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

Lecture 9: Proof of PCP Theorem - The Final Piece

10/31/2005

Lecturer: Venkat Guruswami *Scribe: Raghavendra Prasad*

1 Overview

The proof of PCP presented so far is complete except for one last piece - The Assignment Tester (AT). In today's lecture, we use the techniques developed in the previous class to design an assignment tester. With this, we complete the proof of the PCP theorem. We will also take a step back to reflect upon the overall structure of the proof, and the tools it used.

As this lecture relies heavily on the techniques of previous class, we recall a few definitions and results here.

Definition 1.1. *An q-query assignment tester AT $(\gamma > 0, \Sigma_0)$ is a reduction algorithm P whose input is a Boolean Circuit Φ over boolean variables X and output is a system of constraints Ψ over X and a set Y of auxilary variables, such that*

- *Variables in Y take values in Σ_0.*
- *Each $\psi \in \Psi$ depends on at most q variables in $X \cup Y$.*
- *For any assignment a to variables X we have*
  - COMPLETENESS: *If $\Phi(a) = 1$ then there exists an assignment b to variables in Y, such that $a \cup b$ satisfy every constraint $\psi \in \Psi$.*
  - SOUNDNESS: *If assignment $a : X \to \{0, 1\}$ is δ-far from every satisfying assignment to Φ, then for any $\forall b : Y \to \Sigma_0$ at least $\gamma\delta_0$ fraction $\psi \in Psi$ are violated by $a \cup b$.*

Remark: The soundness condition is implied by the following statement: $\exists\delta_0 > 0$ and $\gamma_0 > 0$ such that for $\delta \leq \delta_0$, if assignment $a : X \to \{0, 1\}$ is δ-far from every satisfying assignment to Φ, then for any $\forall b : Y \to \Sigma_0$ at least $\gamma_0\delta_0$ fraction $\psi \in Psi$ are violated by $a \cup b$. Indeed this implies the above definition with $\gamma = \gamma_0\delta_0$. We will use this form to establish the soundness.

1.1 Arithmetizing a Circuit

Any circuit over boolean variables can be reduced to a system of quadratic equations over $\mathbb{F}_2$, such that there exists a satisfying assignment to the circuit, iff there is a solution to the system of quadratic equations. This reduction from circuits to polynomial equations known as Arithmetization of Circuits, is a recurring technique in complexity theory.

Let us assume we have a boolean circuit C with input variables X and gates Y. Arithmetization of C in to quadratic constraints can be done as follows

- Introduce a variable $z_i \in A$ for each input variable X and for each gate Y. (i.e., $|A| = |X| + |Y|$).
- For each gate $y_i \in Y$ introduce a quadratic constraint P_i, to model the relation between input and output.
  - An AND gate with input z_i, z_j and output $z_k : z_i \cdot z_j - a_k = 0$.
  - An OR gate with input z_i, z_j and output $z_k : z_i + z_j + z_i \cdot z_j - z_k = 0$.
  - A NOT gate with input z_i and output $z_j : z_j + z_i - 1 = 0$.
- Introduce a constraint to force the output of the circuit to $1 : z_k - 1 = 0$ where z_k is the variable corresponding to the output gate of the circuit.

1.2 Assignment Tester

Given a boolean circuit Φ over variables X with gates Y, we first arithmetize Φ to obtain a system of quadratic equations $\mathcal{P} = \{P_1 \dots P_m\}$ over variables $A = \{z_1 \dots z_N\}$ where $N = |X| + |Y|$. Therefore the task of an assignment tester is reduced to checking a solution for a system of quadratic equations, using very few queries.

2 Testing a Quadratic System with Few Queries

The naive method is to check if an assignment $a : A \longrightarrow \{0, 1\}$ is a solution to a system of quadratic equations $\mathcal{P} = \{P_1, P_2 \dots P_m\}$, is to substitute $a_j = a(z_j)$ for various j in each P_i and check if $P_i = 0$. But observe that in order to do this, we need to know the entire assignment a, which would require a large number (not constant) of queries. Therefore, the first objective is to reduce the number of queries in this naive test.

