Sampling Lovász Local Lemma for General Constraint Satisfaction Solutions in Near-Linear Time

Authors: Kun He (Chinese Academy of Sciences)
Chunyang Wang (Nanjing University)
Yitong Yin (Nanjing University)

Slides made by: Chunyang Wang (Nanjing University)
Presenter: Weiming Feng (University of Edinburgh)

FOCS2022

Constraint Satisfaction Problem

$$\Phi = (V, \boldsymbol{Q}, \mathscr{C})$$

Variables: $V = \{v_1, v_2, \ldots, v_n\}$ with finite domains Q_v for each $v \in V$

Constraints: $\mathscr{C} = \{c_1, c_2, \ldots, c_m\}$ with each $c \in \mathscr{C}$ defined on vbl$(c) \subseteq V$

$$c : \bigotimes_{v \in \mathsf{vbl}(c)} Q_v \to \{\text{satisfied}, \text{not satisfied}\}$$

CSP solution: assignment $\boldsymbol{X} \in \bigotimes_{v \in V} Q_v$ s.t. all constraints are satisfied

Constraint Satisfaction Problem

$$\Phi = (V, \boldsymbol{Q}, \mathscr{C})$$

Variables: $V = \{v_1, v_2, \ldots, v_n\}$ with finite domains Q_v for each $v \in V$

Constraints: $\mathscr{C} = \{c_1, c_2, \ldots, c_m\}$ with each $c \in \mathscr{C}$ defined on $\mathsf{vbl}(c) \subseteq V$

$$c : \bigotimes_{v \in \mathsf{vbl}(c)} Q_v \to \{\text{satisfied}, \text{not satisfied}\}$$

CSP solution: assignment $X \in \bigotimes_{v \in V} Q_v$ s.t. all constraints are satisfied

Decision: Can we efficiently decide if Φ has a solution?

Search: Can we efficiently find a solution of Φ?

Sampling: Can we efficiently sample an (almost) uniform random solution of Φ?

$$\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_5 \vee \neg x_6) \wedge (x_3 \vee \neg x_4 \vee \neg x_5)$$

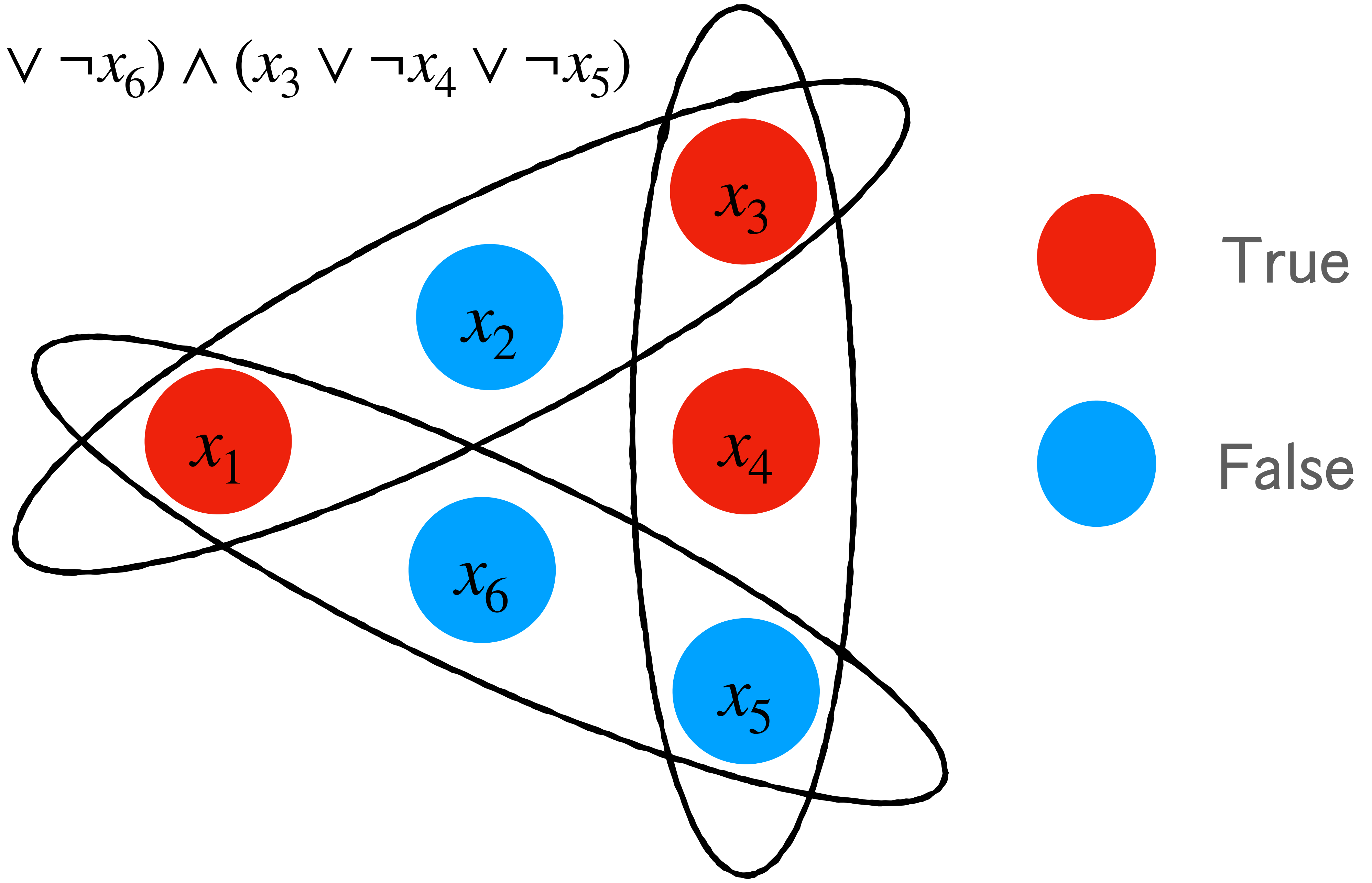
Example: k-CNF

$V = \{x_1, x_2, \ldots, x_n\}$

$\mathscr{C} = (C_1, C_2, \ldots, C_m),\ |C_i| = k$

$Q_v \in \{\text{True}, \text{False}\}$ for each $v \in V$

Solution: an assignment such that each clause (constraint) evaluates to True

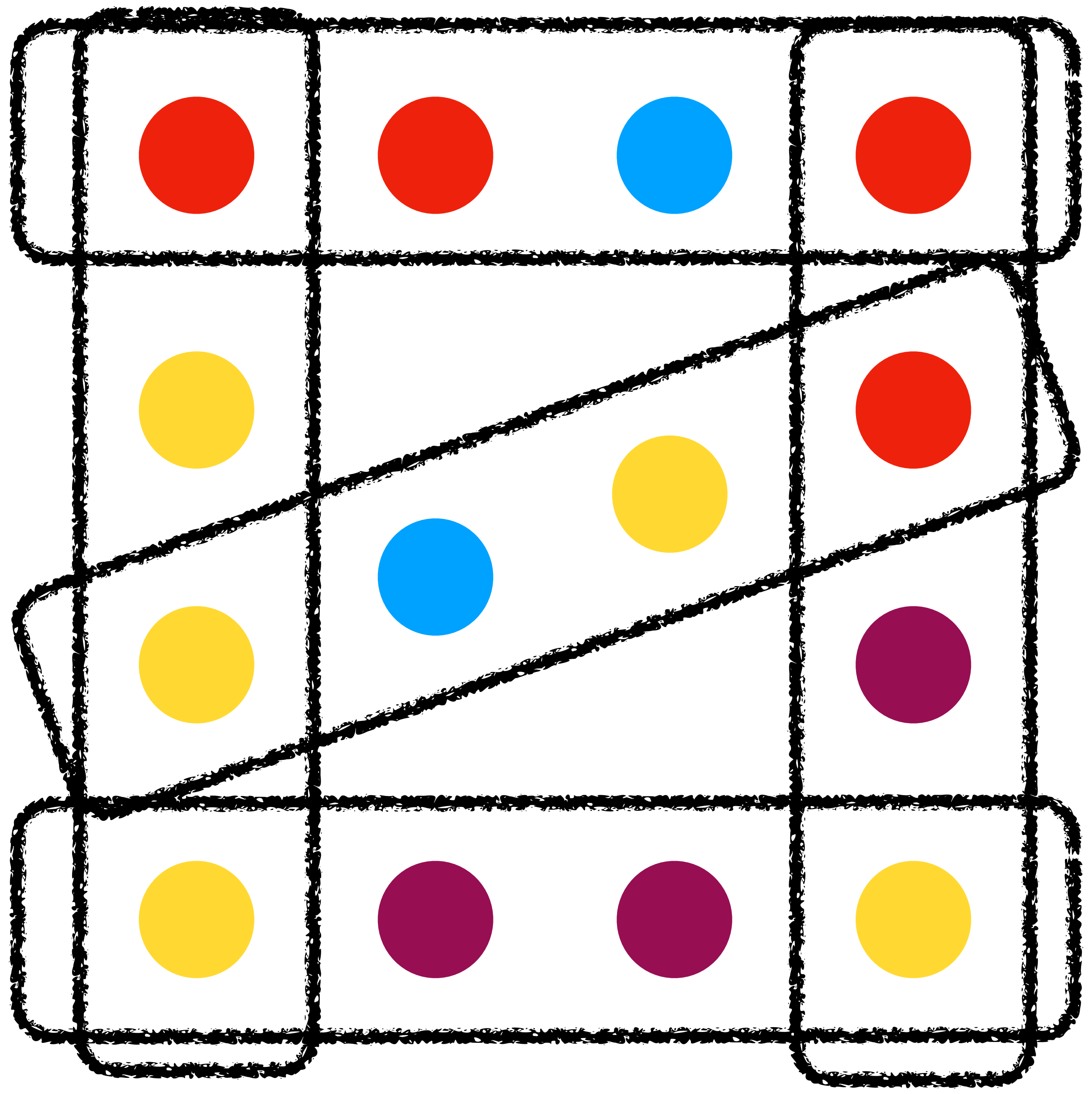


Example: hypergraph q-coloring

k-uniform hypergraph $H = (V, \mathscr{E})$

color set $[q]$ for each $v \in V$

Solution: an assignment such that no hyperedge (constraint) is monochromatic



Lovász Local Lemma

$$\Phi = (V, \boldsymbol{Q}, \mathscr{C})$$

Variable framework

- each $v \in V$ draws from Q_v uniformly and independently at random
- product distribution $\mathscr{P}$

Parameters

- **violation probability** $p = \max_{c \in \mathscr{C}} \Pr_{\mathscr{P}}[\neg c]$
- **constraint degree** $\Delta = \max_{c \in \mathscr{C}} |\{c' \in \mathscr{C} \mid \mathsf{vbl}(c) \cap \mathsf{vbl}(c') \neq \emptyset\}|$

Lovász Local Lemma

$$\Phi = (V, Q, \mathscr{C})$$

Variable framework

- each $v \in V$ draws from Q_v uniformly and independently at random
- product distribution $\mathscr{P}$

Parameters

- **violation probability** $p = \max_{c \in \mathscr{C}} \Pr_{\mathscr{P}}[\neg c]$
- **constraint degree** $\Delta = \max_{c \in \mathscr{C}} |\{c' \in \mathscr{C} \mid \mathsf{vbl}(c) \cap \mathsf{vbl}(c') \neq \emptyset\}|$

$$\boxed{ep\Delta \leq 1}$$

Lovász Local Lemma [Erdos, Lovasz '75]

$\Longrightarrow$

Algorithmic Lovász Local Lemma [Moser, Tardos '10]

A CSP solution exists and can be efficiently found!

Sampling **L**ovász **L**ocal **L**emma

Sampling LLL

Input: a CSP formula $\Phi = (V, \boldsymbol{Q}, \mathscr{C})$ under LLL-like conditions $p\Delta^c \lesssim 1$ → [BGGGS19,GGW22]: **NP-hard** if $p\Delta^2 \gtrsim 1$!

Output: an (almost) uniform satisfying solution of Φ

Sampling Lovász Local Lemma

Sampling LLL

Input: a CSP formula $\Phi = (V, \boldsymbol{Q}, \mathscr{C})$ under LLL-like conditions $p\Delta^c \lesssim 1$ → [BGGGS19,GGW22]: **NP-hard** if $p\Delta^2 \gtrsim 1$!

Output: an (almost) uniform satisfying solution of Φ

Applications:

Approximate counting CSP solutions (Counting LLL)

Almost Uniform Sampling → Approximate Counting

self-reduction [Jerrum, Valiant, Vazirani 1986]

adaptive simulated annealing [Štefankovič, Vempala, Vigoda 2009]

Inference in probabilistic graphical models

Gibbs distribution μ: uniform distribution over all solutions to Φ

Inference: $\Pr_{X \sim \mu}[X_{v_i} = \cdot \mid X_S = x_s]$

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu'16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	poly$(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang'16	monotone k-CNF	$p\Delta^2 \lesssim 1$	poly$(k, \Delta) \cdot n \log n$	MCMC

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu'16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	$\mathrm{poly}(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang'16	monotone k-CNF	$p\Delta^2 \lesssim 1$	$\mathrm{poly}(k, \Delta) \cdot n \log n$	MCMC
Moitra'17	k-CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	mark/unmark + linear programming
Guo, Liu, Lu, Zhang'19	hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Jain, Pham, Vuong'21b	general CSP	$p\Delta^{7} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
		polynomial running time only if $k, q, \Delta = O(1)$		

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu’16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	$\mathsf{poly}(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang’16	monotone k-CNF	$p\Delta^2 \lesssim 1$	$\mathsf{poly}(k, \Delta) \cdot n \log n$	MCMC
Moitra’17	k-CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathsf{poly}(k,\Delta)}$	mark/unmark + linear programming
Guo, Liu, Lu, Zhang’19	hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathsf{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Jain, Pham, Vuong’21b	general CSP	$p\Delta^7 \lesssim 1$	$n^{\mathsf{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu'16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	$\text{poly}(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang'16	monotone k-CNF	$p\Delta^2 \lesssim 1$	$\text{poly}(k, \Delta) \cdot n \log n$	MCMC
Moitra'17	k-CNF	$p\Delta^{60} \lesssim 1$	$n^{\text{poly}(k,\Delta)}$	mark/unmark + linear programming
Guo, Liu, Lu, Zhang'19	hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\text{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Jain, Pham, Vuong'21b	general CSP	$p\Delta^7 \lesssim 1$	$n^{\text{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Feng, Guo, Yin, Zhang'20	k-CNF	$p\Delta^{20} \lesssim 1$	$\text{poly}(k, \Delta) \cdot \tilde{O}(n^{1.001})$	mark/unmark + projected MCMC
Feng, He, Yin'21	atomic CSP	$p\Delta^{350} \lesssim 1/N$	$\text{poly}(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression + projected MCMC
Jain, Pham, Vuong'21a He, Sun, Wu'21	atomic CSP	$p\Delta^{5.713} \lesssim 1/N$	$\text{poly}(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression + projected MCMC

atomic CSP: each constraint forbids a small number $N = \text{poly}(k, \Delta, q)$ of configurations

