Sampling Lovász local lemma for general constraint satisfaction solutions in near-linear time

Chunyang Wang

Nanjing University

Joint work with Kun He(CAS) and Yitong Yin(Nanjing University)

Constraint Satisfaction Problem

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

Constraint Satisfaction Problem

- $\Phi = (V, \mathcal{Q}, \mathcal{C})$
- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ with finite domains Q_v for each $v \in V$

Constraint Satisfaction Problem

- $\Phi = (V, \mathcal{Q}, \mathcal{C})$
- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ with finite domains Q_v for each $v \in V$
- (local) **Constraints**: $\mathcal{C} = C_1, C_2, \ldots, C_m$

Constraint Satisfaction Problem

- $\Phi = (V, \mathcal{Q}, \mathcal{C})$
- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ with finite domains Q_v for each $v \in V$
- (local) **Constraints**: $\mathcal{C} = C_1, C_2, \ldots, C_m$
  - each $c \in \mathcal{C}$ is defined on a subset $\mathsf{vbl}(c)$ of variables

$$c : \bigotimes_{v \in \mathsf{vbl}(c)} Q_v \to \{\mathtt{True}, \mathtt{False}\}$$

Constraint Satisfaction Problem

- $\Phi = (V, \mathcal{Q}, \mathcal{C})$
- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ with finite domains Q_v for each $v \in V$
- (local) **Constraints**: $\mathcal{C} = C_1, C_2, \ldots, C_m$
  - each $c \in \mathcal{C}$ is defined on a subset $\mathsf{vbl}(c)$ of variables

$$c : \bigotimes_{v \in \mathsf{vbl}(c)} Q_v \to \{\texttt{True}, \texttt{False}\}$$

- **CSP formula**: $\forall \boldsymbol{x} \in \mathcal{Q} = \bigotimes_{v \in V} Q_v$

$$\Phi(\boldsymbol{x}) = \bigwedge_{c \in \mathcal{C}} c(\boldsymbol{x}_{\mathsf{vbl}(c)})$$

Constraint Satisfaction Problem

- $\Phi = (V, \mathcal{Q}, \mathcal{C})$
- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ with finite domains Q_v for each $v \in V$
- (local) **Constraints**: $\mathcal{C} = C_1, C_2, \ldots, C_m$
  - each $c \in \mathcal{C}$ is defined on a subset $\mathsf{vbl}(c)$ of variables

$$c : \bigotimes_{v \in \mathsf{vbl}(c)} Q_v \to \{\texttt{True}, \texttt{False}\}$$

- **CSP formula**: $\forall \boldsymbol{x} \in \mathcal{Q} = \bigotimes_{v \in V} Q_v$

$$\Phi(\boldsymbol{x}) = \bigwedge_{c \in \mathcal{C}} c(\boldsymbol{x}_{\mathsf{vbl}(c)})$$

- Example(k-SAT): Boolean variables $V = \{x_1, x_2, x_3, x_4, x_5\}$

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

Lovász Local Lemma

- Variables take independent random values $X_1, X_2, \ldots, X_n$

Lovász Local Lemma

- Variables take independent random values $X_1, X_2, \ldots, X_n$
- **Violation probability**: each $c \in \mathcal{C}$ is violated with probability $\leq p$

Lovász Local Lemma

- Variables take independent random values $X_1, X_2, \ldots, X_n$
- **Violation probability**: each $c \in \mathcal{C}$ is violated with probability $\leq p$
- **Constraint Degree**: each $c \in \mathcal{C}$ shares variable with $\leq \Delta$ constraints $c' \in \mathcal{C}$(including c itself), i.e., $\mathrm{vbl}\,(c) \cap \mathrm{vbl}\,(c') \neq \emptyset$

Lovász Local Lemma

- Variables take independent random values $X_1, X_2, \ldots, X_n$
- **Violation probability**: each $c \in \mathcal{C}$ is violated with probability $\leq p$
- **Constraint Degree**: each $c \in \mathcal{C}$ shares variable with $\leq \Delta$ constraints $c' \in \mathcal{C}$(including c itself), i.e., $\mathrm{vbl}(c) \cap \mathrm{vbl}(c') \neq \emptyset$
- **LLL**[EL75]:

$$\mathrm{e}p\Delta \leq 1 \implies \text{ solution exists}$$

Lovász Local Lemma

- Variables take independent random values $X_1, X_2, \ldots, X_n$
- **Violation probability**: each $c \in \mathcal{C}$ is violated with probability $\leq p$
- **Constraint Degree**: each $c \in \mathcal{C}$ shares variable with $\leq \Delta$ constraints $c' \in \mathcal{C}$(including c itself), i.e., $\mathsf{vbl}(c) \cap \mathsf{vbl}(c') \neq \emptyset$
- **LLL**[EL75]:

$$\mathrm{e}p\Delta \leq 1 \implies \text{solution exists}$$

- **Constructive(Algorithmic) LLL**[MT10]:

$$\mathrm{e}p\Delta \leq 1 \implies \text{solution can be found very efficiently}$$