Instead of checking that $P_i(a) = 0$ for each i, let us take a random linear combination of the numbers $P_i(a)$ and check if it is zero.

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

where $r_i \in \mathbb{F}_2 = \{0, 1\}$ are chosen at random independent of each other.

Observe that the addition in the above linear combination is over $\mathbb{F}_2$. So it is possible that the linear combination sums up to zero, although each of the terms are non-zero. But we show that if the P_i are indeed non-zero, then with probability $1/2$, their linear combination is non-zero. Therefore this test alone has a completeness of 1 and soundness $\frac{1}{2}$.

Lemma 2.1. *If not all $P_i(a), i = 1 \dots m$ are zero, then*

$$Pr[\sum_{i=1}^{m} r_i P_i(a) \neq 0] = \frac{1}{2}$$

Proof: It is known that there exists an i such that $P_i(a) = 1$. Without loss of generality, we assume $P_m(a) = 1$. Then

$$\sum_{i=1}^{m} r_i P_i(a) = \sum_{i=1}^{m-1} r_i P_i(a) + r_m P_m(a) \tag{1}$$

Observe that irrespective of the value of $\sum_{i=1}^{m-1} r_i P_i(a)$, the entire sum takes values $\{0, 1\}$ for the two choices of r_m. Therefore, in one of the two choices of r_m the sum $\sum_{i=1}^{m} r_i P_i(a)$ is not zero. Hence the result follows. □

So instead of testing $P_i(a) = 0$ for every i, we test that $P_{\vec{r}}(a) = 0$ for a random vector $\vec{r}$. However we still have a problem of computing $P_{\vec{r}}(a)$ without looking at the entire assignment a. For a vector $\vec{r} = \{r_1, \dots, r_m\}$ let us express $P_{\vec{r}}(a)$ as follows:

$$P_{\vec{r}}(a) = \sum_{i=1}^{m} r_i P_i(a) = s_0 + \sum_{i=1}^{N} s_i a_i + \sum_{1 \leq i,j \leq N} t_{ij} a_i a_j \tag{2}$$

for some $s_0, s_1, \dots, s_m \in \mathbb{F}_2$ and $t_{ij} \in \mathbb{F}_2$ for $1 \leq i, j \leq m$. Observe that we only require to check that $P_{\vec{r}(a)} = 0$. Therefore, instead of reading the entire assignment a and computing $P_{\vec{r}(a)}$, we let the prover perform most of the computation, and read only the final results. Specifically, we will ask the prover for each of the sums $\sum_i s_i a_i$ and $\sum_{ij} t_{ij} a_i a_j$. Towards this, let us define the following notation.

Definition 2.2. *Given an assignment $a = [a_1, \dots, a_N]$, define*

$$\begin{aligned} L(s) &= \sum_{i=1}^{N} a_i s_i && \textit{for a vector } s \in \{0,1\}^N, \\ Q(t) &= \sum_{i=1}^{N} a_i a_j t_{ij} && \textit{for a } N \times N \textit{ matrix } t \textit{ over } \{0,1\} \end{aligned} \tag{3}$$

Note that for every assignment a, L and Q are linear functions over $\mathbb{F}_2$.

The table of values L is just the Hadamard codeword for the assignment a. Recall the definition of the Hadamard code from last lecture.

Definition 2.3. *For a vector $x = \{x_1, \dots, x_r\}$ over a field $\mathbb{F}$, the Hadamard encoding of x, denoted* $\mathsf{Had}(x)$ *is given by:* $\mathsf{Had}(x)(s) = \sum_{i=1}^{r} s_i x_i$ *for every $s \in \mathbb{F}^r$.*

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

The table of values Q defined above is the *Quadratic Function* encoding of the assignment a. We define this encoding formally below.

