Notes on Complexity Theory Last updated: November, 2011

Lecture 21

Jonathan Katz

1 Probabilistically Checkable Proofs

Work on *interactive* proof systems motivates further exploration of *non-interactive* proof systems (e.g., the class $\mathcal{NP}$). One specific question is: how many bits of the proof does the verifier need to read? Note that in the usual certificate-based definition of $\mathcal{NP}$, the deterministic "verifier" reads the *entire* certificate, and correctness and soundness hold with probability 1. If we allow the verifier to be probabilistic, and are willing to tolerate non-zero soundness error, is it possible to have the verifier read fewer bits of the proof? (Turning as usual to the analogy with mathematical proofs, this would be like probabilistically verifying the proof of a mathematical theorem by reading only a couple of words of the proof!) Amazingly, we will see that it is possible to have the verifier read only a *constant* number of bits while being convinced with high probability.

Abstracting the above ideas, we define the class PCP of *probabilistically checkable proofs*:

Definition 1 *Let r, q be arbitrary functions. We say $L \in \mathsf{PCP}(r(\cdot), q(\cdot))$ if there exists a probabilistic polynomial-time verifier $\mathbf{V}$ such that:*

- *$\mathbf{V}^\pi(x)$ uses $O(r(|x|))$ random coins and reads $O(q(|x|))$ bits of π.*[1]
- *If $x \in L$ then there exists a π such that $\Pr[\mathbf{V}^\pi(x) = 1] = 1$.*
- *If $x \notin L$ then for all π we have $\Pr[\mathbf{V}^\pi(x) = 1] < 1/2$.*

Some remarks are in order:

- One can view a probabilistically checkable proof as a form of interactive proof where the (cheating) prover is restricted to committing to its answers *in advance* (rather than choosing them adaptively based on queries in previous rounds). Since the power of the cheating prover is restricted but the abilities of an honest prover are unaffected, $\mathcal{IP} \subseteq \mathsf{PCP}(\mathsf{poly}, \mathsf{poly}) \stackrel{\text{def}}{=} \bigcup_c \mathsf{PCP}(n^c, n^c)$. In particular, $\mathsf{PSPACE} \subseteq \mathsf{PCP}(\mathsf{poly}, \mathsf{poly})$.
- Since $\mathbf{V}$ runs in polynomial time (in $|x|$), the length of i (cf. footnote 1) is polynomial and so it is only meaningful for the length of π to be at most exponential in $|x|$. In fact, if the verifier uses $r(n)$ random coins and makes $q(n)$ queries then we may as well assume that any proof π for a statement of length n satisfies $|\pi| \leq 2^{r(n)} \cdot q(n)$.
- The soundness error can, as usual, be reduced by repetition. The completeness condition could also be relaxed (as long as there is an inverse polynomial gap between the acceptance probabilities when $x \in L$ and when $x \notin L$). In either case, the parameters r, q may be affected.

[1] Formally, $\mathbf{V}$ has an oracle tape on which it can write an index i and obtain the i^{th} bit of π in the next step.

- The definition allows $\mathbf{V}$ to query π *adaptively* (i.e., it may read the i^{th} bit, and then based on this value determine which index j to read next). We will only consider non-adaptive verifiers. However, any adaptive verifier making a constant number of queries can be converted into a non-adaptive verifier which makes only a (larger) constant number of queries.

1.1 Toward Understanding the Power of PCP

An easy observation is that $\mathsf{PCP}(0, \mathsf{poly}) = \mathcal{NP}$. In fact, we have the following stronger result:

Lemma 1 $\mathsf{PCP}(\log, \mathsf{poly}) = \mathcal{NP}$.

