Massachusetts Institute of Technology
18.434: Seminar in Theoretical Computer Science

Lecturer: Mihai Pătraşcu
May 11, 2006

Unique Games and Inapproximability of MaxCut

Remember that the Goemans-Williamson algorithm can approximate MaxCut to a factor of:

$$\alpha_{GW} = \min_{0<\theta<\pi} \frac{\theta \cdot 2/\pi}{1-\cos\theta} \approx 0.879$$

In this lecture, we present a result of Khot, Kindler, Mossel and O'Donnell [1] which states that approximating MaxCut to within a factor $\alpha_{GW} + \varepsilon$, for any $\varepsilon > 0$, is "hard".

Here hardness is not based under the usual $\mathbf{P} \neq \mathbf{NP}$ assumption, but rather a weaker conjecture called the *Unique Games Conjecture* (UGC). This conjecture was introduced by Khot [2] in 2002, and states that a certain very meaningful problem is hard (e.g. NP-hard, or simply not in P). Recent algorithmic attacks on the problem have increased confidence in the truth of the conjecture, as they were successful only in-as-much as predicted by the conjecture. On the other hand, several inapproximability results based on UGC (such as the one discussed here) have brought the conjecture center-stage attention.

We begin by reviewing PCPs and introducing an alternate view of this concept as a multiprover interactive game. Then, we discuss a general proof strategy for transforming PCPs into inapproximability results, via the *label cover problem*. Armed with this understanding, we state UGC, and discuss its significance. Finally, we sketch the proof of the inapproximability of MaxCut

1 Why you should collaborate on problem-sets

It is easy to understand how PCPs can help professors a lot: if they require homework solutions to be written as probabilistically checkable proofs, they can test whether the student has a correct proof very quickly (probing just a few bits of the proof).

What is less obvious is that PCPs can also help students a great deal, if they solve problems in pairs. Say student A and student B come to the professor saying that they have a proof, but it is a waste of their time to actually write it down on paper. How can the professor make sure the students are telling the truth? He first puts the two students in different rooms. Then, he goes to A and asks for some k bits of the PCP. If the bits show the PCP is wrong, he immediately fails them. But if the bits are correct, he cannot really say anything: maybe A doesn't have a PCP, but he just manufactured the k bits to look like a good proof segment. Now the professor goes to B, and asks for just one of the k bits that he asked A. If B gives a different answer than A, the students are lying. Otherwise, the professor assumes the students really have a proof.

Why does this strategy work? If the students really have a PCP (say, with perfect completeness), they can convince the professor by just being truthful. Now assume the

statement is false, and the soundness of the PCP is p (a proof of a wrong statement is accepted with probability $\leq p$). Imagine asking student B each position in the proof, one at a time. This describes a PCP. By soundness, the verification algorithm will only accept with probability p. For a set of k bits which the algorithm will accept anyway, student A can just give the correct answer. Otherwise, he must lie about the k bits. Since at least one of the k bits is different, he will be caught with probability $\frac{1}{k}$. Thus, the soundness of the protocol is $1 - \frac{1-p}{k}$, which is bounded away from one for $k = O(1)$.

To summarize, our model describes a game between a verifier and two provers. The verifier tosses some random coins and sends a message privately to each of the two provers. Each message has $O(\lg n)$ bits. Then, each prover replies with $O(1)$ bits, after which the verifier accepts or rejects.

What about the soundness and completeness? Given such a game whose soundness is bounded away from the completeness, we can think of repeating the protocol t times to improve this gap towards one (like amplification for randomized algorithms). The repetition is still done in one round, by sending t questions to A, and t questions to B at the same time. Raz's parallel repetition theorem [3] shows that indeed this improves parameters exponentially in t. Thus, up to constant factors, we can assume soundness is ε and completeness is $1 - \varepsilon$, for any constant $\varepsilon > 0$.

We have seen that any language in NP can be proven in this framework. In fact, we see below that NP is exactly the class of languages provable by such games. It should be noted that many settings for interactive proofs have been studied (varying the number of rounds, the number of provers, the lengths of the messages etc), leading to a number of interesting results in complexity theory.

2 The Label Cover Problem

To formalize our model for interactive proofs, let V be the space of possible messages for the first prover, and W for the second prover. Also let M be the space of possible responses for both provers. We have $|V|, |W| = n^{O(1)}$ and $|W| = O(1)$. We can interpret the communication game as a labeled graph with constraints, as follows. Consider a bipartite graph, where one party is V and the other W. Every vertex is labeled with a value from M, representing the answer of the corresponding prover to the message identified by the vertex. For every choice of the verifier's random coins, if the messages sent are $v \in V$ and $w \in W$, draw an edge between v and w. To each edge we associate a constraint on the labels of the two end-points. The constraint is a subset of M^2 listing all pairs of answers which make the verifier accept.

