The PCP Theorem July 4, 2005

Lecture 1

Lecturer: Madhu Sudan *Scribe: Vincenzo Bonifaci*

1 Outline of the course

1. Definitions and history of PCP.
2. Approximation of MAX-SAT.
3. Approximation of MAX-CLIQUE.

4-10. A new proof of the PCP Theorem due to Irit Dinur.

2 Definitions of PCP

Let us begin by stating informally the PCP Theorem: *there are a format for writing proofs and a probabilistic method for verifying these proofs that*

- *accepts valid proofs;*
- *rejects invalid proofs with probability at least* $1/2$*;*

while reading only a constant number of bits of the proof.

This seems to be a statement regarding only logic and probability. So why should we care? Also, we should clarify what we mean by theorems and proofs and why we should be interested in them.

Here is a possible motivation. Everybody doing scientific research is periodically reviewing papers. For example, a theoretical computer scientist might be faced with a lot of papers claiming to have a proof that P=NP, or that P≠NP. How could one check the correctness of these claims efficiently?

We will call such a claim an *assertion*. A *theorem* is a true assertion. Purported "proofs" will also be called *evidence*. For our purposes, assertions/theorems and evidence/proofs will be simply strings of bits.

In this context, logic is precisely the science of deriving theorems, telling us how to concatenate statements to obtain proofs of new statements. However, we will abstract on the exact details of the derivation. For us, the correctness of the derivation will be checked by a *verifier*, that is, a verification algorithm. The verifier V takes as inputs an assertion A and some evidence π. The verifier has to respect the following properties:

- A is a theorem $\Rightarrow \exists$ evidence π s.t. $V(A, \pi)$ accepts;
- A is not a theorem $\Rightarrow \forall$ evidence π, $V(A, \pi)$ rejects.

Notice that if $V(A, \pi)$ rejects, it does not mean that A is not a theorem; it simply means that π is not a proof of A. Finally, we want the verification process to be efficient and thus assume that the verifier executes in time at most polynomial in the length of its input.

Another difference with the usual approach in logic is that we will be interested in finding short proofs. For example the following question will be of interest to us: *Does A have proofs of length at most some parameter n?* Short proofs give insights: usually, mathematical proofs convey more information if they are short. This is why we would like to have short proofs of Fermat's last theorem, of the 4 colors theorem, etc.

In some sense, the entire theory of computation is focusing on the difference between proving and verifying assertions. Consider the following language:

$$\text{SHORTPROOFS} = \{(V, A, 1^n) | \exists \text{ proof } \pi, |\pi| \leq n, \text{ s.t. } V(A, \pi) \text{ accepts.}\}$$

It is not difficult to see that SHORTPROOFS is NP-complete. This means that while verifying short proofs is easy, finding them is apparently hard. If SHORTPROOFS was easily (i.e., poly-time) recognizable, then we could think of replacing every mathematician with a computer, given the assumption that the main job of the mathematician is finding short proofs of interesting theorems.

3 History of PCPs

In the middle 80s the definition of a verifier was modified in order to allow interaction and randomness; that is, proofs are no longer static and the verifier is probabilistic. This notion was introduced by Goldwasser, Micali and Rackoff and by Babai and Moran; the resulting class of languages was called *Interactive Proofs* (IP). What can one prove in this system? An interesting language in IP is GRAPH NON-ISOMORPHISM. In order to be convinced by a prover that two graphs are not isomorphic to each other, one could first show the prover one of the graphs, and then pick one of the two at random, randomly shuffle its vertices and ask the prover if that graph is isomorphic to the one that was first shown. This system is apparently more powerful than deterministic static verifiers, since GRAPH NON-ISOMORPHISM is not known to be in NP.

