Lovász Local Lemma

Let $E_1, \ldots, E_n$ is set of "bad" events.

say want to show $\Pr\left(\bigcap_{i=1}^{n} \overline{E_i}\right) > 0$ "pos prob that nothing bad happens"

2 cases where easy

(1) E_i are mutually indep

(2) $\sum_{i=1}^{n} \Pr(E_i) < 1$ union bound suffices

LLL is clever combination

Defn: E mutually indep of $E_1, \ldots, E_n$ if for any subset $I \subseteq [1..n]$

$$\Pr\left(E \mid \bigcap_{j \in I} E_j\right) = \Pr(E)$$

Defn: A dependency graph for $E_1, \ldots, E_n$ is $G = (V, E)$

$V = \{1, \ldots, n\}$ and E_i mutually indep of $\{E_j \mid (i,j) \notin E\}$

Thm Lovász Local Lemma

Let $E_1, \dots, E_n$ be set of events s.t.

① $\Pr(E_i) < p \quad \forall i$

② the max degree in dependency graph is d

③ $4dp \le 1$

Then $\Pr(\bigcap_{i=1}^{n} \bar{E}_i) > 0$

Proof Show $\Pr(\bigcap_{i \in S} \bar{E}_i) > 0$ and $\Pr(E_k \mid \bigcap_{i \in S} \bar{E}_i) \le 2p \quad \forall k$

by induction on $|S|$

Base case: $|S| = 1 \quad \Pr(\bar{E}_i) = 1 - \Pr(E_i) = 1 - p > 0$

Case 1: no edge $k-i$
$\Pr(E_k \mid \bar{E}_i) = \Pr(E_k) \le p$

Case 2: $\exists$ edge $(k-i)$
$\Pr(E_k \mid \bar{E}_i) = \frac{\Pr(E_k \cap \bar{E}_i)}{\Pr(\bar{E}_i)} \le \frac{p}{1-p} < 2p$
$(p < \frac{1}{4})$

Induction Step:

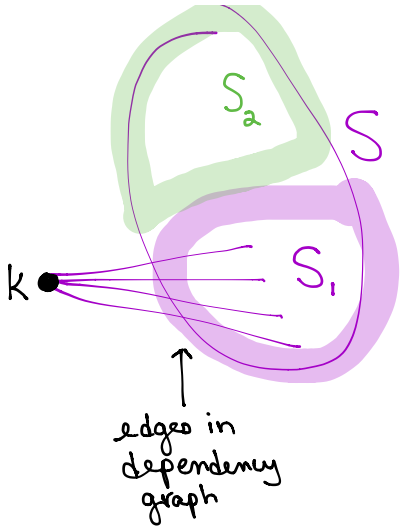
IH: $|T| < s \quad \Pr(\bigcap_{i \in T} \bar{E}_i) > 0 \quad \Pr(E_k \mid \bigcap_{i \in T} \bar{E}_i) \le 2p$

wlog $S = \{1, \dots, s\}$

$$\Pr(\bigcap_{i=1}^{s} \bar{E}_i) = \Pr(\bar{E}_1) \Pr(\bar{E}_2 \mid \bar{E}_1) \Pr(\bar{E}_3 \mid \bar{E}_1 \cap \bar{E}_2) \cdots \Pr(\bar{E}_s \mid \bar{E}_1 \cap \bar{E}_2 \cap \dots \cap \bar{E}_{s-1})$$

$$= \prod_{i=1}^{s} \left(1 - \Pr(E_i \mid \bigcap_{j=1}^{i-1} \bar{E}_j)\right) \ge \prod_{i=1}^{s} (1-2p) > 0$$

