Towards derandomising Markov chain Monte Carlo

Weiming Feng
University of Edinburgh

Joint work with

Heng Guo
Edinburgh

Chunyang Wang
Nanjing

Jiaheng Wang
Edinburgh

Yitong Yin
Nanjing

USTC, Hefei, China
Feb 24th, 2023
Hypergraph independent set

Hypergraph \( H = (V, \mathcal{E}) \)
- \( k \)-uniform: each hyperedge contains \( k \) vertices
- \( \Delta \)-max degree: each vertex belongs to \( \leq \Delta \) hyperedges

**Independent set** \( S \subseteq V \) in hypergraph \( H \)
- For any \( e \in \mathcal{E}, e \not\subseteq S \)

**Example**: an independent set in 4-uniform hypergraph

**Example**: an independent set in graph \((k = 2)\)
Counting independent sets

**Input:** a hypergraph \( H = (V, \mathcal{E}) \)
- \( k \) uniform and max degree \( \Delta \) \( (k, \Delta = O(1)) \)
- number of vertices \( n \)

**Output:** the total number of independent sets in \( H \)

**Hardness of exact counting**

Counting independent sets is \#P complete
- Hardness result holds even if \( k = 2 \) and \( \Delta = 3 \) [Greenhill 2000]
Approximate counting

Approximate counting problem

**Input:** a hypergraph $H = (V, \mathcal{E})$
- $k$ uniform and max degree $\Delta$ ($k, \Delta = O(1)$)
- number of vertices $n$

an error bound $0 < \epsilon < 1$

**Output:** a number $\hat{Z}$ such that

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

$Z$: the total number of independent sets in $H$
Approximate counting algorithms

**FPTAS** (fully polynomial time approximation scheme)

*Deterministic* algorithm that solves the problem in time \( \text{poly} \left( n, \frac{1}{\epsilon} \right) \)

**FPRAS** (fully polynomial time randomised approximation scheme)

*Randomised* algorithm that *solves* the problem in time \( \text{poly} \left( n, \frac{1}{\epsilon} \right) \)

output a random number \( \hat{Z} \) such that

\[
\Pr\left[ (1 - \epsilon)Z \leq \hat{Z} \leq (1 + \epsilon)Z \right] \geq \frac{2}{3}
\]
Graph case \((k = 2)\): **computational phase transitions**

**Algorithm**

- **FPTAS** in time \(\left(\frac{n}{\varepsilon}\right)^{O(\log \Delta)}\) [Weitz05]
- **FPRAS** in time \(\tilde{O}\left(\frac{n^2}{\varepsilon^2}\right)\) [CLV21, ŠVV07]

\[\Delta \leq 5\]

**Complexity**

- The approximate counting is **NP-Hard**

\[\Delta \geq 6\]
## General hypergraphs \((k > 2)\)

| Work            | Regime             | Time                    |
|-----------------|--------------------|-------------------------|
| [BGGGŠ 16]      | \(k \geq \Delta \geq 200\) | \(\left(\frac{n}{\epsilon}\right)^{O(\log(k\Delta))}\) |
| [Moitra 19]     | \(k \gtrsim 60 \log \Delta\) | \(\left(\frac{n}{\epsilon}\right)^\text{poly}(k\Delta)\) |
| [JPV 21]        | \(k \gtrsim 7 \log \Delta\) | \(\left(\frac{n}{\epsilon}\right)^\text{poly}(k\Delta)\) |
| [HWY 22]        | \(k \gtrsim 5 \log \Delta\) | \(\left(\frac{n}{\epsilon}\right)^\text{poly}(k\Delta)\) |

**FPTAS** (deterministic algorithm)

**FPRAS** (randomised algorithm)

**Hardness:** the approximate counting is **NP-Hard** if \(k \leq 2 \log \Delta - C\) [BGGGŠ 16]

**Our result** [F., Guo, Wang, Wang, Yin, 22]: there is a **FPTAS** if \(k \gtrsim 2 \log \Delta\)
Theorem [this work] Let \( k \geq 2 \) and \( \Delta \geq 2 \) be two constants s.t.
\[
k \geq 2 \log \Delta + 4 \log k + O(1).
\]
There is a **deterministic algorithm** such that

