CS 6550: Randomized Algorithms **Spring 2019**

Lecture 16: Algorithmic Lovász Local Lemma

March 5, 2019

Lecturer: Eric Vigoda *Scribes: Congshi Zou & Feng Feng*

16.1 Algorithmic Lovász Local Lemma

Definition 16.1 *Let $\{x_1, x_2, \ldots, x_m\}$ be a finite set of mutually independent random variables. Let $\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ be a finite set of events determined by these variables. For event $\mathcal{B}_i$,*

$$vbl(\mathcal{B}_i) := \{x_j : \mathcal{B}_i \text{ depends on } x_j\}$$

$$D_i := \{\mathcal{B}_j : \mathcal{B}_j \in \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\} \backslash \{\mathcal{B}_i\} \ \& \ vbl(\mathcal{B}_j) \cap vbl(\mathcal{B}_j) \neq \emptyset\}$$

$$D_i^+ := D_i \cup \{\mathcal{B}_i\}$$

If $\mathcal{B}_i$ occurs, we say $\mathcal{B}_i$ is violated.

We will analyze the following Moser-Tardos Algorithm.

Algorithm 1: Moser-Tardos Algorithm

for $x_j \in \{x_1, x_2, \ldots, x_m\}$ **do**
  Choose x_j from {0,1} uniformly at random;
while $\exists \mathcal{B}_i \in \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ *is violated* **do**
  Pick an arbitrary violated event $\mathcal{B}_i$;
  for $x_j \in vbl\{\mathcal{B}_i\}$ **do**
    Choose x_j randomly from {0,1} ;

Our goal is to prove the following Algorithmic Lovász Local Lemma related to Moser-Tardos Algorithm.

Theorem 16.2 *Let $\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ be a finite set of events. If there exists $\{\beta_1, \beta_2, \ldots, \beta_n\} \in [0, 1)$, such that,*

$$\Pr(\mathcal{B}_i) \leq \beta_i \prod_{j \in D_i} (1 - \beta_j) \quad \forall i$$

the Moser-Tardos Algorithm terminates in expected time at most $\sum_{i=1}^{n} \frac{\beta_i}{1-\beta_i}$.

16.2 Witness trees

Definition 16.3 *An execution of Moser-Tardos Algorithm is a sequence $E := E(1), E(2), \ldots, E(T)$, where $E(t)$ is the violated event $\mathcal{B}_i$ resampled at step t of the algorithm. (The execution may be either finite, if the algorithm terminates, or infinite in length.) For convenience, let $D^+(E[t])$ denote D_i^+.*

Definition 16.4 *For a tree T, let $V(T)$ denote the set of its vertices. For $v \in V(T)$, let $d(v)$ denote the depth of v (distance from v to the root r of tree T). For example, $d(r) = 0$, and its children have depth 1.*

Given an execution E, now we define a witness tree $T(t)$ for each step t of E as follows.

Algorithm 2: Witness Tree

```
Label the root of tree T(t) with event E(t);
for t' ← t − 1 to 1 do
    if ∃ a vertex in the current tree with label E[i] such that E[t'] ∈ D⁺(E[i]) then
        Choose among all such vertices the one which has the maximum depth, and break ties
          arbitrarily;
        Add E(t') as a child of the vertex;
    else
        Do not add a vertex for E[t'] to tree T(t);
```

Claim 16.5 *In a witness tree, the labels on all children of any vertex are distinct and independent. Besides, at each depth, an event $\mathcal{B}_i$ occurs at most once and all labels are independent.*

Proof: When adding $\mathcal{B}_i$, if it already occurs at depth d, then we can add $\mathcal{B}_i$ as a child of that vertex at depth d or a vertex with higher depth. Thus the labels on all children of any vertex are distinct and an event $\mathcal{B}_i$ occurs at most once at each depth.

If there is an event $\mathcal{B}_j$ at depth d and $\mathcal{B}_j$ is dependent with $\mathcal{B}_i$, then we can add $\mathcal{B}_i$ as a child of the vertex at depth d or a vertex with higher depth. Thus the labels on all children of any vertex are independent and all labels are independent at each level. ■

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

Lemma 16.6 *Let T be a witness tree and E a random execution of the algorithm. Then*

$$\Pr(T \textit{ appears in } E) \leq \prod_{v \in V(T)} Pr(\mathcal{B}_v)$$

where $\mathcal{B}_v$ denotes the event labeling node $v \in V(T)$.

Proof: Fix a witness tree T.

Define an evaluation for T. In reverse BFS order, visit $v \in V(T)$ and resample their variables $\mathrm{vbl}(\mathcal{B}_v)$ (independently of previous resamplings).

We say that T was violated, if for all $v \in V(T)$, event $\mathcal{B}_v$ was violated by resampling of $\mathcal{B}_v$. Obviously,

$$\Pr(T \text{ was violated}) = \prod_{v \in V(T)} \Pr(\mathcal{B}_v)$$

For each variable x_j, image an infinite list of independent random resamplings. Then, when x_j needs to be resampled, it takes the next value in this sequence, and thus the Moser-Tardos 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.

