

A Lower Bound for the Distributed Lovász Local Lemma

Tuomo Lempäinen

Department of Computer Science, Aalto University

Helsinki Algorithms Seminar, University of Helsinki
3rd December 2015

This is joint work with

- Sebastian Brandt,
- Orr Fischer,
- Juho Hirvonen,
- Barbara Keller,
- Joel Rybicki,
- Jukka Suomela,
- Jara Uitto.

Outline

- 1 The Lovász local lemma
- 2 Our model of distributed computing
- 3 A lower bound for the distributed LLL

The probabilistic method

- Primarily used in combinatorics to give existence proofs.
- Randomly choose objects from a certain class, and show that the probability that the object is of the desired kind is larger than zero.
- It follows that at least one such object has to exist.

The probabilistic method

- Primarily used in combinatorics to give existence proofs.
- Randomly choose objects from a certain class, and show that the probability that the object is of the desired kind is larger than zero.
- It follows that at least one such object has to exist.

- Let $\mathcal{E} = \{E_1, \dots, E_n\}$ be a set of *bad* events that make the object undesirable.
- If the events are mutually independent and $\Pr(E_i) < 1$ for each i , we have trivially $\Pr(\bigcap_{i=1}^n \overline{E}_i) > 0$.
- What if there is some dependence between the events?

The Lovász local lemma (LLL)

Theorem (Erdős and Lovász, 1975)

Let $\mathcal{E} = \{E_1, \dots, E_n\}$ be a finite set of events such that each E_i depends on at most d other events. If $\Pr(E_i) \leq p$ and $4pd \leq 1$, then there is a positive probability that none of the events occur.

The Lovász local lemma (LLL)

Theorem (Erdős and Lovász, 1975)

Let $\mathcal{E} = \{E_1, \dots, E_n\}$ be a finite set of events such that each E_i depends on at most d other events. If $\Pr(E_i) \leq p$ and $4pd \leq 1$, then there is a positive probability that none of the events occur.

Theorem (Lovász, 1977)

Let $\mathcal{E} = \{E_1, \dots, E_n\}$ be a finite set of events such that each E_i depends on at most d other events. If $\Pr(E_i) \leq p$ and $ep(d+1) \leq 1$, then there is a positive probability that none of the events occur.

$e = 2.718\dots$ is the base of the natural logarithm.

LLL: an example

Proposition

Any instance ϕ of k -SAT where no variable appears in more than $\frac{2^{k-2}}{k}$ clauses is satisfiable.

LLL: an example

Proposition

Any instance ϕ of k -SAT where no variable appears in more than $\frac{2^{k-2}}{k}$ clauses is satisfiable.

Proof.

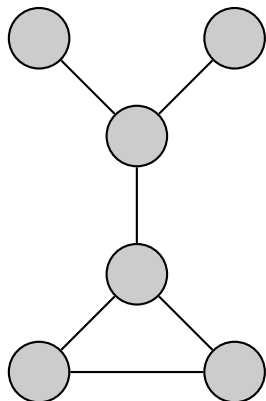
- 1 Pick a truth assignment uniformly at random.
- 2 Let E_i denote the event “clause i is not satisfied”.
- 3 $\Pr(E_i) = 2^{-k} =: p$.
- 4 E_i depends on at most $d := k \frac{2^{k-2}}{k} = 2^{k-2}$ other events.
- 5 We have $4pd = 4 \cdot 2^{-k} \cdot 2^{k-2} = 1$.
- 6 Now LLL implies that $\Pr(\bigcap \bar{E}_i) > 0$.



The algorithmic LLL

- LLL itself does not give a method for finding the object whose existence it proves.
- Beck showed in 1991 that there exist a deterministic polynomial-time algorithm for a weaker variant of LLL.
- This inspired a long line of research about algorithms for various versions of LLL.
- The breakthrough result of Moser and Tardos (2010) shows that there is a simple randomised resampling algorithm for a very general form of LLL.
- But we are interested in the *distributed* algorithmic LLL.

Distributed computing: the LOCAL model

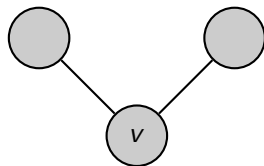


A simple connected undirected graph $G = (V, E)$, where each node $v \in V$

- is given its own input,
- runs the same algorithm,
- communicates with its neighbours,
- produces its own output.

Communication in synchronous rounds

Initially, each node v knows the total number of nodes n , the maximum degree of the graph Δ , and a task-specific local input $f(v)$.