Sampling & Counting LLL

Sampling & Counting LLL

Input: a CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$
Output:

- (sampling): uniform random satisfying solution
- (counting): number of satisfying solutions

Sampling & Counting LLL

Sampling & Counting LLL

Input: a CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$

Output:

- (sampling): uniform random satisfying solution
- (counting): number of satisfying solutions

- μ: uniform distribution over all satisfying solutions of Φ

Sampling & Counting LLL

Sampling & Counting LLL

Input: a CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$
Output:

- (sampling): uniform random satisfying solution
- (counting): number of satisfying solutions

- μ: uniform distribution over all satisfying solutions of Φ

Rejection Sampling

generate a uniform random $\boldsymbol{x} \in \mathcal{Q}$;
if $\Phi(\boldsymbol{x}) = \texttt{True}$ then accept else reject;
μ is the distribution of $(\boldsymbol{x} \mid \text{accept})$

Sampling & Counting LLL

Sampling & Counting LLL

Input: a CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$
Output:

- (sampling): uniform random satisfying solution
- (counting): number of satisfying solutions

- μ: uniform distribution over all satisfying solutions of Φ

Rejection Sampling

generate a uniform random $\boldsymbol{x} \in \mathcal{Q}$;
if $\Phi(\boldsymbol{x}) = \texttt{True}$ then accept else reject;
μ is the distribution of $(\boldsymbol{x} \mid \text{accept})$

- Satisfying solutions may be exponentially rare!

Sampling & Counting LLL

- Exact counting is #P-hard

Sampling & Counting LLL

- Exact counting is #P-hard
- 

Sampling & Counting LLL

- Exact counting is #P-hard
- 

- Application: inference in probabilistic graphical models

Gibbs distribution $\quad \mu(\boldsymbol{x}) \propto \Phi(\boldsymbol{x}) = \prod_{c \in \mathcal{C}} c(\boldsymbol{x}_{\mathrm{vbl}(c)})$

$$\text{where each } c : \bigotimes_{i \in \mathrm{vbl}(c)} Q_i \to \mathbb{R}_{\geq 0}$$

Sampling & Counting LLL

- Exact counting is $\#$P-hard
- 

- Application: inference in probabilistic graphical models

Gibbs distribution $\quad \mu(\boldsymbol{x}) \propto \Phi(\boldsymbol{x}) = \prod_{c \in \mathcal{C}} c(\boldsymbol{x}_{\mathsf{vbl}(c)})$

$$\text{where each } c : \bigotimes_{i \in \mathsf{vbl}(c)} Q_i \to \mathbb{R}_{\geq 0}$$

- Inference: $\mathop{\mathbf{Pr}}_{\boldsymbol{X} \sim \mu} [X_i = \cdot \mid \boldsymbol{X}_S = \boldsymbol{x}_S]$

Sampling & Counting LLL

- Exact counting is #P-hard
- 

- Application: inference in probabilistic graphical models

$$\text{Gibbs distribution} \quad \mu(\boldsymbol{x}) \propto \Phi(\boldsymbol{x}) = \prod_{c \in \mathcal{C}} c(\boldsymbol{x}_{\mathrm{vbl}(c)})$$

$$\text{where each } c : \bigotimes_{i \in \mathrm{vbl}(c)} Q_i \to \mathbb{R}_{\geq 0}$$

- Inference: $\mathop{\mathbf{Pr}}_{\boldsymbol{X} \sim \mu}[X_i = \cdot \mid \boldsymbol{X}_S = \boldsymbol{x}_S]$
- Sampling almost uniform constraint satisfcation solutions under LLL-like condition?

Sampling & Counting LLL

- Exact counting is #P-hard
- 

- Application: inference in probabilistic graphical models

$$\text{Gibbs distribution} \quad \mu(\boldsymbol{x}) \propto \Phi(\boldsymbol{x}) = \prod_{c \in \mathcal{C}} c(\boldsymbol{x}_{\mathsf{vbl}(c)})$$

$$\text{where each } c : \bigotimes_{i \in \mathsf{vbl}(c)} Q_i \to \mathbb{R}_{\geq 0}$$

- Inference: $\Pr_{\boldsymbol{X} \sim \mu} [X_i = \cdot \mid \boldsymbol{X}_S = \boldsymbol{x}_S]$
- Sampling almost uniform constraint satisfcation solutions under LLL-like condition?

