CSE 525: Randomized Algorithms **Spring 2023**

Lecture 6: Algorithmic Lovász Local Lemma

Lecturer: Shayan Oveis Gharan *04/13/2023*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

Given underlying independent random variables $Z_1, \ldots, Z_m$ with product measure μ. The "bad events" $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are each determined by a certain subset of the random variables, which we de- note $\text{var}(\mathcal{A}_i) \subseteq [n]$. The dependency graph G has vertices $[n]$ and edges $(i,j) \in E(G)$ whenever $\text{var}(\mathcal{A}_i) \cap \text{var}(\mathcal{A}_j) \neq \emptyset$. Note that this is a valid choice of a dependency graph, since each event $\mathcal{A}_i$ is **independent** of any conditioning on the variables outside of $\text{var}(\mathcal{A}_i)$.

Given that the conditions of the Lovász Local Lemma hold, we want to find a realization of the random variables $Z_1, \ldots, Z_m$ such that no events $\mathcal{A}_i$ happen.

Moser Tardos's Algorithm

1. Sample $Z_1, Z_2, \ldots, Z_m$ from the distribution μ.
2. As long as any event A_i is satisfied by the current values of $Z_1, \ldots, Z_m$, choose the smallest such i and resample $\text{var}(\mathcal{A}_i)$: replace $(Z_a : a \in \text{var}(\mathcal{A}_i))$ by new independent samples.

It is clear that if the algorithm terminates, then we have found an assignment avoiding all events. The key is to analyze the expected number of resampling steps. By one resampling step, we mean the operation of resampling all the variables of an event.

Theorem 6.1 (Moser-Tardos [**MT10**])**.** *The expected number of resampling steps before termination of the algorithm is at most* $\sum_{i=1}^{n} \frac{x_i}{1-x_i}$*, provided that* $\mathbb{P}[A_i] \leq x_i \prod_{j \in \Gamma(i)}(1-x_j)$*, where as usual* $\Gamma(i)$ *are the set of neighbors of* i *in the dependency graph.*

In these notes we also use $\Gamma^+(i) := \Gamma(i) \cup \{i\}$ to denote the set of neighbors of i and including i itself.

Note that in applications, x_i's are usually small ($x_i \leq 1/2$), so this means the expected number of resampling operations is $O(n)$.

6.1 Execution Log and Stable Set Sequences

We define the execution log of the algorithm as the sequence of events that get resampled: $(\mathcal{A}_{i_1}^{(1)}, \mathcal{A}_{i_2}^{(2)}, \ldots)$, where $\mathcal{A}_i^{(t)}$ denotes the fact that the event $\mathcal{A}_i$ was resampled at time t. We want to prove that for all $i \in [n]$

$$\mathbb{E}[\text{the number of times } \mathcal{A}_i \text{ gets resampled}] \leq \frac{x_i}{1-x_i}.$$

Stable set sequences An important notion in the analysis will be that of stable set sequences. First, given the log, we define a directed graph R on vertices $\mathcal{A}_i^{(s)}$ as follows. For each pair of entries in the log, $\mathcal{A}_i^{(s)}$ and $\mathcal{A}_{i'}^{(s')}$, we add a directed edge $(\mathcal{A}_i^{(s)}, \mathcal{A}_{i'}^{(s')})$ to R if $s < s'$ and $(\mathcal{A}_i, \mathcal{A}_{i'}) \in E(G)$.

For a fixed entry $\mathcal{A}_i^{(t)}$ in the log, let us consider a subgraph $R(t) \subseteq R$, induced by the vertices that have a directed path to $\mathcal{A}_i^{(t)}$. We call $\mathcal{A}_i^{(t)}$ the **root** of $R(t)$. For each $\ell \geq 0$, we define a set of events:

$$I_\ell = \{j : \exists \mathcal{A}_j^{(s)} \in V(R(t)), \text{the longest path from } \mathcal{A}_j^{(s)} \text{ to the root } \mathcal{A}_i^{(t)} \text{ has length } \textbf{exactly } \ell\}.$$

Note that $I_0 = \mathcal{A}_j^{(s)}$. We have the following properties:

i) For every $\ell \geq 0$, I_ℓ is an independent set in G.

Proof: If $j, j' \in I$ and $(j, j') \in E(G)$, then there must be a directed edge $(\mathcal{A}_j^{(s)}, \mathcal{A}_{j'}^{(s')}) \in R(t)$ (we assume wlog that $s < s'$). This means that $\mathcal{A}_j^{(s)}$ has a path to the root through $\mathcal{A}_{j'}^{(s')}$ that has length 1 more than the longest path from $\mathcal{A}_{j'}^{(s')}$ to the root. This contradicts the fact that the longest paths from both nodes to the root have length exactly ℓ.

ii) For every $\ell \geq 0$, $I_{\ell+1} \subseteq \Gamma^+(I_\ell)$.

