An Algorithmic Proof of the Lopsided Lovasz Local Lemma

Nick Harvey
University of British Columbia

Jan Vondrak
IBM Almaden
Lovasz Local Lemma

- Let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ be events in a discrete probability space.
- Let $G$ be a graph on $\{1, \ldots, n\}$. Let $\Gamma(i)$ be neighbors of $i$, and let $\Gamma^+(i) = \Gamma(i) \cup \{i\}$.
- $G$ is called a dependency graph if, for all $i$, $\mathcal{E}_i$ is jointly independent of $\{ \mathcal{E}_j : j \notin \Gamma^+(i) \}$.

E.g., Pairwise Independent Events

- Erdos-Lovasz ‘75: Suppose $|\Gamma^+(i)| \leq d$ and $\mathbb{P}[\mathcal{E}_i] \leq p \ \forall i$. If $p \cdot d \leq 1$ then $\mathbb{P}[\bigcap_i \mathcal{E}_i^c] \geq (1-1/d)^n$. 
Example: k-SAT

- Let $\phi$ be a k-SAT formula.
- Pick a uniformly random assignment to the variables.
- Let $\mathcal{E}_i$ be the event clause $i$ is unsatisfied. Note $\mathbb{P}[\mathcal{E}_i] = 2^{-k} = p$.
- Suppose that each variable appears in $2^{k/e}k$ clauses.
- Dependency graph: $|\Gamma^+(i)| \leq 2^{k/e} = d$.
- Example: $\phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (x_4 \lor x_5 \lor x_6) \land (x_2 \lor x_3 \lor x_6)$

- Since $ped \leq 1$, LLL implies $\mathbb{P}[\bigcap_i \mathcal{E}_i^c] > 0$, so $\phi$ is satisfiable.
Stronger forms of LLL

• **Symmetric LLL:**  $\mathbb{P}[\mathcal{E}_i] \leq 1/(e \cdot \max_j |\Gamma^+(j)|) \ \forall i.$

• **Asymmetric LLL:**  $\sum_{j \in \Gamma^+(i)} \mathbb{P}[\mathcal{E}_j] \leq 1/4 \ \forall i.$

• **General LLL:**  $\exists x_1,\ldots,x_n \in (0,1)$ such that
  $\mathbb{P}[\mathcal{E}_i] \leq x_i \cdot \prod_{j \in \Gamma(i)} (1-x_j) \ \forall i.$
Stronger forms of LLL

- **Symmetric LLL:** \( P[\mathcal{E}_i] \leq 1/(e \cdot \max_j |\Gamma^+(j)|) \) \( \forall i \).

- **Asymmetric LLL:** \( \sum_{j \in \Gamma^+(i)} P[\mathcal{E}_j] \leq 1/4 \) \( \forall i \).

- **General LLL:** \( \exists y_1,\ldots,y_n > 0 \) such that
  \[
P[\mathcal{E}_i] \leq y_i / \prod_{j \in \Gamma^+(i)} (1 + y_j) \quad \forall i.
  \]
Stronger forms of LLL

• Symmetric LLL: \( P[E_i] \leq 1/(e \cdot \max_j |\Gamma^+(j)|) \ \forall i. \)

• Asymmetric LLL: \( \sum_{j \in \Gamma^+(i)} P[E_j] \leq 1/4 \ \forall i. \)

• General LLL: \( \exists y_1,\ldots,y_n > 0 \) such that
  \[ P[E_i] \leq y_i / \sum_{J \subseteq \Gamma^+(i)} y^J \ \forall i. \]
  where \( y^J = \prod_{j \in J} y_j. \)

• Let \( \text{Ind} \) be the independent sets of the dependency graph.

• Cluster Expansion: \( \exists y_1,\ldots,y_n > 0 \) such that
  \[ P[E_i] \leq y_i / \sum_{J \subseteq \Gamma^+(i), \bar{J} \in \text{Ind}} y^J \ \forall i. \]
Stronger forms of LLL

• Let \( \text{Ind} \) be the independent sets of the dependency graph.
• Let \( p_i = P[\mathcal{E}_i] \).
• For \( S \subseteq V \), let \( q_S = q_S(p) = \sum_{I \supseteq S, I \in \text{Ind}} (-1)^{|I| - |S|} p^I \). "multivariate independence polynomial"

• Shearer ‘85: If \( q_I > 0 \ \forall I \in \text{Ind} \) then \( P[\bigcap_i \mathcal{E}_i^c] \geq q_\emptyset > 0 \). Moreover, for every dependency graph, this is the optimal criterion under which the LLL is true.
Algorithmic Results

• LLL gives a non-constructive proof that $\cap_i E_i^c \neq \emptyset$.

