Alternative proofs of the asymmetric Lovász local lemma and Shearer's lemma

I. Giotis[1] L. Kirousis[1] J. Livieratos[1]
K. Psaromiligkos[2] D. Thilikos[1]

[1]Department of Mathematics, National and Kapodistrian University of Athens
[2]Department of Mathematics, University of Chicago

GASCom, 2018

Outline

Outline

The variable framework

- $X_1, \ldots, X_l$ mutually independent random variables.

The variable framework

- $X_1, \dots, X_l$ mutually independent random variables.
- Ω probability space of all assignments.

The variable framework

- $X_1, \ldots, X_l$ mutually independent random variables.
- Ω probability space of all assignments.
- $E_1, \ldots, E_m \subseteq \Omega$ "undesirable" events.

The variable framework

- $X_1, \ldots, X_l$ mutually independent random variables.
- Ω probability space of all assignments.
- $E_1, \ldots, E_m \subseteq \Omega$ "undesirable" events.
- $\text{sc}(E_j) \subseteq \{X_1, \ldots, X_l\}$ scope of E_j.

The variable framework

- $X_1, \ldots, X_l$ mutually independent random variables.
- Ω probability space of all assignments.
- $E_1, \ldots, E_m \subseteq \Omega$ "undesirable" events.
- $\mathrm{sc}(E_j) \subseteq \{X_1, \ldots, X_l\}$ scope of E_j.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

The variable framework

- $X_1, \ldots, X_l$ mutually independent random variables.
- Ω probability space of all assignments.
- $E_1, \ldots, E_m \subseteq \Omega$ "undesirable" events.
- $\mathrm{sc}(E_j) \subseteq \{X_1, \ldots, X_l\}$ scope of E_j.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$x_i \leftrightarrow X_i : \{0,1\}^6 \mapsto \{0,1\}$

The variable framework

- $X_1, \ldots, X_l$ mutually independent random variables.
- Ω probability space of all assignments.
- $E_1, \ldots, E_m \subseteq \Omega$ "undesirable" events.
- $\text{sc}(E_j) \subseteq \{X_1, \ldots, X_l\}$ scope of E_j.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$x_i \leftrightarrow X_i : \{0,1\}^6 \mapsto \{0,1\}$
$E_i \leftrightarrow C_j$ is unsatisfied.

Algorithmic idea (Moser, 2009)

Algorithmic idea (Moser, 2009)

- Sample the random variables.

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$(0, 1, 0, 0, 0, 0): \; E_2, \overline{E}_1, \overline{E}_3, \overline{E}_4$

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$(0, 0, 0, 0, 0, 0)$: $E_1, E_3, \overline{E}_2, \overline{E}_4$

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$(0, 0, 1, 0, 0, 0) : E_3, \overline{E}_1, \overline{E}_2, \overline{E}_4$

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$(0, 0, 1, 0, 1, 0): \ E_4, \overline{E}_1, \overline{E}_2, \overline{E}_3$

Algorithmic idea (Moser, 2009)

- Sample the random variables.
- Resample them until no undesirable event occurs.

- Select an occurring event and resample the variables in its scope.
- Check the events affected by this resampling.

SAT

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$(0, 0, 1, 0, 1, 1)$: satisfying assignment.

Dependency graph

$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset,$

Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1, \ldots, m\}, \{\{i, j\} \mid E_i \sim E_j\}\Big)$$

Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1,\ldots,m\}, \{\{i,j\} \mid E_i \sim E_j\}\Big)$$

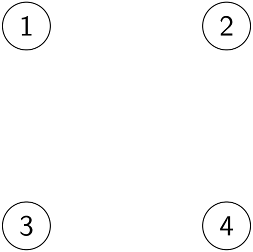
$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

.

Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1, \ldots, m\}, \{\{i,j\} \mid E_i \sim E_j\} \Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

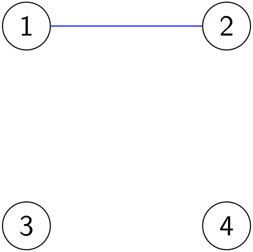


Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1, \dots, m\}, \{\{i, j\} \mid E_i \sim E_j\}\Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$E_1 \sim E_2$,

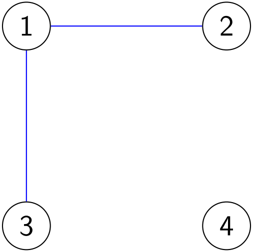


Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1,\dots,m\}, \{\{i,j\} \mid E_i \sim E_j\}\Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$E_1 \sim E_2, E_1 \sim E_3,$.

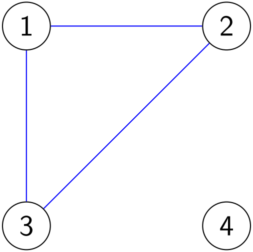


Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1, \ldots, m\}, \{\{i, j\} \mid E_i \sim E_j\}\Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$E_1 \sim E_2, E_1 \sim E_3, E_2 \sim E_3,$.

