23.1 The Algorithmic Lovász Local Lemma

In this lecture we prove an algorithmic version of one variant of the LLL, due to Moser and Tardos [MT10]. (A slightly more restricted version of the same result was proved a little earlier by Moser [Mos09], using a different but very elegant method based on entropy compression.) We will focus on the so-called “variable setting,” in which there is an underlying family of mutually independent random variables $Z_1, \ldots, Z_n$ (“variables”) on a common probability space. Each bad event $A_i$ is determined by a subset of the $Z_j$, which we will call $\text{vbl}(A_i)$. The dependency graph is a graph with vertices $A_i$ and an edge between pair $A_i, A_j$ if and only if $\text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset$. We let $D_i$ denote the neighborhood of $A_i$ in this graph. With abuse of terminology, we say that $A_i$ is “violated” by an assignment of values to the variables $Z_j$ if $A_i$ is true under this assignment. Our goal is to find an assignment of the variables so that none of the $A_i$ is violated. Note that, since the $Z_j$ are independent, $\Pr[A_i]$ is determined by a product distribution on the $Z_j$.

We have seen this set-up in the last lecture, e.g., in the example of $k$-SAT. There the variables $Z_j$ correspond to the boolean variables in a given $k$-SAT formula, and there is one $A_i$ for each clause which of course depends on the set of variables appearing in that clause. $A_i$ is violated by an assignment iff its clause is false under that assignment. Of course, any constraint satisfaction problem can be viewed in this way.

We will analyze the following simple randomized local search algorithm for this problem.

```
pick an independent random assignment to the $Z_j$
while there exists a violated $A_i$ do
    choose a violated $A_i$ according to some rule (e.g., pick the lowest indexed such $A_i$)
    resample the values of $Z_j \in \text{vbl}(A_i)$
return the values $\{Z_j\}$
```

Our goal is to prove the following algorithmic version of the Lovász Local Lemma, due to Moser and Tardos [MT10]:

**Theorem 23.1** In the above setting, if there exist real numbers $x_i \in (0, 1)$ such that

$$\Pr[A_i] \leq x_i \prod_{j \in D_i} (1 - x_j) \text{ for all } i,$$

then the above algorithm finds an assignment of the $\{Z_j\}$ that violates none of the $A_i$ in expected time at most $\sum_i \frac{x_i}{1 - x_i}$.

As we shall see, the running time bound (which is linear in $n$, the number of bad events or constraints) comes from the fact that the variable set of each $A_i$ is resampled at most $\frac{x_i}{1 - x_i}$ times in expectation.
Note that the condition in Theorem 23.1 is exactly that of the general LLL (Lemma 22.8 in the previous lecture); the only additional assumption is the underlying set of random variables \( Z_j \). Since this pioneering result, there has been a flood of interest in this topic, resulting in all known versions of the LLL now being algorithmic; more remarkably, this work has even inspired stronger forms of the LLL, and efficient local search algorithms that don’t correspond to any recognizable version of the LLL at all! For a representative selection of papers on this topic, see [AI16, AIS19, HSS11, HV15] as well as the original breakthrough [Mos09].

The rest of this lecture is devoted to proving Theorem 23.1. We first introduce two key notions we’ll use throughout.

**Definition 23.2** An execution of the algorithm is a sequence \( E := E(1), E(2), \ldots, E(t), \ldots \), where \( E(t) \in \{ A_i \} \) is the violated event chosen at step \( t \) of the algorithm. (The execution may be either finite, if the algorithm terminates, or infinite in length.)

**Definition 23.3** A witness tree is a rooted tree \( T \) whose nodes are labeled with events \( A_i \), such that if \( A_j \) is a child of \( A_i \), then \( A_j \in D^+ \), the dependency set of \( A_i \) augmented with \( A_i \) itself. We call \( T \) proper if the labels on all children of any node are distinct.

Given an execution \( E \), define a witness tree \( T(t) \) for each time step \( t \) of \( E \) as follows. The root is labeled with the event \( E(t) \). Then, for \( i = t - 1, t - 2, \ldots \), going back in time, attach a node labeled with the event \( E(i) \) as a child of the deepest (i.e., furthest from the root) node with label in \( D^+_{E(i)} \), breaking ties arbitrarily. If no such node exists, do not add a node for \( E(i) \) to the tree.

The intuition for this construction is that \( T(t) \) explains (or “witnesses”) the sequence of resamplings that led to the resampling of the event \( E(t) \) at the root.

We say that a witness tree \( T \) occurs in \( E \) if \( T = T(t) \) for some \( t \).

We begin with a key property of witness trees.

**Claim 23.4** Let \( T \) be a witness tree and \( E \) a random execution of the algorithm.

(i) If \( T \) occurs in \( E \) then \( T \) is proper.

(ii) \( \Pr[T \text{ occurs in } E] \leq \prod_{v \in V(T)} \Pr[A_{v}] \), where \( V(T) \) is the set of nodes of \( T \) and \( A_{v} \) denotes the event labeling node \( v \in V(T) \).

**Proof:** To see part (i), for any node \( u \in V(T) \), let depth\((u)\) denote the depth of \( u \) and time\((u)\) the time step in the execution \( E \) at which \( u \) was attached. If \( \text{time}(u) < \text{time}(v) \) and \( \text{vbl}(A_{[u]}) \cap \text{vbl}(A_{[v]}) \neq \emptyset \), then \( \text{depth}(u) > \text{depth}(v) \) since we must have attached \( u \) either to \( v \) or to another node with depth at least \( \text{depth}(v) \). Hence no two nodes at the same level of \( T \) can share variables, and in particular, \( T \) is proper.