• Can we efficiently find a point in that set?

• Long history:
  Beck ‘91, Alon ‘91, Molloy-Reed ‘98, Czumaj-Scheideler ‘00, Srinivasan ‘08

• Can find a satisfying assignment for k-SAT instances where each variable appears in $\leq 2^{k/7}$ clauses.
Algorithmic Results

• Efficiently finding a point in $\bigcap_i E_i^c$.

• Moser-Tardos ‘10: Algorithm for General LLL in “variable model”.
  — Probability space has product distribution on variables.
  — Each events is a function of some subset of the variables.
  — Dependency graph connects events that share variables.

• Many extensions: (but still in variable model)
  — Cluster Expansion criterion [Pegden ‘13]
  — Shearer’s criterion [Kolipaka-Szegedy ‘11]
Algorithmic Results

- Efficiently finding a point in $\bigcap_i E_i^c$.

- Moser-Tardos ‘10: Algorithm for General LLL in “variable model”.
  - Probability space has product distribution on variables.
  - Each event is a function of some subset of the variables.
  - Dependency graph connects events that share variables.

- Many extensions in other directions
  - Derandomization [Chandrasekaran et al. ‘10]
  - Exponentially many events [Haeupler, Saha, Srinivasan ‘10]
  - Column-sparse IPs [Harris, Srinivasan ‘13]
  - Distributed algorithms [Chung, Pettie, Su ‘14]
Extending LLL Beyond Dependency Graphs

• The definition of a dependency graph is equivalent to
  \[ P[\mathcal{E}_i \mid \bigcap_{j \in J} \mathcal{E}_j^c] = P[\mathcal{E}_i] \quad \forall J \subseteq [n]\setminus \Gamma^+(i). \]

• Define a lopsidependency graph to be one satisfying
  \[ P[\mathcal{E}_i \mid \bigcap_{j \in J} \mathcal{E}_j^c] \leq P[\mathcal{E}_i] \quad \forall J \subseteq [n]\setminus \Gamma^+(i). \]

• Intuitively, edges between events that are “negatively dependent”

• (Easy) Theorem: (Erdos-Spencer ‘91) All existential forms of the LLL remain true with lopsidependency graphs.
Extending LLL Beyond Dependency Graphs

- The definition of a dependency graph is equivalent to

\[ P[\mathcal{E}_i \mid \bigcap_{j \in J} \mathcal{E}_j^c] = P[\mathcal{E}_i] \quad \forall J \subseteq [n] \setminus \Gamma^+(i). \]

- Define a lopsidependency graph to be one satisfying

\[ P[\mathcal{E}_i \mid \bigcap_{j \in J} \mathcal{E}_j^c] \leq P[\mathcal{E}_i] \quad \forall J \subseteq [n] \setminus \Gamma^+(i). \]

- Intuitively, edges between events that are “negatively dependent”

- Eg: Let \( \pi \) be a permutation on \{1,2\}.
  Let \( \mathcal{E}_1 \) be “\( \pi(1)=1 \)” and \( \mathcal{E}_2 \) be “\( \pi(2)=2 \)”.
  Then \( P[\mathcal{E}_1 \mid \mathcal{E}_2^c] = 0 < 1/2 = P[\mathcal{E}_1] \).

\( \mathcal{E}_1 \) and \( \mathcal{E}_2 \) are “positively dependent”
Extending LLL Beyond Dependency Graphs

• The definition of a dependency graph is equivalent to
  \[ \mathbb{P}[\mathcal{E}_i \mid \bigcap_{j \in J} \mathcal{E}_j^c] = \mathbb{P}[\mathcal{E}_i] \quad \forall J \subseteq [n] \backslash \Gamma^+(i). \]

• Define a lopsidependency graph to be one satisfying
  \[ \mathbb{P}[\mathcal{E}_i \mid \bigcap_{j \in J} \mathcal{E}_j^c] \leq \mathbb{P}[\mathcal{E}_i] \quad \forall J \subseteq [n] \backslash \Gamma^+(i). \]

