A constructive proof of the general Lovász local lemma

Robin A. Moser and Gábor Tardos, 2010

Leonid Prokupets

May 3, 2015

Outline

Introduction

- The typical goal of the probabilistic method is to prove the existence of combinatorial objects meeting a prescribed collection of criteria.
- Usually, we have a collection of bad events $A_1, \ldots, A_n$ that we are trying to avoid.
- If $\sum_{i=1}^{n} \Pr[A_i] < 1$ then clearly there is a positive probability that none of them occurs. However, in many cases this approach is not powerful enough, because $\sum_{i=1}^{n} \Pr[A_i]$ may be substantially larger than $\Pr[\bigcup_{i=1}^{n} A_i]$.
- If the events $A_1, \ldots, A_n$ are mutually independent (and non-trivial), then we have $\Pr[\bigwedge_{i=1}^{n} \overline{A}_i] = \prod_{i=1}^{n} \Pr[\overline{A_i}] > 0$ even though the probabilities $\Pr[Ai]$ can be very close to 1 and their sum can be arbitrarily large.
- It is natural to expect that something similar holds even if the events are not entirely independent.

General Lovász local lemma

Theorem (Erdős and Lovász, 1975)

Let $\mathcal{A}$ be a finite set of events in a probability space.
For $A \in \mathcal{A}$ let $\Gamma(A)$ be a subset of $\mathcal{A}$ satisfying that A is independent from the collection of events $\mathcal{A} \setminus (\{A\} \cup \Gamma(A))$.
If there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that

$$\forall A \in \mathcal{A} : \Pr[A] \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)),$$

then with positive probability no event $A \in \mathcal{A}$ holds.

Symmetric Lovász local lemma

Special case of the General Lovász local lemma:

Theorem

Let $\mathcal{A}$ be a finite set of events in a probability space.
For $A \in \mathcal{A}$ let $\Gamma(A)$ be a subset of $\mathcal{A}$ satisfying that A is independent from the collection of events $\mathcal{A} \setminus (\{A\} \cup \Gamma(A))$.
If the following conditions are met:

- $\forall A \in \mathcal{A} : |\Gamma(A)| \leq d$, *i.e. each event A depends on at most d other events*
- $\forall A \in \mathcal{A} : \Pr[A] \leq p$, *i.e. the probability of each event A is at most p*
- $ep(d+1) \leq 1$, *where e is the base of the natural logarithm*

Then with positive probability no event $A \in \mathcal{A}$ holds.

Example application

Definition

A hypergraph $H = (V, E)$ is **two-colorable** if there is a coloring of V by two colors so that no edge $f \in E$ is monochromatic.

Theorem

Let H be a hypergraph in which every edge has at least k vertices and intersects at most d other edges. If $e(d+1) \leq 2^{k-1}$, then H is two-colorable.

Motivation

- In the basic probabilistic method, we usually prove that almost all of the considered objects are good. So if we want to find a good object, we can select an object at random, and we have a very good chance of selecting a good one.
- In contrast, the Lovász local lemma guarantees that the probability of avoiding all bad events is positive, but this probability is typically very small!
- For example, if $A_1, \ldots, A_n$ are independent events, with probability $1/3$ each, say, in which case the Local Lemma applies, then the probability of none A_i occurring is only $(2/3)^n$. So good objects guaranteed by the local Lemma can be extremely rare.
- The original proof of the local lemma is non-constructive and does not yield an efficient procedure for searching the probability space for a point with the desired property.
- The purpose is to give an alternative, algorithmic proof that provides such a procedure.

History

- 1991: J. Beck demonstrated that algorithmic versions of the Local Lemma exist. He formulated his strategy in terms of hypergraph 2-coloring as a specific application of the lemma and proved that if in a hypergraph, every edge contains at least k vertices and shares common vertices with no more than roughly $2^{k/48}$ other edges, then a polynomial time algorithm can 2-color the vertices without producing a monochromatic edge.
- 1991: N. Alon improved the threshold to essentially $2^{k/8}$.
- 1998: M. Molloy B. Reed provided a general framework capturing the requirements a particular application has to meet so as to become tractable by the tools of Beck and Alon.
- 2008: A. Srinivasan provided another improvement that reduced the gap to a threshold of essentially $2^{k/4}$.
- 2008: R. Moser, improvement to roughly $2^{k/2}$.
- 2009: R. Moser lowered the threshold to roughly $2^k/32$.