For part (ii), define an evaluation of \( T \) as follows. In reverse bfs order, visit the nodes of \( T \) and resample their variables (independently of previous resamplings). Say that the evaluation succeeds if all events were violated by these resamplings. Obviously we have

\[
\Pr[\text{evaluation succeeds}] = \prod_{v \in V(T)} \Pr[A_{v}] .
\]

To complete the proof, we show that if \( T \) occurs in \( E \) then we can couple the evaluation of \( T \) with the execution \( E \) so that the evaluation succeeds. This will imply

\[
\Pr[T \text{ occurs in } E] \leq \Pr[\text{evaluation succeeds}] = \prod_{v \in V(T)} \Pr[A_{v}] ,
\] (23.1)
as required.

The coupling is defined as follows. We specify in advance, for each variable $Z_j$, an infinite sequence of independent random boolean values. Then, when either the algorithm or the evaluation needs to sample $Z_j$, it takes the next value in this sequence; thus the algorithm and the evaluation both take the same value for a given variable if it has been sampled the same number of times in both processes.

Now consider a time at which $Z_j$ is being resampled in the evaluation, at some node $v$. Note first that, as we saw in the proof of part (i) above, there are no other nodes at the same level as $v$ that sample $Z_j$. Hence, by the bfs ordering, the number of times $Z_j$ has been sampled prior to the resampling at $v$ is equal to the number of nodes of depth greater than depth($v$) that include $Z_j$ in their variable sets. Call this number $n_{j,v}$.

Next consider the corresponding resampling operation for event $A_{[v]}$ in the execution of the algorithm itself. By the construction of the tree, the number of times $Z_j$ has been sampled prior to the resampling of $A_{[v]}$ is $n_{j,v}+1$, since $Z_j$ was sampled initially and then at all the other times corresponding to nodes below depth($v$) in the tree.

Hence, when the evaluation process resamples at $v$, it will use the same values for each variable $Z_j$ as the algorithm has set immediately prior to its resampling of $A_{[v]}$. But this set of values causes $A_{[v]}$ to be violated since otherwise the algorithm would not select it for resampling. Hence we deduce that the evaluation succeeds. This verifies (23.1) and hence completes the proof of part (ii) of the claim.

Now for each event $A_i$, define the random variable $N_i$ to be the number of times event $A_i$ is resampled during the execution. Our goal is to compute the expectation $E[N_i]$. The sum of these expectations will be the expected running time of the algorithm.

Note that $N_i$ is the number of distinct proper witness trees occurring in a random execution $E$ that have root labeled $A_i$. (The trees are distinct as they each contain a different number of labels $A_i$.) By part (ii) of Claim 23.4, we have

$$E[N_i] = \sum_{T:\text{root}(T)=A_i} \Pr[T \text{ appears in } E] \leq \sum_{T:\text{root}(T)=A_i} \prod_{v \in V(T)} \Pr[A_{[v]}].$$

To compute this sum over probabilities, we will relate it to a seemingly unrelated random process, namely a multi-type Galton-Watson branching process.

In this process, we specify a root label $A_i$ and then, for each $A_j \in D_i^+$ independently, with probability $x_j$ we add as a child of the root a node labeled $A_j$. We then proceed to expand the tree by adding children to the children of the root according to the same rule, and so on. As with a standard Galton-Watson tree, this process may or may not die out eventually.

Claim 23.5 Let $T$ be a proper witness tree with root label $A_i$. The probability that the above Galton-Watson process yields tree $T$ is given by

$$p_T = \frac{1 - x_i}{x_i} \prod_{v \in V(T)} x'_{[v]},$$

where $x'_{[i]} = x_i \prod_{j \in D_i} (1 - x_j)$. 

Proof: For \( v \in V(T) \), let \( W_v \) be the subset of \( D^+_{A[v]} \) that do not occur as children of \( v \) in \( T \). Then we have

\[
\begin{align*}
    p_T &= \frac{1}{x_i} \prod_{v \in V(T)} \left( x[v] \prod_{u \in W_v} (1 - x[u]) \right) \\
    &= \frac{1 - x_i}{x_i} \prod_{v \in V(T)} \left( \frac{x[v]}{1 - x[v]} \prod_{u \in D^+_{A[v]}} (1 - x[u]) \right) \\
    &= \frac{1 - x_i}{x_i} \prod_{v \in V(T)} \left( x[v] \prod_{u \in D^+_{A[v]}} (1 - x[u]) \right) \\
    &= \frac{1 - x_i}{x_i} \prod_{v \in V(T)} x'[v].
\end{align*}
\]

Finally, we're in a position to bound \( E[N_i] \) and thus prove Theorem 23.1.

Proof of Theorem 23.1: We have, for any event \( A_i \),

\[
E[N_i] = \sum_{T: \text{root}(T) = A_i} \Pr[T \text{ appears in } E] \\
\leq \sum_{T: \text{root}(T) = A_i} \prod_{v \in V(T)} \Pr[A[v]] \\
\leq \sum_{T: \text{root}(T) = A_i} \prod_{v \in V(T)} x'[v] \\
= \frac{x_i}{1 - x_i} \sum_{T: \text{root}(T) = A_i} p_T \\
\leq \frac{x_i}{1 - x_i}.
\]

The second line here comes from Claim 23.4, the third line is by the LLL condition in the theorem, the fourth line is by Claim 23.5, and the last line follows from the fact that the trees \( T \) are distinct and \( \sum_T p_T \leq 1 \).

References