For a vertex $v \in V(T)$, consider the resampling of $\text{vbl}(\mathcal{B}_v)$ in evaluation for T. Consider $x_j \in \text{vbl}(\mathcal{B}_v)$. According to the previous claim, x_j does not occur again on the same level of T. Thus, by reverse BFS ordering, the number of times x_j has been sampled prior to the resampling at v is equal to the number of vertices that have greater depth than $\text{depth}(v)$ and depend on variable x_j, and let $n_{j,v}$ denote this number.

Then consider the resamplings of $\mathcal{B}_v$ in the execution E of Moser-Tardos Algorithm. The number of times x_j has been resampled prior to the resampling of $\mathcal{B}_v$ is $n_{j,v} + 1$, since x_j was sampled for the initial setting and then at all the other times corresponding to vertices that have greater depth than $\text{depth}(v)$ in the tree.

So we define a coupling between the evaluation of T and the execution E: for the random choice of variables $\{x_1, x_2, \ldots, x_m\}$, using them for the tree T evaluation and then the Moser-Tardos Algorithm with setting immediately prior to its resampling of $\mathcal{B}_v$ as well so that the first resampling of x_j in the tree T evaluation gives the initial setting of x_j in E.

In this way, if $\mathcal{B}_v$ is violated in T, in E at the corresponding time the event $\mathcal{B}_v$ will be violated prior to this time since otherwise the algorithm would not select $\mathcal{B}_v$ for resampling.

Therefore,

$$\Pr(T \text{ appears in } E) \leq \Pr(T \text{ was violated}) = \prod_{v \in V(T)} \Pr(\mathcal{B}_v)$$

∎

16.3 Proof of Algorithmic Lovász Local Lemma

Definition 16.7 *For event $\mathcal{B}_i$, let N_i denote the number of times that $\mathcal{B}_i$ appears in original algorithm E. Thus N_i is the number of trees with root $\mathcal{B}_i$ in execution E.*

Consider the following Galton-Watson process to build a tree T randomly:

Algorithm 3: Galton-Watson Process

```
1 Fix the root to be B_i;
2 for B_j ∈ D_i^+ do
3     Add B_j as a child of B_i with probability β_j;
4     Leave out B_j with probability 1 − β_j;
5 Repeat if B_j is added
```

Fix a tree with root $\mathcal{B}_i$ and let $P_T = \Pr(\text{Galton-Watson process produces } T)$. We have the following lemma:

Lemma 16.8

$$P_T = \frac{\beta_i}{1 - \beta_i} \prod_{v \in V(T)} \beta'_v$$

where

$$\beta'_v = \beta_v \prod_{j \in D_v} (1 - \beta_j)$$

Proof: For $v \in V(T)$, let w_v denote dependencies of $\mathcal{B}_v$ which are not children of v in T, namely, $w_v = D_v^+ \backslash N_T^-(v)$ where $N_T^-(v)$ denotes the children of v in T. Then

$$\begin{aligned}
P_T &= \frac{1}{\beta_i} \prod_{v \in V(T)} \beta_v \prod_{j \in w_v} (1 - \beta_j) \\
&= \frac{1 - \beta_i}{\beta_i} \prod_{v \in V(T)} \frac{\beta_v}{1 - \beta_v} \prod_{j \in D_v^+} (1 - \beta_j) \\
&= \frac{1 - \beta_i}{\beta_i} \prod_{v \in V(T)} \beta_v \prod_{j \in D_v} (1 - \beta_j) \\
&= \frac{1 - \beta_i}{\beta_i} \prod_{v \in V(T)} \beta'_v
\end{aligned}$$

■

Now, we are in a position to bound $\mathbb{E}[N_i]$.

Lemma 16.9

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

Proof:

$$\begin{aligned}
\mathbb{E}[N_i] &= \sum_T \Pr(T \text{ appears in } E) \\
&\leq \sum_T \prod_{v \in V(T)} \Pr(\mathcal{B}_v) \\
&\leq \sum_T \prod_{v \in V(T)} \beta_v \prod_{j \in D_v} (1 - \beta_j) \\
&\leq \sum_T \prod_{v \in V(T)} \beta'_v \\
&= \frac{\beta_i}{1 - \beta_i} \sum_T P_T \\
&= \frac{\beta_i}{1 - \beta_i}
\end{aligned}$$

as the Galton-Watson Process produces 1 tree. ■

Note the running time of the algorithm is proportional to $\sum_{i=1}^n N_i$. As $\mathbb{E}[N_i] \leq \frac{\beta_i}{1-\beta_i}$, we have proved the algorithmic version of Lovász Local Lemma.

References

[1] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász Local Lemma. *Computing Research Repository (CoRR)*, abs/0903.0544, 2009.

[2] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and Finite Sets*, 10(2):609–627, 1975.