CSE 533: The PCP Theorem and Hardness of Approximation **(Autumn 2005)**

Lecture 6: Powering Completed, Intro to Composition

Oct. 17, 2005

Lecturer: Ryan O'Donnell and Venkat Guruswami *Scribe: Anand Ganesh*

1 Powering Completed

In the previous lecture we began the Powering step; this step takes $(G, \mathcal{C})$, a constraint graph on an (n, d, λ)-expander over a constant-sized alphabet Σ, and outputs $(G', \mathcal{C}')$. The properties of the new constraint graph are that $\text{size}' = O(\text{size})$, the alphabet size remains a constant (albeit a much much larger one), $\text{gap} = 0 \Rightarrow \text{gap}' = 0$, and the main property:

$$\text{gap}' \geq \frac{t}{O(1)} \cdot \min(\text{gap}, 1/t).$$

Here t is a fixed constant parameter. The transformation is based on the idea of constructing G' from G by having edges in G' correspond to walks of length about t in the original graph.

Let us go over the construction of G' in more detail. As we said in the previous lecture, the new alphabet is $\Sigma' = \Sigma^{1+d+d^2+d^3...+d^t}$. The new edge set E' and constraint set C' are defined as follows:

- E': Pick a random vertex a. Do an A.S.R.W. from a, ending on b. This generates a weighted edge $e' = (a, b)$. As we noted, this seems like it might be a problem, since we get a) weighted edges, and b) too many edges. We discuss this shortly.
- $\mathcal{C}'$: Here is the test that is then applied to the labelling $\sigma' : V' \to \Sigma'$:
  - If the number of steps in the A.S.R.W. was greater than $B := (10ln|\Sigma|)t$, then accept.
  - Otherwise, for each step $u \to v$ along the path, if $dist_G(a, u) \leq t$ and $dist_G(b, v) \leq t$ (i.e. the vertices are close enough that there will be opinions), *and* if $(\sigma'(a)_u, \sigma'(b)_v)$ is violating for the old constraint on $(u, v) \in E$, reject

Fixing the problems: Given this construction of a weighted constraint graph (with multiple edges and edges between all possible pairs!), throw away all paths $e' \in E'$ corresponding to walks of length greater than B. Since these all had *always-satisfied* constraints, this can only cause the gap to go up. Having done this, we get:

$$deg(G) \leq 1 + d + d^2 + d^3... + d^B = \text{constant}$$
$$size' \leq O(size).$$

The only problem left is that the edges still have weights, and we want unweighted constraint graphs. However note that all the weights are simply rational numbers depending only on the universal constants d and t; hence we can replace them appropriately with parallel unweighted edges. Thus we have fixed the annoying details associated with our randomized way of describing the edges E' and can proceed to analyze the gap.

We now need to show that $\forall \sigma' : V' \to \Sigma'$,

$$\Pr[\text{test rejects } \sigma'] \geq \frac{t}{O(1)} \min(\text{gap}, \frac{1}{t}).$$

So let σ' be the "best" assignment for our $\mathcal{C}'$. We extract an assignment $\sigma : V \to \Sigma$ from it based on the plurality vote method analyzed in the previous lecture. Let $F \subset E$ be the set of edges in G violated by σ; we know that $|F|/|E| \geq \text{gap}$. Throw away some edges from F is necessary so that $|F|/|E| = \min(\text{gap}, 1/t)$.

We will now recall the notion of a "faulty step" from the previous lecture; however we will also introduce the notion of a "faulty* step", which is needed to analyze our B-truncated verifier.

Definition 1.1. *Let $e' : a \to b$ be a random path as chosen by our verifier. Step $u \to v$ is* faulty *if the following hold:*

- $(u, v) \in F$,
- $dist_G(a, u) \leq t$ *and* $\sigma'(a)_u = \sigma(u)$,
- $dist_G(b, v) \leq t$ *and* $\sigma'(b)_v = \sigma(v)$.

We further define a step to be faulty* *if*

- *the step is faulty,*
- *the number of steps in the overall $a \to b$ walk was at most B.*

Let us also define some random variables based on the verifier's A.S.R.W.:

Definition 1.2.

- $N =$ *number of faulty steps.*
- $N^* =$ *number of faulty* steps.*
- $S =$ *total number of steps.*
- $N_F =$ *number of steps that were in F.*

Note that by the definitions we have

$$N^* = N \cdot \mathbf{1}_{\{S \leq B\}},$$

and

$$N^* \leq N \leq N_F.$$

1.1 Analysis

The key point of the definition of faulty* steps is that whenever the verifier picks a random walk that has a faulty* step (i.e., whenever $N^* > 0$), the verifier rejects σ'. (Note that the verifier may still reject even if it has no faulty* steps.) Thus we have

$$\mathrm{gap}' = \Pr[\text{test rejects } \sigma'] \geq \Pr[N^* > 0] \geq \frac{\mathrm{E}[N^*]^2}{\mathrm{E}[(N^*)^2]},$$

where we used the Second-Moment Method in the last step as discussed in the last lecture. To complete the proof we need to show that $\mathrm{gap}' \geq \frac{t}{O(1)} \cdot \frac{|F|}{|E|}$. Hence we are done if we can show the following two lemmas:

Lemma 1.3. $\mathrm{E}[N^*] \geq \frac{t}{8|\Sigma|^2} \cdot \frac{|F|}{|E|}$

Lemma 1.4. $\mathrm{E}[(N^*)^2] \leq O(1) \cdot t \cdot \frac{|F|}{|E|}$

Proof. (Lemma 1.3.) To prove the lower bound on $\mathrm{E}[N^*]$ we use the lemma from the last lecture that said $\mathrm{E}[N] \geq \frac{t}{4|\Sigma|^2} \cdot \frac{|F|}{|E|}$. We have

$$\begin{aligned}
\mathrm{E}[N^*] &= \mathrm{E}[N \cdot \mathbf{1}_{\{S \leq B\}}] \\
&= \mathrm{E}[N \cdot (1 - \mathbf{1}_{\{S > B)\}}] \\
&= \mathrm{E}[N] - \mathrm{E}[N \cdot \mathbf{1}_{\{S > B\}}] \\
&\geq \frac{t}{4|\Sigma|^2} \cdot \frac{|F|}{|E|} - E[N \cdot \mathbf{1}_{\{S > B\}}],
\end{aligned}$$

using the earlier lemma. Now,

$$\begin{aligned}
\mathrm{E}[N \cdot \mathbf{1}_{\{S > B\}}] &= \Pr[S > B] \cdot \mathrm{E}[N \mid S > B] \\
&= (1 - 1/t)^B \cdot \mathrm{E}[N \mid S > B] \\
&\geq \exp(-B/t) \cdot \mathrm{E}[N_F \mid S > B] \\
&= \exp(-B/t) \cdot \mathrm{E}[S \mid S > B] \cdot \frac{|F|}{|E|} \\
&= \exp(-B/t) \cdot (B + t) \cdot \frac{|F|}{|E|} \\
&\geq \frac{1}{|\Sigma|^{10}} \cdot (20 \ln |\Sigma| \cdot t) \cdot \frac{|F|}{|E|} \qquad \text{(by definition of } B\text{)} \\
&\geq \frac{t}{8|\Sigma|^2} \cdot \frac{|F|}{|E|},
\end{aligned}$$

since $(20 \ln |\Sigma|)/|\Sigma|^{10} \leq 1/(8|\Sigma|^2)$. Combining the above two calculations completes the proof. □

Proof. (Lemma 1.4) The proof of this lemma is the only place where we use the fact that G is an (n, d, λ)-expander. In fact, this is all we use — that in a length L random walk on an expander, the number of times you hit a fixed set of edges is about what you would get if you just picked L random edges. In particular, we start with the trivial upper bound

$$\mathrm{E}[(N^*)^2] \leq \mathrm{E}[(N_F)^2].$$

Let us express $N_f = \sum_{i=1}^{\infty} \chi_i$, where $\chi_i = \mathbf{1}[i\text{th step is in } F]$. We have:

$$\begin{aligned}\mathrm{E}[(N_F)^2] &= \sum_{i,j=1}^{\infty} \mathrm{E}[\chi_i \cdot \chi_j] \\ &\leq 2\sum_{i=1}^{\infty} \Pr[\chi_i = 1] \cdot \sum_{j \geq i} \Pr[\chi_j = 1 \mid \chi_i = 1] \qquad (*)\end{aligned}$$

Now $\Pr[\chi_j = 1 \mid \chi_i = 1]$ is 1 if $j = i$; otherwise it equals

$$\begin{aligned}&\Pr[\text{the walk takes at least } j-i \text{ more steps}] \\ &\qquad \times \Pr[\text{a walk, starting from a random } F \text{ endpoint, takes its } (j-i)\text{th step in } F].\end{aligned}$$

The first quantity here is just $(1 - 1/t)^{j-i}$. The second quantity, from Lecture 3's expander lemma on this subject, is at most $|F|/|E| + (\lambda/d)^{j-i-1}$. Now substituting this into (*), we get:

$$\begin{aligned}\mathrm{E}[(N^*)^2] &\leq 2\sum_{i=1}^{\infty} \Pr[\chi_i = 1] \cdot \left(1 + \sum_{\ell=1}^{\infty} (1-1/t)^{\ell} \left(\frac{|F|}{|E|} + (\lambda/d)^{\ell-1}\right)\right) \\ &\quad 2\sum_{i=1}^{\infty} \Pr[\chi_i = 1] \cdot \left(1 + \sum_{\ell=1}^{\infty} (1-1/t)^{\ell} \cdot \frac{|F|}{|E|} + \sum_{\ell=1}^{\infty} (\lambda/d)^{\ell-1}\right) \\ &\leq 2\sum_{i=1}^{\infty} \Pr[\chi_i = 1] \cdot \left(1 + (t-1) \cdot \frac{|F|}{|E|} + O(1)\right) \qquad \text{(since } \lambda < d \text{ are consts)} \\ &\leq O(1) \cdot \sum_{i=1}^{\infty} \Pr[\chi_i = 1] \qquad \text{(since } |F|/|E| \leq 1/t) \\ &= O(1) \cdot \mathrm{E}[N_F] \\ &= O(1) \cdot t\frac{|F|}{|E|},\end{aligned}$$

as claimed. □

2 Introduction to Composition

We are now at stage 4 of the proof of the PCP theorem called Alphabet Reduction or Composition. Given a constraint graph with a large alphabet size, the problem is to reduce the alphabet size

