The Lovasz Local Lemma

1 Proof of the Lemma

Recall we defined a **dependency graph** G on events $(\mathcal{E}_i)$ as follows: Let

$$T_j \coloneqq \{\mathcal{E}_i \mid (\mathcal{E}_j, \mathcal{E}_i) \notin G\}.$$

Then $\mathcal{E}_j$ is mutually independent of T_j. In other words, non-edges guarantee independence and edges capture possible dependence. Notice that the complete graph is a dependency graph which gives trivial bounds; we want *sparse* dependency graphs.

Theorem 1. *[Lovasz Local Lemma].* Let G be a dependency graph on $(\mathcal{E}_i)$. Assume that there exists x_i with $i \in [n]$ and

$$\Pr[\mathcal{E}_i] \leq x_i \prod_{(i,j)\in E} (1 - x_j)$$

for all i. Then

$$\Pr[\bigcap_i \bar{\mathcal{E}}_i] \geq \prod_i (1 - x_i).$$

Conceptually, the x_i are upper bounds on the probability of $\mathcal{E}_i$ subject to some conditioning on limited independence.

Corollary 2. Let $(\mathcal{E}_i)$ be events with $\Pr[\mathcal{E}_i] \leq p$ for all i, and each $\mathcal{E}_i$ be mutually independent of all except at most d other events. If

$$p \leq \frac{1}{e(d+1)}$$

then $\Pr[\bigcap_i \bar{\mathcal{E}}_i] > 0$. In contrast, if we were using the union bound, we would need $p \leq {}^1\!/_n$. So we get a factor d instead of a factor n.

Proof. (of the corollary). The $(\mathcal{E}_i)$ are events with $\Pr[\mathcal{E}_i] \leq p$. Set $x_i = \frac{1}{d+1}$ for all i. Then

$$\begin{aligned}
x_i \prod_{(i,j)\in E} (1 - x_j) &= \frac{1}{d+1} \prod_{(i,j)\in E} (1 - \frac{1}{d+1}) \\
&\geq \frac{1}{d+1} (1 - \frac{1}{d+1})^d \\
&\geq \frac{1}{e(d+1)} \\
&\geq p \\
&\geq \Pr[\mathcal{E}_i].
\end{aligned}$$

So the x_i satisfy the hypothesis of the Lovasz Local Lemma, and we get

$$\Pr[\bigcap_i \bar{\mathcal{E}}_i] \geq \prod (1 - \frac{1}{d+1}) > 0.$$

$\square$

Proof. (of the LLL). We are interested in

$$\begin{aligned}
\Pr[\bigcap_i \bar{\mathcal{E}}_i] &= \Pr[\bar{\mathcal{E}}_1] \cdot \Pr[\bar{\mathcal{E}}_2 \mid \bar{\mathcal{E}}_1] \cdot \Pr[\bar{\mathcal{E}}_2 \mid \bar{\mathcal{E}}_1 \cap \bar{\mathcal{E}}_2] \cdot \dots \cdot \Pr[\bar{\mathcal{E}}_n \mid \bigcap_{i<n} \bar{\mathcal{E}}_i] \\
&= \prod_{j=1}^{n} (1 - \Pr[\mathcal{E}_j \mid \bigcap_{i<j} \bar{\mathcal{E}}_i]).
\end{aligned}$$

We will show that $\Pr[\mathcal{E}_j \mid \bigcap_{i \in S} \bar{\mathcal{E}}_i] \leq x_j$ for all j and all sets S of indices. Then, applying this with $S = \{i : i < j\}$ we obtain

$$\Pr[\bigcap_j \bar{\mathcal{E}}_j] \geq \prod_{j=1}^{n} (1 - x_j).$$

We will prove this (general) inequality by induction on $|S|$. The base case is $|S| = 0$. Here, we have $\Pr[\mathcal{E}_j] \leq x_j \prod_{(j,i) \in E} (1 - x_i) \leq x_j$.

