Lectures on the PCP Theorems Course Instructor: V. Arvind

Lecture 1

Lecturer: Partha Mukhopadhyay *Scribe: Ramprasad Saptharishi*

We shall be following the notes of Venkatesan Guruswami's course titled "The PCP Theorems and Hardness of Approximation", can be found on his homepage.

1 The Theorem

The motivation for this was to look at NP and other classes as a proof checking system like the IP characterisation of $PSPACE$ or the AM characterisation of $BP \cdot NP$. The notion of the *Probabilistically Checkable Proof* systems was first given by (blah), and they also gave the first proof of the PCP theorem which characterises NP.

1.1 Definition of a PCP System

A PCP(r, p) system for a language L is an interactive proof system between a Prover and a Verifier and is done as follows:

- $x \in \Sigma^n$ is provided as input
- Prover does some computation on x and write a proof of length $poly(n)$
- Verifier does some computation, takes $r(n)$ random bits and probes the proof at $p(n)$ places. And then does some compution to decide whether he is accepting the proof or not.
- *Completeness:* If $x \in L$, there exists a proof such that the verifier will accept it with probability 1.
- *Soundness:* If $x \notin L$, then whatever proof is given to the verifier, the verifier rejects with probability $\geq 1/3$

1.2 The Statement of the Theorem

Theorem 1 (The PCP Theorem)**.** *For all $L \in NP$, there exists a PCP$(O(\log n), O(1))$ system for L.*

That is, there exits a proof such that the verifier makes $O(\log n)$ coin tosses and chooses a constant number of bits to probe in the proof and can provide a complete and sound verification.

2 The Constraint Graph

Definition 2. $\mathfrak{G} = (G = (V, E), C, \Sigma), C = \{C_e\}_{e \in E} \subseteq \Sigma \times \Sigma$ *is called a constraint graph, with* C_e *being the constraints on every edge as relations over* Σ*.*

An assignment is a map $\sigma : V \to \Sigma$ *and we say* σ *satisfies an edge* $(u, v) \in E$ *if* $(\sigma(u), \sigma(v)) \in C_e$

2.1 $GAP \cdot CG_{c,s}(\mathfrak{G})$: The Gap Version

Given as input is a constraint graph $\mathfrak{G} = ((V, E), C, \Sigma)$.

- If there exists an assignment $\sigma : V \to \Sigma$ such that it satisfies more than c fraction of edges, output "YES"
- If every assignment σ satisfies atmost s fraction of vertices,
- Output anything otherwise

Theorem 3 (Dinur's Theorem)**.** $GAP \cdot CG_{1,s}(\mathfrak{G})$ *is* $NP-$*hard for a fixed alphabet* Σ *and constant* s*.*

And it is clear that one can amplify this gap to $\frac{1}{2}$ with $O(\log n)$ random bits, and hence

Claim 4. *Dinur's Theorem* $\implies$ *PCP Theorem*

2.2 Roadmap of Proof of Dinur's Theorem

To show that $GAP \cdot CG_{1,s}$ is $NP-$hard, we shall reduce graph three-colourability to to this. The idea is that we can represent the decision problem of graph three-colourability as a gap problem by itself; the assignment considered as just the colour of the vertex.

$$3Colour(G) = GAP \cdot CG_{1,1-\frac{1}{m}}((G, C, \{1, 2, 3\}))$$

where the constraints are just the inequality constraints $(\Sigma \times \Sigma - diag)$ and $m = |E|$

Thus the reduction is just amplifying the gap $\left(1, 1 - \frac{1}{m}\right)$ to $(1, s)$. This will be done by applying a partial reduction for logarithmically many steps.

The partial reduction involves 4 steps:

1. *Degree Reduction*: Make it a constant degree graph, with not too much blow up in the size fo the graph. The gap reduces, but only by a constant factor

2. *Expanderize*: Convert the graph to an expander, again with just constant factor troubles in graph size and gap.

3. *Gap Amplification:* This was the innovation of Dinur. You amplify gap, compensate for the losses in the other 3 steps, but on the other hand your Σ size blows up a lot.

4. *Alphabet Reduction:* The alphabet size comes down, finally gap is now twice the original gap, with not too much blow up in the size of the graph.

3 The Reduction

3.1 Degree Reduction

Let $G = (V, E)$ be the graph with the set of constraints being C. For every vertex $v \in V$, blow it up into a cloud of $deg(v)$ many vertices, and embed an $(deg(v), d_0, \lambda_0)$ expander over the cloud for some predefined d_0 and λ_0.

Now the vertex set cardinality of the new graph is $\sum deg(u) = 2|E|$, and the edge set cardinality is $|E| + \sum \frac{deg(u)d_0}{2} = |E|(1 + d_0)$. As for the constraints, retain the edge constraints to the appropriate edges in the new graph, and equality constraints within each cloud. We then have a constant degree constraint graph over the same alphabet.

Of course, we want the gap to be respected. Define gap =fraction of edges unsatisfied by the best assignment. Let the gap of G be gap and that of the new graph is gap'. Clearly $gap = 0 \implies gap' = 0$, but we don't want to lose much on any non-zero gap.

Claim 5. *If $gap \neq 0$, $gap' \geq \frac{gap}{O(1)}$*

Proof. Let σ' be the best assignment for G'. Define an assignment σ for G as $\sigma(v) = maj(\sigma'(u))$ over all $u \in cloud(v)$. Let $|F|$ be the set of edges violated by σ, then clearly $gap \leq \frac{|F|}{|E|}$.

Let $e = (u, v) \in F$, and let $e' = (u', v')$ be the corresponding intercloud edge in G'. And let $S_u = \{v \in cloud(u) | \sigma'(v) \neq \sigma(u)\}$. Now either σ' satisfies e' or it doesn't.

Suppose σ' satisfies e', then either $u' \in S_u$ or $v' \in S_v$ since otherwise e' would be satisfied. Hence we have

$$gap|E| \leq gap'|E'| + \sum_{u \in V} |S_u|$$

- Case 1: if $gap'|E'| \geq \frac{gap|E|}{2}$, then we are done since

$$gap' \geq \frac{gap|E|}{2|E|(1+d_0)} = \frac{gap}{2(1+d_0)} = \frac{gap}{O(1)}$$

- Case 2: Otherwise, $\sum |S_u| \geq \frac{gap|E|}{2}$. Let S_u^a be the set of vertices in S_u that are coloured a by the assignment σ'.

