

# An MCMC approach to the sampling Lovász local lemma

---

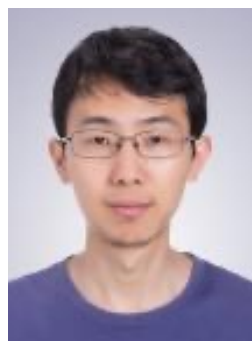
Weiming Feng

University of Edinburgh

**Base on joint works with:**



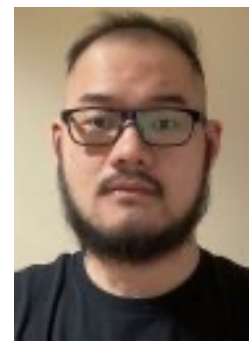
Heng Guo  
Edin



Ku He  
ICT



Chunyang Wang  
NJU



Jiaheng Wang  
Edin



Yitong Yin  
NJU



Chihao Zhang  
SJTU

DIMAP Seminar, University of Warwick, UK

23<sup>rd</sup> Nov 2022

# Conjunctive normal form (CNF)

- **Instance:** a formula  $\Phi = (V, C)$ , for example

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

clause

$V = \{x_1, x_2, x_3, x_4, x_5\}$ : set of Boolean variables;  $C$ : set of clauses.

- **SAT solutions:** an assignment of variables in  $V$  s.t.  $\Phi = \text{true}$ .
- **Computational tasks:**
  - **Decision:** Does SAT solution exist?  
NP-Complete problem [Cook 1971, Levin 1973]
  - **Counting:** How many SAT solutions?  
#P-Complete problem [Valiant 1979].

$(k, d)$ -CNF formula  $\Phi = (V, C)$

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

variable degree  $d$                       clause size  $k$

**Lovász local lemma (LLL)** [Erdős and Lovász, 1975]

SAT solution exists if  $k \gtrsim \log d$  ( $k \geq \log d + \log k + C$ )

**Algorithmic Lovász local lemma (ALLL)** [Moser and Tardos, 2009]

Find a SAT solution when  $k \gtrsim \log d$

**Sampling Lovász local lemma (SLLL)**

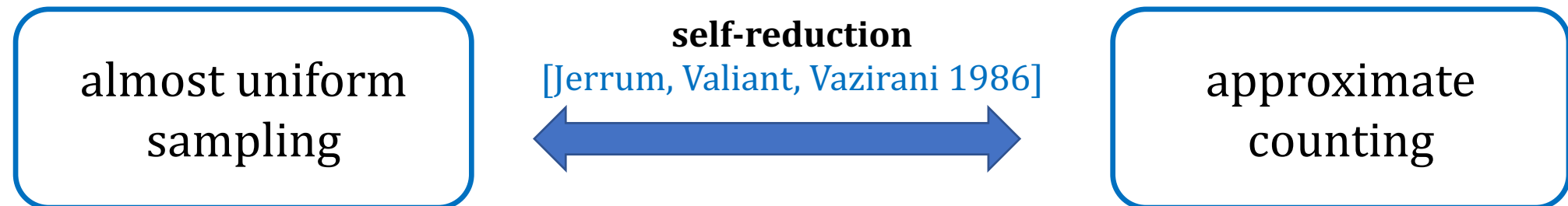
*Sample* a SAT solution uniformly at random if  $k = \Omega(\log d)$

**Counting Lovász local lemma (CLLL)**

*Approximately count* number of SAT solutions if  $k = \Omega(\log d)$

# Sampling & counting $k$ -SAT solutions

- **Input:** a  $(k, d)$ -CNF formula  $\Phi = (V, C)$  with  $|V| = n$ , an error bound  $\epsilon > 0$ .
- **Almost uniform sampling:** random solution  $\mathbf{X} \in \{\text{true}, \text{false}\}^V$  s.t.  
the *total variation distance*  $d_{TV}(\mathbf{X}, \mu) \leq \epsilon$ ,  
 $\mu$ : the uniform distribution of all SAT solutions
- **Approximate counting:** estimate the number of SAT solutions, output  
 $(1 - \epsilon)Z \leq \hat{Z} \leq (1 + \epsilon)Z$ ,  
 $Z$  = the number of SAT solutions



Work	Regime	Running time or lower bound	Technique
HSZ19	Monotone CNF <sup>[1]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n \log n$	Markov chain Monte Carlo (MCMC)

[1] *Monotone CNF*: all variables appear **positively**  $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (x_3 \vee x_4 \vee x_6)$ .

Work	Regime	Running time or lower bound	Technique
HSZ19	Monotone CNF <sup>[1]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n \log n$	Markov chain Monte Carlo (MCMC)
GJL17	$s \geq \min(\log dk, k/2)$ <sup>[2]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n$	partial rejection sampling

[1] *Monotone CNF*: all variables appear **positively**  $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (x_3 \vee x_4 \vee x_6)$ .

[2]  $s$ : two dependent clauses share **at least**  $s$  variables.

Work	Regime	Running time or lower bound	Technique
HSZ19	Monotone CNF <sup>[1]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n \log n$	Markov chain Monte Carlo (MCMC)
GJL17	$s \geq \min(\log dk, k/2)$ <sup>[2]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n$	partial rejection sampling
Moitra'17	$k \gtrsim 60 \log d$	$n^{\text{poly}(dk)}$	linear programming

[1] *Monotone CNF*: all variables appear **positively**  $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (x_3 \vee x_4 \vee x_6)$ .

[2]  $s$ : two dependent clauses share **at least**  $s$  variables.

Work	Regime	Running time or lower bound	Technique
HSZ19	Monotone CNF <sup>[1]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n \log n$	Markov chain Monte Carlo (MCMC)
GJL17	$s \geq \min(\log dk, k/2)$ <sup>[2]</sup> $k \gtrsim 2 \log d$	$\text{poly}(dk)n$	partial rejection sampling
Moitra'17	$k \gtrsim 60 \log d$	$n^{\text{poly}(dk)}$	linear programming
BGGGŠ15	$k \leq 2 \log d - C$	<b>NP-hard</b>	-

[1] *Monotone CNF*: all variables appear **positively**  $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (x_3 \vee x_4 \vee x_6)$ .

[2]  $s$ : two dependent clauses share **at least**  $s$  variables.

## Open Problem

fast sampling algorithm when  $k \gtrsim 2 \log d$



## **Our Results** [F, Guo, Yin and Zhang, 2020 & F, He and Yin, 2021 ]

For any  $(k, d)$ -CNF formula satisfying  $k \gtrsim 13 \log d$ ,

- ***sampling algorithm (main algorithm)***  
draw almost uniform random SAT solution in time  $\tilde{O}(d^2 k^3 n^{1.001})$ ;
- ***counting algorithm (by reduction)***  
count SAT solutions approximately in time  $\tilde{O}(d^3 k^3 n^{2.001})$ .