- **Input**: a \( k \)-uniform hypergraph with \( n \) vertices and max degree \( \Delta \), an error bound \( \epsilon \)
- **Output**: an \( (1 \pm \epsilon) \)-approximation to the number of independent sets
- **Running time**: \( (n/\epsilon)^{O(\Delta^2 k^4)} \)

**Hardness**: NP-Hard if \( k \leq 2 \log \Delta - C \) [BGGGŠ 16]

**Linear hypergraph**: two hyperedges share at most 1 vertex

**Better condition**: Let \( \delta > 0 \) be a constant. Constants \( k \geq \frac{25(1+\delta)^2}{\delta^2} \) and \( \Delta \geq 2 \) satisfy
\[
k \geq (1 + \delta) \log \Delta + 3(1 + \delta) \log k + O(1)
\]
**Running time**: \( (n/\epsilon)^{\text{poly}(\Delta k/\delta)} \)

**Hardness for linear hypergraph**: NP-Hard if \( k \leq \log \Delta - C \) [QW 22]
| Work  | Regime       |
|-------|--------------|
| [JPV 21] | $k \gtrsim 7 \log \Delta$ |
| [HWY 22] | $k \gtrsim 5 \log \Delta$ |

**FPTAS (deterministic algorithm)**

Q: Why there is a gap between the regimes for FPTAS and FPRAS?

A: Previous FPTASes and FPRASes are based on very different techniques.

**Techniques for FPTAS:**
- Dynamic programming on computation tree [BGGGŠ 16]
- Linear programming [Moitra 19] [JPV 21]
- Derandomisation of a marginal recursive sampler [HWY 22, AJ 21]

**Techniques for FPRAS:**
- **MCMC sampling algorithm & reduction from counting to sampling** [HSZ 22, QWZ 22]
The sampling problem

• **Input:** a hypergraph $H = (V, E)$
  
  $\Omega = \{X \in \{0,1\}^V \mid X \text{ is an independent set in } H\}$

  $\mu$: the **uniform distribution** over $\Omega$

  \[
  \forall X \in \Omega, \quad \mu(X) = \frac{1}{Z} = \frac{1}{|\Omega|}
  \]

• **Output:** a random sample $X \sim \mu$.

The approximate sampling problem

• **Input:** a hypergraph $H = (V, E)$ specifying the uniform distribution $\mu$
  
  an error bound $\epsilon > 0$

• **Output:** a random sample $X \in \{0,1\}^V$ such that

  total variation distance $d_{TV}(X, \mu) \leq \epsilon$
Counting-to-sampling reduction

Fix the independent set $\emptyset$, denote it by $0$

$$\mu(0) = \frac{1}{Z}$$

approximate $Z$ \quad \leftrightarrow \quad approximate $\mu(0)$

Fix an arbitrary ordering of vertices $V = \{v(1), v(2), ..., v(n)\}$. By \textit{chain rule}

$$\mu(0) = \mu_{v(1)}(0) \times \mu_{v(2)}(0 | v(1) = 0) \times \cdots \times \mu_{v(n)}(0 | \forall j < n, v(j) = 0)$$

$$= \prod_{i=1}^{n} \mu_{v(i)}(0 | \forall j < i, v(j) = 0)$$

\textbf{Conditional Marginal distribution} $\mu_{v(i)}(0 | \forall j < i, v(j) = 0)$

given $X \sim \mu$, conditional on $X_{v(j)} = 0$ for all $j < i$, the prob of $X_{v(i)} = 0$
Counting-to-sampling reduction

approximate $Z$ $\leftrightarrow$ approximate $\mu(0)$ $\Rightarrow$ approximate $\mu_{v(i)}(0 \mid \forall j < i, v(j) = 0)$

with error $\epsilon/n$

Self-reduction [JVV86]

Conditional marginal distribution $\mu_{v(i)}(0 \mid \forall j < i, v(j) = 0)$

Marginal distribution $\pi_{v(i)}$

$\pi$: uniform distribution in another hypergraph
Counting-to-sampling reduction

approximate $Z$ \quad \leftrightarrow \quad \text{approximate } \mu(0) \quad \rightarrow \quad \text{approximate } \mu_v(0) \text{ with error } \epsilon/n

