfying at least αk clauses, so $\varphi \notin NO$ and the algorithm outputs "yes." Conversely, if $\alpha k > k'$, then since $k' \geq \alpha OPT(\varphi)$ it is the case that $k > OPT(\varphi)$, so there is no assignment satisfying at least k clauses. Thus $\varphi \notin YES$ and the algorithm outputs "no."

($\Leftarrow$) Suppose instead there is an algorithm B that solves gap-3SAT$_\alpha$. Then

```
$A$ : "On input $\varphi$
    1. Let $m$ be the number of clauses of $\varphi$.
    2. Run $B$ on $\langle\varphi,1\rangle, \langle\varphi,2\rangle, \langle\varphi,3\rangle, \ldots, \langle\varphi,m\rangle$.
    3. Let the largest $k$ such that $B$ accepted $\langle\varphi, k\rangle$
    4. Output $\alpha k$                                  "
```

is an α-approximation to $MAX3SAT$. For if B rejects $\langle\varphi, k+1\rangle$, we know it is *not* a YES instance, so φ cannot have any assignment satisfying strictly more than k clauses, i.e. $k \geq OPT(\varphi)$, or, multiplying both sides by α, $\alpha k \geq \alpha OPT(\varphi)$. But since B accepted $\langle\varphi, k\rangle$, there must be some assignment to φ satisfying at least αk clauses, i.e. $OPT(x) \geq \alpha k$. Thus $OPT(x) \geq \alpha k \geq \alpha OPT(x)$. $\square$

Although gap-3SAT$_\alpha$ is a nice example, it will be convenient for us to use what we call gap-3SAT$'_\alpha$, defined as follows:

$$\begin{aligned} YES &= \{\varphi | \varphi \in 3SAT\} \\ NO &= \{\varphi | \text{all assignments satisfy } \leq \alpha m \text{ clauses}\} \end{aligned}$$

(where, as above, φ is a 3CNF formula with m clauses). gap-3SAT$'_\alpha$ is identical to gap-3SAT$_\alpha$ but for the fact that the second argument k of the instance $\langle \varphi, k \rangle$ of gap-3SAT$_\alpha$ is forced to be m, the number of clauses.

We can now state the PCP Theorem, from the perspective of hardness of approximation as follows

PCP Theorem 1. *There exists $0 < \alpha < 1$ such that the 3-coloring problem 3-COLOR Karp-reduces to gap-3SAT$'_\alpha$, i.e. there is some polynomial time reduction $R : \{graphs\} \to \{3CNF\} \times \mathbb{N}$, such that*

$$\begin{aligned} G \in \textit{3-COLOR} &\Rightarrow R(G) = \langle \varphi, k \rangle \in YES \\ G \notin \textit{3-COLOR} &\Rightarrow R(G) = \langle \varphi, k \rangle \in NO. \end{aligned}$$

Corollary 1.3. *There exists $0 < \alpha < 1$ such that gap-3SAT$'_\alpha$ is NP-hard and so is gap-3SAT$_\alpha$*

Corollary 1.4. *α-approximating $MAX3SAT$ is NP-hard.*

1.2 Proof Verification

A *proof system* consists of a verifier V and prover P. Given a statement x, such as "φ is satisfiable" or "G is 3-colorable," P produces a candidate proof π for the statement φ. The verifier V then reads the statement-proof pair (φ, π) and either accepts or rejects the proof π for φ. We require two properties of any proof system proof system:

Completeness Every true statement has a proof. In other words

$$\text{if } x \text{ is true, then } \exists \pi \text{ such that } V(x, \pi) \text{accepts.}$$

Soundness A false statement does not have a proof. In other words,

$$\text{if } x \text{ is false, then } \forall \pi,\ V(x, \pi) \text{ rejects.}$$

Variations of this general definition are (can be) used to define many complexity classes of importance, for instance:

Definition 1.5. *A language L is in NP if there is a deterministic polynomial time verifier V and a polynomial p such that*

1. *(Completeness) $(\forall x \in L)(\exists \pi)(|\pi| = p(|x|)$ and $V(x, \pi)$ accepts)*
2. *(Soundness) $(\forall x \notin L)(\forall \pi(|\pi| \leq p(|x|) \Rightarrow V(x, \pi)$ rejects)*

We can then ask the question: what happens if we allow V to be randomized? Historically, this led to the development of the classes AM (Arthur-Merlin), IP (interactive proofs), which further led to the classes ZK (zero-knowledge proofs), MIP (multi-prover interactive proofs), , and eventually PCPs. For a nice writeup of this history, see [Bab90] and [O'D05]. The original proof of the PCP theorem in fact goes through almost all of these: $IP, AM, MIP, \ldots$, but in this course we will not follow this route, instead we will give Dinur's proof of the PCP Theorem [Din07].

We now state the PCP theorem in the language of proof verification

PCP Theorem 2. *Every $L \in NP$ has a probabilistically checkable proof (PCP) where the verifier V tosses $\leq C_L \log n$ coins, probes $\leq Q$ locations of the proof, and*

1. $x \in L \Rightarrow \exists^p \pi Pr[V^\pi(x) \text{ accepts}] = 1$
2. $x \notin L \Rightarrow \forall^p \pi Pr[V^\pi(x) \text{ accepts}] \leq 1/2$

(where $\exists^p \pi$ and $\forall^p \pi$ means "there exists a π of length polynomial in $|x|$" and "for all π of length at most $p(|x|)$"). Here, C_L is a constant dependent on L and Q an universal constant.