• Intuitively, edges between events that are “negatively dependent”

• Eg: Let \( \pi \) be permutation on \([n]\). [Erdos-Spencer ‘91, Harris-Srinivasan ‘14] Let \( \mathcal{E}_i \) be “\( \pi(a_i) = b_i \)”.
  Add edges between \( \mathcal{E}_i \) and \( \mathcal{E}_j \) if \( a_i = a_j \) or \( b_i = b_j \).
Lopsided Association

- Define a lopsidepency graph to be one satisfying
  \[ P[E_i \mid \cap_{j \in J} E_j^c] \leq P[E_i] \quad \forall J \subseteq [n] \setminus \Gamma^+(i). \]

- **Definition**: A lopsided association graph is a graph where
  \[ P[E_i \mid F] \geq P[E_i] \quad \forall F \in \mathcal{F}_i, \]
  where \( \mathcal{F}_i \) contains all monotone functions of \( \{ E_i : j \notin \Gamma^+(i) \} \).

- **Easy**: Lopsided association \( \Rightarrow \) Lopsidepency.

- Analogous to “negative association” vs “negative cylinder dependent”
Lopsided Association

- Define a lopsidependency graph to be one satisfying
  \[ P[E_i | \cap_{j \in J} E_j^c] \leq P[E_i] \quad \forall J \subseteq [n] \setminus \Gamma^+(i). \]

- Definition: A lopsided association graph is a graph where
  \[ P[E_i | F] \geq P[E_i] \quad \forall F \in \mathcal{F}_i, \]
  where \( \mathcal{F}_i \) contains all monotone functions of \( \{ E_i : j \not\in \Gamma^+(i) \} \).

- Easy: Lopsided association \( \Rightarrow \) Lopsidependency.

- Main Result: For every probability space with a lopsided association graph, all existential forms of LLL have an algorithmic proof.
Algorithmic LLL

- Other Distributions
- Variable model
- Dependency graph
- Lopsided association graph
- Lopsidependency graph

- Symmetric
- General
- Graph Expansion

- Moser-Tardos
- Pegden
- Kolipaka-Szegedy
Algorithmic LLL beyond variable model
Algorithmic LLL beyond variable model

- Spanning Trees in $K_n$?
- Perfect Matchings in $K_n$
- Permutations
- Variable model
- Dependency graph
- Lopsided association graph
- Lopsidependency graph

- Achlioptas-Iliopoulos
- Moser-Tardos
- Pegden
- Kolipaka-Szegedy

* Needs slack
+ Hamilton cycles in $K_n$...
Algorithmic LLL beyond variable model

Spanning Trees in $K_n$?

Perfect Matchings in $K_n$

Permutations

Variable model

Dependency graph

Lopsided association graph

Lopsidependency graph

* Needs slack

+ Hamilton. Cycles...
Algorithmic LLL beyond variable model

* Intractability?
All probability spaces

Spanning Trees in $K_n$

Perfect Matchings in $K_n$

Permutations
Variable model

Dependency graph
Lopsided association graph
Lopsidependency graph

Our Results

Symmetric
General
Cluster Expansion
Shearer

* Need slack in Shearer's criterion
Resampling Oracles

• Need some algorithmic access to probability space $\mu$

• Def: A resampling oracle for $\mathcal{E}_i$ is $r_i : \Omega \rightarrow \Omega$ satisfying
  
  — (R1) If $\omega \sim \mu \mid \mathcal{E}_i$ then $r_i(\omega) \sim \mu$

          Removes conditioning on $\mathcal{E}_i$

  — (R2) If $j \not\in \Gamma^+(i)$ and $\mathcal{E}_j$ does not occur in $\omega$,
          then $\mathcal{E}_j$ does not occur in $r_i(\omega)$.

          Cannot cause non-neighbors to occur

• Roughly, $r_i$ implements a coupling between $\mu \mid \mathcal{E}_i$ and $\mu$
    such that (R2) holds.
Resampling Oracles

- Need some algorithmic access to probability space $\mu$

- **Def:** A resampling oracle for $E_i$ is $r_i : \Omega \rightarrow \Omega$ satisfying
  - (R1) If $\omega \sim \mu | E_i$ then $r_i(\omega) \sim \mu$
  - (R2) If $j \not\in \Gamma^+(i)$ and $E_j$ does not occur in $\omega$, then $E_j$ does not occur in $r_i(\omega)$.

