

# Advanced Algorithms

南京大学

尹一通

# Constraint Satisfaction Problem (CSP)

- **variables:**  $X = \{x_1, x_2, \dots, x_n\}$ 
  - each variable ranges over a finite **domain**  $\Omega$
  - an **assignment**  $\sigma \in \Omega^X$  assigns each variable a value in  $\Omega$
- **constraints:**  $C_1, C_2, \dots, C_m$ 
  - each constraint  $C_i$  is a **Boolean function**  
$$C_i : \Omega^{S_i} \rightarrow \{\text{true}, \text{false}\}$$

defined on a subset of variables  $S_i \subseteq X$
  - a constraint  $C_i$  is **satisfied** by an assignment  $\sigma \in \Omega^X$  if  
$$C_i(\sigma_{S_i}) = \text{true}$$

# Constraint Satisfaction Problem (CSP)

- **variables:**  $x_1, x_2, \dots, x_n \in \Omega$
- **constraints:**  $C_1, C_2, \dots, C_m$   
 $C_i : \Omega^{S_i} \rightarrow \{\text{true}, \text{false}\}$

**Examples:** satisfiability, optimization, counting, ...

- **graph cut:**  $\Omega = \{0,1\}$ , constraints:  $x_u \neq x_v$  for each edge  $uv$
- **$k$ -coloring:**  $\Omega = [k]$ , constraints:  $x_u \neq x_v$  for each edge  $uv$
- **matching/cover:**  $\Omega = \{0,1\}$ , constraints:  
$$\sum_{j \in S_i} x_j \leq 1 \text{ (matching)} \quad \text{or} \quad \sum_{j \in S_i} x_j \geq 1 \text{ (cover)}$$
- **SAT:**  $\Omega = \{\text{true}, \text{false}\}$ , constraints are *clauses*

# Algorithmic Problems for CSP

CSP	Satisfiability	Optimization	Counting
2SAT	<b>P</b>	<b>NP</b> -hard	<b>#P</b> -complete
3SAT	<b>NP</b> -complete	<b>NP</b> -hard	<b>#P</b> -complete
matching	perfect matching <b>P</b>	max matching <b>P</b>	<b>#P</b> -complete
cut (2-coloring)	bipartite test <b>P</b>	max-cut <b>NP</b> -hard	<b>FP</b> (poly-time)
3-coloring	<b>NP</b> -complete	max-3-cut <b>NP</b> -hard	<b>#P</b> -complete

# Algorithmic Problems for CSP

Given a CSP instance:

- **satisfiability**: determine whether  $\exists$  an assignment satisfying all constraints
- **search**: return a satisfying assignment
- **optimization**: find an assignment satisfying as many constraints as possible
- **refutation** (dual): find a “proof” of “no assignment can satisfy  $> m^*$  constraints” for  $m^*$  as small as possible
- **counting**: estimate the number of satisfying assignments
- **sampling**: random sample a satisfying assignments
- **inference**: calculate the possibility of a variable being assigned certain value

# $k$ -SAT

**Instance:** a  $k$ -CNF formula  $\phi$ .

Determine whether  $\phi$  is **satisfiable**.

( $\exists$  a satisfying assignment  $\sigma$  s.t.  $\phi(\sigma) = \text{true}$ )

**CNF (Conjunctive Normal Form):**

$$(x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_3 \vee x_5) \wedge (x_2 \vee x_4 \vee x_5)$$

# $k$ -SAT

**Instance:** a  $k$ -CNF formula  $\phi$ .

Determine whether  $\phi$  is **satisfiable**.

( $\exists$  a satisfying assignment  $\sigma$  s.t.  $\phi(\sigma) = \text{true}$ )

**CNF (Conjunctive Normal Form):**

- $n$  **Boolean variables:**  $x_1, x_2, \dots, x_n \in \{\text{true}, \text{false}\}$
- $m$  **clauses:**  $C_1 \wedge C_2 \wedge \dots \wedge C_m$
- each clause is in the form  $C_i = \ell_{i_1} \vee \ell_{i_2} \vee \dots \vee \ell_{i_{k_i}}$
- each **literal**  $\ell_{i_j} \in \{x_s, \neg x_s\}$  for some  $s \in \{1, 2, \dots, n\}$

**$k$ -CNF: (exact- $k$ -CNF)**

- each clause contains **exactly  $k$**  variables

# $k$ -SAT

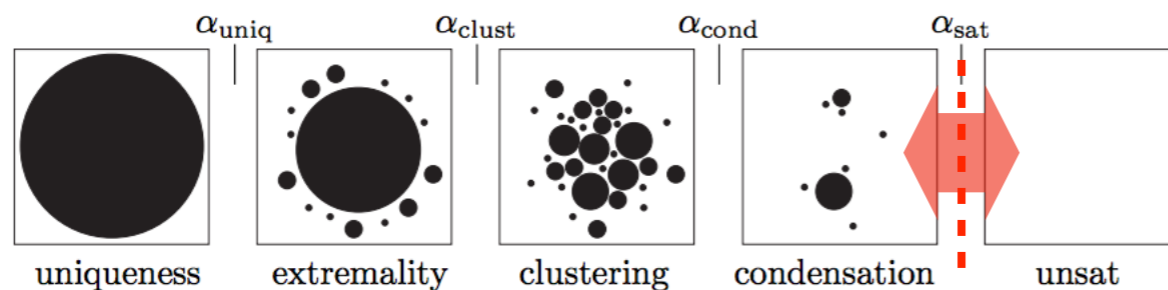
**Instance:** a  $k$ -CNF formula  $\phi$ .

Determine whether  $\phi$  is **satisfiable**.

$$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

random  $k$ -CNF formula with  $m = \alpha n$  clauses

phase transition of satisfiability  
for random CSP:

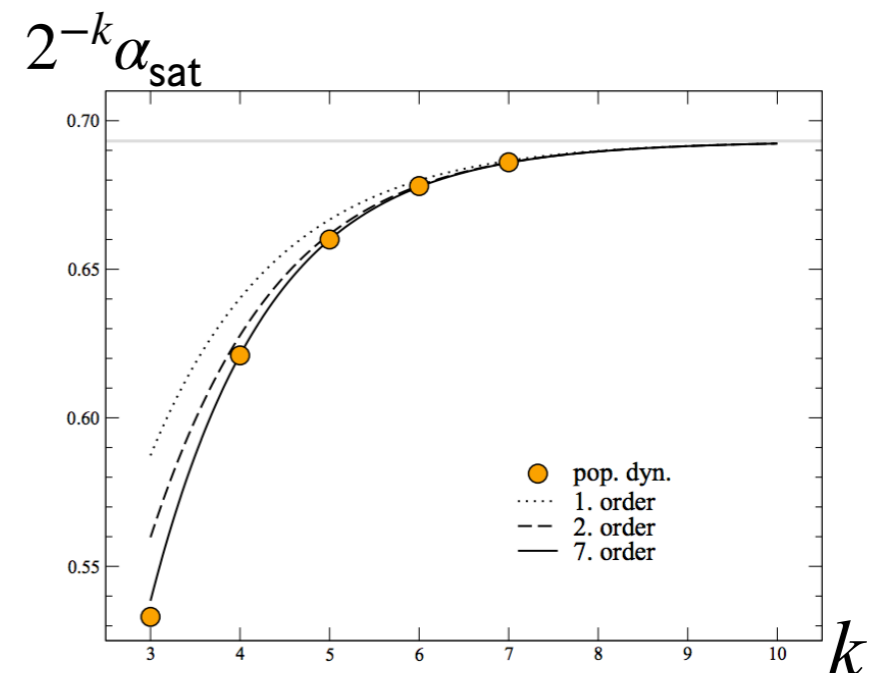


[Ding, Sly, Sun, **STOC**'15]