Proof: For every $j \in I_{\ell+1}$, there is a longest directed path from $\mathcal{A}_j^{(s)}$ to the root of length exactly $\ell+1$. So the next vertex on the path must have a longest path to the root of length ℓ. This vertex corresponds to an event $\mathcal{A}_{j'} \in I_\ell$ and by construction of the directed graph, we have that $j \in \Gamma^+(\mathcal{A}_{j'})$.

This motivates the following definition.

Definition 6.2. *A stable set sequence for G is a finite sequence of sets $I = (I_0, I_1, \ldots, I_r)$ such that for every $0 \leq \ell \leq r$, I_ℓ is an* **independent set** *in G and for every $0 \leq \ell < r$, $I_{\ell+1} \subseteq \Gamma^+(I_\ell)$.*

By the discussion above, every sequence $I = (I_0, I_1, \ldots, I_r)$ produced from a log of execution of the algorithm is a stable set sequence (note that it must be finite, since for a fixed root $\mathcal{A}_i^{(t)}$ the induced subgraph $R(t)$ is finite).

Definition 6.3. *A stable set sequence I is said to be a* **witness** *of a resampling $\mathcal{A}_i^{(t)}$ if it is produced from the log by the above process, starting from root $\mathcal{A}_i^{(t)}$. We say that I occurs in the execution log if there is t such that I is a witness of the resampling $\mathcal{A}_i^{(t)}$.*

Lemma 6.4. *For every stable set sequence $I = \{I_0, I_1, \ldots, I_r\}$,*

$$\mathbb{P}\left[I \text{ occurs in the log}\right] \leq \prod_{\ell=0}^{r} \prod_{i \in I_\ell} p_i$$

where $p_i = \mathbb{P}\left[\mathcal{A}_i\right]$.

Proof. We first modify the algorithm as follows (which does not change its behavior). We prepare an infinite table of samples to be used: For each Z_a, the a-th row of the table contains an infinite sequence $Z_{a_1}, Z_{a_2}, Z_{a_3}, \ldots$, each sampled independently according to the distribution of Z_a. The algorithm maintains a pointer $\pi(a)$ for each $a \in [m]$. We start with $\pi(a) = 1$ for each $a \in [m]$. The "current values" of Z_a are given by $Z_a^{\pi(a)}$. Whenever the algorithm "resamples" Z_a , we increment $\pi(a)$ by 1, which means moving on to the next sample. Clearly, this is equivalent to the original description of the algorithm.

We claim that if a certain stable set sequence I occurs in the execution log, then for each of its events we can determine a particular set of samples in the table that must satisfy the event. Given $I = (I_0, I_1, \ldots)$, we obtain the locations of these samples as follows: For every $a \in \text{var}(\mathcal{A}_j)$ where $j \in I_\ell$, let $n_{a,\ell}$ denote the

number of indices $\ell' \geq \ell$ such that $j' \in I_{\ell'}$ and $a \in \text{var}(\mathcal{A}_{j'})$. (Note that for each ℓ', at most one event in $I_{\ell'}$ can depend on Z_a, since $I_{\ell'}$ is an independent set.)

Then, we claim that $(Z_a^{n(a,\ell)} : a \in \text{var}(\mathcal{A}_j))$ are exactly the samples of Z that were checked by the algorithm to determine that $\mathcal{A}_j$ occurs, before the resampling that makes $\mathcal{A}_j$ a member of I_ℓ. This is because the only times when $\pi(a)$ is incremented is when we resample an event depending on Z_a. If $\mathcal{A}_j \in I_\ell$ and this is due to a resampling at time s, then any event resampled before time s that also depends on Z_a will be part of the directed graph $R(s)$ and hence also part of the stable set sequence. These are the only times when the pointer $\pi(a)$ is incremented prior to the resampling $\mathcal{A}_j^{(s)}$ and hence the value of $\pi(a)$ just before this resampling is exactly $n(a, \ell)$.