Solve approximate counting problem via sampling algorithm

The sampling algorithm $S(H, \epsilon)$
- Input: a hypergraph $H$ and error bound $\epsilon$
- Output: a random sample $X$ satisfying $d_{TV}(X, \mu) \leq \epsilon$

The algorithm for estimating $\mu_v(0)$
- Run $S\left(H, \frac{\epsilon}{2n}\right)$ independently $N = \text{poly}\left(\frac{n}{\epsilon}\right)$ times to get $X^{(1)}, X^{(2)}, \ldots, X^{(N)}$
- Compute the value $\hat{m} = \frac{\text{number of } i \text{ with } X_{v}^{(i)} = 0}{N}$
MCMC for hypergraph independent sets

Systematic scan for hypergraph independent sets

**Input:** a hypergraph $H = (V, E)$, each $v \in V$ has a unique label in \{0,1,2, ..., $n-1$\}

Start from an arbitrary independent set $X \in \{0,1\}^V$

For each $t$ from 1 to $T$

- Pick the vertex $v \in V$ with label$(v) = t \mod n$
- Update $X(v) \sim \mu_v(\cdot | X(V\backslash\{v\}))$

**Output** $X$

```
update $X_v \leftarrow 0$

update $X_v \leftarrow r$

$r \in \{0,1\}$ is a uniform random bit
```
MCMC for hypergraph independent sets

Systematic scan for hypergraph independent sets

**Input:** a hypergraph $H = (V, E)$, each $v \in V$ has a unique label in $\{0, 1, 2, ..., n - 1\}$

Start from an arbitrary independent set $X \in \{0, 1\}^V$

For each $t$ from 1 to $T$

- Pick the vertex $v \in V$ with label $t \mod n$
- Update $X(v) \sim \mu_v(\cdot | X(V \setminus \{v\}))$

**Output** $X$

Mixing time of systematic scan [HSZ16, JPV21, HSW21]

Systematic scan $(X_t)_{t=0}^T : X_t \in \{0, 1\}$ random independent set after the $t$-th update

$$k \geq 2 \log \Delta$$

$$d_{TV}(X_T, \mu) \leq \text{poly} \left( \frac{\epsilon}{n} \right),$$

where $T = O \left( n \log \frac{n}{\epsilon} \right)$
Approximate counting via MCMC algorithm

- approximate $Z$
- approximate $\mu(0)$
- approximate $\mu_v(0)$ with error $\epsilon/n$

Run $O\left(\frac{n \log \frac{n}{\epsilon}}{\epsilon}\right)$-step systematic scan to generate $N = \text{poly} \left(\frac{n}{\epsilon}\right)$ independent samples

Compute the fraction

\[ \hat{m} = \frac{\text{number of } i \text{ with } X_v^{(i)} = 0}{N} \]

Previous results [HSZ16, JPV21, HSW21]: there is a FPRAS if $k \geq 2 \log \Delta$
Our idea: Derandomising MCMC

systematic scan \((X_t)_{t=0}^{T}\)

output

\(X_T \in \{0,1\}^V\) s.t. \(d_{TV}(X_T, \mu) \leq \text{poly}\left(\frac{\epsilon}{n}\right)\)

use \(T = O\left(n \log \frac{n}{\epsilon}\right)\) random bits
Our idea: Derandomising MCMC

- Use $T = O \left(n \log \frac{n}{\epsilon}\right)$ random bits.
- Enumerate all possible choices of $O \left(\log \frac{n}{\epsilon}\right)$ random bits.
- Algorithm?

Deterministically approximate $\mu_v(0)$ with error $\text{poly} \left(\frac{\epsilon}{n}\right)$.

Enumerate all possible choices of $O \left(\log \frac{n}{\epsilon}\right)$ random bits.