[Krzakała, Montanari, Ricci-

Tersenghi, Semerjian, Zdeborová, **PNAS**'07]

[Achlioptas, Naor, Peres, **Nature**'05]





# $k$ -SAT

**Instance:** a  $k$ -CNF formula  $\phi$ .

Determine whether  $\phi$  is **satisfiable**.

$$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

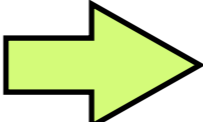
$k$ -CNF: (**exact- $k$ -CNF**)

- each clause contains **exactly  $k$**  variables

degree  $d$ :

(shares variables)

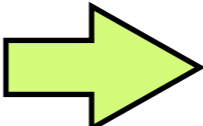
- each clause **intersects** with  $\leq d$  **other** clauses

**Theorem:**  $d \leq 2^{k-2}$    $\phi$  is always satisfiable

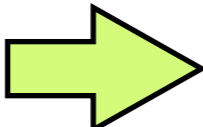
# The *Lovász Local Lemma*

$\phi$ : a *k*-CNF formula of degree *d*

The *Lovász Local Lemma* (LLL) for *k*-SAT:

**Theorem:**  $d \leq 2^{k-2}$    $\phi$  is always satisfiable

**Algorithmic LLL** for *k*-SAT:

**Theorem** (Moser 2009):  $\exists$  constant  $c > 0$   
 $d \leq 2^{k-c}$   satisfying assignment can be found  
in time  $O(n + km)$  w.h.p.

# The *Probabilistic Method*

$\phi$ : a *k*-CNF formula of degree *d*

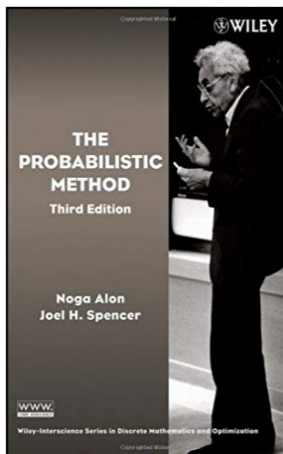
**Theorem:**  $d \leq 2^{k-2} \Rightarrow \phi$  is always satisfiable

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

***NaïveRandomGuess*( $\phi$ )**

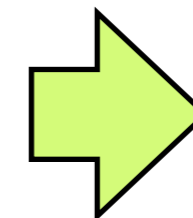
sample a **uniform random** assignment

$$X_1, X_2, \dots, X_n \in \{\text{true}, \text{false}\};$$



**The  
*Probabilistic  
Method:***

$$\Pr[\phi(\mathbf{X})=\text{true}] > 0$$



$\exists$  a satisfying  
assignment  
( $\phi$  is satisfiable)

# The *Probabilistic Method*

$\phi$ : a *k*-CNF formula of degree *d*

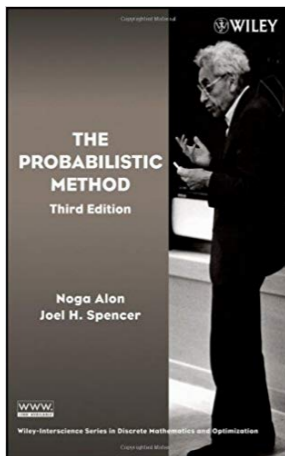
**Theorem:**  $d \leq 2^{k-2}$   $\Rightarrow$   $\phi$  is always satisfiable

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

sample a **uniform random** assignment

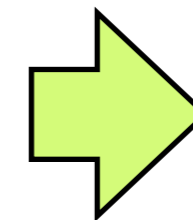
$$X_1, X_2, \dots, X_n \in \{\text{true}, \text{false}\};$$

**bad event**  $A_i$ : clause  $C_i$  is **unsatisfied**



**The  
Probabilistic  
Method:**

$$\Pr \left[ \bigwedge_{i=1}^m \overline{A_i} \right] > 0$$



$\exists$  a satisfying  
assignment  
( $\phi$  is satisfiable)

# The *Lovász Sieve*

$m$  bad event:  $A_1, A_2, \dots, A_m$

**Goal:**  $\Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > 0 \quad (\star)$

- **union bound:**  $\sum_{i=1}^m \Pr[A_i] < 1 \Rightarrow (\star)$
- **principle of inclusion exclusion (PIE):**

$$\sum_{\substack{S \subseteq \{1, \dots, m\} \\ S \neq \emptyset}} (-1)^{|S|-1} \Pr \left[ \bigwedge_{i \in S} A_i \right] < 1 \Rightarrow (\star)$$

- **LLL:** every  $A_i$  is independent of all but  $\leq d$  other bad events (degree  $\leq d$ )

$$\forall i : \Pr[A_i] \leq \frac{1}{4d} \Rightarrow (\star)$$

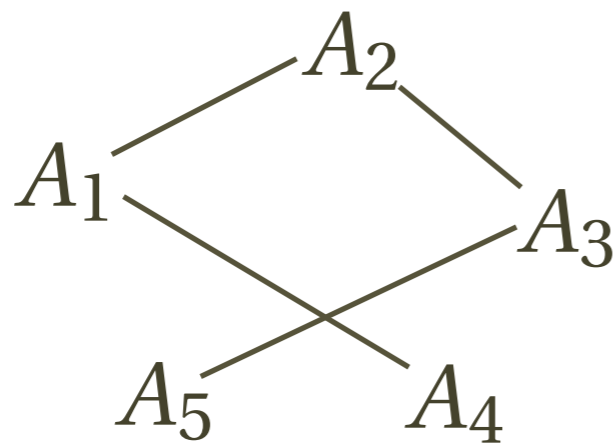
$m$  bad event:  $A_1, A_2, \dots, A_m$

every  $A_i$  is independent of all but  $\leq d$  other bad events

**Lovász Local Lemma** (Erdos-Lovász 1975):

$$\forall i : \Pr[A_i] \leq \frac{1}{4d} \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

**Example:**



dependency graph  
(max degree  $d$ )

$$A_1(X_1, X_4)$$

$$A_4(X_4)$$

$$A_2(X_1, X_2)$$

$$A_5(X_3)$$

$$A_3(X_2, X_3)$$

$$X_1, X_2, X_3, X_4$$

are mutually independent

$m$  **bad event**:  $A_1, A_2, \dots, A_m$

every  $A_i$  is **independent of all but  $\leq d$  other** bad events

**Lovász Local Lemma** (Lovász 1977):

$$\forall i : \Pr[A_i] \leq \frac{1}{e(d+1)} \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

$$\uparrow \quad \alpha_1 = \alpha_2 = \dots = \alpha_m = \frac{1}{d+1}$$

**Lovász Local Lemma** (**asymmetric version**):

$$\exists \alpha_1, \alpha_2, \dots, \alpha_m \in [0, 1)$$

$$\forall i : \Pr[A_i] \leq \alpha_i \prod_{j \sim i} (1 - \alpha_j) \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > \prod_{i=1}^m (1 - \alpha_i)$$

