Distributed Algorithms for MCMC Sampling

Yitong Yin
Nanjing University

Shonan Meeting No. 162: Distributed Graph Algorithms
Outline

• Distributed Sampling Problem
  • Gibbs Distribution (distribution defined by local constraints)

• Algorithmic Ideas
  • Local Metropolis Algorithm
  • LOCAL Jerrum-Valiant-Vazirani
  • Local Rejection Sampling

• Distributed Simulation of Metropolis (with ideal parallelism)

MCMC: Markov chain Monte Carlo
Single-Site Markov Chain

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

at each step:

for a uniform random vertex $v$

propose a random color $c \in [q]$;
change $v$’s color to $c$ if it’s proper;

Metropolis Algorithm
($q$-coloring)
Single-Site Markov Chain in 1960s

Each vertex holds an independent rate-1 Poisson clock.

When the clock at $v$ rings:
- propose a random color $c \in [q]$;
- change $v$’s color to $c$ if it’s proper;

Metropolis Algorithm ($q$-coloring)

continuous time $T$ \hspace{2cm} discrete time $\theta(nT)$ sequential steps
Distributed Simulation of Continuous-Time Process

**Goal:** Give a distributed algorithm that **perfectly simulates** the time $T$ continuous Markov chain. (Have the same behavior given the same random bits.)

do NOT allow adjacent vertices update their colors in the same round:

$O(\Delta T)$ rounds

[Feng, Hayes, Y. ’19]:

$O(T + \log n)$ rounds w.h.p.
(under some mild condition)
2-Phase Paradigm

for each vertex $v \in V$:

Phase I:

- locally generate all update times $0 < t_1 < t_2 < \cdots < t_{M_v} < T$
  and proposed colors $c_1, c_2, \ldots, c_{M_v} \in [q]$;
- send the initial color and all $(t_i, c_i)_{1 \leq i \leq M_v}$ to all neighbors;

Phase II:

- For $i = 1, 2, \ldots, M_v$ do:
  once having received enough information:
  resolve the $i$-th update of $v$ and send the result
  ("Accept / Reject") to all neighbors;
for each vertex $v \in V$:

- For $i = 1, 2, \ldots, M_v$ do:
  - once having received *enough information*:
  - resolve the $i$-th update of $v$ and send the result ("Accept / Reject") to all neighbors;

“enough info” to resolve the $i$-th update at $v$: $(t_i^v, c_i^v)$

all adjacent updates before $t_i^v$ have been resolved and received by $v$

$\exists$ a path $v_1, v_2, \ldots, v_L$

$\#\text{rounds} > L \quad T > t_{i_1}^v > t_{i_2}^v > \cdots > t_{i_L}^v > 0$

which occurs w.p. $<(eT/L)^L$

$\#\text{rounds} = O(\Delta T + \log n)$ w.h.p.
Resolve Update In Advance

“enough info” to resolve the $i$-th update at $v$: $(t, c)$

\[
\begin{aligned}
&\text{curr-color} = \begin{cases} \\
\end{cases} \\
&c = \begin{cases} \\
\end{cases}
\end{aligned}
\]

If $c \notin \bigcup_{u \sim v} S_u(t)$: “Accept!”

$S_u(t)$ : set of possible colors of $u$ at time $t$
Resolve Update In Advance

“enough info” to resolve the $i$-th update at $v$: $(t, c)$

\[
\text{If } c \notin \bigcup_{u \sim v} S_u(t) : \text{“Accept!”}
\]

\[
\text{If } \exists u \sim v \text{ s.t. } S_u(t) = \{c\} : \text{“Reject!”}
\]
to resolve the $i$-th update at $v$: $(t, c)$

Construct $S_u(t)$ for every neighbor $u$ of $v$;

upon $c \notin \bigcup_{u \sim v} S_u(t)$:
   send “Accept!” to all neighbors and $i++$;

upon $\exists u \sim v$ s.t. $S_u(t) = \{c\}$:
   send “Reject!” to all neighbors and $i++$;

upon receiving “Accept!” or “Reject!” from neighbor $u$:
   update $S_u(t)$ accordingly;

curr-color = [image of colors]