## **Further Improvements on our algorithm**

- Better analysis [Jain, Pham and Vuong, 2021]: improve to  $k \gtrsim 5.741 \log d$ .
- Better accuracy [He, Sun and Wu, 2021]: perfect sampling via CTFP

## **State-of-the-art** [He, Wang and Yin, 2022]

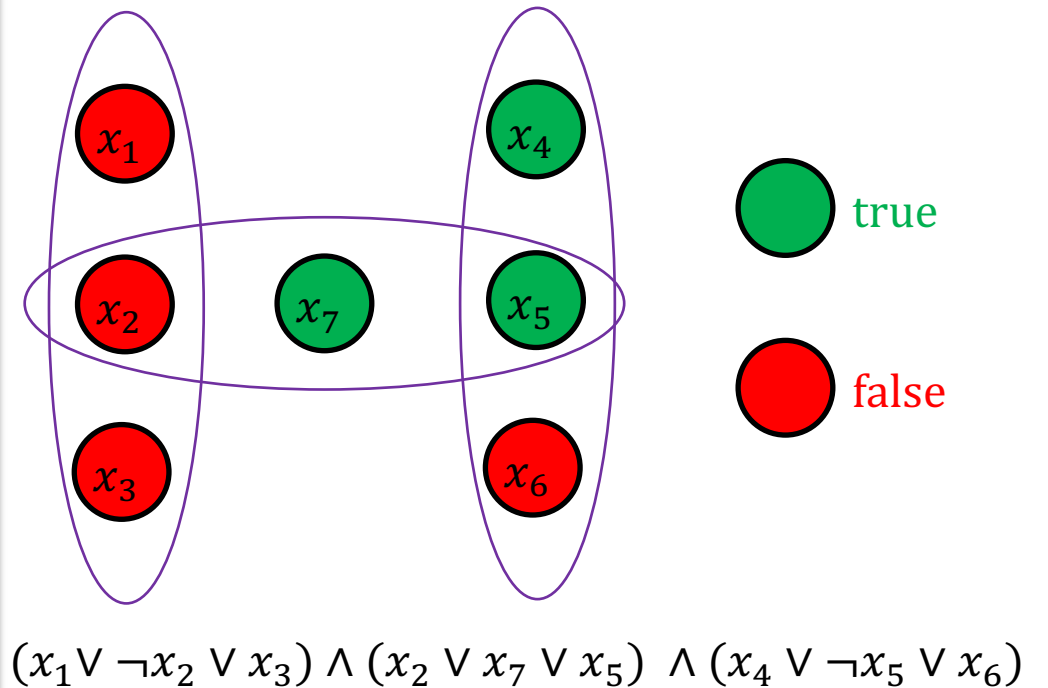
- $k \gtrsim 5 \log d$  (non-MCMC sampling algorithm)

# Glauber dynamics (Gibbs sampling)

Start from an arbitrary solution  $Y \in \{T, F\}^V$ ;

**For** each  $t$  from 1 to  $T$  **do**

- Pick  $v \in V$  uniformly at random;
- Resample  $Y_v \sim (\cdot \mid Y_{V \setminus v})$ ;

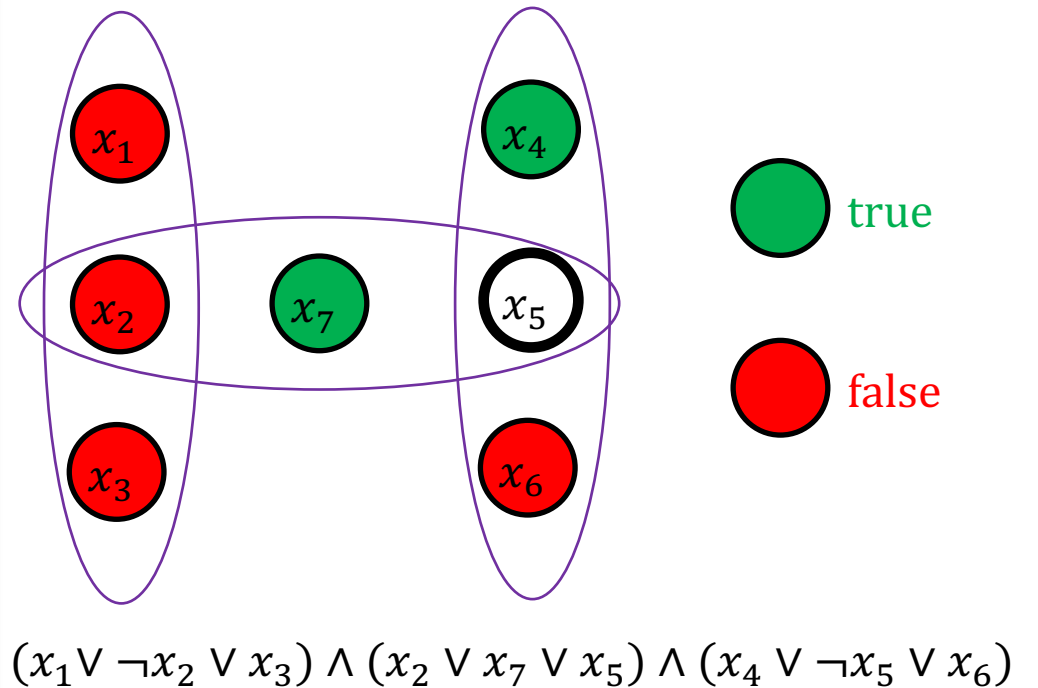


# Glauber dynamics (Gibbs sampling)

Start from an arbitrary solution  $Y \in \{T, F\}^V$ ;

**For** each  $t$  from 1 to  $T$  **do**

- Pick  $v \in V$  uniformly at random;
- Resample  $Y_v \sim (\cdot \mid Y_{V \setminus v})$ ;

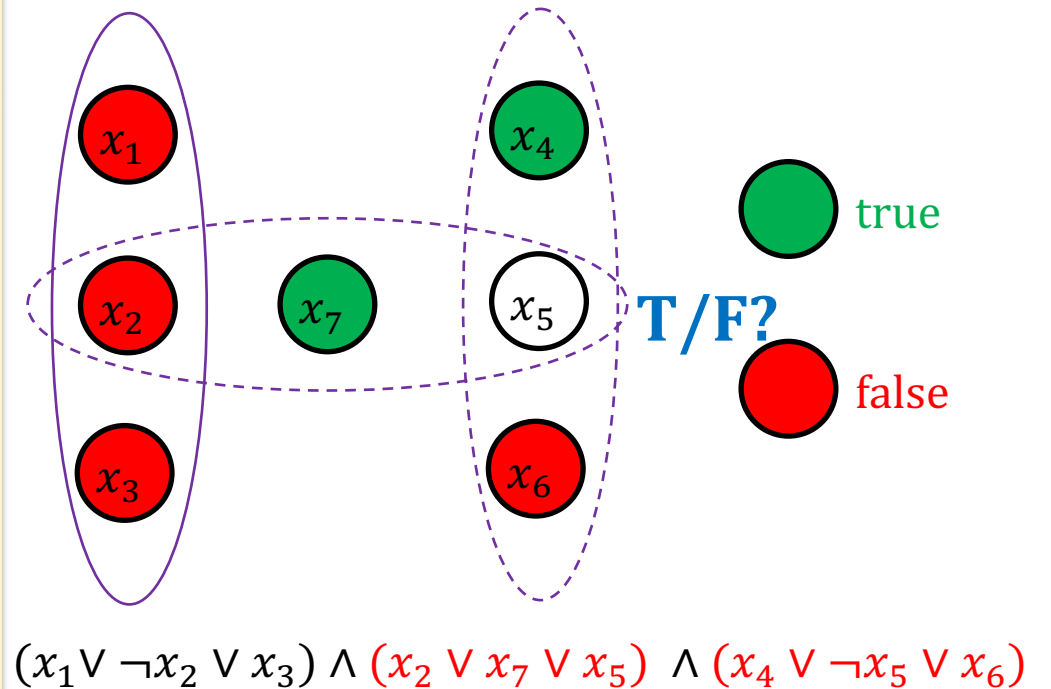


# Glauber dynamics (Gibbs sampling)

Start from an arbitrary solution  $Y \in \{T, F\}^V$ ;

**For** each  $t$  from 1 to  $T$  **do**

- Pick  $v \in V$  uniformly at random;
- Resample  $Y_v \sim \mu_v(\cdot \mid Y_{V \setminus v})$ ;

