The **PCP** theorem - overview of the proof.

April 14, 2006

**Constraint satisfaction problems** A *constraint satisfaction problem* (CSP) is a collection \( f \) of functions \( f_1, \ldots, f_m \), where each \( f_i \) depends on only \( q \) inputs. These \( f_i \)'s are called *clauses* or *constraints*. We call \( f \) an *instance* or *formula*.

The *decision problem* is whether there exists an assignment \( \vec{x} = (x_1, \ldots, x_n) \in \Sigma^n \) such that \( f_i(\vec{x}) = 1 \) for all \( i \).

The *maximization problem* is to find \( \vec{x} \) that maximizes the number of \( f_i \) such that \( f_i(\vec{x}) = 1 \). We let \( \mu(f) \) denote the maximum fraction of clauses that can be satisfied by any assignment. The *approximation problem* within a factor \( c \geq 1 \) is, given \( f \), find a number \( \tilde{\mu} \) such that \( \frac{\mu}{c} \leq \tilde{\mu} \leq c \mu \).

The \( \epsilon \)-*gap problem* is to distinguish between \( f \)'s such that \( f(\vec{x}) = 1 \) for some \( \vec{x} \) and \( f \)'s such that for any \( \vec{x} \) less than \( 1 - \epsilon \) fraction of the constraints can be satisfied. That is, distinguish between \( f \)'s with \( \mu(f) = 1 \) and \( f \)'s with \( \mu(f) < 1 - \epsilon \). Thus, the decision problem is the 0-gap problem. Note that approximating the maximization problem within a factor smaller than \( 1/(1 - \epsilon) > 1 + \epsilon \) implies solving the \( \epsilon \)-gap problem.

**Example of CSPs** 3SAT: \( \Sigma = \{0, 1\} \), \( f_i \)'s are OR's, \( q = 3 \).

3COL: \( \Sigma = \{1, 2, 3\} \), \( f_i \)'s are \( \neq \), \( q = 2 \).

**General parameters:**

- Number of variables = \( n \)
- Number of clauses = \( m \) (we assume \( m \geq n \) and we consider \( m \) to be the *size* of the formula). Thus, we denote also \( |f| = m \).
- Alphabet size = \( |\Sigma| \), which we’ll denote by \( \sigma(f) \). We’ll always use finite size alphabet.
- Size of clause / number of queries = \( q(f) \)
- Degree: \( d(f) \) = the maximum number of constraints that involve one particular variable. (In 3COL this is the degree of the graph.)
- Gap \( \epsilon \) (as mentioned above, we’ll be mostly interested in the gap problem of distinguishing between fully satisfiable inputs and inputs that can be satisfiable with at most \( 1 - \epsilon \) fraction). The decision problem is equivalent to the gap problem with \( \epsilon = 1/m \).
- Satisfying fraction: \( \mu(f) \) = the maximum number of of \( f \)'s constraints that can be satisfied divided by \( m \).
We define \((q, \sigma, \epsilon) - \text{CSP}\) to be the \(\epsilon\)-gap problem of determining for a given instance \(f\) of that form with \(|\Sigma| = \sigma\) and number of queries \(q\). Note that length of description of such a instance is \(m(q \log n + q\sigma)\) which in our setting will always be less than \(m^2\).

**The PCP theorem.** The PCP theorem is the following:

**Theorem 1.** There exist constants \(q, \sigma, \epsilon > 0\) such that \((q, \sigma, \epsilon) - \text{CSP}\) is \(\text{NP-hard}\).

In an exercise you are asked to prove that \(\text{MAX3SAT}\) is hard to approximate within a constant factor.

In fact, what we’ll prove is that this holds for \(q = 2\) and some constants \(\sigma\) and \(\epsilon\). It’s already known that \((2, \sigma, 1/m) - \text{CSP}\) is \(\text{NP}\) hard (as \(\text{3-Coloring}\) is a special case of this). Thus, the result will follow from the following lemma:

**Lemma 2 (PCP main lemma).** There exist constants \(\sigma\) and \(c\) and a polynomial-time transformation \(T\) whose domain and range are \(\text{CSP}\) problems with \(|\Sigma| = \sigma\) and \(q = 2\) such that:

- **Linear blowup** For every input \(f\), \(|T(f)| \leq C|f|\).
- **Completeness** If \(\mu(f) = 1\) then \(\mu(T(f)) = 1\).
- **Gap amplification** There’s a constant \(\epsilon_0\) such that for every \(\epsilon < \epsilon_0\), if \(\mu(f) \leq 1 - \epsilon\) then \(\mu(T(f)) \leq 1 - 2\epsilon\).

The main lemma implies the PCP theorem since by repeating the transformation \(O(\log m)\) times we get a polynomial-time reduction from \((2, \sigma, 1/m) - \text{CSP}\) to \((2, \sigma, \epsilon_0)\). (Note that because of the linear blowup the size of the resulting formula will be indeed \(|f|C^{O(\log m)} = \text{poly}(m)|\).

