Approximation Algorithms and Hardness of Approximation April 23, 2013

Lecture 16

Lecturer: Ola Svensson *Scribes: Farah Charab*

In this lecture, we will describe the PCP (*probabilistically checkable proofs*) theorem([1],[2], a simplified proof can be found in [4]), an important discovery in the field of complexity theory dating back to 1992. As it will be discussed through out this lecture, there are two ways to view the PCP theorem. One point of view of the PCP theorem is that it is a hardness of approximation result, i.e the PCP theorem can be used to show that for many NP-complete problems, obtaining an approximate solution is as hard as getting an exact solution. Yet an alternative view of the PCP theorem is that it constructs locally testable proof systems. That it is, the PCP theorem gives a way to transform every mathematical proof into a form that is checkable by only looking at very few,probabilistically chosen symbols of the proof.

1 Point of View 1 : Hardness of Approximation

To prove the hardness of a problem P, we can show that if we can solve P using a polynomial time algorithm, and if we have a polynomial time reduction from an NP-hard problem H, then we can solve H in polynomial time. However, assuming P≠NP, this should not be possible given that H is an NP-hard problem. We will demonstrate this using the following example.

1.1 Hardness of TSP

The travelling salesman problem (TSP) addresses the following question: Given a list of n cities with pairwise distances $d(i,j)$ for all $i,j \in n$, find the shortest tour. To prove the hardness of TSP, we will reduce the problem of Hamiltonicity(HAM), an NP-complete problem, to TSP. The problem HAM addresses is the following: Given a graph, does the graph have a tour which visits every vertex exactly once? To reduce HAM to TSP, we do the following reduction. Given an instance $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to HAM, we produce an instance $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \mathbf{d})$ TSP with $|\mathcal{V}|$ cities.

$$d(i,j) = \begin{cases} 1 & \text{if } e \in \mathcal{E} \\ M & \text{o.w} \end{cases}$$

To prove the hardness of TSP, we need to show that indeed given such a reduction, solving TSP will allow us to determine whether the graph G had a Hamiltonian cycle:

- **Completeness:** If $\mathcal{G}$ was Hamiltonian, then $\exists$ a tour in $\mathcal{G}'$ of value $\leq n$.
- **Soundness:** If $\mathcal{G}$ wasn't Hamiltonian, then any tour in $\mathcal{G}'$ has value $\geq M + n - 1$.

Figure 1 shows on the left the division of the Hamiltonian instances to satisfiable and unsatisfiable instances. After the aforementioned reduction, all the "Yes" instances of the Hamiltonian problem are mapped to instances in the TSP problem with value at most n. However, all the "No" instances in the Hamiltonian are mapped to instances in TSP with value greater than $M + n$, hence introducing a gap if we set M to be e.g. n^2(in fact we can set it to be any number whose description is polynomial in n). Furthermore, from this reduction, we obtain the following corollary.

Corollary 1 *It is NP-hard to approximate TSP within* $\frac{M}{n}$

To see that, assume there is an algorithm which approximates TSP within a factor better than $\frac{M}{n}$; then, using this algorithm, we would be able to distinguish between the two groups of instances the above reduction would produce, and we would be able to decide HAM.

This notion gives us the first way to look at the PCP theorem: it is a way to produce gap-introducing reductions, and prove that certain NP-hard problems are actually hard to approximate within some degree.

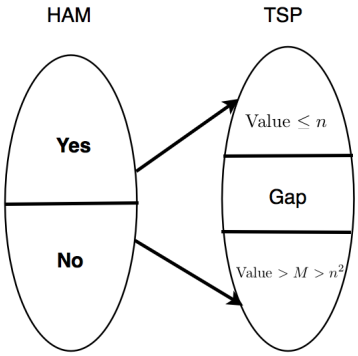

Figure 1: Gap-Introducing Reduction

Theorem 2 ***PCP Theorem:*** *There exist $\epsilon > 0$ such that for every instance I of 3-SAT , there is a polynomial-time reduction that reduces I to another instance I' such that the following hold:*

1. *I' is satisfiable iff I is satisfiable.*
2. *Any assignment to I' satisfy at most $(1 - \epsilon)$ fraction of the clauses if I is not satisfiable.*