  Clearly $|S_u^a| \leq \frac{1}{2}|cloud(u)|$, and since $cloud(u)$ is an expander, there exists $c|S_u^a|$ many edges going out of S_u^a for some appropriate c. And all these edges are violations since we have forces equality constraints over the intracloud edges.

  Thus counting the number of unsatisfied edges,

$$|F'| = gap'|E'| \geq c \sum_{u \in V} |S_u^a| \geq \frac{c}{2} gap|E|$$

  and we then have that $gap' = \frac{gap}{O(1)}$

□

Hence we now have a constant degree graph with decent gap.

3.2 Expanderize

We have our graph G to be a d-regular graph with n vertices. Now take a $H = (n, d_0, \lambda_0)$ expander (λ_0 is the second largest eigenvalue of the non-normalized adjacency matrix) for suitable parameters and just superimpose on G.

The adjacency matrix (non-normalized) is clearly satisfies

$$A_{G'} = A_G + A_H$$

And by the Rayleigh characterisation of the second largest eigenvalue, the second largest eigenvalue of the non-normalized adjacency matrix of G':

$$\lambda_2(G') \leq d + \lambda_H < d + d_0$$

and hence G' is an expander.

As for the new constraints, just add trivial (always true) constraints on the new edges. the size of the edge set $|E'| = |E| + \frac{nd_0}{2} = \frac{n(d+d_0)}{2}$ and

$$gap' = \frac{gap|E|}{\frac{n}{2}(d+d_0)} = \frac{gap}{O(1)}$$

Hence we now have reduced the generalised gap version to a special case on constant degree expanders.

3.3 Gap Amplification: a sketch

G is a (n, d, λ) expander. Fix a parameter t, the value shall be chosen later. The new constraint graph has the same set of vertices, $\Sigma' = \Sigma^{1+d+\cdots+d^t}$ and edges in the new graph are vaguely like t step walks in G.

For every $u \in V$, it can reach $1 + d + \cdots + d^t$ vertices in a t step walk, vertices counted with repetition. In this context, an assignment $\sigma' : V \to \Sigma'$ can be thought of as a vector with $1 + d + \cdots + d^t$ coordinates where each coordinate $\sigma(u)_v$ can be thought of u's opinion about v.

The constraints are the following, for every edge $e = (a, b)$, it can be intepretted as a t step walk from a to b. For every (u, v) in the t-walk, check if $(\sigma'(a)_u, \sigma'(b)_v)$ must be in C_e of the input graph.

Now, suppose σ' is the best assignment, we shall again extract a σ for G. Look at all the vertices that could have an opinion about u, and take the majority of the opinions.

Let F be the set of unsatisfied edges in G with respect to σ, and gap' the new gap.

$$gap' = \Pr_{e'}[e' \text{ is rejected under } \sigma']$$

By the way we've chosen our σ, there is a "good" chance that $\sigma'(a)_u = \sigma(u)$ and $\sigma'(b)_v = \sigma(v)$. Hence if $(u, v) \in F$, there is a good chance that e' contains the edge (u, v) to be rejected.

Roughly speaking,

$$gap' \geq \frac{1}{O(1)} \Pr_{e'}[e' \text{ passes through } F]$$

Since we have an underlying expander, missing $|F|$ is "rare"

$$\begin{aligned} gap' &\geq \frac{1}{O(1)}\left(1-\left(1-\frac{|F|}{|E|}\right)^t\right) \\ &\geq \frac{t}{O(1)}\frac{|F|}{|E|} \end{aligned}$$

and t is chosen such that the amplification, taking into account the losses in the other steps, is $geq2$.

But our alphabet is now $\Sigma^{d^{t+1}}$, and thus we need to reduce the alphabet size back to Σ, which is the next sub-routine in the reduction.

Lectures on the PCP Theorems Course Instructor: V. Arvind

Lecture 2

Lecturer: Partha Mukhopadhyay *Scribe: Ramprasad Saptharishi*

4 Overview

This class we shall look at the gap-amplification routine. This isn't the original method that Dinur adopted in her paper, this was suggested by Jaikumar Radhakrishnan by considering some special kinds of random walks

5 Gap Amplification

5.1 Lazy Random Walks

Let $G(V, E)$ be a d-regular graph. We can then talk of two kinds of random walks:

1. *After stopping random walk* (ASRW): Follows the subroutine
   (a) Pick $v \in V$ at random
   (b) Take a random step
   (c) Stop there with probability $\frac{1}{t}$
   (d) Go to step (b).
2. *Before stopping random walk* (BSRW): Given a start vertex v, follows the subroutine
   (a) Stop at v with probability $\frac{1}{t}$
   (b) Take a random step
   (c) Go to step (a)

It's clear that the length of an ASRW is atleast 1 whereas that of a BSRW could be 0 as well. And also, every ASRW walk stops after finitely many steps[1].

[1] one can easily show that probability of any infinite long ASRW is 0

Let (u, v) be some edge and Y be the random variable that counts the number of $u - v$ moves in an ASRW. Suppose you are given a promise that the edge is traversed k times, we want to estimate the distributions on the end points of the random walk, that is $\Pr[b = w|Y = k]$ where b is the end of the walk.

Before we actually get to this, let us look at $\Pr[b = w|Y \geq k]$, when the edge is traversed atleast k times. Once we've made the k-th move on (u, v), the rest is just a BSRW starting at v! Hence for every $w \in V$,

$$\Pr[b = w|Y \geq k] = P_{v,w}$$

where $P_{v,w}$ is probability of reaching w from v through a BSRW.

And turning the walk around, we can also see that for all $w' \in V$

$$\Pr[a = w'|Y \geq k] = P_{w',u}$$

Now,

$$\begin{aligned}
P_{v,w} = \Pr[b = w|Y \geq k] &= \frac{\Pr[b = w \wedge Y \geq k]}{\Pr[Y \geq k]} \\
&= \frac{\Pr[b = w \wedge Y = k] + \Pr[b = w|Y \geq k+1]}{\Pr[Y = k] + \Pr[Y \geq k+1]} \\
&= \left(\frac{\Pr[b = w \wedge Y = k]}{\Pr[Y = k]} + \frac{\Pr[b = w \wedge Y \geq k+1]}{\Pr[Y \geq k+1]}\right) / \left(\frac{\Pr[Y = k]}{\Pr[Y \geq k+1]} + 1\right) \\
&= \left(\frac{\Pr[b = w \wedge Y = k]}{\Pr[Y = k]} + P_{v,w}\right) / \left(\frac{\Pr[Y = k]}{\Pr[Y \geq k+1]}\right)
\end{aligned}$$

and with some cross-multiplication and cancelling, we get

$$\frac{\Pr[b = w \wedge Y = k]}{\Pr[Y = k]} = \Pr[b = w|Y = k] = P_{v,w}$$

and similarly for w'. This then shows that the conditioned on a fixed edge being traversed k times, the end points of the AFRW are independantly distributed.

