

# The Lovász local lemma

## *An introduction and some recent results*

Lefteris Kirousis

National and Kapodistrian University of Athens

Joint work with:



I. Giotis



J. Livieratos



K. Psaromiligkos



D. Thilikos

# Trivial observation

If

- ▶  $E_1, \dots, E_m$  mutually independent “undesirable” events over a probability space  $\Omega$ .
- ▶  $\Pr[E_i] \leq p < 1, i = 1, \dots, m$ , where  $p$  constant ( $m$  could be large).

Then

$$\Pr \left[ \bigwedge_{i=1}^m \overline{E_i} \right] \geq (1 - p)^m > 0.$$

Therefore there exists a point in  $\Omega$  where all undesirable events do *not* occur. Probabilistic method.

# Lovász Local Lemma

Suppose that each  $E_i$  is mutually independent of all but  $d$  of the other events.

Any sufficient condition for positive probability to avoid all  $E_i$ 's must involve  $p$  and  $d$ .

## Theorem (Lovász Local Lemma (LLL), 1975)

*If*

$$4pd \leq 1$$

(INITIAL SUFFICIENT CONDITION)

*then*

$$\Pr \left[ \bigwedge_{i=1}^m \overline{E_i} \right] \geq (1 - 2p)^m > 0.$$

# Non-algorithmic proof of LLL

Hint: Prove by induction on  $s$  that for every  $S \subseteq \{1, \dots, m\}$  with  $|S| \leq s$  and any  $i$ ,

$$\Pr[E_i \mid \bigwedge_{j \in S} \overline{E_j}] \leq 2p. \quad (1)$$

To get that

$$\Pr \left[ \bigwedge_{i=1}^m \overline{E_i} \right] \geq (1 - 2p)^m > 0.$$

*“The proof is so elementary that it could, and I think it should, be taught in a first course in probability. It has had and continues to have a profound effect on probabilistic method.”*

– J. Spencer, *Ten lectures...* (Durango lectures)

# Applications for SAT

$(k, s)$ -CNF Boolean formulas: exactly  $k$  literals per clause and *every* variable appears in at most  $s$  clauses.

$(k, s)$ -SAT problem: satisfiability of  $(k, s)$ -CNF formulas.

Let  $f(k)$  be the max number, such that every  $(k, f(k))$ -CNF formula is satisfiable.

Trivially,  $f(k) < 2^k$

# More on SAT

## Theorem (Kratohvíl et al. 1993)

For  $k \geq 3$ ,  $(k, f(k) + 1)$ -SAT is NP-complete.

- ▶  $f(k)$  is a threshold "jump" from always satisfiable to being hard to check if satisfiable.

Let  $l(k)$  be the max number  $x$ , such that if all clauses of a formula  $\phi$  share variables with at most  $x$  *other* clauses, then  $\phi$  is satisfiable.

LLL immediately implies:

$$l(k) \geq 2^{k-2}.$$

Obviously:  $f(k) \geq \frac{l(k)}{k} + 1$ . Therefore:  $f(k) \geq \frac{2^{k-2}}{k} + 1$ .

## Weaker sufficient conditions

$$ep(d+1) \leq 1 \quad (\text{EULER NUMBER SUFFICIENT CONDITION})$$

$G = (V, E)$  be a dependency graph for  $E_i$ :

- ▶ A *simple* graph with vertices the events  $E_i$  so that every event is mutually independent of all events not connected with.

$\exists$  numbers  $\chi_i \in [0, 1)$  :

$$\Pr[E_i] \leq \chi_i \prod_{\{i,j\} \in E} (1 - \chi_j) \quad (\text{ASYMMETRIC LLL})$$

# Algorithmic solution

An algorithmic solution presupposes to locate a point in  $\Omega$  whose probability can be as small as

$$(1 - 2p)^m.$$

- J. Spencer 1994, *Ten lectures...* (Durango lectures):  
“*Algorithm? Sometimes!*”



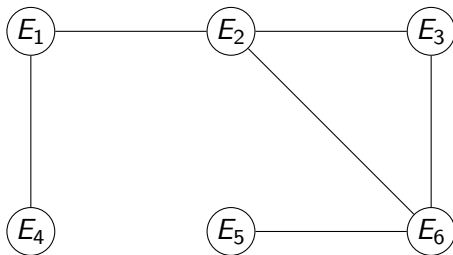
# The variable framework

- Let  $X_1, \dots, X_l$  independent random variables.
- Each event  $E_i$  depends on a subset of the variables.
- This subset is called the scope of  $E_i$ ; denoted by  $e_i$ .
- Two events  $E_i, E_j$  are dependent ( $E_i \sim E_j$ ) iff  $e_i \cap e_j \neq \emptyset$ .
- Dependency graph: Vertices correspond to events, unconnected vertices must correspond to independent events. Degree:  $d$