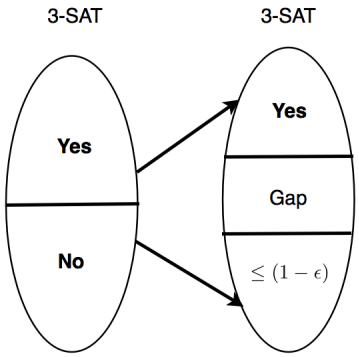

Figure 2: PCP Theorem

It is interesting to see that the classic Cook-Levin Theorem can be rephrased to be a gap-introducing reduction, although the gap it introduces is not a constant but goes to 1 as m goes to infinity:

Theorem 3 ***Cook-Levin Theorem:*** *It is NP-Hard to distinguish whether a 3-SAT formula is*

1. *Satisfiable*
2. *Any assignment satisfy at most $(1 - \frac{1}{m})$ fraction of the clauses.*

2 Point of View 2: Extremely Robust Proof

Definition 4 ***NP:*** *A Language $\mathcal{L} \in$NP if there exist a polynomial-time Turing machine V ("Verifier"), that is when given a certificate or a proof π, and x verifies whether $x \in \mathcal{L}$*

- **Completeness**: *if $x \in \mathcal{L}$, $\exists$ proof π such that $V^{\pi}(x)$ accepts*
- **Soundness**: *if $x \notin \mathcal{L}$, $\forall$ proofs π, $V^{\pi}(x)$ rejects*

PCP is basically a generalization of the above definition, since the key concept is again to verify certificates that an instance might belong to a language, but using random bits as part of the input in order to compensate for the fact that we might not examine the whole proof. Formally PCP is defined as follows:

Definition 5 ***PCP:*** *A Language $L \in PCP(r, q)$ if there exist a probabilistic polynomial-time Verifier V such that when given a certificate or a proof π , V :*

1. *Read input x and $O(r)$ random bits.*
2. *Based on x and the random bits , query $O(q)$ bits from the proof.*
3. *Compute and either accept of reject.*
   - *if $x \in \mathcal{L}$, $\exists$ proof π such that $P[V^{\pi}(x) = 1] = 1$*
   - *if $x \notin \mathcal{L}$, $\forall$ proofs π $P[V^{\pi}(x) = 1] \leq \frac{1}{2}$*

Theorem 6 ***PCP Theorem***: *NP=PCP(log(n),1)*

Note that $PCP(log(n), 1) \subseteq NP$ is easy to prove. Consider a language $\mathcal{L} \in PCP(log(n), 1)$, then this language has a probabilistic Verifier. We can construct a deterministic verifier using this PCP verifier as follows. Given a proof and an input instance, we simply run the PCP verifier for all possible random bit strings and accept iff PCP verifier accepts for every random string. Since the random string is $O(log(n))$, and the PCP verifier runs in polynomial time,then so does the resulting deterministic verifier.

3 Equivalence between the Two Views of PCP Theorem

It is now time to actually see how the PCP theorem can be seen as a tool to prove hardness of approximation, using the fact that problems in NP have probabilistically checkable proofs. We will show how to use the PCP theorem to describe a gap introducing reduction for 3-SAT. From the PCP theorem, we can conclude that 3-SAT has a verifier V that reads $c \cdot log(n)$and q-bits from the proof π. Fix an instance x of 3-SAT, and a random string r. Now, let $V(x, r)$ be a Boolean formula on the q-bits read from the proof $\pi_{j_1}, \pi_{j_2}, \pi_{j_3} \ldots \pi_{j_q}$, which becomes true iff the corresponding assignment would cause the probabilistic verifier to accept. Transform this Boolean formula into a 3-SAT formula $C_{x,r}$ of constant size(depends on q). Our final 3-SAT is $\underset{r}{\cup} C_{x,r}$.

- if $x \in L$, $\exists \pi$ s.t $P[V^{\pi}(x) = 1] = 1$, then P_r[All clauses of $C_{x,r}$ is satisfied]$= 1$
- if $x \notin L$, $\forall$ π, $P[V^{\pi}(x) = 1] \leq \frac{1}{2}$, then at least $\frac{1}{2}$ of $C_{x,r_1}, C_{x,r_2}, \ldots C_{x,r_n}$ are not satisfied. Since the number of clauses contained in each $C_{x,r}$ is constant, and at least one of them is not satisfied when $C_{x,r}$ is not satisfied, we get that at most $1 - \epsilon$ fraction of the clauses is satisfied, for some constant $\epsilon > 0$.

