CSCI 1951-W Sublinear Algorithms for Big Data **Fall 2020**

Lecture 9: PCP Theorem

Lecturer: Jasper Lee *Scribe: Ryan Greenblatt*

1 Review of Complexity Theory

We will consider the set of strings of 0 or 1 of arbitrary length:

$$\mathscr{L} = \{0,1\}^*$$

This can be thought of as the class of objects in the context of property testing.

A *language* is a set of strings $\mathcal{L} \subseteq \mathscr{L}$. This can also be thought of as a property in the context of property testing.

A *decision problem* is where we are given $x \in \mathscr{L}$ and we must decide if $x \in \mathcal{L}$ (or $x \notin \mathcal{L}$)

Definition 9.1 (Class P) A language $\mathcal{L} \in \mathsf{P}$ if and only if there exists a deterministic algorithm that decides $\mathcal{L}$, which always halts in time poly($|x|$) for all $x \in \mathscr{L}$. $|x|$ is the size of the input x.

Examples:

- Testing Bipartiteness of a graph.
- Existence of Eulerian cycle in a graph.
- Determining if 2 vertices are connected in a graph.

Definition 9.2 (Class NP) A language $\mathcal{L} \in \mathsf{NP}$ if and only if there exists a deterministic algorithm $V(x,w)$ and polynomials $p(n)$ and $q(n)$ such that

1. $V(x,w)$ runs in $p(|x| + |w|)$ time.
2. *(Completeness)* $\forall x \in \mathcal{L}$, $\exists\, w$ of length $q(|x|)$ such that $V(x,w) = 1$.
3. *(Soundness)* $\forall x \notin \mathcal{L}$, $\forall w$ of length $q(|x|)$, $V(x,w) = 0$.

We call V the verifier and w the witness. w can be thought of as the item which shows that $x \in \mathcal{L}$. So if no such item exists, then $x \notin \mathcal{L}$.

Examples:

- 3-SAT
- 3 Colorability (Graph coloring).

Definition 9.3 (Polytime reduction) $\mathcal{L}_A \leq_p \mathcal{L}_B$ if and only if there exists a polytime computable function $f : \mathscr{L} \to \mathscr{L}$ such that $a \in \mathcal{L}_A$ if and only if $f(a) \in \mathcal{L}_B$.

We can say that $\mathcal{L}_A$ reduces to $\mathcal{L}_B$ or that $\mathcal{L}_B$ is "harder" than $\mathcal{L}_A$.

The intuition is that we can use a solver for $\mathcal{L}_B$ to decide membership in $\mathcal{L}_A$.

Definition 9.4 (Class NP-hard) $\mathcal{L} \in \mathsf{NP}$-hard if and only if for all $\mathcal{L}' \in \mathsf{NP}$, $\mathcal{L}' \leq_p \mathcal{L}$.

Intuition: $\mathcal{L}$ is harder than any problem in NP.

Definition 9.5 (Class NP-complete) NP-complete $= \mathsf{NP} \cap \mathsf{NP}$-hard.

Intuition: $\mathcal{L}$ is among the hardest problems in NP.

P vs NP Observations

$\mathsf{P} \subseteq \mathsf{NP}$ because we can just ignore the witness and solve using the verifier.

The question of whether or not $\mathsf{P} = \mathsf{NP}$ is equivalent to asking if finding a witness is as easy as verifying a witness.

NP Languages as "Theorems with Proofs"

Given $\mathcal{L} \in \mathsf{NP}$:

- Interpret any $x \in \mathscr{L}$ as a (perhaps false) theorem statement that $x \in \mathcal{L}$
- $\forall x \in \mathcal{L}$, there exists a **proof** w such that $V(x, w) = 1$.
- $\forall x \in \mathcal{L}$, no such proof w exists.

For example, in the context of 3-colorability:

Theorem statement: "The input graph G is 3 colorable"

Proof: "A proper 3-coloring of G"

2 Probabilistically checkable proofs (PCPs)

Definition 9.6 $\mathsf{PCP}_{c,s}[r, q]$ where

- c: completeness probability
- s: soundness probability
- r: number of random bits required by V
- q: query complexity of V with respect to π

The complexity class $\mathsf{PCP}_{c,s}[r, q]$ consists of $\mathcal{L}$ such that there exists a randomized polytime verifier $V(x, \pi)$ such that

1. If $x \in \mathcal{L}$, there exists a poly-sized π such that $\mathbb{P}(V(x, \pi) = 1) \geq c$.
2. If $x \notin \mathcal{L}$, for all (incorrect proofs) π, $\mathbb{P}(V(x, \pi) = 1) \leq s$.
3. V uses at most $r(|x|)$ bits of randomness.
4. V makes at most $q(|x|)$ non-adaptive queries into π.

Definition 9.7 PCP shorthand

- $\mathsf{PCP}[r, q] = \mathsf{PCP}_{1,\frac{1}{2}}[r, q]$
- $\mathsf{PCP} = \mathsf{PCP}[O(\log n), O(1)]$

Proposition 9.8 $\mathsf{PCP} \subseteq \mathsf{NP}$

Proof. (Sketch)

$O(\log n)$ random bits cause only poly(n) possibilities of the random string which can be enumerated by the NP verifier. This is because we can consider random bits as inputs to a deterministic verifier. □

Theorem 9.9 (PCP theorem) $\mathsf{NP} \subseteq \mathsf{PCP}$. *Every* $\mathcal{L} \in \mathsf{NP}$ *has* PCP *system.*

Interpreting the PCP Theorem: For every $x \in \mathcal{L}$ there must be a NP proof w because $\mathcal{L} \in \mathsf{NP}$. This w can be always be encoded as a PCP proof π where π has length poly($|w|$). Further, $|w|$ is poly($|x|$), so $|\pi|$ is also poly($|x|$).