Now we know that in order for $I = (I_0, I_1, \ldots, I_r)$ to occur, it must be the case that for each $0 \leq \ell \leq r$ and for each event $\mathcal{A}_j \in I_\ell$, the samples $(Z_a^{n(a,\ell)} : a \in \text{var}(\mathcal{A}_j))$ satisfy the event $\mathcal{A}_j$. (Otherwise the algorithm would not choose to resample it.) This happens with probability $\mathbb{P}[\mathcal{A}_j]$. Most importantly, notice that the samples $Z_a^{n(a,\ell)}$ for different values of ℓ are distinct; this follows directly from the definition of $n(a, \ell)$. By the independence of the samples in the table, the probability that for each $\mathcal{A}_j \in I_\ell$, $0 \leq \ell \leq r$, the samples $(Z_a^{n(a,l)} : a \in \text{var}(A_j))$ satisfy $\mathcal{A}_j$, is $\prod_{s=0}^{r} \prod_{j \in I_s} p_j$. □

6.2 Summing Up

Now for each event $\mathcal{A}_i$, define the random variable N_i to be the number of times event $\mathcal{A}_i$ is resampled during the execution. Our goal is to compute the expectation $\mathbb{E}[N_i]$. The sum of these expectations will be the expected running time of the algorithm. Note that N_i is the number of distinct stable set sequences with $I_0 = \{\mathcal{A}_i\}$ in a execution of the algorithm (We remark that although the stable sets are distinct, each one is properly included in the later ones.)

$$\mathbb{E}[N_i] = \sum_{I=(I_0,\ldots),I_0=\{\mathcal{A}_i\}} \mathbb{P}[I \text{ occurs in the log}] \leq \sum_{I=(I_0,\ldots),I_0=\mathcal{A}_i} \prod_{\ell=0}^{r} p^{I_\ell}$$

where for simplicity we write $p^{I_\ell} = \prod_{i \in I_\ell} p_i$. We need to show that $\mathbb{E}[N_i] \leq \frac{x_i}{1-x_i}$.

We prove a more general fact:

Lemma 6.5. *For any $t \geq 1$, and any non-empty independent set $J \in G$, we have*

$$\sum_{I=(I_0,I_1,\ldots,I_t):I_0=J} \prod_{s=0}^{t} p^{I_s} \leq \prod_{i \in J} \frac{x_i}{1-x_i}$$

Proof. We prove by induction. We leave the base case as an exercise.

Going over all possibilities for I_1 based on the definition of a stable set sequence we can write,

$$\begin{aligned}
\sum_{I=(I_0,I_1,\ldots,I_t),I_0=J}\prod_{s=0}^{t}p^{I_s} &= p^J \sum_{\substack{K\subseteq\Gamma^+(J),\\ K \text{ indep in } G}} \sum_{I=(I_0,\ldots,I_{t-1}),I_0=k}\prod_{s=0}^{t-1}p^{I_s} \\
&\underset{\text{by IH}}{\leq} p^J \sum_{\substack{K\subseteq\Gamma^+(J),\\ K \text{ indep in } G}} \prod_{k\in K}\frac{x_k}{1-x_k} \\
&\underset{\text{Assumption of Theorem 6.1}}{\leq} \prod_{i\in J}x_i \prod_{j\in\Gamma_i}(1-x_j) \sum_{\substack{K\subseteq\Gamma^+(J),\\ K \text{ indep in } G}} \prod_{k\in K}\frac{x_k}{1-x_k} \\
&= \prod_{i\in J}\frac{x_i}{1-x_i}\prod_{j\in\Gamma_i^+}(1-x_j) \sum_{\substack{K\subseteq\Gamma^+(J),\\ K \text{ indep in } G}} \prod_{k\in K}\frac{x_k}{1-x_k} \\
&\leq \prod_{i\in J}\frac{x_i}{1-x_i} \sum_{\substack{K\subseteq\Gamma^+(J),\\ K \text{ indep in } G}} \prod_{k\in K}x_k \prod_{k\in\Gamma^+(J)\setminus K}(1-x_k) \\
&\underset{\text{Drop the constraint that } K \text{ is indep}}{\leq} \prod_{i\in J}\frac{x_i}{1-x_i} \sum_{K\subseteq\Gamma^+(J)} x^K(1-x)^{\Gamma^+(J)\setminus K} = \prod_{i\in J}\frac{x_i}{1-x_i}.
\end{aligned}$$

To see the last identity, observe that

$$1 = \prod_{k\in\Gamma^+(J)}(x_k+1-x_k) = \sum_{K\subseteq\Gamma^+(J)} x^K(1-x)^{\Gamma^+(J)\setminus K}$$

This completes the proof of the lemma. □

Note that we can take a limit $t\to\infty$ to show that indeed $\mathbb{E}[N_i]\leq\frac{x_i}{1-x_i}$.

Finally, to prove Theorem 6.1, it is enough to use linearity of expectations together with the above lemma to show that

$$\sum_i \mathbb{E}[N_i] \leq \sum_i \frac{x_i}{1-x_i}$$

as desired.