$j \sim i$ :  $A_i$  and  $A_j$  are adjacent in the **dependency graph**

$m$  **bad event**:  $A_1, A_2, \dots, A_m$

every  $A_i$  is **independent of all but  $\leq d$  other** bad events

**Lovász Local Lemma** (Erdos-Lovász 1975):

$$\forall i : \Pr[A_i] \leq \frac{1}{4d} \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

$$\uparrow \quad \alpha_1 = \alpha_2 = \dots = \alpha_m = \frac{1}{2d}$$

**Lovász Local Lemma** (**asymmetric version**):

$$\exists \alpha_1, \alpha_2, \dots, \alpha_m \in [0, 1)$$

$$\forall i : \Pr[A_i] \leq \alpha_i \prod_{j \sim i} (1 - \alpha_j) \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > \prod_{i=1}^m (1 - \alpha_i)$$

$j \sim i$ :  $A_i$  and  $A_j$  are adjacent in the **dependency graph**



# The Lovász Local Lemma

$\phi$ : a  $k$ -CNF formula of degree  $d$

**Theorem:**  $d \leq 2^{k-2}$   $\Rightarrow$   $\phi$  is always satisfiable

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

sample a **uniform random** assignment

$$X_1, X_2, \dots, X_n \in \{\text{true}, \text{false}\};$$

**bad event**  $A_i$ : clause  $C_i$  is **unsatisfied**

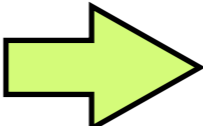
$$\forall i : \Pr[A_i] = 2^{-k} \leq \frac{1}{4d} \xrightarrow{\text{LLL}} \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

( $k$ -CNF)  ( $\phi$  is satisfiable)

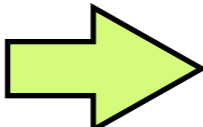
# Algorithmic LLL

$\phi$ : a  $k$ -CNF formula of degree  $d$

The *Lovász Local Lemma* (LLL) for  $k$ -SAT:

**Theorem:**  $d \leq 2^{k-2}$    $\phi$  is always satisfiable

Algorithmic LLL for  $k$ -SAT:

**Theorem** (Moser 2009):  $\exists$  constant  $c > 0$   
 $d \leq 2^{k-c}$   satisfying assignment can be found  
in time  $O(n + km)$  w.h.p.

# Moser's Algorithm

$\phi$ : a  $k$ -CNF formula of degree  $d$

**Solve**( $\phi$ )

sample a uniform random  
assignment  $X_1, X_2, \dots, X_n$ ;  
while  $\exists$  unsatisfied clause  $C$   
**Fix**( $C$ );

**Fix**( $C$ )

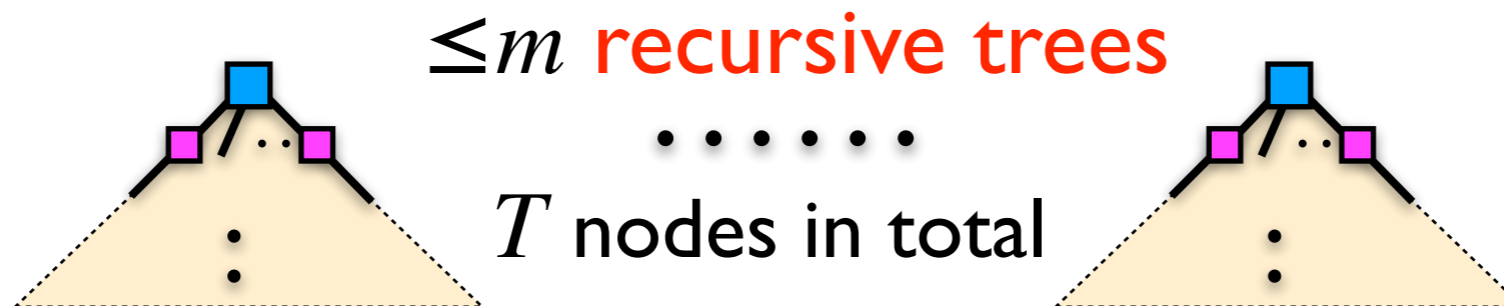
**resample** variables in  $C$  uniformly at random;  
while  $\exists$  unsatisfied clause  $D$  **intersecting**  $C$   
**Fix**( $D$ ); (including  $C$  itself)

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

**Solve**( $\phi$ )  
 sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
 while  $\exists$  unsatisfied clause  $C$   
   **Fix**( $C$ );

**Fix**( $C$ )  
 resample variables in  $C$  uniformly at random;  
 while  $\exists$  unsatisfied clause  $D$  intersecting  $C$   
   **Fix**( $D$ );

- terminate  $\Rightarrow$  successfully return a satisfying solution
- top-level: **Fix**( $C$ ) returned  $\Rightarrow$   $C$  remains satisfied



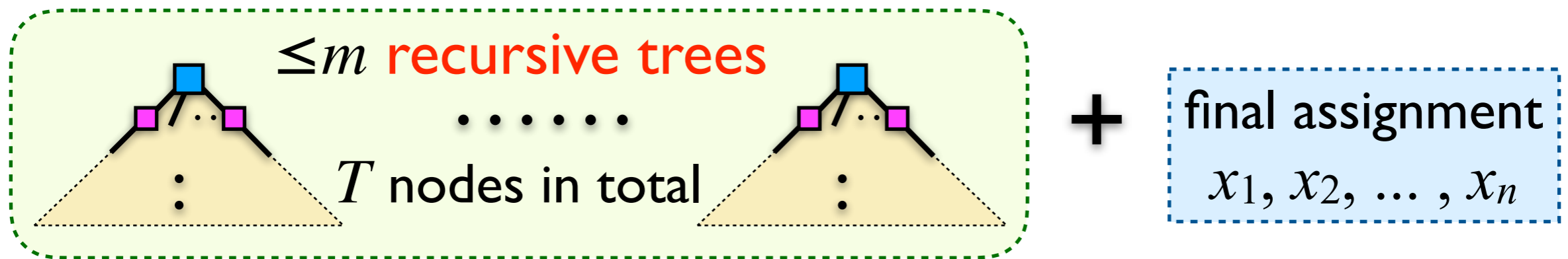
- $T$ : total # of calls to **Fix**( $C$ )  
 (including both top-level and recursive calls)
- total cost:  $n + kT$  (total # of random bits)

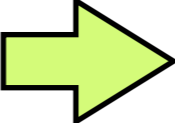
$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

**Solve**( $\phi$ )  
 sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
 while  $\exists$  unsatisfied clause  $C$   
     **Fix**( $C$ );

**Fix**( $C$ )  
 resample variables in  $C$  uniformly at random;  
 while  $\exists$  unsatisfied clause  $D$  intersecting  $C$   
     **Fix**( $D$ );