# The dependency graph, an example

Seven variables:  $X_1, \dots, X_7$ , six events:  $E_1, \dots, E_6$

$e_1 = \{X_1, X_2, X_3\}$ ,  $e_2 = \{X_3, X_4\}$ ,  $e_3 = \{X_4, X_5, X_6, X_7\}$ ,  
 $e_4 = \{X_2\}$ ,  $e_5 = \{X_6\}$ ,  $e_6 = \{X_4, X_5, X_7\}$



## Back to the variable framework

- Let  $X_1, \dots, X_l$  independent random variables.
- Each event  $E_i$  depends on a subset of the variables.
- This subset is called the scope of  $E_i$ ; denoted by  $e_i$ .
- Two events  $E_i, E_j$  are dependent ( $E_i \sim E_j$ ) iff  $e_i \cap e_j \neq \emptyset$ .
- Dependency graph: Vertices correspond to events, unconnected vertices must correspond to independent events. Degree:  $d$
- $N_i$ , the neighborhood of  $E_i$  is the set of events  $E_j$  such that  $E_j \sim E_i$  ( $E_i$  included in  $N_i$ ).
- $|N_i| \leq d + 1, \forall i = 1, \dots, m$ .
- Assume sampling the  $X_i$ 's can be done efficiently.

## Back to a trivial case

- ▶  $E_1, \dots, E_m$  “undesirable” events, each depending on some of the variables  $X_1, \dots, X_l$ , such that  $\forall i, \Pr[E_i] \leq p < 1$  ( $p$  a constant).
- ▶ Assume that the scopes of the events are pairwise disjoint (events are mutually independent).

To find an assignment to the variables such that  $\bigwedge_{i=1}^m \overline{E_i}$ ,

do: for  $i = 1, \dots, m$ , sample and resample the variables in the scope of  $E_i$  until  $E_i$  does not occur. Expected time for each  $i$ :  $\frac{1}{1-p}$ .

What if  $E_i$  are not independent (their scopes intersect)?

# Basic idea: 2nd law of thermodynamics

# Basic idea: 2nd law of thermodynamics

You often see this:

# Basic idea: 2nd law of thermodynamics

However, you never see this:

## 2nd law of thermodynamics

Physicist's view: There is no spontaneous entropy decrease. You need energy to transfer heat from cold to hot.

Layman's view: There is no free lunch. You've got to do work to build something.

Information theorist's view: There is no lossless binary code with average code length of a symbol less than the entropy of a random symbol in the source (source entropy). You have to give up information to suppress more.

Combinatorialist's view: There are far more ways to lay a collection of bricks into a heap than into a building.

Idea: Design a randomized algorithm that can only fail in a structured way.



# Moser's algorithm (2010)

- Start with a random assignment to the variables  $X_1, \dots, X_I$  obtained sampling them independently.
- Choose an event  $E$  that occurs under the current assignment and resample the variables in its scope (independently).
- Repeat
- Always give priority to events that are neighbors to (share a variable with) the event examined last.

# Moser's algorithm, (pseudo-)formally

## ALGORITHM

- 1: Sample all  $X_i$ 's (independently);  $\alpha$  be the resulting assignment.
- 2: **while**  $\exists$  event that occurs under the current assignment, let  $E_i$  be the least indexed such event **do**
- 3:     RESAMPLE( $E_i$ )
- 4: **end while**
- 5: Output current assignment  $\alpha$

## RESAMPLE( $E_i$ )

- 1: Resample the variables in the scope  $e_i$  (independently).
- 2: **while** some  $E_j \in N_i$  occurs for the current assignment  $\alpha$ , let  $E_j$  be the least indexed such event **do**
- 3:     RESAMPLE( $E_j$ )
- 4: **end while**

