CS 354: Unfulfilled Algorithmic Fantasies April 24, 2019

Lecture 8: Unique Games Conjecture Continued

Lecturer: Aviad Rubinstein *Scribe: Andreas Garcia*

In this lecture we restate the Unique Games Conjecture and prove the conditional hardness of approximating max cut via reduction from Unique Games.

1 Unique Games Conjecture

The Unique Games problem is as follows.

- Input: graph $G = (V, E)$
- Constraint: a permutation $\pi_{u,v} : \Sigma \to \Sigma$ for each $u, v \in V$. Σ is a large, but constant sized set of "colors".
- Output: an assignment or "coloring" $\sigma : V \to \Sigma$. An assignment satisfies the constraint on edge (u, v) if $\sigma(u) = \pi_{u,v}(\sigma(v))$.

The problem of deciding whether or not there exists an assignment that satisfies constraints on *all* edges can actually be solved easily. Since the constraints are permutations, an assignment for one node uniquely determines the possible assignments for its neighbors so we can just enumerate through $|\Sigma|$ many possibilities for each connected component.

Instead, we consider the problem of distinguishing between these two sets of instances with a large gap between them. The conjecture states that distinguishing these is NP-hard.

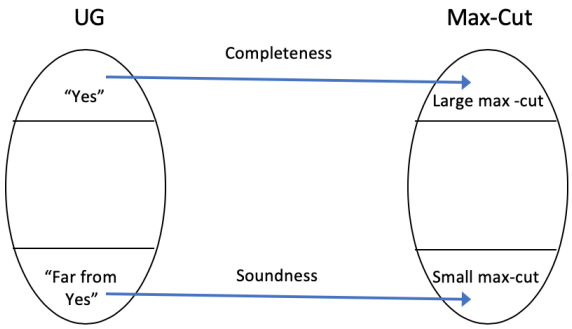
- "Yes": there exists an assignment σ that satisfies a $\geq (1 - \epsilon)$ fraction of the edges.
- "Far from Yes": all assignments σ satisfy an at most ϵ fraction of the edges.

We can now give approximation lower bounds conditioned on the conjecture.

Theorem 1.1 ([KKMO07])**.** *Assuming the Unique Games Conjecture, it is* NP*-hard to approximate Max-Cut to a factor better than* 0.879*.*

2 Reduction to Max-Cut: Attempt 1

Recall that our original goal from last lecture was to prove that the strange approximation factor of ≈ 0.878 for Max-Cut is conjectured to be the best we can do. We now show this by reducing Unique Games (UG) to Max-Cut. Since we want to show that Max-Cut is hard to approximate, our reduction must map to Max-Cut instances with a large gap:



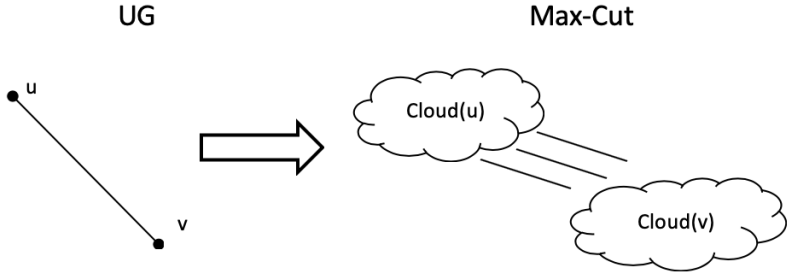
For each vertex $u \in V_{UG}$, we construct a set of $2^{|\Sigma|}$ vertices and label it as

$$\text{Cloud}(u) := \{u_x : x \in \{\pm 1\}^{|\Sigma|}\}$$

Then we add $2^{|\Sigma|}$ edges between each pair of clouds

$$(u_x, v_y) \in E \iff x = -\pi_{u,v}(y)$$

Where $\pi_{u,v}(y)$ denotes the permutation $\pi_{u,v}$ applied to the coordinates of y. We will modify this definition by adding weights later on, but for now, this is enough to give us the completeness part of the reduction.



Note that any cut S can be thought of as a class of functions f^u for $u \in V_{UG}$ where

$$f^u(x) = \begin{cases} 1 & \text{if } u_x \in S \\ -1 & \text{if } u_x \notin S \end{cases}$$

2.1 Completeness

We will show completeness and then attempt to show soundness. Recall the definition of a Long Code. We encode each element $i \in \Sigma$ as a function $f_i : \{\pm 1\}^{|\Sigma|} \to \{\pm 1\}$, $f_i(x) := x_i$.

Now, given a UG coloring σ, we define a cut S on our reduction as

$$f^u(x) = f_{\sigma(u)}(x)$$

I.e. an edge (u_x, v_y) crosses our cut if $f_{\sigma(u)}(x) = -f_{\sigma(v)}(y)$. Now suppose σ satisfies edge $(u, v) \in$

E_{UG}. Then for each edge (u_x, v_y) in our reduction, we have

$$\begin{aligned} f_{\sigma(u)}(x) &= x_{\sigma(u)} \\ &= -y_{\pi_{u,v}(\sigma(u))} \\ &= -y_{\sigma(v)} \\ &= -f_{\sigma(v)}(y) \end{aligned}$$

Therefore, if σ satisfies a $\geq (1-\epsilon)$ fraction of UG edges, then we have shown cut in our reduction that crosses a $\geq (1-\epsilon)$ fraction of edges.