Proof Containment of $\mathcal{NP}$ in $\mathsf{PCP}(\log, \mathsf{poly})$ is obvious. For the reverse containment, let $L \in \mathsf{PCP}(\log, \mathsf{poly})$ and let $\mathbf{V}$ be the verifier for L. For given $x \in L$, we will show how to construct a witness for x; the $\mathcal{NP}$-machine deciding L will follow naturally. Note that we cannot simply use a "good" proof π_x (which is guaranteed to exist since $x \in L$) because π_x may be exponentially long. However, we *can* use a "compressed" version of π_x. Specifically, imagine running $\mathbf{V}$ for all possible settings of its $O(\log n)$ random coins (here, $n = |x|$). This results in a set S of only *polynomially many* indices at which $\mathbf{V}$ potentially reads π_x (for each setting of its random coins, $\mathbf{V}$ reads polynomially many indices; there are only $2^{O(\log n)} = \mathsf{poly}(n)$ possible settings of $\mathbf{V}$'s random coins). These queries/answers $\{(i, \pi_i)\}_{i \in S}$ will be our $\mathcal{NP}$ witness w. Our $\mathcal{NP}$ algorithm for L is simple: on input a witness w of the above form, simulate the computation of $\mathbf{V}$ (in the natural way) for all possible settings of its random coins. (If $\mathbf{V}$ tries to read an index which is not present in w, then $\mathbf{V}$ immediately rejects.) Accept only if $\mathbf{V}$ accepts in all those executions. ■

In fact, we have the more general result that $\mathsf{PCP}(r(n), q(n)) \subseteq \text{NTIME}(2^{O(r(n))} \cdot O(q(n)))$.

At the other extreme, if we allow no queries to π we obtain $\mathsf{PCP}(\mathsf{poly}, 0) = \mathsf{co}\mathcal{RP}$ (at least if we require perfect completeness, as we do in our definition). This, along with the previous result, shows that we only get something interesting from probabilistically checkable proofs if we consider the power of randomness and proof queries in tandem.

We have the following deep and important result:

Theorem 2 (The PCP Theorem) $\mathcal{NP} = \mathsf{PCP}(\log, 1)$.

The number of queries can be taken to be a fixed constant which is the same for all languages $L \in \mathcal{NP}$ (and not, e.g., a constant that depends on the language but not the input length). To see that this follows from the theorem, note that the theorem implies that SAT has a probabilistically checkable proof where the verifier uses $c \log |\phi|$ random coins and reads t bits when verifying the proof for some 3CNF formula ϕ. Now, for any $L \in \mathcal{NP}$ we can construct a probabilistically checkable proof where the verifier first applies a Karp reduction to the input to obtain a 3CNF formula ϕ, and then runs the PCP for SAT on input ϕ. If the Karp reduction maps n-bit inputs to n^k-bit formulae (for some constant k), then the verifier for L will use $ck \log |x|$ random coins and reads t bits when verifying the proof that some $x \in L$.

The above characterization is tight under the assumption that $\mathcal{P} \neq \mathcal{NP}$, in the sense that $\mathcal{P} \neq \mathcal{NP}$ is known to imply $\mathcal{NP} \not\subseteq \mathsf{PCP}(o(\log), o(\log))$. Also, although not explicit in the theorem, the PCP theorem also shows how to efficiently convert any witness w for a given x (with respect to a given $\mathcal{NP}$ relation R) into a proof π_x for which the corresponding PCP verifier always accepts.

For completeness, we also state the following result (that we will not explore further):

Theorem 3 $\mathsf{PCP}(\mathsf{poly}, \mathsf{poly}) = \mathsf{NEXP} = \mathsf{PCP}(\mathsf{poly}, 1)$.

2 PCP and Inapproximability

Assuming $P \neq \mathcal{NP}$, we know that we cannot hope to exactly solve all $\mathcal{NP}$-complete (search) problems in polynomial time. However, we might hope to be able to find an *approximate* solution in polynomial time. The PCP theorem can be used to show limits on the best approximations we can hope to achieve for some specific problems.

