The PCP Theorem July 6, 2005

Lecture 3

Lecturer: Madhu Sudan *Scribe: Elena Losievskaja*

1 MAX Clique problem hard to approximate: PCP approach

Today we will see some applications of the PCP theorem. In particular, we will go through the demonstration of Feige, Goldwasser, Lovasz, Safra and Szegedy showing that the **MAX Clique** problem is hard to approximate.

First recall the definition of the **MAX Clique** problem.

Definition 1 *Given a graph $G = (V, E)$, a set of vertices $C \subseteq V$ is a clique, if $\forall u, v \in C \;\; \exists (uv) \in E$. The* **MAX Clique** *problem is to find the largest clique in G.*

The idea of the demonstration is to express the task of finding the best proof as the task of finding the largest clique in the graph.

Theorem 1 *If there exist constants q, c and s such that $NP = PCP_{c,s}[r, q]$ and $r = O(\log(n))$, then there exist $\alpha < 1$, such that α-approximating* **MAX Clique** *is NP-hard.*

Proof
Let L be an NP-complete laungage, V a PCP verifier for L and x an instance of the problem L. We are going to construct a graph G such that if $x \in L$, then G has a clique of the size at least k and if $x \notin L$, then G has a clique of the size at most αk for some k.

First let's define the concept of *view of a verifier V*. Given a random string R, a verifier reads q bits of the proof: $(b_1, b_2, \ldots, b_q)$, where $b_j := \pi_{i_j(R,b_1,b_2,\ldots,b_{j-1})}$ is the j-th bit and it might depend on the $j - 1$ bits already read by the verifier. We call a string of the form $(R, b_1, b_2, ..., b_q) \in \{0, 1\}^{r+q}$ a *view of the verifier V*. Given this view the verifier accepts or rejects x.

When are two views inconsistent? First consider the case when the random strings used in these views are different, i.e. we have $(R, b_1, b_2, ..., b_q)$ and $(R', b'_1, b'_2, ..., b'_q)$. These views are inconsistent, if for some i, j the bits b_i and b'_j refer to the same location of the proof π, but $b_i \neq b'_j$. Next consider the case when the random strings used in these views are the same, i.e. we have $(R, b_1, b_2, ..., b_q)$ and $(R, b'_1, b'_2, ..., b'_q)$. These views are inconsistent, if for some i $b_i \neq b'_i$. So, we can say that two views are consistent if they query disjoint locations or agree on intersecting locations.

How do we use these views to construct the graph G? The vertices of the graph are all accepting views of the verifier on the input string x. In polynomial time we can write all views and exclude all not accepting views. The views u and u' are connected by an edge, if they are consistent.

Now we claim that a clique C in G corresponds to a partial proof. Assume that initially the proof π has undefined values in all locations. We take the views corresponding to the vertices in the clique C and start filling the locations of the proof. If the location i of π appears in some view, then we take its value from the view and fill the location i in π, if it doesn't appear we leave the value of the location i in π undefined. This way we go through all locations of π and fill them in or leave them undefined. Notice that, for our edges definition, there will not be some location for which we have an inconsistency. Clearly, it might happen that there is no view that queried some location j of π, and so the value for this location is not defined by the set of views in C, and so the proof π is not complete, i.e. partial. To complete the proof we simply change all undefined values to "0". So, any clique in the graph corresponds to a partial proof.

Moreover, any proof corresponds to a clique in the graph. Why is this? Simply because all accepting consistent random strings form a clique in G (by the construction of G). Let C_{max} denote the size of a maximum clique in the graph G. Using the same reasoning used in the previous lectures, we have:

- if $x \in L\ \ \exists\pi:\ \ \Pr_R[V^{\Pi}(x;R)=1] \geq c \Rightarrow C_{max}(G) \geq c2^r$,
- if $x \notin L\ \ \exists\pi:\ \ \Pr_R[V^{\Pi}(x;R)=1] \leq s \Rightarrow C_{max}(G) \leq s2^r$,

So, approximating **MAX Clique** within the factor $\alpha > \frac{s}{c}$ is NP-hard. ■

Consider the standard form $NP = PCP_{1,\frac{1}{2}}[O(log(n)), q]$ for some constant q, then it is NP-hard to approximate **MAX Clique** within the factor $\alpha > \frac{1}{2}$. But what if we run the verifier twice? Then we have $NP \subseteq PCP_{1,\frac{1}{4}}[O(log(n)), 2q]$, i.e. it is still hard to approximate **MAX Clique** even with the factor $\frac{1}{4}$. In general, $\forall t\ \ PCP_{c,s}[r,q] \subseteq PCP_{c^t,s^t}[tr,tq]$ and $\frac{1}{2^t}$-approximating of **MAX Clique** is NP-hard. Then it is hard to approximate **MAX Clique** for any constant factor.

But what happens if we allow the approximation factor α to be a function of N, where N is the number of vertices in the graph G?