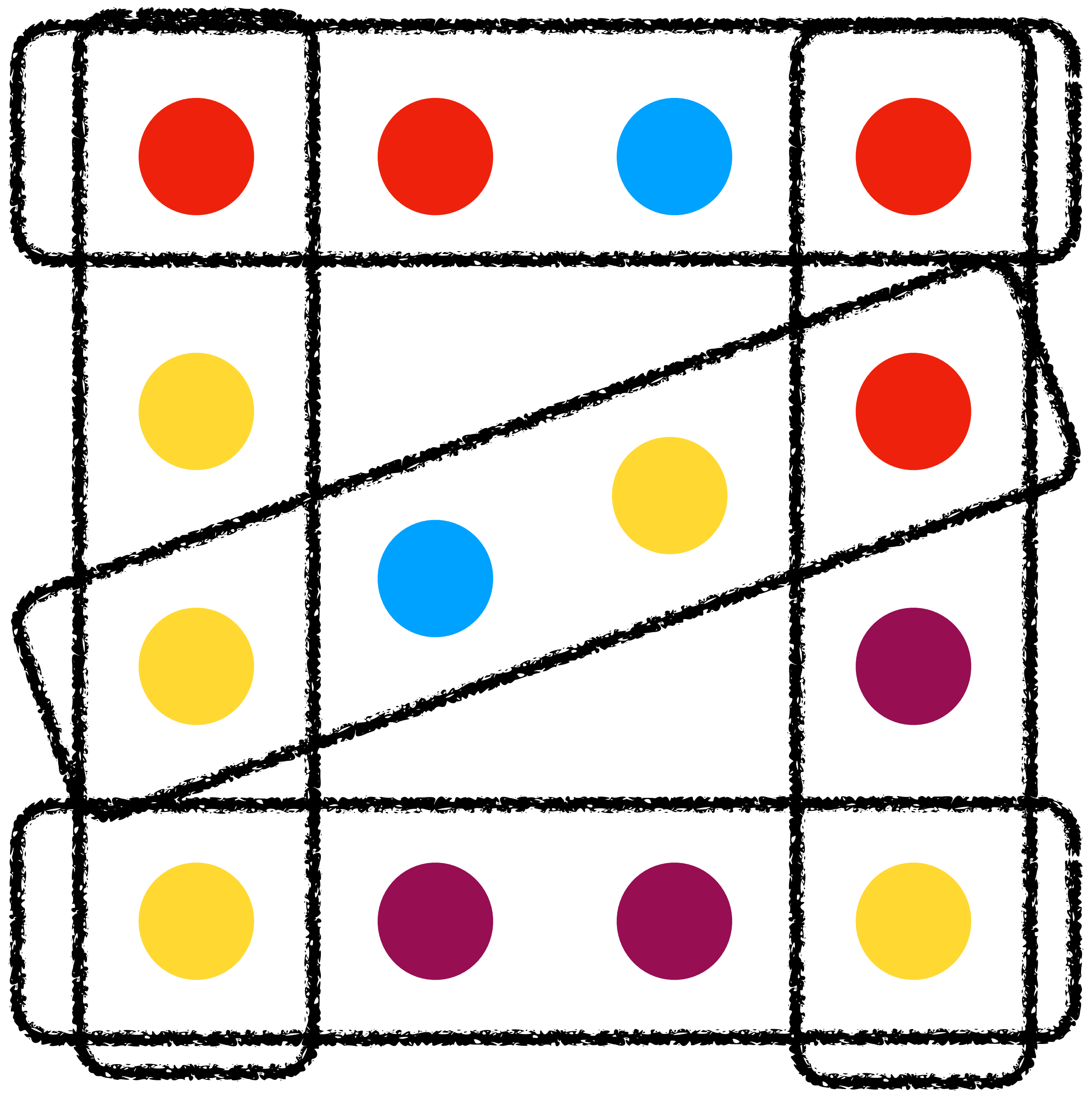
fast sampler: polynomial time even for unbounded degree

Example: hypergraph q-coloring

k-uniform hypergraph $H = (V, \mathscr{E})$

color set $[q]$ for each $v \in V$

Solution: an assignment such that no hyperedge (constraint) is monochromatic



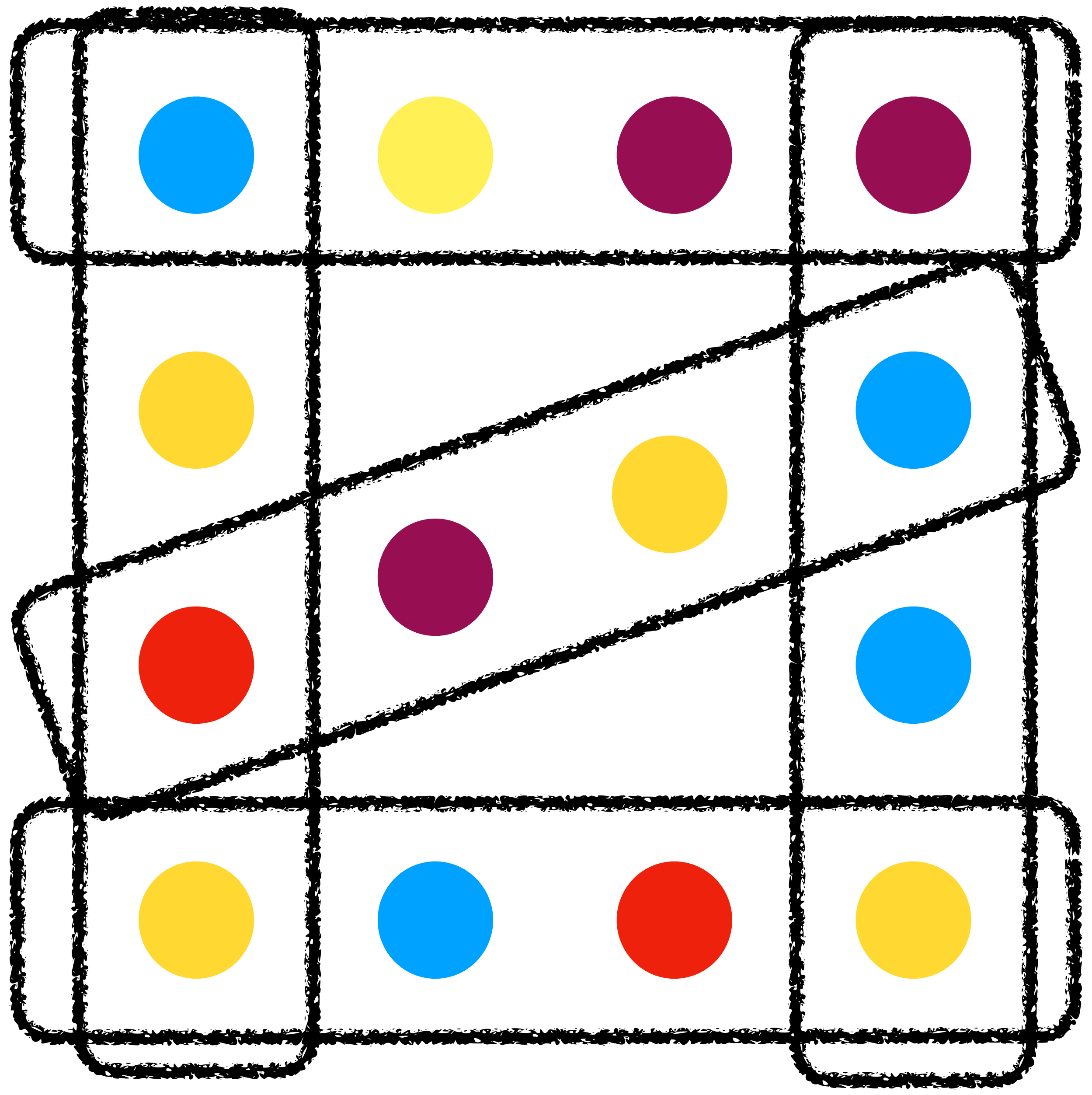
$N = q$, **atomic!**

Example: δ-robust hypergraph q-coloring

k-uniform hypergraph $H = (V, \mathscr{E})$

color set $[q]$ for each $v \in V$

Solution: an assignment such that each hyperedge (constraint) has no $(1-\delta)k$ vertices with the same color



$N \geq q^{\delta k}$, **non-atomic!**

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu'16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	$\mathrm{poly}(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang'16	monotone k-CNF	$p\Delta^2 \lesssim 1$	$\mathrm{poly}(k, \Delta) \cdot n \log n$	MCMC
Moitra'17	k-CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	mark/unmark + linear programming
Guo, Liu, Lu, Zhang'19	hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Jain, Pham, Vuong'21b	general CSP	$p\Delta^7 \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Feng, Guo, Yin, Zhang'20	k-CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k, \Delta) \cdot \tilde{O}(n^{1.001})$	mark/unmark +projected MCMC
Feng, He, Yin'21	atomic CSP	$p\Delta^{350} \lesssim 1/N$	$\mathrm{poly}(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression +projected MCMC
Jain, Pham, Vuong'21a He, Sun, Wu'21	atomic CSP	$p\Delta^{5.713} \lesssim 1/N$	$\mathrm{poly}(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression +projected MCMC

The projected MCMC technique for fast sampling LLL only works for atomic instances

Open problem: fast sampling LLL for general CSP? (new techniques required)

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu'16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	poly$(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang'16	monotone k-CNF	$p\Delta^2 \lesssim 1$	poly$(k, \Delta) \cdot n \log n$	MCMC
Moitra'17	k-CNF	$p\Delta^{60} \lesssim 1$	$n^{\text{poly}(k,\Delta)}$	mark/unmark + linear programming
Guo, Liu, Lu, Zhang'19	hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\text{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Jain, Pham, Vuong'21b	general CSP	$p\Delta^7 \lesssim 1$	$n^{\text{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Feng, Guo, Yin, Zhang'20	k-CNF	$p\Delta^{20} \lesssim 1$	poly$(k, \Delta) \cdot \tilde{O}(n^{1.001})$	mark/unmark +projected MCMC
Feng, He, Yin'21	atomic CSP	$p\Delta^{350} \lesssim 1/N$	poly$(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression +projected MCMC
Jain, Pham, Vuong'21a He, Sun, Wu'21	atomic CSP	$p\Delta^{5.713} \lesssim 1/N$	poly$(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression +projected MCMC
This work	general CSP	$p\Delta^7 \lesssim 1$	poly$(k, \Delta, q) \cdot n$	**recursive marginal sampler**

Work	Instance	Condition	Complexity	Technique
Guo, Jerrum, Liu'16	CSP with large constraint intersections	$p\Delta^2 \lesssim 1, s \geq \min(\log \Delta, k/2)$	poly$(k, \Delta, q) \cdot n$	partial rejection sampling
Hermon, Sly, Zhang'16	monotone k-CNF	$p\Delta^2 \lesssim 1$	poly$(k, \Delta) \cdot n \log n$	MCMC
Moitra'17	k-CNF	$p\Delta^{60} \lesssim 1$	$n^{\text{poly}(k,\Delta)}$	mark/unmark + linear programming
Guo, Liu, Lu, Zhang'19	hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\text{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Jain, Pham, Vuong'21b	general CSP	$p\Delta^{7} \lesssim 1$	$n^{\text{poly}(k,\Delta,\log q)}$	adaptive mark/unmark + linear programming
Feng, Guo, Yin, Zhang'20	k-CNF	$p\Delta^{20} \lesssim 1$	poly$(k, \Delta) \cdot \tilde{O}(n^{1.001})$	mark/unmark +projected MCMC
Feng, He, Yin'21	atomic CSP	$p\Delta^{350} \lesssim 1/N$	poly$(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression +projected MCMC
Jain, Pham, Vuong'21a He, Sun, Wu'21	atomic CSP	$p\Delta^{5.713} \lesssim 1/N$	poly$(k, \Delta, q) \cdot \tilde{O}(n^{1.001})$	state compression +projected MCMC
This work	general CSP	$p\Delta^{7} \lesssim 1$	poly$(k, \Delta, q) \cdot n$	**recursive marginal sampler**

inspired by [Anand, Jerrum '22]!

Our results

fast sampler for general CSPs in the LLL regime

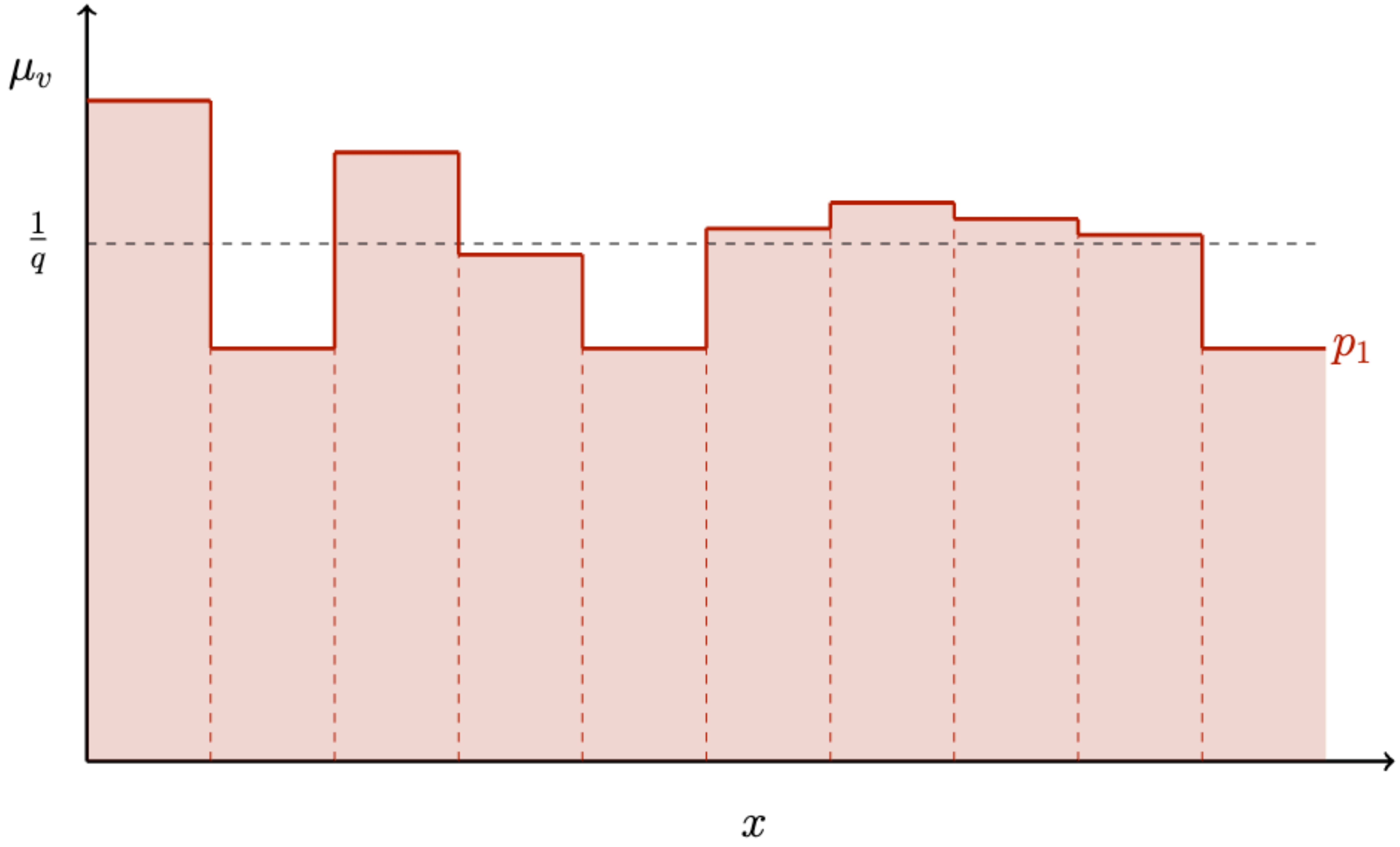
For **general** CSP satisfying

$$q^2 \cdot k \cdot p \cdot \Delta^7 \leq \frac{1}{150\mathrm{e}^3}$$

q: domain size
k: constraint width
p: violation probability
Δ: constraint degree
n: $|V|$

- **Sampling algorithm**:
  draw almost uniform satisfying solution in expected time $O(\mathrm{poly}(k, \Delta, q) \cdot n)$
- **Counting algorithm**:
  approximately count satisfying solutions in expected time $\tilde{O}(\mathrm{poly}(k, \Delta, q) \cdot n^2)$
- **Inference algorithm**:
  approximate marginal probability in expected time $O(\mathrm{poly}(k, \Delta, q))$