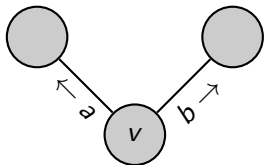


In every round, each node $v \in V$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

Communication in synchronous rounds

Initially, each node v knows the total number of nodes n , the maximum degree of the graph Δ , and a task-specific local input $f(v)$.

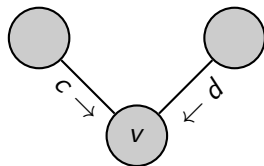


In every round, each node $v \in V$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

Communication in synchronous rounds

Initially, each node v knows the total number of nodes n , the maximum degree of the graph Δ , and a task-specific local input $f(v)$.

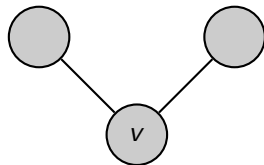


In every round, each node $v \in V$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

Communication in synchronous rounds

Initially, each node v knows the total number of nodes n , the maximum degree of the graph Δ , and a task-specific local input $f(v)$.

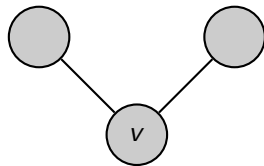


In every round, each node $v \in V$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

Communication in synchronous rounds

Initially, each node v knows the total number of nodes n , the maximum degree of the graph Δ , and a task-specific local input $f(v)$.



In every round, each node $v \in V$

- 1 sends messages to its neighbours,
- 2 receives messages from its neighbours,
- 3 updates its state.

After the final round, each node announces its output.

The running time of an algorithm is the *number of communications rounds* until all nodes have stopped, as a function of n .

Input and output

- The same graph $G = (V, E)$ serves both as the communication network and as the *problem instance*.
- A *graph problem* is defined by a function Π that maps each graph G and each labelling $f: V \rightarrow X$ to a set $\Pi(G, f)$ of *solutions* $S: V \rightarrow Y$.
- The *output* of algorithm A in (G, f) is the function $g: V \rightarrow Y$ such that $g(v)$ is the local output of v for each node v .
- Algorithm A *solves* problem Π if for each graph G and labelling f the output g of A in (G, f) is in $\Pi(G, f)$.

Randomised algorithms

- We assume that each node can toss a countably infinite number of random coins.
- Equivalently, each node v is given a real number $x(v)$ taken uniformly at random from $[0, 1]$.
- With probability 1, the values $x(v)$ are globally unique and can thus be used as identifiers.

Randomised algorithms

- We assume that each node can toss a countably infinite number of random coins.
- Equivalently, each node v is given a real number $x(v)$ taken uniformly at random from $[0, 1]$.
- With probability 1, the values $x(v)$ are globally unique and can thus be used as identifiers.
- Monte Carlo algorithms:
 - Running time is deterministic.
 - Output is a valid solution with high probability (with probability at least $1 - 1/n^c$ for an arbitrarily large constant c).

The distributed Lovász local lemma

- Let $\mathcal{X} = \{X_1, \dots, X_m\}$ be a set of mutually independent random variables and let $\mathcal{E} = \{E_1, \dots, E_n\}$ be a set of events.
- Denote by $\text{vbl}(E_i) \subseteq \mathcal{X}$ the subset of variables that E_i depends on.
- Define a *dependency graph* $G_{\mathcal{E}} = (\mathcal{E}, \mathcal{D})$, where $\mathcal{D} = \{\{E_i, E_j\} : \text{vbl}(E_i) \cap \text{vbl}(E_j) \neq \emptyset\}$.

Problem

Let the communication network be isomorphic to $G_{\mathcal{E}} = (\mathcal{E}, \mathcal{D})$; each node v corresponds to an event $E_v \in \mathcal{E}$ and knows the set $\text{vbl}(E_v)$. The task is to have each node v output an assignment a_v of the variables $\text{vbl}(E_v)$ such that

- 1 for any $\{E_u, E_v\} \in \mathcal{D}$ and $X \in \text{vbl}(E_u) \cap \text{vbl}(E_v)$ it holds that $a_u(X) = a_v(X)$,
- 2 the event E_v does not occur under assignment a_v .

Existing algorithms and lower bounds

- The algorithm of Moser and Tardos (2010) can be adapted to the distributed setting; the running time is $O(\log^2 n)$ rounds.
- Chung et al. (2014) gave a distributed algorithm running in $O(\log n)$ rounds in bounded-degree graphs.
- LLL can be used to properly colour a cycle graph using a constant number of colours. This is known to require $\Omega(\log^* n)$ rounds.