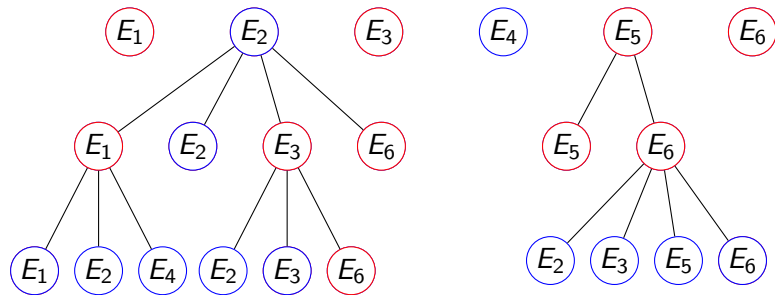
# Obvious, and almost obvious facts

- ▶ *Root Phase*: The part of the algorithm within a root call of `RESAMPLE`
- ▶ *Phase*: The part of the algorithm within a call of `RESAMPLE`
- ▶ Phases are nested

## Facts

- if `ALGORITHM` halts, it produces an assignment that avoids all undesirable events. recall
- None of the events of a phase occurs at the end of it.
- All events that do not occur at the start of a phase, do not occur at the end of it as well.
- Therefore there are at most  $m$  (number of events) successive *root phases*.

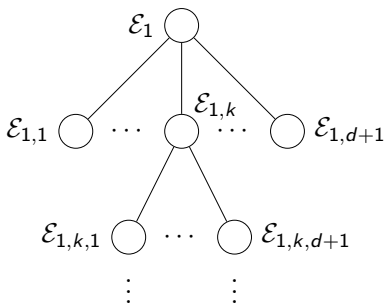
# The witness forest I



# The witness forest, formally

- We restrict our attention to a root phase only, call it  $P$  (therefore, we deal with a tree). recall
- Consider the events of all recursive calls within  $P$  *in the order they appear on top of  $P$ 's recursion stack*.
- Thus we get a tree whose nodes are the events of the recursive calls of  $P$ . We call this tree the *witness tree* of the phase  $P$ .

# The witness forest continued



- $\mathcal{E}_1$  is the event of the root call.
- Let  $\mathcal{E}_{1,k}$  be the  $k$ -th child of  $\mathcal{E}_1$ . Continue recursively.
- The children of any  $\mathcal{E}$  on the tree are events that are  $\sim \mathcal{E}$  and are **pairwise different**.
- Therefore the out-degree of the tree is  $d + 1$ .
- The same event may appear more than once in the tree. recall

# More on the witness forest

ALGORITHM produces a sequence  $\alpha_1, \dots, \alpha_n$  of value assignments such that

- ▶  $\mathcal{E}_i$  occurs for  $\alpha_i$ .
- ▶  $\alpha_1$  is a random assignment to *all* variables.
- ▶  $\alpha_{i+1}$  is obtained from  $\alpha_i$  by resampling the variables in the scope of  $\mathcal{E}_i$ . recall

# The previous approach to analysis

## The entropic argument

Basic ingredients of the **previous** proof

- From the witness tree and the output assignment reconstruct the history of the random choices of the algorithm (uniqueness property).
- So we have a lossless coding of the process.
- Conclude that the expected length of the witness forest is exponentially small.



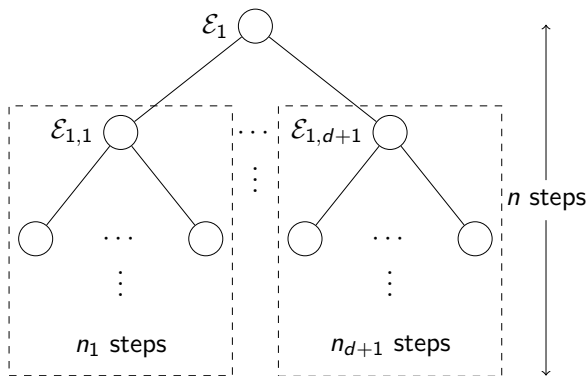
## Alternative proof: Direct computation of the distribution of the $\#(\text{steps})$ of ALGORITHM Giotis et al. 2015

$Q_n$ : Probability that the phase  $P$  lasts for at least  $n$  recursive calls equals the probability that  $P$  has a witness tree of size at least  $n$ .

