U.C. Berkeley — Handout N2
CS294: PCP and Hardness of Approximation — January 23, 2006
Professor Luca Trevisan — Scribe: Luca Trevisan

Notes for Lecture 2

These notes are based on my survey paper [5]. L.T.

Statement of the PCP Theorem and Constraint Satisfaction

1 Some Computational Complexity Preliminaries

NP-Completeness

We assume that all combinatorial objects that we refer to (graphs, boolean formulas, families of sets) are represented as binary strings. For a binary string x, we denote its length as $|x|$. We represent a decision problem as a language, that is, as the set of all inputs for which the answer is YES. We define **P** as the class of languages that can be decided in polynomial time. We define **NP** as the class of languages L such that there is a polynomial time computable predicate V and a polynomial $q()$ such that $x \in L$ if and only if there is w, $|w| \leq q(|x|)$ such that $V(x, w)$ accepts. We think of w as a *proof*, or *witness* that x is in the language.

For two languages L_1 and L_2, we say that L_1 reduces to L_2, and we write $L_1 \leq_m L_2$ if there is polynomial time computable f such that $x \in L_1$ if and only if $f(x) \in L_2$. A language A is **NP**-hard if every language L in **NP** reduces to A. A language is **NP**-complete if it is **NP**-hard and it belongs to **NP**.

NPO Problems

A combinatorial *optimization problem* O is defined[1] by a cost function $\text{cost}_O()$ that given an *instance* x of the problem and a *solution* s outputs $\text{cost}_O(x, s)$ which is either the cost of the solution (a non-negative real number) or the special value $\bot$ if s is not a *feasible* solution for x. For every x, there is only a finite number of feasible solutions s such that $\text{cost}_O(x, s) \neq \bot$. If O is a *maximization* problem (respectively, *maximization*), then our goal is, given x to find a feasible solution s such that $\text{cost}(x, s)$ is smallest (respectively, largest). We denote by $\text{opt}_O(x)$ the cost of an optimal solution for x.

For example, in the independent set problem, an instance is an undirected graph $G = (V, E)$, a feasible solution is a subset $S \subseteq V$ such that no two vertices of S are connected by an edge. The cost of a feasible S is the number of vertices in S.

An optimization problem O is an **NP**-optimization problem, and it belongs to the class **NPO**, if $\text{cost}_O()$ is computable in polynomial time and if there is a polynomial q such that for every instance and every solution s that is feasible for x we have $|s| \leq q(|x|)$. The independent set problem is clearly an **NPO** problem.

[1] Other conventions are possible, which are typically equivalent to the one we describe here.

If $\mathbf{P} = \mathbf{NP}$, then for every **NPO** problem there is a polynomial time optimal algorithm. If $\mathbf{P} \neq \mathbf{NP}$, then none of the **NPO** problems considered in this paper has a polynomial time optimal algorithm.

Approximation

If O is an **NPO** minimization problem, $r > 1$, and A is an algorithm, we say that A is an *r-approximate* algorithm for O if, for every instance x, $A(x)$ is a feasible solution for x and $\text{cost}_O(x, A(x)) \leq r \cdot \text{opt}_O(x)$. If O is a maximization problem and $r \geq 1$, then we say that A is an r-approximate algorithm if $\text{cost}_O(x, A(x)) \geq \text{opt}_O(x)/r$. Another common convention, that we do not use here, is to say that an algorithm is r-approximate for a maximization **NPO** problem O, with $r \leq 1$ if, for every x, $\text{cost}_O(x, A(x)) \geq r \cdot \text{opt}_O(x)$. Finally, sometimes we let $r = r(n)$ be a function of the length of the instance x, or of some other parameter related to the size of x.

For example, a 2-approximate algorithm for the independent set problem would be an algorithm that given a graph G outputs a feasible independent set S such that $|S| \geq |S^*|/2$, where S^* is an optimal independent set. As we will se later, such an approximation algorithm can exist if and only if $\mathbf{P} = \mathbf{NP}$.

2 Probabilistically Checkable Proofs

As discussed in the last lecture, probabilistically checkable proofs (PCPs) provide a "robust" characterization of the class **NP**. When we reduce a generic **NP** problem L to 3SAT using Cook's theorem, we give a way to transform an instance x into a 3CNF formula φ_x so that if $x \in L$ then φ_x is satisfiable, and if $x \notin L$ then φ_x is not satisfiable. Following the proof of Cook's theorem, however, we see that it is always easy (even when $x \notin L$) to construct an assignment that satisfies all the clauses of φ_x except one.

