6.842 Randomness and Computation February 7, 2022

Lecture 3

Lecturer: Ronitt Rubinfeld *Scribe: Edward Jin*

Today we continue the proof from before of using LLL to get an improved bound on the 2-hypergraph coloring problem. There were some issues with the proof from before, and so we'll go through the proof again.

As a reminder, we have the Lovász Local Lemma from before:

Lemma 1 (Lovász Local Lemma) *Suppose that we have events $A_1, \ldots, A_n$ with dependency digraph D such that $\mathbb{P}(A_i) \leq p$, D has degree at most d, and $e \cdot p \cdot (d+1) \leq 1$, then the probability that none of the bad events occur is*

$$\mathbb{P}\left[\bigcap_i \overline{A}_i\right] > 0.$$

1 Continuing Hypergraph Coloring

Theorem 2 *Let X be a set of elements. Given subsets $S_1, \ldots, S_m \subseteq X$ where all subsets have size $|S_i| = \ell$, if each S_i intersects at most d other S_j's, $2e(d+1) \leq 2^{\alpha\ell}$, and $ed^2 p^{\frac{1}{d^2+1}} < \frac{1}{2}$ where ℓ, d are constants, $p \equiv 2^{(H(\alpha)-1)\ell+1}$ where $H(\alpha)$ is the binary entropy function, and α is any constant, then we can find a 2-coloring of X in polynomial time such that no S_i is monochromatic.*

Proof We define a bad set to be one in which there are $\leq \alpha\ell$ objects of a different color. On the first pass, for every $j \in X$, we will color randomly. We retry if there are any 'bad connected components' (defined below) of size $O(d \log m)$. In the second pass, we brute force all the bad connected components in order to find a valid coloring that keeps every other set monochromatic.

Remark This is a quite important idea, of using a randomized algorithm to do most of the work, and then shattering the problem into many small subproblems that can be brute forced.

The problem with defining a connected component only in terms of the dependency digraph is that there could be a good set that could intersect multiple distinct bad sets, which complicates the analysis in pass 2. To fix this problem, we will attempt to combine any bad sets that intersect the same good component. To this end, we will look at D^2, the *square* of the dependency digraph. Two sets are connected if there is a path of length either 1 or 2 between the nodes in the dependency digraph. Notably, if two bad sets intersect the same good set, then they must be connected in D^2. The degree of the squared digraph, and hence of any connected component, is now at most $d + d(d-1) \leq d^2$.

Letting S be a maximal set of connected bad nodes in D^2, we define $S \cup N(S)$ to be a 'bad connected component' (BCC), where $N(S)$ gives the neighborhood of all nodes in S. This ensures that all the good sets that overlap with multiple distinct bad sets are now all grouped up into the same BCC, making each BCC's recolorings independent of each other. Further, the size of the BCC is bounded by d times the number of bad sets in the connected component, as each bad set has at most d neighbors.

Some questions come up:

- How can we guarantee that the second pass can even fix the connected components?
- How fast is the brute forcing of the second pass?
- How many times do we need to retry the first pass?

We'll first show existence of a good solution in the second pass with LLL and the probabilistic method. Consider the case where we randomly recolor every bad set. If S_i is not bad and at most $\alpha\ell$ nodes are in bad neighbors, then we know that no matter what recolorings we do to the bad neighbors of S_i, S_i will still be bichromatic, by definition. On the contrary, if S_i is not bad and has at least $\alpha\ell$ nodes in bad

neighbors, then all of the bad nodes get recolored, and the probability that S_i becomes monochromatic is bounded by $2^{-\alpha\ell+1}$. The probability that the bad sets remain bad is bounded by $2^{-\ell+1}$, as before, which is much smaller than the probability S_i becomes monochromatic, so we don't need to worry about it here. Each set S_i still has degree at most d in the (normal, unsquared) dependency graph D, and so LLL guarantees that a solution exists if $2e(d+1) \leq 2^{\alpha\ell}$.

In terms of runtime, since each BCC's recolorings are independent of each other, we can brute force all of them separately, and each one of them takes time $O((2^\ell)^{O(d \log m)}) = m^{O(\ell d)}$. Overall then, the time for brute-forcing all colors for all BCCs is at most m times this, which is very slow but is polynomial time!

Remark We can also just recurse on the subcomponents to reduce the runtime to something nicer.

The probability that one set is bad can be shown to be $\mathbb{P}[S_i \text{ bad}] \leq p \equiv 2 \cdot 2^{(H(\alpha)-1)\cdot\ell}$, where H is the binary entropy function. We'll now try to bound the probability that all sets in any connected subgraph of size s (in D^2) are all bad. This probability is at most the probability that all the nodes of an independent set of size s' is all bad within that connected subgraph, which is $p^{s'}$. Further, this probability is good enough for us to bound the size of any BCC, as the bad sets in the BCC are a connected subgraph by definition and would be detected. The probability that any all-bad component of size s survives is bounded by the number of potential all-bad components in D^2, times $p^{s'}$.

Unfortunately, the size of the independent set can be 1 (as in a clique), and the number of components can be exponential. We will improve both of these bounds with the degree bound.

Given a subgraph of size s of degree $\leq \Delta$, we can find an independent set in that subgraph of size $\geq \frac{s}{\Delta+1}$ with a greedy approach - every vertex we pick disqualifies at most $\Delta+1$ vertices, so we have at least that many vertices in our independent set. This improves s' to at least $\frac{s}{d^2+1}$.

What about the number of possible big components? We will use a well-known fact that the number of subtrees of size k in a m-node graph of degree at most Δ is $m(e\Delta)^k$. (This comes from the Catalan numbers and enumerating DFS on a rooted tree.) Further, if no big subtree survives, then no BCC can survive either, as a BCC must contain a big subtree. The expected number of all-bad subtrees of size $\geq s$ that survive is then bounded by $\sum_{k=s}^{m} m(ed^2)^k \cdot p^{\frac{k}{d^2+1}} \leq \sum_{k=s}^{m} m \cdot \frac{1}{2^k} \leq \frac{m}{2^{s-1}}$ by assumption, so choosing $s = \log 4m$ and using Markov's inequality, the probability there are no large BCCs is at most $\frac{1}{4}$. This means that the first pass suceeds with constant probability if we retry the first step with a bound of $d \log 4m$, as the size of a BCC is at most ds.

The first pass takes polynomial time to randomize and find connected components, and the second pass also takes polynomial time as stated above, so this method of finding a 2-coloring overall works in polynomial time. ■

2 Polynomial Identity Testing

We need to check if two polynomials $P(x), Q(x)$ of degree at most d are identically equal. Another problem is if we are given a polynomial $R(x)$, and need to check if it is identically 0. These are essentially equivalent, as $P(x) - Q(x) \equiv 0 \iff P(x) = Q(x)$.

Since a degree d polynomial has at most d roots, then we know that there are at most d points for which P and Q are equal. So if we evaluate P and Q on $d+1$ points and they all agree, P must be equal to Q. We'll see a randomized version and multivariate version of this next lecture, and see an application in communication complexity.