Our lower bound

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{R}$ be such that $f(4) \leq 16$. Let A be a Monte Carlo distributed algorithm for LLL that finds an assignment avoiding all the bad events under the LLL criteria $pf(d) \leq 1$ with high probability. Then the running time of A is $\Omega(\log \log n)$ rounds.

Note that we can plug in, for example, either of the LLL criteria $ep(d+1) \leq 1$ or $4pd \leq 1$.

Outline of the proof

- Two new graph problems: *sinkless orientation* and *sinkless colouring*.
- LLL can be used to solve the sinkless orientation in 3-regular graphs.
- A *mutual speedup lemma*:
 - If we can find a sinkless colouring in t rounds, we can find a sinkless orientation in t rounds.
 - If we can find a sinkless orientation in t rounds, we can find a sinkless colouring in $t - 1$ rounds.
- By iterating the lemma, we obtain an algorithm that finds a sinkless orientation in 0 rounds, which leads to a contradiction.

Orientations

- An *orientation* σ of a graph $G = (V, E)$ assigns a direction

$$\sigma(\{u, v\}) \in \{u \rightarrow v, u \leftarrow v\}$$

for each edge $\{u, v\} \in E$.

- For all $v \in V$ define

$$\text{in-deg}(v, \sigma) = |\{u : (u, v) \in \sigma(E)\}|$$

$$\text{out-deg}(v, \sigma) = |\{u : (v, u) \in \sigma(E)\}|$$

$$\text{deg}(v) = \text{in-deg}(v, \sigma) + \text{out-deg}(v, \sigma).$$

- A node v with $\text{in-deg}(v, \sigma) = \text{deg}(v)$ is called a *sink*. We call an orientation σ *sinkless* if no node is a sink, that is, every node v has $\text{out-deg}(v, \sigma) > 0$.

Sinkless orientation

- A node v with $\text{in-deg}(v, \sigma) = \text{deg}(v)$ is called a *sink*. We call an orientation σ *sinkless* if no node is a sink, that is, every node v has $\text{out-deg}(v, \sigma) > 0$.

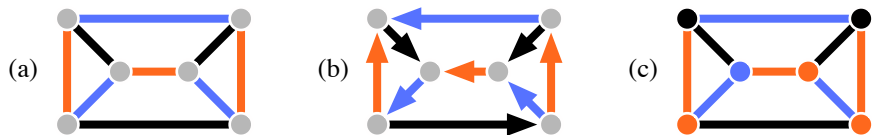


Figure: (a) A 3-regular edge 3-coloured graph. (b) A sinkless orientation. (c) A sinkless colouring.

Colourings

- We write $[k] = \{0, 1, \dots, k - 1\}$.
- $\psi: E \rightarrow [\chi]$ is a *proper edge χ -colouring* if any two adjacent edges have a different colour.
- Given a properly edge χ -coloured graph $G = (V, E, \psi)$, we call $\varphi: V \rightarrow [\chi]$ a *sinkless colouring* of G if for all edges $e = \{u, v\} \in E$ it holds that

$$\varphi(u) = \psi(e) \Rightarrow \varphi(v) \neq \psi(e),$$

that is, if at least one endpoint of each edge has a different colour than the edge.

Sinkless colouring

- Given a properly edge χ -coloured graph $G = (V, E, \psi)$, we call $\varphi: V \rightarrow [\chi]$ a *sinkless colouring* of G if for all edges $e = \{u, v\} \in E$ it holds that

$$\varphi(u) = \psi(e) \Rightarrow \varphi(v) \neq \psi(e),$$

that is, if at least one endpoint of each edge has a different colour than the edge.

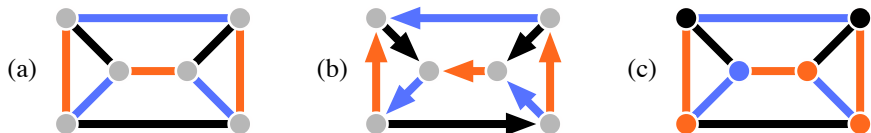


Figure: (a) A 3-regular edge 3-coloured graph. (b) A sinkless orientation. (c) A sinkless colouring.

