PCPs and HDX - Lecture 10

Scribe: Neta Atzmon

January 17, 2017

1 Hardness of approximation

The basic PCP theorem is equivalent to hardness of approximation of constraint satisfaction problems, for example 3SAT, to within some constant factor. Recall that the value of a given system of constraints is the maximal fraction of constraints that is satisfiable by an assignment.

Theorem 1.1 (basic PCP theorem). *There exists some $\varepsilon_0 > 0$ and a polynomial time reduction that inputs a 3SAT instance φ and outputs a 3SAT instance φ' such that*

- *if $val(\varphi) = 1$ then $val(\varphi') = 1$*
- *if $val(\varphi) < 1$ then $val(\varphi') < 1 - \epsilon_0$*

Here $val(\varphi)$ is the fraction of clauses in φ that can be satisfied by any assignment of the variables.

Today, we will talk about a much stronger theorem that is known, that has to do with tight hardness of approximation.

The following theorem is due to Håstad [Hås01],

Theorem 1.2 (tight inapproximability for 3SAT - "strong PCP theorem"). *For every $\delta > 0$ there is a polynomial time reduction that inputs a 3SAT instance φ and outputs a 3SAT instance φ' such that*

- *if $val(\varphi) = 1$ then $val(\varphi') = 1$*
- *if $val(\varphi) < 1 - \varepsilon_0$ then $val(\varphi') < 7/8 + \delta$*

This shows that we cannot expect a polynomial time algorithm that approximates the number of satisfied clauses in a 3SAT instance to within any factor better than 7/8. Why 7/8? because of random assignment satisfies this fraction of clauses in expectation, and this can be made into a polynomial time deterministic algorithm.

This theorem is part of a much more extensive theory of understanding the threshold of approximation for any constraint satisfaction problem, and more generally for any NP hard problem. Håstad proved similarly optimal hardness-of-approximation results for several other problems, including the following inapproximability theorem for 3LIN. Today we will discuss these two problems as flagship examples.
Recall that a 3LIN formula is a collection of linear equations each with 3 variables, over GF(2).

Theorem 1.3 (tight inapproximability for 3LIN - "strong PCP theorem"). *For every $\delta > 0$ there is a polynomial time reduction that inputs a 3SAT instance φ and outputs a 3LIN formula ψ such that*

- *if $val(\varphi) = 1$ then $val(\psi) = 1 - \delta$*

- *if $val(\varphi) < 1 - \varepsilon_0$ then $val(\varphi') < 1/2 + \delta$*

Notice that similarly to 3SAT, this is a tight inapproximability result, since 3LIN can be approximated in polynomial time to within a factor of 1/2, again because of the expected fraction of linear equations satisfied by a random assignment is 1/2.

2 Outline of proof

The proof of the above "strong PCP theorem" consists of two main steps, not unlike the proof of the basic PCP theorem.

- Large gap amplification: a reduction from an initial PCP instance φ to a new constraint satisfaction problem H such that
  - if $val(\varphi) = 1$ then $val(H) = 1$
  - if $val(\varphi) < 1 - \varepsilon_0$ then $val(H) < \delta$

  This reduction is based on the so-called parallel repetition transformation, or, possibly more appropriately, is just a direct product construction. We will see below that describing the transformation is very easy. Its correctness boils down to the parallel repetition theorem of Raz.

- Alphabet reduction, using an optimal PCP gadget called *the long code.* The optimal PCP g gadget is essentially a PCP theorem, where the PCP proof is an encoding of the original proof using a code that is called the long code. This code is locally testable, even when the verifier is restricted to making 3LIN queries[1]. Moreover, the constraints specified by the test constitute more than just a constraint expander. Whereas in a constraint expander we are promised global structure when at least $1 - \epsilon$ fraction of the constraints are satisfied, here, global structure is derived even when $1/2 + \delta$ fraction of the constraints are satisfied.

3 Large gap amplification

Definition 3.1 (LabelCover)**.** *A LabelCover Instance is a bipartite graph $G = (A \cup B, E)$ with a label set Σ such that each vertex in $A \cup B$ is to be assigned a label from the label set, and every edge $uv \in E$ has a constraint $C_{uv} \subseteq \Sigma \times \Sigma$.*
A LabelCover Instance is said to have the "projection property" if each constraint is a function from Σ on to itself, i.e for every label $a \in \Sigma$ for u, C_{uv} accepts exactly one label for v.

