18.408 Topics in Theoretical Computer Science Fall 2022
Lecture 9

Dor Minzer

In the last few lectures, we have described the sum-check protocol and low-degree testing problem, as well as proved the correctness of the sum-check protocol (under the promise the assignment is a low-degree polynomial), and the low-degree testing result in the 1% regime. Today, we will combine these two results in order to prove a PCP theorem with poly-logarithmically many queries.

1 A PCP with $\mathsf{poly}(\log n)$ Queries

To formalize PCPs is in a combinatorial language, we use the language of constraint satisfaction graphs.

Definition 1.1. *A instance of constraint satisfaction graph (abbreviated CSG) consists of sets of nodes $X = \{x_1, \ldots, x_n\}$, an alphabet Σ_i for each node x_i, a collection hyperedges E and a constraint C_e for each edge $e \in E$. For each edge $e \in E$, writing $e = (x_{i_1}, \ldots, x_{i_s})$, the constraint C_e can be any subset of $\Sigma_{i_1} \times \ldots \times \Sigma_{i_s}$; these tuples are thought of as satisfying the constraint.*

The alphabet size of an instance is $\max_i |\Sigma_i|$, and the number of queries is the size of the largest hyperedge in the graph.

Given an instance of CSG, $\Psi = (X, E, \Sigma, \{C_e\}_{e \in E})$, the goal is to find an assignment to the nodes that satisfies as many of the constraints as possible. Namely, find a labeling A of X such that for as many hyperedges $e = (x_{i_1}, \ldots, x_{i_s})$ as possible we have that $(A(x_{i_1}), \ldots, A(x_{i_s})) \in C_e$. The value of an instance Ψ, denoted by $\mathsf{val}(\Psi)$, is the maximum fraction of constraints that can be satisfied by any assignment A. Thus, in the problem gap-CSG$[c, s]$ we are given an instance Ψ of CSG promised to either have $\mathsf{val}(\Psi) \geqslant c$ or $\mathsf{val}(\Psi) < s$, and the goal is to distinguish between these two cases.

We will often refer to hardness results for CSG as PCP constructions, and to measure the efficiency of the PCP construction we will focus on the size of the alphabet, number of queries, completeness and soundness parameters it achieves. In this language, we show:

Theorem 1.2 (PCP with poly-logarithmic number of queries)**.** *There are absolute constants $c, C > 0$ such that gap-CSG$[1, 1/\log(n)^c]$ is NP-hard on instances with alphabet size and number of queries at most $\log(n)^C$.*

The proof of Theorem 1.2 proceeds by a reduction. The starting point of the reduction is the problem gap-QS$_{q=\log(n)^{100}, r=n}[1, \frac{1}{\sqrt{q}}]$ (which we proved to be NP-hard), and we let (X, E) be an instance of quadratic equations over field of size $q = \log(n)^{100}$ (where n is the number of variables). We produce an instance Ψ of CSG in polynomial time as follows.

We choose $d = |\mathbb{H}| = \log(n)$ and $m = \frac{\log(n)}{\log\log(n)}$ in the linearized sum-check protocol and write $X = \mathbb{H}^m$. For each $e \in E$, we run the linearized sum-check protocol. Namely we have $A_0 \colon \mathbb{F}_q^m \to \mathbb{F}_q$ which is assumed to be the low-degree extension of an assignment of (X, E), and for each e we have an auxiliary table G_e which consists of all of the partial sum function needed by the sum-check protocol. Then the linearized sum-check protocol describes a collection tests in G_e and A_0 that satisfies:

1. Each constraint contains two entries from A_0, and at most $(m+1)d$ entries from G_e.
2. If A_0 satisfies e, then all of them are satisfied,
3. Else, if A_0 doesn't satisfy e and is a function of total degree at most d, then at most $\frac{2md+1}{q}$ of the constraints hold.

Thus, we will think of the entries of A_0 and all of the tables G_e as nodes in our CSG instance Ψ, and of the checks generated by the sum-check protocol as defining hyperedges and constraints on them. We are not done with the description of Ψ, though; we must enforce that the table A_0 is a function of total degree d for our analysis of the linearized sum-check protocol to be of use. Towards this end, we will use our low-degree testers.

In addition to the table A_0, we will also have a table A_2 which is supposed to consist of the restriction of A_0 to all planes in $\mathbb{F}_q^m$. Namely, for each $P \in S_2(\mathbb{F}_q^m)$, we include a new node, $A_2[P]$, in Ψ, and the alphabet of this node corresponds to functions of total degree at most d over P.