- Output $X_T \in \{0,1\}^V$ s.t. $d_{TV}(X_T, \mu) \leq \text{poly} \left(\frac{\epsilon}{n}\right)$.
- Output $X_v \in \{0,1\}$ s.t. $d_{TV}(X_v, \mu_v) \leq \text{poly} \left(\frac{\epsilon}{n}\right)$. 

systematic scan $(X_t)^T_{t=0}$
Our results: log-time sampling via MCMC

**Theorem [this work]** Let constants $k \geq 2$, $\Delta \geq 2$ satisfying $k \geq 2 \log \Delta$

There is a sampling algorithm such that

**Input:** a $k$-uniform hypergraph max degree $\Delta$, a vertex $v$, an error bound $\epsilon$

**Output:** a random sample $X_v \in \{0,1\}$ with

$$d_{TV}(X_v, \mu_v) \leq \epsilon$$

**Running time & number of random bits used by alg.:** $\text{poly}(\Delta k) \log \frac{n}{\epsilon}$

**Straightforward derandomisation**

- The algorithm uses $\text{poly}(\Delta k) \log \frac{n}{\epsilon}$ random bits
- Enumerate all $\left(\frac{n}{\epsilon}\right)^{\text{poly}(\Delta k)}$ possible assignments for random bits
- Deterministically compute $\Pr[X_v = 0] \in (1 \pm 2\epsilon)\mu_v(0)$ (as $\mu_v(0) \geq 1/2$)
Our results: log-time sampling via MCMC

**Theorem [this work]** Let constants $k \geq 2$, $\Delta \geq 2$ satisfying $k \gtrsim 2 \log \Delta$

There is a sampling algorithm such that

**Input:** a $k$-uniform hypergraph max degree $\Delta$, a vertex $v$, an error bound $\epsilon$

**Output:** a random sample $X_v \in \{0,1\}$ with

$$d_{TV}(X_v, \mu_v) \leq \epsilon$$

**Running time & number of random bits used by alg.:** $\text{poly}(\Delta k) \log \frac{n}{\epsilon}$

**Result on linear hypergraphs [this work]**

- Let constants $k \geq 2$, $\Delta \geq 2$, $\delta > 0$ satisfying $k \gtrsim (1 + \delta) \log \Delta$ and $k \geq k_0(\delta)$

- Running time & number of random bits used by alg.

$$\text{poly} \left( \frac{\Delta k}{\delta} \right) \log \frac{n}{\epsilon}$$
Systematic scan for hypergraph independent sets

Start from an arbitrary independent set \( X \in \{0,1\}^V \)

For each \( t \) from 1 to \( T \)
- Pick the vertex \( v \in V \) with label \( t \mod n \)
- Update \( X(v) \sim \mu_v (\cdot | X(V\backslash\{v\})) \)

Output \( X \)

**Blocked:** \( X_v \) is updated to 0

\( \exists e \ni v \) s.t. \( \forall u \in e\backslash\{v\}, X_u = 1 \)

**Unblocked:** \( X_v \) is updated to 0 w.p. \( 1/2 \)

\( \forall e \ni v, \exists u \in e\backslash\{v\} \) s.t. \( X_u = 0 \)

\( r \in \{0,1\} \) is a random bit
Systematic scan for hypergraph independent sets

Start from an arbitrary independent set $X \in \{0,1\}^V$
For each $t$ from 1 to $T$
• Pick the vertex $v \in V$ with label $t \mod n$
• Update $X(v) \sim \mu_v (\cdot | X(V\{v}))$
Output $X$

The $t$-th transition step of systematic scan

Sample a random bit $r_t \in \{0,1\}$ uniformly at random;
If $X$ is in the **blocked** case ($\forall e$ with $v \in e$, $\exists u \in e\{v\}$ s.t. $X_u = 0$)
\[ X_v \leftarrow 0; \]
If $X$ is in the **unblocked** case ($\exists e$ with $v \in e$, $\forall u \in e\{v\}$ s.t. $X_u = 1$)
\[ X_v \leftarrow r_t \]

In both cases
\[ r_t = 0 \quad \text{in both cases} \quad X_v \leftarrow 0 \]
Systematic scan for hypergraph independent sets

Start from an arbitrary independent set $X \in \{0,1\}^V$

For each $t$ from 1 to $T$

- Pick the vertex $v \in V$ with label $t \mod n$
- Update $X(v) \sim \mu_v (\cdot | X(V \setminus \{v\}))$

Output $X$

The $t$-th transition step of systematic scan

Sample a random bit $r_t \in \{0,1\}$ uniformly at random;

If $X$ is in the **blocked** case ($\forall e$ with $v \in e$, $\exists u \in e \setminus \{v\}$ s.t. $X_u = 0$)

\[ X_v \leftarrow 0; \]

