CS221: Computational Complexity Prof. Salil Vadhan

Lecture 34: More Hardness of Approximation,
$\mathbf{NP} \subseteq \mathbf{PCP}[\text{polylog}, \text{polylog}]$

12/16 *Scribe: Vitaly Feldman*

Contents

Additional Readings. Papadimitriou, Ch. 13 has a detailed treatment of approximability, with less emphasis on the PCP theorem (which is stated in different language in Thm 13.12).

1 More Tight Hardness of Approximation Results

Last time we saw the tight results for hardness of approximation of MAX3SAT and E3LIN2. Today we will survey some other tight hardness of approximation results.

Definition 1 MAXCLIQUE — *given a graph $G = (V, E)$ find the largest clique.*

This problem is completely equivalent to finding the maximum independent set. Best known approximation algorithm gives $\frac{n}{\log^2 n}$-approximation (which is not much better than the trivial n-approximation).

Theorem 2 (Håstad) *It is* **NP***-hard to approximate* MAXCLIQUE *within $n^{1-\epsilon}$ factor for any positive ϵ.*

The result is proven by tailoring the PCP Theorem for this specific problem (namely, optimizing the so-called "amortized free bit complexity" of the PCP).

Definition 3 SET COVER *problem: given a collection of sets $S_1, \ldots, S_m$ which are subsets of the set $\{1, \ldots, n\}$, find the smallest subcollection of sets $S_{i_1}, \ldots, S_{i_t}$ whose union is equal to $\{1, \ldots, n\}$ (that is, they cover the "universe").*

The greedy algorithm gives $H(n) \approx \ln n$ approximation to this problem ($H(n)$ is the n-th harmonic number).

Theorem 4 (Feige) *It is* **NP***-hard to approximate* SET COVER *to within* $(1-\epsilon)\ln n$ *factor for any positive* ϵ.

These results show that **NP**-complete problems, while equivalent as languages, exhibit wide variety of properties with respect to approximation of their optimization versions. In particular, the best achievable approximation factors for various problems ranged from constant (MAX3SAT, E3LIN2), to logarithmic (SET COVER), to polynomial (MAXCLIQUE).

2 Open Problems

Besides the tight results that we saw, there are several problems for which the current results are not tight. We will survey some of the most important ones.

Definition 5 VERTEX COVER*: given a graph G find the smallest set of vertices that cover all the edges.*

There is a long known algorithm for this problem that approximates the solution within a factor of 2. On the other hand, the best known hardness of approximation result for this problem is for ≈ 1.36 (in a recent paper by Dinur and Safra [DS02]).

Definition 6 GRAPH COLORING*: given a graph G, find a coloring of the vertices of this graph that uses the minimum number of colors.*

Håstad's clique paper also shows that this problem is hard to approximate to within $n^{1-\epsilon}$. However, a more interesting problem is that of trying to color a graph which is promised to be 3-colorable. There is a very nontrivial $n^{\frac{3}{14}}$-approximation algorithm for this problem. On the other hand the best known hardness of approximation factor is $\frac{4}{3}$. (This result does not use the PCP theorem.)

Definition 7 SHORTEST VECTOR *problem: given a lattice in $\mathbb{R}^n$ (specified by a basis consisting of n linearly independent vectors), find the shortest nonzero vector in the lattice. (Similarly, there is the* CLOSEST VECTOR *problem, where you are also given a target vector, and the problem is to find the lattice vector closest to the target.)*

These problems are of particular significance in cryptography (partly because of a worst-case/average-case equivalence established for a variant of approximating SHORTEST VECTOR). The best known polynomial-time approximation algorithm for these problems achieves a $\approx 2^{n/\log n}$ approximation factor, but the best known hardness of approximation results are $\sqrt{2}-\epsilon$ and $n^{O(1/\mathrm{loglog} n)}$ respectively.

3 Sketch of the Proof of the PCP Theorem

We are now going to show some of the ideas that are involved in the proof the PCP theorem. We will "almost" show the following result which was proved independently by Babai *et al.* [BFLS91] and Feige *et al.* [FGLSS91].

Theorem 8 **NP** $\subseteq$ **PCP**(polylog n, polylog n).

We show a PCP proof with the above parameters for 3-SAT. Let $\phi(x_1, \ldots, x_n)$ be a 3-CNF. The main idea of our approach is to arithmetize ϕ and use ideas from the proof that **IP** = **PSPACE** (or #SAT$\in$ **IP** that we saw in class). The arithmetization used for this proof will be different though.

Arithmetization. An assignment to the boolean variables is simply a function A from $[n]$ to $\{0,1\}$. Alternatively, we can view it as a function $A : \{0,1\}^{\log n} \to \{0,1\}$. Similarly we can represent a formula ϕ as a function $\phi : \{0,1\}^{3\log n} \times \{0,1\}^3 \to \{0,1\}$. This function is a characteristic function for every clause, that is,

$$\phi(i_1, i_2, i_3, b_1, b_2, b_3) = \begin{cases} 1 & \text{if clause } (x_{i_1} = b_1) \vee (x_{i_2} = b_2) \vee (x_{i_3} = b_3) \text{ is present in } \phi \\ 0 & \text{otherwise} \end{cases}$$

With these definition the statement "A is a satisfying assignment to ϕ" is equivalent to

$$\forall i_1, i_2, i_3, b_1, b_2, b_3,\ \phi(i_1, i_2, i_3, b_1, b_2, b_3)(A(i_1) - b_1)(A(i_2) - b_2)(A(i_3) - b_3) = 0 \tag{1}$$