Ben-Or, Goldwasser, Kilian and Wigderson extended this notion to that of *Multi-prover Interactive Proofs* (MIP) in which the verifier can query multiple independent provers (something like questioning people claiming their innocence in separate cells). Fortnow, Rompel and Sipser introduced Oracle Interactive Proofs. An oracle is bound to answer the same question always with the same answer; that is, answers must be independent of the history of questions. In an oracle proof we can recognize more languages than in an interactive proof, because we are restricting the ways in which the prover can lie. Also, with an oracle we can simulate a multi-prover proof with any number of provers. However, Fortnow et al. also showed that 2 independent provers are enough to decide all languages recognized with an oracle.

Notice that an oracle can be represented as a huge string of length $2^{p(n)}$, because it can answer yes or no to questions of length $p(n)$. But since the verifier executes in polynomial time, it will read only a polynomial number of bits.

Babai, Fortnow, Levin and Szegedy studied what they called *holographic proofs*; they particularly focused on the complexity parameters given by the length of the proof and the time needed for verification.

Finally, Feige, Goldwasser, Lovasz, Safra and Szegedy and Arora and Safra focused on different parameters, namely randomness (number of coin-tosses made by the verifier) and query complexity (number of bits of the proof). Notice that the proof is read using a random access model.

Definition 1 *An (r, q)-restricted* PCP *verifier V is a polynomial time machine that is given oracle access to evidence π and to a random coin-toss sequence R, such that $|R| \leq r(|A|)$, and for every R, the number of queries to π is at most $q(|A|)$. The machine can only accept or reject, that is $V^\pi(A; R) \in \{0, 1\}$.*

Thus, function $r(\cdot)$ is the *randomness complexity* of the verifier, while $q(\cdot)$ is its *query complexity.*

We can now define a parametrized complexity class.

Definition 2 $\mathrm{PCP}_{c,s}[r, q]$ *is the class of languages L s.t. there is an (r, q)-restricted* PCP *verifier V s.t. $\forall A \in \{0, 1\}^n$,*

- $A \in L \Rightarrow \exists \pi$ *s.t.* $\Pr_R[V^\pi(A; R) = 1] \geq c(n)$*;*
- $A \notin L \Rightarrow \forall \pi \Pr_R[V^\pi(A; R) = 1] \leq s(n)$*.*

Function $c(\cdot)$ measures the *completeness* of V, while $s(\cdot)$ measures the *soundness* of V. Sometimes we omit the first subscript, assuming perfect completeness: $\mathrm{PCP}_s[r, q] = \mathrm{PCP}_{1,s}[r, q]$. When we omit also the second subscript, we assume a soundness of 1/2: $\mathrm{PCP}[r, q] = \mathrm{PCP}_{1/2}[r, q]$.

Our definition is very broad and allows us to capture many complexity classes. For example:

- $\mathrm{PCP}[\mathrm{poly}(n), 0] = \mathrm{RP}$;
- $\mathrm{PCP}_{2/3,1/2}[\mathrm{poly}(n), 0] = \mathrm{BPP}$;
- $\mathrm{PCP}[0, \mathrm{poly}(n)] = \mathrm{NP}$;
- $\mathrm{PCP}[0, 0] = \mathrm{P}$;
- $\mathrm{PCP}[O(\log n), \mathrm{poly}(n)] = \mathrm{NP}$ (exercise).

Now we can also shortly express some results about PCP and NP.

- FGLSS: $\mathrm{NP} \subseteq \mathrm{PCP}[O(\log n \log\log n), O(\log n \log\log n)]$;
- BFLS: $\mathrm{NP} = \mathrm{PCP}[O(\log n), O(\log^3 n)]$;
- AS: $\mathrm{NP} = \mathrm{PCP}[O(\log n), O(\log^{1/2} n)]$;
- ALMSS: $\mathrm{NP} = \mathrm{PCP}[O(\log n), O(1)]$.

Actually, the PCP Theorem is stating something stronger than the last equality; it says that there is a universal constant q such that $\mathrm{NP} = \mathrm{PCP}[O(\log n), q]$ (i.e., the length of the proof is independent of the language). Finally, we mention that Håstad proved that

$$\forall \epsilon > 0 \;\; \mathrm{NP} \subseteq \mathrm{PCP}_{1/2+\epsilon}[O(\log n), 3].$$