If $X$ is in the **unblocked** case ($\exists e$ with $v \in e$, $\forall u \in e \setminus \{v\}$ s.t. $X_u = 1$)

\[ X_v \leftarrow r_t \]

If $r_t = 0$ (with probability $1/2$)

decide $X_v$ **immediately** (**no need** to distinguish blocked or unblocked cases)
systematic scan \((X_t)_{t=0}^T\)  

our goal  

output \(X_T(v) \in \{0,1\}\)

- For any vertex \(v \in V\), any time \(0 \leq t \leq T\),
  \[S(v, t) = \{1 \leq j \leq t \mid \text{vertex } v \text{ is picked in } j^{th} \text{ step, i.e.} \label(v) = j \mod n\}\]

- **Previous** update time for \(v \in V\) up to time \(t\):
  \[\text{Pred}(v, t) = \begin{cases} 
\max \{j \mid j \in S(v, t)\} & \text{if } S(v, t) \neq \emptyset \\
0 & \text{if } S(v, t) = \emptyset 
\end{cases}\]

**Our goal:** output the value of \(X_T(v) = X_{\text{Pred}(v,T)}(v) \in \{0,1\}\)

compute the value of \(v\) after the last time that \(v\) is updated
Resolve($v, t$)

- **Input:** $v \in V$ and $1 \leq t \leq T$ such that $v$ is picked at the time $t$
- **Output:** the random value $X_t(v) \in \{0, 1\}$

Resolve($v, t$)

- **Input:** $v \in V$ and $1 \leq t \leq T$ such that $v$ is picked at the time $t$
- **Output:** the random value $X_t(v) \in \{0, 1\}$

Random bits for simulating Markov chain up to time $T$

reveal the random bit $r_t$;

if $r_t = 0$, return 0;

else  //distinguish blocked or unblocked cases

reveal $r_{\text{pred}(w,t)}$ for all neighbours $w$;

$t = 19$
Resolve\((v, t)\)

- **Input:** \(v \in V\) and \(1 \leq t \leq T\) such that \(v\) is picked at the time \(t\)
- **Output:** the random value \(X_t(v) \in \{0, 1\}\)

Random bits for simulating Markov chain up to time \(T\):

\[
\begin{array}{cccccccccccccccc}
    r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_8 & r_9 & r_{10} & r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} & r_{17} & r_{18} & r_{19} & r_{T=20} \\
\end{array}
\]

reveal the random bit \(r_t\);
if \(r_t = 0\), return 0;
else //distinguish blocked or unblocked cases
    reveal \(r_{\text{pred}(w, t)}\) for all neighbours \(w\);
    for each hyperedge \(e\) incident to \(v\) do
Resolve(v, t)

- **Input:** v ∈ V and 1 ≤ t ≤ T such that v is picked at the time t
- **Output:** the random value X_t(v) ∈ {0,1}

| r₁ | r₂ | r₃ | r₄ | r₅ | r₆ | r₇ | r₈ | r₉ | r₁₀ | r₁₁ | r₁₂ | r₁₃ | r₁₄ | r₁₅ | r₁₆ | r₁₇ | r₁₈ | r₁₉ | r_T=20 |
|-----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

reveal the random bit r_t;

if r_t = 0, return 0;

else  //distinguish blocked or unblocked cases

reveal r_{pred(w,t)} for all neighbours w;

for each hyperedge e incident to v do

∃w ∈ e\{v} s.t. r_{pred(w,t)} = 0 ⇒ X_t(w) = 0  good edge
Resolve\((v, t)\)

- **Input:** \(v \in V\) and \(1 \leq t \leq T\) such that \(v\) is picked at the time \(t\)
- **Output:** the random value \(X_t(v) \in \{0, 1\}\)