For the induction step, fix some j and write $S = S_1 \cup S_2$ where S_1 are the neighbors of j in G and S_2 are the non-neighbors. Thus

$$\begin{aligned}
\Pr[\mathcal{E}_j \mid \bigcap_{i \in S} \bar{\mathcal{E}}_i] &= \frac{\Pr[\mathcal{E}_j \cap \bigcap_{i \in S} \bar{\mathcal{E}}_i]}{\Pr[\bigcap_{i \in S} \bar{\mathcal{E}}_i]} \\
&= \frac{\Pr[\mathcal{E}_j \cap \bigcap_{i \in S_1} \bar{\mathcal{E}}_i \cap \bigcap_{i \in S_2} \bar{\mathcal{E}}_i]}{\Pr[\bigcap_{i \in S_1} \bar{\mathcal{E}}_i \cap \bigcap_{i \in S_2} \bar{\mathcal{E}}_i]} \\
&= \frac{\Pr[\mathcal{E}_j \cap \bigcap_{i \in S_1} \bar{\mathcal{E}}_i \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i] \cdot \Pr[\bigcap_{i \in S_2} \bar{\mathcal{E}}_i]}{\Pr[\bigcap_{i \in S_1} \bar{\mathcal{E}}_i \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i] \cdot \Pr[\bigcap_{i \in S_2} \bar{\mathcal{E}}_i]} \\
&= \frac{\Pr[\mathcal{E}_j \cap \bigcap_{i \in S_1} \bar{\mathcal{E}}_i \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i]}{\Pr[\bigcap_{i \in S_1} \bar{\mathcal{E}}_i \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i]}. \qquad (1)
\end{aligned}$$

Bounding the numerator,

$$\Pr[\mathcal{E}_j \cap \bigcap_{i \in S_1} \bar{\mathcal{E}}_i \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i] \leq \Pr[\mathcal{E}_j \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i] = \Pr[\mathcal{E}_j]$$

by independence of $\mathcal{E}_j$ from non-neighbors. And

$$\Pr[\mathcal{E}_j] \leq x_j \prod_{i:(j,i) \in E} (1 - x_i).$$

If $S_1 = \emptyset$ then the denominator is equal to 1. Otherwise, write $S_1 = \{k_1, k_2, \dots, k_r\}$ for $r > 0$. Then

$$\Pr[\bigcap_{i \in S_1} \bar{\mathcal{E}}_i \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i] = \prod_{\ell=1}^{r} \Pr[\bar{\mathcal{E}_{k_\ell}} \mid \bigcap_{i \in S_2} \bar{\mathcal{E}}_i \cap \bigcap_{p=1}^{\ell-1} \bar{\mathcal{E}_{k_p}}].$$

Applying the induction hypothesis to $S_2 \cup \{k_1, \dots, k_{\ell-1}\}$, whose size is at most $|S| - 1$, this is at least

$$\prod_{\ell=1}^{r} (1 - x_{k_\ell}).$$

Because all k_ℓ are neighbors of j, this is at least

$$\prod_{i:(j,i) \in E} (1 - x_i).$$

Thus, fraction (1) is at most x_j. $\square$

2 Application to k-SAT

Corollary 3. *(of the LLL).* Let Φ be a k-SAT formula such that each variable occurs in fewer than $\frac{2^k}{ek} = \Theta(\frac{2^k}{k})$ clauses. Then, Φ is satisfiable.

Proof. Set each x_i to true *i.i.d.* with probability $1/2$. Each clause C_j is satisfied with probability $1 - 2^{-k}$. Let $\mathcal{E}_j$ be the bad event that C_j is not satisfied. So $\Pr[\mathcal{E}_j] \leq 2^{-k}$.

Note $\mathcal{E}_j$ is mutually independent of all $\mathcal{E}_i$ except possibly those with which C_j shares one or more variables. And, C_j shares a variable with fewer than $k(\frac{2^k}{ek} - 1)$ other clauses. This is at most $\frac{2^k}{e} - 1$. So $d + 1 < \frac{2^k}{e}$ in the dependency graph, and $p = 2^{-k} \leq \frac{1}{e(d+1)}$ as needed.

Thus, $\Pr[\bigcap \bar{\mathcal{E}}_i] > 0$ by the corollary of the LLL. $\square$

3 Constructive LLL (Moser-Tardos)

3.1 Motivation

Note that the corollary in the previous section is entirely non-constructive – it doesn't tell you how to *find* the assignment efficiently. Beck's algorithm does, but it needs stronger assumptions. And, we would like a constructive version of the LLL beyond just k-SAT.