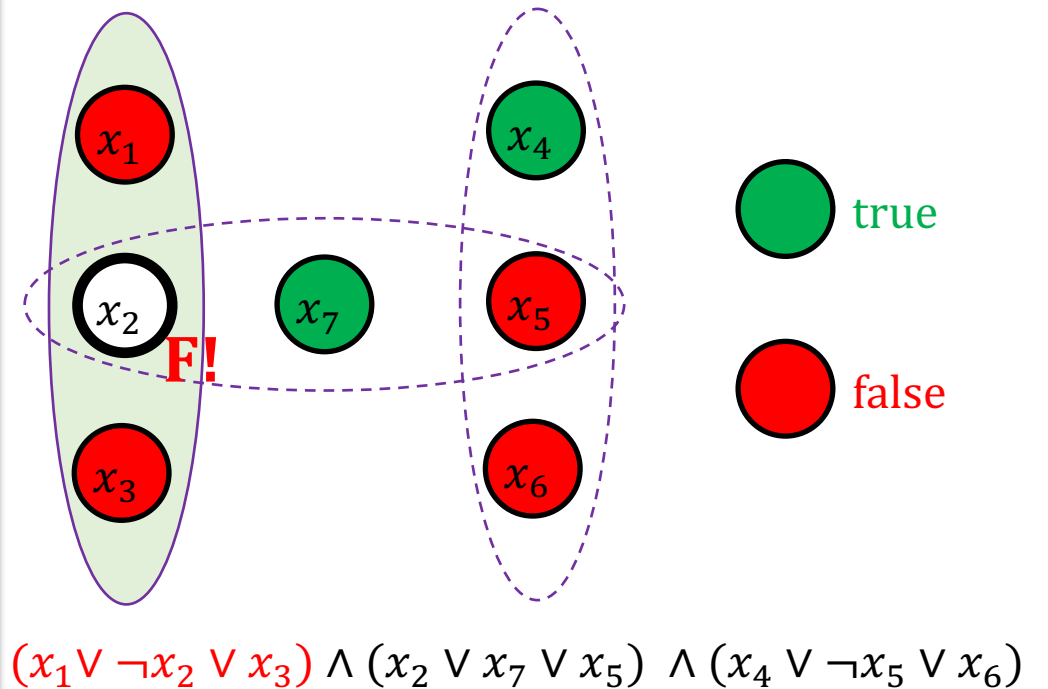


# Glauber dynamics (Gibbs sampling)

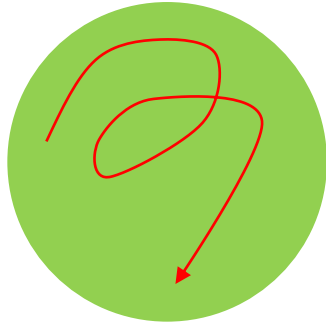
Start from an arbitrary solution  $Y \in \{T, F\}^V$ ;

**For** each  $t$  from 1 to  $T$  **do**

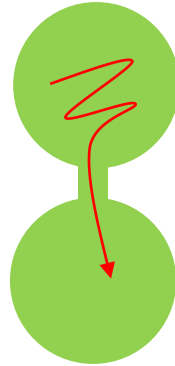
- Pick  $v \in V$  uniformly at random;
- Resample  $Y_v \sim \mu_v(\cdot \mid Y_{V \setminus v})$ ;



**Glauber dynamics:** *random walk* over solution space via *local update*.



rapid mixing



slow mixing



# Connectivity barrier (toy example)

- $(k, d)$ -CNF formula  $\Phi = (V, C)$  with  $V = \{x_1, x_2, \dots, x_k\}$ :

$$\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_k.$$

$$C_1 = (\neg x_1 \vee x_2 \vee x_3 \vee \dots \vee x_k) \text{ forbids } 100 \dots 0$$

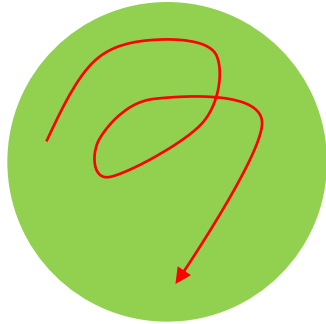
$$C_2 = (x_1 \vee \neg x_2 \vee x_3 \vee \dots \vee x_k) \text{ forbids } 010 \dots 0$$

$$C_k = (x_1 \vee x_2 \vee x_3 \vee \dots \vee \neg x_k) \text{ forbids } 000 \dots 1$$

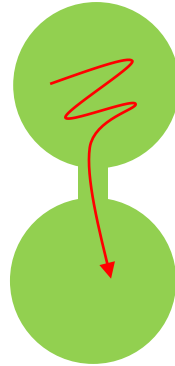
- Any assignment  $X \in \{0,1\}^V$  with  $\|X\|_1 = 1$  is **infeasible**.
- All false solution **0** is **disconnected** with others.



Glauber dynamics: *random walk* over solution space via *local update*.



rapid mixing



slow mixing



### Question for SLLL

Can we obtain *fast* sampling algorithm  
when the solution space is *disconnected*?



# Our technique: projection



Source: [https://www.shadowmatic.com/presskit/images/IMG\\_0650.png](https://www.shadowmatic.com/presskit/images/IMG_0650.png)

Projecting from a high dimension to a lower dimension to improve connectivity

Construct a *good subset* of variables  $S \subseteq V$

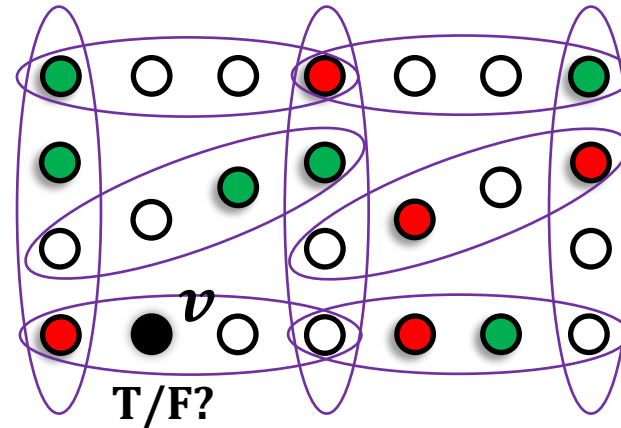
Run *Glauber dynamics* on *projected distribution*  $\mu_S$  to draw sample  $X \sim \mu_S$

Start from a uniform random  $X \in \{\text{true}, \text{false}\}^S$ ;

**For** each  $t$  from 1 to  $T$

- Pick a variable  $v \in S$  uniformly at random;
- Resample  $X_v \sim \mu_v(\cdot | X_{S \setminus v})$ ;

**Return**  $X$ ;



Draw sample  $Y \sim \mu_{V \setminus S}(\cdot | X)$  from the *conditional distribution*

There exists an *efficiently constructible subset*  $S \subseteq V$  such that:

- the Glauber dynamics on  $\mu_S$  is *rapidly mixing*,
- the Glauber dynamics on  $\mu_S$  can be *implemented efficiently* (draw  $X_v \sim \mu_v(\cdot | X_{S \setminus v})$ ),
- sampling assignment for  $V \setminus S$  can be *implemented efficiently* (draw  $Y \sim \mu_{V \setminus S}(\cdot | X)$ ).

