(67902) Topics in Theory and Complexity Jan 25, 2007

Lecture 13

Lecturer: Irit Dinur *Scribe: Lior Aronshtam*

1 Overview

we are in the middle of showing a polynomial reduction from a constraint graph G to a constraint graph $G\prime$ such that:

- $unsat\,(G) = 0 \Rightarrow unsat\,(G\prime) = 0$
- $unsat\,(G) > 0 \Rightarrow unsat\,(G\prime) > \alpha$

the reduction is done by running $\log n$ times a transformation (linear in size and time) from a constraint graph G to a constraint graph $G\prime\prime$ such that:

- $unsat\,(G) = 0 \Rightarrow unsat\,(G\prime\prime) = 0$
- $unsat\,(G) > 0 \Rightarrow unsat\,(G\prime\prime) > \min\,(\alpha, 2unsat\,(G))$

the transformation is constructed of three parts:

1. preprocessing
2. powering
3. reducing Σ

today we will start the third part of reducing Σ.

2 Reducing Σ

1. **Input** a constraint graph G with:
   - $v_1, v_2, , , v_n$ vertexes (variables)
   - $c_1, c_2, , , c_m$ binary constraints
   - and a large Σ .
2. **objective** turn G into a graph $\mathrm{H}((V\prime, E\prime)\,, \Sigma\prime, C\prime)$ where $\Sigma\prime$ is small and:
   - $unsat\,(G) = 0 \Rightarrow unsat\,(H\prime) = 0.$
   - $unsat\,(G) > 0 \Rightarrow unsat\,(H\prime) > c\ unsat\,(G).$

3. **Intuition - encoding the variables, and the constraints** Suppose we encode each variable v by $\log|\Sigma|$ variables according to its binary representation, and each constraint will read $2\log\Sigma$ variables. This lowers the alphabet, but is not good enough since we want each constraint to read a constant number of variables (which does not depend on $|\Sigma|$)[1].

   Instead, the idea will be to also encode the constraints. We will encode each constraint C by a set $\{c\}$ of constraints and if the original constraint C was satisfied than each constrain $c_i \in \{c\}$ will be satisfied. However, this is hard to accomplish the next example demonstrates why:

   Example

   Suppose the constraint on v_1, v_1 is satisfied iff the sum of the number of 1's in their binary representation is even. Checking this constraint requires reading the entire binary representation of both variables.

   Therefore, to achieve our task, we will encode each variable v_i by a more suitable encoding, not the binary representation. This encoding should enable us to replace each original constraint $\Psi : \Sigma \times \Sigma \to \{0,1\}$ by a set of constraints $\{C_\psi\}$ with the property that

   - Ψ is satisfied $\Rightarrow$ unsat$\{C_\psi\} = 0$
   - Ψ is not satisfied $\Rightarrow$ unsat$\{C_\psi\} > \alpha$

   This reduction has a very similar flavor to the reduction we are seeking in order to prove the PCP theorem. Is this a circular argument?

   The PCP theorem we wanted to transform SAT into a constraint graph, by taking each formula ϕ (input to the SAT problem) and transforming it to a set of constraints C_ϕ such that:

   $\phi \in$ SAT $\Rightarrow$ unsat$(C_\phi) = 0$
   $\phi \notin$ SAT $\Rightarrow$ unsat$(C_\phi) > \alpha$

   there exists such a reduction but it is of exponentia size. thus we can not apply this reduction in the PCP theorem but we can apply it here since the reduction is applied on each constraint by itself, and the size of the reduction on each constraint depends only on Σ which is a constant. so if we have n constraints and the size of the reduction on one constraint is $M(|\Sigma|)$ the total size of the reduction is $O(nM(|\Sigma|))$.

   Conclusion we would like to find a way to encode each v_i such that:

   (a) the encoding is interesting

   (b) two encodings of two variables fit there constraint.

4. **Local testing** $p \subseteq \{0,1\}^n$ is the set of strings that fulfill an attribute. we are looking for a way to know if $x \in p$ without reading all the bits in x.

   Definition 1 *T is a local testing algorithm for the attribute $p \subseteq \{0,1\}^n$ if:*

   - *T is a random algorithm and the number of bits it reads from x is $\leq q$.*
   - $x \in P \Rightarrow \Pr(T^x = 1) = 1$
   - $x \notin P \Rightarrow \Pr(T^x = 0) > \alpha$

[1] We remark that for a constant q it is an easy exercise to transform a set of q-ary constraints over a $\Sigma = \{0,1\}$ to a set of binary constraints over a $\Sigma = \{0,1\}^q$, loosing only a constant in the unsat value.

this definition is impossible since if we have $x, x\prime \in \{0,1\}^n$ such that they are different in only one bit, and $x \in P$ and $x\prime \notin P$, the probability that T will read that bit is not constant. so we will define it again with a slight change.