2.1 Inapproximability of max-SAT

As an example, we show that there exists some constant α such that it is infeasible to approximate (in polynomial time) the maximum number of satisfiable clauses in a 3CNF formula to within a multiplicative factor of α. We begin with some definitions.

Definition 2 *For a formula ϕ and an assignment $\mathbf{b}$ to the variables in ϕ, let $\mathsf{SAT}_{\mathbf{b}}(\phi)$ denote the fraction of clauses satisfied by the given assignment. Let $\mathsf{max\text{-}SAT}(\phi) = \max_{\mathbf{b}}\{\mathsf{SAT}_{\mathbf{b}}(\phi)\}$.*

Note that $\mathsf{max\text{-}SAT}(\phi) = 1$ iff ϕ is satisfiable. On the other hand, observe that if ϕ has m clauses and is unsatisfiable then it could be the case that $\mathsf{max\text{-}SAT}(\phi) = 1 - 1/m$; in other words, there is no fixed constant c for which $\mathsf{max\text{-}SAT}(\phi) < c$ iff ϕ is unsatisfiable. As for a lower bound, it is not hard to show that for any 3CNF formula ϕ we have $\mathsf{max\text{-}SAT}(\phi) \geq 7/8$. (*Proof*: A random $\mathbf{b}$ satisfies each clause with probability 7/8, and so satisfies 7/8 of the clauses in expectation. Thus, there must exist a $\mathbf{b}$ that satisfies at least 7/8 of the clauses.)

Definition 3 *Let $\rho < 1$. A value k is a ρ-*approximation for *ϕ if*

$$\rho \cdot \mathsf{max\text{-}SAT}(\phi) \leq k \leq \mathsf{max\text{-}SAT}(\phi).$$

*Polynomial-time algorithm A is an $\rho(\cdot)$-*approximation algorithm for 3SAT *if $A(\phi)$ always outputs a $\rho(|\phi|)$-approximation for ϕ.*

More generally: for an instance of a **maximization** *problem where the best solution has value v, a ρ-approximation ($\rho < 1$) is a value k with $\rho \cdot v \leq k \leq v$. For an instance of a* **minimization** *problem where the best solution has cost c, a ρ-approximation ($\rho > 1$) is a value k with $c \leq k \leq \rho \cdot c$. A polynomial-time algorithm is a ρ-*approximation algorithm *for some problem if it always outputs a ρ-approximation to its input instance.*

A 1-approximation algorithm for 3SAT would imply that we could solve 3SAT in polynomial time. By what we have said above, it is trivial to find an 7/8-approximation in polynomial time by always outputting the answer "7/8." Can we do better? Toward showing that there is a limit to how well we can do (assuming $\mathcal{P} \neq \mathcal{NP}$), we introduce the notion of an *amplifying reduction*.

Definition 4 *Let $c < 1$. A c-*amplifying reduction *of* 3SAT *is a polynomial-time function f on 3CNF formulae such that:*

- *If ϕ is satisfiable, then $f(\phi)$ is satisfiable. I.e., if $\mathsf{max\text{-}SAT}(\phi) = 1$ then $\mathsf{max\text{-}SAT}(f(\phi)) = 1$.*
- *If ϕ is not satisfiable, then every assignment to the variables in $f(\phi)$ satisfies at most a c-fraction of the clauses in $f(\phi)$. I.e., if $\mathsf{max\text{-}SAT}(\phi) < 1$ then $\mathsf{max\text{-}SAT}(f(\phi)) < c$.*

(In particular, an amplifying reduction is a Karp reduction.) We will say that 3SAT has an amplifying reduction *if it has a c-amplifying reduction for some $c < 1$.*

An amplifying reduction for 3SAT implies a hardness-of-approximation result for max-SAT:

Lemma 4 *Assume $\mathcal{P} \neq \mathcal{NP}$ and that* 3SAT *has a c-amplifying reduction. Then there is no c-approximation algorithm for* 3SAT.

