Distributed Metropolis sampler with optimal parallelism

Weiming Feng
Nanjing University

Joint work with: Thomas P. Hayes (University of New Mexico)
Yitong Yin (Nanjing University)

SODA 2021, Online

Sequential Metropolis chain (for graphical model)

Metropolis chain for graph coloring

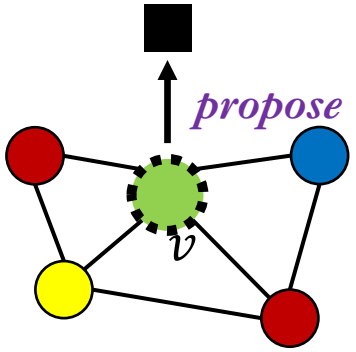
Input: a graph $G = (V, E)$, a set of q-colors $[q] = \{1, 2, \ldots, q\}$

Start from an arbitrary coloring $X \in [q]^V$

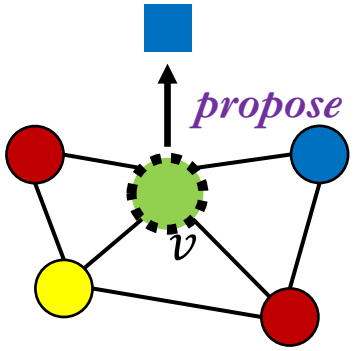
For each t from 1 to T **do**

- pick a node $v \in V$ u.a.r.
- **propose** a color $c \in [q]$ u.a.r.
- **if** $c \neq X(u)$ for all neighbors $u \in N(v)$

  update $X_v \leftarrow c$ (**accept**)
- **otherwise**, keep X_v unchanged (**reject**)



Accept the proposed color

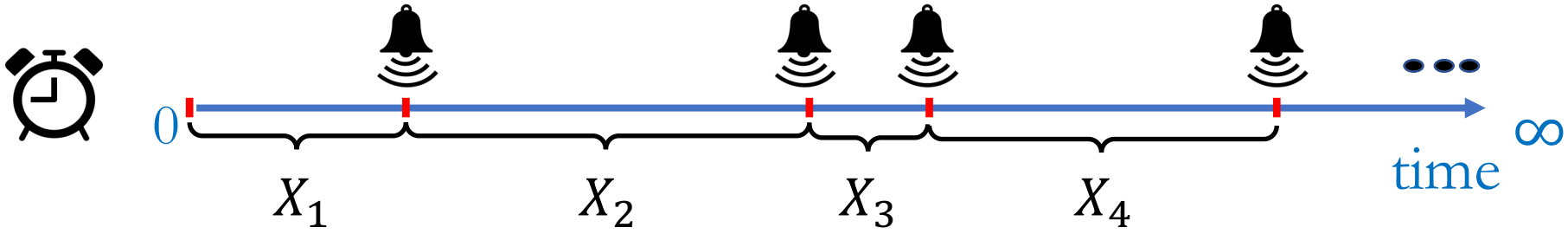


Reject the proposed color

update **one variable** in each iteration

sequential algorithm

Continuous-time Metropolis chain



Poisson clock: each X_i follows ***exponential distribution*** with mean 1.

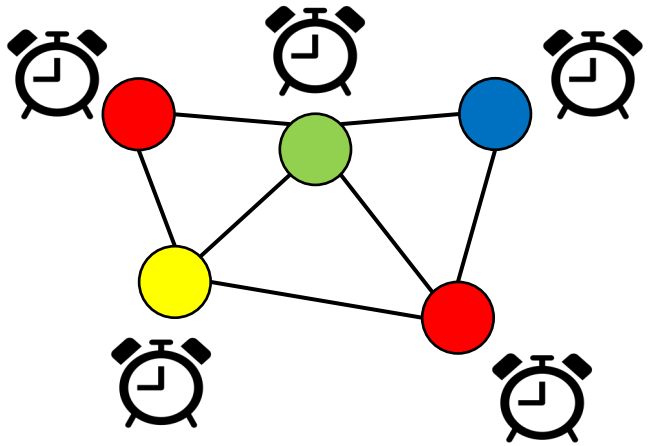
The continuous-time Metropolis chain

Start from an arbitrary proper coloring $Y \in [q]^V$

Each node has an **i.i.d. Poisson clock**

For t from 0 to $T \in \mathbb{R}^+$ **continuously do**

- whenever **the clock at** $v \in V$ ***rings***,
  update Y_v as in Metropolis chain



continuous-chain Y_T = discrete chain $X_{N(T)}$

$N(T) \sim \text{Poisson}(nT)$, where $n = |V|$

$$Y_T \approx X_{nT}$$

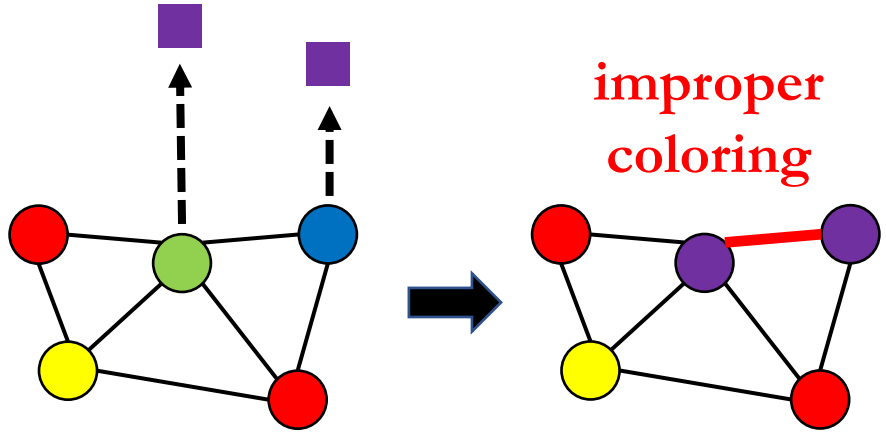
Continuous-time Metropolis chain

The continuous-time Metropolis sampler

- ~~**parallel**~~ **sequential process**
- updates resolved ***one-by-one*** according to Poisson clocks.

