

CMPT 407/710 - Complexity Theory: Lecture 16

Valentine Kabanets

July 10, 2017

1 PCP theorem

Last time, we mentioned that the setting of two non-interacting provers in $\text{MIP}(2)$ is equivalent to the setting where the proof is written down in advance, and the verifier may read (query) arbitrary locations of the proof (i.e., the verifier has random access to the proof). Below we state the “scaled down” version of the result $\text{NEXP} = \text{MIP}$, the famous PCP theorem.

1.1 PCP verifiers

The PCP theorem gives a novel characterization of the class NP in terms of probabilistically checkable proofs (PCPs). The central role in this characterization belongs to a Verifier. Here, the probabilities are taken over the random choices of the verifier, and the notation V^π means that the verifier V has random access to the string π , i.e., V may request to see the i th bit of the string π , for any i .

First, we define some natural restrictions on such verifiers: we restrict the amount of randomness they can use, and the number of queries to the proof they can make.

A verifier V is a probabilistic polytime Turing machine with input tape, work tapes, random tape, and random access to a read-only proof string π . We say that a verifier V is $(r(n), q(n))$ -restricted if, on input of size n ,

1. V uses at most $O(r(n))$ random bits, and
2. V queries at most $O(q(n))$ bits of the proof string.

1.1.1 Adaptive vs. non-adaptive verifiers

A verifier V can be either *non-adaptive* or *adaptive*. A non-adaptive verifier, after flipping its $O(r(n))$ random coins, decides upon $O(q(n))$ locations in the proof string that V wants to see. In other words, a non-adaptive verifier decides upon what locations to see in advance.

On the other hand, an adaptive verifier, after flipping $O(r(n))$ random coins, decides upon a first location of the proof string that V wants to see. Then, after seeing the contents

of this location, V decides which location to see next. After seeing the contents of the second location, V decides which location to see third, and so on for $O(q(n))$ locations.

In general, when talking about PCP verifiers, we mean adaptive verifiers. However, most verifiers used in the literature are usually non-adaptive.

1.2 The class PCP

We say that $L \in \text{PCP}[r(n), q(n)]$ (where PCP stands for Probabilistically Checkable Proofs) if there is a probabilistic polytime $(r(n), q(n))$ -restricted verifier V such that

1. (**completeness**) $x \in L \Rightarrow \exists \pi_x \Pr[V^{\pi_x}(x) = 1] = 1$,
2. (**soundness**) $x \notin L \Rightarrow \forall \pi \Pr[V^\pi(x) = 1] \leq 1/2$.

1.2.1 Some remarks on the definition of PCP

First, note that a non-adaptive $(r(n), q(n))$ -restricted verifier can access (over all of its random string of length $O(r(n))$) at most $2^{O(r(n))}O(q(n))$ bits of the proof string. So, if $r(n) = \log n$, and $q(n) \in \text{poly}(n)$, then such a verifier can see at most a polynomial number of bits of the proof, and so we may as well assume that the proof string is of polynomial length (since if it's longer, the verifier won't be able to see the extra part anyway).

The case of an adaptive verifier is trickier. Once V has flipped its random coins, we can think of the *strategy* of V for requesting the bits of the proof string as a binary tree of height at most $O(q(n))$. The root u of the tree is labeled by the first location of the proof string that V wants to see. The left child v of u is labeled by the next location in the proof that V wants to see if the contents of the first location is 0; the right child w of u is labeled by the next location in the proof that V wants to see if the contents of the first location is 1. The children of v and w are defined similarly. Since V asks at most $O(q(n))$ queries, the height of this strategy tree is $O(q(n))$. Finally, the children of the last query in a given path of the strategy tree are two leaves: left (corresponding to the contents 0) and right (corresponding to the contents 1). The leaves are labeled by Accept or Reject, according to the decision of the verifier V .

For example, if $q(n) = 2$, then for some random string of V , the strategy tree of V may look as follows: root u is labeled 13, the left child v is labeled 5, and the right child w is labeled 7. Both children of v are labeled Accept; and the left child of w is labeled Accept, whereas the right child of w is labeled Reject. This tree corresponds to the strategy: if proof bit 13 is 0, then see proof bit 5; otherwise, see proof bit 7. If bit 13 is 1, and bit 7 is 1, then Reject; in all other cases, Accept.