Definition 2 *T is a local testing algorithm for the attribute* $p \subseteq \{0,1\}^n$ *if:*

- *T is a random algorithm and the number of bits it reads from x is* $\leq q$.
- $x \in P \Rightarrow \Pr(T^x = 1) = 1$
- $dist(x, P) > \delta \Rightarrow \Pr(T^x = 0) > \frac{\delta}{c}$

not all attributes have local testing algorithms for example:
$P = \{x \in \{0,1\}^n \mid \sum x_i mod 2 = 0\}$

Question does there exist an interesting P with a local testing algoritm? with $q = o(1)$? by interesting we mean that P is an error correcting code,that is to say the distance between any two strings in P is very big.

Question intuition it is not trivial that such a code exists, for example if we would look at a random code (that we proved in a previous lecture is an error correcting code) and want to know if by reading q bits from a string we can determine whether it belongs to the code or not. it is obvious that for every qary if all the 2^q options appear in the code the algorithm after reading that qary will always return yes, so a q is worth reading only if one of the options does not appear in the code.
we would like a code who's dual code has many short words, the Hadamard code is of this kind.

5. The Hadamard code

Remainder

$H : \{0,1\}^l \to \{0,1\}^{2^l}$

$a \in \{0,1\}^l \; H(a) \in \{0,1\}^{2^l}$

$\forall x \in \{0,1\}^l$

$H(a)[x] = \sum_{i=0}^{l-1} a_i x_i \; mod 2$

for example:

$x_1 = (1000...0) \; H(a)[x_1] = a_1$

$x_2 = (0100...0) \; H(a)[x_2] = a_2$

$x_3 = (1100...0) \; H(a)[x_3] = a_1 + a_2$

$x_4 = (111...1) \; H(a)[x_1] = \sum a_i$

Algorithm 3 *the local testing algorithm T for the attribute* $L = \left\{H(a) \in \{0,1\}^{2^l} \mid a \in \{0,1\}^l\right\}$

- ***Input*** *a string* $w \in \{0,1\}^{2^l}$.

- ***procedure:***
  (a) *choose a random* $x \in \{o,1\}^l$
  (b) *choose a random* $y \in \{o,1\}^l$
- ***Output*** *we will answer yes iff* $w(x) \oplus w(y) = w(x+y)$

Claim 4 *T is a local testing algorithm with* $q = 3$
Proof

- *if* $w \in L \Rightarrow \Pr(T^w = 1) = 1$ *was proved in exercise 1.*
- *if* $dist(w, L) = \delta > 0 \Rightarrow \Pr(T^w = 0) > \min\left(\frac{\delta}{2}, \frac{2}{9}\right) = \frac{\delta}{5}$

(a) ***step 1***
we will define $\tilde{w}$ *an improvement of* w *this way:*

$\forall x$ *if* $\Pr(w(y) = w(x+y) = 0) > \frac{1}{2} \Rightarrow \tilde{w}(x) = 0$
else $\Rightarrow \tilde{w}(x) = 1$

(b) ***step 2***

Claim 5 $\Pr_{x,y}(T^w = 0) \geq \frac{1}{2} dist(w, \tilde{w})$

Proof $\Pr_{x,y}(T^w = 0) = \Pr_{x,y}(w(x) \neq w(y) + w(x+y)) \underbrace{\geq}_{joint probability}$
$\Pr_x(w(x) \neq \tilde{w}(x)) \Pr_y(w(x) \neq w(y) + w(x+y) \,|\, w(x) \neq \tilde{w}(x))$

$= \Pr_x(w(x) \neq \tilde{w}(x)) \Pr_y(\tilde{w}(x) = w(y) + w(x+y) \,|\, w(x) \neq \tilde{w}(x))$

$\geq dist(w, \tilde{w}) \frac{1}{2}$ ■

(c) ***step 3***

Claim 6 *if* $\left(\Pr(T^w = 0) > \frac{2}{9}\right)$ *than* $\tilde{w} \in L$

Proof
$\forall x \in \{0,1\}^l$ *denote* $P_x = \Pr_y(\tilde{w}(x) = w(y) + w(x+y))$
by its definition $\frac{1}{2} \leq P_x \leq 1$.

