Notes on the Lovasz Local Lemma

Constructive and Nonconstructive Methods in Combinatorics and TCS
U. Chicago, Spring 2018
Instructor/scribe: Andrew Drucker

The Lovasz Local Lemma (Erdös, Lovasz ‘74) is a powerful tool of probabilistic combinatorics.

**Theorem 1** (LLL, general form). Let \( \mathcal{A} \) be a finite set of events over a shared probability space \( \Omega \). Let \( \Gamma(A) \) be a subset of \( \mathcal{A} \) such that \( A \) is independent from \( \mathcal{A} - \{ A \cup \Gamma(A) \} \).

Suppose there is an assignment of reals \( x : \mathcal{A} \to (0, 1) \) such that, for all \( A \),

\[
\Pr[A] \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)) .
\]

Then

\[
\Pr \left[ \bigcup_{A \in \mathcal{A}} A \right] < \prod_{A \in \mathcal{A}} (1 - x(A)) < 1 .
\]

The sets \( \Gamma(A) \) may be regarded as arising from a “dependency digraph” with vertex set \( \mathcal{A} \).

Our first proof follows the original one, and the description in Alon-Spencer’s book.

**Proof.** Let \( \mathcal{A} = A_1, \ldots, A_n \). We identify \( \Gamma(i) := \Gamma(A_i) \) with a subset of \( [n] \), and use \( x_i := x(A_i) \). Central to the proof is the following **Claim:** For any \( S \subset [n] \) and \( i \notin S \),

\[
\Pr \left[ A_i \bigcap_{j \in S} \overline{A_j} \right] \leq x_i .
\]
This is proved by induction on \( s = |S| \). Base case: \( s = 0 \), easy. Now assume true for \( s' < s \) for some \( s > 0 \). Let \( S_1 := \Gamma(A_i) \cap S \), and \( S_2 := S - S_1 \). Then,

\[
\Pr \left[ A_i \left| \bigwedge_{j \in S} \overline{A_j} \right. \right] = \frac{\Pr \left[ A_i \land \bigwedge_{j \in S_1} \overline{A_j} \land \bigwedge_{\ell \in S_2} \overline{A_\ell} \right]}{\Pr \left[ \bigwedge_{j \in S_1} \overline{A_j} \land \bigwedge_{\ell \in S_2} \overline{A_\ell} \right]} \leq \frac{\Pr \left[ A_i \left| \bigwedge_{\ell \in S_2} \overline{A_\ell} \right. \right]}{\Pr \left[ \bigwedge_{j \in S_1} \overline{A_j} \right]} \tag{1}
\]

(we used a general conditional probability fact: \( \Pr[A|B, C] = \Pr[A, B|C]/\Pr[B|C] \)).

As \( A_i \) is independent of \( \{A_\ell : \ell \in S_2\} \), the final numerator above is equal to \( \Pr[A_i] \leq x_i \prod_{j \in \Gamma(i)} (1 - x_j) \).

As for the denominator, we use the induction hypothesis. Say \( S_1 = \{j_1, \ldots, j_r\} \).

If \( r = 0 \) then we are done by our previous work, so assume not. We have that

\[
\Pr \left[ \bigwedge_{j \in S_1} \overline{A_j} \right]
\]

is the product of conditional probabilities

\[
\Pr \left[ A_{j_q} \bigg| \bigwedge_{j_1 \in S_1} A_{j_1} \land \cdots \land A_{j_{q-1}} \land \bigwedge_{\ell \in S_2} \overline{A_\ell} \right]
\]

each of which is, by inductive hypothesis, at least \( 1 - x_{j_q} \).

Thus the denominator in Eq. (1) is at least \( \prod_{j \in S_1} (1 - x_j) \). We conclude that

\[
\Pr \left[ A_i \left| \bigwedge_{j \in S} \overline{A_j} \right. \right] \leq \frac{x_i \prod_{j \in \Gamma(i)} (1 - x_j)}{\prod_{j \in S_1} (1 - x_j)} = x_i .
\]

This extends the induction. The Theorem follows easily:

\[
\Pr \left[ \bigwedge_{j \in [n]} \overline{A_j} \right] = \prod_{j \in [n]} \Pr \left[ A_j \left| \bigwedge_{i < j} \overline{A_i} \right. \right] \geq \prod_{j} (1 - x_j) .
\]

\( \Box \)

**Question:** is there any hope of a direct “translation” of the above, classic proof into a constructive process to find an element of \( \bigwedge_{j \in [n]} \overline{A_j} \)?