Definition 2.4. *For a vector $x = \{x_1, \ldots, x_r\}$ over a field $\mathbb{F}$, the quadratic function encoding of x, denoted $\mathsf{QF}(x)$, is given by $\mathsf{QF}(x)(t) = \sum_{1 \leq i,j \leq r} t_{ij} x_i x_j$ for all $t \in \mathbb{F}^{r \times r}$.*

In other words, quadratic function code for a vector x is the Hadamard code for the vector $x \otimes x$. So every Quadratic function codeword is a Hadamard codeword (or in other words a linear function) but not vice-versa.

Using this new notation, we re-write equation 2 as

$$P_{\vec{r}}(a) \quad = \quad s_0 + L(s) + Q(t)$$

We can directly obtain the values $L(s)$ and $Q(t)$ from the prover, and check if $P_{\vec{r}}(a) = 0$. Observe that the values of s and t depend on the random choice $\vec{r}$. Let us assume that the proof contains tables L and Q, that list the values of $L(s)$ and $Q(t)$ for all values of s and t. Hence, we can get the value for $L(s)$ and $Q(t)$ directly from the proof to check if $P_{\vec{r}}(a) = 0$. By this we have reduced the number of queries to just two $(L(s), Q(t))$. However, there is a catch — there is no guarantee that the tables L and Q are correct.

By the correctness of L and Q, we mean the following

- $\mathcal{C}_1$**:** L is a linear function, say L is the Hadamard encoding of some $c \in \mathbb{F}_2^N$.
- $\mathcal{C}_2$**:** Q is a linear function on $\mathbb{F}_2^{N^2}$, say it is the Hadamard encoding of some $C = (C_{ij})_{1 \leq i,j \leq N}$.
- $\mathcal{C}_3$**:** Q and L are referring the same assignment c, i.e., the coefficient C_{ij} in Q is $c_i c_j$. Note that this condition also implies that Q is a Quadratic Function codeword and not just a Hadamard codeword.
- $\mathcal{C}_4$**:** The assignment c that both Q and L are referring to is indeed the assignment a.

From the previous lecture, we know how to test conditions $\mathcal{C}_1$ and $\mathcal{C}_2$ using only a few queries. Assuming that the tables have passed the linearity tests, the two tables are close to linear functions with constant probability. Hence we can use the Self-Correction Lemma of the previous lecture, to obtain the correct value of the linear function at s instead of reading $L(s)$ directly. Let us denote by $\mathit{SelfCorrect}(L, s)$ output of the Self-Correction routine, i.e the value of the linear function at s. If L and Q, pass the linearity test, then we have at our disposal the two linear functions (Hadamard codewords) $\mathit{SelfCorrect}(L, s)$ and $\mathit{SelfCorrect}(Q, t)$.

Testing Condition $\mathcal{C}_3$

Given any two vectors $s, \acute{s} \in \mathbb{F}^N$, if $L = \mathsf{Had}(c)$ and $Q = \mathsf{QF}(c)$, then observe that

$$\begin{aligned} L(s)L(\acute{s}) &= (\sum_i a_i s_i)(\sum_j a_j \acute{s}_j) \\ &= \sum_i \sum_j (s_i \acute{s}_j) a_i a_j \\ &= Q(s \otimes \acute{s}) \end{aligned} \tag{4}$$

Therefore, in order to test that L and Q are referring to the same assignment, we can check if $L(s)L(\acute{s}) = Q(s \otimes \acute{s})$ for randomly chosen $s, \acute{s}$.