- **Eg:** Moser-Tardos Variable model

\[
\phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (x_4 \lor x_5 \lor x_6) \land (x_2 \lor x_3 \lor x_6)
\]

- $r_i(\omega)$: resample all variables in clause $i$. 

\[
\begin{align*}
E_1 & \quad \text{Clause 1 unsatisfied} \\
E_2 & \quad \text{Clause 2 unsatisfied} \\
E_4 & \quad \text{Clause 4 unsatisfied} \\
E_3 & \quad \text{Clause 3 unsatisfied}
\end{align*}
\]
Resampling Oracles

• Need some algorithmic access to probability space $\mu$

• **Def:** A resampling oracle for $E_i$ is $r_i : \Omega \rightarrow \Omega$ satisfying
  
  – (R1) If $\omega \sim \mu | E_i$ then $r_i(\omega) \sim \mu$
  
  – (R2) If $j \notin \Gamma^+(i)$ and $E_j$ does not occur in $\omega$, then $E_j$ does not occur in $r_i(\omega)$.

• **Theorem:** For any probability space and any events, TFAE
  
  – Existence of a lopsided association graph
  
  – Existence of resampling oracles.

• No guarantees it is compactly representable or efficient.

What if events involve some NP-hard problem?
Resampling Oracles

• **Def:** A resampling oracle for $\mathcal{E}_i$ is $r_i : \Omega \rightarrow \Omega$ satisfying
  
  – (R1) If $\omega \sim \mu \mid \mathcal{E}_i$ then $r_i(\omega) \sim \mu$
  
  – (R2) If $j \notin \Gamma^+(i)$ and $\mathcal{E}_j$ does not occur in $\omega$, then $\mathcal{E}_j$ does not occur in $r_i(\omega)$.

• **Theorem:** For any probability space and any events, TFAE
  
  – Existence of a lopsided association graph
  
  – Existence of resampling oracles.

• No guarantees it is compactly representable or efficient. What if events involve some NP-hard problem?
Spanning Trees in $K_n$

• Let $T$ be a uniformly random spanning tree in $K_n$.

• For edge set $A$, let $E_A = \{A \subseteq T\}$.

• Dependency Graph: Make $E_A$ a neighbor of $E_B$, unless $A$ and $B$ are vertex-disjoint.

• Resampling oracle $r_A$:
  — If $T$ uniform conditioned on $E_A$, want $r_A(T)$ uniform.
  — But, should not disturb edges that are vtx-disjoint from $A$. 
• \( \mathcal{E}_A = \{ A \subseteq T \} \).

• **Resampling oracle** \( r_A(T) \):
  
  — If \( T \) uniform conditioned on \( \mathcal{E}_A \), want \( r_A(T) \) uniform.
  
  — But, should not disturb edges that are vtx-disjoint from \( A \).

  — Contract edges of \( T \) vtx-disjoint from \( A \)
  
  — Delete remaining edges vtx-disjoint from \( A \)
  
  — Let \( r_A(T) \) be a uniformly random spanning tree in resulting graph.
Main Theorem

• Given
  — any probability space and events
  — a lopsided association graph
  — resampling oracles for the events
• Suppose that General LLL, Cluster Expansion or Shearer* holds.
• There is an algorithm to find a point in $\cap_i \mathcal{E}_i^c$, running in oracle-polynomial† time, whp.

* Need slack for Shearer’s condition.
† Actually depending on parameters in General LLL, etc.
Main Theorem

- Given
  - any probability space and events
  - a lopsided association graph
  - resampling oracles for the events

- **General LLL:** \( \exists x_1, \ldots, x_n \in (0,1) \) such that
  \[
  \mathbb{P}[\mathcal{E}_i] \leq x_i \cdot \prod_{j \in \Gamma(i)} (1-x_j) \quad \forall i.
  \]

- There is an algorithm to find a point in \( \bigcap_i \mathcal{E}_i^c \), using
  \( O\left(\sum_i \frac{x_i}{1-x_i} \sum_i \log \frac{x_i}{1-x_i}\right) \) oracle calls, whp.
Main Theorem

• Given
  — any probability space and events
  — a lopsided association graph
  — resampling oracles for the events