5.2 The Amplification Process

The reduction procedure would be a little different, we shall consider a randomized verifier who is constructing the new graph G' from G. Later we shall remove the randomness to get our deterministic polynomial time reduction.

The input is the constraint graph on $G = (V, E)$ which is a (n, d, λ) expander with alphabet Σ and gap as *gap*. We want to get a new graph $G' = (V, E')$ with new constraints and alphabet as $\Sigma' = \Sigma^{1+d+\cdots+d^t}$ for a parameter t which shall be chosen later.

Verifier's Procedure:

- Do an ASRW on the graph G for t steps, let the start vertex be a and the end vertex be b.
- The edge (a, b) is given weight $P_{a,b}$, the probability of the ASRW starting at a and ending at b.
- The edge (a, b) is given the constraints as follows for any assignment $\sigma : V \to \Sigma^{1+d+\cdots+d^t}$:

  For every edge (u, v) in the ASRW, $(\sigma'(a)_u, \sigma'(b)_v)$ should be satisfied in the old graph.

The problem with this is that we now have a complete graph on V, with the probabilities of choosing them as the weights. This is too costly since we need to do this reduction for $O(\log n)$ steps and we can't have anything more than $|E'| = O(1)|E|$. This problem howeever will be fixed later.

5.3 Lowerbounding gap'

Let $\sigma' : V \to \Sigma'$ be the best assignment for G'. We shall extract an assignment for G to get an upperbound on gap'.

$\sigma(u)$ is defined as follows:

- Do a BSRW starting at u. This generates a probability distribution $X(v)$ on the vertices reachable from u with a path of length atmost t.
- For all $a \in \Sigma$, let
$$P_a = \sum_{\sigma'(v)_u = a} X(v)$$
  Define $\sigma(u) = c$ such that $P_c = max_{a \in \Sigma} P_a$, the "most frequent opinion of u from other vertices".

Let F be the set of unsatisfied edges, we may throw in some more unsatisfied edges so that $gap = \frac{|F|}{|E|}$

Definition 6. *A $u \to v$ step in the verifier's $a \to b$ walk is said to be faulty if*

- $(u, v) \in F$
- $dist(a, u) \leq t$ *and* $\sigma'(a)_u = \sigma(u)$

Let N be the random variable that counts the number of faulty steps in the verifier's walks. And clearly $gap' \geq Pr[N \geq 0]$

Lemma 7.

$$E[N] \geq \frac{t}{8|\Sigma|^2} \frac{|F|}{|E|}$$

Proof. Let $(u, v) \in F$. Clearly it's enough to show that the expected value of $u \to v$ faulty steps (say N_{uv}) is bounded below by $\frac{t}{8|\Sigma|^2} \frac{1}{|E|}$

$$E[N_{uv}] = \sum_{k \geq 1} k \Pr[u \to v \text{ is faulty} | k \ (u \to v)\text{steps}] \Pr[k \ (u \to v)\text{steps}]$$

Claim 8. $\Pr[(u \to v)$ *is faulty*$|k \ (u \to v)$*steps*$] \geq \frac{1}{4|\Sigma|^2}$

Pf: The required probability is just checking

1. $dist(a, u) \leq t$ and $\sigma'(a)_u = \sigma(u)$
2. $dist(b, v) \leq t$ and $\sigma'(b)_v = \sigma(v)$

But these two are independant distributions given the promise that the edge (u, v) is traversed k times. So $\Pr[1 \vee 2] = \Pr[1] + \Pr[2]$

$$\Pr[\sigma'(b)_v = \sigma(v) | dist(b, v) \leq t] \cdot \Pr[dist(b, v) \leq t]$$

By our choice of $\sigma(u)$, $\Pr[\sigma'(b)_v = \sigma(v) | dist(b, v) \leq t] \geq \frac{1}{|\Sigma|}$
And

$$\begin{aligned} \Pr[dist(b, v) \leq t] &= \Pr[b \text{ is generated within } t \text{ steps}] \\ &= 1 - \left(1 - \frac{1}{t}\right)^t \\ &> 1 - \frac{1}{e} \\ &> \frac{1}{2} \end{aligned}$$

And together with the two bounds, the claim is done. □

And thus with this claim,

$$\begin{aligned} E[N_{uv}] &\geq \frac{1}{4|\Sigma|^2} E[\text{number of } (u,v) \text{ steps}] \\ E[\text{number of } (u,v) \text{ steps}] &= \sum_{k \geq 1} E[\text{number of } (u,v) \text{ edges} | k \text{ steps in walk}] \Pr[k \text{ steps in walk}] \end{aligned}$$

Now for any d-regular graph, after starting with the uniform distribution on the vertices, after an i step random walk, the probability of traversing any edge is $\frac{1}{2|E|}$ for every i. Hence,

$$\begin{aligned} E[\text{number of } (u,v) \text{ steps}] &= \frac{1}{2|E|t} \sum k \left(1 - \frac{1}{t}\right)^{k-1} \\ &= \frac{1}{2|E|t} \left(1 - \left(1 - \frac{1}{t}\right)\right)^{-2} \\ &= \frac{t}{2|E|} \\ \therefore E[N_{uv}] &\geq \frac{1}{4|\Sigma|^2} \cdot \frac{t}{2|E|} \end{aligned}$$

and summing over every edge $(u,v) \in F$, the lemma is proved. □

Before we go further in bounding gap', let us first prune the graph down.

5.4 Pruning down the graph

We have a complete graph because the verfier is able to look at all paths and hence we have all edges in G'. The natural thing to do is to cut down the verifier's walks.

Modified Verifier: Fix a parameter $B = ct$

- If the verifier makes more than B steps, put no constraint on (a, b).
- Otherwise use the same test as in the earlier veifier.