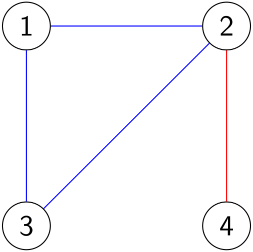


Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset,\ \ G = \Big(\{1,\dots,m\}, \{\{i,j\} \mid E_i \sim E_j\}\Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$E_1 \sim E_2, E_1 \sim E_3, E_2 \sim E_3, E_2 \sim E_4,$.

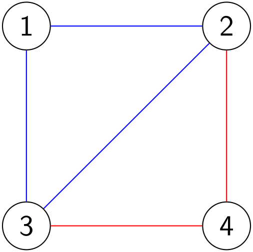


Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1, \ldots, m\}, \{\{i, j\} \mid E_i \sim E_j\}\Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$E_1 \sim E_2, E_1 \sim E_3, E_2 \sim E_3, E_2 \sim E_4, E_3 \sim E_4.$

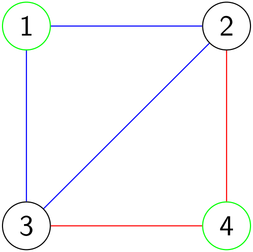


Dependency graph

$$E_i \sim E_j \Leftrightarrow \mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset, \;\; G = \Big(\{1, \ldots, m\}, \{\{i, j\} \mid E_i \sim E_j\}\Big)$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$E_1 \sim E_2, E_1 \sim E_3, E_2 \sim E_3, E_2 \sim E_4, E_3 \sim E_4$.



$\{1, 4\}$: independent set.

Outline

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,
- d is the maximum degree of the dependency graph.

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,
- d is the maximum degree of the dependency graph.

Theorem (Lovász Local Lemma (symmetric))

If $ep(d+1) \leq 1$ *then:*

$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,
- d is the maximum degree of the dependency graph.

Theorem (Lovász Local Lemma (symmetric))

If $ep(d+1) \leq 1$ *then:*

$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Erdős and Lovász (1975): original statement and proof of LLL.

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,
- d is the maximum degree of the dependency graph.

Theorem (Lovász Local Lemma (symmetric))

If $ep(d+1) \leq 1$ *then:*

$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Erdős and Lovász (1975): original statement and proof of LLL.
- Erdős and Spencer (1991): statement and proof of the *lopsided* LLL.

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,
- d is the maximum degree of the dependency graph.

Theorem (Lovász Local Lemma (symmetric))

If $ep(d + 1) \leq 1$ *then:*

$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Erdős and Lovász (1975): original statement and proof of LLL.
- Erdős and Spencer (1991): statement and proof of the *lopsided* LLL.

- $\phi(x_1, \ldots, x_n)$ k-SAT formula and $Pr[x_i = 0] = \frac{1}{2}$.

Simple, symmetric LLL

$p \in [0, 1)$, $d \in \mathbb{Z}_{\geq 0}$.

- $Pr[E_j] \leq p$, $j = 1, \ldots, m$,
- d is the maximum degree of the dependency graph.

Theorem (Lovász Local Lemma (symmetric))

If $ep(d+1) \leq 1$ *then:*

$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Erdős and Lovász (1975): original statement and proof of LLL.
- Erdős and Spencer (1991): statement and proof of the *lopsided* LLL.

- $\phi(x_1, \ldots, x_n)$ k-SAT formula and $Pr[x_i = 0] = \frac{1}{2}$.
- If $d \leq \lfloor \frac{2^k}{e} - 1 \rfloor$, then $\phi(x_1, \ldots, x_n)$ satisfiable.

Asymmetric LLL

- $N_j = \{i \mid E_j \sim E_i\}$.

Asymmetric LLL

- $N_j = \{i \mid E_j \sim E_i\}$.
- $E_j \leftrightarrow \chi_j \in (0,1)$:

Asymmetric LLL

- $N_j = \{i \mid E_j \sim E_i\}$.
- $E_j \leftrightarrow \chi_j \in (0,1)$:

$$Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$

for all $j \in \{1, \ldots, m\}$, then:

Asymmetric LLL

- $N_j = \{i \mid E_j \sim E_i\}$.
- $E_j \leftrightarrow \chi_j \in (0,1)$:

$$Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$

for all $j \in \{1, \ldots, m\}$, then:

$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

Asymmetric LLL

- $N_j = \{i \mid E_j \sim E_i\}$.
- $E_j \leftrightarrow \chi_j \in (0,1)$:
$$Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$
for all $j \in \{1, \ldots, m\}$, then:
$$Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

▶ Moser and Tardos (2009-10): algorithmic proof of the symmetric (simple or lopsided) LLL.

Shearer's lemma, 1985

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.

Shearer's lemma, 1985

$G = (\{1, \dots, m\}, E)$ and $\bar{p} = (p_1, \dots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \dots, I_s\}$ the set of independent sets of G.