Using the PCP Theorem one can prove a stronger version of Cook's theorem, that states that, in the above reduction, if $x \in L$ then φ_x is satisfiable, and if $x \notin L$ then there is no assignment that satisfies even a $1 - \epsilon$ fraction of clauses of φ_x, where $\epsilon > 0$ is a fixed constant that does not depend on x. This immediately implies that Max 3SAT does not have a PTAS (unless $\mathbf{P} = \mathbf{NP}$), and that several other problems do not have a PTAS either (unless $\mathbf{P} = \mathbf{NP}$), using the reductions of Papadimitrious and Yannakakis [4] and others.

We define PCPs by considering a probabilistic modification of the definition of **NP**. We consider probabilistic polynomial time verifiers V that are given an input x and "oracle access" to a witness string w. We model the fact that V is a probabilistic algorithm by assuming that V, besides the input x and the witness w, takes an additional input R, that is a sequence of random bits. Then V performs a deterministic computation based on x, w and R. For fixed w and x, when we say "$V^w(x)$ accepts" we mean "the event that V accepts when given oracle access to witness w, input x, and a uniformly distributed random input R." When we refer to the "probability that $V^w(x)$ accepts," we take the probability over the choices of R.

We say that a verifier is $(r(n), q(n))$-restricted if, for every input x of length n and for every w,

$V^w(x)$ makes at most $q(n)$ queries into w and uses at most $r(n)$ random bits.

We define the class $\mathbf{PCP}[r(n), q(n)]$ as follows. A language L is in $\mathbf{PCP}[r(n), q(n)]$ if there is an $(r(n), q(n))$-restricted verifier V such that if $x \in L$, then there is w such that $V^w(x)$ accepts with probability 1 and if $x \notin L$ then for every w the probability that $V^w(x)$ accepts is $\leq 1/2$.

We also consider the following more refined notation. We say that a language L is in $\mathbf{PCP}_{c(n),s(n)}[r(n), q(n)]$ if there is an $(r(n), q(n))$-restricted verifier V such that if $x \in L$, then there is w such that $V^w(x)$ accepts with probability at least $c(n)$, and if $x \notin L$ then for every w the probability that $V^w(x)$ accepts is at most $s(n)$. Of course, the definition makes sense only if $0 \leq s(n) < c(n) \leq 1$ for every n. The parameter $c(n)$ is called the *completeness* of the verifier and the parameter $s(n)$ is called the *soundness error*, or simply the *soundness* of the verifier.

Note that if $r(n) = O(\log n)$ then the proof-checking can be *derandomized*, that is, V can be simulated by a polynomial time deterministic verifier that simulates the computation of V on each of the $2^{r(n)} = n^{O(1)}$ possible random inputs and then computes the probability that $V^w(x)$ accepts, and then accepts if and only if this probability is one. It then follows that, for example, $\mathbf{PCP}[O(\log n), O(\log n)] \subseteq \mathbf{NP}$. The PCP Theorem shows a surprising converse.

Theorem 1 (PCP Theorem) $\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$.

The theorem was proved in [2, 1], motivated by a relation between PCP and approximation discovered in [3], and based on much previous work. In the notes for the past lectures we discussed the historical context

3 PCP and the Approximability of Constraint Satisfaction Problem

In the Max 3SAT problem we are given a 3CNF boolean formula, that is, a boolean formula in conjunctive normal form (AND-of-OR of literals, where a literal is either a variable or the negation of a variable) such that each term in the conjunction is the OR of at most three literals. The goal is to find an assignment that satisfies the largest possible number of terms.

In the Max qCSP problem, where q is a positive integer, we are given a system of boolean constraints defined over boolean variables such that every constraints involves at most q variables. The goal is to find an assignment that satisfies as many constraints as possible. Note that Max 3SAT is a special case of Max 3CSP. (View each clause as a constraint.)

Theorem 2 *The* $\mathbf{PCP}$ *Theorem implies that there is a constant* q *such that there is no* a*-approximate algorithm for Max qCSP with* $a < 2$*, unless* $\mathbf{P} = \mathbf{NP}$*.*

Proof: Let $L \in \mathbf{PCP}[O(\log n), q]$ be an $\mathbf{NP}$-complete problem, where q is a constant, and let V be the $(O(\log n), q)$-restricted verifier for L. We describe a reduction from L to Max qCSP.

Given an instance z of L, our plan is to construct a Max qCSP instance I_z with $m = |z|^{O(1)}$ constraints such that

$$\begin{array}{rcl} z \in L & \Rightarrow & I_z \text{ is satisfiable} \\ z \notin L & \Rightarrow & \text{opt}_{\text{Max qCSP}}(I_z) \leq \frac{m}{2} \end{array} \tag{1}$$

Once (1) is proved, the theorem follows.

We enumerate all random inputs R for V. The length of each string is $r(|z|) = O(\log |z|)$, so the number of such strings is polynomial in $|z|$. For each R, V chooses q positions $i_1^R, \ldots, i_q^R$ and a Boolean function $f_R : \{0,1\}^q \to \{0,1\}$ and accepts iff $f_R(w_{i_1^R}, \ldots, w_{i_q^R}) = 1$.