Theorem 3.2 (Hardness of label cover)**.** *For every $\delta > 0$ there is a finite set of labels Σ, $|\Sigma| = poly(\delta)$, and a polynomial time reduction that inputs a 3SAT formula φ and outputs a label cover instance H such that*

- *if $val(\varphi) = 1$ then $val(H) = 1$*
- *if $val(\varphi) < 1 - \varepsilon_0$ then $val(H) < \delta$*

Let us describe the transformation converting φ into H.

First we move from φ to a bipartite constraint graph which is the clause versus variable version of φ.

[1]In fact, when proving hardness of approximation for any other CSP we still rely on the same encoding and only vary the test.

Let $C = \{C_1, ..., C_m\}$ be the clauses and $V = \{v_1, ..., v_n\}$ be the variables of φ. We define the bipartite graph $G = (C \cup V, E)$ where there is an edge between a variable v_i and a clause C_j if v_i appears in C_j. Notice that each clause has exactly seven assignments that satisfy it. So we can take the alphabet to be of size 7, such that each symbol translates to a three bits valid assignment to the clauses' variables. The variable vertices are assigned with a $\{0,1\}$-labeling.

The constraint for any edge (C_j, v_i) is satisfied iff the label assignments are consistent, i.e if the restriction of the label assigned to C_j to the variable v_i equals to the label assigned to v_i.

We claim that:

- if $val(\varphi) = 1$ then $val(G) = 1$
- if $val(\varphi) < 1 - \varepsilon_0$ then $val(G) < 1 - \varepsilon_0'$

The completeness part is easy to see: if there is an assignment that satisfies all the clauses, we simply label the vertices according to the satisfying assignment, and thus all the edge constraints are satisfied.
Soundness: Every label assignment to the variables vertices in G corresponds to an assignment of φ and thus fails to satisfy at least ε_0 fraction of the clauses. Every vertex in G that corresponds to such violated clause is assigned (by definition) with a label that satisfy it. Therefore it must disagree with at least one of its variables. This means that every label assignment to the graph violates at least an $\varepsilon_0/3$ fraction of the edge-constraints.

Next we take the direct product of G.

Definition 3.3 (Direct Product/Parallel Repetition).
Given a LabelCover instance $G = (A \cup B, E)$, with a label set Σ and edge-constraints $\{C_{uv}\}_{uv \in E}$, we define the t power of G: $G^{\otimes t} = (A^{\otimes t} \cup B^{\otimes t}, E^{\otimes t})$, with a label set Σ^t, where there is an edge between $(u_1, ..., u_t) \in A^{\otimes t}$ and $(v_1, ..., v_t) \in B^{\otimes t}$ iff $u_i v_i \in E$ for all $1 \le i \le t$.
For each edge in $G^{\otimes t}$ the constraint defined naturally as the "AND" constraint over all the coordinates. That is $C_{\overline{uv}}((a_1, ..., a_t), (b_1, ..., b_t))$ accepts iff $C_{u_i v_i}(a_i, b_i)$ accepts for all $1 \le i \le t$.

The parallel repetition theorem of Raz implies that given a LabelCover instance G, for every $\delta > 0$ there is a constant t, such that $val(G^{\otimes t})$ is at most δ. In other words, we can construct a LabelCover instance with value arbitrarily close to 0. By taking H to be $G^{\otimes t}$, we get the desired result of hardness of label cover stated above.

4 The Long Code

In previous lectures we introduced the Hadamard code. This is the code that maps a message $m \in \{0,1\}^n$ to a string representing the values of all linear functions on m:
$H(m) = \{< m, a > \pmod 2\}_{a \in \{0,1\}^n}$. We saw that this code is locally testable.

For the alphabet reduction step in the proof of the PCP theorem we had to use a more general code, the quadratic functions encoding. The QF code maps a message $m \in \{0,1\}^n$ to a string representing the values of all quadratic functions on m: $QF(m) = \{m^t A m \pmod 2\}_{A \in \{0,1\}^{n \times n}}$.

For the alphabet reduction step in the proof of the strong PCP Theorem, Håstad used a generalization of these codes, called the long code.
The long code of an input massage $m \in \{0,1\}^n$ is a string of length 2^{2^n} where the coordinates correspond to all possible functions $f : \{0,1\}^n \to \{0,1\}$: $LC(m) = \{f(m)\}_{f:\{0,1\}^n \to \{0,1\}}$.
We refer to $\{0,1\}^n$ as Σ. So, the long code for a certain message $a \in \Sigma$ is the function
$A : \{0,1\}^{|\Sigma|} \to \{0,1\}$ such that $A(x) = x_a$.