1.3 The PCP Theorem

We now state the famous PCP theorem, which is one of the deepest results in theoretical computer science.

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

The PCP Theorem was proved by Arora, Safra, and Arora, Lund, Motwani, Sudan, Szegedy in two papers that in turn built upon rich body of work on interactive proof systems done earlier by many researchers. The authors of these two papers have been honored with the Gödel prize for their contribution to computer science.

The proof of the PCP Theorem is quite involved, and may well deserve an independent course devoted entirely to PCP. Recently, a simpler proof of the PCP theorem was discovered by Irit Dinur; her proof is more combinatorial in nature than the original, more algebraic, proof. While significantly simpler than the original proof, the new proof is still too involved to be presented here.

2 Hardness of approximation

The main application of the PCP Theorem is to proving that *many* optimization problems are not just NP-hard to solve exactly, but also remain NP-hard to *approximate*! This can be thought of as extending the classical NP-completeness theory from *exact* (decision) problems to more general, *approximation* problems. Such an extension provides a more refined picture of hardness to approximate for a number of well-known NP-hard optimization problems.

We first explain what we mean by an approximation algorithm.

2.1 Approximation algorithms

Suppose we want to approximate an optimum value of some maximization problem (e.g., given a graph, find a clique of maximum size). Rather than an optimal solution S_{opt} of optimum value v_{opt} , we may wish to find an approximate solution S of value v so that v is approximately v_{opt} . (For example, for CLIQUE, we may want to find a clique that is of size close to the maximum size.)

To quantify the quality of the approximate solution, we say that, for some $0 < \alpha \leq 1$, a solution S of value v is an α -approximation to a maximization problem with an optimal solution S_{opt} of optimum value v_{opt} if

$$\alpha \cdot v_{opt} \leq v \leq v_{opt}.$$

That is, our solution S achieves at least α factor of the optimum value.

For a *minimization* problem, we say that S (of value v) is a c -approximation to S_{opt} (of value v_{opt}), for some $c \geq 1$, if

$$v_{opt} \leq v \leq c \cdot v_{opt}.$$

That is, v is at most c factor bigger than the optimum value v_{opt} .

2.2 2-approximation polytime algorithm for Vertex Cover

As an example, we show that the following minimization problem has a factor 2 approximation algorithm running in polytime. Recall that a Vertex Cover Minimization problem (MIN-VC) is:

Given a graph $G = (V, E)$ find a smallest-size set $S \subseteq V$ that covers all the edges of G (i.e., every edge in G shares at least one of its endpoints with the set S).

We know that the decision version of Vertex Cover is NP-complete. Hence, the Minimization version is NP-hard. Assuming that $P \neq NP$, this means that there is no polytime algorithm that finds an optimum-size vertex cover in any given graph.

If we can't hope to find a smallest vertex cover, perhaps we can efficiently find a set cover which is "almost" optimal. In agreement with our earlier definition, we say that the Vertex Cover Minimization problem can be efficiently *c-approximated*, for some $c \geq 1$, if there is a polytime algorithm that, on any input graph G , returns a vertex cover S of G such that $|S_{opt}| \leq |S| \leq c|S_{opt}|$, where S_{opt} is a smallest-size vertex cover for G .

Theorem 2. *VC is polytime 2-approximable.*

Proof. Let $G = (V, E)$ be any input graph. Here's the algorithm for finding a vertex cover S of G .

Initially set $S = \emptyset$. Until there are no edges left, repeat the following: Take any edge $e = \{u, v\}$. Add vertices u and v to S . Remove e from G , as well as remove all edges that touch e (at u or v).