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k, \Delta) \cdot n \log n$	MCMC

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta) \cdot n \log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot n$	PRS

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta) \cdot n \log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta) \cdot n \log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta) \cdot n \log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta) \cdot n \log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[FGYZ21]	CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot \tilde{O}(n^{1.001})$	Mark/unmark+MCMC

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta) \cdot n \log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[FGYZ21]	CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k,\Delta) \cdot \tilde{O}(n^{1.001})$	Mark/unmark+MCMC
[FHY21]	atomic CSP[iv]	$p\Delta^{350} \lesssim 1$	$\mathrm{poly}(q,k,\Delta) \cdot \tilde{O}(n^{1.001})$	Entropy Compression+MCMC

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta)\cdot n\log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[FGYZ21]	CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot \tilde{O}(n^{1.001})$	Mark/unmark+MCMC
[FHY21]	atomic CSP[iv]	$p\Delta^{350} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot \tilde{O}(n^{1.001})$	Entropy Compression+MCMC
[JPV21b]	general CSP	$p\Delta^7 \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta)\cdot n\log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[FGYZ21]	CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot \tilde{O}(n^{1.001})$	Mark/unmark+MCMC
[FHY21]	atomic CSP[iv]	$p\Delta^{350} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot \tilde{O}(n^{1.001})$	Entropy Compression+MCMC
[JPV21b]	general CSP	$p\Delta^{7} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[JPV21a]	atomic CSP	$p\Delta^{7.043} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot \tilde{O}(n^{1.001})$	Entropy Compression+MCMC

Sampling & Counting LLL

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF[i]	$p\Delta^2 \lesssim 1$[ii]	$\mathrm{poly}(k,\Delta)\cdot n\log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log \Delta k, k/2)$[iii] $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[FGYZ21]	CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot \tilde{O}(n^{1.001})$	Mark/unmark+MCMC
[FHY21]	atomic CSP[iv]	$p\Delta^{350} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot \tilde{O}(n^{1.001})$	Entropy Compression+MCMC
[JPV21b]	general CSP	$p\Delta^{7} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[JPV21a]	atomic CSP	$p\Delta^{7.043} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot \tilde{O}(n^{1.001})$	Entropy Compression+MCMC
[HSW21]	atomic CSP	$p\Delta^{5.714} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot n\log n$ expected	State tensorization+MCMC

[i] Monotone CNF: all variables appear positively, e.g. $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_3 \vee x_4 \vee x_5)$

[ii] $\lesssim$ hides lower order items, e.g., k, q.

[iii] s: two independent clauses share **at least** s variables

[iv] atomic means each constraint of the CSP has exactly one forbidden configuration

Sampling & Counting LLL

- It can be seen there are mainly two lines of work, taking different methods:
  - The line of work [Moi19, GLLZ19, JPV21b] applies the coupling and linear programming method, initiated by Moitra. This method is deterministic and made to work for general CSP instances in [JPV21b]. However, this method suffers from a $n^{\mathrm{poly}(k,\Delta,\log q)}$ running time, which is exponential if $k, \Delta = \Omega(1)$.

Sampling & Counting LLL

- It can be seen there are mainly two lines of work, taking different methods:
  - The line of work [Moi19, GLLZ19, JPV21b] applies the coupling and linear programming method, initiated by Moitra. This method is deterministic and made to work for general CSP instances in [JPV21b]. However, this method suffers from a $n^{\text{poly}(k,\Delta,\log q)}$ running time, which is exponential if $k, \Delta = \Omega(1)$.
  - Another line of work [FGYZ21, FHY21, JPV21a, HSW21] focuses on fast sampling and uses the static mark/unmark paradigm(later refined and generalized to entropy compression/state tensorization) to overcome the connectivity barrier in solution spaces.

A missing piece

- All existing fast algorithms for sampling LLL relied on some projection of the solution space to a much smaller space where the barrier of disconnectivity could be circumvented because the images of the projection might collide and were well connected.

A missing piece

- All existing fast algorithms for sampling LLL relied on some projection of the solution space to a much smaller space where the barrier of disconnectivity could be circumvented because the images of the projection might collide and were well connected.
- Such projection may be hard to find (or not exist) for general CSP instances

A missing piece

- All existing fast algorithms for sampling LLL relied on some projection of the solution space to a much smaller space where the barrier of disconnectivity could be circumvented because the images of the projection might collide and were well connected.
- Such projection may be hard to find (or not exist) for general CSP instances
- It is possible that the non-atomicity of general CSPs might have imposed greater challenges to the sampling LLL than to its constructive counterpart.

A missing piece

- All existing fast algorithms for sampling LLL relied on some projection of the solution space to a much smaller space where the barrier of disconnectivity could be circumvented because the images of the projection might collide and were well connected.
- Such projection may be hard to find (or not exist) for general CSP instances
- It is possible that the non-atomicity of general CSPs might have imposed greater challenges to the sampling LLL than to its constructive counterpart.
- A major open problem:
  Is there a fast algorithm for general CSP instances in the LLL regime?