Now remember that the protocol can achieve soundness ε and completeness $1 - \varepsilon$. A constraint (edge) is precisely an execution of the protocol, so the notion of "probability" for the protocol translates into a fraction of the edges. When the proof is correct, the graph will have a labeling which satisfied $1 - \varepsilon$ of the constraints. On the other hand, when the statement being proven is wrong, no labeling can satisfy even an ε fraction of the constraints.

Thus, by starting with a protocol for an NP-complete problem, we obtain a very strong

hardness result for approximating the maximum number of satisfiable edges. It is NP-complete to distinguish the case when a $1-\varepsilon$ fraction of the edges is satisfiable versus the case when an ε fraction of the edges is satisfiable.

Furthermore, our reduction to label cover makes it transparent that all languages provable in our framework are in NP. The question "can the provers have a strategy that convinces the verifier often?" is equivalent to "can a lot of the edges be satisfied?"

3 The Unique Games Conjecture

The hardness result for label cover is probably not so interesting in itself, since the problem is rather artificial. However, the problem can be used as a building block for many hardness results, as follows. First of all, most problems deal with a fixed alphabet (e.g. binary), so we encode the labels of M using an error-correcting code. Now we express the constraints as verification algorithms, which read bits of the two codewords for the two labels. Since codewords have a large distance, we can usually verify constraints probabilistically by reading just a few bits (like in PCPs). If, for instance, we design a test that reads 3 bits, we can express the test as constraints on three boolean variables. If we want constraints to have a special form (e.g. the XOR of the three bits is one), we must design a test which has that special form (e.g. it accepts only when it reads three bits whose XOR is one). The soundness and completeness of the test will dictate the approximation factor.

Thus, hardness proofs contain two iterations of the same idea: we first encode a PCP verifier (called the *outer verifier*) as a graph with edge constraints, and then encode a constraint verifier (the *inner verifier*) in the target problem.

Now consider MaxCut. We have a boolean decision about each vertex (which side of the cut?), so labels must be coded in a binary alphabet. Also, we only look at two nodes at a time (does the edge cross the cut?), so the inner verifier must make two queries. Unfortunately, there are only so many things we can do with two boolean queries. Nobody has been able to design an inner verifier for arbitrary constraints on M^2 which has a good ratio between completeness and soundness.

To avoid this limitation, Khot [2] proposed that we only consider a special class of constraints on M^2: permutation constraints — for some permutation $\pi : M \to M$, the constraint only accepts pairs $(x, \pi(x))$. This means we are reducing from the *unique label cover problem*, which is the label cover problem in which all edge constraints are permutations. Unfortunately, the NP-hardness of this special case is not known, but only conjectured:

Conjecture 1 (Unique Games Conjecture). *For any constant $\varepsilon > 0$, there exists a label set M of constant size, such that in the unique label cover problem, it is NP-hard to distinguish an instance in which $1-\varepsilon$ of the labels are satisfiable from an instance in which at most an ε fraction are satisfiable.*

In multiprover terminology, we are demanding that NP have a special type of PCPs, in which for every answer from one prover, there is a unique answer from the second prover that will satisfy the verifier. Pragmatically, however, it does not matter if unique label cover is

NP-hard, or simply not polynomially solvable for different reasons. We can call all problems that we reduce to "UG-hard."

One thing to note about this conjectured outer verifier is that it cannot have perfect completeness: it can be decided easily in polynomial time whether one can satisfy *all* edges. Indeed, the choice of one label fixes the labels for all neighbors, by the permutation constraints. To test if all edges are satisfiable, try all possibilities for one label, and try propagating the forced choices around the graph.

Another observation is that $|M|$ must indeed grow with ε for hardness to hold. Indeed, even a random assignment of labels will satisfy a fraction $\varepsilon = \frac{1}{|M|}$ of the edges. In fact, one can solve the problem even for higher values of ε as a function of $|M|$, using semidefinite programming (extensions of Goemans-Williamson). Several recent result have explored the boundary of this approach.

4 Inapproximability of MaxCut

When considering MaxCut, one thing to notice is that it is a special case of the unique label cover problem: $M = \{0, 1\}$ and all constraints are the permutation $0 \to 1, 1 \to 0$. Thus, what we plan to do is use hardness for the general problem to imply hardness for the special case! To understand what is happening, let us draw a parallel to SAT. First, one assumes SAT cannot be solved in polynomial time. A priori, there is no reason why the special case of 3SAT should also be hard. However, a reduction shows it is. Furthermore, a more complicated reduction via Håstad's PCP shows an optimal inapproximability of $\frac{7}{8}$ for 3SAT, just from hardness of the general SAT.