$S_u(t)$ : current set of possible colors of $u$ at time $t$
to resolve the $i$-th update at $v$: $(t, c)$

Construct $S_u(t)$ for every neighbor $u$ of $v$;

**upon** $c \not\in \bigcup_{u \sim v} S_u(t)$:
- send “Accept!” to all neighbors and $i++$;

**upon** $\exists u \sim v$ s.t. $S_u(t) = \{c\}$:
- send “Reject!” to all neighbors and $i++$;

**upon** receiving “Accept!” or “Reject!” from neighbor $u$:
- update $S_u(t)$ accordingly;

$\#\text{round} > L \quad \rightarrow \quad \exists$ a path $v_1, v_2, \ldots, v_L$:

$\#\text{paths} \leq \Delta^L$

$\Pr < O\left(\frac{T}{qL}\right)^L \left\{ \begin{array}{l}
T > t_{i_1}^{v_1} > t_{i_2}^{v_2} > \cdots > t_{i_L}^{v_L} > 0
\end{array}\right.$

along the path: “good events” do not happen

$q > C\Delta$

for constant $C > 0 \quad \rightarrow \quad \#\text{rounds} = O(T + \log n)$ w.h.p.
The Metropolis Algorithm

Each vertex holds an independent rate-1 poisson clock.

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

When the clock at $v$ rings:

let $b = X_v$ and propose a random $c \in [q]$;
change $X_v$ to $c$ with prob. $f_{b,c}^v(X_{N(v)})$;

Metropolis filter:

$$f_{b,c}^v : [q]^{N(v)} \to [0,1]$$

$b \in [q]$: current color of $v$
$c \in [q]$: proposed color of $v$
2-Phase Paradigm

for each vertex $v \in V$:

Phase I:

- locally generate all update times $0 < t_1 < t_2 < \cdots < t_{M_v} < T$
  and proposed colors $c_1, c_2, \ldots, c_{M_v} \in [q]$;
- send the initial color and all $(t_i, c_i)_{1 \leq i \leq M_v}$ to all neighbors;

Phase II:

- For $i = 1, 2, \ldots, M_v$ do:
  once having received enough information:
  resolve the $i$-th update of $v$ and send the result
  ("Accept / Reject") to all neighbors;
to resolve the $i$-th update at $v$: $(t, c)$

- For $i = 1, 2, \ldots, M_v$ do:
  
  once having received enough information:
  
  resolve the $i$-th update of $v$ and send the result
  ("Accept / Reject") to all neighbors;

\[ S_u(t) : \text{set of possible colors of } u \text{ at time } t \]

\[ \forall \tau \in \bigotimes_{u \sim v} S_u(t) \]

\[ f^v_{b,c}(\tau) \text{ gives a biased coin} \]

Idea: Couple all these coins!
to resolve the $i$-th update at $v$: $(t, c)$

Construct $S_u(t)$ for every neighbor $u$ of $v$;

let $b$ be $v$'s current color and:

$$P_{\text{Acc}} \triangleq \min_{\tau \in \bigoplus_{u \sim v} S_u(t)} f_{b,c}(\tau);$$

$$P_{\text{Rej}} \triangleq 1 - \max_{\tau \in \bigoplus_{u \sim v} S_u(t)} f_{b,c}(\tau);$$

sample a uniform random $\beta \in [0,1]$;

upon $\beta \leq P_{\text{Acc}}$:

send “Accept!” to all neighbors and $i++$;

upon $\beta \geq 1 - P_{\text{Rej}}$:

send “Reject!” to all neighbors and $i++$;

upon receiving “Accept!” or “Reject!” from neighbor $u$:

update $S_u(t)$ accordingly and recalculate $P_{\text{Acc}}$ and $P_{\text{Rej}}$;
Universal Distributed Simulation of Metropolis Algorithm