Our results

Paper	Instance	Condition	Complexity	Technique
[HSZ19]	monotone CNF	$p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot n\log n$	MCMC
[GJL19]	general CSP	$s \geq \min(\log\Delta k, k/2)$ $p\Delta^2 \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot n$	PRS
[BGG+19]	monotone CNF	$p\Delta^2 \gtrsim 1$	**NP**-hard	lower bound
[Moi19]	CNF	$p\Delta^{60} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta)}$	Mark/unmark+Coupling+LP
[GLLZ19]	Hypergraph coloring	$p\Delta^{16} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[FGYZ21]	CNF	$p\Delta^{20} \lesssim 1$	$\mathrm{poly}(k,\Delta)\cdot\tilde{O}(n^{1.001})$	Mark/unmark+MCMC
[FHY21]	atomic CSP	$p\Delta^{350} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot\tilde{O}(n^{1.001})$	Entropy Compression+MCMC
[JPV21b]	general CSP	$p\Delta^{7} \lesssim 1$	$n^{\mathrm{poly}(k,\Delta,\log q)}$	Adaptive mark/unmark+Coupling+LP
[JPV21a]	atomic CSP	$p\Delta^{7.043} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot\tilde{O}(n^{1.001})$	Entropy Compression+MCMC
[HSW21]	atomic CSP	$p\Delta^{5.714} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot n\log n$ expected	State tensorization+MCMC
This work	general CSP	$p\Delta^{7} \lesssim 1$	$\mathrm{poly}(q,k,\Delta)\cdot n\log n$ expected	A new marginal sampler

Our results(sampling)

Theorem 1.1(informal)

There is an algorithm such that given as input any $\varepsilon \in (0,1)$ and any CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$ with n variables satisfying

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

the algorithm terminates within $\mathrm{poly}(q, k, \Delta) \cdot n \log\left(\frac{n}{\varepsilon}\right)$ time in expectation and outputs an almost uniform sample of satisfying assignments for Φ within ε total variation distance.

Our results(inference)

Theorem 1.5(informal)

There is an algorithm such that given as input any $\varepsilon \in (0, 1)$, any CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$ satisfying (1), and any $v \in V$, the algorithm returns a random value $x \in Q_v$ distributed approximately as μ_v within total variation distance ε, within $\mathrm{poly}\,(q, k, \Delta, \log(1/\varepsilon))$ time in expectation.

Theorem 1.6(informal)

There is an algorithm such that given as input any $\varepsilon, \delta \in (0, 1)$, any CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$ satisfying (1), and any $v \in V$, the algorithm returns for every $x \in Q_v$ an ε-approximation of the marginal probability $\mu_v(x)$ within $\mathrm{poly}\,(q, k, \Delta, 1/\varepsilon, \log(1/\delta))$ time with probability at least $1 - \delta$.

Some remarks

- As in the case of algorithmic LLL [MT10, HV15], we assume an abstraction of constraint evaluations, because arbitrary constraint functions defined on a super-constant number of variables can be highly nontrivial to express and evaluate.

Some remarks

- As in the case of algorithmic LLL [MT10, HV15], we assume an abstraction of constraint evaluations, because arbitrary constraint functions defined on a super-constant number of variables can be highly nontrivial to express and evaluate.
    - We assume we can efficiently check if some constraint c is already satisfied under some assignment $\mathcal{Q}_\Lambda = \bigotimes_{v \in \Lambda} Q_v$ specified on a subset $\Lambda \subseteq \mathsf{vbl}(c)$ of variables

Some remarks

- As in the case of algorithmic LLL [MT10, HV15], we assume an abstraction of constraint evaluations, because arbitrary constraint functions defined on a super-constant number of variables can be highly nontrivial to express and evaluate.
  - We assume we can efficiently check if some constraint c is already satisfied under some assignment $\mathcal{Q}_\Lambda = \bigotimes_{v \in \Lambda} Q_v$ specified on a subset $\Lambda \subseteq \mathsf{vbl}(c)$ of variables
- Our sampler is perfect if we can further determinsitically estimate the probability that a constraint c is violated given a partially specified assignment σ, possibly with gaps.

Some remarks

- As in the case of algorithmic LLL [MT10, HV15], we assume an abstraction of constraint evaluations, because arbitrary constraint functions defined on a super-constant number of variables can be highly nontrivial to express and evaluate.
  - We assume we can efficiently check if some constraint c is already satisfied under some assignment $\mathcal{Q}_\Lambda = \bigotimes_{v \in \Lambda} Q_v$ specified on a subset $\Lambda \subseteq \mathsf{vbl}(c)$ of variables
- Our sampler is perfect if we can further determinsitically estimate the probability that a constraint c is violated given a partially specified assignment σ, possibly with gaps.
- Our sampler relies on a local marginal sampler inspired from [AJ21], which achieves sublinear running time in inference problem.

The marginal distribution

- μ: uniform distribution over all CSP solutions Ω
- μ_v: distribution of X_v where $X \sim \mu$
- μ_v^σ: μ_v conditional on some partial assignment σ
- partial assignment: assignment only on a subset of variables

Local uniformity

- A crucial property in the local lemma regime is the marginal distribution on any variable is close to uniform.