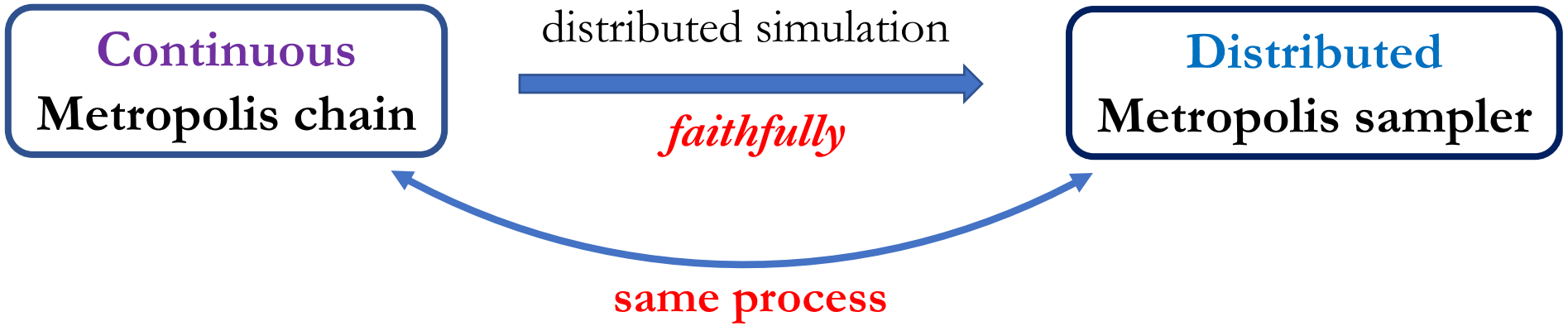
The algorithm in distributed system:

- allow **concurrent updates**
- avoid **race condition**



Concurrent updates may cause improper coloring

Problem: distributed Metropolis sampler



Message-passing model

Network: graph $G = (V, E)$

node: computer;

edge: bidirectional channel.

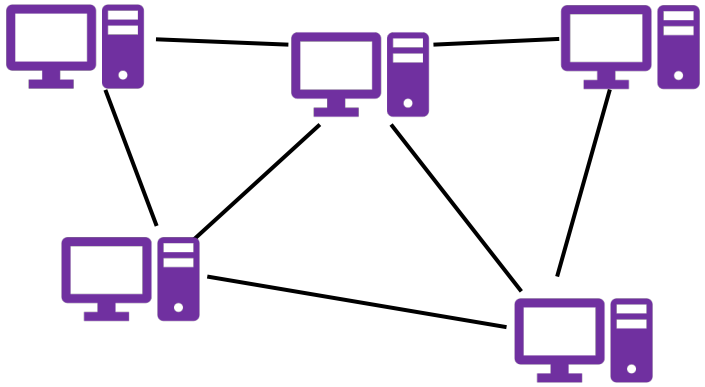
Local computation: **free**.

Synchronous rounds: all nodes

perform local computation;

exchange messages with neighbors.

Time complexity: the number of **rounds**.



our algorithm
event-driven procedure

directly used

asynchronous
message passing model

Problem & algorithm

Problem: simulate the continuous-time Metropolis chain $(Y_t)_{t\in\mathbb{R}_{\geq 0}}$ to time T

Input for node v : color set $[q]$, initial color $Y_0(v) \in [q]$ and time T

Output at node v : random color $Y_T(v) \in [q]$

Phase-1 at node $v \in V$ (initialization) time complexity = one round

- Simulate Poisson clock to generate update times $0 < t_1^v < t_2^v < \cdots < t_{m(v)}^v < T$;
- Propose uniform random color $c_i^v \in [q]$ for each update time t_i^v
- Send $Y_0(v)$ and $(t_i^v, c_i^v)_{1\leq i\leq m(v)}$ with all neighbors $u \in N(v)$.

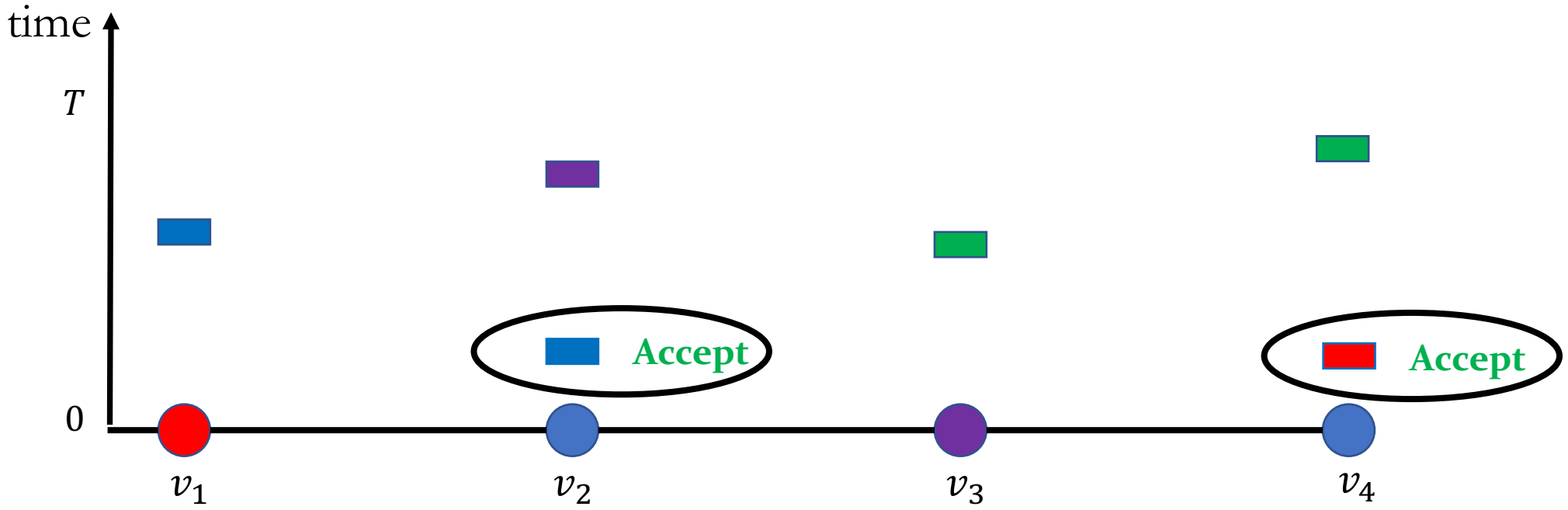
Phase-2 at node $v \in V$ (resolving updates)