Metropolis Algorithm: continuous-time $T$

let $b = X_v$ and propose a random $c \in [q]$;
change $X_v$ to $c$ with prob. $f^v_{b,c}(X_{N(v)})$;

Lipschitz condition: $\exists$ constant $C > 0$:

$$\forall (u, v) \in E, \forall a, a', b \in [q]: \quad \mathbb{E}_c[\delta_{u,a,a'} f^v_{b,c}] < \frac{C}{\Delta}$$

where $\delta_{u,a,a'} f^v_{b,c} \triangleq \max_{\sigma, \tau \text{ differ onl, at } u} |f^v_{b,c}(\sigma) - f^v_{b,c}(\tau)|$

$\#\text{rounds} = O(T + \log n)$ w.h.p.
<table>
<thead>
<tr>
<th>model</th>
<th>Lipschitz condition</th>
<th>Necessary condition for mixing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>q-coloring</strong></td>
<td>$\exists$ constant $C&gt;0$</td>
<td>$q \geq \Delta + 2$</td>
</tr>
<tr>
<td></td>
<td>$q &gt; C\Delta$</td>
<td></td>
</tr>
<tr>
<td><strong>Ising model with temperature $\beta$</strong></td>
<td>$\exists$ constant $C&gt;0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$1 - e^{-2</td>
<td>\beta</td>
</tr>
<tr>
<td><strong>hardcore model with fugacity $\lambda$</strong></td>
<td>$\exists$ constant $C&gt;0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda &lt; \frac{C}{\Delta}$</td>
<td>$\lambda &lt; \frac{(\Delta - 1)^{\Delta-1}}{\Delta - 2} \approx \frac{e}{\Delta - 2}$</td>
</tr>
</tbody>
</table>
Summary

• Universal distributed perfect simulation of Metropolis algorithms, with ideal parallelism under mild Lipschitz condition for Metropolis filter.

• **Open problem**: distributed simulation of general class of single-site Markov chains.
Outline

• Distributed Sampling Problem

• **Gibbs Distribution** (distribution defined by local constraints)

• Algorithmic Ideas [Feng, Hayes, Y., ’19]
  • *Local Metropolis Algorithm* [Feng, Sun, Y., PODC’17]
  • *LOCAL Jerrum-Valiant-Vazirani* [Feng, Y., PODC’18]
  • *Local Rejection Sampling* [Feng, Vishnoi, Y., STOC’19]

• Distributed Simulation of Metropolis
Local Computation

**Locally Checkable Labeling (LCL)** problems:

- CSPs with local constraints.
- Construct a feasible solution: vertex/edge coloring, Lovász local lemma
  - Find local optimum: MIS, MM
  - Approximate global optimum: maximum matching, minimum vertex cover, minimum dominating set

**Quest:** “Find a solution to the locally defined problem.”
“What can be sampled locally?”

- CSP with local constraints.
- Sample a uniform random solution.
- Distribution $\mu$ (over solutions) described by local rules.
  - uniform LCL solution
  - Ising model / RBM / tensor network…

**Quest:** “Generate a sample from the locally defined distribution.”
Markov Random Fields

- Each vertex corresponds to a variable with finite domain $[q]$.
- Each edge $(u,v) \in E$ imposes a binary constraint:
  \[ A_{u,v} : [q]^2 \to \{0,1\} \]
- Gibbs distribution $\mu$:
  \[ \forall \sigma \in [q]^V : \mu(\sigma) \propto \prod_{(u,v) \in E} A_{u,v}(\sigma_u, \sigma_v) \]
- Local conflict colorings:
  [Fraigniaud, Heinrich, Kosowski ’16]
Markov Random Fields

- **Gibbs distribution** $\mu$:
  \[
  \forall \sigma \in [q]^V : \quad \mu(\sigma) \propto \prod_{(u,v) \in E} A_{u,v}(\sigma_u, \sigma_v)
  \]

- **vertex $q$-coloring**:
  \[
  A_{u,v} = \begin{bmatrix}
  0 & 0 & 1 \\
  0 & 1 & \ddots \\
  1 & \ddots & 0 
  \end{bmatrix}
  \]