Random bits for simulating Markov chain up to time \(T\)

\[
\begin{array}{ccccccccccccc}
\text{r}_1 & \text{r}_2 & \text{r}_3 & \text{r}_4 & \text{r}_5 & \text{r}_6 & \text{r}_7 & \text{r}_8 & \text{r}_9 & \text{r}_{10} & \text{r}_{11} & \text{r}_{12} & \text{r}_{13} & \text{r}_{14} & \text{r}_{15} & \text{r}_{16} & \text{r}_{17} & \text{r}_{18} & \text{r}_{19} & \text{r}_{T=20} \\
\end{array}
\]

reveal the random bit \(r_t\);
if \(r_t = 0\), **return** 0;
else \ //distinguish blocked or unblocked cases

reveal \(r_{\text{pred}(w,t)}\) for all neighbours \(w\);
for each hyperedge \(e\) incident to \(v\) do

\[
\begin{align*}
\exists w \in e \setminus \{v\} \text{ s.t. } r_{\text{pred}(w,t)} = 0 & \implies X_t(w) = 0 \quad \text{good edge} \\
\forall w \in e \setminus \{v\}, r_{\text{pred}(w,t)} = 1 & \quad \text{bad edge (cannot decide } X_t(w))
\end{align*}
\]
**Resolve**($v, t$)

- **Input:** $v \in V$ and $1 \leq t \leq T$ such that $v$ is picked at the time $t$
- **Output:** the random value $X_t(v) \in \{0, 1\}$

reveal the random bit $r_t$;

if $r_t = 0$, return 0;

else //distinguish blocked or unlocked cases
  reveal $r_{\text{pred}(w, t)}$ for *all neighbours* $w$;
  for each hyperedge $e$ incident to $v$ do
    • if $\forall w \in e \setminus \{v\}, r_{\text{pred}(w, t)} = 1$ //bad edge
      • if all $w \in e \setminus \{v\}$, $\text{Resolve}(w, \text{pred}(w, t)) = 1$
        return 0; //blocked case

return 1; //unlocked case
reveal the random bit \( r_t \);
if \( r_t = 0 \), return 0;
else //distinguish blocked or unblocked cases
  reveal \( r_{\text{pred}(w,t)} \) for all neighbours \( w \);
  for each hyperedge \( e \) incident to \( v \) do
    • if \( \forall w \in e \setminus \{v\}, r_{\text{pred}(w,t)} = 1 \) //bad edge
      • If all \( w \in e \setminus \{v\}, \text{Resolve}(w, \text{pred}(w, t)) = 1 \)
        return 0; //blocked case
  return 1; //unlocked case

An informal analysis of the branching process

• Vertex \( v \) has \( \leq \Delta \) incident hyperedges and each hyperedge has \( k \) vertices

  \[ \mathbb{E}[\text{#recursive calls}] \leq \Delta k \left( \frac{1}{2} \right)^{k-1} < 1 \Rightarrow k \gtrsim \log \Delta \]

• However, hyperedges share vertices \( \xrightarrow{\text{dependency}} \) of recursive calls

  \( k \gtrsim 2 \log \Delta \) \( \xrightarrow{\text{w.h.p.}} \) \( \#{\text{resolve instances}}= O_{\Delta,k}(\log n) \)

pay extra factor to overcome the dependency
reveal the random bit \( r_t \);

if \( r_t = 0 \), return 0;

else  //distinguish blocked or unblocked cases

reveal \( r_{\text{pred}(w,t)} \) for all neighbours \( w \);

for each hyperedge \( e \) incident to \( v \) do

• if \( \forall w \in e \setminus \{v\}, r_{\text{pred}(w,t)} = 1 \)  //bad edge

• if all \( w \in e \setminus \{v\}, \text{Resolve}(w, \text{pred}(w, t)) = 1 \)

return 0;  //blocked case

return 1;  //unlocked case

Better bound for linear hypergraphs (informal analysis)

• Linear hypergraph: two hyperedges share at most 1 vertex

\[ k \geq (1 + \delta) \log \Delta \]

\[ \text{w.h.p. } \#\{\text{resolve instances}\} = O_{\delta,\Delta,k}(\log n) \]

pay extra \( \delta \)-factor to overcome the dependency, where \( \delta > 0 \) is an arbitrary constant
Hypergraph colouring

Instance
• a $k$-uniform hypergraph with max degree $\Delta$
• colour set $[q] = \{1, 2, \ldots, q\}$

Hypergraph colouring: $X \in [q]^V$ s.t.
• no hyperedge is monochromatic
  $$\forall e \in E, \ |\{X_v : v \in e\}| \geq 2$$

Lovász Local lemma and algorithmic LLL
• find a hypergraph colouring when $q \geq \Delta^{1/k}$ ($q \geq C_k \Delta^{1/(k-1)}$)