computing exact distr.  
can be #P-hard

Construct a *good subset* of variables  $S \subseteq V$

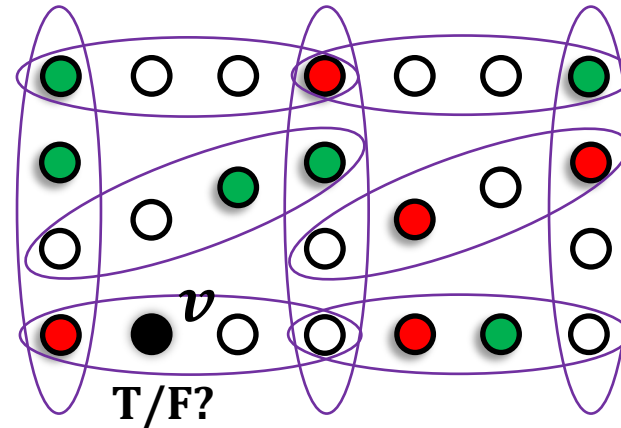
Run *Glauber dynamics* on *projected distribution*  $\mu_S$  to draw sample  $X \sim \mu_S$

Start from a uniform random  $X \in \{\text{true}, \text{false}\}^S$ ;

**For** each  $t$  from 1 to  $T$

- Pick a variable  $v \in S$  uniformly at random;
- Resample  $X_v \sim \mu_v(\cdot | X_{S \setminus v})$ ;

**Return**  $X$ ;



Draw sample  $Y \sim \mu_{V \setminus S}(\cdot | X)$  from the *conditional distribution*

## Our Tasks:

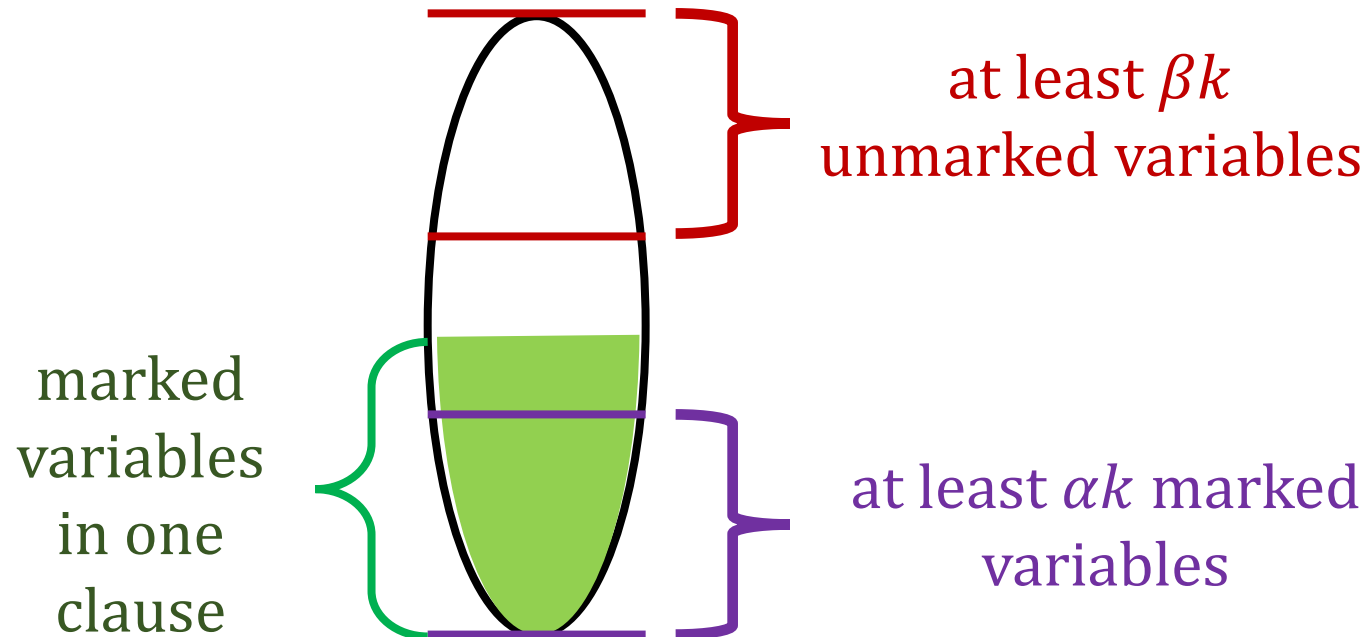
- Construct such a *good subset*  $S \subseteq V$ .
- Show that the Glauber dynamics on  $\mu_S$  is *rapidly mixing*.
- Given assignment on  $S$ , draw samples *efficiently* from the conditional distribution.

# Mark variables [Moitra' 17]

Mark a set of variables  $S \subseteq V$  such that

- each clause contains **at least**  $\alpha k = \Theta(k)$  marked variables;
- each clause contains **at least**  $\beta k = \Theta(k)$  unmarked variables;

where  $0 < \alpha, \beta < 1$  are two constant parameters with  $\alpha + \beta < 1$



Marked set  $S \subseteq V$   
is constructed by  
**algorithmic LLL**  
(Moser-Tardos)

# The rapid mixing of Glauber dynamics on $\mu_S$

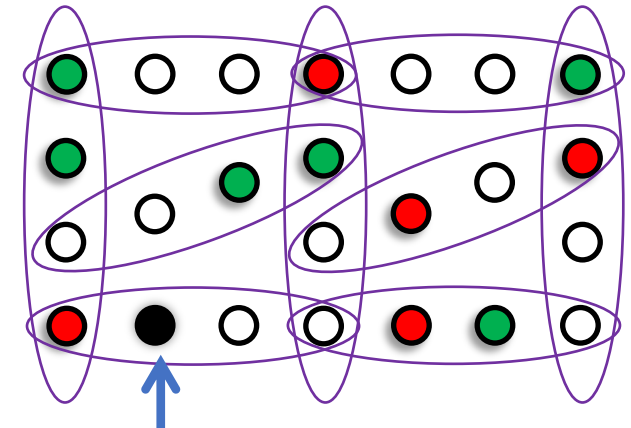
Each clause has **at least**  $\beta k$  **unmarked variables**

*by LLL* ↓

## Key Property: local uniformity

For any  $v \in S$ , any  $X_{S \setminus v} \in \{T, F\}^{S \setminus v}$ ,

$$\forall c \in \{T, F\}, \quad \mu_v(c \mid X_{S \setminus v}) = \frac{1}{2} \left( 1 + \frac{1}{\text{poly}(dk)} \right)$$

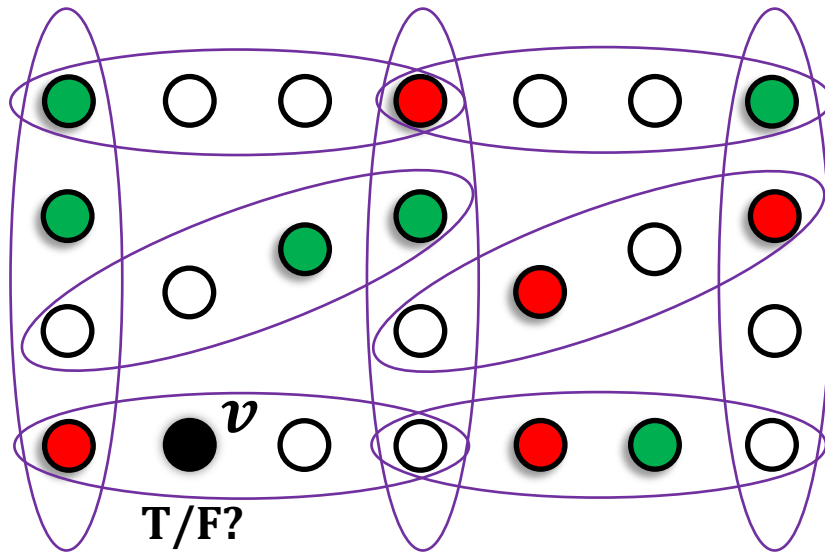


- The Glauber dynamics is **connected** ( $\mu_S(X_S) > 0$  for all  $X_S \in \{T, F\}^S$ )
- The Glauber dynamics **mixes** in  $O(n \log n)$  steps
  - path coupling analysis [F, Guo, Yin and Zhang, 2020] [F, He and Yin, 2021]
  - information percolation analysis [Jain, Pham and Vuong, 2021]