Proof Assume to the contrary that there is a c-approximation algorithm A for 3SAT. We can then deterministically solve 3SAT in polynomial time as follows: on input formula ϕ, run $A(f(\phi))$ to obtain output k. If $k \geq c$, output 1; otherwise, output 0. To see correctness of this algorithm, note that when ϕ is satisfiable then $\mathsf{max\text{-}SAT}(f(\phi)) = 1$ and so the output k of A must be at least c. On the other hand, when ϕ is not satisfiable then $\mathsf{max\text{-}SAT}(f(\phi)) < c$ and so the output k of A must satisfy $k < c$. The claim follows. ■

In general, say we have a reduction f that maps, e.g., boolean formula to instances of a maximization (resp., minimization) problem such that

- If ϕ is satisfiable, then $f(\phi)$ has value (resp., cost) $\alpha(n)$, where n denotes the size of $f(\phi)$;
- If ϕ is not satisfiable, then $f(\phi)$ has value (resp., cost) strictly less than $\beta(n) < \alpha(n)$ (resp., strictly more than $\beta(n) > \alpha(n)$).

Then, assuming $\mathcal{P} \neq \mathcal{NP}$, the maximization (resp., minimization) problem has no $(\beta(n)/\alpha(n))$-approximation algorithm.

To establish the connection between the PCP theorem and inapproximability, we show that the PCP theorem implies the existence of an amplifying reduction for 3SAT. In fact, the implication goes in both directions, thus showing that one way to prove the PCP theorem is to construct an amplifying reduction.

Lemma 5 $\mathcal{NP} \subseteq \mathsf{PCP}(\log, 1)$ *if and only if* 3SAT *has an amplifying reduction.*

Proof One direction is easy. If 3SAT has an amplifying reduction f, then we can construct the following PCP system for 3SAT: On input ϕ, the verifier computes $f(\phi)$. The proof will contain a satisfying assignment for $f(\phi)$ (i.e., position i of the proof contains the assignment to x_i). To check the proof, the verifier chooses a random clause in $f(\phi)$, queries for the assignments to the 3 variables of that clause, and then verifies that the clause is satisfied for those settings of the variables. It accepts if and only if that is the case.

If ϕ is satisfiable then $f(\phi)$ is satisfiable and so a valid proof (consisting of a satisfying assignment for $f(\phi)$) exists. On the other hand, if ϕ is not satisfiable then at most a c-fraction of the clauses in $f(\phi)$ are satisfiable (for *any* assignment to the variables), and so the verifier accepts with probability at most c regardless of the proof. Since c is a constant, repeating the above procedure a constant number of times (and accepting only if each procedure leads to acceptance) will give the desired soundness error $1/2$ using an overall constant number of queries. Also, the number of random bits needed to select a random clause is logarithmic in $|\phi|$ since $|f(\phi)|$ is polynomial in $|\phi|$.

The other direction is the more interesting one. Say $\mathsf{SAT} \in \mathsf{PCP}(\log, 1)$, and let $\mathbf{V}$ be a verifier for SAT using $c \log n$ random coins (on input ϕ with $|\phi| = n$) and making t queries. We now describe an amplifying reduction f. On input a 3CNF formula ϕ do:

- For each setting r of the random coins for $\mathbf{V}$, do the following:
  - Determine the t indices $q_1, \ldots q_t$ that $\mathbf{V}(\phi; r)$ would when using random coins r (recall that without loss of generality these indices are chosen non-adaptively).

– Run $\mathbf{V}(\phi; r)$ on all possible settings for these bits of the proof to determine when $\mathbf{V}$ accepts in this case. In this way, one may define a CNF formula $\hat{\phi}_r$ on the variables $x_{q_1}, \ldots, x_{q_t}$ such that $\hat{\phi}_r$ evaluates to true exactly when $\mathbf{V}(\phi; r)$ would accept. (We stress that variables of the type x_{q_i} are the same for the different settings of r.) The number of clauses in $\hat{\phi}_r$ is constant since t is constant. Using auxiliary variables (different for each r), we may convert $\hat{\phi}_r$ to an equivalent 3CNF formula ϕ_r. The number of clauses in ϕ_r is constant as well.