• Cluster Expansion: \( \exists y_1, \ldots, y_n > 0 \) such that
  \[
  \mathbb{P} [ \mathcal{E}_i ] \leq y_i / \sum_{J \subseteq \Gamma^+(i)} y^J \quad \forall i.
  \]

• There is an algorithm to find a point in \( \cap_i \mathcal{E}_i^c \),
  using \( O(\sum_i y_i \sum_i \log(1 + y_i)) \) oracle calls, whp.
The Algorithm

MaximalSetResample

Draw $\omega$ from $\mu$

Repeat

$J \leftarrow \emptyset$

While there is $i \not\in \Gamma^+(J)$ s.t. $E_i$ occurs in $\omega$

Pick smallest such $i$

$J \leftarrow J \cup \{i\}$

$\omega \leftarrow r_i(\omega)$

End

Until $J = \emptyset$

• Similar to parallel form of Moser-Tardos
Analysis
Review of Moser-Tardos

“Log” or “History” of resampling operations

Dependency graph edges

Start

\[ \mathcal{E}_3 \mathcal{E}_9 \mathcal{E}_2 \mathcal{E}_5 \mathcal{E}_1 \mathcal{E}_7 \mathcal{E}_1 \mathcal{E}_3 \mathcal{E}_9 \ldots \]
Analysis
Review of Moser-Tardos

“Log” or “History” of resampling operations

Dependency graph edges

Start

Witness tree

\[ \mathcal{E}_3, \mathcal{E}_9, \mathcal{E}_2, \mathcal{E}_5, \mathcal{E}_1, \mathcal{E}_7, \mathcal{E}_1, \mathcal{E}_3, \mathcal{E}_9, \ldots \]
MaximalSetResample

Draw $\omega$ from $\mu$
Repeat
  $J \leftarrow \emptyset$
  While there is $i \notin \Gamma^+(J)$ s.t. $E_i$ occurs in $\omega$
    Pick smallest such $i$
    $J \leftarrow J \cup \{i\}$
    $\omega \leftarrow r_i(\omega)$
  End
Until $J = \emptyset$

Resampling sequence: $J_1, J_2, J_3, \ldots$

$J_i \in \text{Ind}, \quad J_{i+1} \subseteq \Gamma^+(J_i)$
• Trivially, \( E[\text{\# iterations}] = \sum_{I_1, I_2, \ldots} \Pr[I_1, I_2, \ldots \text{ is a prefix of resampling sequence}] \)

• **Claim:** \( \Pr[I_1, \ldots, I_k \text{ is a prefix}] \leq \prod_{1 \leq i \leq k} p^{I_i} \)

• Like Moser-Tardos, this is a coupling argument. But instead of using fact that variable have “fresh samples” whenever examined, we use the fact that \( r_i \) returns the state to the underlying distribution \( \mu \).
• Trivially, $E[\text{ # iterations } ] = 
\sum_{I_1,I_2,\ldots} \Pr[ I_1,I_2,\ldots \text{ is a prefix of resampling sequence } ]$

• Claim: $\Pr[ I_1,\ldots,I_k \text{ is a prefix } ] \leq \prod_{1\leq i\leq k} p^{I_i}$

• Proof: Like Moser-Tardos, using $r_i$ returns state to $\mu$.

• Claim: Let $\mathbb{P}[\mathcal{E}_i] \leq (1-\epsilon) x_i \cdot \prod_{j\in \Gamma(i)} (1-x_j)$ (General LLL)
Then $\sum_{I_1,\ldots,I_k} \prod_{1\leq i\leq k} p^{I_i} \leq O\left(\frac{1}{\epsilon} \sum_i \log \frac{x_i}{1-x_i}\right)$

• Proof: Key idea: inductively show that
$\sum_{I_1,\ldots,I_k} \prod_{1\leq i\leq k} p^{I_i} \leq \prod_{j\in J} \frac{x_j}{1-x_j}$
Comparison to Fotis’ work

• Pros for Fotis:
  — Don’t need to sample from $\mu$ (there is no $\mu$)
  — Can handle scenarios that are not even a lopsidedependency graph, e.g., Hamilton cycles, Vtx Coloring.

• Pros for us:
  — Characterize when resampling operations exist
  — Gives new proof of General LLL with Dependency Graphs, in full generality. (Even for Shearer & Lopsided Association Graphs).
  — Don’t need any slack in General LLL or Cluster Expansion conditions
  — “Fractional transitions” seem easier in our setting