$n + kT$  random bits



**Observation:**  
 Fix( $C$ ) is called  assignment of  $C$  is uniquely determined

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

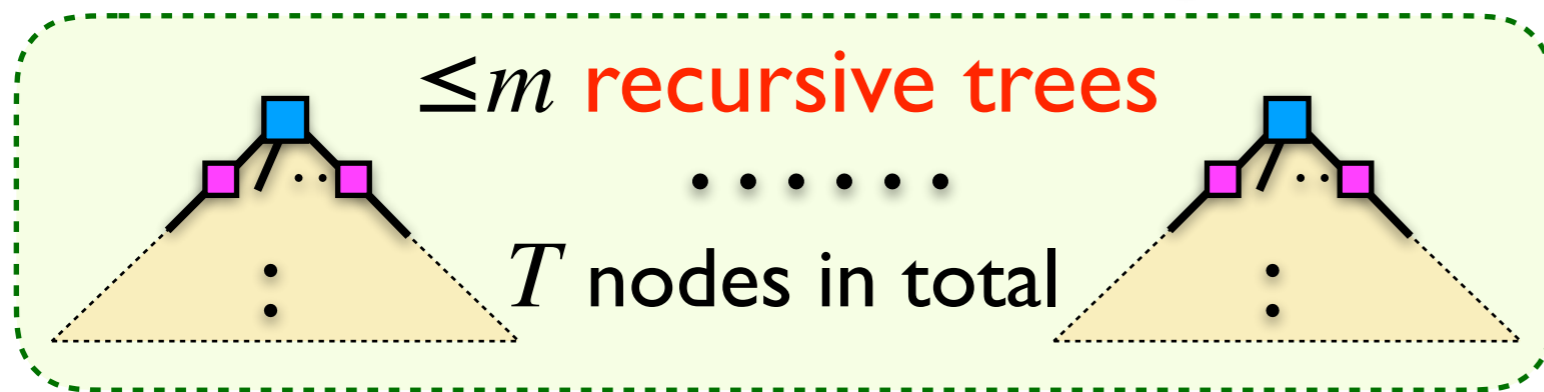
**Solve( $\phi$ )**  
 sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
 while  $\exists$  unsatisfied clause  $C$   
**Fix( $C$ )**;

**Fix( $C$ )**  
 resample variables in  $C$  uniformly at random;  
 while  $\exists$  unsatisfied clause  $D$  intersecting  $C$   
**Fix( $D$ )**;

$n + kT$  random bits



1-1 mapping  $\text{Enc}_\phi$



+ final assignment  
 $x_1, x_2, \dots, x_n$

represented by succinct representation:

$n$  bits

$\leq m \log m + T (\log_2 d + O(1))$  bits

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

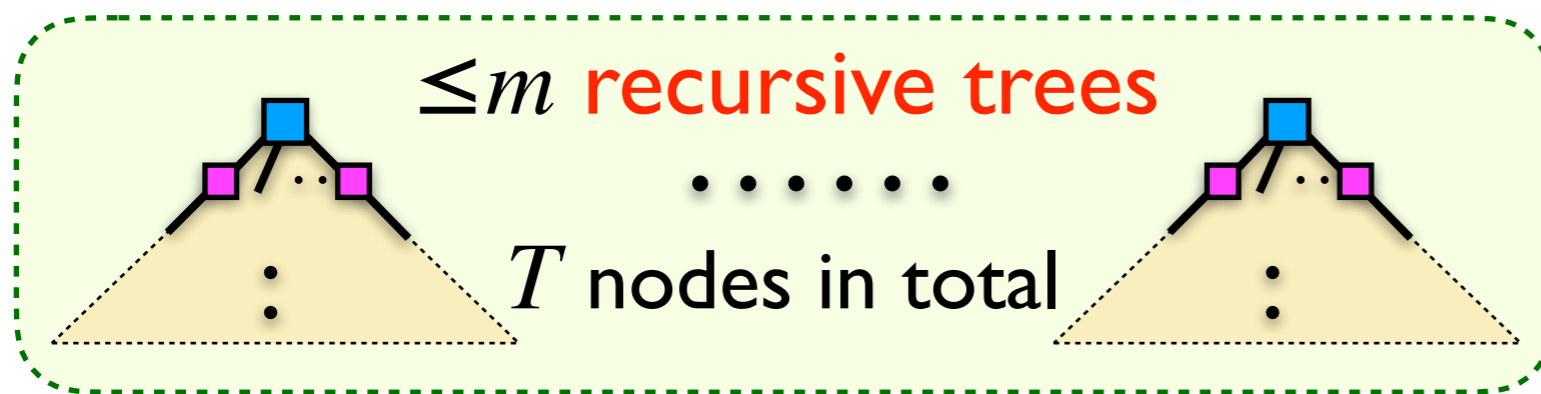
**Solve( $\phi$ )**  
 sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
 while  $\exists$  unsatisfied clause  $C$   
     **Fix( $C$ )**; *lexicographic order*

**Fix( $C$ )**  
 resample variables in  $C$  uniformly at random;  
 while  $\exists$  unsatisfied clause  $D$  intersecting  $C$   
     **Fix( $D$ )**;

$n + kT$  random bits



1-1 mapping  $\text{Enc}_\phi$



+ final assignment  
 $x_1, x_2, \dots, x_n$

represented by succinct representation:

$n$  bits

$\leq m + T (\log_2 d + O(1))$  bits

- an  $m$ -bit vector to indicate the root nodes
- $O(1)$  bits to record the stack operation for each recursive call

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

**Solve( $\phi$ )**  
 sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
 while  $\exists$  unsatisfied clause  $C$   
     **Fix( $C$ )**; *lexicographic order*

**Fix( $C$ )**  
 resample variables in  $C$  uniformly at random;  
 while  $\exists$  unsatisfied clause  $D$  intersecting  $C$   
     **Fix( $D$ )**;

$n + kT$  random bits



1-1 mapping  $\text{Enc}_\phi$

**Incompressibility Theorem** (Kolmogorov):

$N$  uniform random bits cannot be encoded to less than  $N - l$  bits with probability at least  $1 - O(2^{-l})$ .

$$\leq m + T(\log_2 d + O(1)) \text{ bits} \quad + \quad n \text{ bits}$$

w.h.p.:  $n + kT - \log_2 n \leq m + T(\log_2 d + O(1)) + n$

$\longleftrightarrow (k - \log_2 d - O(1))T \leq m + \log_2 n$



$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

**Solve**( $\phi$ )

sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
while  $\exists$  unsatisfied clause  $C$   
**Fix**( $C$ ); *lexicographic order*

**Fix**( $C$ )

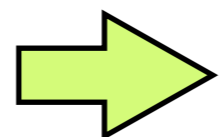
*resample* variables in  $C$  uniformly at random;  
while  $\exists$  unsatisfied clause  $D$  *intersecting*  $C$   
**Fix**( $D$ );

- $T$ : total # of calls to **Fix**( $C$ )  
(including both *top-level* and *recursive* calls)
- total cost:  $n + kT$

*w.h.p.*:  $(k - \log_2 d - O(1))T \leq m + \log_2 n$

$$d \leq 2^{k-c}$$

*for some constant  $c$*


$$T \leq m + \log_2 n$$

satisfying assignment can be found  
in time  $O(n + k(m + \log n))$  *w.h.p.*

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

**Solve**( $\phi$ )

sample a uniform random

assignment  $x_1, x_2, \dots, x_n$ ;

while  $\exists$  unsatisfied clause  $C$

**Fix**( $C$ ); *lexicographic order*

**Fix**( $C$ )

*resample* variables in  $C$  uniformly at random;

while  $\exists$  unsatisfied clause  $D$  *intersecting*  $C$

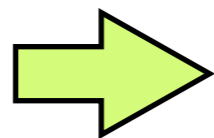
**Fix**( $D$ );

- $T$ : total # of calls to **Fix**( $C$ )  
(including both *top-level* and *recursive* calls)
- total cost:  $n + kT$

*w.h.p.*:  $(k - \log_2 d - O(1))T \leq m + \log_2 n$

**Theorem** (Moser 2009):  $\exists$  constant  $c > 0$

$$d \leq 2^{k-c}$$



satisfying assignment can be found  
in time  $O(n + km)$  *w.h.p.*

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

**Solve( $\phi$ )**

sample a uniform random

assignment  $x_1, x_2, \dots, x_n$ ;

while  $\exists$  unsatisfied clause  $C$

**Fix( $C$ )**; lexicographic order

**Fix( $C$ )**

resample variables in  $C$  uniformly at random;

while  $\exists$  unsatisfied clause  $D$  intersecting  $C$

**Fix( $D$ )**;

- $T$ : total # of calls to **Fix( $C$ )** **Why should  $T$  be finite?**  
(including both top-level and recursive calls)

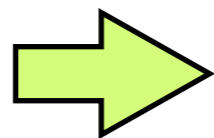
**Incompressibility Theorem** (Kolmogorov):

**Does this hold when  $N$  is random?**

$N$  uniform random bits cannot be encoded to less than  $N - l$  bits with probability at least  $1 - O(2^{-l})$ .