Local Uniformity



μ: uniform distribution over solutions

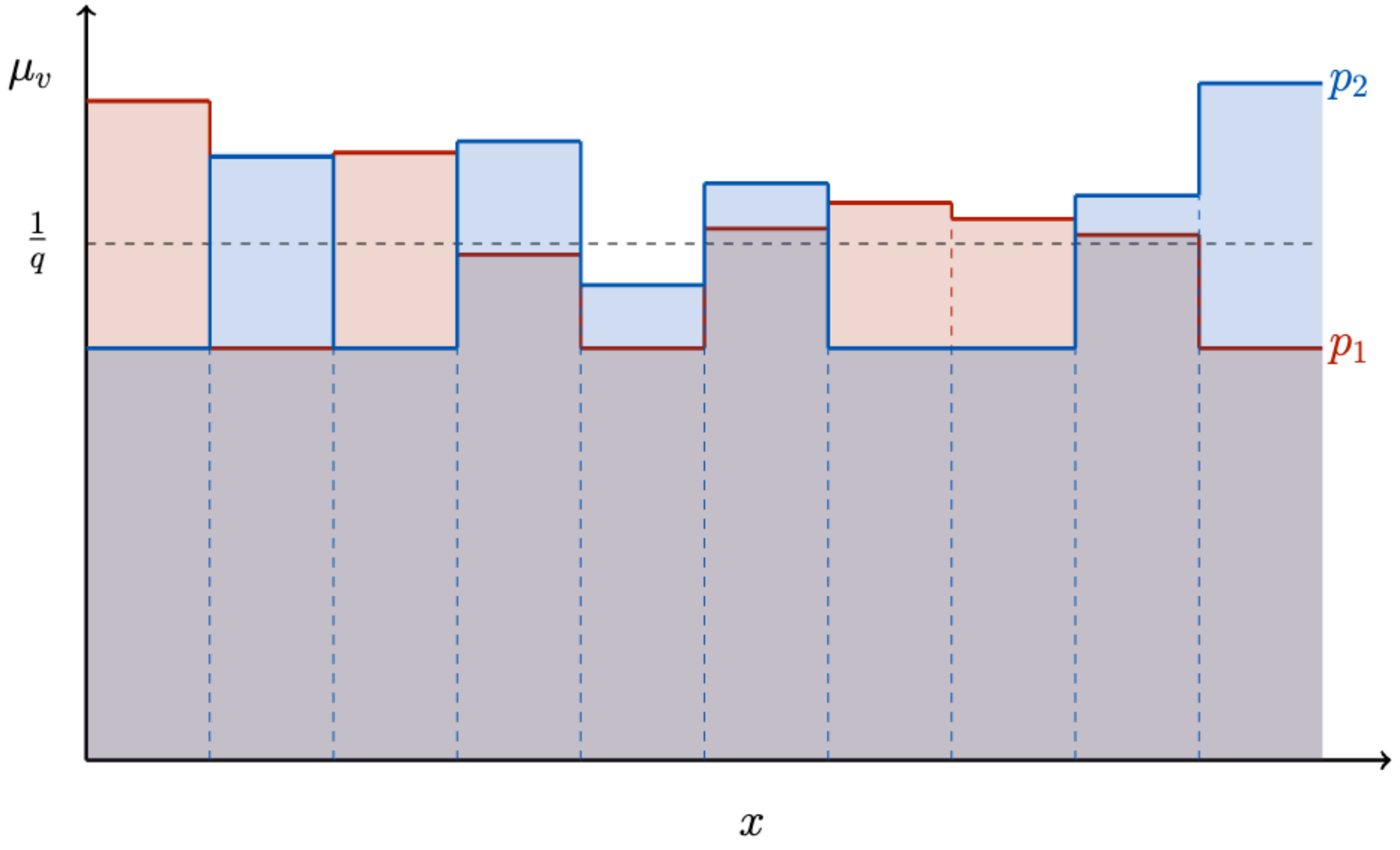
μ_v: marginal distribution at $v \in V$

[Haeupler, Saha, Srinivasan '11]:

LLL condition $\Longrightarrow \mu_v \geq \theta$

where $\theta = (1 - o(1))\dfrac{1}{q}$

Local Uniformity



μ: uniform distribution over solutions

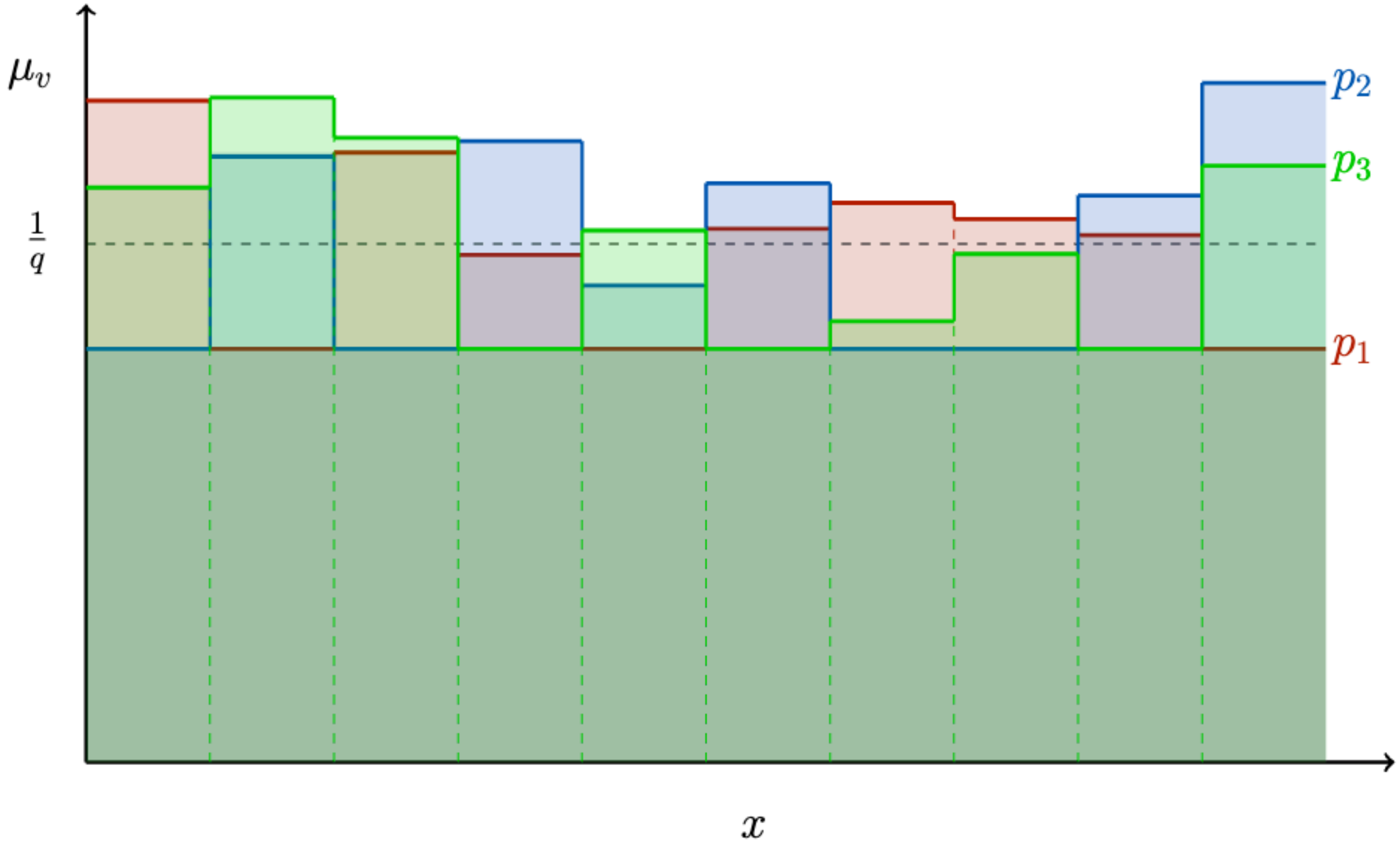
μ_v: marginal distribution at $v \in V$

[Haeupler, Saha, Srinivasan '11]:

LLL condition $\Longrightarrow \mu_v \geq \theta$

where $\theta = (1 - o(1))\dfrac{1}{q}$

Local Uniformity



μ: uniform distribution over solutions

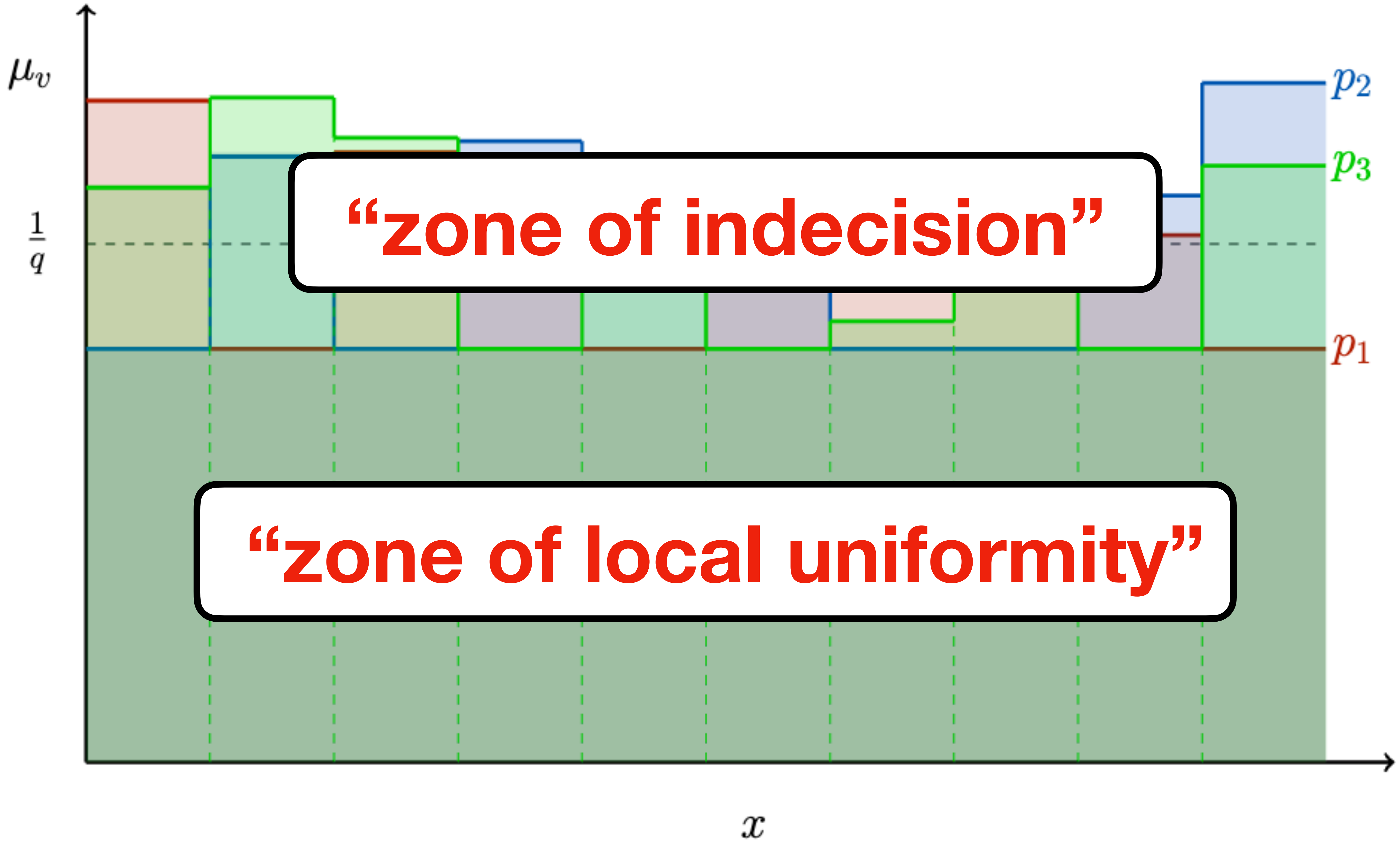
μ_v: marginal distribution at $v \in V$

[Haeupler, Saha, Srinivasan '11]:

LLL condition $\Longrightarrow \mu_v \geq \theta$

where $\theta = (1 - o(1))\frac{1}{q}$

Local Uniformity



μ: uniform distribution over solutions

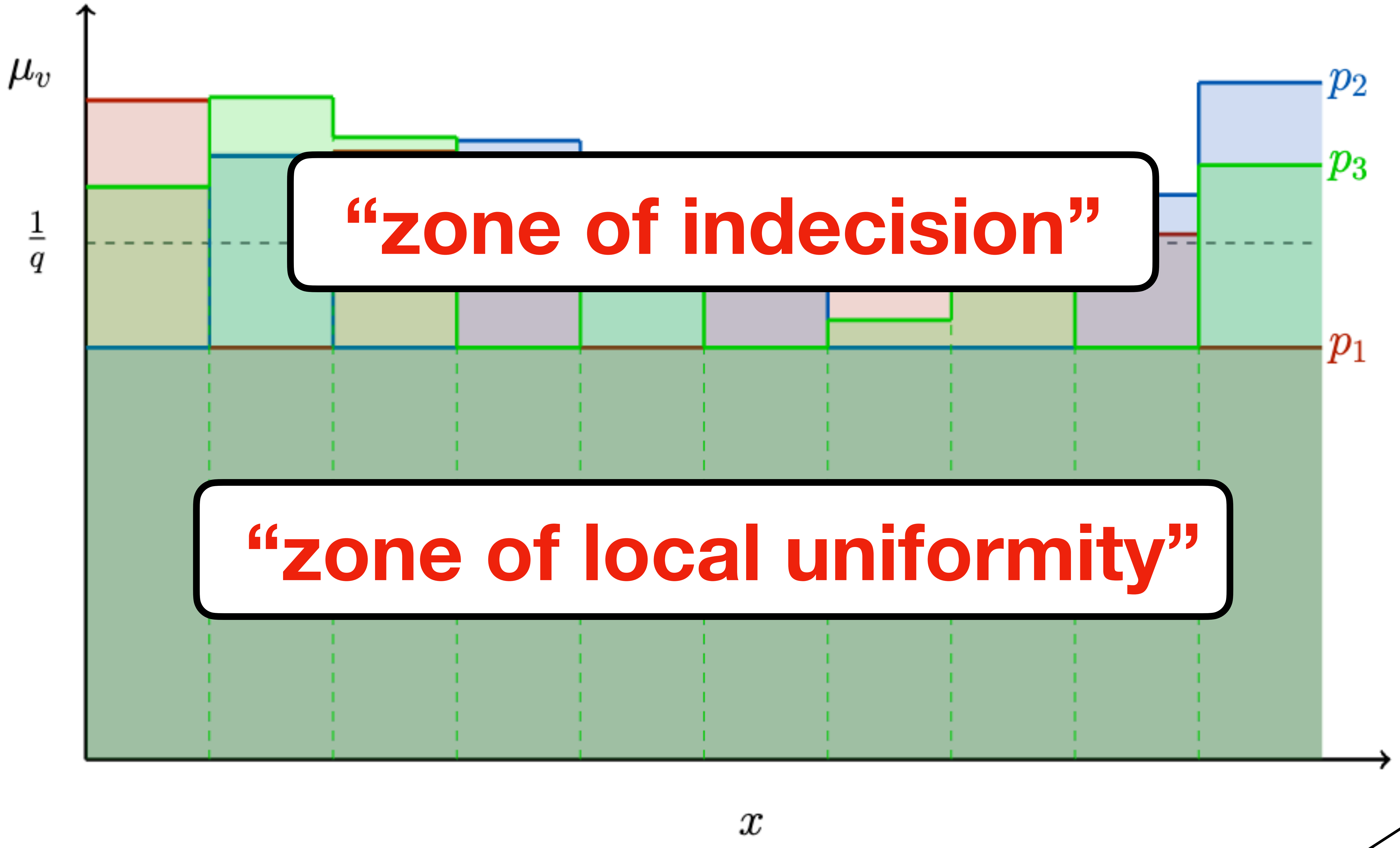
μ_v: marginal distribution at $v \in V$

[Haeupler, Saha, Srinivasan '11]:

LLL condition $\Longrightarrow \mu_v \geq \theta$

where $\theta = (1 - o(1))\dfrac{1}{q}$

Local Uniformity



μ: uniform distribution over solutions

μ_v: marginal distribution at $v \in V$

[Haeupler, Saha, Srinivasan ’11]:

LLL condition $\Longrightarrow \mu_v \geq \theta$

where $\theta = (1 - o(1))\frac{1}{q}$

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

$\mathscr{U}$: uniform over $[q]$

“Overflow distribution”

$$\mathscr{D}(x) = \frac{\mu_v(x) - \theta}{1 - q\theta}$$

A marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$ **Trivial!**

with probability $1 - q\theta$, draw from $\mathscr{D}$ **Non-trivial!**

A marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$ **Trivial!**

with probability $1 - q\theta$, draw from $\mathscr{D}$ **Non-trivial!**

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$

a linear transform of μ_v

sampling from $\mathscr{D}$ $\xrightarrow[\text{[Nacu, Peres '15][Huber '16][Dughmi } et\ al \text{ '16]}]{\text{Bernoulli factory algorithms}}$ sampling from μ_v

A marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$ **Trivial!**

with probability $1 - q\theta$, draw from $\mathscr{D}$ **Non-trivial!**

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$ a linear transform of μ_v

sampling from $\mathscr{D}$ $\xrightarrow[\text{[Nacu, Peres '15][Huber '16][Dughmi } et\ al \text{ '16]}]{\text{Bernoulli factory algorithms}}$ sampling from μ_v

A marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$ **Trivial!**

with probability $1 - q\theta$, draw from $\mathscr{D}$ **Non-trivial!**

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$

a linear transform of μ_v

Bernoulli factory algorithms

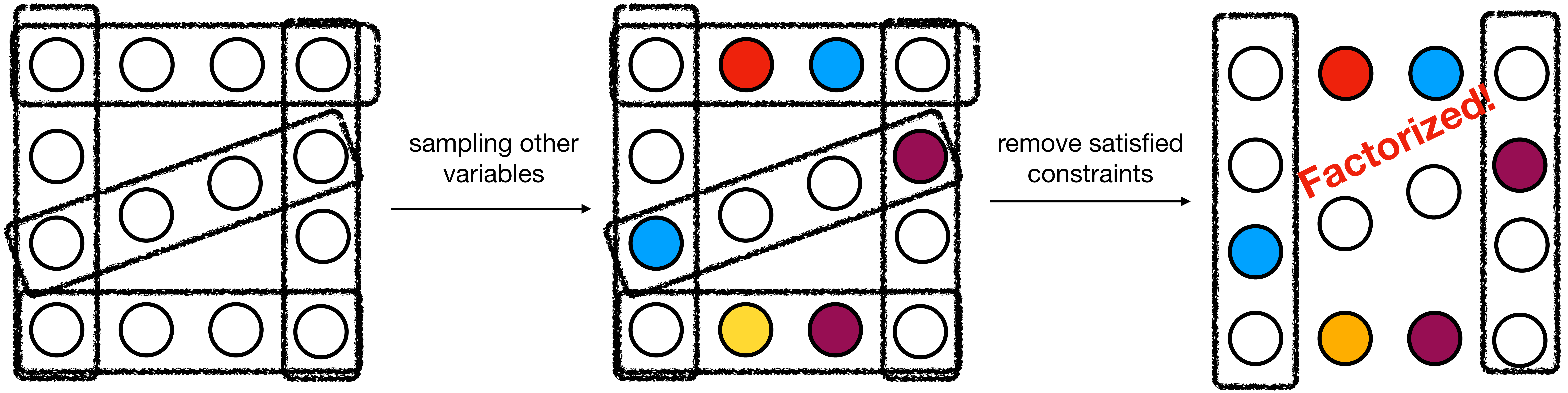
[Nacu, Peres '15][Huber '16][Dughmi *et al* '16]

sampling from $\mathscr{D}$ $\longrightarrow$ sampling from μ_v

A chicken-egg conundrum?

Factorization of the formula

A key observation: sampling other variables first helps factorize the formula!



When the connected component containing v is small,
rejection sampling provides efficient samples from μ_v

A recursive marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$

with probability $1 - q\theta$, draw from $\mathscr{D}$

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$

choose cleverly!

To draw from $\mathscr{D}$:

(1) use recursive calls to repeatedly sample other variables to help factorize the formula

(2) after (1), use Bernoulli Factory algorithms accessing rejection sampling as an oracle

A recursive marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$

with probability $1 - q\theta$, draw from $\mathscr{D}$

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$

choose cleverly!

To draw from $\mathscr{D}$:

(1) use recursive calls to repeatedly sample other variables to help factorize the formula

(2) after (1), use Bernoulli Factory algorithms accessing rejection sampling as an oracle

chain rule→correctness!

A recursive marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$

with probability $1 - q\theta$, draw from $\mathscr{D}$

Caveat: the LLL condition is not self-reducible!

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$

choose cleverly!

To draw from $\mathscr{D}$:

(1) use recursive calls to repeatedly sample other variables to help factorize the formula

(2) after (1), use Bernoulli Factory algorithms accessing rejection sampling as an oracle

chain rule → correctness!

A recursive marginal sampler

$$\mu_v = q\theta \cdot \mathscr{U} + (1 - q\theta) \cdot \mathscr{D}$$

To draw from μ_v:

with probability $q\theta$, draw from $\mathscr{U}$

with probability $1 - q\theta$, draw from $\mathscr{D}$

$$\mathscr{D} = \frac{\mu_v - \theta}{1 - q\theta}$$

choose cleverly!

To draw from $\mathscr{D}$:

(1) use recursive calls to repeatedly sample other variables to help factorize the formula

(2) after (1), use Bernoulli Factory algorithms accessing rejection sampling as an oracle

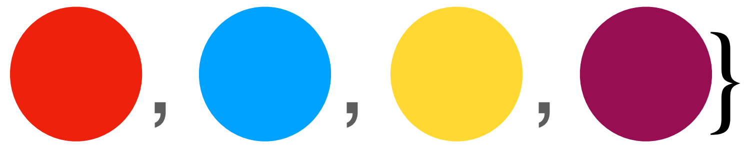
Caveat: the LLL condition is not self-reducible!

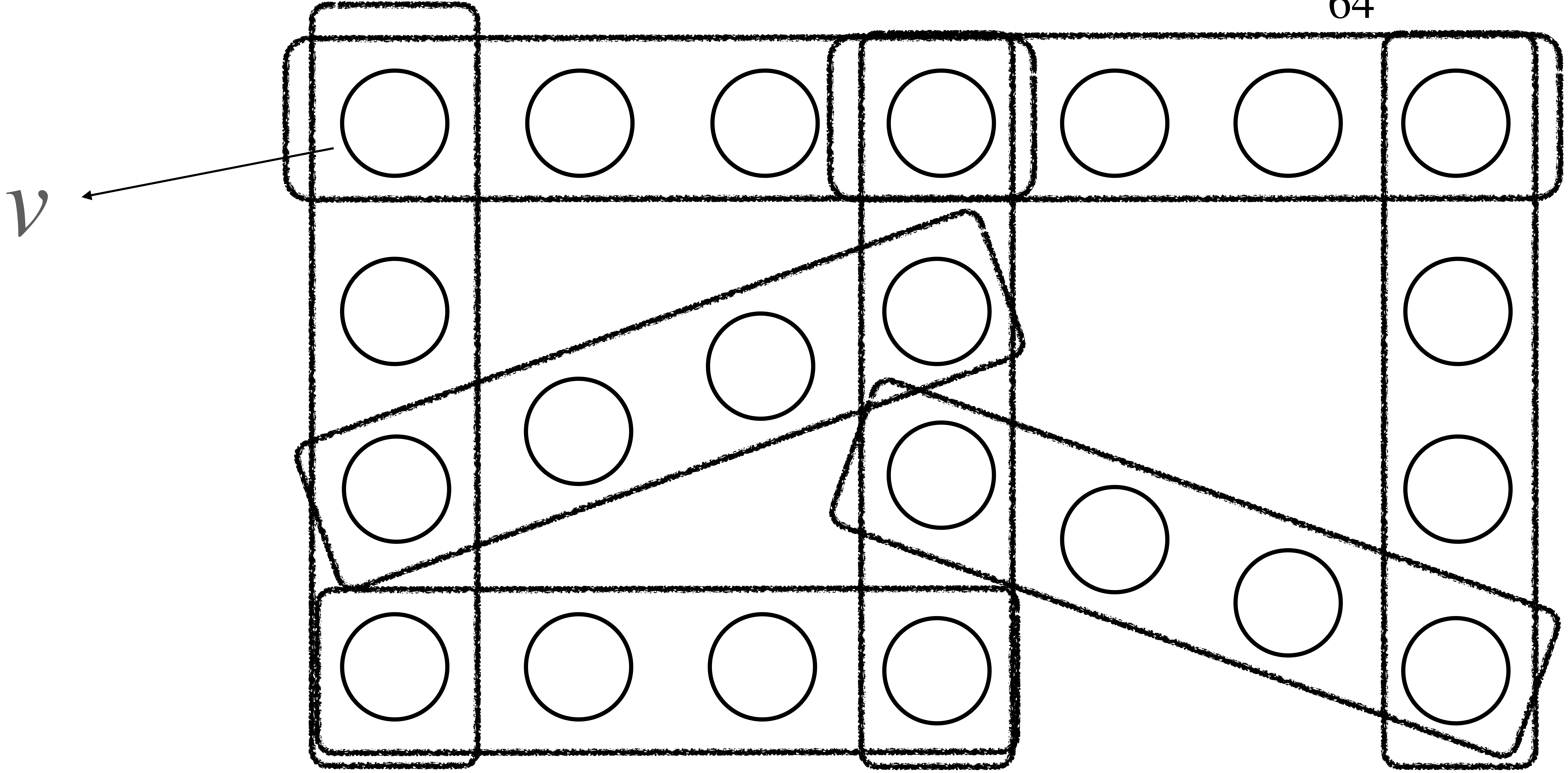
Freezing[Beck' 91][JPV' 21b]

$\Pr_{\mathscr{P}}[\neg c \mid X] > p' \implies$ "freeze" c!

chain rule→correctness!

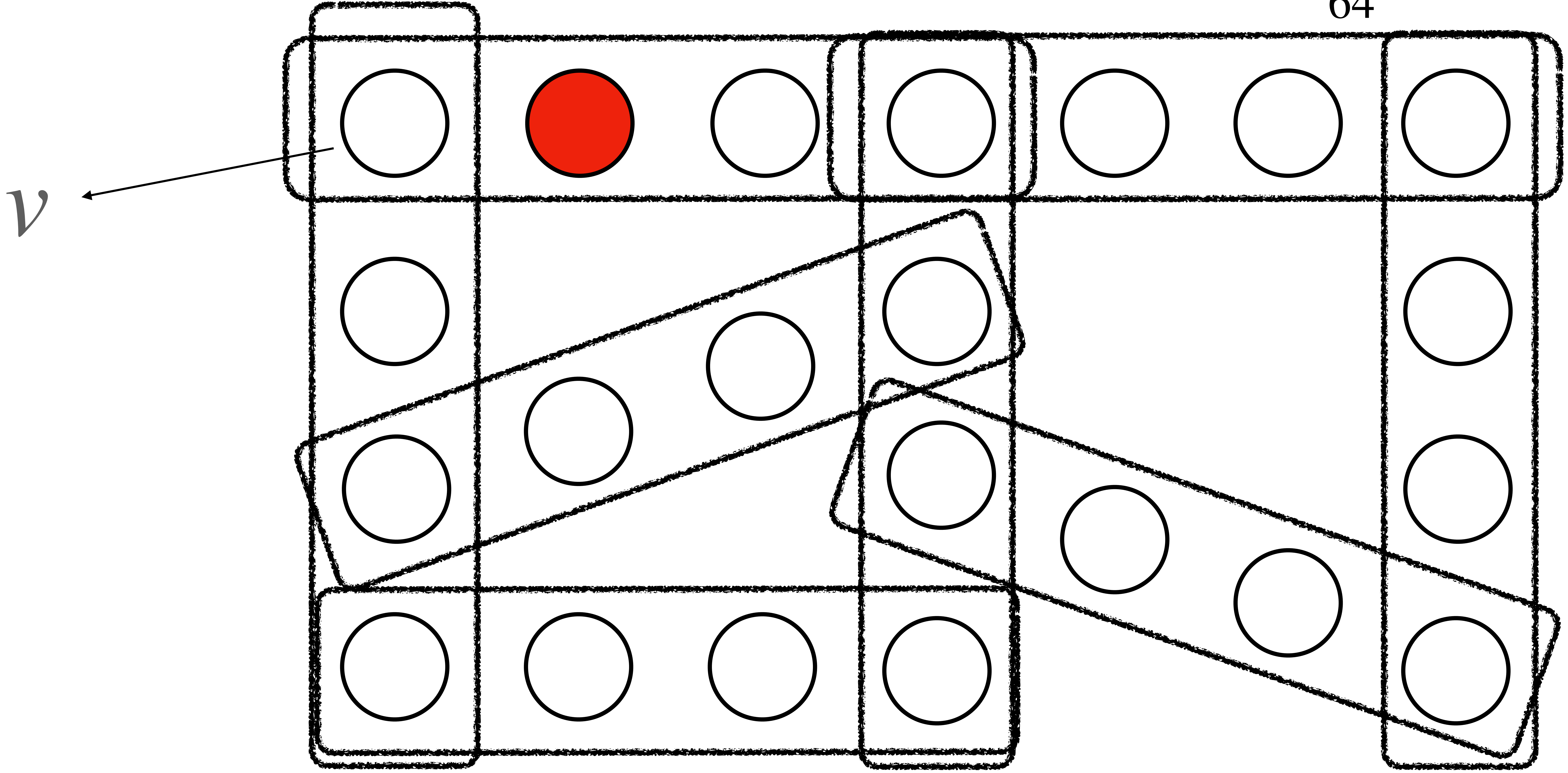
Example: hypergraph coloring

$Q = \{$  $\}^V$, “freezing” threshold $p' = \frac{1}{64}$



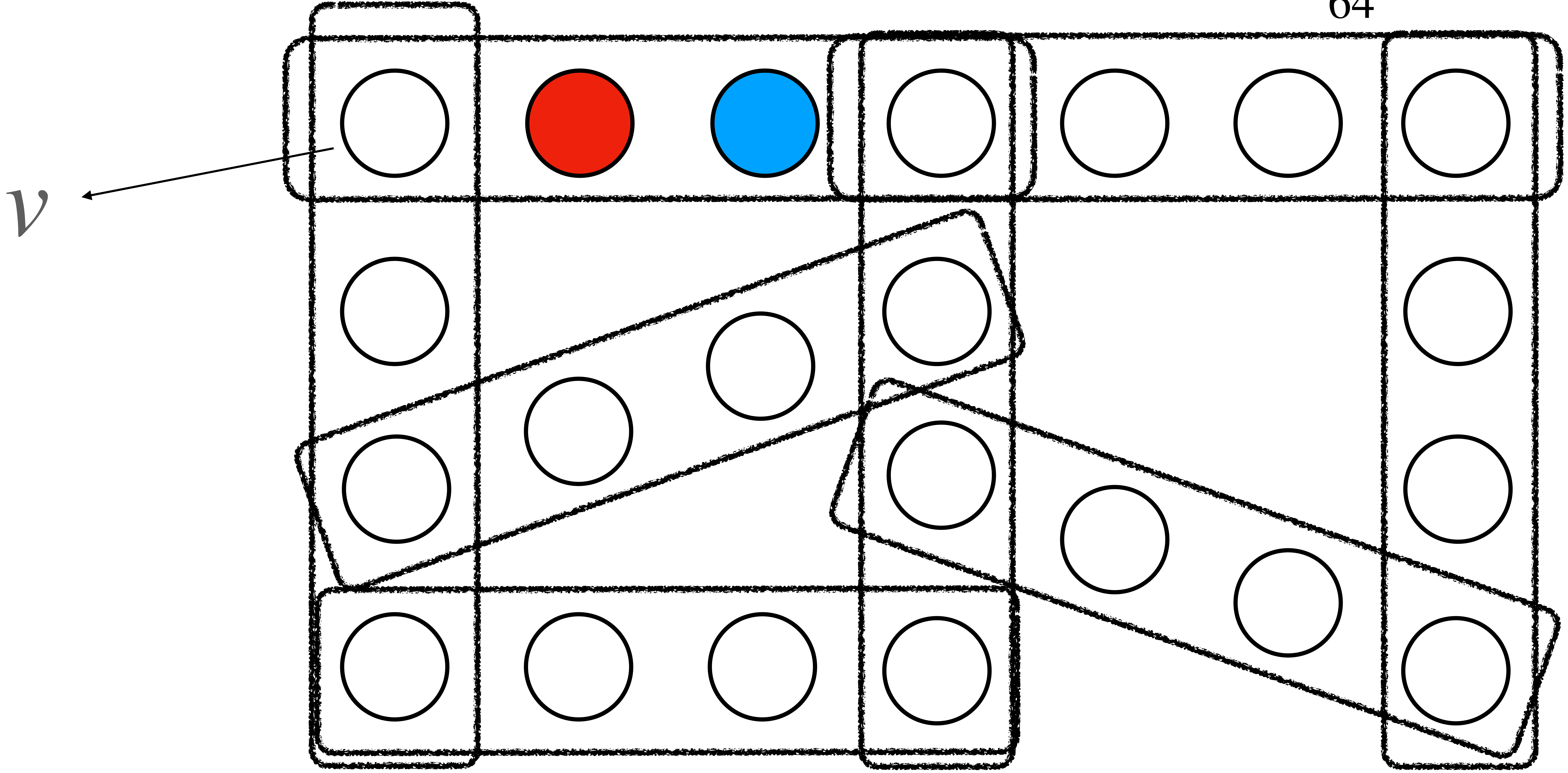
Example: hypergraph coloring

$Q = \{$red, blue, yellow, purple$\}^V$, “freezing” threshold $p' = \frac{1}{64}$