Thus, our PCP will proceed as follows: we sample a test T in the linearized sum-check protocol. Recall that to do so, we sample an equation $e \in E$ and the randomness as in the linearized sum-check protocol, and eventually check some quadratic equation in $G_e, A_0(\vec{i}), A_0(\vec{j})$; suppose it is given as $h(G_e, A_0(\vec{i}), A_0(\vec{j})) = 0$. Upon generating this equation, we sample a plane P containing $\vec{i}$ and $\vec{j}$, query $A_2[P]$ and check that $A_2[P](\vec{i}) = A_0(\vec{i})$. Thus, overall the constraint we generate in Ψ makes both of these checks: i.e. it reads G_e, $A_0(\vec{i})$, $A_0(\vec{j})$ and $A_2[P]$ and checks that $h(G_e, A_0(\vec{i}), A_0(\vec{j})) = 0$ and $A_2[P](\vec{i}) = A_0(\vec{i})$.

We have the following lemma, which establishes that Theorem 1.2 holds:

Lemma 1.3. *The above PCP has the following properties:*

1. ***Completeness:*** *If (X, E) is satisfiable, then* $\mathsf{val}(\Psi) = 1$. *Namely, there are tables $\{G_e\}_{e\in E}$, A_0 and A_2 that satisfy the above checks with probability* 1.
2. ***Soundness:*** *If (X, E) is at most ε satisfiable, then* $\mathsf{val}(\Psi) \leqslant O\left((2md/q + \varepsilon)^{1/3}\right)$. *Namely, any $\{G_e\}_{e\in E}$, A_0 and A_2 satisfy at most fraction $O\left((2md/q + \varepsilon)^{1/3}\right)$ of the constraints of Ψ.*

Proof. The completeness is clear, and we move to the soundness.

For the soundness, suppose that we have $\{G_e\}_{e\in E}$, A_0 and A_2 that pass all of the checks with probability at least δ; denote by E the event that the above checks work. Note that by properties of the sum-check protocol, when we sample a check as above, the distribution of each one of $\vec{i}$ and $\vec{j}$ is uniform over $\mathbb{F}_q^m$. Let $\eta > 0$ to be determined, and let $k = \frac{2}{\eta^2}$. Then by the low-degree testing theorem, we may find $f_1, \ldots, f_k \colon \mathbb{F}_q^m \to \mathbb{F}_q$ of total degree at most d such that

$$\Pr_{\vec{i}\in P \in S_2(\mathbb{F}_q^m)}\left[A_0(\vec{i}) = A_2[P](\vec{i}),\ \forall \ell A_2|_P \neq f_\ell|_P\right] \leqslant \eta.$$

We denote by E' the event that $A_2[P] = f_\ell|_P$ for some ℓ. Then we get that $\Pr\left[E \wedge \bar{E}'\right] \leqslant \eta$, and so $\Pr\left[E \wedge E'\right] \geqslant \delta - \eta$. It follows that there is ℓ, such that the probability that replacing A_0 by f_ℓ, we get that the above checks are all satisfied with probability at least $\frac{1}{k}(\delta - \eta) \geqslant \frac{(\delta-\eta)\eta^2}{2}$.

However, since f_ℓ is of total degree at most d the analysis of the linearized sum-check protocol says that it passes it with probability at most $\frac{2md+1}{q}$ on equations of (X, E) which it doesn't satisfy, and by assumption there are at most ε equation that it satisfies, so overall we must have that

$$\frac{(\delta-\eta)\eta^2}{2} \leqslant \frac{2md+1}{q} + \varepsilon.$$

Taking $\eta = \delta/2$, we conclude the result. □

2 Formalizations of PCPs

As discussed in the first lecture of this course, one can view a PCP from several different equivalent perspectives. While they are equivalent, sometimes, depending on the context and application, it is more natural to view PCP in one of the views rather than another. For the majority of the course, we will stick to the combinatorial view, as formalized by constraint satisfaction graphs as above. You are encouraged, however, to think above what we've seen in the course so far in the different views, and establish the equivalence between them.