**Theorem** (Moser 2009):  $\exists$  constant  $c > 0$

$$d \leq 2^{k-c}$$



satisfying assignment can be found in time  $O(n + km)$  w.h.p.

$\phi$ : a  $k$ -CNF formula of degree  $d$  with  $m$  clauses on  $n$  variables

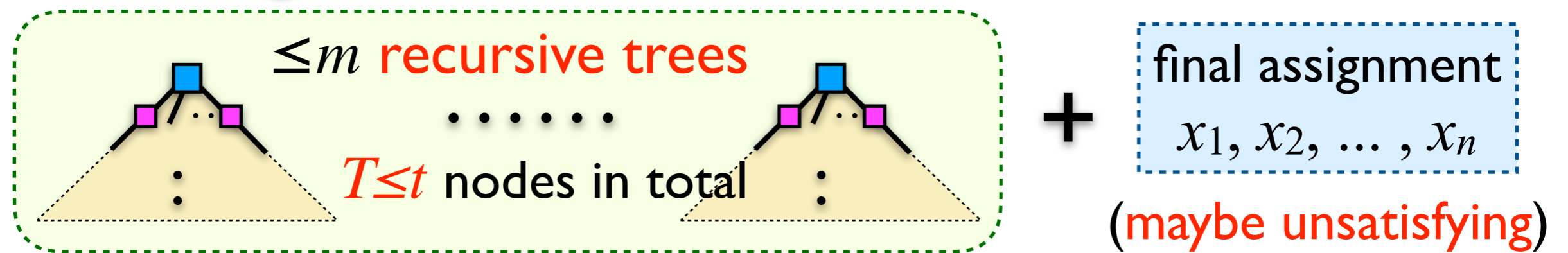
**Solve( $\phi$ )**  
 sample a uniform random assignment  $x_1, x_2, \dots, x_n$ ;  
 while  $\exists$  unsatisfied clause  $C$   
     **Fix( $C$ )**; *lexicographic order*

**Fix( $C$ )**  
 resample variables in  $C$  uniformly at random;  
 while  $\exists$  unsatisfied clause  $D$  intersecting  $C$   
     **Fix( $D$ )**;

- $n + kt$  random bits where  $t = 2(m + \log n)$  is fixed



- used as the random bits for the algorithm;
- force to terminate the algorithm if used up;



$$\leq m + T(\log_2 d + O(1)) \text{ bits} \quad + \quad n \text{ bits} \\ + (t - T)k \text{ unused random bits}$$

w.h.p.:  $(k - \log_2 d - O(1))T \leq m + \log_2 n$

$d \leq 2^{k-c}$  for some constant  $c$   $\Rightarrow T \leq m + \log_2 n$

# Algorithmic LLL

$\phi$ : a  $k$ -CNF formula of degree  $d$

The *Lovász Local Lemma* (LLL) for  $k$ -SAT:

**Theorem:**  $d \leq 2^{k-2}$   $\Rightarrow$   $\phi$  is always satisfiable

Algorithmic LLL for  $k$ -SAT:

**Theorem** (Moser 2009):  $\exists$  constant  $c > 0$   
 $d \leq 2^{k-c}$   $\Rightarrow$  satisfying assignment can be found  
in time  $O(n + km)$  w.h.p.

# The Lovász Local Lemma

$m$  bad event:  $A_1, A_2, \dots, A_m$

every  $A_i$  is independent of all but  $\leq d$  other bad events

**Lovász Local Lemma** (Lovász 1977):

$$\forall i : \Pr[A_i] \leq \frac{1}{e(d+1)} \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

**Lovász Local Lemma** (asymmetric version):

$$\exists \alpha_1, \alpha_2, \dots, \alpha_m \in [0, 1)$$
$$\forall i : \Pr[A_i] \leq \alpha_i \prod_{j \sim i} (1 - \alpha_j) \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > \prod_{i=1}^m (1 - \alpha_i)$$

$j \sim i$ :  $A_i$  and  $A_j$  are adjacent in the dependency graph

# The Lovász Local Lemma

- $n$  mutually independent random variables:  $X_1, \dots, X_n$
- $m$  bad events:  $A_1, A_2, \dots, A_m$ , determined by  $X_1, \dots, X_n$
- $\text{vbl}(A_i)$ : set of variables on which  $A_i$  is defined
- neighborhood:  $\Gamma(A_i) \triangleq \{A_j \mid j \neq i \wedge \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset\}$

**Lovász Local Lemma** (asymmetric version):

$$\exists \alpha_1, \alpha_2, \dots, \alpha_m \in [0, 1)$$

$$\forall i : \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^m \bar{A}_i \right] > \prod_{i=1}^m (1 - \alpha_i)$$

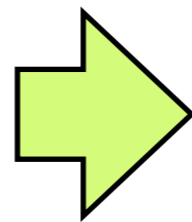
# The *Lovász Local Lemma*

- $n$  mutually independent random variables:  $X_1, \dots, X_n$
- $m$  bad events:  $A_1, A_2, \dots, A_m$ , determined by  $X_1, \dots, X_n$
- $\text{vbl}(A_i)$ : set of variables on which  $A_i$  is defined
- neighborhood:  $\Gamma(A_i) \triangleq \{A_j \mid j \neq i \wedge \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset\}$

**Lovász Local Lemma** (asymmetric version):

$\exists \alpha_1, \alpha_2, \dots, \alpha_m \in [0, 1)$

$\forall i : \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$



$\exists$  an assignment of  $X_1, \dots, X_n$  avoiding all bad events  $A_1, \dots, A_m$



# Moser-Tardos Algorithm

- $n$  mutually independent random variables:  $X_1, \dots, X_n$
- $m$  bad events:  $A_1, A_2, \dots, A_m$ , determined by  $X_1, \dots, X_n$
- $\text{vbl}(A_i)$ : set of variables on which  $A_i$  is defined
- neighborhood:  $\Gamma(A_i) \triangleq \{A_j \mid j \neq i \wedge \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset\}$

**Assumption:** The followings can be done efficiently:

- draw an independent sample of a random variable  $X_j$ .
- check whether a bad event  $A_i$  occurs on current  $X_1, \dots, X_n$ .

