CMPUT 675: Approximation Algorithms **Winter 2005**

Lecture 23: April 6th

Lecturer: Mohammad R. Salavatipour *Scribe: Junfeng Wu*

23.1 Hardness of Clique

Definition 23.1 *(MAX-CLIQUE) Given a graph with n vertices, find a maximum size clique in it, i.e. a complete subgraph of maximum size.*

Last lecture we proved a constant hardness for Clique. The best known algorithm for approximating clique has a factor of $O(\frac{n}{\log^n})$ which is quite high. Note that this is slightly better than the trivial algorithm which picks a single vertex and returns (which has approximation factor $O(n)$). It turns out that we cannot do much better than this. Håstad has proved that for any $\epsilon > 0$ there is no polytime approximation algorithm for clique with factor $n^{\frac{1}{2}-\epsilon}$ (if P$\neq$NP) or $n^{1-\epsilon}$ (if ZPP$\neq$NP). Our goal is to prove a polynomial hardness for clique.

To prove this hardness result, we start with a constant hardness result for Clique. Then show that it is hard to approximate Clique within *any* constant factor. Finally, we show how to improve this to a polynomial.

Consider a $PCP_{1,\frac{1}{2}}(d \log n, q)$ verifier F for SAT, where d and q are constants. Let $r_1, r_2, ..., r_{n^d}$ be the set of all possible random strings to F. Given an instance ϕ of SAT, we construct a graph G from F and ϕ which will be an instance of Clique. G has one vertex $V_{r_i,\tau}$ for each pair (i, σ), where i is for the random string r_i and σ is a truth assignment to q variables. The number of of all the vertices (i.e. the size of the graph) is $|V| = n^d 2^q$.

Equivalently, we define the vertices as follows. An accepting transcript for F on ϕ with random string r_i is q pairs $(p_1, a_1), \ldots, (p_q, a_q)$ s.t. for every truth assignment that has values $a_1, \ldots, a_q$ for variables $p_1, \ldots, p_q$, verifier F given r_i checks positions $p_1, \ldots, p_q$ in that order and accepts. Note that once we have ϕ, r_i, $a_1, \ldots, a_q$ it is easy to compute $p_1, \ldots, p_q$. For each transcript we have a vertex in G.

Two vertices (i, σ) and (i', σ') are adjacent iff σ, σ' don't assign different values to same variable, i.e. they are consistent.

If ϕ is a yes instance: so there is a proof (truth assignment) π, such that F accepts (given π) on all random strings. For each r_i, there is a corresponding σ (which has the same answers as in π) and is an accepting transcript. We have n^d random strings and therefore there are n^d vertices of G (corresponding to those). They form a clique (because they come from the same truth assignment and therefore are consistent, i.e. adjacent); therefore G has a clique of size $\geq n^d$.

if ϕ is a no instance: we want to show that every clique in G has size at most $\frac{n^d}{2}$. By way of contradiction suppose we have a clique C of size $c > \frac{n^d}{2}$. Assume that $(i_1, \sigma_1)...(i_c, \sigma_c)$ are the vertices in the clique. Therefore the transcripts $\sigma_1 ... \sigma_c$ (partial truth assignment) are consistent. We can extend this truth assignment to a whole proof (truth assignment) such that on random strings $i_1, ..., i_c$, verifier F accepts. therefore verifier accepts for $> \frac{n^d}{2}$ strings, which contradicts the assumption that ϕ is a no instance.

The gap created here is exactly the soundness probability and the smaller S is, the larger gap we get.

By simulating a verifier $PCP_{1,\frac{1}{2}}(d \log n, q)$ for k times and accepting iff all of those simulations accept, we get a $PCP_{1,\frac{1}{2^k}}(k \cdot d \log n, k \cdot q)$ verifier. Note that in this case the size of the construction G is $n^{kd}2^{kq}$, which is polynomial as long as k is some constant. This will show a hardness of 2^k, which is a constant.

Corollary 23.2 *For any constant S, it is NP-hard to approximation clique within a factor of S. say ($S = 1/2^k$)*

To get an n^{ε}-gap, we need S to be polynomially small, and for that we need to repeat $k = \Omega(\log n)$ times. Therefore, $k \cdot q = O(\log n)$ which is Ok. But the length of random string becomes $k \log n = \Omega(\log^2 n)$, and the size of graph becomes super-polynomial. Therefore, to get a polynomial hardness result for Clique we need a PCP verifier with $O(\log n)$ random bits and polynomially small soundness probability, i.e. $PCP_{1,\frac{1}{n}}(O(\log n), O(\log n))$.

The trick here is to start with only $O(\log n)$ random bits and use expander graphs to generate $O(\log n)$ random strings, each of length about $\log n$.

Definition 23.3 *(Expander Graph) Every vertex has the same constant degree, say d, and for every non-empty set $S \subset V$, $|E(S,\overline{S})| \geq \min\{|S|, |\overline{S}|\}$.*

There are explicit construction of expander graphs. Let H be an expander graph with n^d nodes. For each node we assign a label which is a binary string of length $d \log n$. We can generate a random walk in H using only $O(\log n)$ random bits: we need $d \log n$ bits to choose the first vertex, and we only need constant number of random bits to choose the neighbor at every step. Therefore, to have a random walk of length $O(\log n)$ we need only $O(\log n)$ bits.

Theorem 23.4 *For any set S of vertices of H with $< \frac{n^d}{2}$ vertices, there is a constant k such that the probability that a random walk of length $k \log n$ lies entirely in S is $< \frac{1}{n}$.*

Here is the outline of the proof of this theorem. By definition of the expander graphs, if you have a set S of vertices, we expect a constant fraction of edges out of the vertices of S be going into $\overline{S}$. Therefore, if you start a random walk from a vertex in S, at every step, there is a constant probability that this walk jumps into $\overline{S}$. So the probability that a random walk of length $\Omega(\log n)$ stays entirely within S is polynomially small.

Next lecture, we will show how, given a PCP verifier F for a language in $PCP_{1,\frac{1}{2}}(d \log n, q)$, generate a PCP verifier F' which uses $O(\log n)$ random bits, queries only $O(\log n)$ bits, has completeness 1, and soundness $\frac{1}{n}$, i.e. $NP = PCP_{1,\frac{1}{n}}(O(\log n), O(\log n))$.