Assumption: for all $u \in N(v)$, node v knows $Y_0(u)$ and $(t_i^u, c_i^u)_{1\leq i\leq m(u)}$

For each i from 1 to $m(v)$ **do**

resolve the update (t_i^v, c_i^v) according to some strategy

Warm-up: a straightforward simulation



All neighbors $N(v)$ resolved updates **before** time t

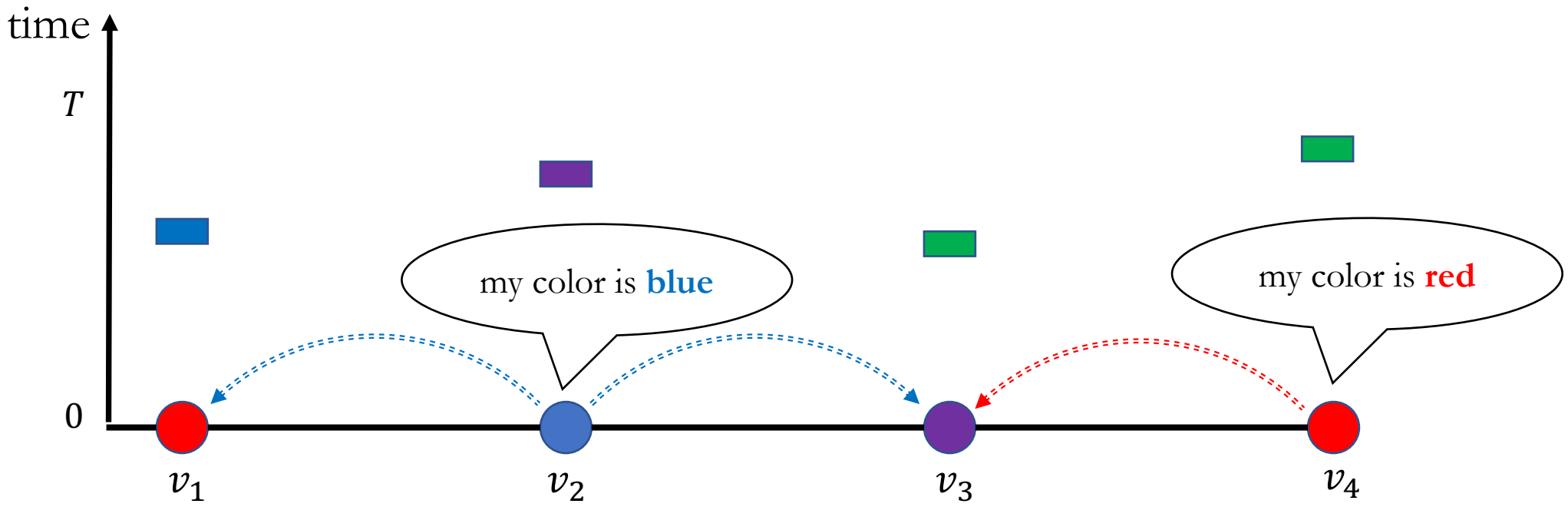
→

Node v resolves updates at time t

→

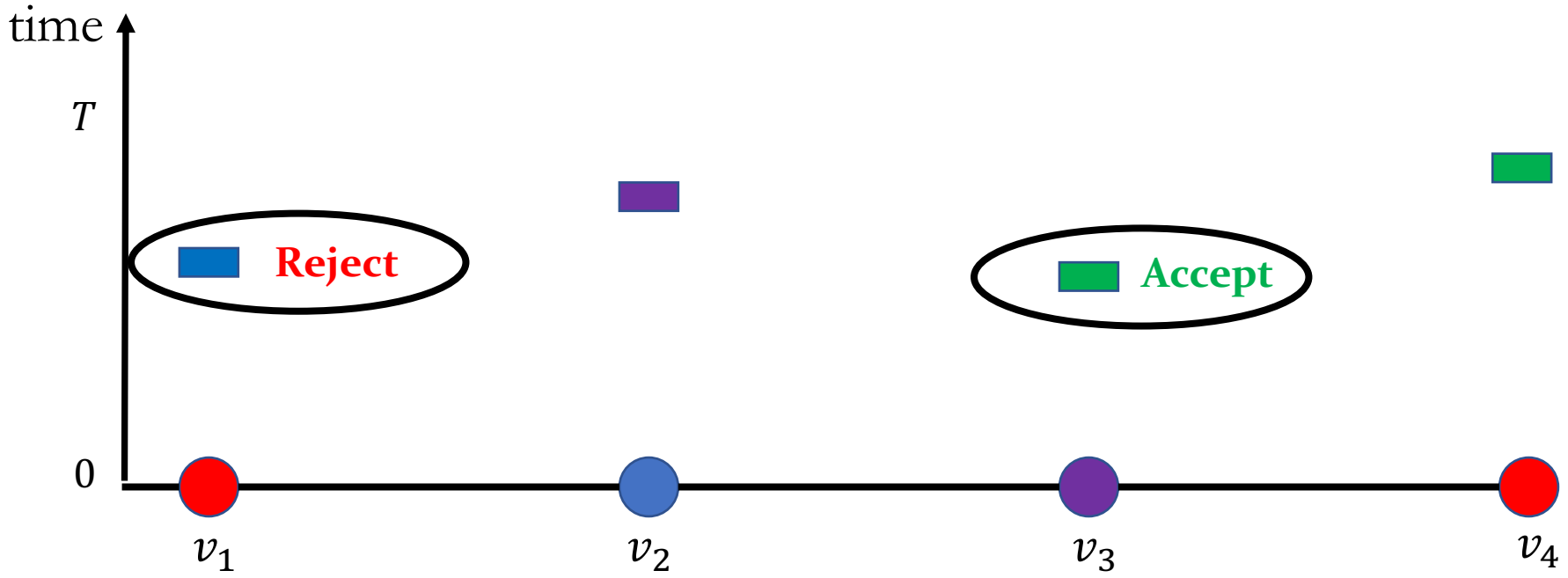
Node v broadcasts new color to neighbors

Warm-up: a straightforward simulation



All neighbors $N(v)$ resolved updates **before** time t → Node v resolves updates at time t → Node v broadcasts new color to neighbors