**Proving the main lemma** The main lemma is proved by combining the following three steps:

**Lemma 3 (Gap amplification: Dinur’s lemma).** There exists a polynomial-time function \(\text{gap-amp}\) such that for every 2-query \(f\), and value \(\ell\) we have

- **Linear blowup** \(\text{gap-amp}(\ell, f)\) is a 2-query CSP such that for some \(C = C(\ell, \sigma(f))\), \(|\text{gap-amp}(\ell, f)| \leq C|f|\) and \(\sigma(\text{gap-amp}(\ell, f)) \leq C\).
- **Completeness** If \(\mu(f) = 1\) then \(\mu(\text{gap-amp}(\ell, f)) = 1\).
- **Gap amplification** There’s a constant \(\epsilon_0\) such that for every \(\epsilon < \epsilon_0/\ell\), if \(\mu(f) \leq 1 - \epsilon\) then \(\mu(\text{gap-amp}(\ell, f)) \leq 1 - \ell\epsilon\).

**Lemma 4 (Alphabet reduction).** There exists a polynomial-time function \(\text{alph-red}\) and absolute constants \(\sigma_0\) and \(q_0\) such that for every 2-query \(\text{CSP}\ f\)

- **Linear blowup** \(\text{alph-red}(f)\) is a \(q_0\)-query CSP with alphabet size less than \(\sigma_0\), and size less than \(C|f|\) for some \(C = C(\sigma(f))\).
- **Completeness** If \(\mu(f) = 1\) then \(\mu(\text{alph-red}(f)) = 1\).
- **Limited loss** There’s an absolute constant \(D\) (not depending on \(f\) or \(\sigma\)) such that if \(\mu(f) \leq 1 - \epsilon\) then \(\mu(\text{alph-red}(f)) \leq 1 - \epsilon/D\).

**Lemma 5 (Query reduction).** There exists a polynomial-time function \(\text{q-red}\) such that for every \(q\)-query \(\text{CSP}\ f\) with alphabet size \(\sigma\)
Linear blowup $q$-red$(f)$ is a 2-query CSP with alphabet size less than $\sigma^4$, and size less than $C|f|$ for some $C = C(q)$.

Completeness If $\mu(f) = 1$ then $\mu(q$-red$(f)) = 1$.

Limited loss If $\mu(f) \leq 1 - \epsilon$ then $\mu(q$-red$(f)) \leq 1 - \epsilon/D$ where $D = D(q, \sigma)$.

The main lemma is obtained by simply combining these three lemmas, choosing $\ell$ large enough as a function of all other constants.

Alphabet reduction The alphabet reduction step follows from the Hadamard-based PCP.

That is, let $f$ be a 2-query CSP (the construction generalizes to CSP’s with a larger constant number of queries) on $n$ variables $x_1, \ldots, x_n$ on alphabet $\sigma$. We will transform $f$ into a $q_0$-CSP $f'$ on the alphabet $\{0, 1\}$ such that $|f'| \leq C(\sigma)|f|$ and if $\mu(f) \leq 1 - \epsilon$ then $\mu(f') \leq 1 - \epsilon/100$.

- Each constraint in $f$ is a function $C : \Sigma \times \Sigma \to \{0, 1\}$. Let’s identify $\Sigma$ with $\{0, 1\}^c$ for some $c$. We can run the reduction of last time to find a system $Q_c$ of quadratic equations on three sets of variables $x, y \in \{0, 1\}^c$ and $z \in \{0, 1\}^{c'}$ (where $z$ is the auxiliary variables) such that $Q$ is satisfiable if and only if $c(x, y) = 1$ (where $x, y$ can be looked as both strings in $\{0, 1\}^c$ and elements of $\Sigma$).
- The CSP $f'$ will have a total of $2^cn + 2^{(2c+c')2}m$ variables which we divide into $n + m$ sets:
  - For every original variable $x_i$ which took values in $\Sigma$ we will have $x_i'$ be a sequence of $2^c 0/1$ variables. The way to translate an assignment of $s$ to $x_i$ to an assignment to the $x_i'$ variables would be to use $Had(s)$ where $Had()$ is the Hadamard encoding.
  - For every constraint $c$ of the original $f$, we’ll have $w_c$ be a sequence of $2^{(2c+c')2}$ 0/1 variables. If $c$ depends on $x_i$ and $x_j$ which are assigned values $s_i$ and $s_j$ satisfying $c(s_i, s_j) = 1$ then we can assign $Had((s_i \circ s_j \circ z)^{c})$ to the sequence $w_c$ where $z$ is the assignment to the auxiliary variables that makes the equation $Q_c$ accept.
- Suppose that we’re given oracle access to an assignment to all these variables, which may or may not correspond to the encoding above. We now need to come up with a test such that if it is the encoding of such a satisfying assignment then we’ll accept with probability one, and if any assignment violates at least an $\epsilon$ fraction of the constraints then we’ll reject with probability related to $\epsilon$.
- First, let’s assume that the assignments are always valid Hadamard encodings of some code words TO BE CONTINUED....