Assuming Q and L satisfy the test for condition $\mathcal{C}_3$, with constant probability it must be true that Q and L are referring to the same assignment . Therefore, to test $\mathcal{C}_4$, it is enough to test if L is the linear function corresponding to a. i.e the coefficients of s_i in $L(s)$ is a_i. Observe that for e_i- the i^{th} unit vector, we get $L(e_i) = a_i$. So inorder to test if L is referring to assignment a, we just check if $L(e_i) = a_i$ for a randomly chosen i (using $SelfCorrect(L, e_i)$ to computer $L(e_i)$ reliably).

Thus we have few query tests for all the conditions $\mathcal{C}_i, i = 1 \ldots 4$, and also a few query test for $P_{\vec{r}}(a) = 0$. We will put this all together in the next section to get an Assignment tester.

3 Assignment Tester

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

Initialization: Arithmetize the circuit to obtain a system of quadratic constraints $\mathcal{P} = \{P_1, \ldots, P_m\}$ over variables $A = \{z_1, \ldots, z_N\}$. Let variables $z_1 \ldots z_{|X|}$ correspond to variables X and variables $z_{|X|+1} \ldots z_N$ correspond to gates Y.

The Proof:

- An assignment $a = (a_1, a_2, \ldots, a_N) \in \{0,1\}^N$ for the variables A (supposedly a satisfying assignment for $\mathcal{P}$).
- A table $L : \mathbb{F}_2^N \to \mathbb{F}_2$, supposedly equal to $\mathsf{Had}(a)$, i.e., satisfying $L(s) = \sum_{i=1}^N a_i s_i$
- A table $Q : \mathbb{F}_2^{N^2} \to \mathbb{F}_2$, supposedly equal to $\mathsf{QF}(a)$, i.e., satisfying $Q(t) = \sum_{i=1}^N a_i a_j t_{ij}$.

The Test:
Step 1

- Run BLR linearity test on L
- Run BLR linearity test on Q

Step 2 Pick random $s, \acute{s} \in \mathbb{F}_2^N$ and check if the following holds

$$\mathit{SelfCorrect}(L, s)\mathit{SelfCorrect}(L, \acute{s}) = \mathit{SelfCorrect}(Q, s \otimes \acute{s})$$

Step 3 Pick a random vector $\vec{r} \in \mathbb{F}_2^m$. Compute the coefficients s_i and t_{ij} such that

$$P_{\vec{r}}(a) = \sum_{i=1}^{m} r_i P_i(a) = s_0 + \sum_{i=1}^{N} s_i a_i + \sum_{1 \le i,j \le N} t_{ij} a_i a_j$$

Check if

$$s_0 + \mathit{SelfCorrect}(L, s) + \mathit{SelfCorrect}(Q, T) = 0$$

Step 4 Pick a random $i \in \{1, \ldots, |X|\}$. Let e_i denote the i^{th} unit vector of dimension N. Check if

$$\mathit{SelfCorrect}(L, e_i) = a_i$$

4 Proof of Soundness

In order to prove the soundness of the assignment tester, we prove the soundness of each of the steps in it by a sequence of lemmas.

Lemma 4.1. *If L or Q is δ_1-far from a linear function then Step 1 will reject with probability greater than or equal to δ_1.*

Proof: This lemma is nothing but the soundness result for BLR test, which was shown in the previous class. □

Lemma 4.2. *Given a non-zero matrix M, for random choice of vectors s and $\acute{s}$, $s^T M s = 0$ with probability at most $\frac{3}{4}$*

Proof: Let $(M)_{ij}$ be a non-zero entry in M. Let e_i and e_j denote the i^{th} and j^{th} unit vectors. Observe that

$$\begin{aligned} s^T M \acute{s} + (s^T + e_i) M \acute{s} + s^T M(\acute{s} + e_j) + (s^T + e_i) M(\acute{s} + e_j) &= e_i M e_j \\ &= (M)_{ij} \end{aligned}$$