Warm-up: a straightforward simulation



All neighbors $N(v)$ resolved updates **before** time t

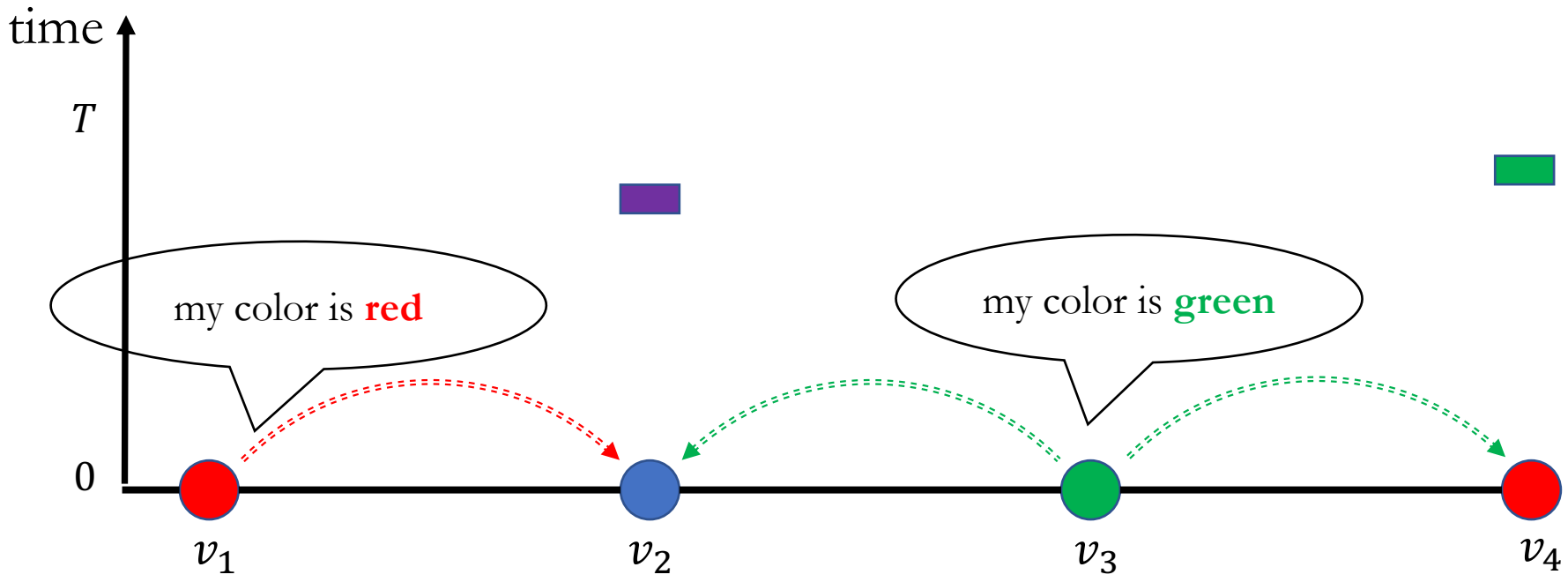
→

Node v resolves updates at time t

→

Node v broadcasts new color to neighbors

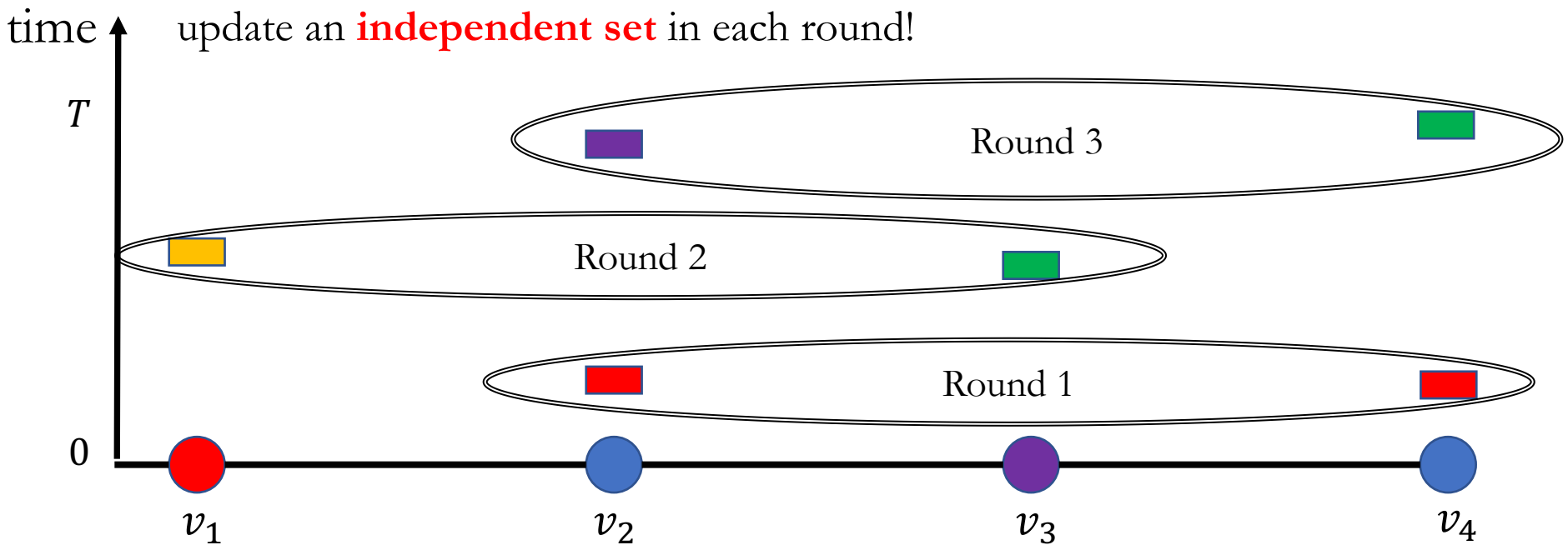
Warm-up: a straightforward simulation



All neighbors $N(v)$ resolved updates **before** time t → Node v