Throw away all unconstrained edges from G'. It's clear that the degree of this graph is bounded by $1 + d + \cdots d^B$ and $size(G') \leq c' size(G)$. Probabilities are rational numbers with the numerators and denominators depending on B and t alone and hence the weights(probabilities) can be removed by having multiple edges.

Now we shall argue that the bounds on gap'. Let σ' be the best assignment for G, we shall extract σ just as in the earlier case, by considering the

most frequent opinion. Again, we can throw some unsatisfied edges to make $gap = \frac{|F|}{|E|}$.

Definition 9. *A $u \to v$ step is said to be $faulty'$ if it is faulty and the verifier stops within B steps in that walk.*

Let N' be the random variable that counts the $faulty'$ steps, and let N_F be the random variable that counts the number of times the walk passes through F and let S be the number of steps. We need to argue that $\Pr[N' > 0]$ is large.

Theorem 10 (Second Moment Method)**.** *For any random variable N,*

$$\Pr[N > 0] \geq \frac{E[N]^2}{E[N^2]}$$

The proof of this theorem is left as an exercise.

Now for a similar lemma as in the earlier case.

Lemma 11.

$$E[N'] \geq \frac{t}{16|\Sigma|^2} \cdot \frac{|F|}{|E|}$$

Proof. $E[N'] = E[N] - E[N|S > B]\Pr[S > B]$ and we know that $\Pr[S > B] = \left(1 - \frac{1}{t}\right)^B$. And again by the same argument used earlier, we get

$$\begin{aligned} E[N|S > B] &= \left(\sum_{k \geq 1} (B+k) \frac{1}{2|E|} \left(1 - \frac{1}{t}\right)^{k-1} \frac{1}{t}\right) |F| \\ &= \frac{B+t}{2|E|}|F| \\ \therefore E[N|S > B]\Pr[S > B] &\leq e^{-B/t}\left(\frac{B+t}{2|E|}|F|\right) \end{aligned}$$

Now we can choose c appropriately and be done. □

Now, we need to bound $E[N'^2]$. Let χ_i be the indicator random variable that checks if the i^{th} step of the verifier is in F. It is clear that $N' \leq N_F$ and hence $E[N'^2] \leq E[N_F^2] = \sum E[\chi_i \chi_j]$. Hence

$$\begin{aligned} E[N'^2] &\leq 2\sum_{i,j} E[\chi_i \chi_j] \\ &\leq 2\sum_i \Pr[\chi_i = 1] \cdot \sum_{j \geq i} \Pr[\chi_j = 1 | \chi_i = 1] \end{aligned}$$

When $i = j$, $\Pr[\chi_j = 1 | \chi_i = 1] = 1$, otherwise, it is equal to

$$\Pr[\text{a walk starting from an endpoint on } F \text{ takes its } (j-i)^{th} \text{ edge in } F] \times \Pr[\text{the walk takes atleast } j - i \text{ more steps}]$$

The second term we know is equal to $\left(1 - \frac{1}{t}\right)^{j-i}$, the bound for the first term follows from the following expander lemma.

Lemma 12. *Let $G = (n, d, \lambda)$ expanderand $F \subseteq E$. Then the probability that a random walk with initial step in F will move in F at the l^{th} step is*

$$\leq \left(\frac{|F|}{|E|} + \left(\frac{\lambda}{d}\right)^{l-1}\right)$$

Proof. The proof is exactly like the proof of the mixing time lemma, same adjacency matrix trick. □

Hence taking $j - i = l$,

$$\begin{aligned} E[N'^2] &\leq 2\sum_i \Pr[\chi_i = 1]\left(1 + \sum_l \left(1 - \frac{1}{t}\right)^l \left(\frac{|F|}{|E|} + \left(\frac{\lambda}{d}\right)^{l-1}\right)\right) \\ &\leq 2\sum_i \Pr[\chi_i = 1]\left(1 + (t-1)\frac{|F|}{|E|} + O(1)\right) \end{aligned}$$

The $O(1)$ comes in because d and λ are constants which are fixed earlier. Now we can assume that $\frac{|F|}{|E|} \leq \frac{1}{t}$, otherwise we already have attained constant gap.

$$\begin{aligned} E[N'^2] &\leq O(1)\sum_i \Pr[\chi_i = 1] \\ &= O(1)E[N_F] \\ &\leq O(1)t\frac{|F|}{|E|} \end{aligned}$$

And now using our second moment method, we have

$$\Pr[N' > 0] \geq \frac{t}{O(1)}\frac{|F|}{|E|}$$

and we are through with gap-amplification.

Lectures on the PCP Theorems Course Instructor: V. Arvind

Lecture 3

Lecturer: Bireswar Das *Scribe: Ramprasad Saptharishi*

6 Overview

We are not at the last step of the subroutine that achieves the reduction to prove Dinur's Theorem. We would be applying the routine for $O(\log n)$ steps and hence this clearly means that the alphabet can't be as large as it is at the end of step 3. Our goal is to now reduce the alphabet size down to a constant without losing the amplification achieved.

We would need a lot of machinary for this step, the crux is the *Assignment Tester*.

7 Assignment Tester $\implies$ Alphabet Reduction

Definition 13. *A q-query assignment tester $AT(\gamma > 0, \Sigma)$ is a reduction algorithm P whose input is a boolean circuit Φ over X and outputs a system of constraints Ψ over X and a set of auxillary variables Y such that:*

- *Y takes values from Σ*
- *Each constraint $\psi \in \Psi$ depends on atmost q variables.*
- *For all $a : X \to \{0, 1\}$*
  - *If $\Phi(a) = 1$, then there exists an assignment $b : Y \to \Sigma$ such that $a \cup b$ satisfies all constraints $\psi \in \Psi$*
  - *If a is δ-far[2] from any satisfying assignment of Φ, then for every assignment $b : Y \to \Sigma$, $a \cup b$ violates atleast $\gamma\delta$ fraction of the constraints.*

But in order to use this in our constraint graphs, we need a 2-query assignment tester since each constraint is an edge in the constraint graph.

[2] by the relative hamming distance

And from the definition if the Φ was satisfiable then $gap = 0$ for the constraints. And if the Φ was not satisfiable, then every assignment is 1-far from a satisfying assignment (since there aren't any satisfying assignments) and hence the gap is atleast γ. This is just like the PCP reduction, except for the running time of the assignment tester, which could potentially be exponential.