Graph problem definitions

Problem (Sinkless colouring)

Given an edge d -coloured d -regular graph $G = (V, E, \psi)$, find a sinkless colouring φ . That is, compute a colouring φ such that for no edge $e = \{u, v\} \in E$ we have $\varphi(u) = \varphi(v) = \psi(e)$.

Problem (Sinkless orientation)

Given an edge d -coloured d -regular graph $G = (V, E, \psi)$, find a sinkless orientation. That is, compute an orientation σ such that $\text{out-deg}(v, \sigma) > 0$ for all $v \in V$.

Relationship between the graph problems

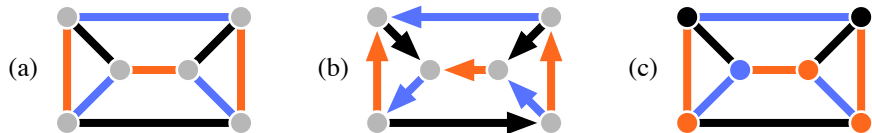


Figure: (a) A 3-regular edge 3-coloured graph. (b) A sinkless orientation. (c) A sinkless colouring.

- From a sinkless orientation we get a sinkless colouring in 0 rounds.
- From a sinkless colouring we get a sinkless orientation in 1 rounds.

From LLL to sinkless orientation

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{R}$ be such that $f(4) \leq 16$. Let A be a Monte Carlo distributed algorithm for LLL such that A finds an assignment avoiding all the bad events under the LLL criteria $pf(d) \leq 1$ in time T for some $T: \mathbb{N} \rightarrow \mathbb{N}$. Then there is a Monte Carlo distributed algorithm B that finds a sinkless orientation in 3-regular graphs of girth at least 5 in time $O(T)$.

From LLL to sinkless orientation

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{R}$ be such that $f(4) \leq 16$. Let A be a Monte Carlo distributed algorithm for LLL such that A finds an assignment avoiding all the bad events under the LLL criteria $pf(d) \leq 1$ in time T for some $T: \mathbb{N} \rightarrow \mathbb{N}$. Then there is a Monte Carlo distributed algorithm B that finds a sinkless orientation in 3-regular graphs of girth at least 5 in time $O(T)$.

- We start with 4-regular graphs $G = (V, E)$.
- Set $\text{vbl}(E_v) = \{X_e : v \in e\}$ for each $v \in V$
- For each $e = \{u, v\} \in E$, the variable X_e ranges over $\{u \rightarrow v, u \leftarrow v\}$
- The bad event E_v occurs exactly when for all neighbours u of v the variable $X_{\{v,u\}}$ takes the value $u \rightarrow v$

From LLL to sinkless orientation

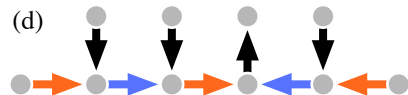
- If the variables X_e are sampled uniformly at random, we have $\Pr(E_v) = 1/2^4 = 1/16$ for each $v \in V$.
- Let $p = 1/16$ and $d = 4$. Now $\Pr(E_v) \leq p$ and E_v depends on d other events for each $v \in V$, and the condition $pf(d) \leq 1$ holds, given $f(4) \leq 16$.
- Run the algorithm A and define an orientation σ of G by setting $\sigma(e) = a_v(X_e)$, where $v \in e$, for each $e \in E$.
- Now σ is a sinkless orientation of G .

From 3-regular to 4-regular graphs

- LLL is not directly applicable: the probability of bad events would be $p = 1/2^3 = 1/8$ and thus $ep(d + 1) > 1$.
- *Contract* edges of one colour class to obtain a 4-regular graph.
- Simulate the algorithm for the 4-regular case in the 3-regular graph.

From 3-regular to 4-regular graphs

- LLL is not directly applicable: the probability of bad events would be $p = 1/2^3 = 1/8$ and thus $ep(d + 1) > 1$.
- *Contract* edges of one colour class to obtain a 4-regular graph.
- Simulate the algorithm for the 4-regular case in the 3-regular graph.



The mutual speedup lemma

Lemma

Suppose B is a sinkless colouring algorithm that runs in t rounds such that for any edge $e = \{u, v\}$ the probability of outputting a forbidden configuration $B(u) = \psi(e) = B(v)$ is at most p . Then there exists a sinkless orientation algorithm B' that runs in t rounds such that for any node u the probability of being a sink is at most $6p^{1/3}$.