Local uniformity

- A crucial property in the local lemma regime is the marginal distribution on any variable is close to uniform.

Local uniformity([HSS11])

Given a CSP formula $\Phi = (V, \mathcal{Q}, \mathcal{C})$, if $\mathrm{e}p\Delta < 1$, then for any variable $v \in V$ and any value $x \in Q_v$, it holds that

$$\frac{1}{q_v} - \eta \leq \mu_v(x) \leq \frac{1}{q_v} + \eta,$$

where $\eta = (1 - \mathrm{e}p)^{-\Delta} - 1$.

Local uniformity

- Let's see an example first. Assume $Q_v = \{1, 2, 3, 4, 5\}$.

Local uniformity

- Let's see an example first. Assume $Q_v = \{1, 2, 3, 4, 5\}$.

Figure: Uniform distribution over Q_v

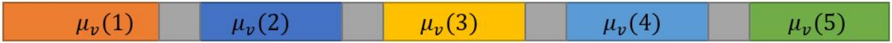


Figure: the marginal distribution $\mu_v(\cdot)$

Local uniformity

- Sampling from $\mu_v(\cdot)$ can be viewed as choosing $r \in [0, 1)$ randomly and taking the interval r falls into as the desired sample

Local uniformity

- Sampling from $\mu_v(\cdot)$ can be viewed as choosing $r \in [0, 1)$ randomly and taking the interval r falls into as the desired sample

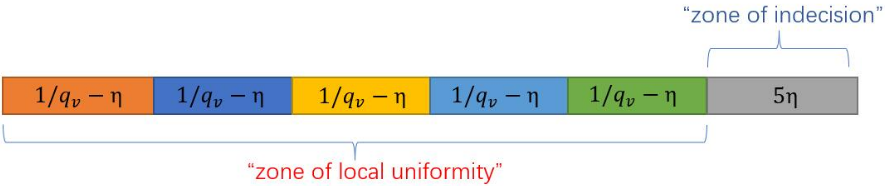


Figure: the marginal distribution $\mu_v(\cdot)$, after rearranging

Marginal sampler, first try

- Let $\theta_v = \frac{1}{q_v} - \eta$, then the "zone of local uniformity" has size $q_v\theta_v$, and the "zone of indecision" has size $1 - q_v\theta_v$.
- Suppose we draw $r \in [0, 1)$ first, if $r \le q_v\theta_v$, then we already know the result!

Marginal sampler, first try

- Let $\theta_v = \frac{1}{q_v} - \eta$, then the "zone of local uniformity" has size $q_v\theta_v$, and the "zone of indecision" has size $1 - q_v\theta_v$.
- Suppose we draw $r \in [0, 1)$ first, if $r \leq q_v\theta_v$, then we already know the result!
- What if $r > q_v\theta_v$?

Marginal sampler, first try

- Let $\theta_v = \frac{1}{q_v} - \eta$, then the "zone of local uniformity" has size $q_v\theta_v$, and the "zone of indecision" has size $1 - q_v\theta_v$.
- Suppose we draw $r \in [0, 1)$ first, if $r \leq q_v\theta_v$, then we already know the result!
- What if $r > q_v\theta_v$?
- The distribution in the "zone of local indecision" is a linear transform $\mathcal{D}$ of the marginal distribution μ_v:

$$\forall x \in Q_v, \quad \mathcal{D}(x) = \frac{\mu_v(x) - \theta_v}{1 - q_v\theta_v}$$

Marginal sampler, first try

- How to sample from $\mathcal{D}(\cdot) = \frac{\mu_v(\cdot) - \theta_v}{1 - q_v \theta_v}$?