- **independent set**:
  \[
  A_{u,v} = \begin{bmatrix}
  1 & 1 \\
  1 & 0 
  \end{bmatrix}
  \]

- **local conflict colorings**:
  \[
  A_{u,v} \in \{0,1\}^{q \times q}
  \]
  [Fraigniaud, Heinrich, Kosowski ’16]
Markov Random Fields

- Each vertex corresponds to a **variable** with finite domain $[q]$.
- Each edge $(u,v) \in E$ imposes a binary constraint:
  \[
  A_{u,v} : [q]^2 \rightarrow [0,1]
  \]
  “soft” constraint
- Gibbs distribution $\mu$:
  \[
  \forall \sigma \in [q]^V : \quad \mu(\sigma) \propto \prod_{(u,v) \in E} A_{u,v}(\sigma_u, \sigma_v)
  \]
  \[\vec{X} \in [q]^V \text{ follows } \mu\]
- **local conflict colorings:**
  [Fraigniaud, Heinrich, Kosowski ’16]
Distributed Sampling

- **Instance**: a Gibbs distribution $\mu$
- **Output**: random $Y \in [q]^V$
  - approx. sampling:
    \[ d_{TV}(Y, \mu) \leq \epsilon \]
  - perfect sampling:
    \[ Y \sim \mu \]

Empirical studies in machine learning:

- [Kandasamy, et al, AISTAT’18]
- [Dasklakis, et al, NIPS’18]
- [De Sa, et al, ICML’16 best paper]
- [De Sa, et al, NIPS’15]
- [Ahmed, et al, WSDM’12]
- [Gonzalez, et al, AISTAT’11]
- [Yan, et al, NIPS’09]
- [Smyth, et al, NIPS’09]
- [Doshi-Velez, et al, NIPS’09]
- [Newman, et al, NIPS’08]
Distributed Sampling

- **Instance**: a Gibbs distribution $\mu$
- **Output**: random $Y \in [q]^V$
  - approx. sampling: $d_{TV}(Y, \mu) \leq \epsilon$
  - perfect sampling: $Y \sim \mu$

[Feng, Sun, Y. ’17]:

<table>
<thead>
<tr>
<th>Easy regime</th>
<th>Hard regime</th>
</tr>
</thead>
<tbody>
<tr>
<td>• $O(\Delta \log n)$-round is easy</td>
<td>• can be $\Omega(Diam)$-hard when $Diam = n^{\Omega(1)}$</td>
</tr>
<tr>
<td>• $O(\log n)$-round is possible</td>
<td></td>
</tr>
<tr>
<td>• $\Omega(\log n)$-round is necessary</td>
<td></td>
</tr>
</tbody>
</table>
Phase Transition

Correlation decay:
\[ \forall \sigma_B, \tau_B \in [q]^B : \]
\[ d_{TV}(\mu_v(\cdot | \sigma_B), \mu_v(\cdot | \tau_B)) \leq \exp(-\Omega(r)) \]

Hard regime: there is long-range correlation
- \((\Delta-1)\)-coloring on triangle-free graph
- independent set when \(\Delta=6\) or higher \(\Omega(Diam)\)-hard

Easy regime: various forms of correlation decays
- Dobrushin-Shlosman condition
- Uniqueness condition (spatial mixing)
- \ldots
Outline

• Distributed Sampling Problem
  • Gibbs Distribution (distribution defined by local constraints)

• Algorithmic Ideas
  • Local Metropolis Algorithm [Feng, Sun, Y., PODC’17]
  • LOCAL Jerrum-Valiant-Vazirani [Feng, Y., PODC’18]
  • Local Rejection Sampling [Feng, Vishnoi, Y., STOC’19]

• Distributed Simulation of Metropolis
Single-Site Markov Chain

Metropolis for \( q \)-coloring:

starting from an arbitrary \( X \in [q]^V \)

at each step:

- pick a \textbf{uniform random} vertex \( v \);
- propose \textbf{a random color} \( c \in [q] \);
- change \( X(v) \) to \( c \) if it’s proper;

Metropolis for \textbf{general MRF}:

- pick a \textbf{uniform random} vertex \( v \);
- propose to change \( X(v) \) to \textbf{a random color} \( c \in [q] \);
- accept the change \textbf{with probability} \( \min \left\{ 1, \frac{\mu(X')}{\mu(X)} \right\} = \min \left\{ 1, \prod_{u \in N(v)} \frac{A_{u,v}(X(u), c)}{A_{u,v}(X(u), X(v))} \right\} \)

[Bubley, Dyer, 97]: path-coupling works \textbf{mixing} in \( O(n \log n) \) steps
The **Local Metropolis Algorithm**

starting from an arbitrary $X \in [q]^V$, at each step:

- each vertex $v \in V$ independently proposes a random $c_v \in [q]$;
- each edge $(u,v) \in E$ passes its test independently with probability:
  
  $$A_{u,v}(X_u, c_v) \cdot A_{u,v}(c_u, X_v) \cdot A_{u,v}(c_u, c_v);$$

- each vertex $v \in V$ accepts to change to its proposed value $c_v$ if all incident edges pass their test;

• converge to the **correct** Gibbs distribution $\mu$.  

[Feng, Sun, Y. ’17]
The **Local Metropolis Algorithm**

**proposals:**

For \( q \)-coloring, at each step:

- Each vertex \( v \in V \) independently proposes a random color \( c_v \in [q] \).
- Each vertex \( v \in V \) accepts to change to its proposed color \( c_v \) if:
  \[
  X_u \neq c_v \wedge c_u \neq X_v \wedge c_u \neq c_v ;
  \]

[Feng, Sun, Y. ’17], [Fischer, Ghaffari ’18], [Feng, Hayes, Yin ’18]:

- Converges in \( O(\log n) \) rounds when:
  - Dobrushin-Shlosman condition
  - \((2+\delta)\Delta\)-coloring
LOCAL Jerrum-Valiant-Vazirani

[Jerrum, Valiant, Vazirani ’86]: (for self-reducible problems)

- approximate counting
- Poly-time TM
- perfect sampling

LOCAL JVV [Feng, Y. ’18]: (for self-reducible problems)

- correlation decay
- LOCAL approx. inference
- SLOCAL perfect sampling
- LOCAL perfect sampling
- unbounded msg/comput.
- local JVV reduction
- network decomposition
- O(log³ n) rounds

- (2+δ)Δ-coloring; 1.733Δ-coloring on triangle-free graph;
- Conjecture: (1+δ)Δ-coloring
Local Rejection Sampling

\[ \forall \sigma \in [q]^V : \quad \mu(\sigma) \propto \prod_{e=(u,v) \in E} A_u(\sigma_u, \sigma_v) \quad \text{where} \quad A_e : [q]^2 \to [0,1] \]

a Moser-Tardos style algorithm [Feng, Vishnoi, Y. ’19]:

- each \( v \in V \) ind. samples a random \( \sigma_v \in [q] \);
- each \( e=(u,v) \in E \) samples \( F_e \in \{0,1\} \) ind. with \( \Pr[F_e = 0] = A_e(\sigma_u,\sigma_v) \);
- while \( \exists e \in E \) s.t. \( F_e = 1 \) do:
  - resample \( \sigma_v \) for all \( v \in R \triangleq \bigcup_{e \in E : F_e = 1} e \);
  - for each \( e=(u,v) \in E \) that \( e \cap R \neq \emptyset \), resample \( F_e \in \{0,1\} \) ind. as:
    \[ \Pr[F_e = 0] = \begin{cases} A_e(\sigma_u, \sigma_v) & u, v \in R \quad \text{(internal edge)} \\ \frac{A_e(\sigma_u, \sigma_v)}{A_e(\sigma_u, \sigma_v^{\text{old}})} \min A_e(\sigma_u, \cdot) & u \notin R, v \in R \quad \text{(boundary edge)} \end{cases} \]
- each \( v \in V \) returns \( \sigma_v \);
Local Rejection Sampling