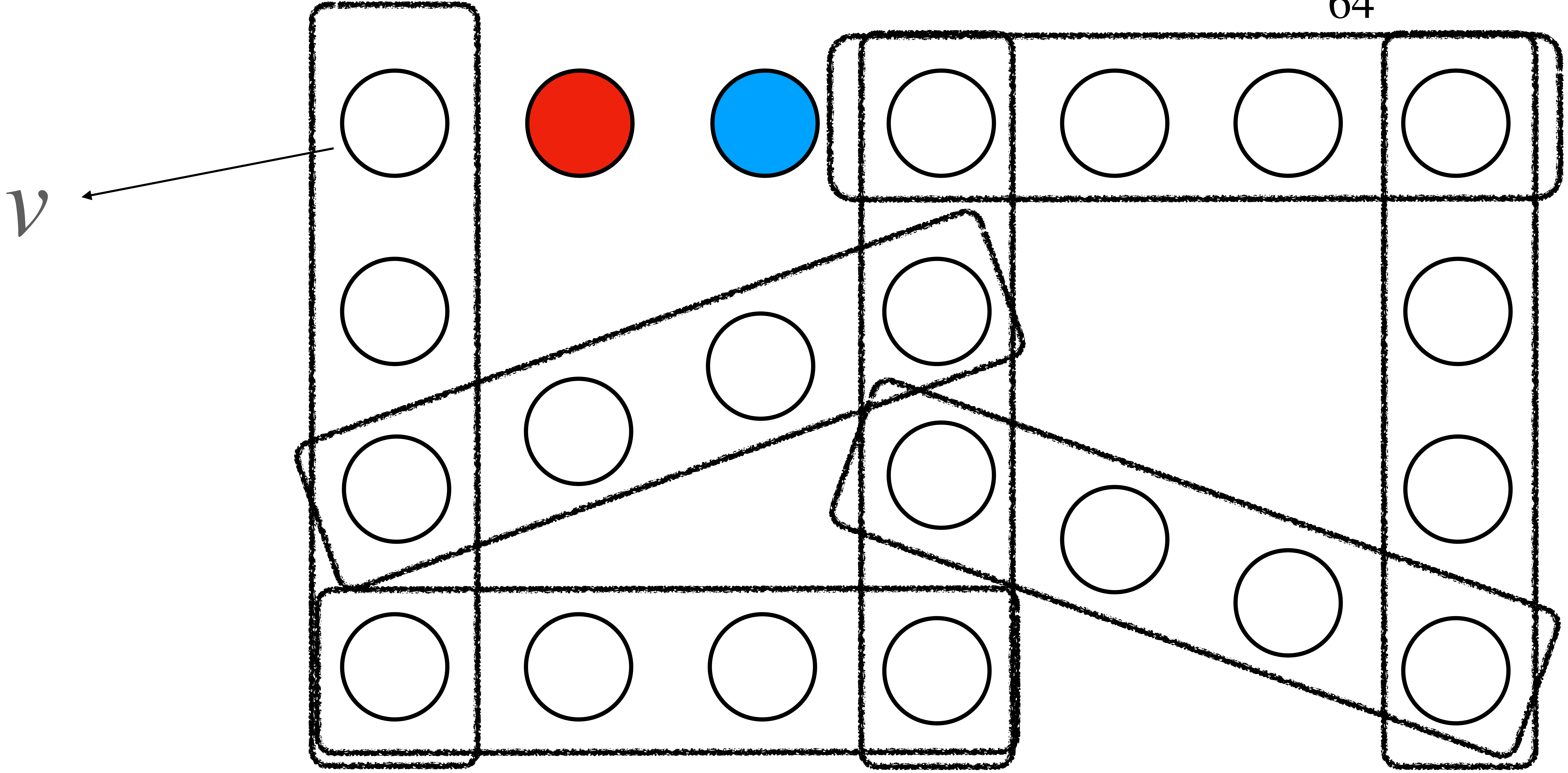
Example: hypergraph coloring

$Q = \{$red, blue, yellow, purple$\}^V$, "freezing" threshold $p' = \dfrac{1}{64}$



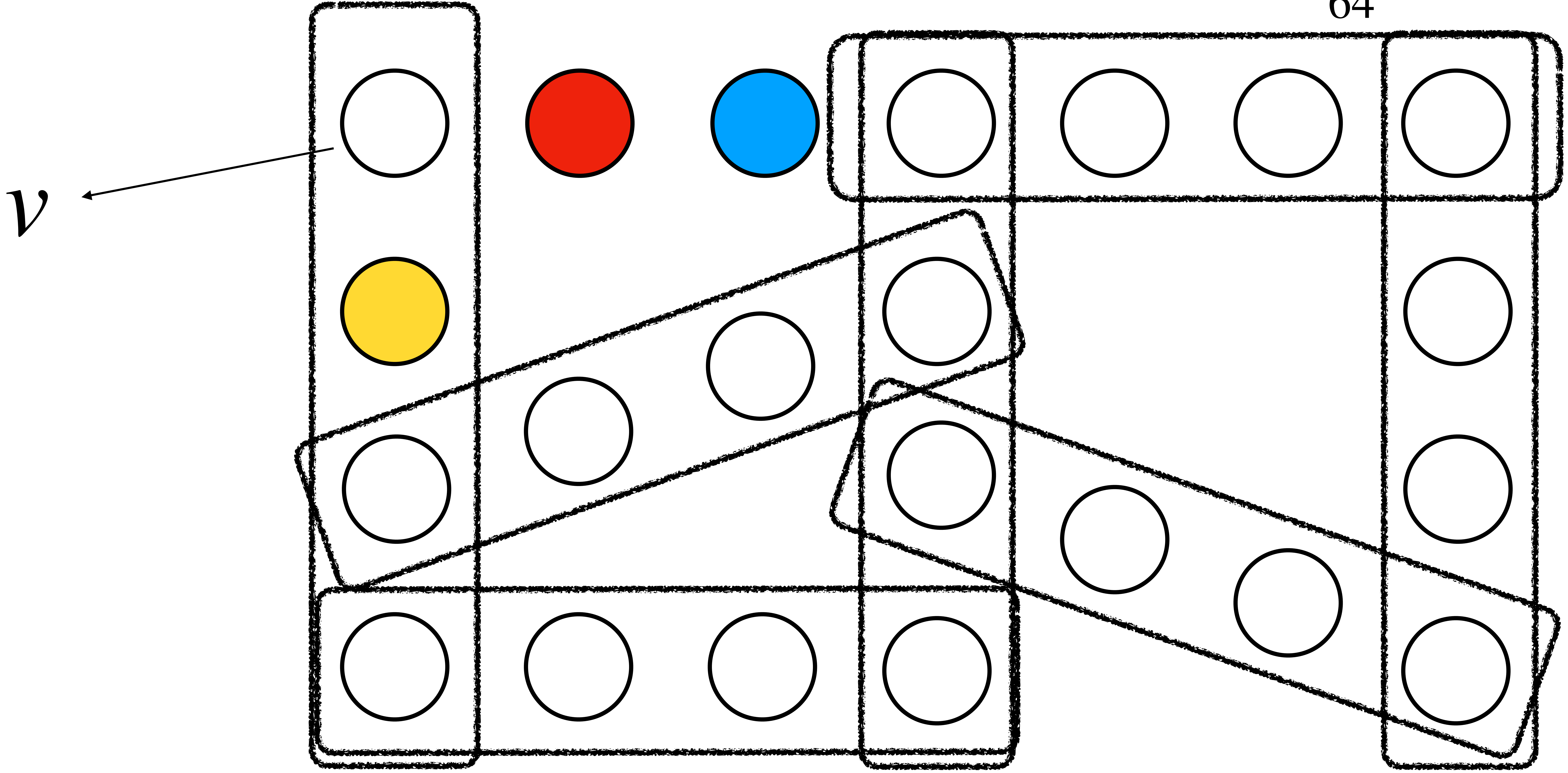
Example: hypergraph coloring

$Q = \{$●, ●, ●, ●$\}^V$, "freezing" threshold $p' = \dfrac{1}{64}$



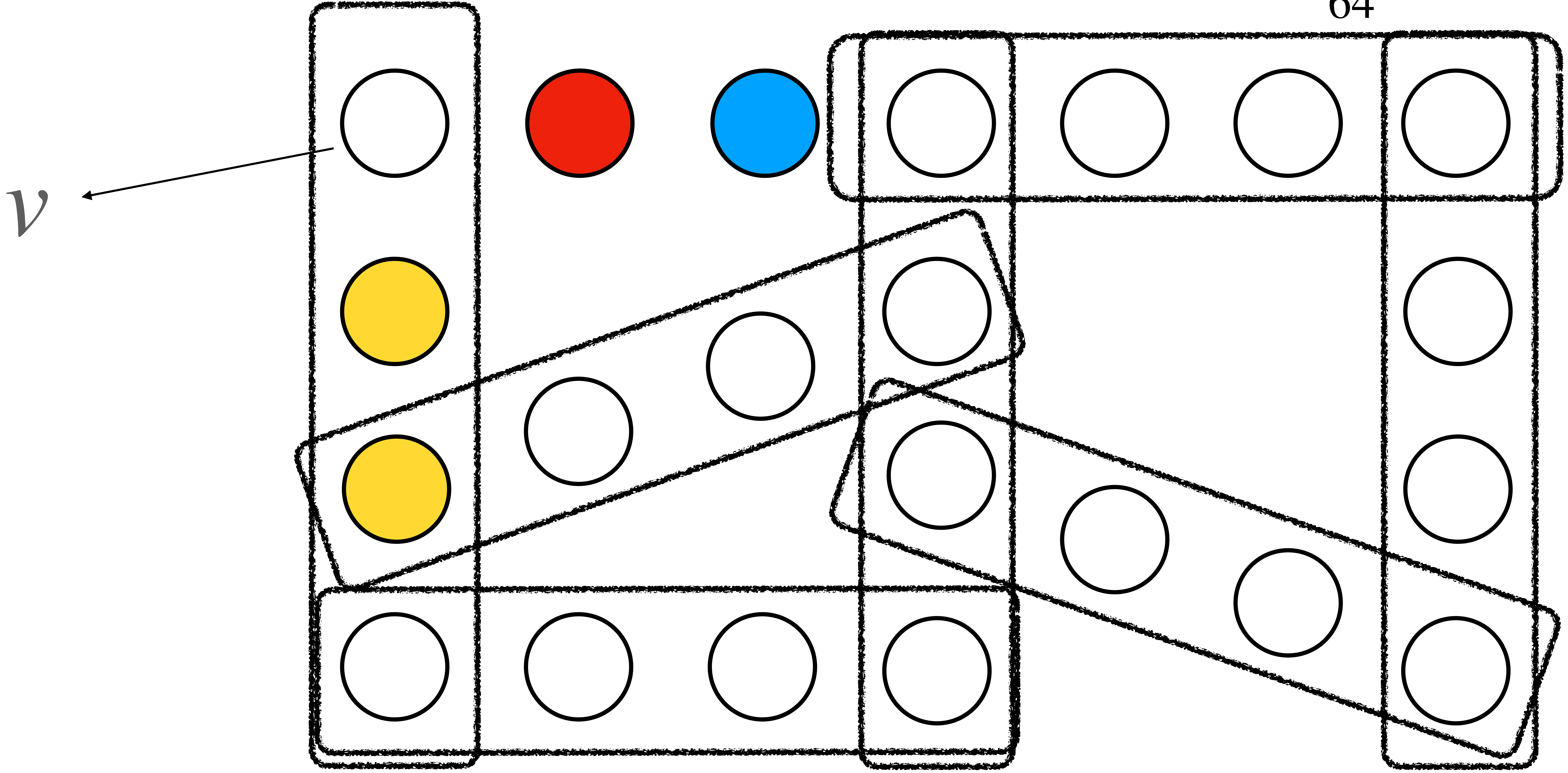
Example: hypergraph coloring

$Q = \{$, , , $\}^V$, "freezing" threshold $p' = \dfrac{1}{64}$



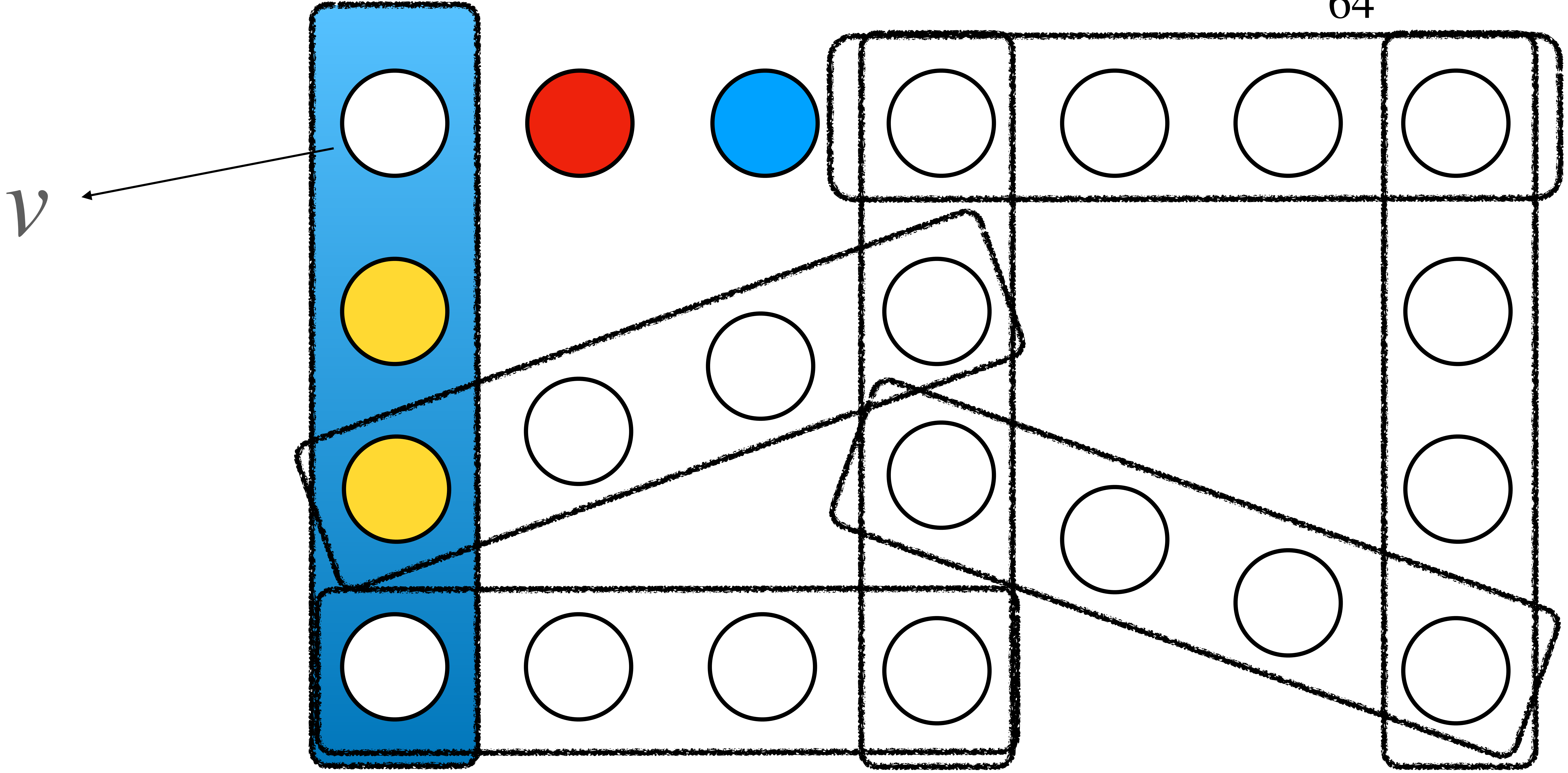
Example: hypergraph coloring

$Q = \{$ red, blue, yellow, purple $\}^V$, "freezing" threshold $p' = \frac{1}{64}$