Clearly, the described algorithm runs in polytime, and produces a vertex cover S . Next we argue that $|S| \leq 2|S_{opt}|$. To this end, observe that the set S corresponds to (the endpoints of) a set of mutually disjoint edges of G . Any vertex cover of G (including an optimal S_{opt}) must cover each of these $|S|/2$ edges; so every vertex cover must contain at least one vertex for each of these edges. This means that $|S_{opt}| \geq |S|/2$, as promised. $\square$

Now that we have a 2-approximating algorithm for VC, we may ask for a 3/2-approximating algorithm, or more generally, for a $(1 + \epsilon)$ -approximating algorithm for any $\epsilon > 0$. If such an algorithm exists, then it's almost as good as being able to solve VC optimally. Does it exist?

It turns out that the answer is "No, unless $P = NP$ ". In other words, it is NP-hard to approximate VC to within some constant $1 < \alpha \leq 2$. Note that it is hopeless to try to show *unconditionally* that VC cannot be γ -approximated. This is because if $P = NP$, then certainly VC can be 1-approximated. Thus, proving that VC is NP-hard to approximate is the next best thing.

This inapproximability result for MIN-VC will follow from the PCP-based arguments!

2.3 MAX 3SAT

First, we show an immediate consequence of the PCP theorem for the hardness of approximation of the following maximization problem

MAX3SAT: Given a 3-CNF ϕ , find the maximum value $0 \leq \alpha \leq 1$ such that at least α fraction of clauses of ϕ can be satisfied simultaneously.

The following is an equivalent formulation of the PCP theorem stated earlier.

Theorem 3 (Arora-Safra, Arora-Lund-Motwani-Sudan-Szegedy). *For some constant $\gamma < 1$, there is a polytime reduction R from SAT to 3SAT such that*

1. $\phi \in SAT \Rightarrow R(\phi)$ is satisfiable, and
2. $\phi \notin SAT \Rightarrow$ no assignment satisfies more than the fraction γ of all clauses of the 3-CNF $R(\phi)$.

Theorem 3 above immediately shows that MAX3SAT is NP-hard to approximate to within any constant factor greater than γ .

Theorem 4. *For the constant $\gamma < 1$ as in Theorem 3, MAX3SAT is NP-hard to α -approximate for any $\alpha > \gamma$.*

Proof. Suppose that there is a polytime algorithm $\mathcal{A}$ that α -approximates MAX3SAT for some $\alpha > \gamma$. That is, this algorithm maps a satisfiable formula to some value $v \geq \alpha \cdot 1 > \gamma$, and an unsatisfiable formula with at most fraction μ of simultaneously satisfiable clauses to some value $v \leq \mu$.

We will use this algorithm to solve SAT as follows: Given a SAT instance ϕ , apply the reduction R from Theorem 3 to ϕ , getting a 3SAT instance ψ . Next run the assumed α -approximation algorithm $\mathcal{A}$ on this ψ , getting the output value v .

We get:

$$\begin{aligned} \phi \in SAT &\Rightarrow \psi \in 3SAT \Rightarrow v \geq \alpha > \gamma \\ \phi \notin SAT &\Rightarrow \psi \text{ has } \leq \gamma \text{ fraction of satisfiable clauses} \Rightarrow v \leq \gamma. \end{aligned}$$

Thus, we correctly decide if $\phi \in SAT$ by outputting YES if $v > \gamma$, and NO otherwise. $\square$

2.4 Hardness to approximate Vertex Cover

Recall the decision version of VC: Given $G = (V, E)$ and an integer $k > 0$, is there a vertex cover of G of size at most k ?

A standard reduction showing that the decision version of Vertex Cover is NP-complete proceeds as follows. It's a reduction from 3SAT to VC. Given a 3cnf ϕ on m clauses, build a graph G_ϕ on $3m$ vertices, by constructing for each clause $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ a triangle graph on nodes labeled by the literals ℓ_1, ℓ_2, ℓ_3 of the clause. Add to G_ϕ the following edges: for

any two nodes u and v in distinct triangles (clauses of ϕ), if u and v are labeled by some complementary literals ℓ and $\bar{\ell}$, then (u, v) is an edge of G_ϕ . Output $(G_\phi, 2m)$.