resolves updates at time t → Node v broadcasts new color to neighbors

resolve update at time t
wait
neighbors resolve updates at time $< t$

NOT allow **adjacent nodes**
resolve updates in the **same round**

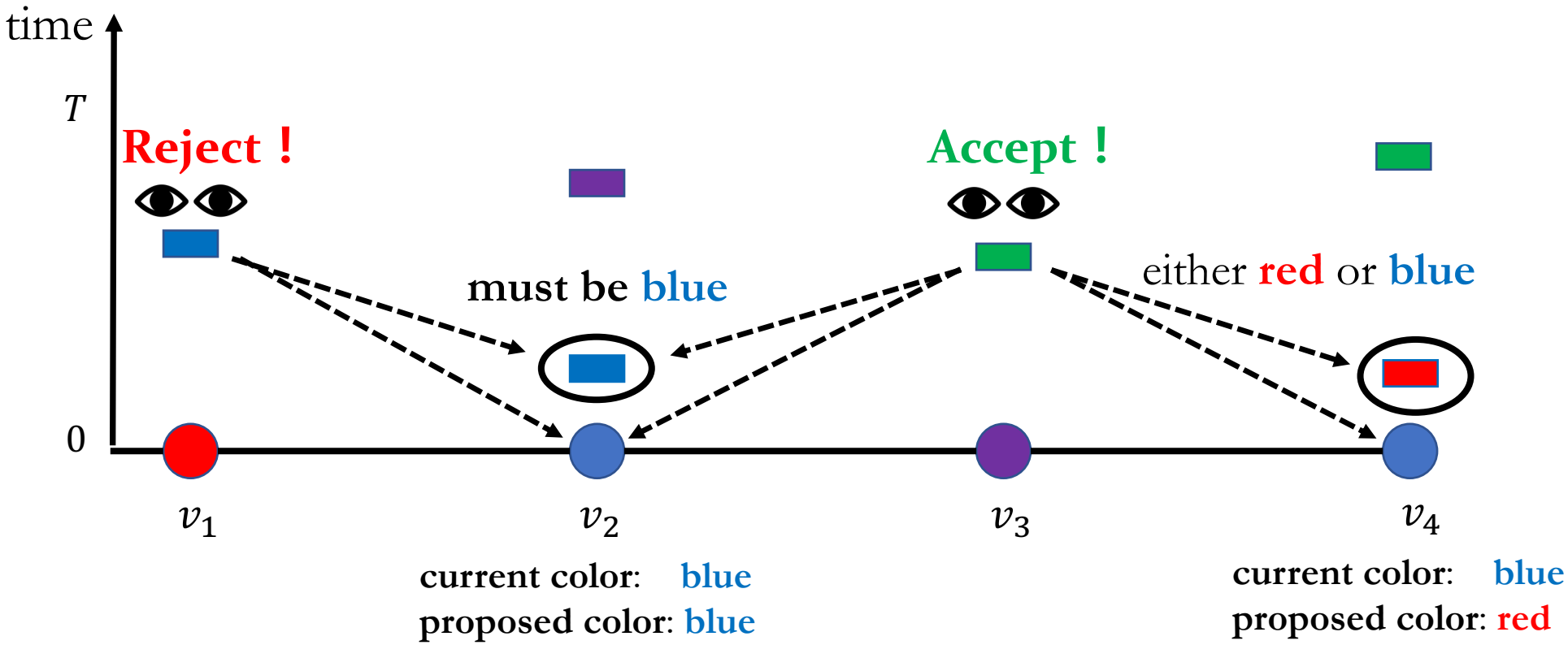


Proposition (for any single-site dynamics)

The straightforward simulation has cost $O(\Delta T + \log n)$ w.h.p.

