Lecture 11

Tuesday, February 09, 2016
11:51 PM

Lovász Local Lemma, Part 2

Recall the general version of LLL:

Thm: Let $B_1, \dots, B_n$ be a set of "bad" events, and let d be a bound on the max degree in the dependency graph. Then if:

$$\forall i, \quad \Pr[B_i] \le x_i \prod_{\substack{(i,j) \text{ edge} \\ \text{in dep graph}}} (1 - x_j) \quad \text{for } x_i \in [0,1)$$

$$\text{Then } \Pr\left[\bigwedge_{i=1}^{n} \overline{B}_i\right] \ge \prod_{i=1}^{n} (1 - x_i) > 0.$$

Remarks:

(1) Even if $d = 0$, $\Pr[\bigwedge \overline{B}_i]$ is exponentially small.
So, LLL as above is non-constructive.

(2) Often times, each B_i is a function of subset S_i of an underlying set of independent random variables $X = \{x_1, \dots, x_N\}$. Then, we want $\forall i$, $\#\{j : S_i \cap S_j \neq \emptyset\} \le d$.

Example: k-SAT with N variables and with each var occurring in $\le \frac{d}{k}$ clauses.

Corollaries:

Symmetric LLL: If $\forall i$, $\Pr[B_i] \le p$ and $ep(d+1) \le 1$,
then $\Pr[\bigwedge \overline{B}_i] > 0$.

Pf: Set $x_i = \frac{1}{d+1}$

Asymmetric LLL: If $\forall i$, $\sum_{\substack{(i,j) \\ \text{dependent}}} \Pr[B_j] \le \frac{1}{4}$, then $\Pr[\bigwedge \overline{B}_i] > 0$.

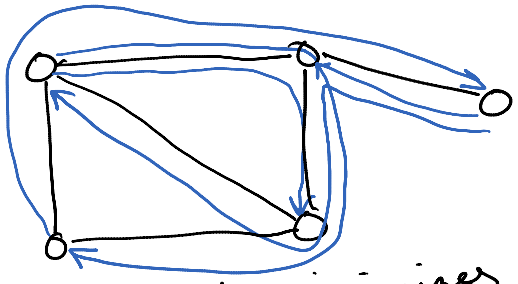
Pf: Set $x_i = 2 \cdot \Pr[B_i]$.

Packet Routing in Networks [Leighton-Maggs-Rao]

Network with bedirectional channels; n packets where i^{th} packet $\dots$ from s_i to t_i thru path P_i.

Each node has a queue and only one packet can go on edge in one step. If more than one packet arrives at a node at the same time, one packet goes thru and others are queued. Synchronized.

Ex:



Goal: Prepare a schedule that minimizes time for all packets to reach destinations.

Two key parameters: (1) d = Dilation
= Length of longest path P_i

(2) c = Congestion
= $\max_e$ [# paths passing thru e]

Observation 1: $OPT_{Time} \geq \max(c, d) = \Omega(c+d)$

Observation 2: $OPT_{Time} \leq c \cdot d$

Lemma: $OPT_{Time} = O(c + d \log(nd))$

Pf sketch: Delay packet i by T_i, uniformly chosen from $[1, \frac{\alpha c}{\log(nd)}]$, then send to destination, for a large constant α.
Then, max # of packets going thru any edge is $O(\log(nd))$ whp.
Why? For any fixed edge e and time t,

$$\Pr[> \log(nd) \text{ packets passing thru } e \text{ at } t]$$
$$\leq \sum_{k=\log(nd)+1}^{c} \binom{c}{k} \left(\frac{\log(nd)}{\alpha c}\right)^k \left(1 - \frac{\log(nd)}{\alpha c}\right)^{c-k}$$
$$\leq \sum_{k > \log(nd)} \left(\frac{e \log(nd)}{\alpha k}\right)^k \leq c \cdot \left(\frac{e}{\alpha}\right)^{\log(nd)} \leq \frac{1}{(nd)^3}$$

$\leq nd$ possibilities

$$\Pr[\exists e, t, > \log(nd) \text{ passing thru } e \text{ at } t] \leq \frac{1}{nd}$$

Then packets one at a time to make schedule feasible.

Total time $\leq \log(nd) \cdot \left(d + \frac{\alpha c}{\log(nd)}\right) = O(c + d\log(nd))$.