- Set the output $f(\phi)$ to be $\bigwedge_{r \in \{0,1\}^{c \log n}} \phi_r$.

Note that the above can be implemented in polynomial time and, in particular, both the number of clauses and the number of variables in $f(\phi)$ are polynomial.[2]

We claim that f, as given above, is an amplifying reduction. It is not hard to see that if ϕ is satisfiable then $f(\phi)$ is (this follows from perfect completeness of the PCP system). On the other hand, assume ϕ is not satisfiable. Then for *any* setting of the variables in $f(\phi)$, at least half of the $\{\phi_r\}$ are not satisfied (this follows from soundness of the PCP system). In each unsatisfied ϕ_r there is at least one unsatisfied clause. Let $t' = O(1)$ denote the maximum number of clauses in any of the $\{\phi_r\}$. It follows that for any setting of the variables, the fraction of unsatisfied clauses in $f(\phi)$ is at least $\beta = 1/2t'$, and so the fraction of satisfied clauses is at most $1 - \beta$. This means that f is a c-amplifying reduction for any $c > 1 - \beta$. ■

An alternate way of viewing the above result is in terms of a promise problem where "yes" instances correspond to satisfiable 3CNF formulae, and "no" instances correspond to 3CNF formulae ϕ for which $\mathsf{max\text{-}SAT}(\phi) < c$. The above result implies that this promise problem is $\mathcal{NP}$-hard.

2.2 Inapproximability of Other Problems

Different $\mathcal{NP}$-complete problems may behave differently when it comes to how well they can be approximated. We discuss some examples here.

2.2.1 Minimum Vertex Cover and Maximum Independent Set

For a given graph G, note that a minimum vertex cover is the complement of a maximum independent set; thus, with respect to exact solutions, the problems are identical. However, they behave differently with respect to approximation. (Actually, this should not be too surprising: For an n-vertex graph with maximum independent set of size $I = n - O(1)$, an independent set of size $\rho \cdot I$ is a ρ-approximation; however, it gives a vertex cover of size $n - \rho I$ which is only an $(n - \rho I)/(n - I) = O(n)$-approximation and so arbitrarily bad as n gets large.)

Lemma 6 *There is a Karp reduction from 3CNF formulae to graphs such that for every 3CNF formula ϕ with $\mathsf{max\text{-}SAT}(\phi) = v$, the graph $G = f(\phi)$ has a maximum independent set of size $\frac{v}{7} \cdot n$ (where n denotes the number of vertices in G)*

The above just uses the standard Karp reduction we have seen in class before. Using our previous inapproximability result for $\mathsf{max\text{-}SAT}$, this immediately implies the following:

[2] Note that at most $2^{c \log n} \cdot t$ indices are ever potentially queried by $\mathbf{V}$.

Corollary 7 *If $\mathcal{P} \neq \mathcal{NP}$ then there are constants $\rho < 1$ and $\rho' > 1$ such that there is no ρ-approximation algorithm for Maximum Independent Set, and no ρ'-approximation algorithm for Minimum Vertex Cover.*

Actually, by reducing directly to the PCP theorem (rather than to 3SAT) we can get a stronger inapproximability result for Maximum Independent Set:

Theorem 8 *There is no 1/2-approximation algorithm for Maximum Independent Set.*

Proof Let $\alpha(n) = \Theta(n)$ be some function we will fix later. Given an arbitrary $\mathcal{NP}$-complete language L, we show a transformation f that takes an input x and outputs a graph G_x such that the following hold:

- If $x \in L$, then $G_x = f(x)$ has a maximum independent set of size $\alpha(n)$ (where n denotes the number of vertices in G_x).
- If $x \notin L$ then $G_x = f(x)$ has a maximum independent set of size at most $\alpha(n)/2$.