# Implementation of the algorithm

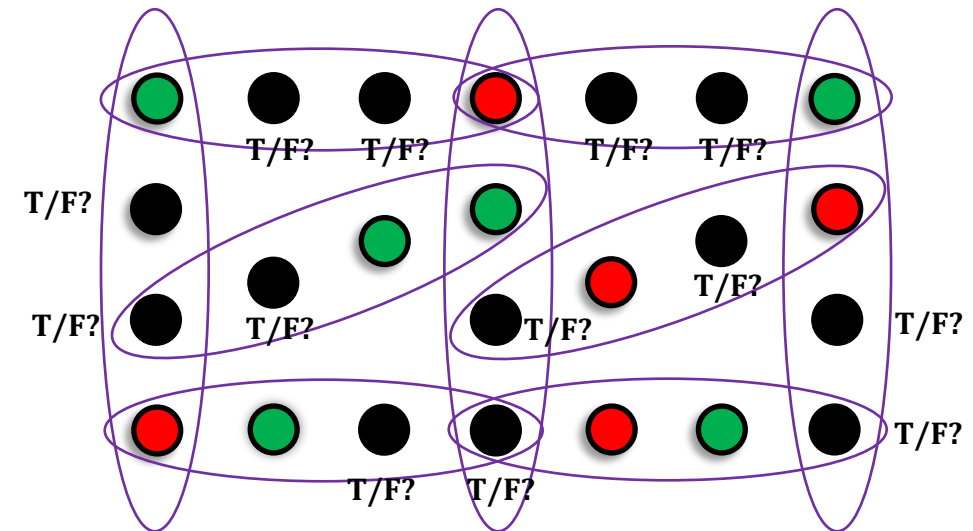
## Transition of Glauber dynamics

resample  $X_v \sim \mu_v(\cdot | X_{S \setminus v})$



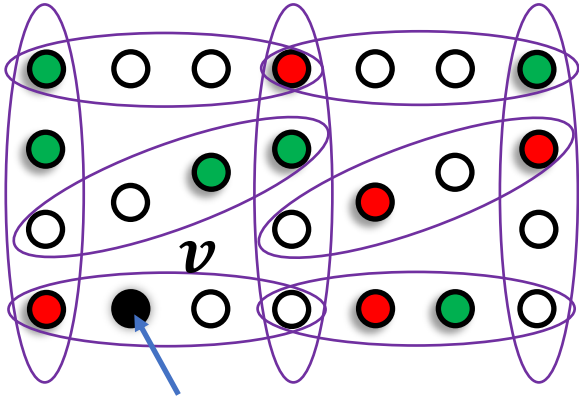
## Sample unmarked variable

sample  $Y \sim \mu_{V \setminus S}(\cdot | X)$



**Challenge:** computing the *exact* conditional distributions can be **#P-hard**.

**Solution:** draw *approximate* samples via rejection sampling

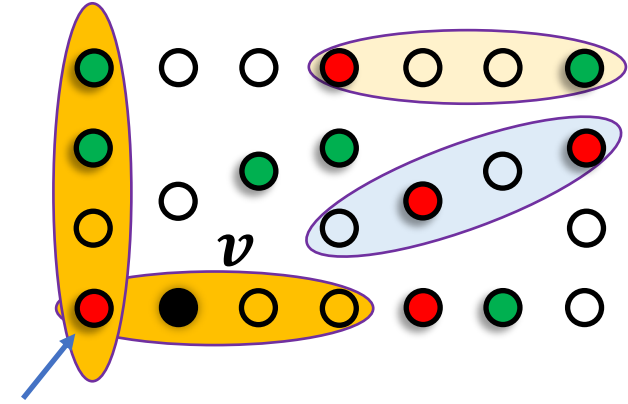


resample  $X_v$  from  $\mu_v(\cdot | X_{S \setminus v})$

*remove satisfied clauses*

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

$$x_1 = \text{T or } x_4 = \text{F} \quad \checkmark$$



$C$ : connected component containing  $v$

each clause has  
**at least**  $\alpha k$  **marked variables**

**local uniformity**

each marked variables takes  
an almost uniform random value

remove  
each clause  
with prob.  
 $\approx 1 - \left(\frac{1}{2}\right)^{\alpha k}$

with high probability  
component size is  
 $O(\log n)$

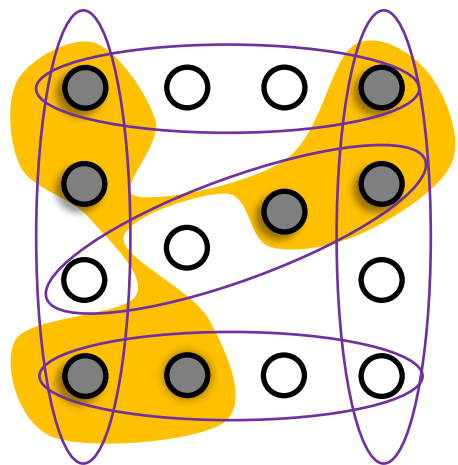
**rejection sampling**  
on component

**Input:** a  $k$ -CNF formula  $\Phi = (V, E)$  with maximum degree  $d$ , an error bound  $\epsilon > 0$ .

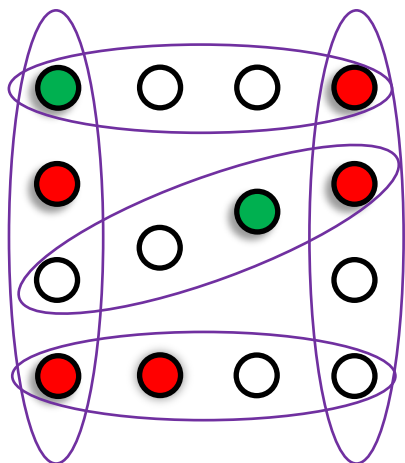
**Output:** a random sample  $\sigma \in \{\text{true}, \text{false}\}^V$  s.t.  $d_{TV}(\sigma, \mu) \leq \epsilon$ .

1. Run *Moser-Tardos* algorithm to construct marked set  $S \subseteq V$ ;
2. Run *Glauber dynamics* on  $\mu_S$  for  $O\left(n \log \frac{n}{\epsilon}\right)$  steps to sample  $X \sim \mu_S$ ;  
(implemented using *rejection sampling*)
3. Run *rejection sampling* to draw  $Y \sim \mu_{V \setminus S}(\cdot | X)$ ;
4. Return  $X \cup Y$ .