But we can apply this assignment to only some small circuits of the graph, and help is reduce the alphabet size.

Theorem 14 (Composition Theorem)**.** *Let $P(\gamma > 0, \Sigma_0)$ be a 2-query assignment tester. Then for any constraint graph $\mathfrak{G} = (G, C, \Sigma)$ can be converted, in polynomial time, to another constraint graph $\mathfrak{G}' = (G', C', \Sigma_0)$ such that:*

- $size(G') \leq O(1) size(G)$
- $gap(G) = 0 \implies gap(G') = 0$
- *There exists a $\beta > 0$, depending only on P and $|G|$, such that $gap(G') \geq \beta \cdot gap(G)$*

Proof. The idea is to apply P on each constraint on the edges to transform the graph into another constraint graph. First we need to give P a circuit, so we shall encode the elements of Σ as binary strings by using an error correcting code.

Let $e : \Sigma \to \{0,1\}^l$ be a constant rate code, (i.e) $l = O(\log |\Sigma|)$ and minimum distance being $\rho = \frac{1}{4}$, that is $x \neq y \implies \Delta(e(x), e(y)) \geq \rho \cdot l$.

Our constraint graph is $(G = (V, E), \mathcal{C}, \Sigma)$, for each constraint $c \in \mathcal{C}$ on the variable u, v of G, define a new constraint on $[u] \cup [v]$, where $[u]$ is the encoding of u under e, as follows:

$\Phi_{u,v}(a, b) = 1$ if and only there exists $\alpha, \alpha' \in \Sigma$ such that $e(\alpha) = a$, $e(\alpha') = b$ and $c(\alpha, \alpha') = 1$ for all constraints c on the edge (α, α').

We can then interpret $\Phi_{u,v} : \{0,1\}^{2l} \to \{0,1\}$ as a boolean circuit and feed it to the 2-query assignment tester. The output obtained will now be a list of constraints $\Psi_{u,v}$, each dependant on just two variables. And hence this can be interpretted as a constraint graph over Σ_0, say $(G_{u,v} = (V_{u,v}, E_{u,v}), \mathcal{C}_{u,v})$ such that $[u] \cup [v] \subseteq V_{u,v}$. Our new constraint graph $(G' = (V', E'), \mathcal{C}', \Sigma_0)$ will just be these small constraint graphs pasted together.

- $V' = \bigcup_{e \in E} V_e$
- $E' = \bigcup_{e \in E} E_e$

- $\mathcal{C}' = \bigcup_{e \in E} \mathcal{C}_e$

Now we need to show that this graph proves the theorem. Firstly, the size bound of G is clear since each edge of G is blown up by the output of the assignment tester's output on the constraints. And since, with possible addition of multiple edges, each G_c is of equal size the size of G' is clearly $O(1)size(G)$; and is also clear that this can be done in polynomial time.

When $gap(G) = 0$, we have a satisfying assignment to our input circuit to the assignment tester. Hence by definition of the assignment tester, $gap(G') = 0$.

As for the other case, let $\sigma' : V' \to \Sigma_0$ be the best assignment. From this, extract an assignment $\sigma : V \to \Sigma$ as follows:

$$\sigma(u) = \arg\min_a(\sigma'([u]), e(a))$$

that is, pick the nearest codeword to the label. There is a catch, σ' should assign $\{0, 1\}$ values for $[u]$ for all $u \in V$, but that has to happen since any other value would only make more constraints unsatisfied.

Atleast $gap(G) \cdot |E|$ edges have to be violated in the old graph; we now want to count the number of violations in G'. Let (u, v) be violated by σ.

Claim 15. *$\sigma'([u]) \cdot \sigma'([v])$ is atleast $\frac{\rho}{4}$-far from any satisfying assignment for the constraint circuit $\Phi_{u,v}$ of the edge (u, v)*

Proof. Let $(e(a), e(b))$ be the satisfyign assignment for $\Phi_{u,v}$ that is closest to $\sigma'([u]) \cdot \sigma'([v])$; this forces $c(a, b) = 1$ for all constraints c on the edge. Hence, either $a \neq \sigma(u)$ or $b \neq \sigma(v)$ otherwise it would contradict the assumption that $(u, v) \in F$. Without loss of generality, let us assume that $a \neq \sigma(u)$. Then

$$\begin{aligned} \rho &\leq \Delta\left(e(a), e(e(u))\right) \\ &\leq \Delta\left(e(a), \sigma'([u])\right) + \Delta\left(\sigma'([u]), e(e(u))\right) \\ &\leq 2\Delta\left(e(a), \sigma'([u])\right) \end{aligned}$$

And since we have a concatenation of $\sigma'([u])$ and $\sigma'([v])$, it has to be atleast $\frac{\rho}{4}$-far from any satisfying assignment for $\Phi_{u,v}$ □

Hence the σ' violates atleast $\gamma \cdot \frac{\rho}{4}$ fraction of the constraints in each such $G_{u,v}$ and hence violates $gap(G)\gamma \cdot \frac{\rho}{4}$ fraction of constraints in G' and thus proves the theorem with $\beta = \gamma \cdot \frac{\rho}{4}$ □

8 In Search of an Assignment Tester

Once we have a 2-query assignment tester, then by the earlier theorem we are done. Infact, the following theorem tells us that finding a q-query assignment tester over $\Sigma = \{0,1\}$ would be enough.

Theorem 16. *Let P be a q-query $(\gamma, \{0,1\})$ assignment tester. Then we can onstruct a 2-query $(\frac{\gamma}{q}, \{0,1\}^q)$ assignment tester from P.*

Proof. On an input Φ over X, let P output Ψ as the set of constraints over the variables $X \cup Y$. We shall define the 2-query assignment tester as follows:

- The auxillary variables are $Y \cup Z$ where $Z = \{z_\psi | \psi \in \Psi\}$ over the alphabet $\{0,1\}^q$
- For each constraint $\psi \in \Psi$, we add q constraints as follows. Let $v_1, \cdots, v_q$ be the inputs that ψ depends on (repeat something if less). Add the constraint (z_ψ, v_i) for every $1 \leq i \leq q$ such that it is satisfied by (a, b) if the assignment a $(\in \{0,1\}^q)$ satisfies ψ and the value that a gives to v_i is b.