General setting

- Let $\mathcal{P}$ be a finite collection of mutually independent random variables in a fixed probability space Ω.
- We consider events A that are determined by the values of some subset $S \subseteq \mathcal{P}$ of these variables.
- We say that an evaluation of the variables in S *violates* A if it makes A happen.
- Denote by $\mathrm{vbl}(A)$ the unique minimal subset $S \subseteq \mathcal{P}$ that determines A.
- Let $\mathcal{A}$ be a finite family of events in Ω determined by $\mathcal{P}$.
- We define the *dependency graph* $G = G_{\mathcal{A}}$ for $\mathcal{A}$ to be the graph on vertex set $\mathcal{A}$ with an edge between events $A, B \in \mathcal{A}$ if $A \neq B$ but $\mathrm{vbl}(A) \cap \mathrm{vbl}(B) \neq \emptyset$.
- For $A \in \mathcal{A}$ we write $\Gamma(A) = \Gamma_{\mathcal{A}}(A)$ for the neighborhood of A in G and $\Gamma^{+}(A) := \Gamma(A) \cup \{A\}$ the inclusive neighborhood of A.
- Given the family $\mathcal{A}$ of events as above our goal is not only to show that there exists an evaluation that does not violate any event in the family but to efficiently find such an evaluation.

The Sequential Solver

Algorithm 1 The Sequential Solver

```
1: for all $P \in \mathcal{P}$ do
2:     $v_P \leftarrow$ a random evaluation of $P$
3: end for
4: while $\exists A \in \mathcal{A} : A$ is violated when $(P = v_P : \forall P \in \mathcal{P})$ do
5:     Pick an arbitrary violated event $A \in \mathcal{A}$
6:     for all $P \in \mathrm{vbl}(A)$ do
7:         $v_P \leftarrow$ a new random evaluation of $P$
8:     end for
9: end while
10: return $(v_P)_{P \in \mathcal{P}}$
```

Main Theorem

Theorem

Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space.
Let $\mathcal{A}$ be a finite set of events determined by these variables.
If there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that

$$\forall A \in \mathcal{A} : \Pr[A] \leq x(A) \prod_{B \in \Gamma_{\mathcal{A}}(A)} (1 - x(B)),$$

then there exists an assignment of values to the variables $\mathcal{P}$ not violating any of the events in $\mathcal{A}$. Moreover the randomized algorithm described above resamples an event $A \in \mathcal{A}$ at most an expected $x(A)/(1 - x(A))$ times before it finds such an evaluation. Thus, the expected total number of resampling steps is at most $\sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}$.

Execution Logs and Witness Trees

Definition

Let $C : \mathbb{N} \to \mathcal{A}$ list the events as they have been selected for resampling in each step. If the algorithm terminates, C is partial and defined only up to the given total number of steps carried out. We call C the **log** of the execution.

Definition

A **witness tree** $\tau = (T, \sigma_T)$ is a finite rooted tree T together with a labelling $\sigma_T : V(T) \to \mathcal{A}$ of its vertices with events such that the children of a vertex $u \in V(T)$ receive labels from $\Gamma^+(\sigma_T(u))$. If distinct children of the same vertex always receive distinct labels we call the witness tree **proper**. To shorten notation, we will write $V(\tau) := V(T)$ and for any $v \in V(\tau)$, we write $[v] := \sigma_T(v)$.

Execution Logs and Witness Trees

- Given the log C, we will associate with each resampling step t carried out a witness tree $\tau_C(t)$ that can serve as 'justification' for the necessity of that correction step.
- Let us define $\tau_C^{(t)}(t)$ to be an isolated root vertex labelled $C(t)$.
- Then, going backwards through the log, for each $i = t-1, t-2, \ldots, 1$, we distinguish two cases:
    - If there is a vertex $v \in \tau_C^{(i+1)}(t)$ such that $C(i) \in \Gamma^+([v])$, then we choose among all such vertices the one having the maximum distance from the root (breaking ties arbitrarily) and attach a new child vertex u to v that we label $C(i)$, thereby obtaining the tree $\tau_C^{(i)}(t)$.
    - Otherwise, we skip time step i and define $\tau_C^{(i)}(t) := \tau_C^{(i+1)}(t)$.
- We say that the witness tree τ occurs in the log C if there exists $t \in \mathbb{N}$ such that $\tau_C(t) = \tau$.

Execution Logs and Witness Trees

Lemma

Let τ be a fixed witness tree and C the (random) log produced by the algorithm.

