CS 6810: Theory of Computing **Fall 2021**

Lecture 20: Nov 11, 2021

Lecturer: Eshan Chattopadhyay *Scribe: Carlos Martínez*

1 Probabilistically Checkable Proofs

We now move on to *probabilistically checkable proofs* (PCPs). One view of PCPs is that they are kinds of non-interactive proof systems. As usual, given a language $L \subseteq \{0,1\}^*$ and a string $x \in \{0,1\}^*$, we ask whether $x \in L$. A prover presents a certificate π, which an efficient verifier can check. Traditionally, doing so requires reading the entire certificate. The PCP Theorem shows that, equivalently, the prover can rewrite their certificate π so that the verifier reads it probabilistically, by querying a small number of its bits. If $x \in L$, the probabilistic verification procedure will always accept a correct certificate. If the $x \notin L$, the probabilistic verification procedure will reject every certificate with high probability. We state this formally below.

Definition 1.1 (PCP verifier)**.** *Let L be a language and $q, r : \mathbb{N} \to \mathbb{N}$. We say that $L \in PCP(r(n), q(n))$ if there is a polynomial-time probabilistic algorithm V (the verifier) satisfying:*

> *Efficiency: On input string $x \in \{0,1\}^n$ and given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ (which we call the proof), V uses at most $r(n)$ random bits and makes at most $q(n)$ queries to locations of π. Let $V^\pi(x)$ denote the random variable representing V's output on input x and with random access to π. Then, $V^\pi(x) = 1$ if the verifier accepts and $V^\pi(x) = 0$ if the verifier rejects.*
>
> *Completeness: If $x \in L$, then there exists $\pi \in \{0,1\}^*$ such that $\Pr[V^\pi(x) = 1] = 1$.*
>
> *Soundness: If $x \notin L$, then for all $\pi \in \{0,1\}^*$, $\Pr[V^\pi(x) = 1] < 1/2$.*

It turns out the constant $1/2$ in the soundness requirement can as well be any other positive constant smaller than 1. The restriction to certificates $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ is because this is how many different locations a verifier could query with non-zero probability for all $2^{r(n)}$ choices for its random string.

Observe that $\text{NP} \subseteq \text{PCP}(0, \text{poly}(n))$. Also, observe that $\text{PCP}(r(n), q(n)) \subseteq \text{NTIME}(2^{O(r(n))}q(n))$ since a nondeterministic machine could guess the right proof in $2^{O(r(n))}q(n)$ time and verify it deterministically by running the verifier for all $2^{r(n)}$ possible random choices. It follows that $\text{PCP}(\log n, 1) \subseteq \text{NTIME}(2^{O(\log n)}) = \text{NP}$.

The PCP Theorem redefines NP in that every NP language has an efficient PCP system. We state it below.

Theorem 1.2 (Arora and Sudan [2], Arora, Lund, Motwani, Sudan, and Szegedy[1])**.**

$$NP = PCP(\log n, O(1))$$

The constant $O(1)$ has been improved over the years from about 10^6 initially to just 3. Now, proving this theorem would take too many lectures. Therefore, our goal for the next few lectures is to instead prove the following easier result.

Theorem 1.3.

$$NP \subseteq PCP(poly(n), O(1))$$

2 Approximation Algorithms and MAX-3SAT

Given a 3CNF Boolean formula ϕ with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$, MAX-3SAT is the problem of finding an assignment that maximizes the fraction of satisfied clauses. Let

$$\text{val}(\phi) := \frac{1}{m} \max_{x \in \{0,1\}^n} |\{j : C_j(x) = 1\}|$$

be the maximum fraction of clauses satisfied by any assignment. Since 3SAT is NP-complete, MAX-3SAT is NP-hard. Therefore, unless P = NP, there is no algorithm that computes val(ϕ) exactly. This motivates the notion of an *approximation algorithm*, which we (typically) require to run in polynomial time.

Definition 2.1. *For every $0 < \rho \leq 1$, an algorithm A is a ρ-approximation algorithm for a maximization problem Π if, for every instance I of Π, $A(I)$ is a solution of value at least $\rho \cdot OPT(I)$, where $OPT(I)$ is the value of an optimal solution.*

Definition 2.2. *For every $\rho \geq 1$, an algorithm A is a ρ-approximation algorithm for a minimization problem Π if, for every instance I of Π, $A(I)$ is a solution of cost at most $\rho \cdot OPT(I)$, where $OPT(I)$ is the cost of an optimal solution.*

In particular, we say algorithm A is a ρ-approximation algorithm for MAX-3SAT if, for every 3CNF formula ϕ, $A(\phi)$ is an assignment with at least a fraction $\rho \cdot \text{val}(\phi)$ of satisfied clauses.