- $Q_n$ 's first factor is  $p$ , because  $\mathcal{E}_1$  occurs under the random  $\alpha_1$ .
- Subsequent factor:  $Q_{n_1} \cdots Q_{n_{d+1}}$ , where  $Q_{n_1}, \dots, Q_{n_{d+1}}$  are the probabilities that the witness subtrees starting from the children of  $\mathcal{E}_1$ , have sizes  $n_1, \dots, n_{d+1}$ , where

$$\sum_{i=1}^{d+1} n_i = n - 1.$$

- $Q_{n_{i+1}}$  should be computed conditional on  $\mathcal{E}_1$  and the events corresponding to the probabilities  $Q_{n_1}, \dots, Q_{n_i}$ .



Formally,

$$Q_n = p \sum_{\substack{n_1 + \dots + n_{d+1} = n-1 \\ n_1, \dots, n_{d+1} \geq 0}} Q_{n_1} \cdots Q_{n_{d+1}}.$$

# Preservation of distribution

- $Q_n$  and  $Q_{n_i}$  “refer to the same distribution”, the original one, where all variables are sampled independently.
- Why?
- Because resampling the variables of an event –immediately after it is “seen” to occur– “destroys” all information exposed by the knowledge of its occurrence.
- Conditioning that the event actually occurred is necessary for this “re-initialization of the distribution”.

Throw two binary fair dice. Output follows uniform distribution:

0 0	0 1	1 0	1 1
-----	-----	-----	-----

Rethrow the second dice if it is an ace (output is not uniform):

0 0	00 01	1 0	10 11
-----	-------	-----	-------

However, conditional the second die was an ace, then the distribution is uniform.

# Recurrence

So we now have a **recurrence relation**:

$$Q_n = p \sum_{\substack{n_1 + \dots + n_{d+1} = n-1 \\ n_1, \dots, n_{d+1} \geq 0}} Q_{n_1} \cdots Q_{n_{d+1}}, \quad Q_0 = 1,$$

Where all  $Q_i$  are computed with respect to the same distribution, that of the independent variables.

The above recurrence holds for all phases.

## Solution of the recurrence

$$Q_n = p \sum_{\substack{n_1 + \dots + n_{d+1} = n-1 \\ n_1, \dots, n_{d+1} \geq 0}} Q_{n_1} \cdots Q_{n_{d+1}}, \quad Q_0 = 1.$$

Multiply both sides with  $z^n$  and add for  $n = 1, \dots, \infty$ , to get that the OGF  $Q(z)$  of  $Q_n$  satisfies:

$$Q(z) - 1 = zpQ^n(z).$$

Now apply the Lagrange inversion and then Stirling approximation to get that  $Q_n$  is bounded by

$$C \left( \left( 1 + \frac{1}{d} \right)^d p(d+1) \right)^n < C (ep(d+1))^n,$$

for some constant  $C$ .

Therefore...

# Algorithmic Lovász Local Lemma

If

$$\left(1 + \frac{1}{d}\right)^d p(d+1) < 1$$

(and therefore if  $ep(d+1) \leq 1$ ),  
then

ALGORITHM stops within  $n$  steps with probability exponentially close to 1  
with  $n$ .

## But there is a catch! 😞

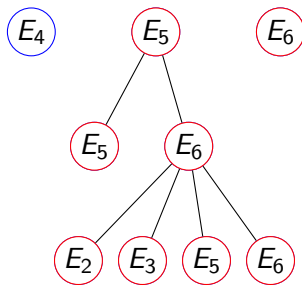
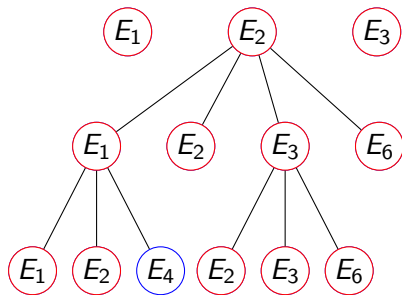
- We claimed that resampling the variables of an event –immediately after it is “seen” to occur– “destroys” *all* information exposed by the knowledge of its occurrence.
- This is *wrong!*
- Because when an event is selected, additional knowledge besides its occurring is utilized to make the selection (e.g. that it is the first occurring event).
- This additional info concerns variables not in the scope of the event selected.
- So biases are introduced that are retained even after the resampling.



## Way out of the catch 😊

- Design an algorithm, we called the **VALIDATION ALGORITHM**, that
- takes as input a candidate for a witness tree, and follows it as a blueprint for which events to consider.
- So the algorithm does *not* have to select events (the act of selecting entails comparing events and therefore biases are introduced).
- The probability  $Q_n$  of our original algorithm, is upper-bounded by the corresponding probability of the **VALIDATION ALGORITHM**.
- So the recurrence is applicable. **DONE!**

# The VALIDATION ALGORITHM



## So what?

What is the advantage of this “direct probabilistic method”?