Hence, any assignment that satisfies everything in Ψ can be clearly extended to satisfy every constraint in the 2-query assignment tester's output. And since we are spreading every constraint over q 2-query constraints, an assignment that is δ-far from a satisfying assignment will violate $\frac{\gamma\delta}{q}$ fraction of constraints. □

We shall construct a 6-query assignment tester and this shall then give us a 2-query assignment tester over the alphabet Σ_0 such that $|\Sigma_0| = 64$. We would require a lot of machinary before that.

9 Linearity Testing

Definition 17. *A function $f : \{0,1\}^n \to \{0,1\}$ is said to be a linear function if there exists $S \subseteq [n]$ such that $f(x) = \oplus_{i \in S} x_i$*

This basically means that for all $x, y \in \{0,1\}^n$, $f(x \oplus y) = f(x) \oplus f(y)$.

For every $S \subseteq [n]$, let $\chi_S : \{0,1\}^n \to \{0,1\}$ be defined as $F_S(x) = \oplus_{i \in S} x_i$. Given an input function f, the linearity test tries to distinguish the two cases:

- f is equal to χ_S for some $S \subseteq [n]$
- f is "far" from every χ_S

9.1 The Blum-Luby-Rubinfeld Test

Definition 18. *Two functions f and g from are said to be δ-far if*

$$\Pr_x[f(x) \neq g(x)] \geq \delta$$

They are said to be δ-close if

$$\Pr_x[f(x) \neq g(x)] \leq \delta$$

BLR Test:
Input $f : \{0,1\}^n \to \{0,1\}$

1. Pick $x, y \in_R \{0,1\}^n$
2. Accept if $f(x \oplus y) = f(x) \oplus f(y)$, reject otherwise.

Claim 19 (BLR Completeness)**.** *If f is linear, BLR always accepts*

Proof. Clear! □

Claim 20 (BLR Soundness)**.** *If f is δ-far from any linear function, then BLR rejects with probability $\geq \delta$*

Proof. We shall do a fourier analysis on the function for the proof. First we shift our notation from $\{0,1\}$ to $\{-1,1\}$, thus each $\chi_S(x) = \prod_{i \in S} x_i$

$$\begin{aligned} \Pr[\text{BLR rejects}] &= \Pr_{x,y}[f(x)f(y) \neq f(xy)] \\ &= E_{x,y}\left[\frac{1 - f(x)f(y)f(xy)}{2}\right] \end{aligned}$$

It is easy to see that $\{\chi_S\}_{S \subseteq [n]}$ forms an orthonormal basis for the set of boolean functions under the following inner product definition:

$$\langle f, g \rangle = \frac{1}{2^n} \sum_x f(x)g(x)$$

And hence, every boolean function f can be written as

$$f(x) = \sum_S f_S \chi_S(x)$$

and the coefficients f_S, under this basis, are called the fourier coefficients. And by Parseval's identity, $\sum_S f_S^2 = 1$

$$\begin{aligned} E_{x,y}[f(x)f(y)f(xy)] &= E\left[\sum f_S\chi_S(x)\sum f_T\chi_T(y)\sum f_T\chi_T(xy)\right] \\ &= E\left[\sum f_S f_T f_U\chi_S(x)\chi_T(y)\chi_U(xy)\right] \end{aligned}$$

Now

$$\begin{aligned} E_{x,y}[\chi_S(x)\chi_T(y)\chi_U(xy)] &= E_{x,y}\left[\prod_{i\in S}x_i\prod_{j\in T}y_j\prod_{j\in U}x_k y_k\right] \\ &= E_{x,y}\left[\prod_{i\in S\Delta T}x_i\prod_{j\in T\Delta U}y_j\right] \\ &= E_x\left[\prod_{i\in S\Delta T}x_i\right]E_y\left[\prod_{i\in T\Delta U}y_i\right] \end{aligned}$$

And this will be non-zero only if $S\Delta T = T\Delta U = \phi \implies S = T = U$. Hence

$$\begin{aligned} E_{x,y}[f(x)f(y)f(xy)] &= \sum_S f_S^3 \\ &\leq \left(\max_S f_S\right)\left(\sum_S f_S^2\right) \\ &= \max_S f_S \end{aligned}$$

Also,

$$\begin{aligned} f_S &= \langle f,\chi_S\rangle \\ &= \frac{1}{2^n}\sum_x f(x)\chi_S(x) \\ &= \frac{1}{2^n}\left(\sum_{f(x)=\chi_S(x)}1-\sum_{f(x)\neq\chi_S(x)}1\right) \\ &= 1-2\Pr_x[f(x)\neq\chi_S(x)] \end{aligned}$$

Hence,

$$\begin{aligned} E_{x,y}[f(x)f(y)f(xy)] &\leq \max_S f_S \\ &= 1-2\min_S\Pr_x[f(x)\neq\chi_S(x)] \\ &= 1-2\min d(f,\chi_S) \\ &= 1-2\delta \end{aligned}$$

Hence

$$\Pr[\text{BLR rejects}] = E_{x,y}\left[\frac{1 - f(x)f(y)f(xy)}{2}\right] \geq \delta$$

□

Lectures on the PCP Theorems Course Instructor: V. Arvind

Lecture 4

Lecturer: Bireswar Das *Scribe: Ramprasad Saptharishi*

10 Overview

Last time we saw that the existence of a constant query assignment tester completed the alphabet reduction process. This class we shall construct a constant query assignment tester, and complete the proof of the PCP theorem.

11 A constant query assignment tester

Given a circuit Φ as input, we need to output a set of constraints with the properties that preserve gap. The first step that we would be doing is arithmetize circuits.

11.1 Arithmetizing Circuit-Sat

Introduce variables for every input bit and also for every gate. Let x, y be the input wires and z the output wire for some gates. Add the following constraints for that gate:

- **AND:** $xy - z = 0$
- **OR:** $x + y - xy - z = 0$
- **NOT:** $x + z = 1$

where all the operations are done over $\mathbb{F}_2$.

And to check if the output of the circuit is 1, if z is the variable of output gate, we have add the constraint $z - 1 = 0$.

We would now have one variable for every gate, apart from the inputs. Let us refer to the set of equations as $\{P_1, \cdots, P_m\}$.

Hence we now have a set of quadratic constraints to be checked to solve circuit satisfiability.