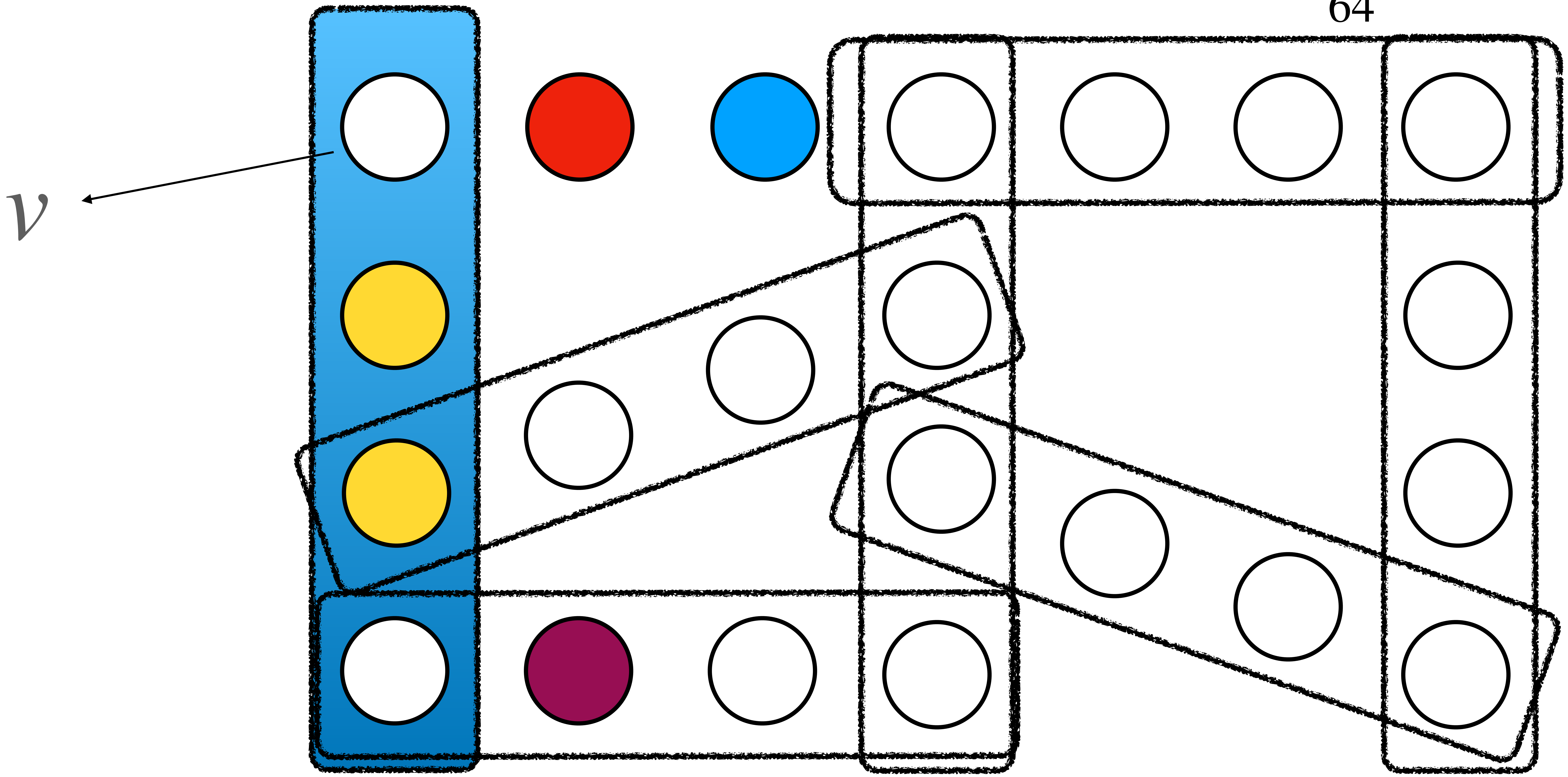
Example: hypergraph coloring

$Q = \{$🔴, 🔵, 🟡, 🟣$\}^V$, "freezing" threshold $p' = \dfrac{1}{64}$



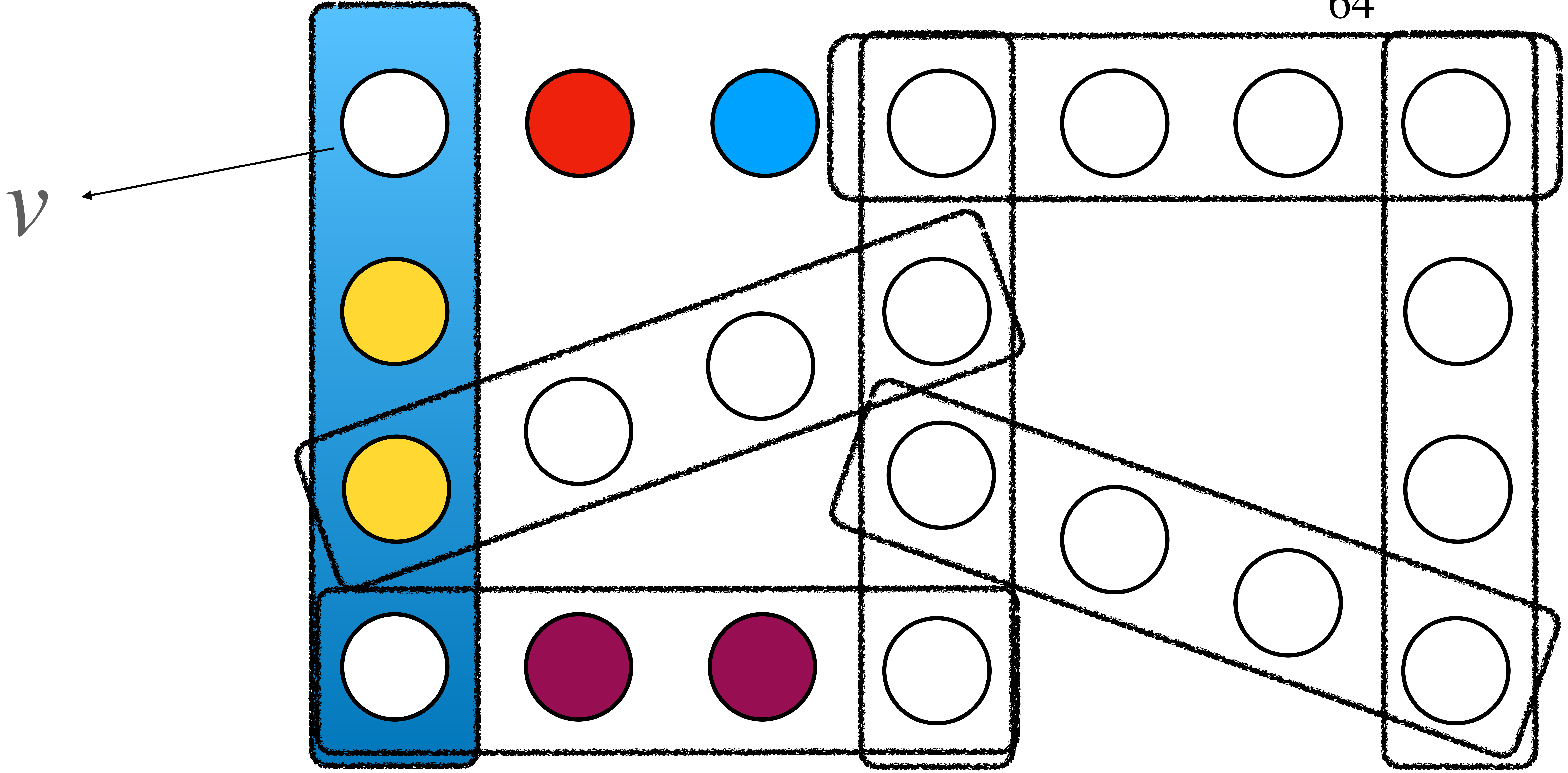
Example: hypergraph coloring

$Q = \{$ red, blue, yellow, purple $\}^V$, "freezing" threshold $p' = \frac{1}{64}$