Shearer's lemma, 1985

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

Shearer's lemma, 1985

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

Shearer's lemma, 1985

$G = (\{1,\ldots,m\}, E)$ and $\bar{p} = (p_1,\ldots,p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G,\bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1,\ldots,E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1,\ldots,m$:

$$q_I(G,\bar{p}) > 0 \iff Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

Shearer's lemma, 1985

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

$$q_I(G, \bar{p}) > 0 \iff Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Kolipaka, Rao and Szegedy (2011): algorithmic proof.

Shearer's lemma, 1985

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

$$q_I(G, \bar{p}) > 0 \iff Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Kolipaka, Rao and Szegedy (2011): algorithmic proof.
- Harris (2015): Proof that the lopsided LLL in the variable framework can be stronger that Shearer's Lemma.

Shearer's lemma, 1985

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

$$q_I(G, \bar{p}) > 0 \implies Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- Kolipaka, Rao and Szegedy (2011): algorithmic proof.
- Harris (2015): Proof that the lopsided LLL in the variable framework can be stronger that Shearer's Lemma.

Outline

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.
    - Use “tree-like” structures to depict the executions of our algorithms.

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.
  - Use “tree-like” structures to depict the executions of our algorithms.
  - Use complementary “validation” algorithms to avoid dependencies introduced by the Moser-like algorithms.

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.
  - Use "tree-like" structures to depict the executions of our algorithms.
  - Use complementary "validation" algorithms to avoid dependencies introduced by the Moser-like algorithms.
- Using direct probabilistic arguments, express a bound to the probability that our algorithms last for at least n steps by a recurrence relation.

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.
  - Use "tree-like" structures to depict the executions of our algorithms.
  - Use complementary "validation" algorithms to avoid dependencies introduced by the Moser-like algorithms.
- Using direct probabilistic arguments, express a bound to the probability that our algorithms last for at least n steps by a recurrence relation.
- Solve the recurrence by analytic means and show that this probability is inverse exponential in n.

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.
  - Use "tree-like" structures to depict the executions of our algorithms.
  - Use complementary "validation" algorithms to avoid dependencies introduced by the Moser-like algorithms.
- Using direct probabilistic arguments, express a bound to the probability that our algorithms last for at least n steps by a recurrence relation.
- Solve the recurrence by analytic means and show that this probability is inverse exponential in n.

- Giotis, Kirousis, Psaromiligkos and Thilikos (2015): symmetric LLL.

- Devise Moser-like algorithms that resample occurring events and then check their neighborhoods.
  - Use "tree-like" structures to depict the executions of our algorithms.
  - Use complementary "validation" algorithms to avoid dependencies introduced by the Moser-like algorithms.
- Using direct probabilistic arguments, express a bound to the probability that our algorithms last for at least n steps by a recurrence relation.
- Solve the recurrence by analytic means and show that this probability is inverse exponential in n.

- Giotis, Kirousis, Psaromiligkos and Thilikos (2015): symmetric LLL.
- Giotis, Kirousis, Livieratos, Psaromiligkos and Thilikos (2018): (variable-directed) lopsidependent LLL.

Outline

M-Algorithm

.

.

M-Algorithm

1. Sample the variables $X_1, \ldots, X_l$.

M-Algorithm

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let E_j be the least indexed such event and *call* $\text{Resample}(E_j)$.

.

M-ALGORITHM

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let E_j be the least indexed such event and *call* RESAMPLE(E_j).

RESAMPLE(E_j)

.

M-ALGORITHM

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let E_j be the least indexed such event and *call* RESAMPLE(E_j).

RESAMPLE(E_j)

3. Resample the variables in sc(E_j).

.

M-ALGORITHM

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let E_j be the least indexed such event and *call* RESAMPLE(E_j).

RESAMPLE(E_j)

3. Resample the variables in sc(E_j).
4. **while** some event in $N_j \cup \{j\}$ occurs, let E_k be the least indexed such event and *call* RESAMPLE(E_k).

M-ALGORITHM

1. Sample the variables $X_1, \dots, X_l$.
2. **while** there exists an occurring event, let E_j be the least indexed such event and *call* $\textsc{Resample}(E_j)$.

$\textsc{Resample}(E_j)$

3. Resample the variables in $\mathrm{sc}(E_j)$.
4. **while** some event in $N_j \cup \{j\}$ occurs, let E_k be the least indexed such event and *call* $\textsc{Resample}(E_k)$.
   - **If** E_j occurs *call* $\textsc{Resample}(E_j)$ **else**
   - **while** some event in N_j occurs, let E_k be the least indexed such event and *call* $\textsc{Resample}(E_k)$

M-ALGORITHM

1. Sample the variables $X_1, \dots, X_l$.
2. **while** there exists an occurring event, let E_j be the least indexed such event and *call* RESAMPLE(E_j).

RESAMPLE(E_j)

3. Resample the variables in sc(E_j).
4. **while** some event in $N_j \cup \{j\}$ occurs, let E_k be the least indexed such event and *call* RESAMPLE(E_k).