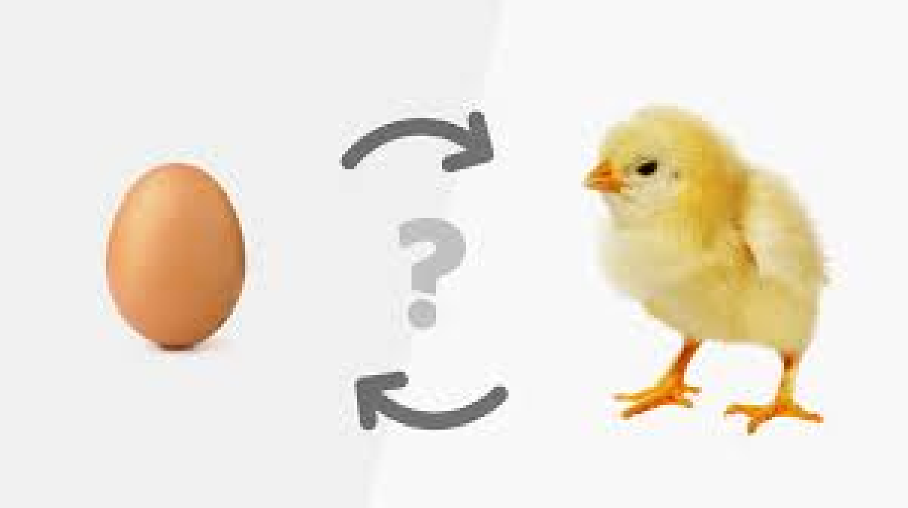
Marginal sampler, first try

- How to sample from $\mathcal{D}(\cdot) = \frac{\mu_v(\cdot)-\theta_v}{1-q_v\theta_v}$?
- Suppose we have access to an oracle to can sample from $\mu_v(\cdot)$, we can use known constructions of Bernoulli Factory algorithms[NP05, Hub16, DHKN17] to sample from $\mathcal{D}(\cdot)$.

Marginal sampler, first try

- How to sample from $\mathcal{D}(\cdot) = \frac{\mu_v(\cdot) - \theta_v}{1 - q_v \theta_v}$?
- Suppose we have access to an oracle to can sample from $\mu_v(\cdot)$, we can use known constructions of Bernoulli Factory algorithms[NP05, Hub16, DHKN17] to sample from $\mathcal{D}(\cdot)$.
- Sampling from $\mu_v(\cdot) \implies$ Sampling from $\mathcal{D}(\cdot)$ with probability $1 - q_v \theta_v$.
- Sampling from $\mathcal{D}(\cdot) \implies$ Sampling from $\mu_v(\cdot)$.

Chicken-egg dilemma?



Back to rejection sampling

- Note that we have a marginal sampler equipped using the idea of rejection sampling.

Back to rejection sampling

- Note that we have a marginal sampler equipped using the idea of rejection sampling.

Sampling from $\mu_v(\cdot)$ using rejection sampling

Repeat the following procedure:

- Use rejection sampling to draw a sample $X \sim \mu$
- return X_v as the sample

Back to rejection sampling

- Note that we have a marginal sampler equipped using the idea of rejection sampling.

Sampling from $\mu_v(\cdot)$ using rejection sampling

Repeat the following procedure:

- Use rejection sampling to draw a sample $X \sim \mu$
- return X_v as the sample

- Constraints satisfied by the partial assignment deconstructs Φ into connected components.

Back to rejection sampling

- Note that we have a marginal sampler equipped using the idea of rejection sampling.

Sampling from $\mu_v(\cdot)$ using rejection sampling

Repeat the following procedure:

- Use rejection sampling to draw a sample $X \sim \mu$
- return X_v as the sample

- Constraints satisfied by the partial assignment deconstructs Φ into connected components.
- This is efficient if the connected component containing v is logarithmically small!

Factorizing(with a marginal oracle)

- If we have an access of an oracle $\mathcal{O}$ that samples the (possibly conditional on partial assignments) marginal distribution on some variables other than v, can we use this oracle to sample from $\mu_v(\cdot)$?

Factorizing(with a marginal oracle)

- If we have an access of an oracle $\mathcal{O}$ that samples the (possibly conditional on partial assignments) marginal distribution on some variables other than v, can we use this oracle to sample from $\mu_v(\cdot)$?

A factorizing process

- Repeatedly choose closest not assigned variable u in the same connected component as v, and use $\mathcal{O}$ to draw a value for u(conditional on current partial assignment) until no such variable exist.
- Ideally, if p is small enough, the connected component containing v is logarithmically small with high probability.

Frozen and fixed

- In our setting of local lemma regime, there's some issue when realizing such factorizing process: we may fall out of local lemma regime when conditioning on partial assignments.

Frozen and fixed

- In our setting of local lemma regime, there's some issue when realizing such factorizing process: we may fall out of local lemma regime when conditioning on partial assignments.
- This is resolved by the idea of "freezing" constraints with a high violation probability, which dates back to [Bec91].

Frozen and fixed

- In our setting of local lemma regime, there's some issue when realizing such factorizing process: we may fall out of local lemma regime when conditioning on partial assignments.
- This is resolved by the idea of "freezing" constraints with a high violation probability, which dates back to [Bec91].
- Mark a constraint as frozen and all its unassigned variables as fixed if its conditional violation probability exceeds some threshold $p' \geq p$. Also we fix all assigned variables.

Factorizing(cont'd)

A factorizing process, adapted in LLL regime

- Repeatedly choose a closest not fixed variable u in the same component as v, and use $\mathcal{O}$ to draw a value for u(conditional on current partial assignment) until no such variable exist.
- It can be shown that at the end of this process, with properly chosen p', the connected component containing v is logarithmically small with high probability.

Putting things together

- Sampling from $\mu_v(\cdot) \Longrightarrow$ Sampling from $\mathcal{D}(\cdot)$ with probability $1 - q_v\theta_v$.
- Sampling from $\mathcal{D}(\cdot) \Longrightarrow$ Sampling from $\mu_v(\cdot)$.