Next, prove $\Pr\left(E_k \mid \bigcap_{i \in S} \overline{E}_i\right) \leq 2p$



If $S_2 = S$,

$$\Pr\left(E_k \mid \bigcap_{i \in S} \overline{E}_i\right) = \Pr(E_k) \leq p$$

So assume $S_2 < S$

$$F_S = \bigcap_{i \in S} \overline{E}_i \qquad F_{S_1} = \bigcap_{i \in S_1} \overline{E}_i \qquad F_{S_2} = \bigcap_{i \in S_2} \overline{E}_i$$

$$\Pr(E_k \mid F_S) = \frac{\Pr(E_k \cap F_S)}{\Pr(F_S)} = \frac{\Pr(E_k \cap F_{S_1} \mid F_{S_2}) \cancel{\Pr(F_{S_2})}}{\Pr(F_{S_1} \mid F_{S_2}) \cancel{\Pr(F_{S_2})}} = \frac{\Pr(E_k \cap F_{S_1} \mid F_{S_2})}{\Pr(F_{S_1} \mid F_{S_2})}$$

$$\Pr(E_k \cap F_{S_1} \mid F_{S_2}) \leq \Pr(E_k \mid F_{S_2}) = \Pr(E_k) \leq p$$

$$= 1 - \Pr\left(\bigcup_{i \in S_1} E_i \mid F_{S_2}\right)$$

$$\geq 1 - \sum_{i \in S_1} \Pr(E_i \mid F_{S_2}) \qquad \text{union bound}$$

$$\geq 1 - \sum_{i \in S_1} 2p \qquad \text{induction hypothesis, } |S_2| < |S|$$

$$\geq 1 - 2dp \geq \frac{1}{2}$$

$$\Pr(E_k \mid E_s) \le \frac{p}{1/2} = 2p$$

Application 1: k-SAT

Given k-SAT formula n vars, m clauses

Thm If no var appears in $> T = \frac{2^k}{4k}$ clauses, then formula has satisfying assignment

Proof Consider random assignment where each var is T with prob $\frac{1}{2}$ independently.

E_i event that clause i **not** satisfied

- $p = \Pr(E_i) = 2^{-k}$
- E_i mutually indep of clauses that don't share vars

$\Rightarrow d \le kT = 2^{k-2} \Rightarrow 4dp \le 1 \underset{\text{LLL}}{\Rightarrow} \Pr(\bigcap_{i=1}^{m} \overline{E_i}) > 0$ Satisfying assignment

Application 2: Packet Routing

Graph; n packets

each packet has

s_i source

t_i destination

and a specific path P_i $\quad s_i \overset{P_i}{\rightsquigarrow} t_i$

only one packet can traverse an edge per time unit

so 

Schedule: specifies for each packet when to move, when to wait

$d = \max_i |P_i|$ dilation

$c = \max_e (\#\text{ paths } P_i \text{ that use } e)$ congestion

How long for each packet to reach its destination?

$\Omega(c+d)$? ? ? $O(cd)$

[Leighton, Rao, Maggs] ∃ schedule of length $O(c+d)$ always!

indep of n!