M-Algorithm

- If and when M-Algorithm stops, it finds a solution.

M-Algorithm

- If and when M-Algorithm stops, it finds a solution.
- (Lemma) Resample(E_j):

M-ALGORITHM

- If and when M-ALGORITHM stops, it finds a solution.
- (Lemma) RESAMPLE(E_j):
  - E_i does not occur at the beginning $\Rightarrow$ E_i does not occur at the end,

M-Algorithm

- If and when M-Algorithm stops, it finds a solution.
- (Lemma) Resample(E_j):
  - E_i does not occur at the beginning $\Rightarrow$ E_i does not occur at the end,
  - E_j does not occur at the end.

M-Algorithm

- If and when M-Algorithm stops, it finds a solution.
- (Lemma) Resample(E_j):
  - E_i does not occur at the beginning $\Rightarrow$ E_i does not occur at the end,
  - E_j does not occur at the end.
- (Corollary) At most m root Resample calls.

M-Algorithm

- If and when M-Algorithm stops, it finds a solution.
- (Lemma) Resample(E_j):
  - E_i does not occur at the beginning $\Rightarrow$ E_i does not occur at the end,
  - E_j does not occur at the end.
- (Corollary) At most m root Resample calls.

$P_n = Pr$[M-Algorithm lasts for at least n rounds].

M-Algorithm

- If and when M-Algorithm stops, it finds a solution.
- (Lemma) Resample(E_j):
  - E_i does not occur at the beginning $\Rightarrow$ E_i does not occur at the end,
  - E_j does not occur at the end.
- (Corollary) At most m root Resample calls.

$$P_n = Pr[\text{M-Algorithm lasts for at least } n \text{ rounds}].$$

Aim: Prove that P_n is inverse exponential in n.

Outline

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of M-ALGORITHM:

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of M-ALGORITHM:

- For each RESAMPLE(E_j) call, create a node labeled by E_j.

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of $\textsc{M-Algorithm}$:

- For each $\textsc{Resample}(E_j)$ call, create a node labeled by E_j.
- If $\textsc{Resample}(E_k)$ is called from within $\textsc{Resample}(E_j)$, E_k is a child of E_j.

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of M-ALGORITHM:

- For each RESAMPLE(E_j) call, create a node labeled by E_j.
- If RESAMPLE(E_k) is called from within RESAMPLE(E_j), E_k is a child of E_j.
- Labels of nodes are pairwise distinct for roots and for siblings,

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of M-ALGORITHM:

- For each $\textsc{Resample}(E_j)$ call, create a node labeled by E_j.
- If $\textsc{Resample}(E_k)$ is called from within $\textsc{Resample}(E_j)$, E_k is a child of E_j.
- Labels of nodes are pairwise distinct for roots and for siblings,
- Out-degree of node $E_j \leq |N_j|$,

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of M-ALGORITHM:

- For each RESAMPLE(E_j) call, create a node labeled by E_j.
- If RESAMPLE(E_k) is called from within RESAMPLE(E_j), E_k is a child of E_j.
- Labels of nodes are pairwise distinct for roots and for siblings,
- Out-degree of node $E_j \leq |N_j|$,
- at most m roots.

Witness Forests

Construct the witness forest $\mathcal{F}$ of an execution of M-ALGORITHM:

- For each RESAMPLE(E_j) call, create a node labeled by E_j.
- If RESAMPLE(E_k) is called from within RESAMPLE(E_j), E_k is a child of E_j.
- Labels of nodes are pairwise distinct for roots and for siblings,
- Out-degree of node $E_j \leq |N_j|$,
- at most m roots.

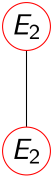
- $$P_n = \sum_{\mathcal{F}:|\mathcal{F}|=n} Pr[\text{M-ALGORITHM executes with witness forest } \mathcal{F}.]$$

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

E_2

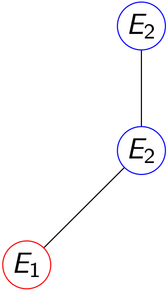
$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$$(0, 1, 0, 0, 1, 0)$$



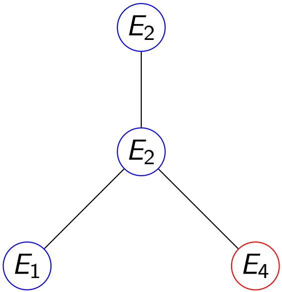
$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$$(0, 1, 0, 0, 1, 0)$$



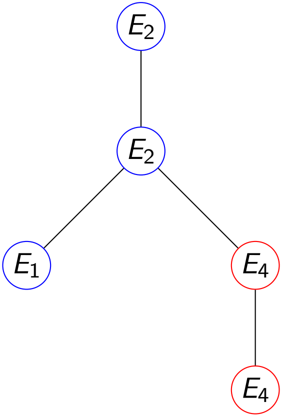
$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$$(0, 0, 0, 0, 1, 0)$$