The "simple" proof is 40 pages long – so we won't prove it.

Related efficiency questions:

1. $O(1)$ query. For example, is it 10^{100}?
2. Length of π might also be very long as a function of $|x|$.

Theorem 9.10 (Håstad's 3-bit PCP) *For any* $\epsilon, \delta > 0$, $\mathsf{NP} = \mathsf{PCP}_{1-\epsilon,\frac{1}{2}+\delta}[O(\log n), 3]$. *Further, the 3 queries are non-adaptive.*

There have been many PCP constructions, and is an ongoing line of work.

PCP Theorem $\iff$ Gap/Inapproximability

We will use CSP (Constraint Satisfaction Problem) to illustrate this connection. This is basically a generalization of SAT.

Definition 9.11 (Constraint) For:

- A set of variables $\mathcal{V} = \{v_1, \ldots, v_n\}$.
- Set of values $\sum$ (finite), called the alphabet.

A q-ary constraint is a tuple $(C \subseteq \Sigma^q, i_1, \ldots, i_q)$ where C denotes the set of acceptable values of variables $v_{i_1}, \ldots, v_{i_q}$.
An assignment $a : \mathcal{V} \to \Sigma$ satisfies the constraint if $(a(v_{i_1}), \ldots, a(v_{iq})) \in C$

Definition 9.12 (q-CSP) An instance is $x = (\mathcal{V}, \Sigma, \mathcal{C})$ where $\mathcal{C}$ is a set of constraints such that all constraints in $\mathcal{C}$ are q-ary.

The question is to decide if there exists an assignment a that satisfies all constraints $C \in \mathcal{C}$.

Proposition 9.13 *For* $q \geq 2$, q*-CSP* $\in$ NP*-complete*

For example, 3-colorability can be modeled as a 2-CSP.

Example 9.14 Given $G = (\mathbb{V}, E)$ for vertices $\mathbb{V}$ and edges E, construct the following 2-CSP instance:

$\mathcal{V} = \mathbb{V}$

$\Sigma = \{0, 1, 2\}$ (colors)

$\mathcal{C}$ is such that the E don't violate the rules of a valid coloring, using 1 constraint per edge.

To illustrate the "gap" analogy, we need to a further notion of UNSAT value.

Definition 9.15 (UNSAT value)

$$\mathsf{UNSAT}(\mathcal{C}) = \min_{\text{asgn. } a} \text{ fraction of } C \in \mathcal{C} \text{ violated by } a$$

In other words, the minimum violating fraction over all assignments a.

Clearly we have that $\mathcal{C}$ is satisfiable if and only if $\mathsf{UNSAT}(\mathcal{C}) = 0$.

Proposition 9.16 (UNSAT value) *For $q \geq 2$ and q-CSP $x = (\mathcal{V}, \Sigma, \mathcal{C})$, NP-hard to decide if $\mathsf{UNSAT}(\mathcal{C}) = 0$ as opposed to $\mathsf{UNSAT}(\mathcal{C}) \geq \frac{1}{|\mathcal{C}|}$.*

Theorem 9.17 (PCP theorem, again) *For some $q \geq 2$ and $|\Sigma| > 1$, it is NP hard to decide if $\mathsf{UNSAT}(\mathcal{C}) = 0$ as opposed to $\mathsf{UNSAT}(\mathcal{C}) \geq \frac{1}{2}$.*

Observation 9.18 Consider q-CSP as an optimization problem where we try to find an assignment a that minimizes UNSAT (equivalently, maximizing the number of satisfied constraints). Theorem 9.17 implies that it is NP-hard to approximate the maximum number of satisfied constraints to a factor of 2.

Lemma 9.19 *Theorem 9.9 is equivalent to Theorem 9.17.*

Proof. We start by showing that Theorem 9.9 implies Theorem 9.17.

We will assume $\mathcal{L} \in \mathsf{NP}$ also has PCP. We now must show that $\mathcal{L} \leq_p q$-CSP. WLOG, we may consider the PCP verifier as a deterministic function which takes in the random string as opposed to a random function. Consider this deterministic PCP verifier: $V(x, \pi, \text{rand})$ with constant query complexity q. We will now construct a new q-CSP instance:

- $\mathcal{V} = [1, \ldots, |\pi|]$
- For each of the poly($|x|$) many possible random strings, we can construct a q-ary constraint. This is because we can constrain the queried variables to be such that V accepts for all possible random strings.

$\mathsf{UNSAT}(l)$ has gap $\geq \frac{1}{2}$, since PCP has perfect completeness and has soundness probability $\frac{1}{2}$.

Now we show the other directions, that Theorem 9.17 implies Theorem 9.9.

We have that the GAP-q-CSP $\in$ NP-hard. We now must show that we have PCP for every $\mathcal{L} \in \mathsf{NP}$. Fix any arbitrary $\mathcal{L} \in \mathsf{NP}$ for any instance x and consider the corresponding q-CSP instance $(\mathcal{V}, \Sigma, \mathcal{C})$. This q-CSP instance exists because the GAP-q-CSP $\in$ NP-hard by Theorem 9.17.

We can construct a PCP verifier of $\mathcal{L}$:

- The PCP proof π will be an assignment to the q-CSP instance described above.
- We may then randomly pick a constraint in $\mathcal{C}$ and check the q variables.
- Because this is a GAP-q-CSP instance, we have that the rejection probability $\geq \mathsf{UNSAT}(\mathcal{C})$.

□