The long code is very robustly testable in the sense that it has a 3CNF test that has the "structure versus randomness" behavior: any function that passes this test with probability slightly more than random, must already have a global structure.

Consider for example the following 3CNF test:

1. Pick $x, y \in \{0,1\}^{|\Sigma|}$ uniformly and independently.
2. Set $z = x + y$
3. Accept iff $A(z) = A(x) + A(y)$

If A is a correct long code, then the test accepts with probability 1. A random A is accepted with probability 1/2 and thus this acceptance probability does not have any implications on the structure of A. However, any A that is accepted with probability $1/2+\delta$ must already have some special structure. In fact, the above test is exactly the BLR test we saw before, for the Hadamrd code. Thus, every linear function passes this test with probability 1. This does not serve our purpose of distinguishing whether a function is a long code or "very different" from a long code (in some sense which we will not describe at this course). For this purpose, Håstad used a similar test with a perturbation. He gave up perfect completeness and allowed for a small probability of rejecting a correct long code.

1. Pick $x, y \in \{0,1\}^{|\Sigma|}$ uniformly and independently.
2. Pick $\mu \in \{0,1\}^{|\Sigma|}$ by choosing $\mu_i = 0$ with probability $1-\epsilon$ for each $i \in \Sigma$ independently.
3. Set $z = x + y + \mu$
4. Accept iff $A(x) + A(y) + A(z) = 0$

In this test, a correct long code is accepted with probability $1-\varepsilon$: Suppose A is the long code of some $a \in \Sigma$. Then, $A(x) + A(y) + A(z) = \mu_a$ which, by definition, equals 0 w.p $1-\varepsilon$.

So, long codes are still accepted with probability close to 1, but functions that are "very different" from long codes are rejected with significant probability.

5 Combining the two

In order to reduce the alphabet size of the LabelCover instance, we will ask the prover to encode the proof using the long code. We would then need to check that the encodings are correct long codes, and in addition, that the un-encoded assignments satisfy the constratint. Håstad's test combines these two checks into one check, with only 3 queries.

It is easier to explain this if we first assume some extra structure on the label cover instance. namely, that it is a unique game.

Definition 5.1. *A LabelCover instance is a UniqueGames instance if all of the constraints are permutations, i.e. for each edge $uv \in E$, the constraint $\pi_{uv} \subset \Sigma \times \Sigma$ can be written as $\pi_{uv} = \bigcup_{a\in\Sigma}((a, \sigma(a))$ where σ is some permutation over Σ.*

Conjecture 5.2. *(Unique Games Conjecture - UGC - due to Subhash Khot). For every $\delta > 0$, given an instance of UniqueGames, it is NP-hard to decide if it has value at least $1-\delta$ or at most δ.*

We will show a reduction that given a UniqueGames instance H constructs a 3-LIN formula ψ such that the following holds:

- if $val(H) = 1-\delta$ then $val(\psi) = 1$
- if $val(H) < \delta$ then $val(\psi) < 1/2 + poly(\delta)$

The reduction from UniqueGames proceeds by replacing each vertex v in the graph H by a "cloud" of $2^{|\Sigma|}$ Boolean variables whose assignment is viewed as a Boolean function $A_v : \{0,1\}^{|\Sigma|} \to \{0,1\}$. This is equivalent to asking the prover to encode each symbol from Σ assigned to a vertex by the long code. Each constraint π_{uv} is replaced by a bunch of linear equations on the variables in the clouds of u and of v that is best described as a randomized process as follows

1. Pick a random edge uv.
2. Pick $x, y \in \{0,1\}^{|\Sigma|}$ uniformly and independently.
3. Pick $\mu \in \{0,1\}^{|\Sigma|}$ by choosing $\mu_i = 0$ with probability $1 - \epsilon$ for each $i \in \Sigma$ independently.
4. Set $z = x + y + \mu$
5. Accept iff $A_u(x) + A_v(y) + A_v(z \circ \pi) = 0$,
   where $z \circ \pi$ is the permuted string $z_{\pi(1)}, z_{\pi(2)}, ..., z_{\pi(|\Sigma|)}$.

The main idea is that if there exists an assignment that satisfy a large fraction of the equations (i.e the above test accepts with high probability), then we can "decode" the supposed Long Codes and deduce a labeling for the UniqueGames instance that achieves a non-negligible value depending on δ, ε. This gives a contradiction provided that, to begin with, the Val of the UniqueGames (LabelCover) instance is chosen to be sufficiently small.

References

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.