To make this meaningful, we need to find a point ω in the sample space Ω such that $\omega \in \bigcap \bar{\mathcal{E}}_i$. To state this cleanly, we need some assumptions about the structure of Ω and $\mathcal{E}_j$.

Assume there is a finite set $\mathcal{P} = \{P_i\}$ of **independent** random variables (not necessarily binary), and for each i there is an efficient procedure for sampling P_i. We are explicitly given a bipartite graph between events $\mathcal{E}_j$ and variables P_i such that $\mathcal{E}_j$ is mutually independent of all variables it does not have an edge to. Write $L(\mathcal{E}_j)$ for the P_i that $\mathcal{E}_j$ has an edge to. For the dependency graph, $(\mathcal{E}_i, \mathcal{E}_j)$ exists only if $L(\mathcal{E}_i) \cap L(\mathcal{E}_j) \neq \emptyset$.

The Moser-Tardos Algorithm is quite simple: For each $P \in \mathcal{P}$, let y_P be a random draw of P (independent of all $y_{P'}$). While there is some $\mathcal{E}_j$ that is true under $\boldsymbol{y}$, re-draw all $P \in L(\mathcal{E}_j)$.

Theorem 4. *(Moser-Tardos 2009).* Assume that there exists x_i with

$$\Pr[\mathcal{E}_i] \leq x_i \prod_{(i,j) \in E} (1 - x_j)$$

for all i. Then this algorithm finds an assignment $\boldsymbol{y}$ satisfying $\bigcap \bar{\mathcal{E}}_j$ and uses at most

$$\sum_j \frac{x_j}{1 - x_j}$$

iterations in expectation. More specifically, each $\mathcal{E}_j$ is resampled at most $\frac{x_j}{1-x_j}$ times in expectation.

Note 5. The convergence is guaranteed by the non-constructive LLL because with some probability we keep guessing correctly; the interesting part is the fast convergence.

To gain some more intuition, consider the case where the dependency graph is empty. For event $\mathcal{E}_j$, the number of resamples needed is geometrically distributed with parameter $\Pr[\bar{\mathcal{E}}_j]$. So the expectation is

$$\begin{aligned} \frac{1}{\Pr[\bar{\mathcal{E}}_j]} - 1 &= \frac{1}{1 - \Pr[\mathcal{E}_j]} - \frac{1 - \Pr[\mathcal{E}_j]}{1 - \Pr[\mathcal{E}_j]} \\ &= \frac{\Pr[\mathcal{E}_j]}{1 - \Pr[\mathcal{E}_j]}. \end{aligned}$$

In the empty graph we can set $x_j = \Pr[\mathcal{E}_j]$, so the total number of resamples in expectation is at most $\frac{x_j}{1-x_j}$.

3.2 Execution Logs and Witness Trees

The **execution log** C has $C(t)$ the event that $\mathcal{E}_j$ is resampled in step t. Resampling was necessary because $\mathcal{E}_j$ was true under $\boldsymbol{y}_t$. In the independent case, this was because the previous time we sampled $L(\mathcal{E}_j)$, we made $\mathcal{E}_j$ true). In the general case this is more subtle because resampling one or more $\mathcal{E}_i$ with $(j, i) \in E$ may together have made $\mathcal{E}_j$ true.

We want a structure keeping track of the "blame" for resampling. We define a **witness tree** T_t for time t as follows: The root of T_t is labeled $\lambda(r) = \mathcal{E}_j$. We iterate through $t' = t-1, t-2, \ldots$ and consider the event $\mathcal{E}_i = C(t')$ that happened at t' in C.

In the first case, there is no edge $(i', i) \in E$ for any $\mathcal{E}_{i'}$ that is (so far) a label in T_t. Then, $\mathcal{E}_i$ cannot have caused any of the resampling in T_t, so we skip $\mathcal{E}_i$.

In the second case, there is a node $v \in T_t$ with label $\lambda(v) = \mathcal{E}_{i'}$ such that $(i', i) \in E$. Choose the deepest such v (breaking ties arbitrarily), create a new node u with $\lambda(u) = \mathcal{E}_i$, and make it a child of v.