**CSE431: Complexity Theory** February 27, 2012

# Lecture 14 Proof of PCP Theorem

*Lecturer: Anup Rao* Scribe:

## References

## 1 Overview

Recall that we wish to prove a theorem of the type:

**Theorem 1.** *For every constant $\epsilon > 0$, there is a polynomial time computable function $f$ mapping* 3SAT *formulas to* 3SAT *formulas such that if $\phi$ is a satisfiable formula, $f(\phi)$ is also satisfiable, and if $\phi$ is not satisfiable, then any assignment can satisfy at most $(7/8 + \epsilon)$ fraction of all clauses in $f(\phi)$.*

We shall work with Constraint Gaphs.

A constraint graph with alphabet size $k$ is an undirected graph on $n$ vertices, such that every edge $e = \{u, v\}$ is labeled with a constraint function $c_e : [k] \times [k] \to \{0, 1\}$. Throughout, we shall think of $k$ as a constant. The graph is satisfiable, if every vertex can be assigned a value from $[k]$ in such a way that the constraint of *every* edge is satisfied.

One can actually prove the following theorem:

**Theorem 2.** *For every constant $\epsilon > 0$, there is a constant $k$ and a polynomial time computable function $f$ mapping constraint graphs to constraint graphs with alphabet size $k$ such that if $G$ is a satisfiable graph, $f(G)$ is also satisfiable, and if $G$ is not satisfiable, then any assignment can satisfy at most $\epsilon$ fraction of the edges of $G$.*

Theorem 2 allows one to prove some version of Theorem 1 as follows. Start with any 3SAT instance, and build the following bipartite constraint graph. Every vertex on the left corresponds to a clause of the formula, and every vertex on the right corresponds to a variable of the formula. The alphabet size will be 8. There is an edge between a clause vertex and a variable vertex if and only if the clause contains the variable.

Any assignment to a *clause* vertex corresponds to assigning values to all the variables of that clause (there are 8 possible ways to do this) and any assignment to a variable vertex corresponds to assigning a binary value to the variable: say that the value is 0 if and only if the assigned value is 1. The constraint between a clause and a variable is satisfied if and only if the clause is satisfied by the assignment *and* the corresponding assignment to the variable is consistent with the value that the variable is given in the assignment to the clause. So, for example, if the clause is $(x_1 \vee x_2 \vee x_3)$ and the assignment to the vertex corresponding to $x_1$ is 1 and the assignment to the clause is $(0, 1, 1)$, then the corresponding constraint is *not* satisfied.

Then observe that the constraint graph is satisfiable if and only if the formula is satisfiable. Indeed, if the formula is satisfiable, one can simply use the obvious assignment to satisfy the constraint graph. On the other hand, if the constraint graph is satisfiable, the assignment to the variable vertices must satisfy all clauses. Thus we can use Theorem 2 on the resulting constraint

graph. Now suppose we have an arbitrary constraint graph with alphabet size $k$. Every assignment to the variables can be encoded with a binary string, and each constraint $c_e$ can be encoded with a 3SAT formula $\phi_e$, whose size depends only on $k$, on the binary variables that corresponds to the assignment to the vertices of the edge $e$. Then consider the formula $\bigwedge_e \phi_e$. If each small formula has $m$ clauses, then we get that if the original graph was satisfiable, so is this formula, but if the original graph was not satisfiable, then any assignment can satisfy at most $\epsilon$ edges in the graph, and so can satisfy at most $1 - (1 - \epsilon)/m = 1 - \Omega(1)$ fraction of the clauses of the final formula! Thus, although we don't get a result as strong as $7/8 + \epsilon$, we do get some constant fraction of clauses that will not be satisfied.

Our main goal will be to prove Theorem 2 (or some variant of it). The idea for doing this is similar in spirit to the Zig-Zag construction. We shall give several transformations on constraint graphs. All of these transformations will preserve the property that the new graph is satisfiable if and only if the old graph is. The goal is it to reduce the fraction of satisfiable edges:

**Degree reduction** Takes an arbitrary constraint graph $G$ with edges $E$ and outputs a new graph $G'$ with edges $E'$ such that $|E'| = O(|E|)$.