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):

\[
\begin{array}{ccc}
\text{class:} & \text{NP} & = & \text{NP} \\
\downarrow & & \downarrow \\
\text{viewed as:} & \text{Complexity of} & \iff & \text{Complexity of} \\
& \text{theorems and proofs} & \iff & \text{optimization} \\
\downarrow & & \downarrow \\
\text{"faced" by:} & \text{Probabilistic} & \iff & \text{Approximation of} \\
& \text{checking of proofs} & \iff & \text{optimization}
\end{array}
\]

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 \( \Sigma \), an instance of \( \text{Max } k\text{-CSP}\Sigma \) is constituted by \( n \)-variables, \( x_1, \ldots, x_n \), that take values in \( \Sigma \) and \( m \)-constraints \( C_1, \ldots, C_m \) of the form

\[
C_j = ((i^j_1, \ldots, i^j_k), f^j : \Sigma^k \rightarrow \{0, 1\}),
\]

where \( 1 \leq i^j_h \leq n \).

The **goal** is to find an assignment \( A : \{x_1, \ldots, x_k\} \rightarrow \Sigma \) that maximizes the number of satisfied constraints; a constraint \( C_j \) is satisfied by \( A \) if \( f^j(A(x_{i^j_1}), \ldots, A(x_{i^j_k})) = 1 \).

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

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

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

then

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

(wher $\alpha = f(G)$).

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

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

A kind of converse also holds.
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:

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

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

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

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