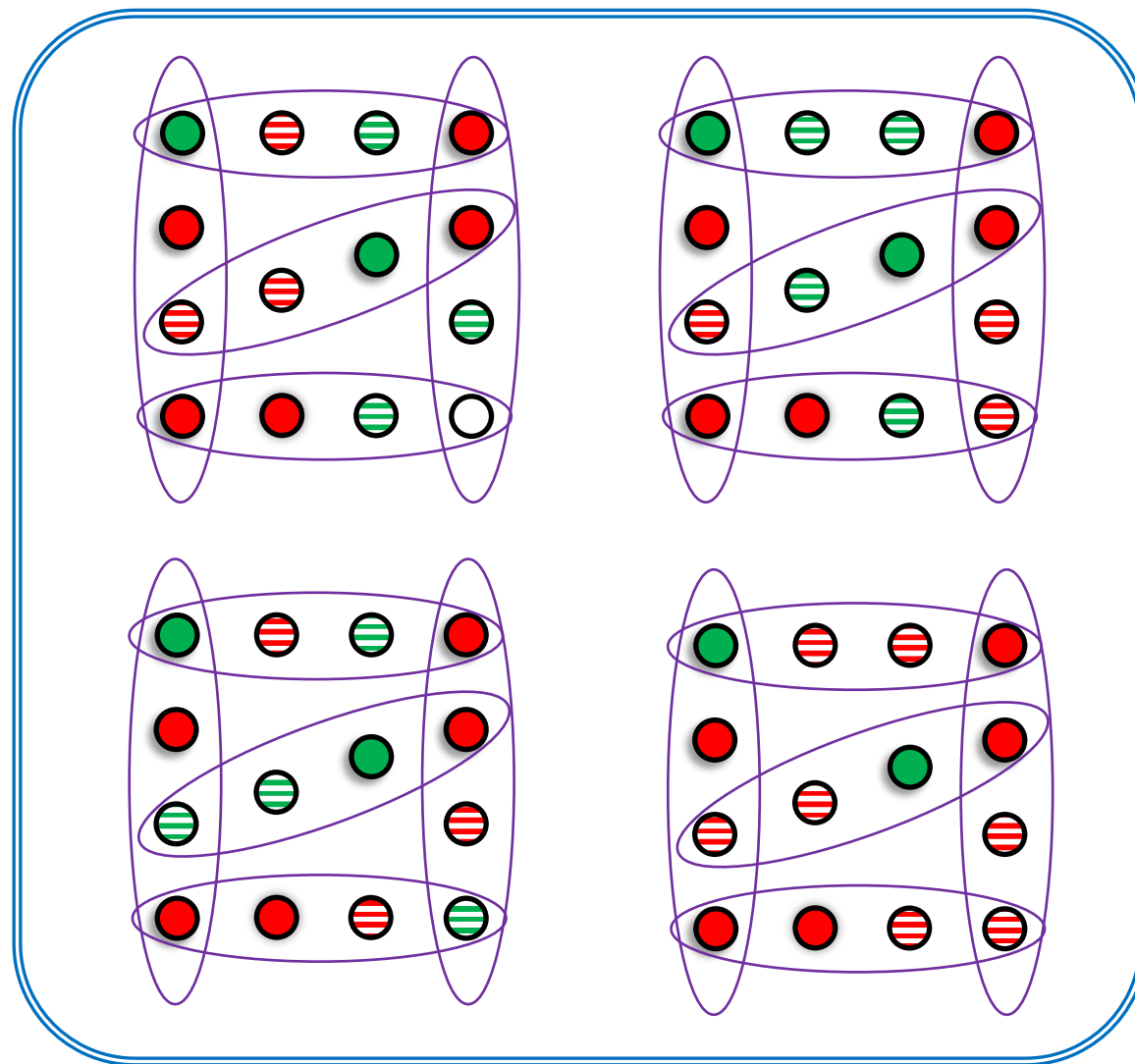


set of marked variables  $S \subseteq V$



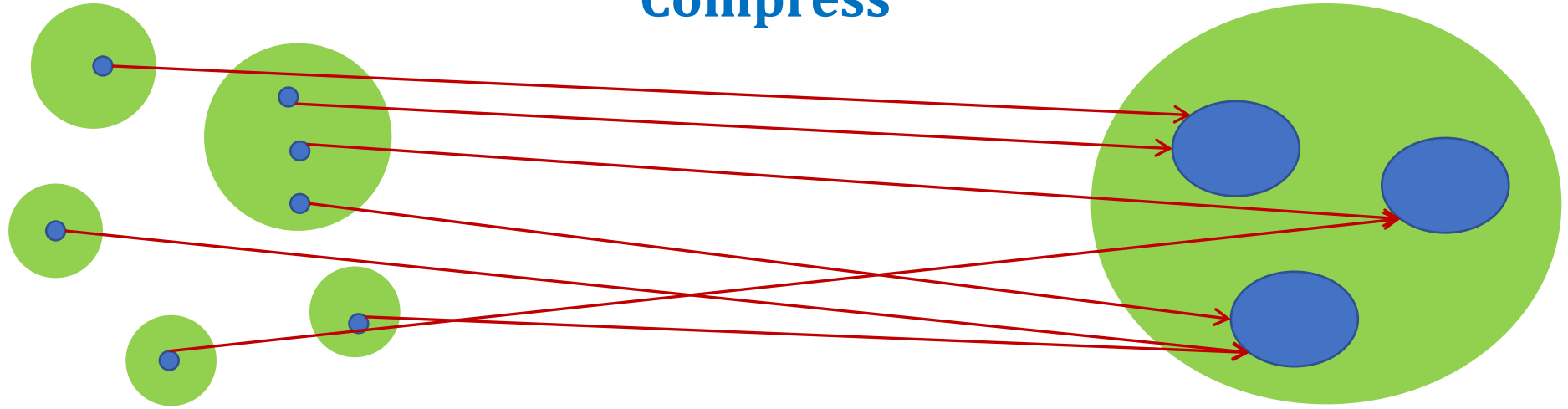
an assignment  $X \in \{T, F\}^S$

*compress*



a set of assignment  $Y \in \{T, F\}^V$  with  $Y_S = X$

**Compress**

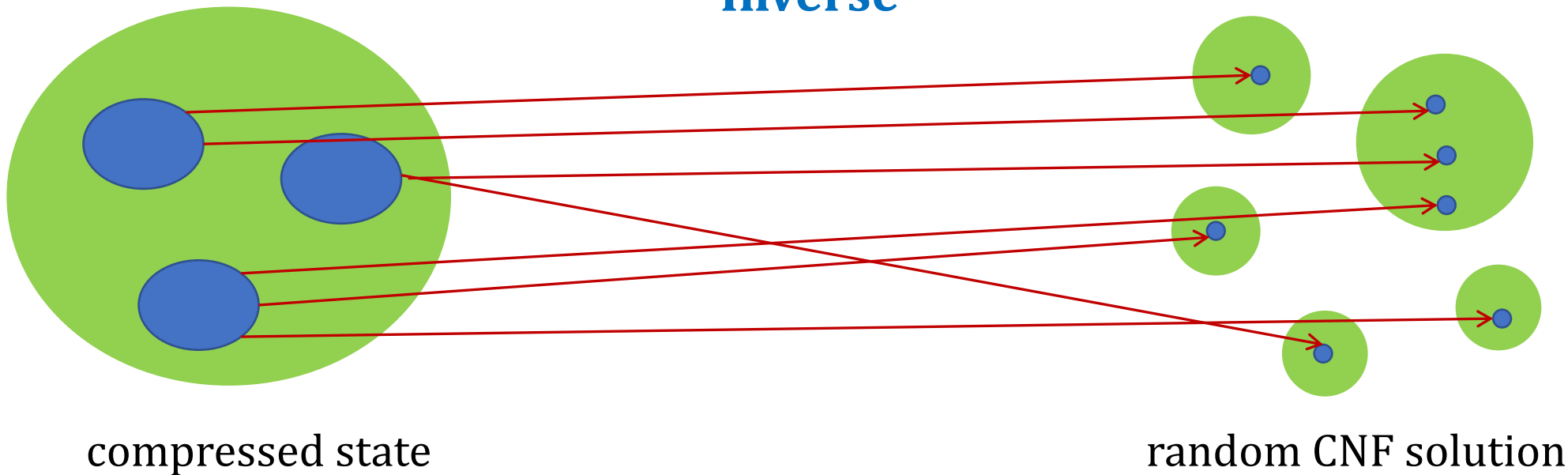


CNF solution space: **disconnected**

compressed space: **connected**

**Rapid mixing of Glauber dynamics**

**Inverse**



**Fast implementation of algorithm**

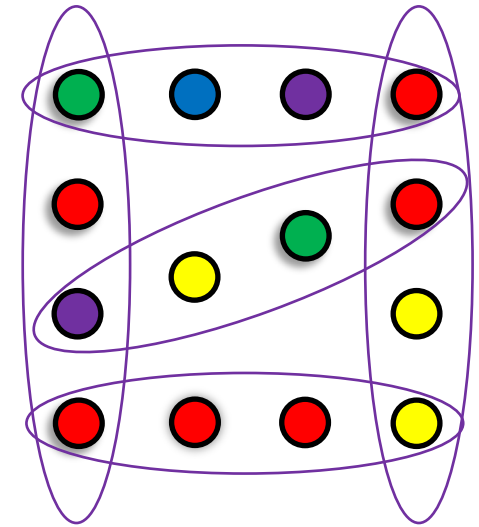
# Hypergraph colouring

**Instance:** a  $k$ -uniform hypergraph  $H(V, E)$  with max degree  $d$  and  $q$  colours

- each hyperedge contains  $k$  vertices
- each vertex belongs to  $\leq d$  hyperedges

**Hypergraph colouring:**  $X \in [q]^V$  assign  $v \in V$  a colour  $X_v$

- no hyperedge is monochromatic



## Lovász Local lemma and algorithmic LLL

- find a hypergraph colouring when  $q \gtrsim d^{1/k}$  ( $q \geq C_k d^{1/(k-1)}$ )

## Sampling Lovász Local lemma

- Sample a uniform hypergraph colouring in the local lemma regime

Work	Regime	Running time or lower bound	Technique
FM19	Linear hypergraph <sup>[1]</sup> $q \gtrsim \max\{\log n, d^{1/k}\}$	$O(n \log n)$	Markov chain Monte Carlo (MCMC)

[1] *Linear hypergraph*: for all distinct hyperedge  $e_1, e_2 \in E$ ,  $|e_1 \cap e_2| \leq 1$

Work	Regime	Running time or lower bound	Technique