A very convenient and more easily-remembered special case of LLL is given below.
**Theorem 2** (LLL, “low-degree” form). Let $\mathcal{A}$ be a finite set of events over a shared probability space $\Omega$. Let $\Gamma(A)$ be a subset of $\mathcal{A}$ such that $A$ is independent from $A - \{A \cup \Gamma(A)\}$.

Suppose that for all $A$ we have

$$|\Gamma(A)| \leq d , \quad \Pr[A] \leq p$$

where

$$p(d+1)e \leq 1.$$ 

Then

$$\Pr\left[\bigcup_{A \in \mathcal{A}} A\right] < 1.$$ 

**Proof.** To apply Theorem 1, let $x(A) := 1/(d+1)$, and note

$$\Pr[A] \leq p \leq \frac{1}{(d+1)e} \leq \frac{1}{d+1} \cdot \left(1 - \frac{1}{d+1}\right)^d .$$

$\square$

We will sometimes refer to the assumptions of Theorem 2 as the “low-degree setting” (which is not standard lingo; this form is sometimes called the “symmetric” Local Lemma).

The following is nearly as fully-general as Theorem 1, but puts it in a more “hands-on” form that provides a starting point for attempts at constructive/algorithmic proofs.

**Theorem 3** (LLL, “syntactic independence” form). The special case of Theorem 1, where the underlying $\Omega$ is a product space over independent random variables $Y_1, \ldots, Y_t$, each with finite support; and each $A$ is determined by a subset of the $Y_i$s, described by index set $R_A \subseteq [t]$, such that

$$A' \notin \Gamma(A) \implies R_{A'} \cap R_A = \emptyset ,$$

which guarantees the independence property.

It is this form of LLL for which Moser and Tardos gave a beautiful constructive proof, described below. But first, let us mention some examples of LLL’s many applications.
Example: Say that a hypergraph $G = (V,E)$ has Property B if its vertices can be colored avoiding monochromatic hyperedges. The low-degree LLL implies that any $k$-uniform $G$ with each edge intersecting at most $d$ other edges has Property B if
\[ e(d + 1) \leq 2^{k-1} . \]

Example: Let $F(x) = \bigwedge_j C_j(x)$ be an Exact-$k$-CNF formula\(^1\) where each variable $x_a$ appears in at most $t$ clauses. Consider a random assignment, and let $A_j$ be the event that $C_j(x) = 0$. We have $\Pr[A_j] = 2^{-k}$ by the exactness property. Also, $A_j$ is syntactically independent from all but $d = k(t - 1)$ other $A_\ell$, so as long as
\[ p(d + 1)e = 2^{-k}(k(t - 1) + 1)e \leq 1 \]
the formula $F$ is satisfiable by the low-degree form of LLL. Thus, we can “handle” up to $t \approx 2^k/(ke)$ occurrences of each variable and get satisfiability. Examples show this is approximately tight (ref. needed).

Example: let $S \subset \mathbb{R}$ be a finite set of $m > 1$ real numbers. If
\[ e(m(m - 1) + 1)k(1 - 1/k)^m \leq 1 \]
then there is a $k$-coloring of $\mathbb{R}$ such that each translate $S + x$ is multicolored (contains all colors). For this it suffices that
\[ m > (3 + \varepsilon)k \log k \]
for large enough $k$.

Proof idea: first prove the existence of a coloring valid for all $x \in X$, where $X$ is some fixed finite subset $X$ of the reals. The main idea: the event $A_x = [S + x$ not multicolored] is dependent only upon those of translates $S + x'$ that intersect $S + x$, of which there are are at most $m(m - 1)$. Then extend to the whole shebang, $X = \mathbb{R}$, by a “compactness” argument. See Alon-Spencer for more details.

Example: LLL yields the best known lower bounds for the “off-diagonal” Ramsey numbers $R(k, 4) > k^{5/2 - o(1)}$ on number of vertices needed to yield a $k$-clique or size-4 independent set. See Alon-Spencer for more discussion (not proof) and references.