Δ =max degree, slow in dense graphs

Our technique: resolve update in advance



Our technique: resolve update in advance

Phase-2 at node $v \in V$:

- **For** each i from 1 to m_v **do**
  - Listen to each channel and receive messages from neighbors.
  - If v gets **enough information**
    - resolve the update (t_i^v, c_i^v);
    - pass "**Accept**" or "**Reject**" to all neighbors.

Resolve update in advance:

Resolve update at time t ***before*** neighbors resolve all updates at time $< t$.

Resolve update in advance (coloring)

Resolution of the update (t, c) at node $v \in V$

For each neighbor $u \in N(v)$, maintain set

S_u = set of ***possible colors*** for u at time t

Upon $c \notin S_u$ for all $u \in N(v)$

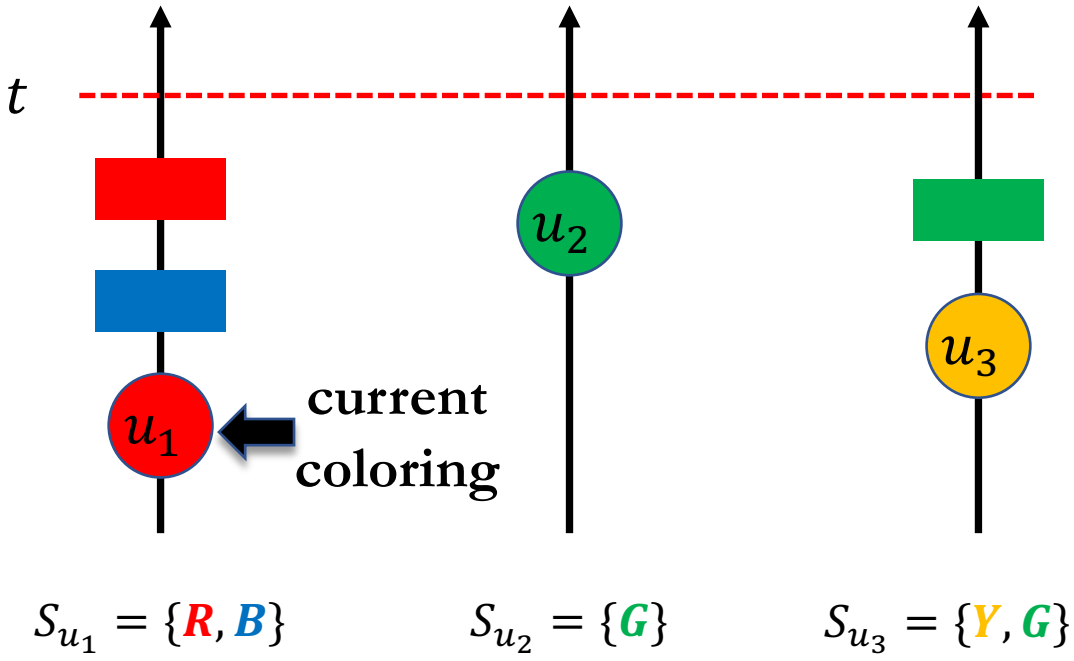
Resolve update, pass "**Accept**" to all neighbors

Upon $S_u = \{c\}$ for some $u \in N(v)$

Resolve update, pass "**Reject**" to all neighbors

Upon receiving messages from neighbors $u \in N(v)$

Update the set S_u;



Main results

Theorem [This work] if $q \geq C\Delta$ ($\Delta =$ max degree) for some constant $C > 0$,

w.h.p. simulate **continuous-time Metropolis chain** to time T within $O(T + \log n)$ rounds.

Corollary [This work] if $q \geq C\Delta$ for some constant $C > 0$,

w.h.p. simulate **discrete-time Metropolis chain** for M steps within $O\left(\frac{M}{n} + \log n\right)$ rounds.

$\boldsymbol{M = \Omega(n \log n)}$ ⟹ $\boldsymbol{O\left(\frac{M}{n}\right)}$ **cost (factor $\boldsymbol{\Omega(n)}$ speed-up)**

general **lower bound** for **mixing**
[Hayes and Sinclair 05]

optimal speed-up

General Metropolis chain

Graphical model in $G = (V, E)$ with Gibbs distribution μ

- node: random variable in $[q]$; edge: pairwise interaction;
- examples: uniform proper coloring, hardcore model, Ising model…

Metropolis update for graphical model with current configuration $Y \in [q]^V$

When the Poisson clock at node $v \in V$ **rings**, let $c = Y_v$;

- **Propose** a random value $c' \sim \boldsymbol{\pi_v}$;
- **With prob.** $P_{AC} = f^v_{c,c'}\left(Y_{N(v)}\right)$, **accept**;
- **With prob.** $P_{RE} = 1 - f^v_{c\to c'}\left(Y_{N(v)}\right)$, **reject**.