Putting things together

- Sampling from $\mu_v(\cdot) \implies$ Sampling from $\mathcal{D}(\cdot)$ with probability $1 - q_v\theta_v$.
- Sampling from $\mathcal{D}(\cdot) \implies$ Sampling from $\mu_v(\cdot)$.
- Sampling from $\mu_v(\cdot) \implies$ First sampling from other variables to factorize the formula, then use rejection sampling to sample from $\mu_v(\cdot)$.

Putting things together

- Sampling from $\mu_v(\cdot) \Longrightarrow$ Sampling from $\mathcal{D}(\cdot)$ with probability $1 - q_v\theta_v$.
- Sampling from $\mathcal{D}(\cdot) \Longrightarrow$ Sampling from $\mu_v(\cdot)$.
- Sampling from $\mu_v(\cdot) \Longrightarrow$ First sampling from other variables to factorize the formula, then use rejection sampling to sample from $\mu_v(\cdot)$.
- Recurse!

Putting things together

- Sampling from $\mu_v(\cdot) \implies$ Sampling from $\mathcal{D}(\cdot)$ with probability $1 - q_v\theta_v$.
- Sampling from $\mathcal{D}(\cdot) \implies$ Sampling from $\mu_v(\cdot)$.
- Sampling from $\mu_v(\cdot) \implies$ First sampling from other variables to factorize the formula, then use rejection sampling to sample from $\mu_v(\cdot)$.
- Recurse!
- Rejection Sampling serves as the basis of the recursion.

Putting things together

- Sampling from $\mu_v(\cdot) \Longrightarrow$ Sampling from $\mathcal{D}(\cdot)$ with probability $1 - q_v\theta_v$.
- Sampling from $\mathcal{D}(\cdot) \Longrightarrow$ Sampling from $\mu_v(\cdot)$.
- Sampling from $\mu_v(\cdot) \Longrightarrow$ First sampling from other variables to factorize the formula, then use rejection sampling to sample from $\mu_v(\cdot)$.
- Recurse!
- Rejection Sampling serves as the basis of the recursion.
- The recursion converges if $1 - q_v\theta_v$ is small relative to the number of recursive calls needed in the factorizing process.

The marginal sampler

Algorithm for sampling from $\mu_v^\sigma(\cdot)$

1. Choose $r \in [0, 1)$ uniformly at random
2. If $r < q_v\theta_v$, return the $\lceil r/\theta_v \rceil$-th value in Q_v
3. Otherwise, return a sample from $\mathcal{D}_v^\sigma = \frac{\mu_v^\sigma - \theta_v}{1 - q_v\theta_v}$

The marginal sampler(cont'd)

Algorithm for sampling from $\mathcal{D}_v^\sigma(\cdot)$

1. If v is already factorized conditioning on σ, return a sample from $\mathcal{D}_v^\sigma(\cdot)$ using Bernoulli factory algorithm with rejection sampling procedure as input coins.
2. Otherwise,
   2.1 Properly choose some variable u.
   2.2 Choose $r \in [0, 1)$ uniformly at random.
   2.3 If $r < q_v\theta_v$, set $\sigma(u)$ as the $\lceil r/\theta_v \rceil$-th value in Q_u.
   2.4 Otherwise, set $\sigma(u)$ as a sample from $\mathcal{D}_u^\sigma$ by recursively calling this algorithm.
   2.5 return a sample from $\mathcal{D}_v^\sigma$ by recursively calling this algorithm.

From marginal sampler to a full sampler

- Sequential sampling while using the same idea of "freezing" with the same parameter p' to guarantee the properties in the local lemma regime.

From marginal sampler to a full sampler

- Sequential sampling while using the same idea of "freezing" with the same parameter p' to guarantee the properties in the local lemma regime.
- It can be shown that after a first sequential sampling, the whole formula scatters into connected components with logarithmic sizes with high probability.

From marginal sampler to a full sampler

- Sequential sampling while using the same idea of "freezing" with the same parameter p' to guarantee the properties in the local lemma regime.
- It can be shown that after a first sequential sampling, the whole formula scatters into connected components with logarithmic sizes with high probability.
- Then use rejection sampling to complete the assignment.

The sampling algorithm

Algorithm for sampling from μ

1. Set X as the empty assignment
2. For each $v \in V$
   2.1 If v is not fixed conditioning on X, sample $X(v)$ from μ_v^X using the marginal sampler introduced before.
3. Complete X using rejection sampling.

Analysis of efficiency

- Construct an abstract data structure recursive cost tree that for each (σ, v), if one calls the algorithm for sampling from μ_v^σ, calculates for each (X, u) the probability that the algorithm for sampling from $\mathcal{D}_u^X$ is called

Analysis of efficiency

- Construct an abstract data structure recursive cost tree that for each (σ, v), if one calls the algorithm for sampling from μ_v^σ, calculates for each (X, u) the probability that the algorithm for sampling from $\mathcal{D}_u^X$ is called
- Use linearity of expectation to analyze the possible length of path generated by another process $\mathrm{Path}(\sigma)$.