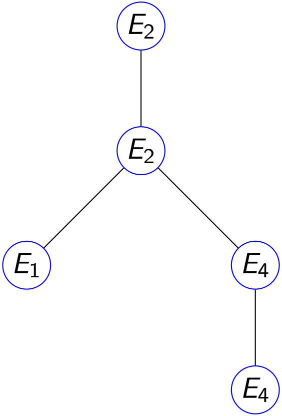
$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$$(1, 0, 0, 0, 1, 0)$$



$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$$(1, 0, 0, 0, 1, 0)$$



$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_3 \vee x_4)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_5)}_{C_3} \wedge \underbrace{(x_4 \vee \bar{x}_5 \vee x_6)}_{C_4}$$

$$(1, 0, 0, 0, 1, 1)$$

Validation Algorithm

M-ALGORITHM introduces dependencies as it selects which event to resample.

Validation Algorithm

M-Algorithm introduces dependencies as it selects which event to resample.

<u>ValAlg</u> (**Input:** $E_{j_1}, \ldots, E_{j_n}$ of $\mathcal{F}$)

Validation Algorithm

M-ALGORITHM introduces dependencies as it selects which event to resample.

<u>VALALG</u> (**Input:** $E_{j_1}, \ldots, E_{j_n}$ of $\mathcal{F}$)

1. Sample X_i, $i = 1, \ldots, l$.

Validation Algorithm

M-ALGORITHM introduces dependencies as it selects which event to resample.

<u>VALALG</u> (**Input:** $E_{j_1}, ..., E_{j_n}$ of $\mathcal{F}$)

1. Sample X_i, $i = 1, ..., l$.
2. **for** $s = 1, ..., n$ **do**
3.   **if** E_{j_s} does not occur **return** `failure` and exit

Validation Algorithm

M-ALGORITHM introduces dependencies as it selects which event to resample.

<u>VALALG</u> (**Input:** $E_{j_1}, \ldots, E_{j_n}$ of $\mathcal{F}$)

1. Sample X_i, $i = 1, \ldots, l$.
2. **for** $s = 1, \ldots, n$ **do**
3.   **if** E_{j_s} does not occur **return** `failure` and exit
4. **else** Resample sc(E_{j_i}).
5. **end for**

Validation Algorithm

M-ALGORITHM introduces dependencies as it selects which event to resample.

<u>VALALG</u> (**Input:** $E_{j_1}, \ldots, E_{j_n}$ of $\mathcal{F}$)

1. Sample X_i, $i = 1, \ldots, l$.
2. **for** $s = 1, \ldots, n$ **do**
3. **if** E_{j_s} does not occur **return** `failure` and exit
4. **else** Resample sc(E_{j_i}).
5. **end for**
6. **return** `success`.

Validation Algorithm

M-ALGORITHM introduces dependencies as it selects which event to resample.

<u>VALALG</u> (**Input:** $E_{j_1}, ..., E_{j_n}$ of $\mathcal{F}$)

1. Sample X_i, $i = 1, ..., l$.
2. **for** $s = 1, ..., n$ **do**
3. **if** E_{j_s} does not occur **return** `failure` and exit
4. **else** Resample sc(E_{j_i}).
5. **end for**
6. **return** `success`.

$$P_n = \sum_{\mathcal{F}:|\mathcal{F}|=n} Pr[\text{M-ALGORITHM executes with witness forest } \mathcal{F}]$$
$$\leq \sum_{\mathcal{F}:|\mathcal{F}|=n} Pr[\text{VALALG succeeds on } \mathcal{F}]$$

Outline

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

- T_1's root has one child labeled by E_j,

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

- T_1's root has one child labeled by E_j,
- T_2's root has children with labels in N_j,

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

- T_1's root has one child labeled by E_j,
- T_2's root has children with labels in N_j,
- $\mathbf{n} = (n_1, \ldots, n_{2m}) : \sum_{i=1}^{2m} n_i = n$,

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

- T_1's root has one child labeled by E_j,
- T_2's root has children with labels in N_j,
- $\mathbf{n} = (n_1, \ldots, n_{2m}) : \sum_{i=1}^{2m} n_i = n$,
- $Q_{\mathbf{n},j}$, $R_{\mathbf{n},j}$:
  - $Pr[V_{T_1}] \leq Q_{\mathbf{n},j}$

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

- T_1's root has one child labeled by E_j,
- T_2's root has children with labels in N_j,
- $\mathbf{n} = (n_1, \ldots, n_{2m}) : \sum_{i=1}^{2m} n_i = n$,
- $Q_{\mathbf{n},j}$, $R_{\mathbf{n},j}$:
  - $Pr[V_{T_1}] \leq Q_{\mathbf{n},j}$
  - $Pr[V_{T_2}] \leq R_{\mathbf{n},j}$.