FM19	Linear hypergraph <sup>[1]</sup> $q \gtrsim \max\{\log n, d^{1/k}\}$	$O(n \log n)$	Markov chain Monte Carlo (MCMC)
GLLZ17	$q \gtrsim d^{16/k}$	$n^{\text{poly}(dk \log q)}$	linear programming

[1] *Linear hypergraph*: for all distinct hyperedge  $e_1, e_2 \in E$ ,  $|e_1 \cap e_2| \leq 1$

Work	Regime	Running time or lower bound	Technique
FM19	Linear hypergraph <sup>[1]</sup> $q \gtrsim \max\{\log n, d^{1/k}\}$	$O(n \log n)$	Markov chain Monte Carlo (MCMC)
GLLZ17	$q \gtrsim d^{16/k}$	$n^{\text{poly}(dk \log q)}$	linear programming
GGW22	$q \lesssim d^{2/k}$	<b>NP-hard</b>	-
GGW22	Linear hypergraph $q \lesssim d^{1/k}$	<b>NP-hard</b>	

[1] *Linear hypergraph*: for all distinct hyperedge  $e_1, e_2 \in E, |e_1 \cap e_2| \leq 1$

### Open Problem

fast sampling algorithm when  $q \gtrsim d^{2/k}$  (general) and  $q \gtrsim d^{1/k}$  (linear)



# Results obtained by MCMC with compression

## **MCMC with compression** [F., He and Yin, 2021]

- $\tilde{O}(\text{poly}(dk) \cdot n^{1.001})$  running time if  $q \gtrsim d^{9/k}$

## **Improved analysis on general hypergraph** [Jain, Pham and Vuong, 2021]

- $\tilde{O}(\text{poly}(dk) \cdot n^{1.001})$  running time if  $q \gtrsim d^{3/k}$

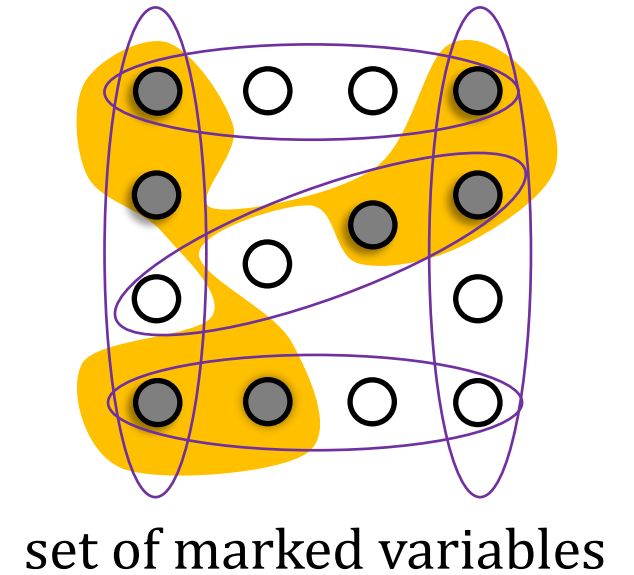
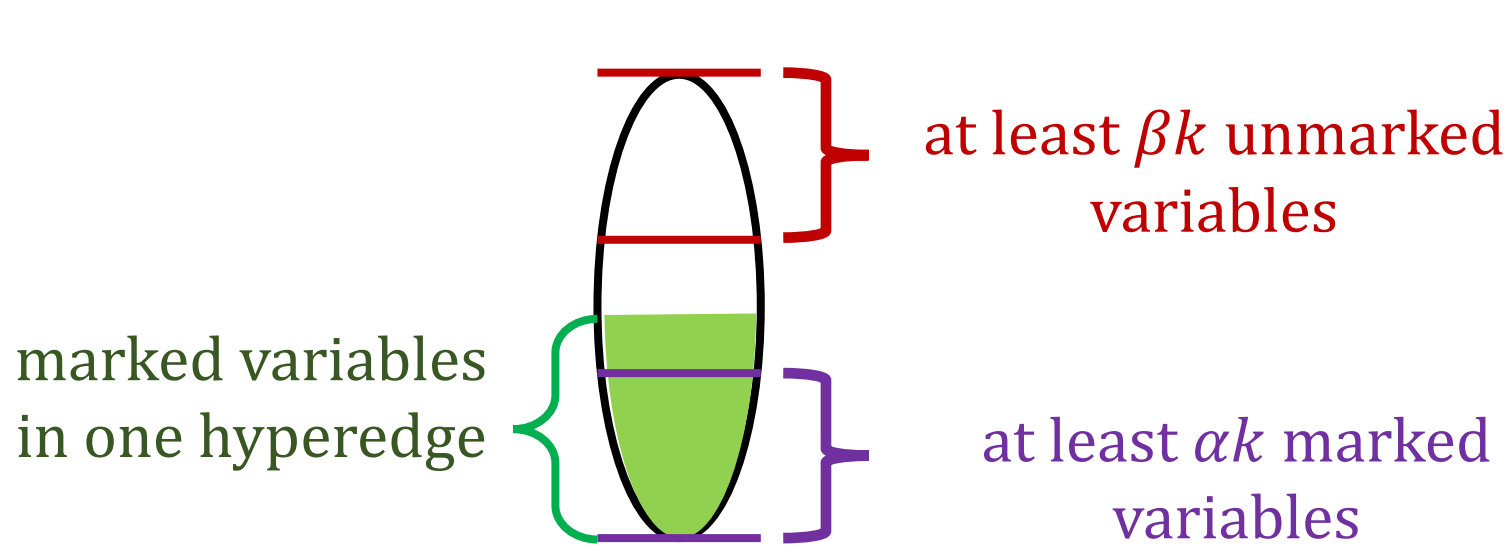
## **Improved analysis on linear hypergraph** [F., Guo and Wang, 2022]