We want to simulate the possible computation of the verifier (for different random inputs R and different witnesses w) as a Boolean formula. We introduce Boolean variables $x_1, \ldots, x_\ell$, where ℓ is the length of the witness w.

For every R we add the constraint $f_R(x_{i_1^R}, \ldots, x_{i_q^R}) = 1$. This completes the description of I_z.

Now, if $z \in L$, then there is a witness w that is accepted by V with probability 1. Consider the assignment $x_i \leftarrow w_i$, where w_i is the i-th bit of w: then such an assignment satisfies all the constraints of I_z.

If I_z has an assignment $x_i \leftarrow a_i$ that satisfies more than $m/2$ constraints, then the witness w defined as $w_i := a_i$ is accepted with probability more than $1/2$ by V, which implies that $z \in L$. So if $z \notin L$ then I_z has no assignment that satisfies more than $m/2$ constraints. □

Theorem 3 *The* **PCP** *Theorem implies that is an* $\epsilon_1 > 0$ *such that there is no polynomial time* $(1 + \epsilon_1)$*-approximate algorithm for Max-3SAT, unless* $\mathbf{P} = \mathbf{NP}$.

Proof: Given an instance I of Max qCSP (where q is the constant in the PCP Theorem) over variables $x_1, \ldots, x_n$ and with m constraints, we show how to construct an instance φ_I of Max 3SAT with m' clauses such that

$$\begin{array}{rcl} I \text{ is satisfiable} & \Rightarrow & \varphi_I \text{ is satisfiable} \\ \text{opt}_{\text{Max qCSP}}(I) \leq \frac{m}{2} & \Rightarrow & \text{opt}_{\text{Max 3SAT}}(\varphi_I) \leq (1 - \epsilon_1) m' \end{array} \tag{2}$$

Once (2) is proved, the theorem follows.

For every constraint $f(x_{i_1}, \ldots, x_{i_q}) = 1$ in I, we first construct an equivalent qCNF of size $\leq 2^q$. Then we "convert" clauses of length q to clauses length 3, which can be done by introducing additional variables, as in the standard reduction from kSAT to 3SAT (for example $x_2 \vee x_{10} \vee x_{11} \vee x_{12}$ becomes $(x_2 \vee x_{10} \vee y_R) \wedge (\bar{y}_R \vee x_{11} \vee x_{12})$). Overall, this transformation creates a formula φ_I with at most $q2^q$ 3CNF clauses for each constraint in I, so the total number of clauses in φ_I is at most $q \cdot 2^q \cdot m$.

Let us now see the relation between the optimum of φ_z as an instance of Max 3SAT and the optimum of I as an instance of Max qCSP.

If I is satisfiable, then set the x variables in φ_I to the same values and set the auxiliary variables appropriately, then the assignment satisfies all clauses, and φ_I is satisfiable.

If every assignment of I contradicts at least half of the constraints, then consider an arbitrary assignment to the variables of φ; the restriction of the assigment to the x variables contradicts at least $m/2$ constraints of I, and so at least $m/2$ of the clauses of φ_I are also contradicted. The number $m' - m/2$ is at most $m'(1-\epsilon_1)$ if we choose $\epsilon_1 \leq \frac{1}{2q2^q}$. $\Box$

Interestingly, the converse also holds: any gap-creating reduction from an **NP**-complete problem to Max qCSP implies that the **PCP** Theorem must be true.

Theorem 4 *If there is a reduction as in (1) for some problem L in* **NP**, *then $L \in \mathbf{PCP}[O(\log n), q]$. In particular, if L is* **NP**-*complete then the* **PCP** *Theorem holds.*

Proof: We describe how to construct a verifier for L. V on input z expects w to be a satisfying assignment for I_z. V picks a constraint of at random, and checks that the assignment $x_i \leftarrow w_i$ satisfies it. The number of random bits used by the verifier is $\log m = O(\log |z|)$. The number of bits of the witness that are read by the verifier is q.

$$\begin{array}{ll} z \in L & \Rightarrow I_z \text{ is satisfiable} \\ & \Rightarrow \exists w \text{ such that } V^w(z) \text{ always accept.} \end{array}$$

$$\begin{array}{ll} z \notin L & \Rightarrow \forall w \text{ a fraction } \frac{1}{2} \text{ of constraints of } I_z \text{ are unsatisfied by } w \\ & \Rightarrow \forall w\ V^w(z) \text{ rejects with probability } \geq \frac{1}{2} \end{array}$$

$\Box$

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92*. 3

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*. 3

[3] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proc. of FOCS91*. 3

[4] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*. 2

[5] Luca Trevisan. Inapproximability of combinatorial optimization problems. Technical Report TR04-065, Electronic Colloquium on Computational Complexity, 2004. 1