11.2 Testing Quadratic Systems

Given an assignment $a \in \{0,1\}^N$ for the set of quadratic equations, one could naively check by plugging in a into every constraint and check. But for this we would need to know the entire assignment, we are looking for a constant number of queries to test the system.

Hence instead of checking if $P_i(a) = 0$ for every i, we shall look at a random linear combination of them and check if the following is zero:

$$P_{\bar{r}}(a) \sum_{i=1}^{m} r_i P_i(a) = 0$$

Clearly, if each of the $P_i(a) = 0$, then the linear combination would be zero. And even otherwise, we have the following easy claim for the soundness of the test.

Claim 21. *If $P_i(a) \neq 0$ for some i, then*

$$\Pr_{\bar{r}}\left[\sum r_i P_i(a)\right] = 0$$

By collecting all linear and quadratic terms separately,

$$P_{\bar{r}}(a) = s_0 + \sum_{i=1}^{N} s_i a_i + \sum_{1 \leq i,j \leq N} t_{ij} a_i a_j$$

Definition 22. *Given an assignment $a \in \{0,1\}^N$, define the following functions.*

- $L(s) = \sum a_i s_i$
- $Q(t) = \sum a_i a_j t_{ij}$

Hence our test amounts to checking if $P_{\bar{r}}(a) = s_0 + L(s) + Q(t) = 0$. Recall the Hadamard encoding of a string x.

Definition 23. *For a vector $x = \{x_1, \cdots, x_r\}$ over a field $\mathbb{F}$, the Hadamard encoding of x, denoted by $Had(x)$ is given by:*

$$Had(x)(s) = \sum_{i=1}^{r} s_i x_i$$

The Hadamard code is the longest possible linear code that does not have any repeated symbols as it contains all possible linear combinations of x.

And the function Q defined above is the *Quadratic Function* encoding of the assignment a.

Definition 24. *Given a vector $x = \{x_1, \cdots, x_r\}$ in a field $\mathbb{F}$, the quadratic function encoding of x, denoted by $QF(x)$ is given by:*

$$QF(x)(t) = \sum_{1 \leq i,j \leq r} x_i x_j t_{ij}$$

Note that the quadratic function encoding of x is just the hadamard encoding of the vector $x \otimes x$.

The assignment tester will now be described as a Prover-Verifier protocol, since we are applying the assignment tester only on the edges (which are of constant size) in our graph we are not worried about the running time of the assignment tester.

Given a circuit, the verifier picks the random vector $\bar{r}$ and based on that the prover provides the satisfying assignment (if any), the table for L and the table for Q for the satisfying assignment. Now with just two queries (one for L and another for Q) the verifier can check if $P_{\bar{r}}(a) = 0$. But there is no guarantee that the tables that the prover provided was correct, we need some way to reject malicious provers.

The verifier needs to check for the following:

1. L is a linear function, the hadamard encoding of some $c \in \mathbb{F}_2^N$, and Q is a liner function on $\mathbb{F}_2^n$, the hadamard encoding of $C = (C_{ij})_{1 \leq i,j \leq N}$.

2. Q and L are tables for the same assignment c, that is, $C_{ij} = c_i c_j$. (this means that Q is a quadratic function encoding, not just a hadamard encoding of a vector)

3. The assignment that both the tables are referring to is indeed the assignment to the set of quadratic constraints.

The BLR test would check conditions 1 with few queries (with high probability). As for the other conditions, we would need some way of finding the actual value of L and Q though the tables could be slightly distorted; we need a way to self-correct.

The Self-Correct Algorithm: $SC(f, x)$

- Pick y at random
- Output $f(x \oplus y) \oplus f(y)$

Lemma 25. *If f is δ-close to a linear function L for some $\delta < \frac{1}{4}$, then for any $x \in \{0,1\}^n$ the algorithm $SC(f, x)$ computes $L(x)$ with probability atleast $1 - 2\delta$.*

Proof. Since y and $x \oplus y$ are distributed uniformly at random,

$$\begin{aligned} \Pr_y[f(y) \neq L(y)] &\leq \delta \\ \Pr_y[f(x \oplus y) \neq L(x \oplus y)] &\leq \delta \\ \therefore \Pr_y[SC(f, x) = L(x)] &\geq 1 - 2\delta \end{aligned}$$

□

Now for condition 2, pick random $s, s' \in \{0,1\}^N$ and the following must be true.

$$\begin{aligned} L(s)L(s') &= \left(\sum a_i s_i\right)\left(a_j s'_j\right) \\ &= \sum a_i a_j s_i s'_j \\ &= Q(s \oplus s') \end{aligned}$$

And this can be checked by the self-correction procedure.

And finally for condition 3, to check if a was the actual assignment, we need to check if the coefficient of s_i is a_i. And hence we can randomly pick and i and check if $SC(L, e_i) = a_i$. We shall now put the assignment tester together and then formally prove the completeness and soundness.

11.3 The Final Assignment Tester

Input

A boolean circuit over variables X and gates Y such that $|X| + |Y| = N$.

Initialization

Arithmetize the circuit to get a set $\mathcal{P} = \{P_1, \cdots, P_r\}$ of quadratic constraints over the variables $A = \{z_1, \cdots, z_n\}$. Let $\{z_1, \cdots, z_{|X|}\}$ correspond to the variables of X and the rest for the gates Y.

The Proof Provided

- The assignment $a = (a_1, \cdots, a_N) \in \{0,1\}^N$ for the variables A
- A table L
- A table Q

Verification

Step 1:

- Run BLR on L
- Run BLR on Q

Step 2: Pick s, s' at random and check if the following holds:

$$SC(L, s)SC(L, s') = SC(Q, s \otimes s')$$

Step 3: Pick a random vector $\bar{r}$ and compute the coefficients s_i and t_{ij} such that

$$P_{\bar{r}}(a) = s_0 + \sum_{i=1}^{N} s_i a_i + \sum_{1 \leq i,j \leq N} t_{ij} a_i a_j$$

and check if

$$s_0 + SC(L, S) + SC(Q, T) = 0$$

Step 4: Pick a random $i \in \{1, 2, \cdots, |X|\}$. Check if

$$SC(L, e_i) = a_i$$

11.4 Proof of Correctness

Completeness of the entire test is clear, an honest prover will convince the verifier with probability 1 since he would pass every step of the verifier's procedure.