4 $NP \subseteq PCP(poly(n), 1)$

In this section, we prove a weaker version of the PCP theorem. This theorem basically shows that every NP problem has an exponentially-long proof that can be locally tested by looking only at a constant number of bits. This theorem will be used in the proof of the general PCP theorem i.e $NP \subseteq PCP(log(n), 1)$

Theorem 7 $NP \subseteq PCP(poly(n), 1)$

We prove this theorem by designing an appropriate verifier for an NP-complete language. The verifier expects the proof to contain an encoded version of the usual certificate. The verifier checks such an encoded certificate by simple probabilistic tests.

4.1 Walsh-Hadamard Codes

To encode the certificate, we use Walsh-Hadamard Codes. It encodes n bits using 2^n bits. The encoding function $WH : \{0,1\}^n \to \{0,1\}^{2^n}$ maps a string $u \in \{0,1\}^n$ to the truth table of the function $x \to x \odot u$, where for $x, y \in \{0,1\}^n$ $x \odot y = \sum_{i=1}^{n} x_i y_i \mod 2$. The key observation to do here is that the codewords can be also viewed as a linear function from $\{0,1\}^n$ to $\{0,1\}$.

4.1.1 Local Testing of Walsh-Hadamard Codes

Suppose we are given a function f from $\{0,1\}^n$ to $\{0,1\}$ and we want to test whether f is an encoding of u for some $u \in \{0,1\}^n$. Since, Walsh-Hadamard codewords is precisely the set of all linear functions, f can be tested by checking if f is linear or not.

$$f(x+y) = f(x) + f(y) \quad \forall x, y \in \{0,1\}^n \tag{1}$$

One way to test f is to choose x, y at random and verify (1). Such test would accept a linear function with probability 1. However, we can't guarantee that every non-linear function is rejected with high probability. For example, if f is very close to a linear function, meaning that f is obtained by modifying a linear function on a very small fraction of inputs, then such a test would test the non-linear part with very low probability. Hence, we set the goal less ambitiously and we ask for a test that accepts every linear function , and rejects with high probability every function that is far from linear.

Definition 8 *Let $\epsilon > 0$. We say that $f, g : \{0,1\}^n \to \{0,1\}$ are $(1-\epsilon)$-close if they agree on all but ϵ fraction of their output.*

Theorem 9 *(Linearity Testing [3]) Let $f\colon \{0,1\}^n \to \{0,1\}$ be such that:*

$$P_{x,y}[f(x+y) = f(x) + f(y)] \geq \rho$$

for some $\rho > \frac{1}{2}$. Then f is ρ-close to a linear function

In [3] it is shown that, for any $\delta \in (0, \frac{1}{2})$, we can have a δ-linearity test which reads $O(\frac{1}{\delta})$ many bits and rejects a function with $p \geq \frac{1}{2}$ if f is not $(1-\delta)$ close to a linear function. Repeating this procedure constantly many times, we can actually achieve any constant error probability we wish.

4.2 Local Decoding of Walsh-Hadamard Code

Suppose that f is $(1-\delta)$-close to a linear function $\hat{f}$. Now we are interested in $\hat{f}$ for any x. Querying $f(x)$ might be bad because for that specific x, it might hold that $f(x) \neq \hat{f}(x)$. Fortunately, there is still a simple way to learn $\hat{f}(x)$ with good probability while making only two queries to f:

1. Choose $x' \in \{0,1\}^n$ uniformly at random
2. Output $f(x'+x) + f(x')$

Since x' and $x + x'$ are uniformly distributed, by the union bound, we have with $p \geq (1 - 2\delta)$ both $f(x' + x)$ and $f(x')$ are good (i.e $f(x' + x) = \hat{f}(x' + x)$ and $f(x') = \hat{f}(x')$). In that case:

$$\begin{aligned} f(x' + x) + f(x') &= \hat{f}(x' + x) + \hat{f}(x') \\ &= \hat{f}(x') + \hat{f}(x) + \hat{f}(x') \\ &= \hat{f}(x) \end{aligned}$$

In the coming lecture , the proof of $NP \subseteq PCP(poly(n), 1)$ will be continued.

References

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

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1): 70-122, 1998.

[3] M. Blum, M. Luby and R. Rubinfeld. Self-testing/correcting with applications to numerical problems.*J. Comput. Syst. Sci.*, 47(3):549-595, 1993.

[4] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), 2007.