For the correctness of the described reduction, observe that if ϕ is satisfiable, then G_ϕ has an independent set¹ of size m (with each clause/triangle contributing exactly one node to the independent set). The remaining $3m - m = 2m$ nodes of the graph G_ϕ form a vertex cover (as can be easily seen).

Conversely, if G_ϕ has a vertex cover of size $2m$, then its complement set of size $3m - 2m = m$ is an independent set of G_ϕ , and it can be used to satisfy ϕ (by making true at least one literal in each of m clauses of ϕ , namely, the literal corresponding to the node from our independent set of size m).

The following observation will be crucial in our NP-hardness result for approximating MIN-VC.

Lemma 5. *The reduction from 3SAT to VC described above has the property:*

$$\phi \in 3SAT \Rightarrow G_\phi \text{ has minimal VC of size } 2m$$

$$\phi \text{ has } \leq \mu \text{ fraction of satisfiable clauses} \Rightarrow G_\phi \text{ has minimal VC of size } \geq 3m - \mu \cdot m.$$

Proof. Use the fact that the maximal number of simultaneously satisfiable clauses of ϕ is exactly equal to the maximal size of an independent set in the graph G_ϕ . (Exercise!) Then use the fact that the complement of an independent set is a vertex cover, and vice versa. $\square$

We use this NP-completeness reduction from 3SAT to VC (in particular, Lemma 5), together with the fact that MAX3SAT is NP-hard to γ -approximate (Theorem 3), to derive the following.

Theorem 6. *There is a constant $1 < c < 2$ such that MIN-VC is NP-hard to c -approximate.*

Proof. Suppose there is a polytime c -approximation algorithm $\mathcal{A}$ for MIN-VC. We will use it to decide SAT as follows.

Given a formula ϕ , apply the reduction R from Theorem 3 Lemma 5 to ϕ , getting a 3cnf formula ψ on m clauses (for some m polynomial in the size of ϕ). Then apply the reduction from Lemma 5 to this ψ , getting a graph G_ψ on $3m$ nodes.

By Theorem 3 and Lemma 5, we get:

- if $\phi \in SAT$, then G_ψ has a minimal VC of size $2m$, and
- if $\phi \notin SAT$, then G_ψ has minimal VC of size at least $3m - \gamma \cdot m$ (for a constant $\gamma > 0$).

Our assumed c -approximation algorithm $\mathcal{A}$ applied to G_ψ will output the value v such that:

- if $\phi \in SAT$, then (G_ψ has a minimal VC of size $2m$, and hence) $2m \leq v \leq c \cdot 2m$,

¹Recall that an independent set in a graph is a set of nodes such that no pair of nodes in the set are connected by an edge.

- if $\phi \notin SAT$, then (G_ψ) has minimal VC of size at least $3m - \gamma \cdot m$, and hence) $v \geq (3 - \gamma)m$.

Thus, if c is such that $c \cdot 2m < (3 - \gamma)m$, then we would be able to distinguish the case of $\phi \in SAT$ from the case $\phi \notin SAT$! We conclude that for $c < \frac{3-\gamma}{2}$, it's **NP**-hard to c -approximate MIN-VC. Finally, note that since $\gamma < 1$ in Theorem 3, we get that our c above can be made strictly greater than 1, as promised. $\square$

The above theorem shows that for some constant $c > 1$, it's **NP**-hard to approximate MIN-VC. For the best known value of γ in Theorem 3, which is about $7/8$, we get $c \approx 1.06$. With much more work, Dinur and Safra showed that MIN-VC is **NP**-hard to c -approximate for $c \approx 1.36$. This is the current world record!

It is still an open question whether MIN-VC is **NP**-hard to approximate to within the factor $2 - \epsilon$, for any $\epsilon > 0$. Under a certain PCP conjecture (known as the Unique Games Conjecture, UGC²) by Subhash Khot, such a $(2 - \epsilon)$ -factor hardness of approximation for MIN-VC (as well as tight inapproximability results for a number of other optimization problems) would follow.

²The work on the UGC is an active area of research. Currently, the complexity community seems to be approximately evenly split into those who believe the UGC and those who don't. Resolving the conjecture either way, however, would be an important breakthrough.