Claim 7 $\forall x P_x > \frac{2}{3}$

Proof
$\Pr_{y_1,y_2}{}^2(w(y_1) + w(x+y_1) = w(y_2) + w(x+y_2))$
$= P_x^2 + (1-P_x)^2 = \Pr_{y_1,y_2}(w(y_1) + w(x+y_2) = w(y_2) + w(x+y_1))$
denote $y\prime = y_1$ *and* $x\prime = y_1 + y_2 + x$
since $\Pr(T^w = 0) < \frac{2}{9}$ *than*
$\Pr(w(y\prime) = w(x\prime + y\prime) = w(y_1 + y_2 + x)) \geq 1 - \frac{2}{9}$

[2] y_1 and y_2 are chosen in an independent way.

so the $\Pr(w(y_1)+w(x+y_2)=w(y_1+y_2+x))$
$=\Pr(w(y_2)+w(x+y_1)=w(y_1+y_2+x))\geq 1-\frac{2}{9}$
by using u.b we get $\Pr_{y_1,y_2}(w(y_1)+w(x+y_2)=w(y_2)+w(x+y_1))>1-\frac{2}{9}-\frac{2}{9}=\frac{5}{9}$
$\Rightarrow {P_x}^2+(1-P_x)^2>\frac{5}{9}$
$\Rightarrow P_x>\frac{2}{3}$ ■

we will now prove that $\forall x,z\tilde{w}(x)+\tilde{w}(z)=\tilde{w}(x+z)$ *thus* $\tilde{w}\in L$

i. $\Pr_y(\tilde{w}(x)=w(y)+w(x+y))>\frac{2}{3}$

ii. $\Pr_y(\tilde{w}(z)=w(y)+w(z+y))>\frac{2}{3}$

iii. denote $y\prime=y+z$. $\Pr_y(\tilde{w}(x+z)=w(y\prime)+w(x+z+y\prime))>\frac{2}{3}$

so there exists one y who satisfies a,b and c will denote that y as $\hat{y}$ *and we get:*

$(x)+\tilde{w}(z)=w(\hat{y})+w(\tilde{x}+\hat{y})+w(\hat{y})+w(z+\hat{y})$

$=w(\hat{y}+z)+w(x+\hat{y})=w(y\prime)+w(x+z+\hat{y}+z)$

$=w(y\prime)+w(x+z+y\prime)$

$=\tilde{w}(x+z)$■

so if $dist(w,L)=\delta$ *and* $\Pr(T^w=0)<\frac{2}{9}$

$\Pr(T^w=0)\geq\frac{1}{2}dist(w,\tilde{w})\geq\frac{1}{2}dist(w,L)=\frac{\delta}{2}$

■

Theorem 8 *there exist codes* E_1 *and* E_2*:*
$E_1:\Sigma\rightarrow\{0,1\}^{L_1}$ *that encodes each variable V to a set of binary variables* $[v]$.
and $E_2:\Sigma\times\Sigma\rightarrow\{0,1\}^{L_2}$ *that encodes each constraint C to a set of binary constraints* $[c]$.
that are error correcting codes with a relative distance $>\frac{1}{3}$.

and three local testing algorithms T^1,T^2,T^3*:*

- T^1*- locally checks if* $[v]$ *is a legal word in the range of* E_1.
- T^2*-locally checks if* $[c]$ *is a legal word in the range of* E_2 *and that* $[c]$ *has the attribute:* $P=\{E_2((a_1,a_2))|C(a_1,a_2)=1\}$.
- T^3*- locally checks if the input* $[v]$,$[c]$ *(where v is a variable in c) is consistent.*

we will prove the theorem god willing next week for now we will assume it is correct, and define the next algorithm.

Algorithm 9
- ***Input:*** $\bigcup_{c\in C}[c]\cup\bigcup_{v\in V}[v]$ *of a constraint graph* $G(V,E)$.
- ***Procedure****:*

(a) *choose* $i \in \{1,2,3\}$.

(b)
- $i = 1$ *chooses a random* v *and runs* T^1 *on* $[v]$.
- $i = 2$ *chooses a random* c *and runs* T^2 *on* $[c]$.
- $i = 1$ *chooses a random* c *and a random* v *from* c*'s two variables and runs* T^3 *on* $[c]$ *and* $[v]$.

Remark we can transform the algorithm into a set of constraints on the variables $\bigcup_{c \in C} [c] \cup \bigcup_{v \in V} [v]$,by a serial move upon all the random options.
denote this set of constraints $G \circ P$

Lemma 10 *if* $unsat\,(G) = 0$ *than* $unsat\,(G \circ P) = 0$

Lemma 11 *if* $unsat\,(G) = \delta > 0$ *than* $unsat\,(G \circ P) > \frac{\delta}{c}$