Witness Trees

T_1, T_2 trees with n nodes each and roots labeled by E_j:

- T_1's root has one child labeled by E_j,
- T_2's root has children with labels in N_j,
- $\mathbf{n} = (n_1, \ldots, n_{2m}) : \sum_{i=1}^{2m} n_i = n$,
- $Q_{\mathbf{n},j}$, $R_{\mathbf{n},j}$:
  - $Pr[V_{T_1}] \leq Q_{\mathbf{n},j}$
  - $Pr[V_{T_2}] \leq R_{\mathbf{n},j}$.

$$P_n \leq \sum_{\mathbf{n}} \sum_{\mathbf{n}^1+\cdots+\mathbf{n}^m=\mathbf{n}} \left(Q_{\mathbf{n}^1,1} + R_{\mathbf{n}^1,1}\right) \cdots \left(Q_{\mathbf{n}^m,m} + R_{\mathbf{n}^m,m}\right).$$

Multivariate generating functions

Multivariate generating functions

$\mathbf{t} = (t_1, \ldots, t_{2m})$ and $\mathbf{t}^{\mathbf{n}} = (t_1^{n_1}, \ldots, t_{2m}^{n_{2m}})$.

Multivariate generating functions

$\mathbf{t} = (t_1, \dots, t_{2m})$ and $\mathbf{t}^{\mathbf{n}} = (t_1^{n_1}, \dots, t_{2m}^{n_{2m}})$.

$$Q_j(\mathbf{t}) = \sum_{\mathbf{n}: n_j \geq 1} Q_{\mathbf{n},j} \mathbf{t}^{\mathbf{n}},$$

$$R_j(\mathbf{t}) = \sum_{\mathbf{n}: n_{m+j} \geq 1} R_{\mathbf{n},j} \mathbf{t}^{\mathbf{n}}.$$

Multivariate generating functions

$\mathbf{t} = (t_1, \dots, t_{2m})$ and $\mathbf{t}^{\mathbf{n}} = (t_1^{n_1}, \dots, t_{2m}^{n_{2m}})$.

$$Q_j(\mathbf{t}) = \sum_{\mathbf{n}: n_j \geq 1} Q_{\mathbf{n},j} \mathbf{t}^{\mathbf{n}},$$

$$R_j(\mathbf{t}) = \sum_{\mathbf{n}: n_{m+j} \geq 1} R_{\mathbf{n},j} \mathbf{t}^{\mathbf{n}}.$$

$$Q_{\mathbf{n},j} = Pr[E_j] \Big(Q_{\mathbf{n}-(1)_j,j} + R_{\mathbf{n}-(1)_j,j} \Big),$$

$$R_{\mathbf{n},j} = Pr[E_j] \sum_{\mathbf{n}^1 + \cdots + \mathbf{n}^{k_j} = \mathbf{n} - (1)_{m+j}} \Big(Q_{\mathbf{n}^1, j_1} + R_{\mathbf{n}^1, j_1} \Big) \cdots \Big(Q_{\mathbf{n}^{k_j}, j_{k_j}} + R_{\mathbf{n}^{k_j}, j_{k_j}} \Big).$$

- Solve $(Q_1(\mathbf{t}), \dots, Q_m(\mathbf{t}), R_1(\mathbf{t}), \dots, R_m(\mathbf{t}))$ by the multivariate Lagrange inversion formula [Bender and Richmond, 1998].

Outline

The Lemma

$G = (\{1, \dots, m\}, E)$ and $\bar{p} = (p_1, \dots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \dots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \dots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \dots, m$:

$$q_I(G, \bar{p}) > 0 \implies Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

The Lemma

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

$$q_I(G, \bar{p}) > 0 \implies Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- $N(I)$: I and its neighbors in the dependency graph.

The Lemma

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

$$q_I(G, \bar{p}) > 0 \implies Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- $N(I)$: I and its neighbors in the dependency graph.
- vbl(I): set of variables in the scopes of the events in I.

The Lemma

$G = (\{1, \ldots, m\}, E)$ and $\bar{p} = (p_1, \ldots, p_m)$.
$I(G) := \{\emptyset = I_0, I_1, \ldots, I_s\}$ the set of independent sets of G.

$$q_I(G, \bar{p}) := \sum_{J \in I(G) : I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j.$$

For $E_1, \ldots, E_m$ with dependency graph G and $Pr[E_j] = p_j$, $j = 1, \ldots, m$:

$$q_I(G, \bar{p}) > 0 \implies Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0.$$

- $N(I)$: I and its neighbors in the dependency graph.
- vbl(I): set of variables in the scopes of the events in I.
- I covers J if $J \subseteq N(I)$.

GenResample

GenResample

1. Sample the variables $X_1, \ldots, X_l$.

GenResample

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let I_i be the least indexed maximal independent set containing only occurring events and **do**

GenResample

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let I_i be the least indexed maximal independent set containing only occurring events and **do**
3. Resample $\text{vbl}(I_i)$.

GenResample

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let I_i be the least indexed maximal independent set containing only occurring events and **do**
3. Resample $\text{vbl}(I_i)$.