We'll see weaker version

schedule of length $O\left((c+d)(1+\alpha)^{\alpha(\ln^*(c+d))}\right)$

α: const

$\ln^*(n)$ # times need to take ln to get ≤ 1

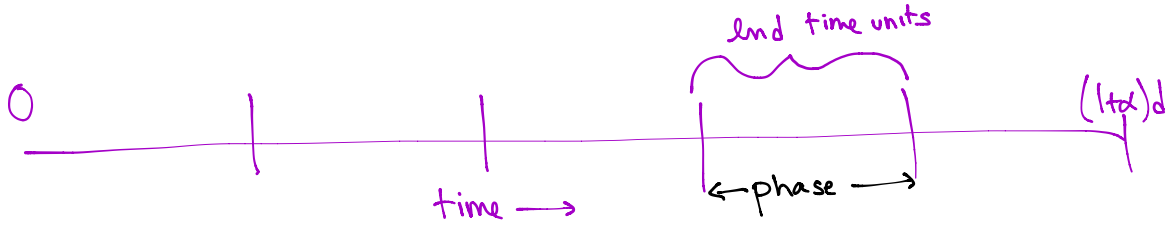
grows very slowly

assume $c=d$

Approach: for each packet, assign an initial delay $[1, \alpha d]$

Suppose that after initial delay packets can move without waiting

$\Rightarrow$ arrive at destination after at most $(1+\alpha)d$ steps



Show that in a phase, max congestion on any edge is $\ln d$

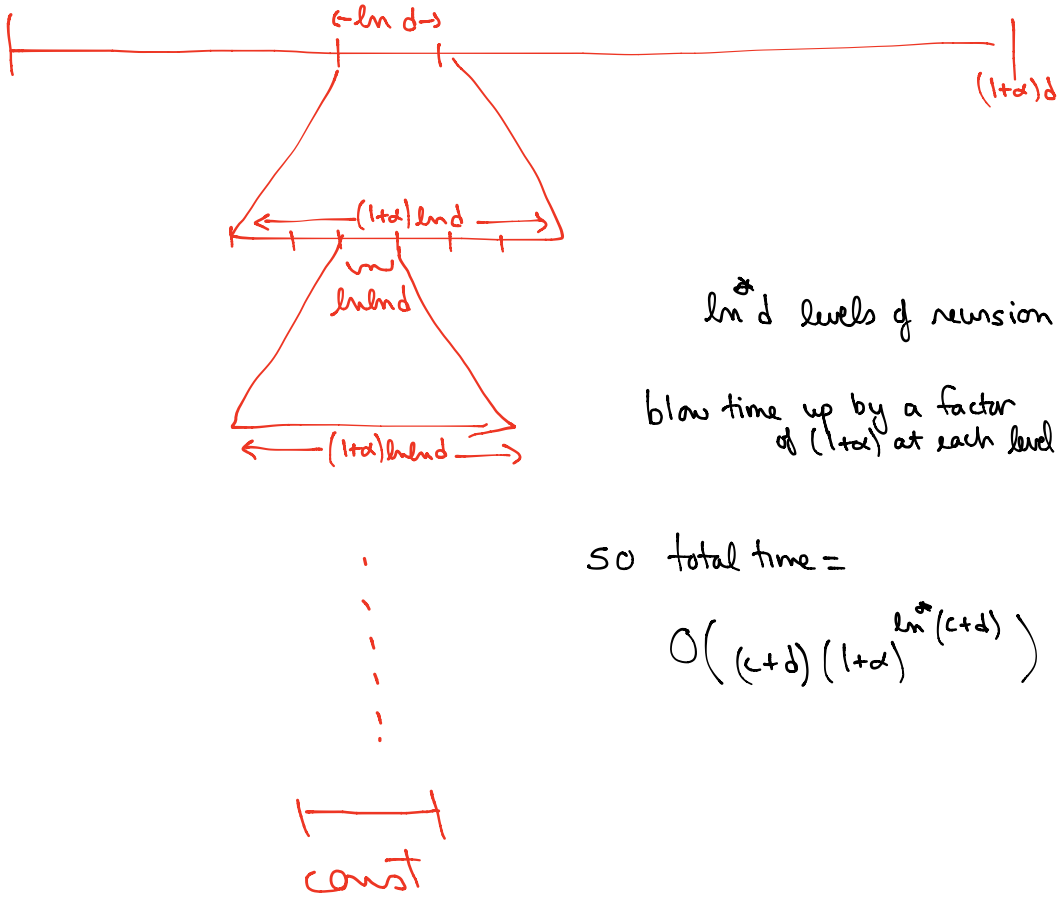
$\Rightarrow$ within a phase we have a subproblem

with $c' = d' = \ln d$

Solve recursively. [introduce random delay, divide into phases of length $\ln\ln d$...]

Keep doing this till subproblems have constant size

then just do greedy.