We now extend A and ϕ to *multilinear* functions $\hat{A} : \mathbb{F}^{logn} \to \mathbb{F}$ and $\hat{\phi} : \mathbb{F}^{3logn+3} \to \mathbb{F}$ over a field $\mathbb{F} = \mathbb{Z}_p$ for $p \geq 8n^3$. As we know, $\hat{A}$ and $\hat{\phi}$ are low degree polynomials (of total degree $\log n$ and $3\log n + 3$ respectively). We set $m = 3\log n + 3$ and define $P : \mathbb{F}^m \to \mathbb{F}$ by

$$P(i_1, i_2, i_3, b_1, b_2, b_3) = \hat{\phi}(i_1, i_2, i_3, b_1, b_2, b_3)(\hat{A}(i_1) - b_1)^2(\hat{A}(i_2) - b_2)^2(\hat{A}(i_3) - b_3)^2$$

(note that in fact P takes m field elements and not only Boolean values and in particular, i's are sequences of $\log n$ field elements). We want to prove that $P|_{\{0,1\}^m} \equiv 0$, or in other words

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_m \in \{0,1\}} P(x_1, \ldots, x_m) = 0 \tag{2}$$

The Sum-Check Protocol. We now build the PCP in a way that allows us to imitate the proof that **IP** = **PSPACE**. To maintain the analogy we say that the PCP sends some information to the verifier while this information is actually written in some place in the proof known to the verifier. The first stage of the protocol is the following

Verifier		PCP
Check $p_1(0) + p_1(1) = 0$	$\xleftarrow{p_1(x)}$	$p_1(x) = \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_m \in \{0,1\}} P(x, x_2, \ldots, x_m)$

To verify that that $p_1(x)$ is correct the verifier then chooses a random α_1 and "sends" (α_1 addresses a particular portion of the proof that will be read) it to the prover and asks him to prove that $p_1(\alpha_1) = \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_m \in \{0,1\}} P(\alpha_1, x_2, \ldots, x_m)$. This is done by proceeding recursively as in the first step, namely

Check $p_2(0) + p_2(1) = p_1(\alpha_1)$	$\xleftarrow{p_2(x)}$ $\xrightarrow{\alpha_2}$	$p_2(x) = \sum_{x_3 \in \{0,1\}} \cdots \sum_{x_m \in \{0,1\}} P(\alpha_1, x, x_3, \ldots, x_m)$
	$\vdots$	
Check $p_m(0) + p_m(1) = p_{m-1}(\alpha)$	$\xleftarrow{p_m(x)}$	$p_m(x) = P(\alpha_1, \ldots, \alpha_{m-1}, x)$

And finally, the verifier chooses a random α_m and verifies that $p_m(\alpha_m) = P(\alpha_1, \ldots, \alpha_m)$. In order to do this, the verifier needs to evaluate P, which in turn requires 3 evaluations of $\hat{A}$. This can be done by making the complete description of $\hat{A}$ part of the PCP.

The parameters. Even though this protocol will not work as is, let us examine the parameters of this protocol. The verifier needs to choose m random α_i's. By our definitions, $m \log |\mathbb{F}| = O(\log n \log \log n)$. Thus the randomness complexity is polylogarithmic, as desired. For the query complexity, we need to specify the coefficient of the p_i's. The degree of each p_i is bounded by a degree of P in variable number i, which, as can be easily seen from the definition of P, is $O(\log n)$. Each coefficient is just a field element. Three more field elements are required for the evaluations of $\hat{A}$. The complete computation yields that the total number of bits read is $O(\log^2 n \log \log n)$. Thus the algorithm achieves the parameters of the theorem but does not actually work ...

The problem. The problem with this protocol is that its soundness relies on the fact that the polynomial P for which we do the sum check protocol is a low degree polynomial (in the **IP** = **PSPACE** protocol the verifier knew the polynomial itself while here it depends on $\hat{A}$ which is given to the verifier as a table in the PCP). Unfortunately, the verifier cannot know whether the table for $\hat{A}$ is actually a multilinear (or at least low-degree) polynomial. This allows the dishonest prover to use any function as $\hat{A}$ and thus to cheat the verifier in the last step.[1] This gives rise to the following problem which is exactly the missing component.

Definition 9 Low degree testing*: given an oracle access to an arbitrary function $f : \mathbb{F}^m \to \mathbb{F}$ check whether f is a low-degree polynomial and do so using only a "small" number of queries to the oracle.*

We will discuss this problem in the next lecture.

References

[BFLS91] Laszlo Babai, Lance Fortnow, Leonid Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. STOC 1991, pp. 21–31.

[DS02] Irit Dinur, Shmuel Safra. The importance of being biased. STOC 2002, pp: 33–42.

[FGLSS91] Uriel Feige, Shafi Goldwasser, Laszlo Lovasz, Shmuel Safra, Mario Szegedy. Approximating Clique is Almost **NP**-Complete. FOCS 1991, pp. 2–12

[1] Another problem is that the prover may write a low-degree polynomial for $\hat{A}$, but one which does not have $\{0,1\}$ values on $\{0,1\}^m$, making it possible that the sum-check Equation (2) holds even though the Equations (1) do not. A fix for this is to instead have the prover prove that a "random" linear combination of $\{P(x) : x \in \{0,1\}^m\}$ is zero. (The coefficients are chosen by the verifier in a way that uses few random bits and guarantees that the sum will be nonzero with high probability if any of the $P(x)$'s are nonzero.)