**Moser-Tardos Algorithm:**

sample all  $X_1, \dots, X_n$ ;

while  $\exists$  an occurring bad event  $A_i$ :

resample all  $X_j \in \text{vbl}(A_i)$ ;

- $n$  mutually independent random variables:  $X_1, \dots, X_n$
- $m$  bad events:  $A_1, A_2, \dots, A_m$ , determined by  $X_1, \dots, X_n$
- $\text{vbl}(A_i)$ : set of variables on which  $A_i$  is defined
- neighborhood:  $\Gamma(A_i) \triangleq \{A_j \mid j \neq i \wedge \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset\}$

### Moser-Tardos Algorithm:

sample all  $X_1, \dots, X_n$ ;

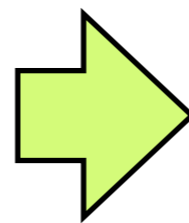
while  $\exists$  an occurring bad event  $A_i$ :

resample all  $X_j \in \text{vbl}(A_i)$ ;

### Lovász Local Lemma (Moser-Tardos 2010):

$\exists \alpha_1, \alpha_2, \dots, \alpha_m \in [0, 1)$

$\forall i : \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$



a satisfying assignment is returned within  $\sum_{i=1}^m \frac{\alpha_i}{1 - \alpha_i}$  resamples in expectation

- $n$  mutually independent random variables:  $X_1, \dots, X_n$
- $m$  bad events:  $A_1, A_2, \dots, A_m$ , determined by  $X_1, \dots, X_n$
- $\text{vbl}(A_i)$ : set of variables on which  $A_i$  is defined
- neighborhood:  $\Gamma(A_i) \triangleq \{A_j \mid j \neq i \wedge \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset\}$

### Moser-Tardos Algorithm:

sample all  $X_1, \dots, X_n$ ;

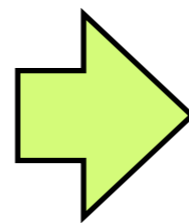
while  $\exists$  an occurring bad event  $A_i$ :

resample all  $X_j \in \text{vbl}(A_i)$ ;

### Lovász Local Lemma (Moser-Tardos 2010):

$$\forall i : \Pr[A_i] \leq \frac{1}{e(d+1)}$$

where  $d \triangleq \max_i |\Gamma(A_i)|$



a satisfying assignment is  
returned within  $m/d$   
resamples in expectation

- $n$  mutually independent random variables:  $X_1, \dots, X_n$
- $m$  bad events:  $A_1, A_2, \dots, A_m$ , determined by  $X_1, \dots, X_n$
- $\text{vbl}(A_i)$ : set of variables on which  $A_i$  is defined
- neighborhood:  $\Gamma(A_i) \triangleq \{A_j \mid j \neq i \wedge \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset\}$

### Moser-Tardos Algorithm:

sample all  $X_1, \dots, X_n$ ;

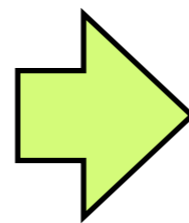
while  $\exists$  an occurring bad event  $A_i$ :

resample all  $X_j \in \text{vbl}(A_i)$ ;

### Lovász Local Lemma (Moser-Tardos 2010):

$$\forall i : \Pr[A_i] \leq \frac{1}{4d}$$

where  $d \triangleq \max_i |\Gamma(A_i)|$



a satisfying assignment is returned within  $m/(2d-1)$  resamples in expectation

# $k$ -SAT

$\phi$ : a  $k$ -CNF formula of degree  $d$

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

## Moser-Tardos Algorithm:

sample a uniform random assignment  $x_1, x_2, \dots, x_n \in \{\text{true}, \text{false}\}$ ;

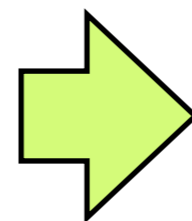
while  $\exists$  an unsatisfied clause  $C$ :

resample values of variables in  $C$  uniformly at random;

**bad event**  $A_i$ : clause  $C_i$  is **unsatisfied**

$$\forall i : \Pr[A_i] = 2^{-k} \leq \frac{1}{4d}$$

(assuming  $d \leq 2^{k-2}$ )



a satisfying assignment is returned within  $m/(2d-1)$  resamples in expectation

# $k$ -SAT

$\phi$ : a  $k$ -CNF formula of degree  $d$

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

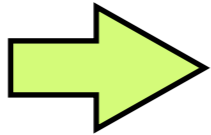
## **Moser-Tardos Algorithm:**

sample a uniform random assignment  $x_1, x_2, \dots, x_n \in \{\text{true}, \text{false}\}$ ;

while  $\exists$  an unsatisfied clause  $C$ :

resample values of variables in  $C$  uniformly at random;

**Theorem** (Moser-Tardos 2010):

$d \leq 2^{k-2}$   satisfying assignment can be found  
in time  $O(n + km/d)$  in expectation

- **mutually independent random variables:**  $\mathcal{X} \triangleq \{X_1, \dots, X_n\}$
- **bad events:**  $\mathcal{A} \triangleq \{A_1, A_2, \dots, A_m\}$
- $\forall A \in \mathcal{A}, \text{vbl}(A) \subseteq \mathcal{X}$  : set of variables determining  $A$
- **neighborhood:**  $\forall A \in \mathcal{A}, \Gamma(A) \triangleq \{B \neq A \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset\}$

### Moser-Tardos Algorithm:

sample all  $X \in \mathcal{X}$ ;

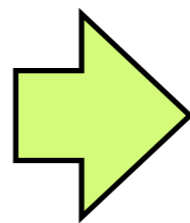
while  $\exists$  an occurring event  $A \in \mathcal{A}$  :

**resample** all  $X \in \text{vbl}(A)$ ;

### Lovász Local Lemma (Moser-Tardos 2010):

$\exists \alpha : \mathcal{A} \rightarrow [0,1)$

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



a satisfying assignment is returned within  $\sum_{A \in \mathcal{A}} \frac{\alpha_A}{1 - \alpha_A}$  resamples in expectation

**Moser-Tardos Algorithm:**sample all  $X \in \mathcal{X}$ ;while  $\exists$  an occurring event  $A \in \mathcal{A}$ :**resample** all  $X \in \text{vbl}(A)$ ;execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

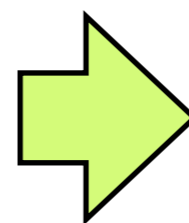
random sequence of **resampled** events

$$\forall A \in \mathcal{A}, \quad N_A \triangleq |\{i \mid \Lambda_i = A\}|$$

total # of times that  $A$  is resampled**Lovász Local Lemma (Moser-Tardos 2010):**

$$\exists \alpha : \mathcal{A} \rightarrow [0,1)$$

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



$$\forall A \in \mathcal{A} :$$

$$\mathbb{E}[N_A] \leq \frac{\alpha_A}{1 - \alpha_A}$$



### Moser-Tardos Algorithm:

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
    **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

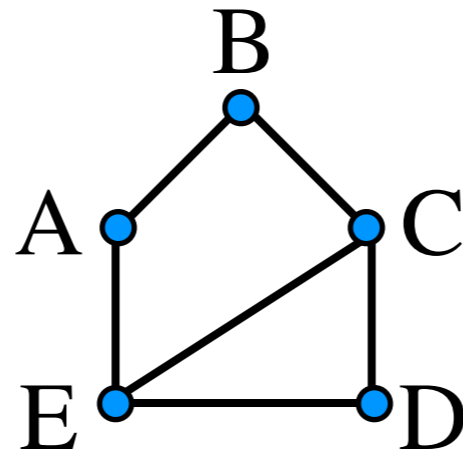
**witness tree**: A witness tree  $\tau$  is a labeled tree in which every vertex  $v$  is labeled by an event  $A_v \in \mathcal{A}$ , such that *siblings* have distinct labels.