$\ln^* d$ levels of recursion

blow time up by a factor of $(1+\alpha)$ at each level

so total time =

$$O\left((c+d)(1+\alpha)^{\ln^*(c+d)}\right)$$

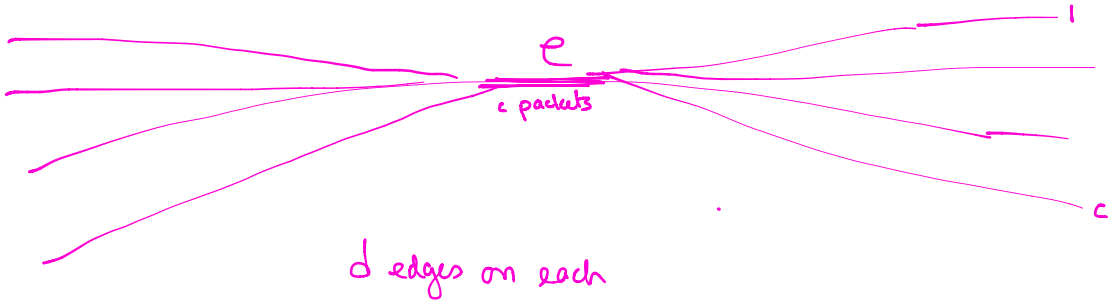
Lemma For α constant (large enough), the congestion in each phase $\leq \log d$ with positive probability

A_e: event that edge e has congestion $> \alpha \log d$ in some phase

depends on c packets that use it.

A_e and A_f are indep unless $\exists$ packet traverses both

A_e dependent on $\leq cd$ other packets



d edges on each

$\Rightarrow$ degree in dependency graph $\leq cd = d^2$

$Pr(\forall e\ \overline{A_e})$? use Lovasz Local Lemma

need upper bound on $\Pr(A_e)$ use Chernoff bound

$$E\left(A_e^{\text{phase } i}\right) \le d \cdot \ln d \cdot \frac{1}{\alpha d} = \frac{\ln d}{\alpha} \leftarrow \text{call this } \mu$$

d: # packets that use e

$\ln d$: # time steps in interval

$\frac{1}{\alpha d}$: for fixed $t' \in [t, t + \ln d]$ and fixed packet r, $\Pr(r \text{ traverses } e \text{ at time } t')$

indep for different packets $\quad Bin\left(d, \frac{\ln d}{\alpha d}\right)$

By Chernoff,

$$\Pr\left(A_e^{\text{phase } i} > \underbrace{\ln d}_{[1+(\alpha-1)]\mu}\right)$$

δ

$$\le \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \le \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu}$$

$$= \left(\frac{e}{\alpha}\right)^{\ln d} \le \frac{d^{-4}}{\alpha} \quad \text{for } \alpha = e^{11}$$

Since $\le \frac{(1+\alpha)d}{\ln d} < \alpha d$ phases

$$\left(\frac{e}{e^{11}}\right)^{\ln d} = \frac{1}{e^{10 \ln d}} = \frac{1}{d^{10}} = d^{6} \cdot \frac{1}{d^{4}}$$

$d^6 > e^{11}$

$$\Pr(A_e) \le \alpha d \Pr(A_e^{\text{phase } i}) \le \alpha d \frac{d^{-4}}{\alpha} = d^{-3}$$

So $4 d^{-3} \cdot \underbrace{cd}_{\substack{\text{max degree} \\ \text{in dependency} \\ \text{graph}}} = 4 d^{-1} \le 1$ for $d \ge 4$

$\Rightarrow$ by local lemma $\exists$ choice of initial delays s.t. no period has congestion $> \ln d$.

Efficient Algs for finding outcome guaranteed by LLL

..... [Moser] [Moser, Tardos]

Thm S set of m length k clauses, s.t. support of each clause intersects $\leq 2^{k-c}$ clauses [c is a sufficiently large constant]

Then the clauses in S simultaneously satisfiable

Proof: constructive version

Fix ordering $C_1, \ldots, C_m$ of clauses

Solve SAT

Choose random assignment of vars

for $i := 1$ to m

if C_i not satisfied

Fix(C_i)

Fix(C)

replace vars in C by new random vars

while $\exists$ clause D that shares vars with C & D not satisfied

choose such D with smallest index

Fix(D)