Since, $(M)_{ij}$ is non-zero, it follows that at least one of the numbers $s^T M \acute{s}, (s^T + e_i) M \acute{s}, s^T M(\acute{s} + e_j), (s^T + e_i) M(\acute{s} + e_j)$ is non-zero. This implies that with probability at least $\frac{1}{4}$ over random choice of s and $\acute{s}$, $s^T M \acute{s}$ is not zero, which implies the result. □

Lemma 4.3. ***Completeness:*** *If $L = \mathsf{Had}(c)$, and $Q = \mathsf{QF}(c)$ for some vector $c \in \mathbb{F}_2^N$, then Step 2 always accepts.*
Soundness: *If L is δ_1-close to $\mathsf{Had}(c)$ and Q is δ_1-close to $\mathsf{Had}(C)$ such that $C_{ij} \neq c_i c_j$ for some i, j, then Step 2 rejects with probability at least $\frac{1}{4} - 6\delta_1$*

Proof: In Step 2 we are checking for randomly chosen $s, \acute{s}$, that $L(s)L(\acute{s}) = Q(s \otimes \acute{s})$. This identity holds whenever L and Q are Hadamard and quadratic function encodings of the same vector c. This proves the completeness part of the lemma.

For the soundness proof, define two matrices M_1 and M_2 as follows

$$\begin{aligned} (M_1)_{ij} &= c_i c_j & \qquad M_1 &= cc^T \\ (M_2)_{ij} &= C_{ij} & \qquad M_2 &= C \end{aligned}$$

From the property of self correction discussed in the previous lecture, Observe that if L is δ_1-close to $\mathsf{Had}(c)$, then with probability greater than $1 - 2\delta_1$,

$$\mathit{SelfCorrect}(L, s) = s^T c$$

That is with the value of $\mathit{SelfCorrect}(L, s)$ can be different from the linear function's value with probability at most $2\delta_1$. Likewise, except with probability at most $2\delta_1$, we have $\mathit{SelfCorrect}(L, \acute{s}) = \acute{s}^T c$. Similarly with probability at least $1 - 2\delta_1$, we know that

$$\mathit{SelfCorrect}(Q, s \otimes \acute{s}) = s^T C \acute{s}$$

Therefore with probability at least $1 - 6\delta_1$ the equality being tested in Step 2 is

$$\begin{aligned} s^T M_1 \acute{s} &= s^T M_2 \acute{s} \\ s^T (M_1 - M_2) \acute{s} &= 0 \\ s^T M \acute{s} &= 0 \end{aligned}$$

where $M = M_1 - M_2$ is a non-zero matrix. From Lemma 4.2 we conclude that $s^T M \acute{s}$ is nonzero with probability $\frac{1}{4}$. Therefore, with probability at least $(1 - 6\delta_1) - \frac{3}{4}$, the test made is $s^T M \acute{s} = 0$ and the test fails. So Step 2 rejects with probability at least $\frac{1}{4} - 6\delta_2$. $\square$

Lemma 4.4. *If L is δ_1-close to $\mathsf{Had}(c)$ and Q is δ_1-close to $\mathsf{QF}(c)$, and for some j, $P_j(c) \neq 0$ then Step 3 rejects with probability at least $\frac{1}{2} - 4\delta$.*

Proof: By Self-Correction we know that with probability at most $2\delta_1$, the value $\mathit{SelfCorrect}(L, s) \neq \sum_{i=1}^N c_i s_i$. Similarly with probability at most $2\delta_1$, $\mathit{SelfCorrect}(Q, T) \neq \sum_{i=1}^N c_i c_j t_{ij}$. So with probability at least $1 - 4\delta_1$ the test

$$s_0 + \mathit{SelfCorrect}(L, s) + \mathit{SelfCorrect}(Q, T) = 0$$

is equivalent to

$$P_{\vec{r}}(a) = s_0 + \sum_{i=1}^N s_i c_i + \sum_{1 \leq i,j \leq N} t_{ij} c_i c_j = 0$$