*Whereas Moser and Tardos use a backward-looking argument to find witness trees in the algorithms “log”, Giotis et al. analyze a forward-looking structure: the tree of resampled events and their dependencies, looking forward in time. This viewpoint seems more natural and suitable for extensions.*

— Harvey and Vondrak, 2015.

Moreover, our method yields an exponentially small estimate for Moser-type algorithms to fail. Whereas, the entropy based approach gives only an estimate of the expectation until success.

# Extensions

**Lopsided** LLL: Only dependencies between **negatively** dependent events count (Erdős, 1991).


We give below, in the variable framework, a notion of dependency stronger than the classical negative dependence that it leads to sparser graphs, which in certain examples turn out to be “quite” sparser.

## Definition

The event  $E_j$  is called *d-dependent* on the event  $E_i$  if there is an assignment  $\alpha$  such that:

- under  $\alpha$ ,  $E_i$  holds but  $E_j$  does not, and
- the variables of  $E_i$  can be resampled so that  $E_i$  stops holding, but  $E_j$  holds.

# d-Dependence

- Notice that d-dependency is a directed notion. Intuitively  :  $E_j$  is d-dependent on  $E_i$  if it is *possible* that some *successful* attempt to avoid the occurrence of  $E_i$  may end up with  $E_j$  occurring, although initially it did not.
- d-Dependence is reminiscent of the notion of lopsidedependency introduced by Moser and Tardos (not a directed notion).
- Nevertheless, directed dependence notions were introduced by Harvey and Vondrak (2015) and by Achlioptas and Iliopoulos (2014).
- However, d-dependency is stronger than all (the latter two however are stated for general probability spaces, not products of independent random variables).
- Actually, there are cases where the d-dependency graph is empty, whereas the dependency graphs for all previous notions are not.

## Further applications? Even if non-algorithmic (with a grain of salt – not published yet)

### Theorem (K., Livieratos et al.)

*Assume that that any variable belongs to the scope of at most  $k$  events and that the probability of at least one of these events occurring is at most  $q$ . If  $ekq \leq 1$  then there is an assignment that avoids all events.*

- ▶ If  $q$  is computed by union bound, then we essentially get the classical result.
- ▶ So the above result offers an advantage in case  $q$  can be estimated more cleverly than using union bound.

# Drawbacks of the previous approach

But how to algorithmically find this occurring neighbor?

- ▶ Outsource: call an external agent that will make the selection of occurring events.
- ▶ Since, however, one never can completely trust the quality a product an outsourced product, a situation reminiscent of Interactive Protocols arises.
- ▶ But, you loose efficiency.

## Reminder: A classical example – Graph non-isomorphism

- Two graphs  $G_1, G_2$ , we (the “verifier”) want to find out whether they are isomorphic or not.
- We are willing to ask a omniscient “prover”, however we do not trust her, as she may lie. So what to do?
- We randomly (and secretly) choose one of the graphs, permute its vertices and ask the prover which is the source graph. We repeat several times.
- If the graphs are non-isomorphic, there is a prover (the conscientious one) that will always give the correct answer.
- If the graphs are non-isomorphic, then whoever the prover, it is highly improbable to get the right answer sufficiently more than half the times.
- Technical definition: Graph non-isomorphism belongs to the complexity space known as IP (has been shown to be equal to PSPACE).



# An interactive protocol for LLL

- ▶ The verifier moves from variable to variable.
- ▶ Asks the prover to send an occurring event
- ▶ Verifies that the event is indeed occurring
- ▶ If it is, then it resamples its variables
- ▶ Recourses.

# Applications: acyclic coloring —work in progress

**Acyclic coloring:** Proper coloring so that any two-color subgraph has no cycles.

Dates back to Grünbaum [1973]. Originally it referred only to planar graphs. It was then proved that the number of colors needed for a planar graph is 5.

More recently, upper bounds for general graphs in terms of the max degree  $\Delta$  were sought.

Best result to-date:  $\alpha(G) = 1.5\Delta^{4/3} + \Delta + o(\Delta)$  [Gonçalves et al. 2010].

Such problems amenable to interactive LLL, because of the possibility to express the existence of a bicolored cycle that contains a given vertex (edge).

THANK YOU FOR YOUR ATTENTION

