The PCP Theorem	July 5, 2005

Lecture 2

Lecturer: Madhu Sudan *Scribe: Matteo Paganin, Maria Carla Palmeri, Guzman Tierno*

1 Constraint satisfaction problems

NP may be considered as the complexity class of theorems and proofs. To face the fact that NP-problems are difficult one is lead to introduce probabilistic checking of proofs (that is, to use probabilistic verifiers instead of requiring the complete assurance of proof correctness).

On the other hand NP may be considered as the complexity class of optimization problems. In this case one is lead to consider approximations of the optimization problems (that is, to look for solutions that are feasible but only "close" to the best solution):

class:	NP	=	NP
	↓		↓
viewed as:	Complexity of theorems and proofs	$\Longleftrightarrow$	Complexity of optimization
	↓		↓
"faced" by:	Probabilistic checking of proofs	$\Longleftrightarrow$	Approximantion of optimization

The power of the probabilistic approach is somehow conveyed by the fact that errors in the "proof" do not completely destroy our possibility to find some truth: a "proof" with some minor errors is still, in some sense, a vehicle of some partial truth.

Let us now introduce a new class of problems: **Constraint Satisfaction Problems**.
Constraint satisfaction problems (CSP) are a special category of optimization problems that arise naturally in PCP. An instance of the problem consists of a collection of constraints $C_1, \ldots, C_m$ on some variables that take values from some set. The goal is to find an assignment that maximizes the number of satisfied constraints. Let define this class more precisely.

Definition 1 *Given an integer k and a finite alphabet Σ, an* **instance** *of Max k-CSP-Σ is constituted by n-variables, $x_1, \ldots, x_n$, that take values in Σ and m-constraints $C_1, \ldots, C_m$ of the form*

$$C_j = ((i_1^j, \ldots, i_k^j),\ f^j : \Sigma^k \to \{0,1\}),$$

where $1 \le i_h^j \le n$.

The **goal** *is to find an assignment $A : \{x_1, \ldots, x_k\} \to \Sigma$ that maximizes the number of satisfied constraints; a constraint C_j is satisfied by A if $f^j\left(A(x_{i_1^j}), \ldots, A(x_{i_k^j})\right) = 1$.*

Theorem 1 *If there exist constants q, c and s such that*

$$NP = PCP_{c,s}[r, q],$$

where $r \in O(\log(n))$, then there exist $\alpha < 1$, k and Σ such that α-approximating Max k-CSP-Σ is NP-hard.

Proof Let H be an NP-complete problem. We want to show that there exists a polynomial reduction f from H to an instance of Max k-CSP-Σ such that, for some $t \in \{1, \ldots, m\}$ (where m is the number of constraints of the instance),

$$G \in H \Rightarrow \text{opt}(\varphi_G) \geq t \tag{1}$$

$$G \notin H \Rightarrow \text{opt}(\varphi_G) < t \cdot \alpha \tag{2}$$

(where, $\varphi_G = f(G)$).

Since $H \in NP = PCP_{c,s}[r, q]$ there exists a verifier V such that

- $G \in H \Rightarrow \exists \Pi : \quad \Pr_R[V^\Pi(G; R) = 1] \geq c$,
- $G \notin H \Rightarrow \forall \Pi : \quad \Pr_R[V^\Pi(G; R) = 1] \leq s$.

We consider the class of problems Max k-CSP-Σ where $k = 2^q$ and $\Sigma = \{0, 1\}$. For each G we construct an instance φ_G of Max k-CSP-Σ (with the above k and Σ) satisfying 1 and 2.

Since q is a constant and $r \in O(\log(|G|))$ we have that $2^{q+r} \in O(|G|)$. Set $n = 2^{q+r}$ and observe that V can "hit" at most n addresses (bits) of the proof tape (note that V could be adaptive). We choose as a set of variables for our Max k-CSP-Σ instance the set $\{x_1, \ldots, x_n\}$ constituted by the hit addresses of the proof tape. Now, let $m = 2^r$ be the number of constraints and define, for each $R \in \{0, 1\}^r$, a constraint f_R as follows

$$f_R\left(x_{i_1^R}, \ldots, x_{i_k^R}\right) = V^\Pi(G, R)$$

(where Π has $x_1, \ldots, x_n$ as significant addresses and $1 \leq i_h^R \leq n$ for $h = 1, \ldots, k$).

If $G \in H$ there exists a proof Π such that $\Pr_R[V^\Pi(G; R) = 1] \geq c$ and this says that at least $c \cdot m$ constraints are satisfied by Π so that $\text{opt}(\varphi_G) \geq c \cdot m$. Analogously we obtain $\text{opt}(\varphi_G) \leq s \cdot m$. Defining $t = c\dot{m}$ and choosing $\alpha > s/c$ the conditions 1 and 2 are satisfied.

If a polynomial algorithm A α-approximating Max k-CSP-Σ existed this polynomial reduction would allow us to decide H in polynomial time as follows. Given G, if $A(\varphi_G) < \alpha \cdot t$ then $G \notin H$ otherwise $G \in H$, indeed:

$$G \in H \Rightarrow A(\varphi_G) \geq \alpha \cdot \text{opt}(\varphi_G) \geq \alpha \cdot t,$$

$$G \notin H \Rightarrow A(\varphi_G) \leq \text{opt}(\varphi_G) < \alpha \cdot t.$$

This would mean that $P = NP$ so that A is NP-hard. ∎

A kind of converse also holds.

Theorem 2 *Let H be an NP-complete problem. If there exist k, Σ and a polynomial reduction f from H to an instance of Max k-CSP-Σ such that, for some $t \in \{1, \ldots, m\}$ (where m is the number of constraints of the instance) and some $\alpha < 1$, we have*

$$G \in H \Rightarrow opt(\varphi_G) \geq t$$

$$G \notin H \Rightarrow opt(\varphi_G) < t \cdot \alpha$$

then

$$H \in PCP_{c,s}[r, q],$$

(where $r \in O(\log(n))$ and q is a constant).

Proof Our aim is to construct a verifier V for H. Such a verifier will proceed as follows:

1. constructs φ_G from G (in poly-time) with variables $x_1, \ldots, x_n$ and constraints $C_1, \ldots, C_m$,
2. receives a proof Π (i.e. an n-bit string, that may also be interpreted as an assignment to $x_1, \ldots, x_n$),

3. uses the random string R (of length $r \in O(\log(|G|))$) to choose a constraint C_j and verifies, with $q = k$ queries, if C_j is satisfied; if so, accepts the input, else rejects.

Finally observe that V is complete and sound:

$$G \in H \Rightarrow \mathrm{opt}(\varphi_G) \geq t \Rightarrow \exists \Pi : \ \mathrm{Pr}_R[V^{\Pi}(G; R) = 1] \geq \frac{t}{m} = c,$$

$$G \notin H \Rightarrow \mathrm{opt}(\varphi_G) < t \cdot \alpha \Rightarrow \forall \Pi : \ \mathrm{Pr}_R[V^{\Pi}(G; R) = 1] < \frac{t \cdot \alpha}{m} = s.$$

So H is in $PCP_{c,s}[r, q]$. ■

Dinur's Theorem actually asserts that such reduction exists for all H in NP so that

$$NP \subset PCP_{c,s}[r, q].$$