$T(\Lambda, t)$  is a **witness tree** constructed from exe-log  $\Lambda$ :

- initially,  $T$  is a single root with label  $\Lambda_t$
- for  $i = t-1, t-2, \dots, 1$ 
  - if  $\exists$  a vertex  $v$  in  $T$  with label  $A_v \in \Gamma^+(\Lambda_i)$ 
    - add a new child  $u$  to the *deepest* such  $v$  and label it with  $\Lambda_i$
- $T(\Lambda, t)$  is the resulting  $T$

**inclusive neighborhood**:  $\Gamma^+(A) \triangleq \{B \in \mathcal{A} \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset\}$   
 $= \Gamma(A) \cup \{A\}$

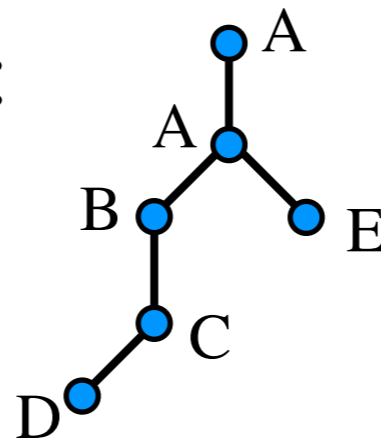
dependency graph:



exe-log  $\Lambda$ : D, C, E, D, B, A, C, A, D, ...



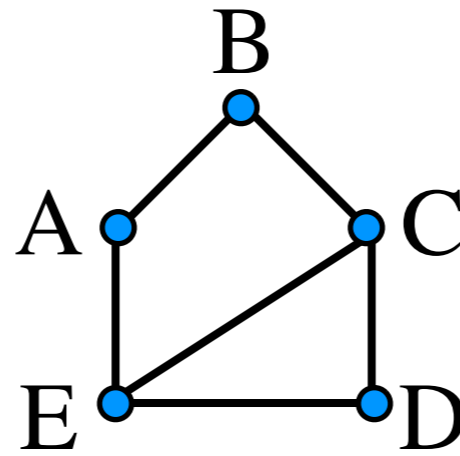
$T(\Lambda, 8)$ :



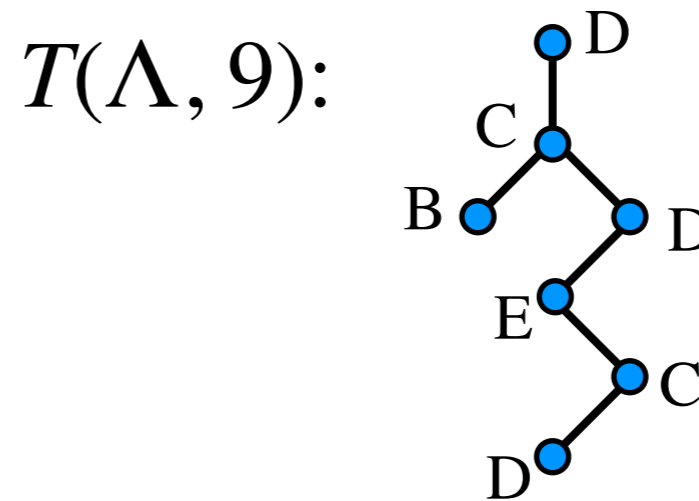
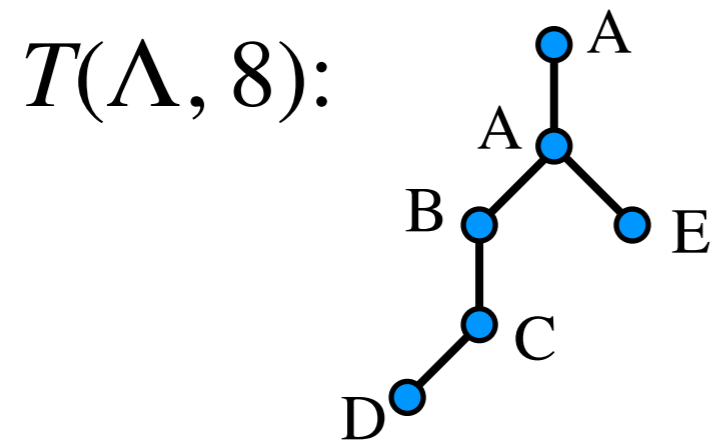
$T(\Lambda, t)$  is a **witness tree** constructed from exe-log  $\Lambda$ :

- initially,  $T$  is a single root with label  $\Lambda_t$
- for  $i = t-1, t-2, \dots, 1$ 
  - if  $\exists$  a vertex  $v$  in  $T$  with label  $A_v \in \Gamma^+(\Lambda_i)$ 
    - add a new child  $u$  to the *deepest* such  $v$  and label it with  $\Lambda_i$
- $T(\Lambda, t)$  is the resulting  $T$

dependency graph:



exe-log  $\Lambda$ : D, C, E, D, B, A, C, A, D, ...  
△



$T(\Lambda, t)$  is a **witness tree** constructed from exe-log  $\Lambda$ :

- initially,  $T$  is a single root with label  $\Lambda_t$
- for  $i = t-1, t-2, \dots, 1$ 
  - if  $\exists$  a vertex  $v$  in  $T$  with label  $A_v \in \Gamma^+(\Lambda_i)$ 
    - add a new child  $u$  to the *deepest* such  $v$  and label it with  $\Lambda_i$
- $T(\Lambda, t)$  is the resulting  $T$

### Moser-Tardos Algorithm:

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
    **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

**witness tree**: A witness tree  $\tau$  is a labeled tree in which every vertex  $v$  is labeled by an event  $A_v \in \mathcal{A}$ , such that *siblings* have distinct labels.

$T(\Lambda, t)$  is a **witness tree** constructed from exe-log  $\Lambda$ :

- initially,  $T$  is a single root with label  $\Lambda_t$
- for  $i = t-1, t-2, \dots, 1$ 
  - if  $\exists$  a vertex  $v$  in  $T$  with label  $A_v \in \Gamma^+(\Lambda_i)$ 
    - add a new child  $u$  to the *deepest* such  $v$  and label it with  $\Lambda_i$
- $T(\Lambda, t)$  is the resulting  $T$

$T(\Lambda, s) \neq T(\Lambda, t)$  if  $s \neq t$

$\mathcal{T}_A$ : set of all witness trees with root-label  $A$


$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

**Moser-Tardos Algorithm:**

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
    **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

$T(\Lambda, t)$  is a **witness tree** constructed from exe-log  $\Lambda$ :

- initially,  $T$  is a single root with label  $\Lambda_t$
- for  $i = t-1, t-2, \dots, 1$ 
  - if  $\exists$  a vertex  $v$  in  $T$  with label  $A_v \in \Gamma^+(\Lambda_i)$ 
    - add a new child  $u$  to the *deepest* such  $v$  and label it with  $\Lambda_i$
- $T(\Lambda, t)$  is the resulting  $T$

**Lemma 1** For any particular witness tree  $\tau$ :

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

**Moser-Tardos Algorithm:**

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
    **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

**Lemma 1** For any particular witness tree  $\tau$ :

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

$N_A = |\{ i \mid \Lambda_i = A \}|$  total # of times that  $A$  is resampled

$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau] \quad \mathcal{T}_A: \text{ set of all witness trees with root-label } A$$

$$\text{(lemma 1)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$$

### Moser-Tardos Algorithm:

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
    **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