Analysis of efficiency

- Construct an abstract data structure recursive cost tree that for each (σ, v), if one calls the algorithm for sampling from μ_v^σ, calculates for each (X, u) the probability that the algorithm for sampling from $\mathcal{D}_u^X$ is called
- Use linearity of expectation to analyze the possible length of path generated by another process $\mathrm{Path}(\sigma)$.
- A classical technique dates back to [Alo91]: If the length of path is too long, then there exists a large $\{2, 3\}$-tree of several bad events, which happens with exponentially low probability.

Analysis of efficiency

- Construct an abstract data structure recursive cost tree that for each (σ, v), if one calls the algorithm for sampling from μ_v^σ, calculates for each (X, u) the probability that the algorithm for sampling from $\mathcal{D}_u^X$ is called
- Use linearity of expectation to analyze the possible length of path generated by another process $\mathrm{Path}(\sigma)$.
- A classical technique dates back to [Alo91]: If the length of path is too long, then there exists a large $\{2, 3\}$-tree of several bad events, which happens with exponentially low probability.
- Still many technicalities are omitted.

Thank you!

arXiv:2204.01520

Bibliography I

Konrad Anand and Mark Jerrum.
Perfect sampling in infinite spin systems via strong spatial mixing.
CoRR, abs/2106.15992, 2021.

Noga Alon.
A parallel algorithmic version of the local lemma.
Random Struct. Algorithms, 2(4):367–378, 1991.
(Conference version in *FOCS*'91).

József Beck.
An algorithmic approach to the Lovász local lemma.
Random Struct. Algorithms, 2(4):343–365, 1991.

Bibliography II

Ivona Bezáková, Andreas Galanis, Leslie A. Goldberg, Heng Guo, and Daniel Štefankovič.
Approximation via correlation decay when strong spatial mixing fails.
SIAM J. Comput., 48(2):279–349, 2019.

Shaddin Dughmi, Jason D. Hartline, Robert Kleinberg, and Rad Niazadeh.
Bernoulli factories and black-box reductions in mechanism design.
In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 158–169. ACM, New York, 2017.

Bibliography III

Paul Erdős and László Lovász.
Problems and results on 3-chromatic hypergraphs and some related questions.
Infinite and finite sets, volume 10 of Colloquia Mathematica Societatis János Bolyai, pages 609–628, 1975.

Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang.
Fast sampling and counting k-sat solutions in the local lemma regime.
Journal of the ACM (JACM), 68(6):1–42, 2021.

Weiming Feng, Kun He, and Yitong Yin.
Sampling constraint satisfaction solutions in the local lemma regime.
In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1565–1578, 2021.

Bibliography IV

Heng Guo, Mark Jerrum, and Jingcheng Liu.
Uniform sampling through the Lovász local lemma.
J. ACM, 66(3):Art. 18, 31, 2019.

Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang.
Counting hypergraph colorings in the local lemma regime.
SIAM Journal on Computing, 48(4):1397–1424, 2019.

Bernhard Haeupler, Barna Saha, and Aravind Srinivasan.
New constructive aspects of the Lovász local lemma.
J. ACM, 58(6):28, 2011.
(Conference version in *FOCS*'10).

Kun He, Xiaoming Sun, and Kewen Wu.
Perfect sampling for (atomic) lovász local lemma.
CoRR, abs/2107.03932, 2021.

Bibliography V

Jonathan Hermon, Allan Sly, and Yumeng Zhang.
Rapid mixing of hypergraph independent sets.
Random Struct. Algorithms, 54(4):730–767, 2019.

Mark Huber.
Nearly optimal Bernoulli factories for linear functions.
Combin. Probab. Comput., 25(4):577–591, 2016.

Nicholas J. A. Harvey and Jan Vondrák.
An algorithmic proof of the Lovász local lemma via resampling oracles.
In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1327–1345. IEEE Computer Soc., Los Alamitos, CA, 2015.

Bibliography VI

Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong.
On the sampling lovász local lemma for atomic constraint satisfaction problems.
CoRR, abs/2102.08342, 2021.

Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong.
Towards the sampling lovász local lemma.
In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 173–183. IEEE, 2021.

Ankur Moitra.
Approximate counting, the Lovász local lemma, and inference in graphical models.
J. ACM, 66(2):10:1–10:25, 2019.
(Conference version in *STOC*'17).

Bibliography VII

Robin A. Moser and Gábor Tardos.
A constructive proof of the general Lovász local lemma.
J. ACM, 57(2):11, 2010.

Şerban Nacu and Yuval Peres.
Fast simulation of new coins from old.
Ann. Appl. Probab., 15(1A):93–115, 2005.