Firstly, if L or Q is δ-far from a linear function, then by the soundness of the BLR test we know that step 1 would reject with probability $\geq \delta$. For step 2, we would be needing this simple lemma.

Lemma 26. *Given a non-zero matrix (atleast* 1 *non-zero entry)* M*, for a random choice of vectors* s, s'*,*

$$\Pr_{s,s'}[s^T M s' = 0] \leq \frac{3}{4}$$

Proof. Let M_{ij} be the non-zero entry, note that

$$s^T M s' + (s+e_i)^T M s' + s^T M(e_j + s') + (s+e_i)^T M(s'+e_j) = e_i^T M e_j = M_{ij}$$

And hence this is non-zero, atleast one of the the above has to be non-zero. And since they are all identically distributed, the lemma follows. □

Lemma 27. *If L is δ-close to $Had(c)$ and Q is δ-close to $Had(C)$ such that $C_{ij} \neq c_i c_j$ for some i, j, then step 2 rejects with probability atleast $\frac{1}{4} - 6\delta$*

Proof. Since L is δ-close to $Had(c)$, by our earlier lemma we know that with prob atleast $1 - 2\delta$, $SC(L, s) = L(s) = s^T c$, and similarly for Q.

And thus with probability atleast $1 - 6\delta$, the equality being tested in step 2 is

$$\begin{aligned} s^T c c^T s' &= s^T C s' \\ \implies s^T (cc^T - C) s' &= 0 \end{aligned}$$

And since $cc^T - C$ is a non-zero matrix by assumption, the earlier lemma bounds this probability by $\frac{3}{4}$. And hence, the lemma follows. □

Lemma 28. *If L is δ-close to $Had(c)$ and Q is δ-close to $QF(c)$ and if $P_j(c) \neq 0$ for some j, then step 3 rejects with probability atleast $\frac{1}{2} - 4\delta$*

Proof. With the accuracy of the self-correct routine, with probability atleast $1 - 4\delta$,

$$P_{\bar{r}}(a) = s_0 + \sum_{i=1}^{N} s_i a_i + \sum_{1 \leq i,j \leq N} t_{ij} a_i a_j$$

is equivalent to the check in step 3:

$$s_0 + SC(L, S) + SC(Q, T)$$

And from our earlier lemma, we know that if $P_i(c) \neq 0$ for some j, then with probability $\frac{1}{2}$ this above evalutes to 0. Hence with probability atleast $1 - 2\delta - \frac{1}{2}$

$$s_0 + SC(L, S) + SC(Q, T) \neq 0$$

and thus step 3 would reject. □

Theorem 29. *For a suitable bound on δ, if an assignment a to our initial circuit Φ was δ-far from the closest satisfying assignment to the circuit, then the test rejects wiht probability atleast $\frac{\delta}{8}$ irrespective of the contents of the tables.*

Proof. Firstly, if L or Q were δ-far from the nearest Hamming codeword, then step 1 rejects it with probability atleast δ. And if they do not refer to the same assignment, then step 2 would catch that with probability atleast $\frac{1}{4} - 6\delta$.And if a is not a satisfying assignment to $\mathcal{P}$, then with probability atleast $\frac{1}{2} - 4\delta$ we catch that in step 3.

If the assignment a_X(the restriction to just the input variables) was δ-far from any satisfying assignment for the circuit Φ, and in particular c_X(c being the satisfying assignment for $\mathcal{P}$). And with probability atleast $1 - 2\delta$, $SC(L, e_i) = c_i$. Since a_X is δ-far from c_X, then with probability δ, $a_i \neq c_i$ over the random choice of i in $\{1, 2, \cdots, |X|\}$ and hence with probability atleast $\delta(1 - 2\delta)$ the verifier rejects in step 4.

The total accumulated error can be shown to be less than $\frac{\delta}{8}$ for a suitable bound on δ. [3] □

12 The End Game

All that's left to be done is to derandomize this protocol, and since we don't care about any resource bounds, we can enumerate all possible random strings and do this. We can then output the set of constraints, such that each constraint depends on only a constant number of variables (6, I think; the BLR takes 6 queries, 24 is definitely an upper bound).

And hence for every edge in our old constraint graph, we now have exponentially many constraints, but we can do this without and hastles since it still only is a constant size increase in the size of G.

However, if we were to apply the 4 steps on the entire original graph, we would get an exponential sized proof. This was more or less the proof of the earlier result that $NP \subseteq PCP(n^3, O(1))$.

Now we have a constant query assignment tester, and hence alphabet reduction, and thus the PCP reduction required to prove Dinur's theorem with suitable choice of the parameter t to double the gap. And that completes the proof of the PCP theorem. □

[3] in class we did each of the steps with probability $\frac{1}{4}$ and for that $\delta < \frac{1}{28}$ was good enough

13 Brief Sketch of the Proof

1. Showing that the gap version of three colourability was NP-hard implied the PCP theorem. Hence we wanted to amplify the gap from $(1, 1 - 1/m)$ to $(1, s)$ for some s.

2. The PCP Reduction was done in 4 steps:

3. Degree reduction: We just spread every vertex of our graph into a cycle, and put an expander on every cloud.

4. Expanderize: Super-imposed an expander over this graph.

5. Gap Amplification: We used "lazy random walks" to amplify the gap. This on the other hand increased the alphabet size.

6. In order to reduce the alphabet size, we were on the lookout for a 2 query assignment tester. We then showed that any constant query assignment tester can be used to construct a 2-query assignment tester. The idea was to put this assignment tester on every edge, since edges were of constant size, running time of the assignment tester wasn't crucial.

7. In order to get a constant query assignment tester, we arithmetized circuits, to give is a set of quadratic constraints.

8. With the BLR tests and the self-correction procedure, we deviced a Prover-Verifier protocol where the verifier, which on derandomization gave us a constant query assignment tester.

9. The derandomized version is now used for every edge in our constraint graph, thereby reduces the alphabet size back to constant size with just a constant factor increase in the size and gap, which can be compensated for in the "Gap Amplification" process with the choice of t in hand.

10. And thus we have a way of doubling the gap with just a constant size increase in the graph size. This then allowed us to repeat this for $O(\log n)$ steps to get the gap from $(1, 1 - 1/m)$ to $(1, s)$, and thus proving Dinur's Theorem, and hence the PCP Theorem