Chapter 55

Algorithmic Version of Lovász Local Lemma

By Sariel Har-Peled, April 26, 2022[1]

As for me, they took away my toy merchant, wishing with him to banish all toys from the world.

The tin drum, Gunter Grass

55.1. Introduction

Let $\mathcal{P}$ be a collection of independent random variables in some probability space Ω. We are interested in (bad) events that are determined by these variables. Specifically, for an event B, there is a set of variables $S \subseteq \mathcal{P}$ that determine if B happens. There minimal such set of variables is denoted by $\mathrm{vbl}(B)$ (note, that it is unique). We assume that the set $\mathrm{vbl}(B)$ is either easily computable or available, for any event of interest.

Consider a (finite) family $\mathcal{B}$ of such (bad) events. The ***dependency graph*** $\mathsf{G} = \mathsf{G}(\mathcal{B})$, with the vertices being the events of $\mathcal{B}$, and two events $B_1, B_2 \in \mathcal{B}$ are connected by an edge $\iff$ $\mathrm{vbl}(B_1) \cap \mathrm{vbl}(B_2) \neq \emptyset$. Let $\Gamma(B)$ be all the neighbors B in G, and let $\Gamma_+(B) = \{B\} \cup \Gamma(B)$. Observe that B is independent of the events in $\mathcal{B} \setminus (\{B\} \cup \Gamma(B))$.

A specific assignment to the variables of S ***violates*** B, if it causes it to happen (remember, it is a bad event).

Task. Our purpose is to compute an assignment to the variables of $\mathcal{P}$, such that none of the bad events of $\mathcal{B}$ happens.

Algorithm. Initially, the algorithm assigns the variables of $\mathcal{P}$ random values. As long as there is a violated event $B \in \mathcal{B}$, resample all the variables of $\mathrm{vbl}(B)$ (independently according to their own distributions) – this is a ***resampling*** of B. The algorithm repeats this till no event is violated.

Finer details. We fix some arbitrary strategy (randomized or deterministic) of how to pick the next event. This now fully specify the algorithm.

Remark. Of course, it is not clear that the algorithm would always succeeds. Let us just assume, for the time being, that we are in a case where the algorithm always finds a good assignment (which always exists).

55.1.1. Analysis

Let $L(i) \in \mathcal{B}$ be the event that was resampled in the ith iteration of the algorithm, for $i > 0$. The sequence formed by L is the ***log*** of the execution.

A ***witness tree*** $T = (T, \sigma_T)$ is a rooted tree T together with a labeling σ_T. Here, every node $v \in T$ is labeled by an event $\sigma_T(v) \in \mathcal{B}$. If a node u is a child of v in T, then $\sigma_T(u) \in \Gamma_+(v)$. Two nodes in a

tree are ***siblings*** if they have a common parent. If all siblings have distinct labels then the witness tree is ***proper***.

For a vertex v of T, let $[v] = \sigma_T(v)$.

Theorem 55.1.1. *Let $\mathcal{P}$ be a finite set of independent random variables in a probability space, and let $\mathcal{B}$ be a set of (bad) events determined by these variables. If there is an assignment $x : \mathcal{B} \to (0,1)$, such that*

$$\forall B \in \mathcal{B} \qquad \mathbb{P}[B] \leq x(B) \prod_{C \in \Gamma(B)} \big(1 - x(C)\big),$$

then there exists an assignment for the variables of $\mathcal{P}$ such that no event in $\mathcal{B}$ happens. Furthermore, in expectation, the algorithm described above resamples, any event $B \in \mathcal{B}$, at most $x(B)/\big(1 - x(B)\big)$ times. Overall, the expected number of resampling steps is at most $\sum_{B \in \mathcal{B}} x(B)/\big(1 - x(B)\big)$.