Lemma

If $I_{i_1}, \ldots, I_{i_n}$ are the independent sets selected by GenResample, *then I_{i_t} covers $I_{i_{t+1}}$.*

GenResample

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let I_i be the least indexed maximal independent set containing only occurring events and **do**
3. Resample vbl(I_i).

Lemma

If $I_{i_1}, \ldots, I_{i_n}$ are the independent sets selected by GenResample*, then I_{i_t} covers $I_{i_{t+1}}$.*

Proof Sketch

- E_j in $I_{i_{t+1}}$ and not in $N(I_{i_t})$.

GenResample

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let I_i be the least indexed maximal independent set containing only occurring events and **do**
3. Resample vbl(I_i).

Lemma

If $I_{i_1}, \ldots, I_{i_n}$ are the independent sets selected by GenResample, *then I_{i_t} covers $I_{i_{t+1}}$.*

Proof Sketch

- E_j in $I_{i_{t+1}}$ and not in $N(I_{i_t})$.
- E_j did not occur at the beginning of t.

GenResample

1. Sample the variables $X_1, \ldots, X_l$.
2. **while** there exists an occurring event, let I_i be the least indexed maximal independent set containing only occurring events and **do**
3. Resample $\text{vbl}(I_i)$.

Lemma

If $I_{i_1}, \ldots, I_{i_n}$ are the independent sets selected by GenResample, *then I_{i_t} covers $I_{i_{t+1}}$.*

Proof Sketch

- E_j in $I_{i_{t+1}}$ and not in $N(I_{i_t})$.
- E_j did not occur at the beginning of t.
- E_j does not depend on $E_t \in I_{i_t}$, thus it does not occur at the end of round t. **Contradiction**.

Outline

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \ldots, I_{i_n}$.

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \ldots, I_{i_n}$.

$$\mathbb{P}_n := Pr[\textsc{GenResample} \text{ lasts for at least } n \text{ rounds}]$$
$$= \sum_{P:|P|=n} Pr[\textsc{GenResample} \text{ executes with path } P].$$

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \ldots, I_{i_n}$.

$$\mathbb{P}_n := Pr[\textsc{GenResample} \text{ lasts for at least } n \text{ rounds}] = \sum_{P:|P|=n} Pr[\textsc{GenResample} \text{ executes with path } P].$$

GenVal

On input P with labels $I_{i_1}, \ldots, I_{i_n}$:

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \dots, I_{i_n}$.

$$\mathbb{P}_n := Pr[\textsc{GenResample} \text{ lasts for at least } n \text{ rounds}] = \sum_{P:|P|=n} Pr[\textsc{GenResample} \text{ executes with path } P].$$

<u>GenVal</u>

On input P with labels $I_{i_1}, \dots, I_{i_n}$:

1. Sample the variables $X_1, \dots, X_l$.

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \ldots, I_{i_n}$.

$$\mathbb{P}_n := Pr[\textsc{GenResample} \text{ lasts for at least } n \text{ rounds}] = \sum_{P:|P|=n} Pr[\textsc{GenResample} \text{ executes with path } P].$$

GenVal

On input P with labels $I_{i_1}, \ldots, I_{i_n}$:

1. Sample the variables $X_1, \ldots, X_l$.
2. Check each I_{i_t}. **If** there is a non-occurring E_j in I_{i_t}, `fail`. **Else**, resample vbl(I_{i_t}).

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \ldots, I_{i_n}$.

$$\mathbb{P}_n := Pr[\textsc{GenResample} \text{ lasts for at least } n \text{ rounds}]$$
$$= \sum_{P:|P|=n} Pr[\textsc{GenResample} \text{ executes with path } P].$$

GenVal

On input P with labels $I_{i_1}, \ldots, I_{i_n}$:

1. Sample the variables $X_1, \ldots, X_l$.
2. Check each I_{i_t}. **If** there is a non-occurring E_j in I_{i_t}, `fail`. **Else**, resample vbl(I_{i_t}).
3. If P has no other nodes, GenVal `succeeds`.

Witness Paths

P: path with n nodes, labeled by $I_{i_1}, \ldots, I_{i_n}$.

$$\mathbb{P}_n := Pr[\textsc{GenResample} \text{ lasts for at least } n \text{ rounds}]$$
$$= \sum_{P:|P|=n} Pr[\textsc{GenResample} \text{ executes with path } P].$$

<u>GenVal</u>

On input P with labels $I_{i_1}, \ldots, I_{i_n}$:

1. Sample the variables $X_1, \ldots, X_l$.
2. Check each I_{i_t}. **If** there is a non-occurring E_j in I_{i_t}, `fail`. **Else**, resample vbl(I_{i_t}).
3. If P has no other nodes, GenVal `succeeds`.

$$\mathbb{P}_n \leq \sum_{P:|P|=n} Pr[\textsc{GenVal} \text{ succeeds with path } P].$$

Recurrence

$$Pr[V_P] := Pr[\textsc{GenVal} \text{ succeeds on input P}].$$

Recurrence

$$Pr[V_P] := Pr[\textsc{GenVal} \text{ succeeds on input P}].$$

$$Q_{n,i}: \; Pr[V_P] \leq Q_{n,i},$$

where P has n nodes, whith source I_i.