Sampling Lovász Local lemma
• **sampling / approx. counting** hypergraph colourings in the local lemma regime
### Previous results for approximate counting

| Work                        | Regime                                           | Algorithm Type | Type  |
|-----------------------------|--------------------------------------------------|----------------|-------|
| [Frieze, Anastos, 17]      | Linear & $q \geq \max\{\log n, \Delta^{1/k}\}$ | Randomised     |       |
| [Guo, Liao, Lu, Zhang, 19] | $q \geq \Delta^{16/k}$                          | Deterministic  |       |
| [Jain, Pham, Vuong, 21]    | $q \geq \Delta^{7/k}$                           |                |       |
| [Feng, Guo, Wang, 22]      | $q \geq \Delta^{(1+\epsilon)/k}$                |                |       |
## Previous results for approximate counting

| Work                                      | Regime                      | Algorithm Type |
|-------------------------------------------|-----------------------------|----------------|
| [Frieze, Anastos, 17]                    | Linear & $q \geq \max\{\log n, \Delta^{1/k}\}$ | Randomised     |
| [Guo, Liao, Lu, Zhang, 19]               | $q \geq \Delta^{16/k}$     | Deterministic  |
| [Jain, Pham, Vuong, 21]                  | $q \geq \Delta^{7/k}$      |                |
| [Jain, Pham, Vuong, 21]                  | $q \geq \Delta^{7/k}$      |                |
| [Feng, He, Yin, 21]                      | $q \geq \Delta^{9/k}$      |                |
| [Feng, Guo, Wang, 22]                    | Linear & $q \geq \Delta^{(2+\delta)/k}$ |                |

**Lower Bound** [Galanis, Guo, Wang, 21]

- NP-Hard for $q \lesssim \Delta^{2/k}$ (general hypergraph) and $q \lesssim \Delta^{1/k}$ (linear hypergraph)

**MCMC on projected distribution**
Our results for approximate counting

| Work                                      | Regime                                      | Algorithm Type |
|-------------------------------------------|---------------------------------------------|----------------|
| [Frieze, Anastos, 17]                    | Linear & $q \geq \max\{\log n, \Delta^{1/k}\}$ | Randomised     |
| [Guo, Liao, Lu, Zhang, 19]               | $q \geq \Delta^{16/k}$                     | Deterministic  |
| [Jain, Pham, Vuong, 21]                  | $q \geq \Delta^{7/k}$                      |                |
| [F., Guo, Wang, Wang, Yin, 22]           | $q \geq \Delta^{3/k}$                      | Deterministic  |
|                                          | Linear & $q \geq \Delta^{(2+\delta)/k}$   |                |

**Lower Bound [Galanis, Guo, Wang, 21]**

- NP-Hard for $q \leq \Delta^{2/k}$ (general hypergraph) and $q \leq \Delta^{1/k}$ (linear hypergraph)
Counting-to-sampling reduction

**Instance:** hypergraph $H = (V, \mathcal{E})$ and colour set $[q]$

**Colouring:** $\Omega \subseteq [q]^V$ set of all proper colourings

$$Z = |\Omega|$$

**Distribution:** $\mu$ the uniform distribution over all proper colourings

$$\forall X \in \Omega, \quad \mu(X) = \frac{1}{Z}$$

**Approximate counting to sampling reduction**

approximate $Z$ $\rightarrow$ approximate $\mu_e$ within TV distance $\text{poly}\left(\frac{e}{\Delta n}\right)$

$e \in \mathcal{E}$ is a hyperedge

$\mu_e$ over $[q]^e$: Marginal distribution on $k$ vertices
MCMC on projected distribution

- The systematic scan on $\mu$ does not work (connectivity issue)
- Use systematic scan on projected distribution

\[ Y = \{h(X_v)\}_{v \in V} \sim \pi \text{ if } X \sim \mu \]

Balanced projection scheme

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

for any $j \in [s]$, $|h^{-1}(j)| \in \frac{q}{s} \pm 1$

the set of $q$ colours

Bucket #1

Bucket #2

Bucket # s

the set of $s$ buckets, $s = q^c$ for $c < 1$

Projected distribution $\pi$ over $[s]^V$ [Feng, He, Yin, 21]
Systematic scan for projected distribution