From Lemma 2.1 we know that for random $\vec{r}$, $P_{\vec{r}}(c) = 0$ with probability $1/2$ (since there exists j such that $P_j(c) \neq 0$). Therefore, with probability at least $1 - 6\delta_1 - \frac{1}{2}$,

$$s_0 + \mathit{SelfCorrect}(L, s) + \mathit{SelfCorrect}(Q, T) \neq 0$$

and Step 3 rejects. $\square$

Theorem 4.5. *Perform each of the steps 1,2,3,4 with probability $\frac{1}{4}$ each. Let a_X denote the assignment a restricted to variables X of the original boolean circuit Φ. Then*

- *If $L = \mathsf{Had}(a)$ and $Q = \mathsf{QF}(a)$ and $P_j(a) = 0$ for all $1 \leq j \leq m$ then the test accepts with probability 1.*
- *Let $\delta \leq 1/28$. If the assignment a_X is δ-far from every satisfying assignment to boolean circuit Φ, then the test rejects with probability at least $\frac{\delta}{8}$ irrespective of the contents of tables L and Q.*

Proof: The completeness part of the proof is trivial, since each of the steps in the assignment tester, has completeness 1.

Suppose L or Q is δ-far from the nearest Hadamard codeword, then from Lemma 4.1 with $\delta_1 = \delta$, Step 1 rejects with probability at least δ. Since with probability $\frac{1}{4}$ Step 1 is performed, the test rejects with probability at least $\frac{\delta}{4}$. Without loss of generality we assume that L and Q are δ-close to their nearest Hadamard codewords.

Further if L and Q do not 'refer' to the same assignment, by applying Lemma 4.3 with $\delta_1 = \delta$, Step 2 rejects with probability at least $\frac{1}{4} - 6\delta$. Observe that for $\delta < \frac{1}{28}$, $\frac{1}{4} - 6\delta \geq \delta$. As Step 2 is chosen with probability $\frac{1}{4}$, the test rejects with probability at least $\frac{\delta}{4}$. So without loss of generality, we can assume that L and Q also refer to the same assignment c.

Suppose c is not a satisfying assignment for $\mathcal{P}$, then by applying Lemma 4.4 with $\delta_1 = \delta$, we know that Step 3 rejects with probability at least $\frac{1}{2} - 4\delta$. Since $\delta < \frac{1}{28}$, Step 3 rejects with probability at least $\frac{1}{2} - 4\delta > \delta$. As Step 3 is chosen with probability at least $\frac{1}{4}$, the test rejects with probability at least $\frac{\delta}{4}$. So without loss of generality we can assume that c is a satisfying assignment for $\mathcal{P}$.

We know that a_X is δ-far from every satisfying assignment to Φ, so in particular it is δ-far from c_X (since c being a satisfying assignment for $\mathcal{P}$ implies that c_X satisfies the circuit Φ). By the property of Self Correction, and that L is δ-close to $\mathsf{Had}(c)$, we know that $\mathit{SelfCorrect}(L, e_i) = c_i$ with probability at least $1 - 2\delta$. Therefore with probability at least $1 - 2\delta$, Step 4 is testing if $a_i = c_i$. Since a is at least δ-far from c, $a_i \neq c_i$ with probability δ over the random choice of i in $\{1, \ldots, |X|\}$. Therefore Step 4 rejects with probability at least $\delta(1 - 2\delta) \geq \frac{\delta}{2}$. As Step 4 is chosen with probability at least $\frac{1}{4}$, the test rejects with probability at least $\frac{\delta}{8}$.

□

4.1 Remarks

- Observe that by end of Step 3 in the assignment tester, the verfier is convinced that there is some satisfying assignment to $\mathcal{P}$ and L and Q are referring to it. Therefore, steps $1, 2, 3$ already form a PCP system albeit with exponential size proofs. Step 4 just tests if the given assignment a is close to the assignment, that L and Q are referring to. This is the extension to get the Assignment Tester property.