Recurrence

$$Pr[V_P] := Pr[\textsc{GenVal} \text{ succeeds on input P}].$$

$$Q_{n,i} : \; Pr[V_P] \leq Q_{n,i},$$

where P has n nodes, whith source I_i.

$$Q_{n,i} = \prod_{j \in I_i} p_j \sum_{I_i \text{ covers } J} Q_{n-1,J}.$$

Recurrence

$$Pr[V_P] := Pr[\text{GenVal succeeds on input P}].$$

$$Q_{n,i}: \; Pr[V_P] \leq Q_{n,i},$$

where P has n nodes, whith source I_i.

$$Q_{n,i} = \prod_{j \in I_i} p_j \sum_{I_i \text{ covers } J} Q_{n-1,J}.$$

- Show that Q_{n_i} are inverse exponential to n.

Outline

Stable set matrix

- M $s \times s$ matrix, $M(i,j) = \prod_{j \in I} p_j$ if I covers J. Else it is 0.

Stable set matrix

- M $s \times s$ matrix, $M(i,j) = \prod_{j \in I} p_j$ if I covers J. Else it is 0.
- $q_n = (Q_{n,1}, \ldots, Q_{n,s})$:

Stable set matrix

- M $s \times s$ matrix, $M(i,j) = \prod_{j \in I} p_j$ if I covers J. Else it is 0.
- $q_n = (Q_{n,1}, \ldots, Q_{n,s})$:

$$q_n = M q_{n-1} \Rightarrow q_n = M^{n-1} q_1.$$

Stable set matrix

- M $s \times s$ matrix, $M(i,j) = \prod_{j \in I} p_j$ if I covers J. Else it is 0.
- $q_n = (Q_{n,1}, \ldots, Q_{n,s})$:

$$q_n = Mq_{n-1} \Rightarrow q_n = M^{n-1} q_1.$$

$$\mathbb{P}_n \leq \sum_{i=1}^{s} Q_{n,i} = ||q_n||_1,$$

where $|| \cdot ||_1$ is the 1-norm.

Matrix norms

$$||M||_1 := sup_{x \neq 0} \frac{||Mx||_1}{||x||_1} \geq \frac{||Mq_1||_1}{||q1||_1},$$

where $||M||_1$ is the induced norm for squared matrices that $|| \cdot ||_1$ yields [Horn and Johnson, *Matrix Analysis*, 1990].

Matrix norms

$$||M||_1 := sup_{x \neq 0} \frac{||Mx||_1}{||x||_1} \geq \frac{||Mq_1||_1}{||q1||_1},$$

where $||M||_1$ is the induced norm for squared matrices that $|| \cdot ||_1$ yields [Horn and Johnson, *Matrix Analysis*, 1990].

$$||q_n||_1 = ||M^{n-1}||_1 \cdot ||q_1||_1.$$

Matrix norms

$$||M||_1 := sup_{x \neq 0} \frac{||Mx||_1}{||x||_1} \geq \frac{||Mq_1||_1}{||q1||_1},$$

where $||M||_1$ is the induced norm for squared matrices that $|| \cdot ||_1$ yields [Horn and Johnson, *Matrix Analysis*, 1990].

$$||q_n||_1 = ||M^{n-1}||_1 \cdot ||q_1||_1.$$

- It suffices to prove that $||M^{n-1}||_1$ is inverse exponential to n.

Matrix norms

$$||M||_1 := sup_{x \neq 0} \frac{||Mx||_1}{||x||_1} \geq \frac{||Mq_1||_1}{||q1||_1},$$

where $||M||_1$ is the induced norm for squared matrices that $|| \cdot ||_1$ yields [Horn and Johnson, *Matrix Analysis*, 1990].

$$||q_n||_1 = ||M^{n-1}||_1 \cdot ||q_1||_1.$$

- It suffices to prove that $||M^{n-1}||_1$ is inverse exponential to n.
- $p(M)$ spectral radius of M.

Bounding $||M^{n-1}||_1$

Bounding $||M^{n-1}||_1$

- Kolipaka et. al: $q_I(G, \bar{p}) > 0 \Rightarrow p(M) < 1$.

Bounding $||M^{n-1}||_1$

- Kolipaka et. al: $q_I(G, \bar{p}) > 0 \Rightarrow p(M) < 1$.
- Gelfand's formula: $p(M) = \lim_{n\to\infty} ||M^n||^{1/n}$,

Bounding $||M^{n-1}||_1$

- Kolipaka et. al: $q_I(G, \bar{p}) > 0 \Rightarrow p(M) < 1$.
- Gelfand's formula: $p(M) = \lim_{n\to\infty} ||M^n||^{1/n}$,

Thus:

$$||M^{n-1}||_1 < (p(M) + \epsilon)^{n-1}.$$

Thank you!