By the PCP theorem, there exists a probabilistically checkable proof system for L with a polynomial-time verifier $\mathbf{V}$ that on input x make t queries to its proof and uses $\ell = O(\log |x|)$ coin tosses. Let $r_1, \ldots, r_m \in \{0,1\}^\ell$ denote the sequence of all possible coin tosses of $\mathbf{V}$ (note $m = \mathsf{poly}(|x|)$), and let $q_1^i, \ldots, q_t^i$ denote the queries made on random coin tosses r_i. (Recall we assume queries are made non-adaptively.) Let $a_1^i, \ldots, a_t^i$ be a sequence of possible answers. Define a graph G_x as follows:

Vertices For each set of random coins r_i and each possible set of answers $a_1^i, \ldots, a_t^i$, the tuple

$$\left(r_i, (q_1^i, a_1^i), \ldots, (q_t^i, a_t^i)\right)$$

is a vertex if and only if $\mathbf{V}$ would accept x when using random coins r_i and receiving those answers to its queries.

Since r_i and x uniquely determine the queries, there are at most $m \cdot 2^t$ vertices in G_x.

Edges Two vertices v and u have an edge between them if and only if they are not consistent. (Two vertices are *not* consistent if they contain different answers to the same query.) Note that if vertices u, v contain the same random tape r_i then they cannot be consistent and so will share an edge.

Finally, add isolated vertices (if necessary) to obtain a graph with exactly $m \cdot 2^t$ vertices.

Define $\alpha(n) \stackrel{\text{def}}{=} n/2^t$, so that $\alpha(m \cdot 2^t) = m$. We show that G_x satisfies our desiderata:

- When $x \in L$, there exists a proof π for which $\mathbf{V}$ accepts for every setting of its random tape. This implies the existence of an independent set in G_x of size at least m.
- When $x \notin L$, the existence of an independent set with $m/2$ (or more) vertices would imply the existence of a proof that would cause $\mathbf{V}$ to accept with probability at least $1/2$, in contradiction to the soundness of the PCP system. ■

We can further amplify the above results, and show that there is *no* constant-factor approximation algorithm for Maximum Independent Set.

Theorem 9 *Assume $\mathcal{P} \neq \mathcal{NP}$. Then for* **every** $\rho \in (0, 1]$*, Maximum Independent Set cannot be ρ-approximated in polynomial time.*

Proof Given a graph G and an integer k, define the graph G^k as follows: vertices of G^k correspond to subsets of k vertices of G; two vertices S_1, S_2 of G^k have no edge between them iff $S_1 \cup S_2$ is an independent set in G. Note that G^k can be generated from G in polynomial time. If G has n vertices, then G^k has $\binom{n}{k}$ vertices. If G has an independent set S then the vertices in G^k corresponding to all k-size subsets of S form an independent set in G^k. Conversely, if there is some independent set $S_1, \ldots, S_\ell$ of vertices in G^k, then the vertices in $\cup_i S_i$ form an independent set in G. Thus, if the maximum independent set in G has size $|S|$ then the maximum independent set in G^k has size $\binom{|S|}{k}$.

Applying the reduction from Lemma 6, followed by the reduction above (using some fixed, constant value of k), we get a reduction f mapping boolean formulae to graphs, such that if ϕ is satisfiable then $G^k = f(\phi)$ has a maximum independent set of size $\binom{|S|}{k}$, while if ϕ is not satisfiable then G^k has a maximum independent set of size $\binom{\rho \cdot |S|}{k}$ for some $\rho < 1$, where $|S| = n/7$ (and n is the number of nodes in the intermediate graph produced by the reduction from Lemma 6). Taking k sufficiently large, the ratio $\binom{\rho \cdot |S|}{k} / \binom{|S|}{k} \approx \rho^k$ can be made arbitrarily small. ■