1. *If τ occurs in C, then τ is proper.*
2. *The probability that τ appears in C is at most $\prod_{v \in V(\tau)} \Pr[[v]]$.*

Execution Logs and Witness Trees

- Let C be the log of the execution of our algorithm.
- For any event $A \in \mathcal{A}$, denote by N_A the random variable that counts how many times the event A is resampled during the execution of our algorithm, that is the number of time steps t with $C(t) = A$.
- Let t_i denote the i-th such time step.
- Let $\mathcal{T}_A$ denote the set of all proper witness trees having the root labelled A
- We saw that for each i:
  - $\tau_C(t_i) \in \mathcal{T}_A$.
  - $\tau_C(t_i)$ contains exactly i vertices labeled A and thus $\tau_C(t_i) \neq \tau_C(t_j)$ for $i \neq j$.
- Therefore, N_A not only counts the number of distinct proper witness trees occurring in C that have their root labeled A, that is,

$$N_A = \sum_{\tau \in \mathcal{T}_{\mathcal{A}}} 1_{\{\tau \text{ occurs in C}\}}$$

Random Generation of Witness Trees

- Fix an event $A \in \mathcal{A}$.
- Consider the following branching process for generating a proper witness tree having its root labeled A.
- In the first round, we produce a singleton vertex labeled A.
- Then, in each subsequent round, we consider each vertex v produced in the previous round independently and for each event $B \in \Gamma^+([v])$, we add to v a child node carrying the label B with probability $x(B)$ or skip that label with probability $1 - x(B)$. All these choices are independent.
- The process continues until it dies out naturally because no new vertices are born in some round (depending on the probabilities used, there is, of course, the possibility that this never happens).

Random Generation of Witness Trees

Lemma

Let τ a fixed proper witness tree with its root vertex labeled A.
The probability p_τ that the process described above yields exactly the tree τ is

$$p_\tau = \frac{1 - x(A)}{x(A)} \prod_{v \in V(\tau)} x'([v])$$

where $x'(B) := x(B) \prod_{C \in \Gamma(B)} (1 - x(C))$.

Random Generation of Witness Trees

To conclude the proof the main theorem, we have

$$\begin{aligned}
\mathbb{E}(N_A) &= \sum_{\tau \in \mathcal{T}_A} \Pr[\tau \text{ appears in the log } C] \\
&\leq \sum_{\tau \in \mathcal{T}_A} \prod_{V \in V(\tau)} \Pr[[v]] \quad \text{(First lemma)} \\
&\leq \sum_{\tau \in \mathcal{T}_A} \prod_{V \in V(\tau)} x'([v]) \quad \text{(Theorem assumption)} \\
&= \frac{x(A)}{1 - x(A)} \sum_{\tau \in \mathcal{T}_A} p_\tau \quad \text{(Second lemma)} \\
&\leq \frac{x(A)}{1 - x(A)}
\end{aligned}$$

□

The Parallel Solver

Algorithm 2 The Parallel Solver

```
 1: for all $P \in \mathcal{P}$ do in parallel
 2:     $v_P \leftarrow$ a random evaluation of $P$
 3: end for
 4: while $\exists A \in \mathcal{A} : A$ is violated when $(P = v_P : \forall P \in \mathcal{P})$ do
 5:     $S \leftarrow$ a maximal independent set in the subgraph of $\mathcal{A}$ induced by
            all events which where violated when $(P = v_P : \forall P \in \mathcal{P})$,
            constructed in parallel
 6:     for all $P \in \bigcup_{A \in S} \mathrm{vbl}(A)$ do in parallel
 7:         $v_P \leftarrow$ a new random evaluation of $P$
 8:     end for
 9: end while
10: return $(v_P)_{P \in \mathcal{P}}$
```

The Parallel Solver

Theorem

Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space.
Let $\mathcal{A}$ be a finite set of events determined by these variables.
If $\epsilon > 0$ there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that

$$\forall A \in \mathcal{A} : \Pr[A] \leq (1-\epsilon)x(A) \prod_{B \in \Gamma_{\mathcal{A}}(A)} (1 - x(B)),$$

then the parallel version takes an expected

$$O\left(\frac{1}{\epsilon} \log \sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}\right)$$

steps before it finds an evaluation violating no event in $\mathcal{A}$.

The Parallel Solver

- Consider an arbitrary execution of the parallel version of the algorithm
- Choose an arbitrary ordering of the violated events selected for resampling at each step and consider that these resamplings are done in that order sequentially.
- This way we obtain an execution of the sequential algorithm.
- Let S_j be the segment of the log C of this execution that corresponds to resamplings done in step j of the parallel algorithm.
- We call the maximal depth of a vertex in a witness tree the *depth* of the tree.

Lemma

If $t \in S_j$, then the depth of $\tau_C(t)$ is $j - 1$.

The Parallel Solver

- Let $Q(k)$ denote the probability that the parallel algorithm makes at least k steps.
- By the last lemma, some witness tree of depth $k-1$ (and thus, at least k vertices) must occur in the log in this case.
- Let $\mathcal{T}_A(k)$ be the set of wtness trees in $\mathcal{T}_A$ having at least k vertices.
- We have

$$Q(k) \leq (1-\epsilon)^k \sum_{A \in \mathcal{A}} \frac{x(A)}{1-x(A)}$$

which implies the last theorem.

$\square$

Thank you!