LLL condition:  $\exists \alpha : \mathcal{A} \rightarrow [0, 1)$

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

$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

$\mathcal{T}_A$ : set of all witness trees  
with root-label  $A$

$$\text{(lemma 1)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$$

$$\text{(LLL cond.)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha(B)) \right]$$

$$\text{goal:} \leq \frac{\alpha_A}{1 - \alpha_A}$$

**Moser-Tardos Algorithm:**

sample all  $X \in \mathcal{X}$ ;

while  $\exists$  an occurring event  $A \in \mathcal{A}$ :

resample all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

**Lemma 1** For any particular witness tree  $\tau$ :

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

$X_i^{(t)}$  :  $t$ -th sampling of variable  $X_i \in \mathcal{X}$

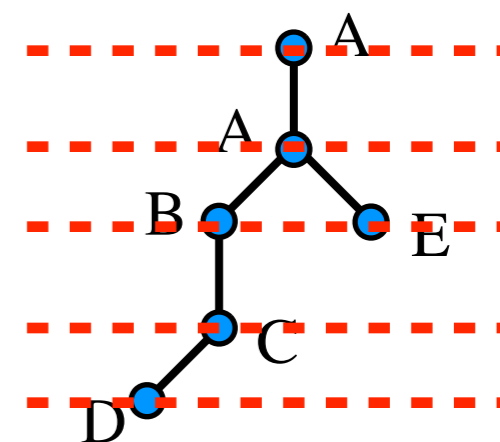
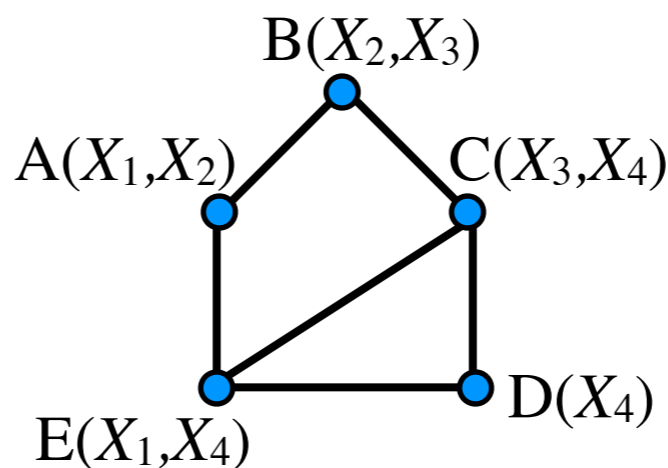
$X_1$  :  $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

$X_2$  :  $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

$X_3$  :  $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

$X_4$  :  $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

exe-log  $\Lambda$ : **D**,C,E,D,B,A,C,A,D, ...





**Moser-Tardos Algorithm:**

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
**resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

**Lemma 1** For any particular witness tree  $\tau$ :

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

$N_A = |\{ i \mid \Lambda_i = A \}|$  total # of times that  $A$  is resampled

$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau] \quad \mathcal{T}_A: \text{ set of all witness trees with root-label } A$$

$$\text{(lemma 1)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$$

### Moser-Tardos Algorithm:

sample all  $X \in \mathcal{X}$ ;  
while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
    **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

LLL condition:  $\exists \alpha : \mathcal{A} \rightarrow [0, 1)$

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

$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

$\mathcal{T}_A$ : set of all witness trees  
with root-label  $A$

$$\text{(lemma 1)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$$

$$\text{(LLL cond.)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha(B)) \right]$$

$$\text{goal:} \leq \frac{\alpha_A}{1 - \alpha_A}$$

grow a random witness tree  $T_A \in \mathcal{T}_A$  :

- initially,  $T_A$  is a single root with label  $A$
- for  $i = 1, 2, \dots$ 
  - for every vertex  $v$  at depth  $i$  (root has depth 1) in  $T_A$
  - for every  $B \in \Gamma^+(A_v)$ :
    - add a new child  $u$  to  $v$  independently with probability  $\alpha_B$ ;
    - and label it with  $B$ ;
- stop if no new child added for an entire level

**inclusive neighborhood:**  $\Gamma^+(A) \triangleq \{B \in \mathcal{A} \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset\}$   
 $= \Gamma(A) \cup \{A\}$

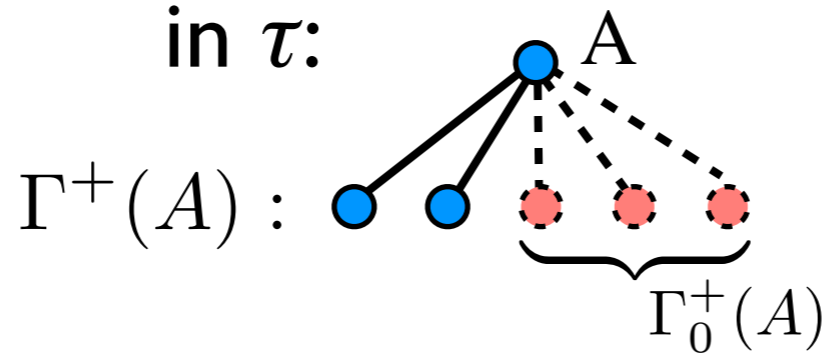
**Lemma 2** For any particular witness tree  $\tau \in \mathcal{T}_A$ :

$$\Pr[T_A = \tau] = \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha_B) \right]$$

**Lemma 2** For any particular witness tree  $\tau \in \mathcal{T}_A$ :

$$\Pr[T_A = \tau] = \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha_B) \right]$$

in  $\tau$ :



$$\begin{aligned} \Pr[T_A = \tau] &= \frac{1}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma_0^+(A_v)} (1 - \alpha_B) \right] \\ &= \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \frac{\alpha(A_v)}{1 - \alpha(A_v)} \prod_{B \in \Gamma^+(A_v)} (1 - \alpha_B) \right] \\ &= \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha_B) \right] \end{aligned}$$

**Moser-Tardos Algorithm:**

sample all  $X \in \mathcal{X}$ ;  
 while  $\exists$  an occurring event  $A \in \mathcal{A}$ :  
   **resample** all  $X \in \text{vbl}(A)$ ;

execution log  $\Lambda$ :

$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A}$$

random sequence of **resampled** events

LLL condition:  $\exists \alpha : \mathcal{A} \rightarrow [0, 1)$

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

$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

$\mathcal{T}_A$ : set of all witness trees  
with root-label  $A$

$$\text{(lemma 1)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$$

$$\text{(LLL cond.)} \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha(B)) \right]$$

$$\text{(lemma 2)} \leq \frac{\alpha_A}{1 - \alpha_A} \sum_{\tau \in \mathcal{T}_A} \Pr[T_A = \tau] \leq \frac{\alpha_A}{1 - \alpha_A}$$

- **mutually independent random variables:**  $\mathcal{X} \triangleq \{X_1, \dots, X_n\}$
- **bad events:**  $\mathcal{A} \triangleq \{A_1, A_2, \dots, A_m\}$
- $\forall A \in \mathcal{A}, \text{vbl}(A) \subseteq \mathcal{X}$  : set of variables determining  $A$
- **neighborhood:**  $\forall A \in \mathcal{A}, \Gamma(A) \triangleq \{B \neq A \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset\}$

### **Moser-Tardos Algorithm:**

sample all  $X \in \mathcal{X}$ ;

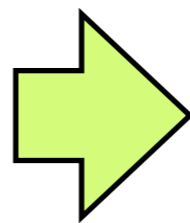
while  $\exists$  an occurring event  $A \in \mathcal{A}$  :

**resample** all  $X \in \text{vbl}(A)$ ;

### **Lovász Local Lemma (Moser-Tardos 2010):**

$\exists \alpha : \mathcal{A} \rightarrow [0,1)$

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



a satisfying assignment is returned within  $\sum_{A \in \mathcal{A}} \frac{\alpha_A}{1 - \alpha_A}$  resamples in expectation