Max Queue Size $\leq \log(nd)$

What happens if we use LLL instead of union bound?

Suppose delays are random in $\left[\frac{\alpha c}{\lg(cd)}\right]$

B_e is event that $\geq \lg(cd)$ packets pass thru e at some time.

Degree of dependency graph is $\leq c \cdot d$.

$$\Pr[B_e] \leq \left(d + \frac{\alpha c}{\lg(cd)}\right) \cdot \sum_{k \cdot \lg(cd)}^{c} \binom{c}{k} \cdot \left(\frac{\lg(cd)}{\alpha c}\right)^k \leq \frac{1}{e\, c^2 d^2}$$

Then, $e \cdot \Pr[B_e] \cdot (cd+1) \leq \frac{1}{cd} < 1$. By LLL, result follows.

Then, total schedule time $\leq O(c + d\lg(cd))$

Queue length $\leq O(\lg(cd))$

By using divide and conquer, and careful parameter tuning, we can bring down schedule time to $O(c+d)$ and queue length to $O(1)$!

Constructive LLL

- Event guaranteed by LLL has exponentially small probability. But we want to efficiently find a point in sample space where event occurs!
- Get back to k-SAT example. If each clause intersects with $\leq 2^k/e$ clauses, then LLL guarantees satisfying assignment. How to find it efficiently?
- Note that even when clauses are disjoint, random assignment is satisfying with exponentially small prob. But then finding a sat assignment trivially in poly time!

a result by Ph.D. student Robin Moser in '09 [illegible] 2^{k-3} later

finds sat. assignment if clause overlap is ≤ 2, ...
improved to the optimal $2^k/e$ by Moser and Tardos.

Algorithm:

```
Solve(φ):
    Pick a random assignment A for x_1, x_2, ..., x_n
    While A violates a clause C:                          (*)
        A ← Fix(C, A)
    return A

Fix(C, A):
    For variable v in C:
        Set A(v) randomly
    While ∃ clause D overlapping with C that A violates:  (**)
        A ← Fix(D, A)                                     (***)
    return A
```

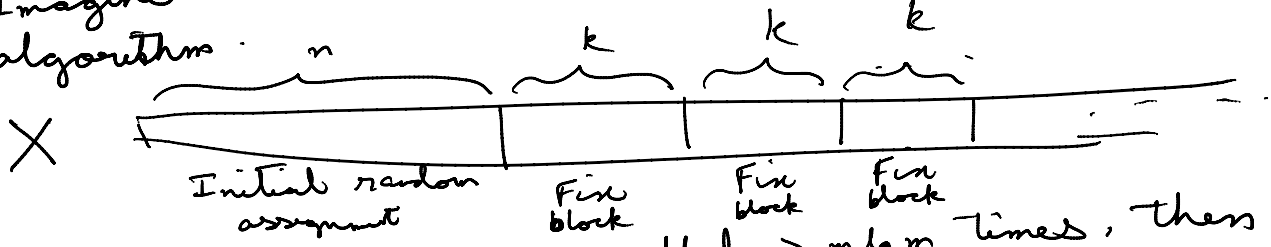
↖ In fact, this last if can be omitted!!

Claim: If Fix terminates, then assignment returned by Fix(C, A) satisfies C and doesn't violate any clause which A satisfied.

Pf: Easy.

So, number of violated clauses keeps decreasing if Fix terminates. But how do we show that?

Imagine all random bits needed laid out before the algorithm:



Thm: If Fix is called $> m \lg m$ times, then ... missed. Contradiction!

The first $n + k \lg m$ bits of X can be compressed.
(Entropy compression)

Compression procedure:
- If Fix is called in line (*), change first n bits to new assignment and replace block by $\lg m$-size clause id.
- If Fix is called in line (**), change first n bits to new assignment and replace block by i if D is the i'th clause intersecting C.
- At line (***), insert a special "RET" symbol after the last edited block.

After T calls of Fix, length of string is
$\leq m \lg m + T(\lg d + O(1)) + n$

Note that compression is reversible, because unique ass violates a clause.

~~n~~ $+ Tk \geq m \lg m + T(\lg d + O(1)) +$ ~~n~~ if $T = m \lg m$

$= T(\lg d + O(1))$

$\lg d \leq k - O(1)$

$d \leq 2^{k - O(1)}$ Done!