\(^1\)(i.e., each clause is an OR of exactly $k$ literals, with no variable appearing twice in a clause, in either valence)
The Moser-Tardos constructive proof

Algorithm MT:

- sample each \( Y_1, \ldots, Y_t \) according to its own distribution.
- **while** some \( A_i \) holds:
  - arbitrarily choose any such \( A_i \), and resample all its domain variables \( Y_j \) according to their distribution.

**Theorem 4.** Under the Syntactic LLL hypotheses, the above Algorithm halts with a satisfying assignment after a number of resampling steps that, in expectation, is at most

\[
\sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}.
\]

In fact, each \( A \) is resampled at most \( x(A)/(1 - x(A)) \) times in expectation (yielding the above bound).

Note, this will yield \( O(n) \) expected resampling steps in many cases.

The algorithm parallelizes easily: take a maximal syntactically-independent subset of violated events, and resample all of them at once. Correctness is easy (it behaves identically to a particular implementation of the original algorithm). Parallel time complexity can be bounded under stronger assumptions on the weights \( x(A) \), see Moser-Tardos.

**Definition.** A labeled, rooted “Blame Tree” for the \( t \)th resample step, denoted \( T_t \) with labeling

\[
\sigma : V(T_t) \rightarrow \mathcal{A},
\]

is constructed as follows. The root is labeled with \( A_i \), the “bad” event being resampled at this \( t \)th step. Now work backward thru the transcript of bad events resampled at all previous steps \( t - 1, t - 2, \ldots, 1 \). For each \( A_j \) encountered, if \( A_j \) has shared neighborhood with any label \( A_\ell \) in our tree so far, add it as label to a new child node on the deepest such node. Break ties arbitrarily.

This defines \( T = T_t \). Note that \( T \) describes a collection of resampling events, but the structure of \( T \) only imposes a partial order on those events: namely if \( v \) is at greater distance from the root than \( v' \) (but not necessarily along the same path in \( T \)), this implies that the resampling event associated with \( v \) happened earlier.
Observe: a layer of our tree $T$ always consists of variable-disjoint (independent) event labels. In particular, all children of a node receive distinct labels (though possibly their own label); we say such a labeled tree is proper. Also, they’re dependency-respecting: a child is syntactically dependent on its parent. These properties will be important in part because they restrict the number of possible blame trees of a given size that could possibly arise.

An easy observation is that $(T, \sigma)$ can arise as blame tree at most once, $T = T_t$ for one value of $t$, in any execution of MT. This is because the $r^{th}$ resampling step for $A_i$ has an associated blame tree with $A_i$ as root label and $r - 1$ other appearances of label $A_i$.

The following trickier claim is the crux of the analysis of the MT algorithm.

Claim 1. Fix a proper labeled rooted tree $(T, \sigma)$. The probability that $T$ appears as blame tree, $T = T_t$ for any step $t$ at all is at most

$$\prod_{v \in V(T)} \Pr[\sigma(v)].$$

Proof. Let each underlying random variable $Y_j$ be sampled and resampled in MT by a stream

$\hat{Y}_j^1, \hat{Y}_j^2, \ldots$

of outcomes. So e.g. $\hat{Y}_j^1$ is the first outcome of $Y_j$; then $\hat{Y}_j^2$ is seen in the first resampling event for any event $A_i$ for which $Y_j \in R_i$; etc.

For each $v \in T$, with associated label $\sigma(v) = A_i$, and for each $j \in R_i$, let

$$\text{ind}(j, v) := k + 1,$$

where $k$ is the number of vertices $v'$ at greater depth in $T$ than $v$ whose labeling event $\sigma(v') = A_i$ satisfies

$$j \in R_\ell.$$

Now our key assertion is that, if $T$ occurs as a blame tree for step during the execution of MT, then the value of $Y_j$ right before the resampling of $A_i$ associated with blame-tree vertex $v$, is precisely the particular outcome $Y_j^{\text{ind}(j, v)}$. This follows from the definition of the MT algorithm and of blame trees. Thus, it must be the case that the particular values

$$\left\{ Y_j^{\text{ind}(j, v)} \right\}_{j \in R_i}$$
cause $A_i$ to occur at the step corresponding to $v$—for, that is needed to trigger the resampling.