- $\tilde{O}(\text{poly}(dk) \cdot n^{1.001})$  running time if  $q \gtrsim d^{(2+\delta)/k}$  for any constant  $\delta > 0$

## **Perfect sampling via CFTP** [He, Sun and Wu, 2021]

- $\tilde{O}(\text{poly}(dk) \cdot n)$  expected running time if  $q \gtrsim d^{3/k}$

# Mark/unmarked paradigm

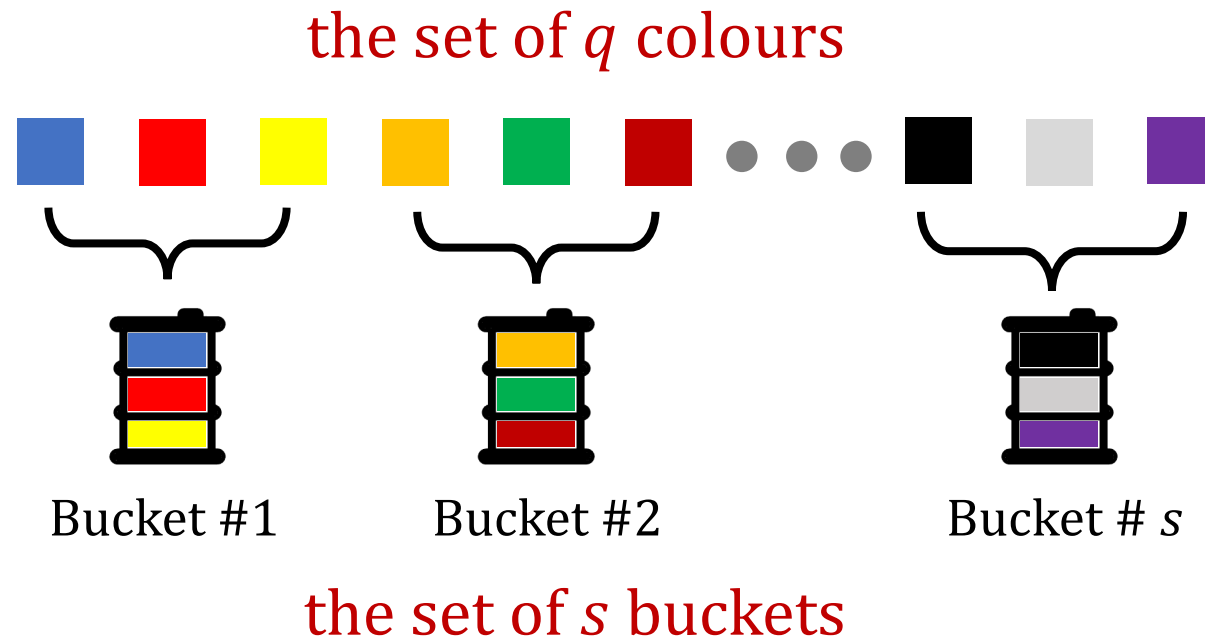


By **Lovász Local lemma**,  
such marked set exists if  
 $k = \Omega(\log d)$

**Local lemma regime:**

- CNF:  $k = \Omega(\log d)$  ✓
- Hypergraph colouring:  $q = \Delta^{\Omega(1/k)}$  ✗

# State compression for hypergraph colouring



## *Balanced projection scheme*

$$h: [q] \rightarrow [s]$$

for any  $j \in [s]$ ,

$$|h^{-1}(j)| = \frac{q}{s} \pm 1$$

$$s = q^\gamma \text{ for constant } 0 < \gamma < 1$$

## Projection of colouring

for any hypergraph colouring  $X \in [q]^V$ ,

$$Y = h(X) \text{ s.t. } Y_v = h(X_v) \text{ for all } v \in V$$

**Compression:** *different* colouring  $X \in [q]^V$  may be mapped to the *same*  $Y \in [s]^V$

Distribution  $\pi = \pi_h$  over the **compressed space**  $[s]^V$

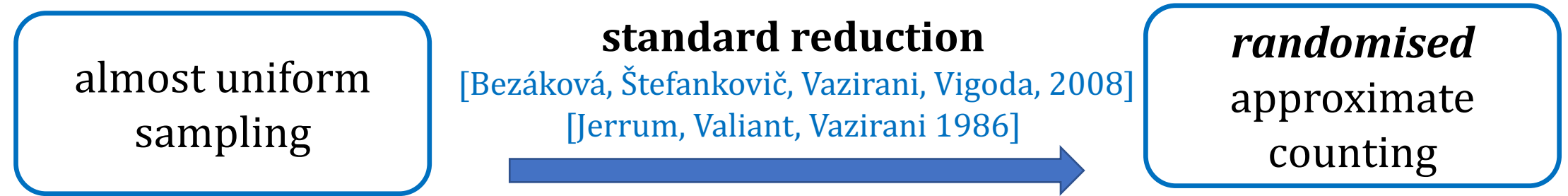
$$h(X) \sim \pi \text{ if } X \sim \mu,$$

$\mu$  is the **uniform** distribution over all hypergraph colourings

### Sampling algorithm for hypergraph colourings

1. Choose a proper  $s = q^V$  to define the balanced projection scheme  $h$ ;
2. Run **Glauber dynamics** on  $\pi_h$  for  $O(n \log n)$  steps to sample  $Y \sim \pi_h$ ;  
(implemented using **rejection sampling**)
3. Run **rejection sampling** to draw  $X \in h^{-1}(Y)$  uniformly at random;
4. Return  $X$ .

# Deterministic approximate counting



***Randomised counting:*** **with probability at least  $2/3$** , output  $\hat{Z}$  satisfying

$$(1 - \epsilon)Z \leq \hat{Z} \leq (1 + \epsilon)Z,$$

$Z$  total number of solutions (say total number of hypergraph colourings)

***Deterministic counting:*** output  $\hat{Z}$  satisfying

$$(1 - \epsilon)Z \leq \hat{Z} \leq (1 + \epsilon)Z$$



# Deterministic approximate counting for hypergraph colourings

Work	Regime	Running time	Technique
GLLZ17	$q \gtrsim d^{14/k}$	$n^{\text{poly}(dk \log q)}$	linear programming
JPV21	$q \gtrsim d^{7/k}$	$n^{\text{poly}(dk \log q)}$	linear programming
HWY22	$q \gtrsim d^{5/k}$	$n^{\text{poly}(dk \log q)}$	derandomisation

MCMC & Compression: sampling full colouring  $X \in [q]^V$  in  $O(n \log n)$  transition steps

↓ Coupling towards the past [F., Guo, Wang, Wang and Yin, 2022]

Sample from marginal distribution  $\mu_S$  for a small subset  $S \subseteq V$  in  $O(\log n)$  step

↓ derandomisation

$n^{\text{poly}(dk \log q)}$ -time deterministic approximate counting if  $q \gtrsim d^{3/k}$

# Open problems

## CNF formula

NP-Hard  
 $k \lesssim 2 \log d$

?

Poly-Time Algorithm  
 $k \gtrsim 5 \log d$

my guess

## Hypergraph colouring

NP-Hard  
 $q \lesssim \Delta^{2/k}$

?

Poly-Time Algorithm  
 $q \lesssim \Delta^{3/k}$

my guess

Thank you! Q&A