Proposal distribution at v: π_v over $[q]$;

Metropolis filter at v with current value c and proposal c':

$$f^v_{c\to c'}: [q]^{N(v)} \to [0,1];$$

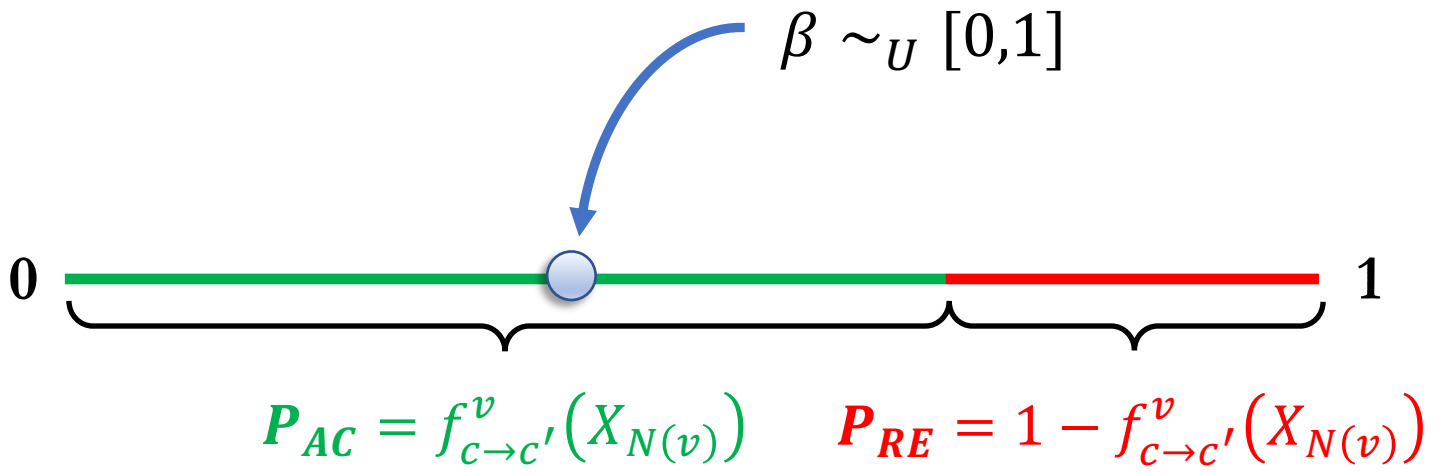
Coloring

$$f^v_{c\to c'}\left(Y_{N(v)}\right) = \prod_{u\in N(v)} \mathbf{1}[c' \neq Y_u]$$

Metropolis update for graphical model with current configuration $Y \in [q]^V$

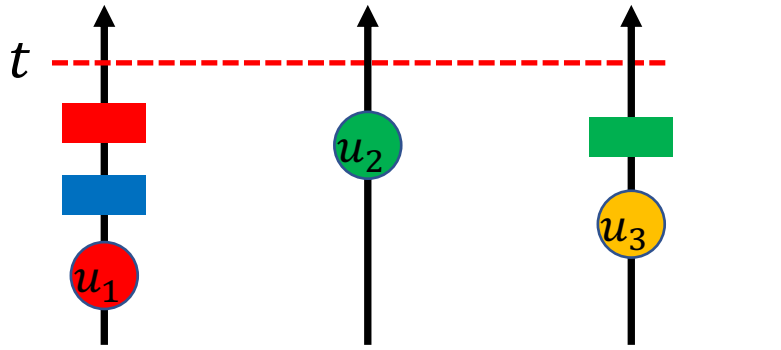
When the Poisson clock at node $v \in V$ **rings**, let $c = X_v$;

- **Sample** a random real number $\beta \in [0,1]$ uniformly at random;
- **Propose** a random value $c' \sim \boldsymbol{\nu_v}$;
- **Compute threshold** $f^v_{c\to c'}(X_{N(v)})$;
- **If** $\beta < f^v_{c\to c'}(X_{N(v)})$, **accept**;
- **If** $\beta \geq f^v_{c\to c'}(X_{N(v)})$, **reject**.

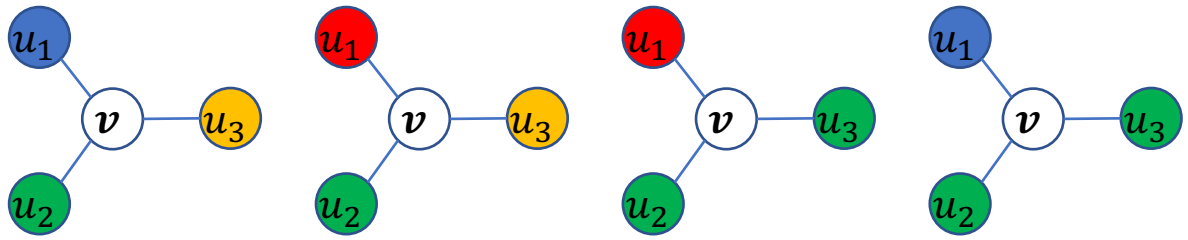


Resolve update in advance (general)

Node $v \in V$ resolves the update (t, c) ***before*** the $Y_t(N(v))$ is known



$S_{u_1} = \{R, B\}$ $S_{u_2} = \{G\}$ $S_{u_3} = \{Y, G\}$



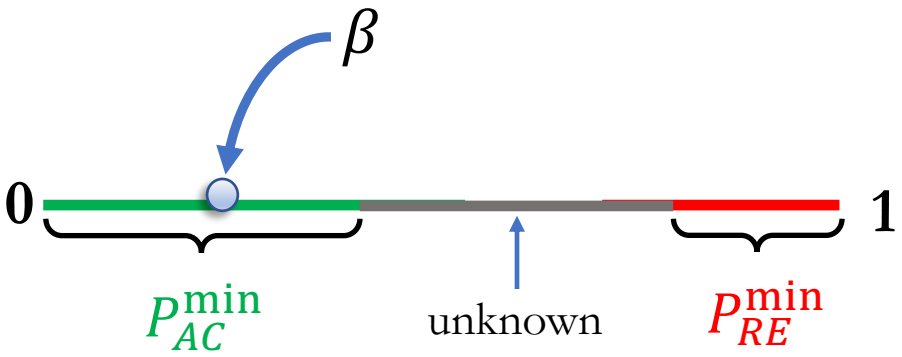
Set of ***possible configurations*** on $N(v)$