Let $t(n)$ be a slowly growing function of n. Since $NP \subseteq PCP_{1,\frac{1}{2^{t(n)}}}[O(t(n)log(n)), t(n)q]$ by repeating the previous construction for any instance x of length n, we obtain a graph G_x with the number of vertices $N \leq 2^{q+r} = 2^{t(n)(r+q)}$. This construction takes time polynomial in N. If we run an $\alpha(N)$-approximation algorithm A on G_x, we would be able to decide if $x \in L$ by considering the size of the clique found by A for some value of $\alpha(N)$. Indeed:

- if $A(G_x) > \alpha(N)2^{t(n)(r+q)} \Rightarrow x \in L$,
- if $A(G_x) \leq \frac{2^{t(n)(r+q)}}{2^{t(n)}} \Rightarrow x \notin L$,

for $\alpha(N) \geq 1/2^{t(n)}$. For $t(n) = log(n)$ the value of $\alpha(N)$ should be roughly greater then $1/2^{(log(N))^{1/2}}$.

In other words, such an algorithm A would imply that $NP = DTIME(n^{log(n)})$ which is conjectured to be false. It means that then **MAX Clique** can not be approximated even if the approximation factor α is a slowly growing function of N (unless $NP = DTIME(n^{log(n)})$).

2 MAX Clique problem hard to approximate: PCP and expander walk approach

As we saw above **MAX Clique** is hard to approximate within any factor α (unless $P = NP$), which is a constant or a slowly growing function of n (unless $NP = DTIME(n^{log(n)})$) for n-vertices graphs. In this section we prove much stronger result, namely: **MAX Clique** is hard to approximate within a factor $\frac{1}{n^{\epsilon}}$ for any $\epsilon > 0$ on a n-vertex graph unless $P = NP$.

The intuition tells us that we have to increase the randomness, if we want to prove any stronger result. But if we increase the randomness, the algorithm takes superpolynomial time (as we saw at the end of the previous section). Then the idea is to construct "almost" random strings, using small randomness.

First, we introduce the concept of an *expander walk.*

Definition 2 *A graph $H = (V, E)$ is γ-expander, if $\forall S \subseteq V$ such that $|S| \leq \frac{|V|}{2}$ the number of edges between the set S and the set $\bar{S}$ is at least $\gamma|S|$, where $\gamma \geq 0$*

Definition 3 *A graph $H = (V, E)$ is d-regular, if every vertex in the graph has d neighbors.*

Definition 4 *A random walk of length t in a graph $G = (V, E)$ is a set of vertices $v_1, ..., v_t$, where $v_i \in V$ for any $i \in 1, ..., t$, v_1 is selected at random and v_i is a random neighbour of v_{i-1} for any $i \in 2, \ldots, t$.*

Some facts that we use without going into the proves:

1. d-regular γ-expanders can be constructed in polynomial time
2. a random walk of length t on a γ-expander $H = (V, E)$ produces a set of vertices completely enclosed in a set B of density $\rho = \frac{|B|}{|V|}$ with probability $\rho^{\Omega(t)}$

Now we can use expander walks to prove the following theorem:

Theorem 2 *If there exist a constant q such that* **MAX Clique** $\subseteq PCP_{1,\frac{1}{2}}[r, q]$, *where* $r = O(\log(n))$, *then* **MAX Clique** $\in PCP_{1,\frac{1}{2^{\Omega(t)}}}[r + O(t), qt]$, *where* $t = \log(n)$.

Idea of the proof

Let L be an NP-complete language, V - a verifier for L and x - an instance of the problem L. Suppose that there exists a constant q such that **MAX Clique** $\subseteq PCP_{1,\frac{1}{2}}[r, q]$, where $r = O(\log(n))$, then we have a verifier V' that accepts $x \notin L$ with the probability at most $\frac{1}{2}$ using r bits of randomness. The idea is the following: construct a verifier V that will use the verifier V' as a subroutine to decide if $x \in L$ using at most $r + O(t)$ bits of randomness.

First, we construct an expander H on 2^r vertices (this construction takes polynomial time according to the fact (1) above). Let each vertex of H be a random string that the verifier V' might use to decide x. The verifier V works as follows:

1. takes a vertex u in H at random
2. creates a set of t random strings by performing a random walk of length t in H starting from u
3. run V' on each t random strings and input string x
4. accept x, if V' accepts x on all t random strings

Note that the verifier V uses only $r + O(t)$ bits of randomness.

What is the probability that V accept $x \in L$? If $x \in L$, there exist a proof such that the subroutine verifier V' accepts x with probability 1 using any random string. So, the verifier V will also accepts x with probability 1.

What is the probability that V accepts $x \notin L$? The soundness of the verifier V' is $\frac{1}{2}$, so there are at most $\left(\frac{1}{2}\right) 2^r$ random strings that make V' accept x. Let B be a set of these strings with regard to some proof π. The second fact above tells us that the random walk will produce a set of vertices completely contained in B with probability at most $\left(\frac{|B|}{|V|}\right)^{\Omega(t)} = \left(\frac{1}{2}\right)^{\Omega(t)}$. So, the verifier V will also accepts $x \notin L$ with probability at most $\left(\frac{1}{2}\right)^{\Omega(t)}$.

Therefore, **MAX Clique** $\in PCP_{1,\frac{1}{2^{\Omega(t)}}}[r + O(t), qt]$. If we let $t = \log(n)$, **MAX Clique** $\in PCP_{1,\frac{1}{2^{\Omega(\log(n))}}}[O(log(n)), O(log(n))]$. But this means that it is hard to approximate **MAX Clique** within the factor $\alpha = \left(\frac{1}{2}\right)^{\Omega(\log(n))} = \left(\frac{1}{n}\right)^{\Omega(1)} = \frac{1}{n^{\epsilon}}$ for any $\epsilon > 0$ on n-vertex graph.