2.2 Soundness (attempt 1)

For soundness we want to show that bad UG instances map to bad Max-Cut instances. Suppose σ does not satisfy the constraint on edge (u, v) so $\sigma(u) \neq \pi_{u,v}(\sigma(v))$. Then because our Long Code has distance 1/2 (any two code words differ in 1/2 of their bits), we have

$$\Pr_x[(f_{\sigma(u)}(x) = -f_{\sigma(v)}(-\pi_{u,v}(x))] = \Pr_x[f_{\sigma(u)}(x) = f_{\pi_{u,v}(\sigma(v))}(x)] = 1/2.$$

So only half of the edges between Clouds u and v cross the cut defined by long code f on the coloring σ. Since in the "far from yes" case we can satisfy at most ϵ-fraction of the constraints, this should give a soundness of $\leq (1/2 + \epsilon)$-fraction of the edges; compared to $\geq (1-\epsilon)$ in the "yes" case, we conclude that Max-Cut is UG-hard to approximate to within $1/2 + 2\epsilon$. Note: this is clearly wrong since we know that the Goemans-Williamson SDP gives a ≈ 0.878-approximation!

The above argument goes in the wrong direction: given a bad coloring for the UG instance we construct a bad cut. This is not so interesting because every instance (of both problems) has bad assignments. Instead, we want to show that given a good cut, we can recover a good coloring.

Since we think about cuts as encoded in the Long Code, we would like to encode the intersection of a good cut with cloud to obtain an assignment to the corresponding variable. This doesn't work well when the intersection of the cut and cloud is far from every codeword. For example, the majority function:

$$\text{Maj}(x) := \text{sign}\big(\sum x_i\big)$$

Observe that since the majority is function is odd, the corresponding cut actually crosses all the edges. Since the majority function is symmetric accross its coordinates, it doesn't "look" at all like any Long Code codeword (we will formalize this soon). So it doesn't tell us anything about any coloring for the UG instance.

Fortunately, we can adjust our reduction so that cuts that are not close to code words have low value.

3 Fixing the reduction

To fix our problem, we want to decease the value of cut functions that are far from Long Codes. For this, we use the fact that Long Codes are *locally testable*. This means that we can test if a string is a codeword or far from every codeword by reading only a few bits. In the Long Code case, we only need to read *two* bits to test if string is a codeword!

- Test 1: pick a random x and test that $f(x) = -f(-x)$. This is satisfied by a long code but it is also satisfied by any odd function (like majority) so it is not a good test.
- Test 2: given x, sample $z = x +$ noise by flipping each bit of x with probability p. Then check that $f(x) = -f(-z)$.

To see why Test 2 works, we see that if f is a codeword, i.e. $f \equiv x_i$ for some $i \in \Sigma$, then

$$\Pr\left[f_i(x) = -f_i(-z)\right] = 1 - p$$

Contrast that to the majority function:

$$\Pr\left[\text{Maj}(x) = -\text{Maj}(-z)\right] = \theta/\pi$$

Where $\cos(\theta) = z \cdot x = 1 - 2p$. Minimizing $\dfrac{\theta/\pi}{1-p}$ over all values of p gives the constant ≈ 0.878.

So we can distinguish between Maj and Long Code codewords, but what about other functions? It turns out that all Boolean functions either perform just as bad as Majority on this test, or "look like a long code codeword".

3.1 Majority is Stablest Theorem

Theorem 3.1 ([MOO05]). *For all ϵ, p, there are constants k, δ such that for all $f : \{\pm 1\}^\Sigma \to \{\pm 1\}$, one of the following holds*

- $\Pr\left[f(x) = -f(-z)\right] \leq \Pr\left[Maj(x) = -Maj(-z)\right] + \epsilon$
- *f "looks like a long code codeword": formally f is ϵ-close to a k-*junta.

Long Code codewords are called *dictatorships* because their value is determined by a single "dictator" coordinate. This is generalized to *k-juntas*, which are functions that are completely determined by k variables (i.e. $f_T(x) := g((x_j)_{j \in T})$ for some set T of size k).

Intuitively, this theorem can be translated as "majority vote is the stablest with the exceptions of dictatorships and juntas, which are much more stable".

3.2 Weighted edges

Now we are ready to revise our reduction using this test. Instead of just adding edges, we add weighted edges between each pair u_x, v_y:

$$w(u_x, v_y) := \Pr_z\left[z = -\pi_{u,v}(y)\right]$$

Where the probability is over z sampled as x with each bit flipped with probability p.

With this change, it is no longer true that the cut from our previous completeness argument crosses all edges between clouds with satisfying assignments. Instead, we have that the cut crosses a $1 - p$ fraction of these edges, which is still good.

3.3 Actual Soundness

For soundness, we want to show that if there exists a cut that does not have low value in the reduction, then there exists an assignment in the UG instance that satisfies $> \epsilon$ fraction of the edges. Intuitively, given some cut f on our reduction, we have that either:

- f^u is far from all k-junta. According to the Majority is Stablest Theorem, this means that only a small θ/π-fraction of edges are crossed so the cut will have low value.
- f^u is close to a k-junta. In this case we can assign a color to $u \in V_{UG}$ by picking randomly from the junta. If v's function is also close to a k-junta and those juntas intersect in at least one coordinate[1] , then we have that

$$\Pr\left[\sigma(u) = \pi_{u,v}(\sigma(v))\right] \approx \frac{1}{k^2} \gg \epsilon$$

  Therefore, there must be an assignment that satisfies $> \epsilon$ fraction of the edges.

References

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007.

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low in.uences invariance and optimality. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 21–30, 2005.

[1] To make this argument completely rigorous we have to think why the juntas should indeed intersect...