$\mathcal{C} = \bigotimes_{u \in N(v)} S_u$

$P_{AC}^{\min}, P_{RE}^{\min}$: **min prob.** for **accept** and **reject**

$$P_{AC}^{\min} = \min_{\sigma \in \mathcal{C}} f_{c \to c'}^{v}(\sigma),$$

$$P_{RE}^{\min} = \min_{\sigma \in \mathcal{C}} (1 - f_{c \to c'}^{v}(\sigma))$$



Resolve update in advance if

$\beta < P_{AC}^{\min}$ *or* $\beta \geq 1 - P_{RE}^{\min}$

Main result

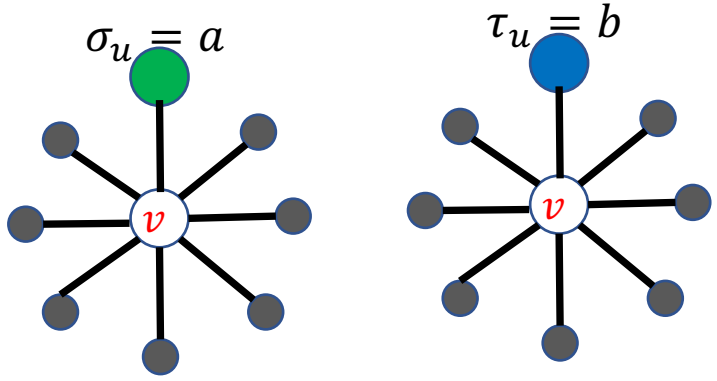
Theorem [this work]

If all Metropolis filters satisfy a **Lipschitz condition**,
w.h.p. simulate continuous-Metropolis chain to time T with cost $O(T + \log n)$.

Let any $(u, v) \in E$ and $a, b, c, c' \in [q]$

$$\delta_{u,a,b} f^{v}_{c \to c'} = \max_{\sigma,\tau} \left| f^{v}_{c \to c'}(\sigma) - f^{v}_{c \to c'}(\tau) \right|$$

where $\sigma_u = a, \tau_u = b$ and $\sigma_w = \tau_w$ for all $w \in N(v) \backslash \{u\}$.



Lipschitz condition:
There is a constant C such that for any $(u, v) \in E$ and $a, b, c, c' \in [q]$

$$\mathbb{E}_{c' \sim \pi_v} \left[\delta_{u,a,b} f^{v}_{c \to c'} \right] \leq C/\Delta$$

Main theorem

Models	Lipschitz condition	Uniqueness condition
q-coloring	$q > C\Delta$	$q > \Delta$
Hardcore	$\lambda < \frac{C}{\Delta}$	$\lambda < \frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta}} \approx \frac{e}{\Delta}$
Ising	$1 - \exp(-2\lvert\beta\rvert) < \frac{C}{\Delta}$	$1 - \exp(-2\lvert\beta\rvert) < \frac{2}{\Delta}$

$C > 0$ is an **arbitrary constant**

The Metropolis chain can be ***simulated efficiently*** even if it is ***slow mixing***.

Correctness of simulation

Our simulation algorithm $(\hat{Y}_t)_{t\in[0,T]}$

Perfect Coupling

Continuous-time Metropolis chain $(Y_t)_{t\in[0,T]}$

Theorem of correctness

The simulation algorithm will **terminate eventually**, $(\hat{Y}_t)_{t\in[0,T]}$ and $(Y_t)_{t\in[0,T]}$ are the **same process**.

Efficiency of simulation

Resolution of the k-th update at node $v \in V$

Self-triggered resolution

node v resolves $(v, k-1)$ → node v resolves (v, k)

Neighborhood-triggered resolution

neighbor $u \in N(v)$ resolves (u, j) → pass message → node v resolves (v, k)

unique predecessor for update (v, k)

Dependency chain

$(v_0, k_0 = 1)$ **trigger** (v_1, k_1) **trigger** (v_2, k_2) **trigger** (v_3, k_3) **trigger** (v_ℓ, k_ℓ)

time complexity $\leq$ length of the longest dependence chain $+ O(1)$

Proof idea: **long dependence chain** appears with **low probability**.

Fix a path $P = v_0, v_1, \ldots, v_\ell$ of length ℓ, where $v_{i+1} \in N(v_i) \cup \{v_i\}$.

Event $\mathcal{E}$: there is a dependency chain of length ℓ alone P,

i.e. there is a dependency chain $(v_i, k_i)_{1 \le i \le \ell}$.

Lipschitz condition with constant C

$$\Pr[\mathcal{E}] \le \left(\frac{eT}{\ell}\right)^{\ell} \times \left(\frac{2C}{\Delta}\right)^{R}$$

R: number of neighborhood-triggered resolutions

update times alone the chain are increasing
randomness of Poisson clocks

Updates alone the chain are triggered one by one
randomness of proposals & Metropolis filters

🎯 *the power of resolving update in advance*

union bound over all paths

Main Theorem: For Metropolis chain with **Lipschitz condition** with high probability, the cost is $O(T + \log n)$.

Summary

A distributed algorithm that **faithfully simulates** continuous Metropolis chain to time T s.t.

- w.h.p., the cost of simulation is $O(\Delta T + \log n)$;
- if a **Lipschitz condition** is satisfied, w.h.p., the cost of simulation is $O(T + \log n)$.

A new technique: **resolve update in advance**.

Open problems

Simulate the **general single-site dynamics** beyond the Metropolis chain

- Gibbs sampling (**conditional update rule**)
  - **Coloring**: update Y_v as a **uniform color** in $[\boldsymbol{q}] \setminus \{\boldsymbol{Y_u} \mid \boldsymbol{u} \in \boldsymbol{N(v)}\}$.
- New technique to resolve the conditional updates.

Thank you!