Start from a uniform random $Y \in [s]^V$;
For each $t$ from 1 to $T$
- Pick the vertex $v \in V$ with label $t \mod n$
- Update $Y(v) \sim \pi_v(\cdot | Y(V \setminus \{v\}))$ conditional marginal distribution induced by $\pi$

Output $Y$

Local uniformity in the local lemma regime

For any $v \in V$, any condition $\sigma \in [q]^{V \setminus \{v\}}$

$$\forall j \in [s], \quad \mu_v(j | \sigma) \in \left(1 \pm \frac{1}{s}\right) \frac{|h^{-1}(j)|}{q} \approx \left(1 \pm \frac{1}{s}\right) \frac{1}{s}$$

given any condition, the marginal on $v$ is close to a uniform distribution

Mixing in the local lemma regime: If $T = O \left(n \log \frac{n}{\epsilon} \right)$ then $d_{TV}(\pi, X_T) \leq \frac{\epsilon}{n^2}$
Log-time sampling (informal)

Resolve($v, t$)

- **Input:** $v \in V$ and $1 \leq t \leq T$ such that $v$ is picked at the time $t$
- **Output:** the random value $Y_t(v) \in [s]$

Local uniformity $\implies$ Marginal lower bound

For any $v \in V$, any condition $\sigma \in [q] \setminus \{v\}$

$$\forall j \in [s], \quad \mu_v(j \mid \sigma) \geq \left(1 - \frac{1}{s}\right) \frac{1}{s}$$

By marginal lower bound, even if $Y_t(V \setminus \{v\})$ is unknown, we can decide $Y_t(v)$ w.p.

$$p_{LB} \approx \sum_{j \in [s]} \left(1 - \frac{1}{s}\right) \frac{1}{s} = 1 - \frac{1}{s} \approx 1 - \Delta^{-\Omega(1/k)}$$
Resolve($v, t$) (informal description)

- Reveal the randomness used in $t$-th step
- **With probability** $p_{LB} = 1 - 1/s$
  - Determine the value of $X_t(v)$ and **return**.
- **With probability** $1 - p_{LB}$ **get enough information** to determine $\mu_v(\cdot | Y_t(V \setminus \{v\}))$
  - Reveal other randomness & call resolve recursively if necessary

How to sample from $\mu_e$ using a partial sample from projected distribution $\pi$?

1. **Sample** $Y_M$ for some $M \subseteq V$
2. **Sample** $X_e$ conditional on $Y_M$

How to sample $Y_M$?

1. **Generalise** $\text{Resolve}(v, t)$ returns $X_t(v) \in [s]$
2. $\text{Resolve}(M, t)$ returns $X_t(M) \in [s]^M$
Summary

Fully Poly-time \textit{deterministic} approximate counting algorithms:

- Hypergraph independent sets
  - general case $k \gtrapprox 2 \log \Delta$
  - linear case $k \gtrapprox (1 + \delta) \log \Delta$
  - almost match the \textbf{hardness conditions}

- Hypergraph colourings
  - general case $q \gtrapprox \Delta^{3/k}$
  - linear case $q \gtrapprox \Delta^{(2+\delta)/k}$
  - match the conditions of \textbf{current best randomised algorithms}

Technique

- MCMC
- Log-time sampling algorithm for marginal distributions
- FPTAS for approx. counting
Open problems

Close the gap for hypergraph colouring
• general case $q \gtrsim \Delta^{3/k}$ v.s. $q \lesssim \Delta^{2/k}$
• linear case $q \gtrsim \Delta^{(2+\delta)/k}$ v.s. $q \lesssim \Delta^{1/k}$

Faster algorithm for deterministic approximate counting
• FPRAS running time $\tilde{O}(n^2/\epsilon^2)$
• FPTAS running time $n^{\text{poly}(k\Delta)}$
• Question: $f(k\Delta)n^C$ running time (can we use pseudorandom generator?)

Sublinear time sampling (related to local access to huge random objects [BRY ITCS2020])
• Input: distribution $\mu$ over $[q]^V$ and $v \in V$
• Output: a sample or an approximate sample from $\mu_v$
• For which $\mu$, it can be solved in sublinear time (say $n^{1-\epsilon}$ time or even $O(\log n)$ time)