The above theorem is succinctly written using the notation (which we will formally define in the next lecture) a $L \in PCP_{1,1/2}[C_L \log n, Q]$. Thus, the theorem states that $\exists q$ such that $NP \subseteq \bigcup_{c \geq 0} PCP_{1,1/2}[c \log n, q]$. Note that with only $\log n$ coins, the verifier cannot even select random indices in a proof that is more than polynomially long, so the restriction to polynomially long proofs is redundant. A similar result holds for NEXP in which the verifier tosses polynomially many random coins (or using the notation above, $NEXP \subseteq PCP_{1,1/2}[\text{poly}(n), q]$. In fact, this is one of the results that led to the PCP Theorem. The constants Q and soundness parameter $1/2$ can be traded off against each other. In fact, the constant Q can be made as small as 3 while maintaining the soundness parameter to by any constant greater than $1/2$.

1.3 Equivalence of the two versions of the PCP Theorem

Our final topic will be to show that the PCP theorem, stated in two different languages, one – the hardness of approximation viewpoint and the proof checking viewpoint are equivalent.

Lemma 1.6. *PCP Theorem 1 $\Leftrightarrow$ PCP Theorem 2.*

Proof. (PCP Theorem 1 $\Rightarrow$ PCP Theorem 2) Suppose there is a reduction R from 3-COLOR to gap-3SAT$'_\alpha$ as stated in PCP Theorem 1. The probabilistic proof system as stated in the PCP Theorem 2 is then obtained using R as follows: Both the verifier and the prover apply the reduction R to transform the input graph G to a formula φ. The proof π is an assignment for φ. The verifier chooses a random clause of φ, say $(x_{i_1} \vee x_{i_1} \vee x_{i_3})$ (where the x's represent variables or their negations). Then the verifier probes the proof π for the values of these three variables, and accepts if and only if the clause is satisfied. Since R is a reduction from 3-COLOR to gap-3SAT$'_\alpha$, it follows that this proof system satisfies the properties stated in the PCP Theorem 2, thus proving 3-COLOR $\in PCP_{1,\alpha}[O(\log n), 3]$.

(PCP Theorem 2 $\Rightarrow$ PCP Theorem 1) Given the PCP Theorem 2, we wish to Karp-reduce 3-COLOR to gap-3SAT$'_\alpha$. The basic idea is to encode the verifier's possible actions by a Boolean formula. For each random string R, the verifier's action is a q-ary Boolean function h_R (where q is the number of bits of the proof probed by the verifier). We will now use the following fact, which we state without proof.

Fact 1.7. *For every q, there exists $\ell(q), k(q)$ such that any q-ary Boolean function h can be encoded by a 3CNF formula φ_h with $k(q)$ clauses over $q + \ell(q)$ variables $x_1, \ldots, x_q, z_1, \ldots, z_{\ell(q)}$ such that*

$$\begin{aligned} h(x) = 1 \quad &\Rightarrow \quad \exists z, \varphi_h(x, z) = 1 \\ h(x) = 0 \quad &\Rightarrow \quad \forall z, \varphi_h(x, z) = 0 \end{aligned}$$

The variables $z_1, \ldots, z_{l(q)}$ are called extension variables.

This fact, essentially follows from the Cook-Levin Theorem, that shows the NP-completeness of 3SAT.

Thus, given a verifier V, we construct the formula

$$\varphi = \bigwedge_{\text{coins } R} \varphi_{h_R}.$$

Let $M = 2^R k(q)$ be the number of clauses in φ. If $G \in$ 3-COLOR, there exists some proof π such that V accepts for all random coins R or equivalently $h_R(\pi) = 1$ for all R. Hence, we can set the value of all the extension variables such that $\varphi_{h_R}(\pi, z) = 1$ for all R. Hence, φ is satisfiable. On the other hand, if $G \notin$ 3-COLOR, then at most half of the choices of R cause the verifier to accept or equivalently for all π, for at least half the number of random coins R, $h_R(\pi) = 0$. It then follows from Fact 1.7 that for all assignments π, z, at least half of the $\varphi(h_R)$ are not satisfied. For each $\varphi(h_R)$ not satisfied, at least one clause of it must be satisfied, so at most $k(q) - 1$ clauses of it can be satisfied. Thus the total number of clauses of φ satisfies is at most $\frac{M}{2} + \frac{M}{2}(1 - \frac{1}{k}) = M(1 - \frac{1}{2k})$. Thus, 3-COLOR Karp-reduces to gap-3SAT$'_\alpha$ for $\alpha = 1 - 1/2k$. □

References

[Bab90] László Babai. *E-mail and the unexpected power of interaction.* In *Proc. 5th IEEE Conference on Structure in Complexity Theory*, pages 30–44. Barcelona, Spain, 8–11 July 1990. doi:10.1109/SCT.1990.113952.

[Din07] Irit Dinur. *The PCP theorem by gap amplification.* J. ACM, 54(3):12, 2007. (Preliminary Version in *38th STOC*, 2006). doi:10.1145/1236457.1236459.

[O'D05] Ryan O'Donnell. *A history of the pcp theorem*, 2005. Available from: `http://www.cs.washington.edu/education/courses/533/05au/pcp-history.pdf`.