[Feng, Vishnoi, Y. ’19], [Feng, Guo, Y. ’19]

a Moser-Tardos style algorithm:

- perfect sampling, Las Vegas
- parallel/distributed (CONGEST)
- $O(\log n)$-round when converge
- works for dynamic input

- require stronger types of correlation decay:
  - $O(\Delta^2)$-coloring (for a variant of the algorithm)
<table>
<thead>
<tr>
<th>Features/Limitations</th>
<th>Fast regimes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local Metropolis</strong></td>
<td>• synchronous parallel Markov chain</td>
</tr>
<tr>
<td></td>
<td>• Monte Carlo sampling</td>
</tr>
<tr>
<td></td>
<td>• CONGEST model</td>
</tr>
<tr>
<td></td>
<td>• path-coupling works for sequential process</td>
</tr>
<tr>
<td></td>
<td>(Dobrushin-Shlosman cond.)</td>
</tr>
<tr>
<td></td>
<td>• ((2+\delta)\Delta)-coloring</td>
</tr>
<tr>
<td><strong>LOCAL JVV</strong></td>
<td>• perfect sampling</td>
</tr>
<tr>
<td></td>
<td>• abuses LOCAL model</td>
</tr>
<tr>
<td></td>
<td>• (O(\log^3 n)) rounds</td>
</tr>
<tr>
<td></td>
<td>• needs only necessary correlation decay</td>
</tr>
<tr>
<td></td>
<td>• conjecture:</td>
</tr>
<tr>
<td></td>
<td>((1+\delta)\Delta)-coloring</td>
</tr>
<tr>
<td><strong>Local Rejection Sampling</strong></td>
<td>• Moser-Tardos style</td>
</tr>
<tr>
<td></td>
<td>• Las Vegas, perfect sampling</td>
</tr>
<tr>
<td></td>
<td>• CONGEST model</td>
</tr>
<tr>
<td></td>
<td>• works on dynamic input</td>
</tr>
<tr>
<td></td>
<td>• requires faster correlation decay</td>
</tr>
<tr>
<td></td>
<td>• (O(\Delta^2))-coloring</td>
</tr>
<tr>
<td>Features/Limitations</td>
<td>Fast regimes</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>Universal Simulation of Metropolis</strong></td>
<td></td>
</tr>
<tr>
<td>• Monte Carlo sampling • CONGEST model</td>
<td>• as long as sequential Metropolis algorithm has $O(n \log n)$ mixing time</td>
</tr>
<tr>
<td><strong>LOCAL JVV</strong></td>
<td></td>
</tr>
<tr>
<td>• perfect sampling • abuses LOCAL model • $O(\log^3 n)$ rounds</td>
<td>• needs only necessary correlation decay • conjecture: $(1+\delta)\Delta$-coloring</td>
</tr>
<tr>
<td><strong>Local Rejection Sampling</strong></td>
<td></td>
</tr>
<tr>
<td>• Moser-Tardos style • Las Vegas, perfect sampling • CONGEST model • works on dynamic input</td>
<td>• requires faster correlation decay • $O(\Delta^2)$-coloring</td>
</tr>
</tbody>
</table>
Thank you!

Feng, Guo, Y. **Perfect sampling from spatial mixing.** arXiv:1907.06033.

Feng, Hayes, Y. **Distributed Metropolis Sampler with Optimal Parallelism.** arxiv:1904.00943

Feng, Hayes, Y. **Distributed Sampling Almost-Uniform Graph Coloring with Fewer Colors.** arxiv: 1802.06953.

Feng, Vishnoi, Y. **Dynamic Sampling from graphical models.** STOC’19. arxiv: 1807.06481.

Feng, Y. **On local distributed sampling and counting.** PODC’18. arxiv: 1802.06686.

Feng, Sun, Y. **What can be sampled locally?** PODC’17. arxiv: 1702.00142.