It turns out the most intuitive algorithm one could design for MAX-3SAT, namely the greedy algorithm, is a 1/2-approximation algorithm. The greedy algorithm selects one variable at a time and assigns to it the value that results in satisfying at least 1/2 of the remaining clauses in which it appears. Any clause that is satisfied after this assignment is removed and not considered in the subsequent assignment of values to the remaining variables. There is also a randomized algorithm with an expected performance guarantee of 7/8, provided each clause has three distinct variables. The algorithm assigns each variable to 1 with probability 1/2 and to 0 otherwise. Then, for any fixed clause, the probability of it not being satisfied is 1/8. Equivalently, the probability of it being satisfied is 7/8. The performance guarantee then follows by linearity of expectation. This algorithm can be derandomized via the *method of conditional expectations*.

Can there be an efficient 9/10 approximation algorithm for MAX-3SAT? As we will see next, the PCP Theorem implies there is a constant $\rho < 1$ such that there is no polynomial-time ρ-approximation algorithm for MAX-3SAT unless P = NP. In fact, a stronger PCP Theorem implies that for every $\epsilon > 0$, there is no polynomial-time $7/8 + \epsilon$-approximation algorithm for MAX-3SAT unless P = NP.

3 Hardness of Approximation

Another view of the PCP Theorem is that it is a result about hardness of approximation.

Theorem 3.1 (PCP Theorem: Hardness of approximation view.)**.** *There exists $\rho < 1$ such that for every $L \in NP$ there is a polynomial-time function f mapping strings x to 3CNF formulas such that*

$$x \in L \implies val(f(x)) = 1$$
$$x \notin L \implies val(f(x)) < \rho.$$

To formally show the equivalence between the two views, we introduce the notion of *constraint satisfaction problems* (CSPs).

Definition 3.2 (CSP)**.** *If $q \in \mathbb{N}$, then a qCSP instance ϕ is a collection of functions $\phi_1, \ldots, \phi_m$ (called constraints) from $\{0,1\}^n$ to $\{0,1\}$ such that each function ϕ_i depends on at most q of its input locations. That is, for every $i \in [m]$, there exists $j_1, \ldots, j_q$ and $f : \{0,1\}^q \to \{0,1\}$ such that $\phi_i(u) = f(u_{j_1}, \ldots, u_{j_q})$ for every $u \in \{0,1\}^n$.*

We say an assignment $u \in \{0,1\}^n$ satisfies constraint ϕ_i if $\phi_i(u) = 1$. The fraction of constraints satisfied by u is $\frac{\sum_{i=1}^m \phi_i(u)}{m}$, and we let val($\phi$) denote the maximum of this value over all $u \in \{0,1\}^n$. We say ϕ is satisfiable if val(ϕ) $= 1$. We call q the arity of ϕ.

For example, 3SAT is the special class of CSPs where $q = 3$ and the constraints are OR's of the involved literals. MAX-qCSP is the problem of finding an assignment that maximizes the fraction of satisfied clauses.

We now introduce the notion of a *promise problem*. We are given a pair of disjoint languages L_{YES} and L_{NO} in $\{0,1\}^*$. All the inputs in L_{YES} are to be accepted and all inputs in L_{NO} are to be rejected. The set $L_{\text{YES}} \cup L_{\text{NO}}$ is called the promise and there are no requirements on the output if the input does not belong to the promise. We are ready to define ρ-GAP qCSP, which is a type of promise problem.

Definition 3.3 (ρ-GAP qCSP)**.** *For every $q \in \mathbb{N}$ and $\rho \leq 1$, ρ-GAP qCSP is the problem of determining for a given qCSP instance ϕ whether val(ϕ) $= 1$ (in which case we say ϕ is a YES instance of ρ-GAP qCSP) or whether val(ϕ) $= 1$ (in which case we say ϕ is a NO instance of ρ-GAP qCSP).*

We say ρ-GAP qCSP is NP-hard if for every language L in NP there is a polynomial-time function f mapping strings x to qCSP instances satisfying:

Completeness: $x \in L \implies$ *val*$(f(x)) = 1$

Soundness: $x \notin L \implies$ *val*$(f(x)) < \rho$

Theorem 3.4. *For some constant q, 1/2-GAP qCSP is NP-hard.*

Proof. It suffices to reduce 3SAT, an NP-complete language, to 1/2-GAP qCSP for some constant q. By the "proof view" of the PCP Theorem, 3SAT has a PCP system in which the verifier V makes a constant number q of queries and uses $c \log n$ random coins for some constant c. Given input x and $r \in \{0,1\}^{c \log n}$, define $V_{x,r}$ to be the function that given proof π outputs 1 if the verifier V accepts it on input x and coins r. Note that $V_{x,r}$ depends on at most q locations. Therefore, for every $x \in \{0,1\}^n$, the collection $\phi = \{V_{x,r}\}_{r \in \{0,1\}^{c \log n}}$ is a polynomial-sized qCSP instance. Moreover, since V runs in polynomial-time, the transformation of x to ϕ can also be carried out in polynomial-time. By the completeness and soundness of the PCP system, ϕ will satisfy val(ϕ) $= 1$ if $x \in$ 3SAT and val(ϕ) $\leq 1/2$ if $x \notin$ 3SAT. □

In the next lecture we will show the other direction. Namely, that if ρ-GAP qCSP is NP-hard for some constants ρ, q, then we can produce a PCP system with q queries, ρ soundness, and logarithmic randomness for any language in NP.

References

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[2] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.