As discussed above, we only need to design a coding of the labels and an inner verifier. We use a very inefficient code, taking M to the space of boolean function over the cube $\{\pm 1\}^M$ (such a function is represented by 2^M bits). The code will associate to $i \in M$ the dictator function $f(x) = x_i$. In the business, this is called the *Long Code*.

Now let us define the inner verifier, which will be parameterized by an arbitrary $\rho \in (-1, 0)$. First, pick randomly $v \in V$ and two of its neighbors $w, w' \in W$. Let σ and σ' be the permutation constraints on the edges (v, w) and (v, w'). Also let $f_w, f_{w'}$ be the codes of w and w. For uniformly random $x \in \{\pm 1\}^{|M|}$, and y a ρ-correlated copy, our test will accept iff $f_w(x \circ \sigma) \neq f_{w'}(y \circ \sigma)$. Here we define $x \circ \sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(|M|)})$, i.e. x after we permute coordinates through σ.

The intuition is that f_w and $f_{w'}$ should be f_v (the label of v), permuted through σ and σ'. Then, the test is essentially a stability test on the function labeling v (i.e. f_v): we evaluate it at two random ρ-correlated inputs, and test if the value changes. If the function is a dictator, we have stability ρ, and accept probability $\frac{1-\rho}{2}$. If the function is far from a dictator, in the sense that all influences are small, it has stability at most that of the majority function, giving an accept probability of $\frac{\arccos \rho}{\pi}$. We can show that the intermediate regime does not occur (the function is not a dictator, but close to one). This relies on the gap in the label cover problem: either a lot of edges are satisfiable, or very few are.

Now imagine encoding the inner verifier as an instance of MaxCut. In one case, we will

have $\frac{1-\rho}{2}$ of the edges crossing the cut (the test accepts). In the other case, at most $\frac{\arccos\rho}{\pi}$ of the edges cross. Furthermore, it is NP-hard to distinguish between the cases. Then, we obtain an approximation ratio of $\frac{\arccos\rho}{\pi}/\frac{1-\rho}{2}$. But note that ρ was an arbitrary constant, so we can optimize over it to get the minimum ratio. This gives exactly the α_{GW} bound.

To complete the proof, one needs to show the accept probabilities claimed above. This part is somewhat technical, relying on the Fourier identities from last time for the calculations. For full details, see [1]. Here we outline the main steps:

- assume that $1-\eta$ of the edges in the label cover instance can be satisfied. Encode the labels of the solutions via the Long Code. Use the stability of the dictator functions to obtain that the accept probability is $(1-2\eta)\frac{1-\rho}{2} \approx \frac{1-\rho}{2}$.

- now assume codes are arbitrary (not dictators). Define $g_v(z) = \mathbb{E}_w[f_w(z \circ \sigma_{(v,w)})]$, where w is a random neighbor of w, and $\sigma_{(v,w)}$ is the constraint on the edge. Note that our test never looked at the labels on the V side. Instead, we try to recover v's label from g_v, the average opinion among neighbors about what the code of v should be.

- by direct computation, show that the acceptance probability is $\frac{1}{2} - \frac{1}{2}\mathbb{E}_v[\mathbb{S}_\rho(g_v)]$. If for $1-\varepsilon$ of vertices in V, the stability is at most the stability of majority, we have bounded the accept probability as desired.

- the remaining case is when ε of the vertices in V have stability significantly above majority. Then, they have a coordinate with $\Omega(1)$ influence. Since the coordinate is so influent, choose it as the label. Since g_v is an average of f_w's, a Markov bound shows most neighboring f_w's also have an influential coordinate. By the lemmas shown last time, we can work with $\mathrm{Inf}_i^{\leq k}(f_w)$ instead of $\mathrm{Inf}_i(f_w)$, so we know there can only be $O(1)$ influent coordinates. Guess one randomly as the label. This will satisfy a constant fraction of the edges in expectation. Choosing η small enough, this contradicts the gap in UGC. Thus, this case is impossible.

References

[1] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell: *Optimal Inapproximability Results for Max-Cut and Other 2-Variable CSPs?* FOCS 2004, pp 146-154

[2] Subhash Khot: *On the power of unique 2-prover 1-round games.* STOC 2002, pp 767-775.

[3] Ran Raz: *A Parallel Repetition Theorem.* SIAM J. Comput. 27(3): 763-803 (1998). See also STOC'95.