($\Sigma^{d^t} \to \Sigma_0$ where $|\Sigma_0|$ is an absolute constant, say 64) without adversely affecting other parameters like the gap. In simplified terms, the composition step may be thought of as a recursive step (with some extra features) within the larger PCP reduction as we will describe below.

Recall that the PCP may be cast in the following form. It is a reduction P such that maps a Boolean constraint (a 3SAT formula, or a Boolean circuit) Φ into a constraint graph $(G, \mathcal{C})$ over a fixed alphabet Σ_0 such that

$$\begin{aligned} P \ : \ \Phi &\to (G, \mathcal{C}) \text{ such that} \\ \Phi \text{ satisfiable} &\implies (G, \mathcal{C}) \text{ satisfiable} \\ \Phi \text{ not satisfiable} &\implies \mathrm{gap}(G) > \epsilon \text{ i.e. } < 1 - \epsilon \text{ of the constraints in } \mathcal{C} \text{ are satisfiable} \end{aligned}$$

Consider a constraint graph $H = G^t$ obtained after the powering step. Let c_e be the constraint on edge e. This is a binary constraint over a large alphabet (like Σ^{d^t}). We can express it as a Boolean constraint Φ_{c_e} over several Boolean variables (using some standard encoding), and apply a PCP reduction as above, call it P_e, to this constraint to get a new set of binary constraints over the alphabet Σ_0. Thus, we can reduce the alphabet size, and if the original c_e was not satisfied at least an ϵ fraction of the newly produced constraints over Σ_0 must be unsatisfied by any assignment. Therefore, one also expect that the new gap is at least $\epsilon \cdot \mathrm{gap}(H)$, and the alphabet is now Σ_0.

Of course, our whole goal is to construct a PCP reduction, so how can we use a PCP reduction recursively without falling prey to a circular argument? The key is that we will apply this "inner" PCP reduction only to constraints of *constant size* and thus the reduction can be arbitrarily inefficient (and hence perhaps easier to construct). Therefore, the hope would be to construct from scratch a highly inefficient PCP reduction, and use it as above.

Consistency. However, there is a subtle issue which makes the above recursion not as straightforward as we suggested. Suppose $e = (u, v)$ and $e' = (v, w)$ are two edges in the constraint graph H that share a vertex v. The reductions P_e and $P_{e'}$ ensure that the constraints c_e and $c_{e'}$ are both individually satisfiable. However, we need to ensure that the constraints in H are all satisfied by a single, common assignment to the variables. In particular, this means that we need to check not only that c_e and $c_{e'}$ are satisfiable, but that they are satisfied by assignments which are consistent on the shared variable v. This consistency will be achieved by imposing additional requirements on the PCP reduction using an entity called the Assignment Tester.

Roughly speaking, an assignment tester checks that a constraint is satisfied by a *particular* assignment to its variables. So in the above recursive appraoch, we assume that we are given an assignment $\sigma(v)$ to each of the variables in H, and the assignment tester corresponding to edge $e = (u, v)$ must check that $(\sigma(u), \sigma(v))$ satisfies the constraint c_e. Now consider the case where $(\sigma(u), \sigma(v))$ differs from a satisfying assignment to c_e in just one bit. Then, most constraints of the assignement tester may accept, whereas we want a constant fraction of them to reject. Thus this is too stringent a task to impose on the assignment tester.

We relax the requirement so that the assignment tester must check *proximity* of the claimed assignment to a satisfying assignment of the constraint in the Hamming metric. As we will see

in the next two lectures, this task becomes feasible. But let us see how this relaxation affects the composition idea.

Consider edges $e = (u, v)$ and $e = (v, w)$ sharing a vertex v. Given vertex assignments for vertices u, v, w, P_e checks that $\sigma(v)$ is close to x_{1v} that (together with some assignment to u) satisfies c_e. $P_{e'}$ checks that $\sigma(v)$ is close to x_{2v} thatsatisfies $c_{e'}$. We want to enforce consistency on the label to v, i.e., $x_{1v} = x_{2v}$ — this is the goal. In other words, for any assignment $\sigma(v)$ there should be a unique legal, satisfying assignment that is very close to $\sigma(v)$. This condition can be met if and only if the legal assignments to the vertices are all pairwise far apart from each other, or in other words they form an error-correcting code. Hence, codes enter naturally into this framework.