The combinatorial, constraint satisfaction view. In this lecture, we have chosen the combinatorial view in which a PCP construction is thought as a hypergraph. The nodes of the graph represent variables and each one of its edges is associated with a constraint on its nodes. The goal is to assign labels to the nodes of the graph so as to satisfy as many of the constraints as possible. The parameters of most interest here are the completeness (the fraction of constraints that can be satisfied in the YES case), the soundness (the fraction of constraints that can be satisfied in the NO case), the alphabet size, the number of queries (the number of alphabet symbols that need to be read to check constraints), and the instance size. In this language, we achieved perfect completeness, i.e. 1, soundness $(\log n)^{-\Omega(1)}$, alphabet $\mathsf{poly}(\log n)$, queries $\mathsf{poly}(\log n)$ and polynomial instance size, i.e. $n^{O(1)}$.

The verifier view. Another view is the verifier view, wherein we think of some NP statement, and the verifier is given a proof π. The verifier selects randomly (in a correlated manner) a few locations in π, checks that they satisfy some constraint, and if so the verifier accepts (and otherwise the verifier rejects). One can define the same parameters as before, wherein the amount of random bits the verifier uses becomes an important parameter; this parameter is analogous to the logarithm of the number of edges in the graph in the combinatorial view.

The k-prover, verifier view. Finally, a third view of PCPs is when we have a polynomial time verifier which wants to be convinced of some NP statement, and towards this end the verifier can ask questions to k all powerful provers that cannot communicate with each other. In this view, one should think of the verifier as generating locations in a supposed proof π, but since the verifier does not have a proof, he can send each chosen location to one of the provers, and expect in return the value that was supposed to be in that location in π. One can again discuss various analogous parameters to before, and in particular the number of provers is the analog of the number of queries in the previous views.

3 What's Ahead?

Theorem 1.2 is already a very substantial result in complexity theory. In fact, with some additional work one can use it towards constructing a quasi-polynomial size PCP with constantly many queries and constant alphabet size which can serve as the basis for many hardness of approximation results. The catch is that these results would not be NP-hardness result, since the transformation from Theorem 1.2 to the constant queries, constant alphabet size PCP would require a quasi-polynomial time reduction (i.e. time $n^{\mathsf{poly}(\log n)}$), so to be meaningful one would need to assume that NP has no quasi-polynomial time algorithms.

There are several avenues one may continue in their path of study in PCP, and below we outline a few topics that will be discussed in the upcoming weeks.

1. **Recursion:** One can think of the sequence of reductions (sum-check, low-degree testing) we did as a method that took us from a PCP with n queries over alphabet of size q (the gap version of Quadratic equations from the second lecture), to a PCP with $\mathsf{poly}(\log n)$, while incurring only small loses in the soundness of the PCP and polynomial size blowup in the size of the instance. One may hope that there is a way to recursively apply such process to get a PCP with $\mathsf{poly}(\log\log n)$ queries, then $\mathsf{poly}(\log\log\log n)$ and so on. This turns out to indeed be possible, analogously to the composition step we saw in error correcting codes.

   Doing so though, requires several more ideas as well as a great deal of care. We will present the main ideas needed to carry out this composition, and in particular a technique called "aggregation of queries".

2. **Reducing the number of queries to $O(1)$:** Once the number of queries in the PCP is sufficiently small ($\mathsf{poly}(\log\log n)$ will do) one can use a code with much worse rate — more specifically the Hadamard code — in order to reduce the number of queries to be constant. We will see this step in the next few lectures.

3. **Applications in hardness of approximation:** Once we have establish a PCP with constantly many queries and constant alphabet size, we will discuss some of their applications in hardness of approximation. First we will see some of the more basic applications of it (such as hardness of Clique and APX hardness of various problems), which will motivate the question of optimal hardness of approximation results. This will lead us to discuss extreme forms of the PCP theorem, and in particular we will discuss the parallel repetition theorem, the long-code framework and the Unique-Games Conjecture.

As discussed above then, for the next installment in the course we shall assume the following improved version of Theorem 1.2.

Theorem 3.1 (PCP with poly-loglog number of queries)**.** *There are absolute constants $C, c > 0$ such that gap-CSG$[1, 1/\log(n)^c]$ is NP-hard on instances with alphabet size $\mathsf{poly}(\log n)$ and number of queries at most $(\log\log(n))^C$.*

Our focus on the next few lectures will be to deduce the following result, which is often referred to as the basic form of the PCP theorem:

Theorem 3.2 (PCP with constantly many queries and constant alphabet)**.** *There is an absolute constant $\varepsilon > 0$ such that gap-CSG$[1, 1-\varepsilon]$ is NP-hard on instances with alphabet size and number of queries at most $O(1)$.*