- We have presented the assignment tester in a Prover-Verifier description. This description can be readily converted to suit the original definition of assignment tester in terms of constraints. The auxilary variables of the assignment tester, are

  - The variables a_i corresponding to the gates in the original boolean circuit Φ.
  - The entries in tables L and Q.

  Thus there are $m + 2^N + 2^{N^2}$ auxilary variables in total, all over the alphabet $\mathbb{F}_2$. All the constraints have at most six variables each, and are linear or quadratic equations over $\mathbb{F}_2$.

5 PCP Proof - Another look

With the construction of a Assignment Tester, we have completed the proof of the PCP theorem. Considering the importance of the theorem, and the variety of tools used in its proof, it is worth taking another look at the proof.

PCP theorem as it is stated normally, implies that for every language in NP there is a polynomial-size proof, that can be checked by random verifier using very few queries. However, as we showed in the first class, the PCP theorem can also be stated as a NP-hardness result. Towards this, we observed that PCP theorem is equivalent to the fact that $GAP-CG_{1,s}$ is NP-hard for some constant $s < 1$.

In order to show that $GAP - CG_{1,s}$ is NP-hard, we need a polynomial time reduction from a known NP-complete problem to it. But we also know that Constraint Graph satisfaction is NP-hard. Therefore, a possible proof of the PCP theorem would be to reduce a Constraint graph satisfaction problem to $GAP - CG_{1,s}$. Observe that Constraint Graph satisfaction problem is a special case of $GAP-CG_{1,s}$ for $s = 1-\frac{1}{|E|}$ where $|E|$ is the number of constraints in the constraint graph. Let us define the "gap" to be the unsatisfiability value of a constraint graph (i.e., 1 minus the fraction of constraints satisfied by the best assignment). So the PCP theorem guarantees a reduction from instances with gap $\frac{1}{|E|}$ to instances with gap $1 - s = \Omega(1)$. Therefore PCP theorem can be viewed as a Gap Producing or Amplifying Reduction.

The proof that we presented, creates the gap slowly and persistently, a little increase each time, keeping the other parameters under control. In the original proof of the PCP theorem, a large gap was created in one-single step and the other parameters had to be remedied later.

The Gap producing reduction consisted of several iterations. In each iteration, there were four steps that modified the parameters of the constraint graph appropriately. At the end of each iteration, the gap of the graph doubled, with a accompanying constant factor increase in its size. Therefore, by the end of $O(\log n)$ iterations, the gap increases to a constant, while the size is still polynomial in the original size of the graph.

The following table , sums up the central ideas and tools used in each of the four steps.

Step	Main Ideas	Effects	Proof Techniques
Degree Reduce	Split every vertex in to many vertices,and introduce an Expander cloud with equality constraints among the split vertices.	Size $\uparrow$ a $O(d)$ factor, Gap decreases by a constant factor, Alphabet remains same	Basic expansion property of expanders
Expanderize	Superimpose a constant degree expander with trivial constraints, on to the constraint graph G	Size $\uparrow$ a factor of 2 to 3, Gap decreases by a constant factor, Alphabet remains same	Existence of constant degree expanders and Property that Expander + Graph gives an expander.
Gap-Amplification	Each vertex's value is its opinion,on the values of vertices at a distance $<\ t$,Add edges corresponding to consistency on random walks	Size $\uparrow$ by a large constant factor ,Gap increases by $O(t)$, Alphabet size becomes $\|\Sigma\|^{O(d^t)}$	Properties of random walks on the graph
Alphabet-Reduce	Encode the assignment with error correcting codes, Build a circuit that checks if assignment satisfies and is a valid codeword, Use an assignment tester for the circuit	Size $\uparrow$ a constant factor, Gap decreases by a constant factor,Alphabet size reduced to 2^6	Hadamard codes, Linearity Testing, Fourier Analysis

Table 1: Proof of PCP