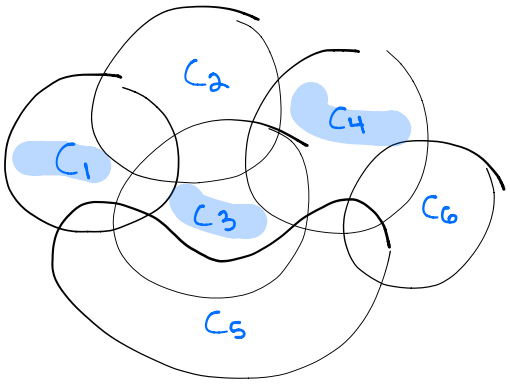
Observation

Once Fix(C_i) terminates, C_i remains satisfied

after each Fix C_j $j>i$

$\Rightarrow$ suffices to prove that Fix(C_i) terminates in reasonable amt of time

Recursive call structure



Proof approach: "entropy compression"

if doesn't terminate quickly $\Rightarrow$ gives alg to compress

ℓ random bits $\Rightarrow$ $< \ell$ bits, which is impossible

A random var cannot be compressed losslessly into a string of expected size $<$ Shannon entropy of r.v.

Suppose Fix( ) called t times

$\Rightarrow$ # random bits used $n + t \cdot k$

We show how to "encode" entire computation so that we can reconstruct all random bits used.

encoding will use $m(\log m + 4) + t(\log d + 4) + n$

$d = 2^{k-c}$

$\Rightarrow$ random bits can't be compressed

$m(\log m + 4) + t(\log d + 4) + n \geq tk + n$

$\Leftrightarrow t(k - \log d - 4) \leq m(\log m + 4)$

$\log d < k - 4$

$\Rightarrow t = O(m \log m)$

Encoding

① n bits for final assignment

② $\forall$ clause C_i s.t. $\exists$ call to Fix(C_i) from main $\log m$ bits

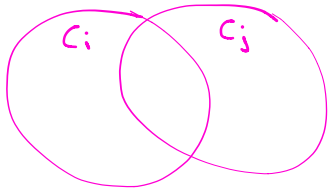
(3) $\forall$ recursive call $\log d$ bits to indicate which of dependent clauses is being called

E.g. clauses that are dependent with C_i

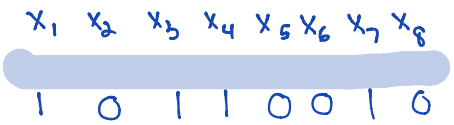
$C_{i_1}, C_{i_2} \cdots C_{i_j}$ C_{i_d} (one of these is C_i)

when $Fix(C_{i_j})$ called from $Fix(C_i)$
record j using $\log d$ bits

Upon termination of recursive call, record termination symbol



If $Fix(C_j)$ called from $Fix(C_i)$, then know precisely current values of bits in C_j

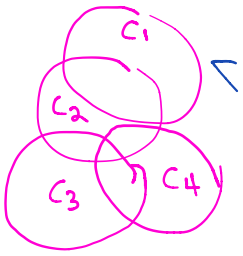


$C_1 = (x_1 \vee x_2 \vee x_3)$

$C_2 = (\bar{x}_1 \vee \bar{x}_3 \vee x_5)$

$C_3 = (x_5 \vee \bar{x}_6 \vee x_7)$

$C_4 = (x_5 \vee x_6 \vee x_8)$



initially C_2 & C_4 not satisfied

Original random bits:

$v_1, \dots, v_n$, $r_1^{(1)} \dots r_k^{(1)}$ $r_1^{(2)} \dots r_k^{(2)}$ - - - - $r_1^{(t)} \dots r_k^{(t)}$ $n + tk$

Encoding

$00\, c_1 \cdots c_{\log m}$ $01\, n_1^{(1)} \cdots n_{\log d}^{(1)}$ $01\, n_1^{(2)} \cdots n_{\log d}^{(2)}$ $||$ $n_1^{(3)} \cdots n_{\log d}^{(3)}$ $||$ $||$ $n_1^{(4)} \cdots n_{\log d}^{(4)}$ $||$ $||$

C_i intersects $i_1 \dots i_d$

$1 \le j \le d$ — C_{i_j} intersects $C_{r_1} \dots C_{r_d}$

$1 \le l \le d$

$1 \le l' \le d$

$1 \le j' \le d$

- - - - - $u_1, u_2 \dots, u_n$ final var vals