The mutual speedup lemma

Lemma

Suppose B is a sinkless colouring algorithm that runs in t rounds such that for any edge $e = \{u, v\}$ the probability of outputting a forbidden configuration $B(u) = \psi(e) = B(v)$ is at most p . Then there exists a sinkless orientation algorithm B' that runs in t rounds such that for any node u the probability of being a sink is at most $6p^{1/3}$.

Lemma

Suppose B' is a sinkless orientation algorithm that runs in time t such that the probability that any node u is a sink is at most ℓ . Then there exists a sinkless colouring algorithm B'' that runs in time $t - 1$ such that the probability for any edge $e = \{u, v\}$ having a forbidden configuration $B''(u) = \psi(e) = B''(v)$ is less than $4\ell^{1/4}$.

From colouring to orientation: the proof idea

- Given a randomised sinkless *colouring* algorithm B running in t rounds, construct a randomised sinkless *orientation* algorithm B' that also runs in t rounds.
- We write $B(u)$ for the colour that u outputs according to B and $B'(e)$ for the orientation B' outputs for edge e .
- We denote the radius- t neighbourhood of a node u by

$$N^t(u) = \{v \in V : \text{dist}(u, v) \leq t\},$$

where $\text{dist}(u, v)$ is the length of the shortest path between u and v .

- The radius- t neighbourhood of an *edge* $\{u, v\}$ is

$$N^t(\{u, v\}) = N^t(u) \cap N^t(v).$$

From colouring to orientation: the proof idea

Consider any node $u \in V$. Algorithm B' consists of three steps:

- 1 Node u gathers its radius- t neighbourhood $N^t(u)$ in t rounds.
- 2 Node u computes the set $C(u)$ of *candidate colours*:

$$C(u) = \left\{ \psi(e) : \Pr[B(u) = \psi(e) \mid N^t(e)] \geq p^{1/3} \text{ and } e = \{u, v\} \right\},$$

In addition, for each $e = \{u, v\}$ node u calculates the probability of v outputting the colour $\psi(e)$ when executing B . Thus u can determine whether $\psi(e) \in C(v)$.

From colouring to orientation: the proof idea

Consider any node $u \in V$. Algorithm B' consists of three steps:

- 1 Node u gathers its radius- t neighbourhood $N^t(u)$ in t rounds.
- 2 Node u computes the set $C(u)$ of *candidate colours*:

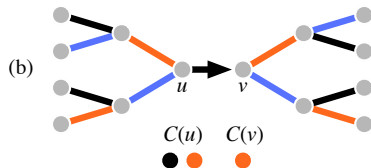
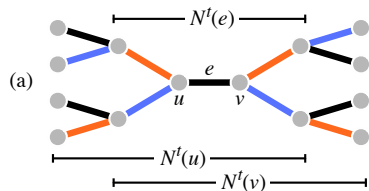
$$C(u) = \left\{ \psi(e) : \Pr[B(u) = \psi(e) \mid N^t(e)] \geq p^{1/3} \text{ and } e = \{u, v\} \right\},$$

In addition, for each $e = \{u, v\}$ node u calculates the probability of v outputting the colour $\psi(e)$ when executing B . Thus u can determine whether $\psi(e) \in C(v)$.

- 3 If $\psi(e) \in C(u) \cap C(v)$ or $\psi(e) \notin C(u) \cup C(v)$, choose the orientation $B'(e)$ of edge e arbitrarily. Otherwise, edge e is oriented according to the following rule:

$$B'(e) = \begin{cases} u \rightarrow v & \text{if } \psi(e) \in C(u) \text{ and } \psi(e) \notin C(v), \\ u \leftarrow v & \text{if } \psi(e) \notin C(u) \text{ and } \psi(e) \in C(v). \end{cases}$$

From colouring to orientation: the proof idea



- Here the running time $t = 2$.
- In algorithm B , the colour of node u is determined by the random bits in $N^t(u)$.
- Black is a candidate colour of u if, based on the information in $N^t(e)$, the probability of u outputting black in algorithm B is at least $p^{1/3}$.
- If black is one of the candidate colours of u , and it is *not* one of the candidate colours of v , then algorithm B' will orient the edge $u \rightarrow v$.

Conclusion

- There is a randomised distributed algorithm for LLL that runs in $O(\log n)$ rounds in bounded-degree graphs.
- The best previously known lower bound was $\Omega(\log^* n)$ rounds.
- We show that any randomised Monte Carlo algorithm for LLL that finds a satisfying assignment with high probability requires $\Omega(\log \log n)$ rounds.