What we have done is to associate with each node $v$ a collection of random variables (according to an allocation determined entirely by the structure of $T$ itself); each such collection is separate and independent from those for every other $v'$. Also, each $v$’s associated collection distributed as $Y_1 \times \ldots \times Y_N$—or more precisely, as the restriction of these variables to the portion relevant to $A_i = \sigma(v)$. And for $T$ to be applicable as a blame tree in the execution determined by $\hat{Y}_1, \hat{Y}_2, \ldots \ i = 1, 2, \ldots, t$,

each of the indicated collections, ranging over $v \in T$, must (independently!) cause the associated bad event $A_i = \sigma(v)$ to occur. Claim 1 follows.

The probability bound in Claim 1 is, in the low-degree setting, at most $p |V(T)|$, where $p$ is the uniform upper bound on the probabilities. More generally, we have:

**Observation 1.** Under the assumptions of Theorem 1,

$$\prod_{v \in V(T)} \Pr[\sigma(v)] \leq \prod_{v \in V(T)} x'(\sigma(v)),$$

where

$$x'(A_i) = x(i) \prod_{j \in \Gamma(i)} (1 - x(j)).$$

This is direct from our assumptions. Note, the product above will include multiple terms $\Pr[A_j]$ if $A_j$ appears multiple times under the labeling of $T$ (as it well may).

It remains to use the bound in a clever way that allows us to sum over all possible $T$. To do this, we describe a classical branching process generating a random tree, whose outcome probabilities are related to our probability bound above. This process is going to spit out labeled trees that “look like” blame trees, but are not directly associated with any particular execution of MT.

**Galton-Watson processes and random trees.** Fix $A_i$. Define the GW process to create a proper, labeled, $A_i$-rooted tree as follows: if a node is generated labeled $A_j$, it independently spawns a child with label $A_\ell$ with probability $x_\ell$ for each $\ell \in \Gamma^+(A_j)$. The end result is a tree $T$, possibly infinite.
Claim 2. The probability $GW$ with seed node label $A_i$ outputs a given finite labeled tree $(T, \sigma)$ as before is exactly

$$\frac{1 - x(i)}{x(i)} \prod_{v \in V(T)} x'(\sigma(v)).$$

Combining this with previous results, and summing over $i$ and all finite $T$ (the probabilities sum to at most 1 for each fixed choice of $i$, since $GW$ can output just one tree!), we get

$$\sum_{T, \sigma \ i\text{-rooted}} \Pr[T \text{ appears as blame tree for } MT] \leq \frac{x(i)}{1 - x(i)},$$

from which Theorem 4 follows.

Proof (sketch). Tedious rearrangement using the definition of $GW$; or, one can prove for all possible $T$ rooted at $i$ using induction on the tree structure (adding one leaf at a time). \qed

In the low-degree setting, one can choose to avoid the Galton-Watson process entirely, instead using the following simple estimate:

Claim 3. In the low-degree setting (with dependency digraph of degree bound $d$), the number of proper labeled rooted, dependency-respecting trees $T$ on $s$ vertices is at most

$$n((d + 2)e)^s.$$

Proof. Encode such a tree by a binary string of length $(d + 1)s$ in the natural way: for each successive node (in a depth-first traversal, say) indicate which of its $d + 1$ possible child nodes is present. The number of 1s in this encoding string is at most $s - 1$. Thus for a fixed start vertex, we have a number of encoding strings that is at most

$$\left(\frac{(d + 1)s}{s - 1}\right)^s \leq \left(\frac{(d + 1)se}{s - 1}\right)^s \leq ((d + 2)e)^s.$$

Finally, we have $n$ possible root labels. \qed

Claim 4. In the low-degree setting, the expected number of resampling steps is at most the expected number of blame trees that are applicable at all, which is at most

$$\sum_s p^s \cdot n((d + 2)e)^s.$$

The proof is immediate if we sum over all possible size trees that could arise. If $p(d + 2)e < 1 - \varepsilon$, this is $O_{\varepsilon}(n)$. This doesn’t quite recover the original low-degree LLL parameters, but is reasonably close.
Bibliography