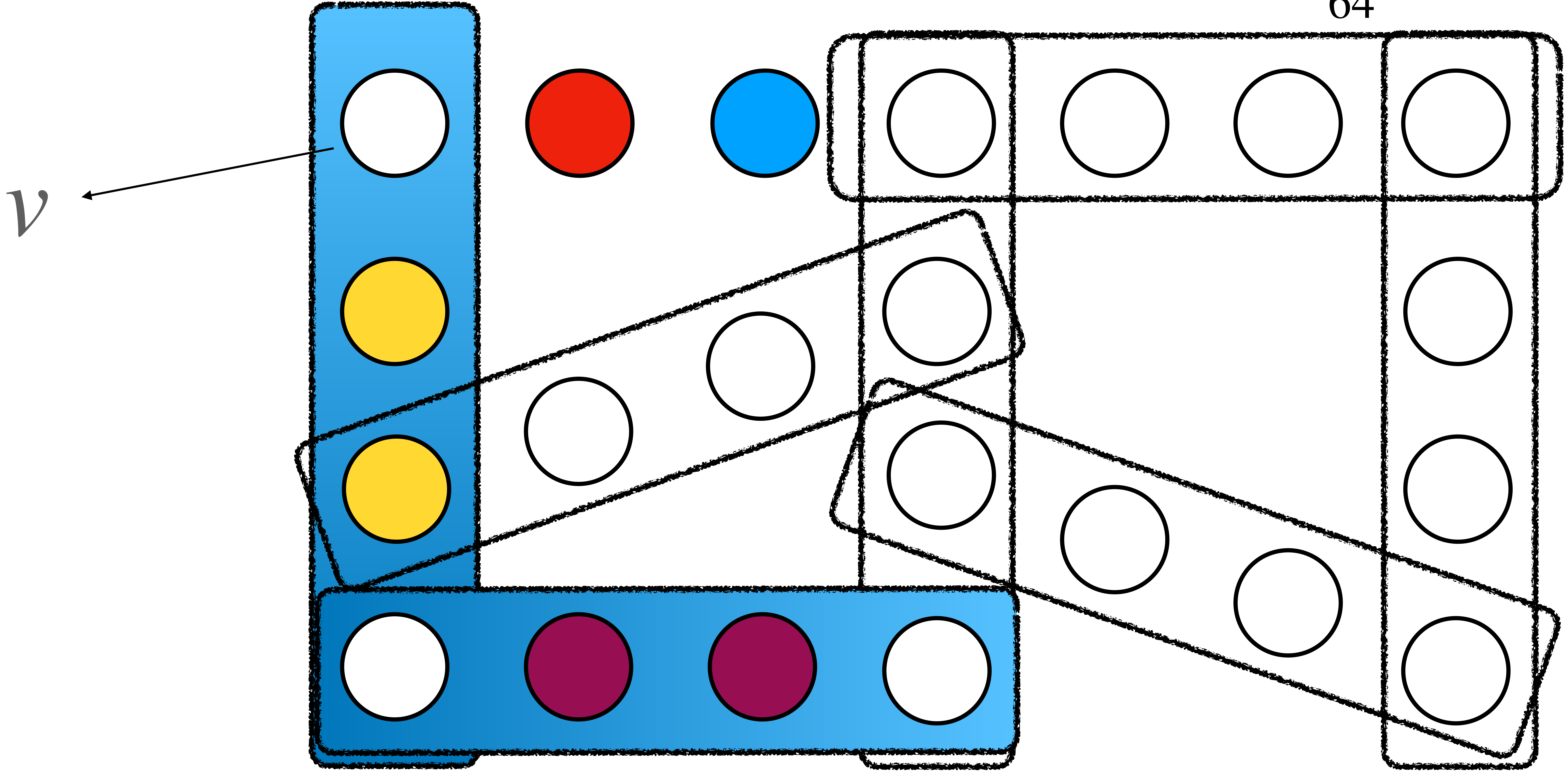
Example: hypergraph coloring

$Q = \{$🔴, 🔵, 🟡, 🟣$\}^V$, "freezing" threshold $p' = \frac{1}{64}$



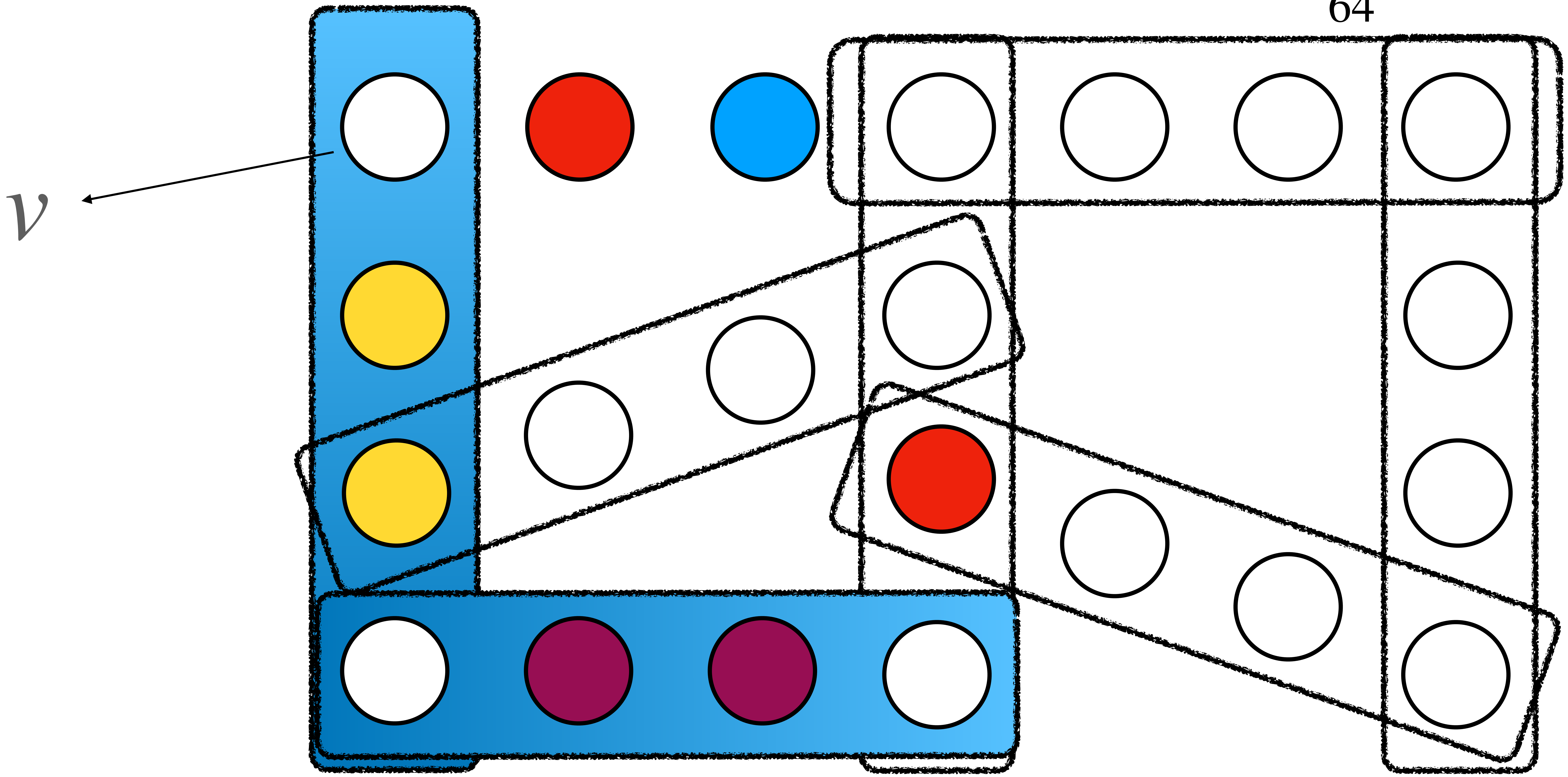
Example: hypergraph coloring

$Q = \{$●, ●, ●, ●$\}^V$, "freezing" threshold $p' = \dfrac{1}{64}$



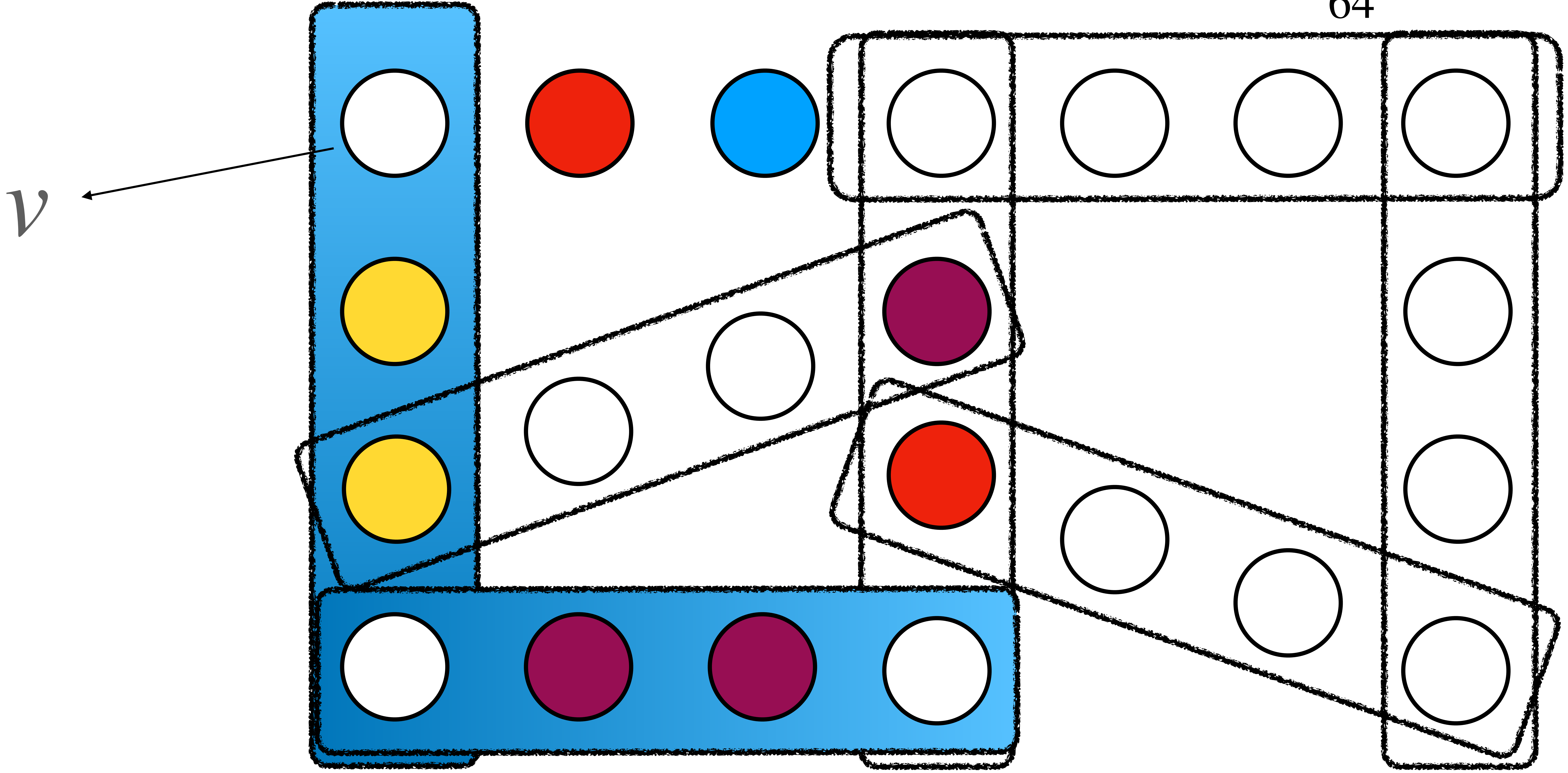
Example: hypergraph coloring

$Q = \{$●, ●, ●, ●$\}^V$, “freezing” threshold $p' = \frac{1}{64}$



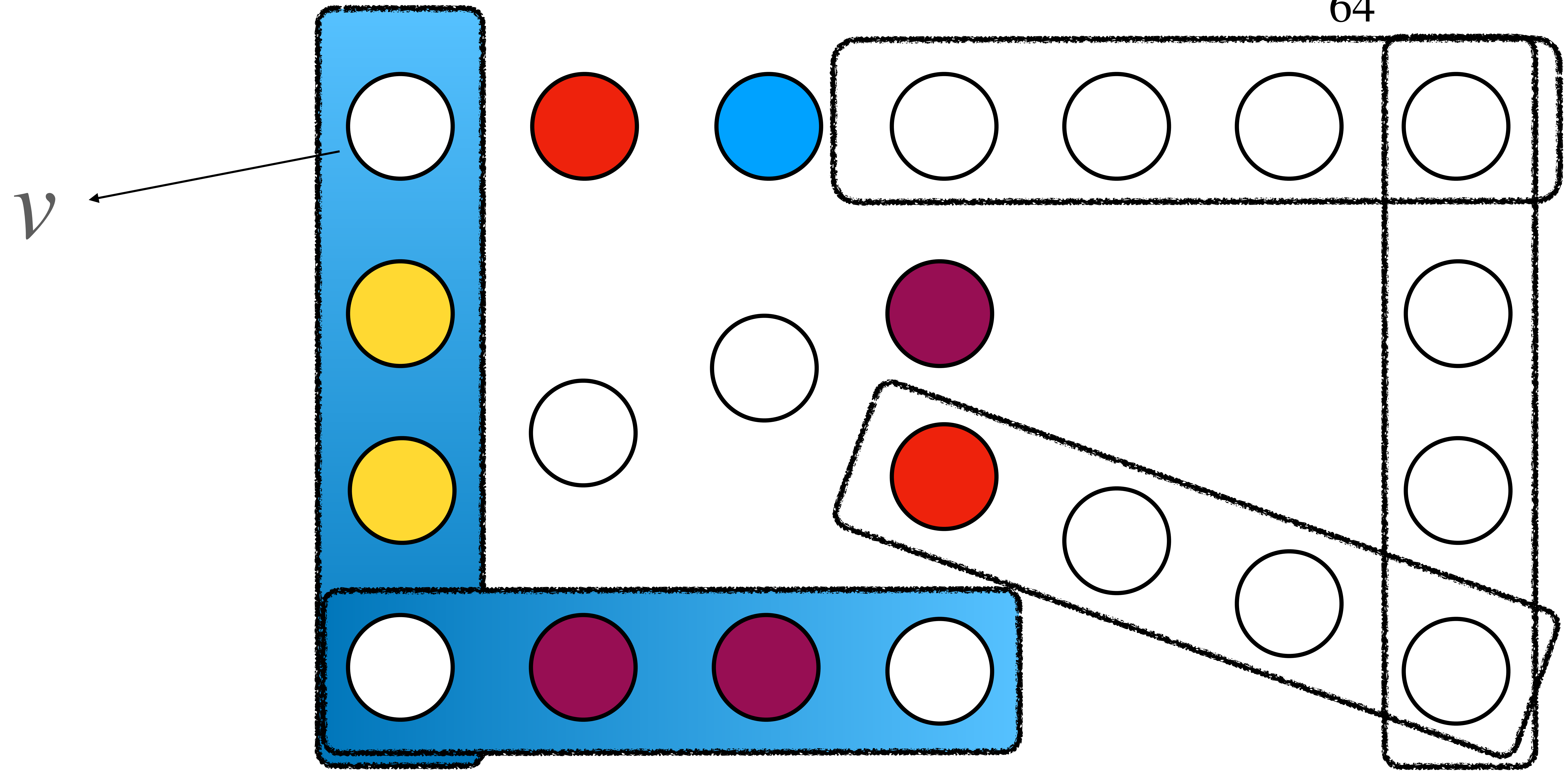
Example: hypergraph coloring

$Q = \{$●, ●, ●, ●$\}^V$, "freezing" threshold $p' = \frac{1}{64}$



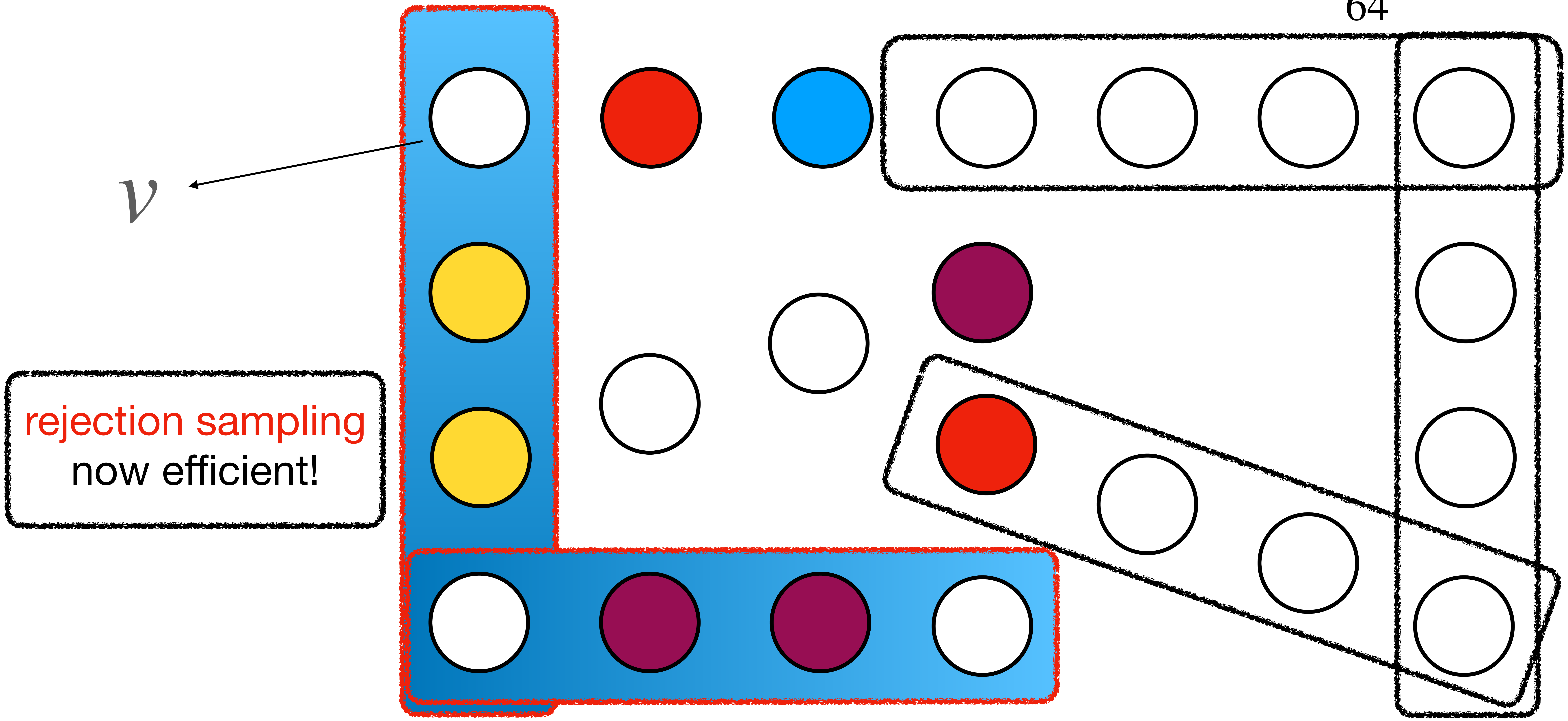
Example: hypergraph coloring

$Q = \{$ red, blue, yellow, purple $\}^V$, "freezing" threshold $p' = \frac{1}{64}$



Example: hypergraph coloring

$Q = \{\bullet, \bullet, \bullet, \bullet\}^V$, "freezing" threshold $p' = \dfrac{1}{64}$



Marginal Sampler

MarginSample(v, X): draw from μ_v^X

Choose $r \in [0,1]$ uniformly at random;

If $r < q\theta$ then returns the $\lceil r/\theta \rceil$-th value in $Q = [q]$;

else return ***MarginOverflow***(v, X);

μ_v^X: μ_v conditional on X

local uniformity

[Haeupler, Saha, Srinivasan '11]:

worst-case LLL cond. $\Longrightarrow \mu_v^X \geq \theta$

where $\theta = (1 - o(1))1/q$

MarginOverflow(v, X): draw from $(\mu_v^X - \theta)/(1 - q\theta)$

While $u = \mathrm{NextVar}(X, v) \neq \perp$

$X_u \leftarrow$ ***MarginSample***(u, X);

Draw from $(\mu_v^X - \theta)/(1 - q\theta)$ by *Bernoulli factory* accessing an oracle drawing from μ_v^X, realized by *rejection sampling*;

$\mathrm{NextVar}$: subroutine for choosing the next variable to sample

(informal) boundary variable over connected component of frozen constraints containing v

The main sampling algorithm

Main Sampling Algorithm:

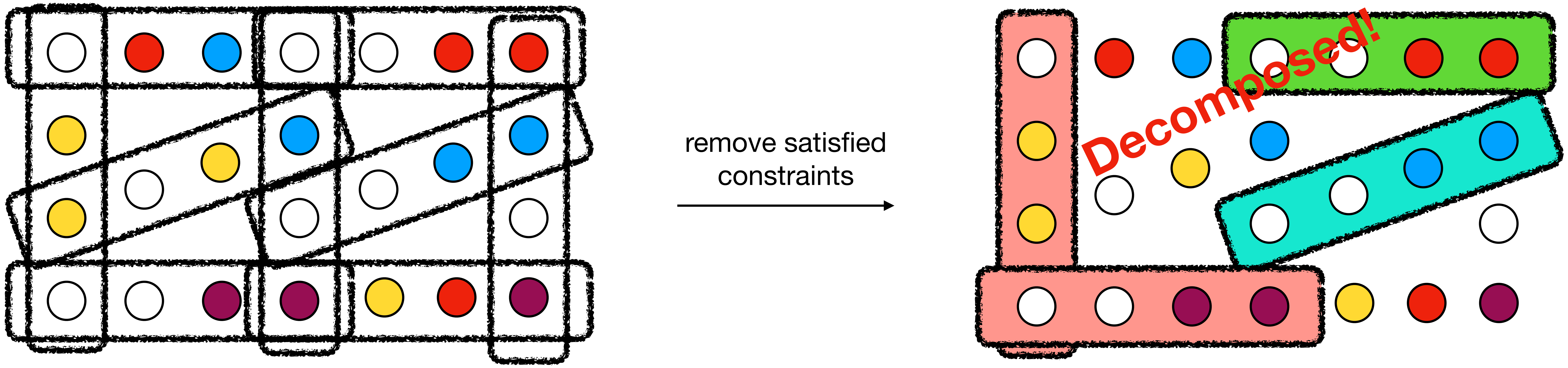
Initially, X is an empty partial assignment (with no variable assigned);

For $i = 1, \ldots, n$ do:

  if v_i is not involved in any constraint c frozen by X, then

    $X_{v_i} \leftarrow$ ***MarginSample***(v_i, X);

Complete X to a uniform random satisfying assignment by *rejection sampling*;



The main sampling algorithm

Main Sampling Algorithm:

Initially, X is an empty partial assignment (with no variable assigned);

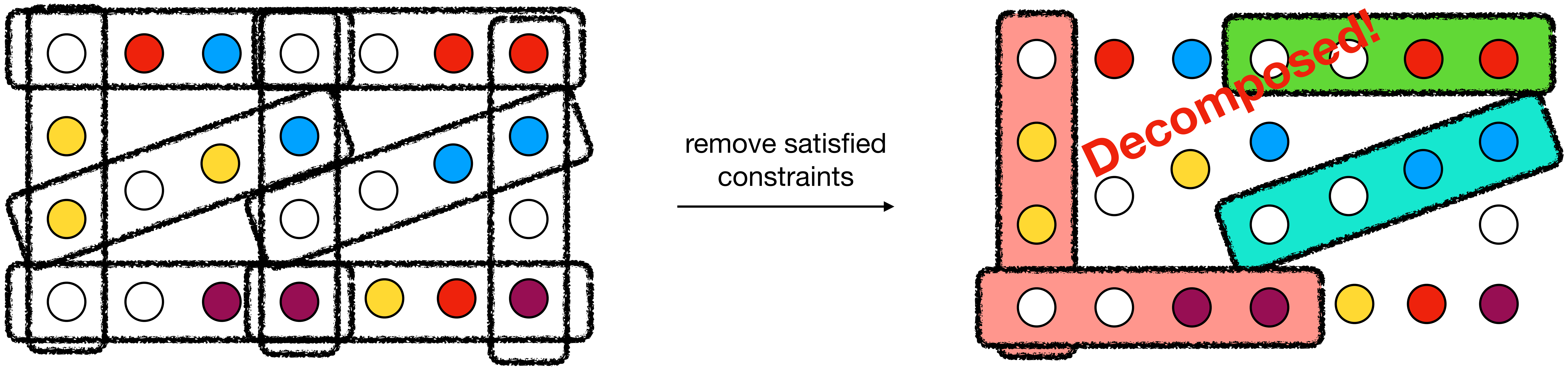
For $i = 1, \ldots, n$ do:

  if v_i is not involved in any constraint c frozen by X, then

    $X_{v_i} \leftarrow$ ***MarginSample***(v_i, X);

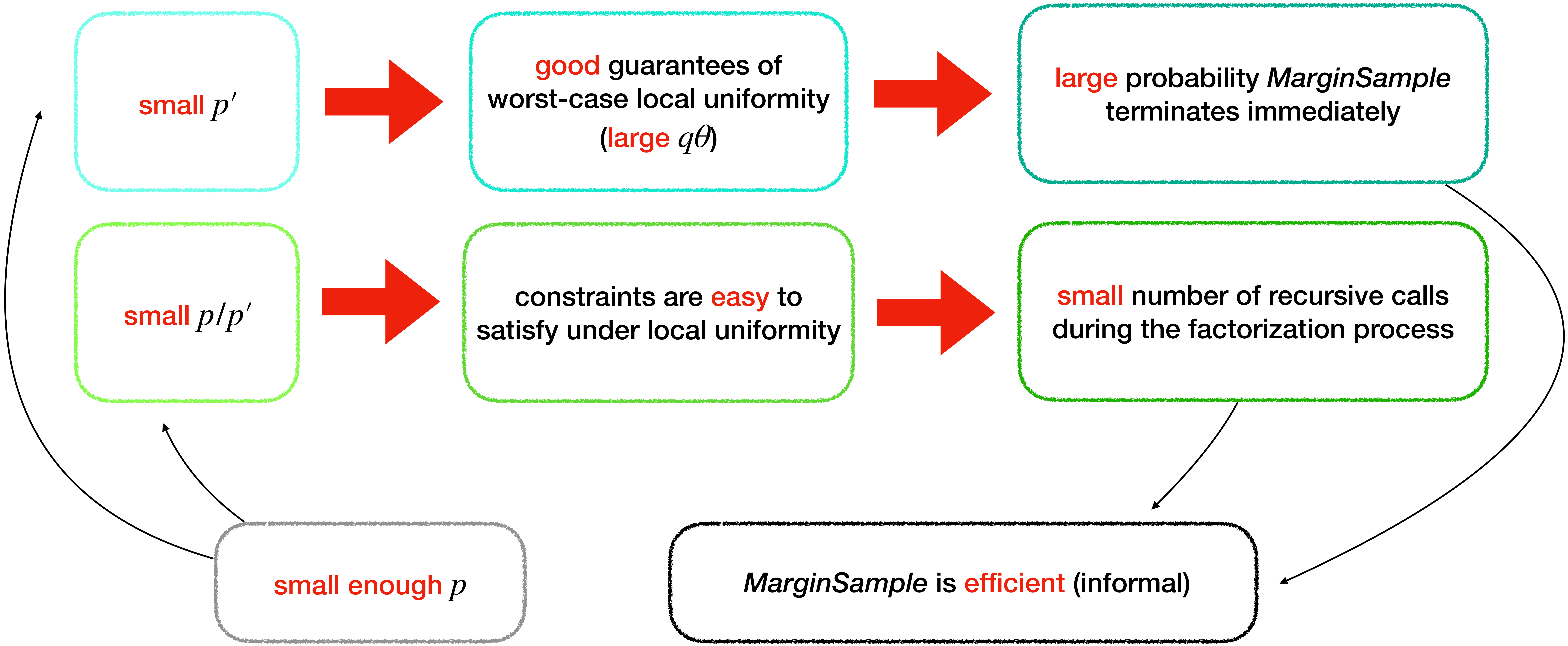
Complete X to a uniform random satisfying assignment by *rejection sampling*;

chain rule→correctness!



The freezing threshold p'

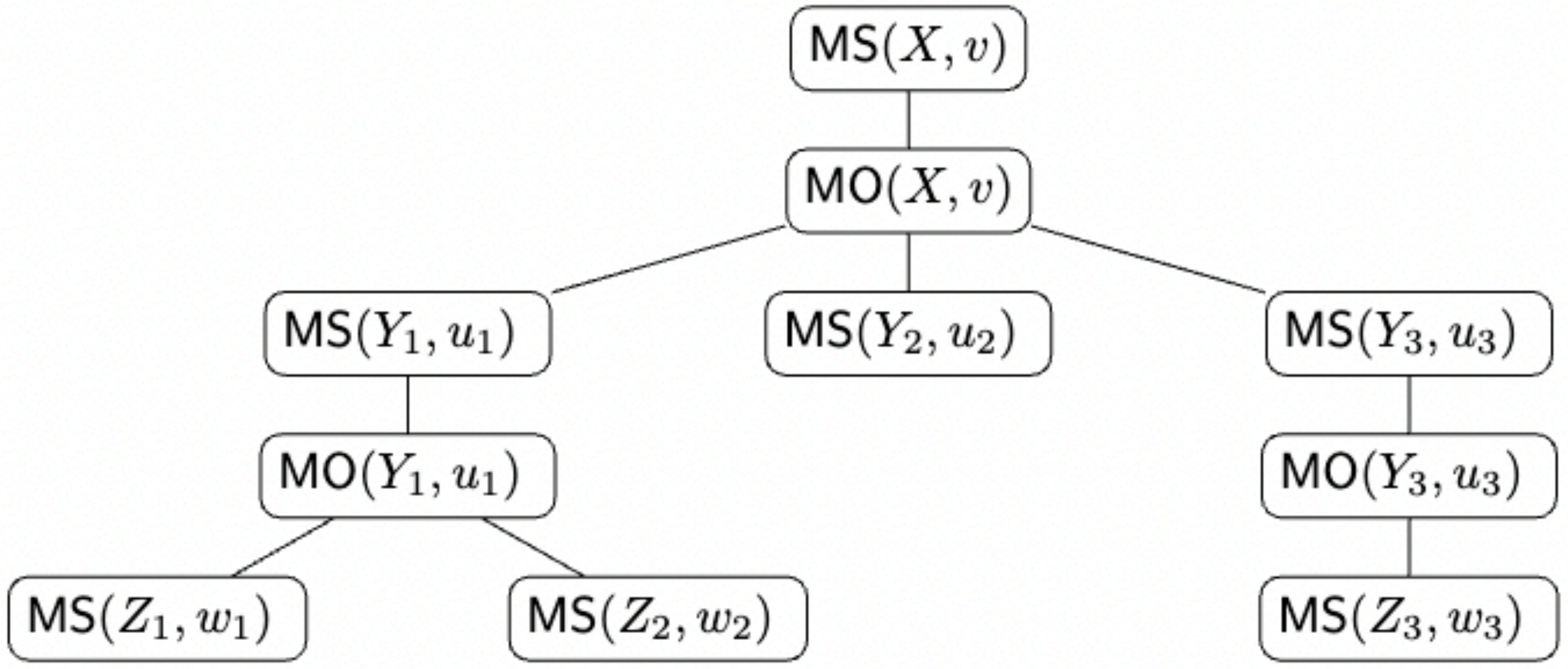
and how it relates to efficiency of MarginSample



Efficiency of the algorithm

MS : MarginSample

MO : MarginOverflow



Behavior of the algorithm: a branching process!

$\mathrm{MS}(X, v)$: offspring $\mathrm{MO}(X, v)$ with probability $1 - q\theta$

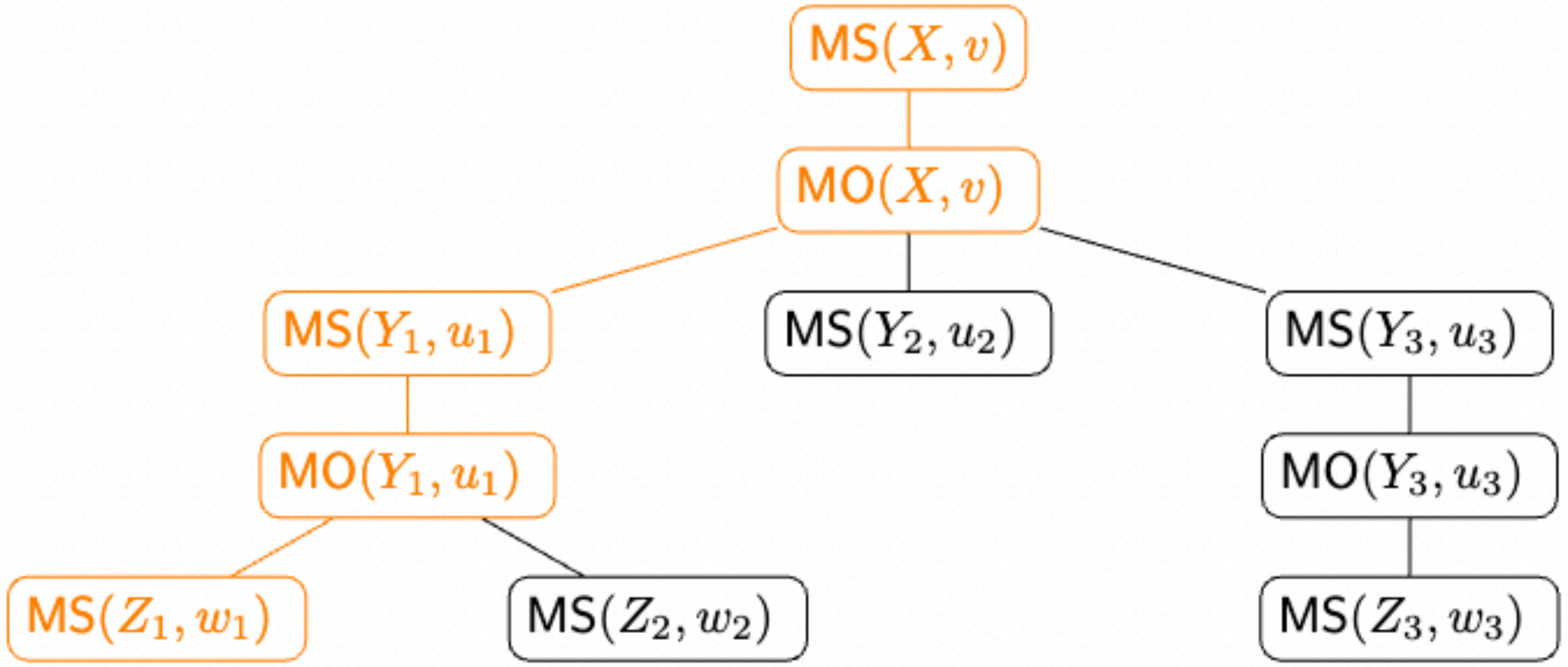
$\mathrm{MO}(X, v)$: offspring $\mathrm{MS}(Y_1, u_1), \mathrm{MS}(Y_2, u_2), \ldots,$ with distribution determined by (X, v)

Challenges

- The number of offsprings of $\mathrm{MO}(X, v)$ cannot be locally upper bounded (may be up to n)
- The offspring distribution of $\mathrm{MO}(X, v)$ is dependent with (X, v) (multi-type BP with exponentially many types)

Efficiency of the algorithm

MS : MarginSample
MO : MarginOverflow



Behavior of the algorithm: a branching process!

$\mathrm{MS}(X, v)$: offspring $\mathrm{MO}(X, v)$ with probability $1 - q\theta$

$\mathrm{MO}(X, v)$: offspring $\mathrm{MS}(Y_1, u_1), \mathrm{MS}(Y_2, u_2), \ldots,$ with distribution determined by (X, v)

Percolation-style analysis

analyze the probability each path occurs

Witness argument

A long path $\longrightarrow$ a large $\{2,3\}$-tree [Alon' 91] with ind. bad events

Bad events

Type 1: $\mathrm{MS}(X, v) \longrightarrow \mathrm{MO}(X, v)$ (probability $q\theta$)

Type 2: $c \in \mathscr{C}$ is frozen (probability roughly p'/p)

The Δ^7 bound is from balancing the probability of the two bad events and the structure of $\{2,3\}$-tree

Summary

We give the first polynomial-time algorithm for sampling general CSP solutions in the LLL regime with unbounded degree.

We introduce a new technique for sampling LLL: recursive marginal sampler.

Summary

We give the first polynomial-time algorithm for sampling general CSP solutions in the LLL regime with unbounded degree.

We introduce a new technique for sampling LLL: recursive marginal sampler.

Open Problems

The exact threshold: closing the gap between $p\Delta^7 \lesssim 1$ and $p\Delta^2 \gtrsim 1$ for sampling LLL

Generalizations: sampling LLL for non-uniform distribution and/or non-variable framework

Extending the idea: other applications of the recursive marginal sampler, or shed some light on the design for Markov chain algorithms?

Summary

We give the first polynomial-time algorithm for sampling general CSP solutions in the LLL regime with unbounded degree.

We introduce a new technique for sampling LLL: recursive marginal sampler.

Thank you!

Open Problems

The exact threshold: closing the gap between $p\Delta^7 \lesssim 1$ and $p\Delta^2 \gtrsim 1$ for sampling LLL

Generalizations: sampling LLL for non-uniform distribution and/or non-variable framework

Extending the idea: other applications of the recursive marginal sampler, or shed some light on the design for Markov chain algorithms?