2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-6654-5519-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/FOCS54457.2022.00021

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

Kun He
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences
Beijing, China
hekun@ict.ac.cn

Chunyang Wang
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
wcysai@smail.nju.edu.cn

Yitong Yin
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
yinyt@nju.edu.cn

***Abstract*—We give a fast algorithm for sampling uniform solutions of *general* constraint satisfaction problems (CSPs) in a local lemma regime. The expected running time of our algorithm is near-linear in n and a fixed polynomial in Δ, where n is the number of variables and Δ is the max degree of constraints. Previously, up to similar conditions, sampling algorithms with running time polynomial in both n and Δ, only existed for the almost atomic case, where each constraint is violated by a small number of forbidden local configurations.**

***Index Terms*—sampling, constraint satisfaction problem, Lovász Local Lemma**

I. Introduction

Constraint satisfaction problems (CSPs) are one of the most fundamental objects in computer science. A CSP is described by a collection of constraints defined on a set of variables. Formally, an instance of constraint satisfaction problem, called a *CSP formula*, is denoted by $\Phi = (V, \boldsymbol{Q}, C)$. Here, V is a set of $n = |V|$ variables; $\boldsymbol{Q} \triangleq \bigotimes_{v\in V} Q_v$ is a product space of all assignments of variables, where each Q_v is a finite domain of size $q_v \triangleq |Q_v| \geq 2$ over where the variable v ranges; and C gives a collection of local constraints, such that each $c \in C$ is a constraint function $c : \bigotimes_{v\in \mathsf{vbl}(c)} Q_v \to \{\texttt{True}, \texttt{False}\}$ defined on a subset of variables, denoted by $\mathsf{vbl}(c) \subseteq V$. An assignment $\boldsymbol{x} \in \boldsymbol{Q}$ is called *satisfying* for Φ if

$$\Phi(\boldsymbol{x}) \triangleq \bigwedge_{c\in C} c\left(\boldsymbol{x}_{\mathsf{vbl}(c)}\right) = \texttt{True}.$$

A full version of the paper is available at https://arxiv.org/abs/2204.01520.

This work is supported by National Key R&D Program of China 2018YFB1003202. Kun He is supported by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA27000000, the National Natural Science Foundation of China Grants No. 62002231, 61832003.

The followings are some key parameters of a CSP formula $\Phi = (V, \boldsymbol{Q}, C)$:

- *domain size* $q = q_\Phi \triangleq \max_{v\in V} |Q_v|$;
- *width* $k = k_\Phi \triangleq \max_{c\in C} |\mathsf{vbl}(c)|$;
- *constraint degree* $\Delta = \Delta_\Phi$ where $\Delta_\Phi \triangleq \max_{c\in C} |\{c' \in C \mid \mathsf{vbl}(c) \cap \mathsf{vbl}(c') \neq \emptyset\}|$;
- *violation probability* $p = p_\Phi \triangleq \max_{c\in C} \mathbb{P}[\neg c]$, where $\mathbb{P}$ denotes the law for the uniform assignment, in which each $v \in V$ draws its evaluation from Q_v uniformly and independently at random.

The famous *Lovász Local Lemma (LLL)* [1] provides a sufficient criterion for the satisfiability of Φ. Specifically, a satisfying assignment for a CSP formula Φ exists if

$$\mathrm{e}p\Delta \leq 1. \tag{1}$$

Due to a lower bound of Shearer [2], such "LLL condition" for the existence of satisfying solution is essentially tight if only knowing p and Δ. On the other hand, the *algorithmic* or *constructive LLL* seeks to find a solution efficiently. A major breakthrough was the Moser-Tardos algorithm [3], which guarantees to find a satisfying assignment efficiently under the LLL condition in (1).

The sampling LLL. We are concerned with the problem of *sampling Lovász Local Lemma*, which has drawn considerable attention in recent years [4]–[15]. In the context of CSP, it seeks to provide an efficient sampling algorithm for (nearly) uniform generation of satisfying assignments for the CSPs in an LLL-like regime. This sampling LLL problem is closely related to the problem of estimating the volume of solution spaces or the partition functions of statistical physics systems, and is motivated by fundamental tasks, including the

978-1-6654-5519-0/22/$31.00 ©2022 IEEE
DOI 10.1109/FOCS54457.2022.00021

probabilistic inferences in graphical models [5] and the network reliability problems [4], [16], [17].

This problem of sampling LLL turns out to be computationally more challenging than the traditional algorithmic LLL, which requires constructing an arbitrary satisfactory assignment, not necessarily following the correct distribution. For example, when used as a sampling algorithm, the Moser-Tardos algorithm can only guarantee correct sampling on restrictive classes of CSPs [4]. Due to the computational lower bounds shown in [14], [18], a strengthened LLL condition with $c \geq 2$:

$$p\Delta^c \lesssim 1, \tag{2}$$

is necessary for the tractability of sampling LLL, even restricted to typical specific sub-classes of CSPs, such as CNF or hypergraph coloring. Here $\lesssim$ ignores the lower-order terms and the constant factor.

In a seminal work of Moitra [5], a very innovative algorithm was given for sampling almost uniform k-CNF solutions assuming an LLL condition $p\Delta^{60} \lesssim 1$. This sampling algorithm was based on deterministic approximate counting by solving linear programs on properly factorized formulas and has a running time of $n^{\mathrm{poly}(k,\Delta)}$. This LP-based approach was later extended to hypergraph coloring [6] and random CNF formulas [7], and finally in a work of Jain, Pham and Vuong [12] to all CSPs satisfying a substantially improved LLL condition $p\Delta^7 \lesssim 1$. All these deterministic approximate counting based algorithms suffered from an $n^{\mathrm{poly}(k,\Delta)}$ time cost.

Historically, rapidly mixing Markov chains have been the canonical sampling algorithms, and often have near-linear time efficiency. However, for sampling LLL, there used to be a fundamental barrier for Markov chains. That is, despite the ubiquity of solutions, the solution space of CSPs may be highly disconnected through the transition of local Markov chains.

This barrier of disconnectivity was circumvented in a breakthrough of Feng *et al.* [9], in which a rapidly mixing *projected* random walk was simulated efficiently on a subset of variables constructed using the marking/unmarking strategy invented in [5]. Assuming an LLL condition $p\Delta^{20} \lesssim 1$, this new algorithm could generate an almost uniform k-CNF solution using a time cost within $\mathrm{poly}(k,\Delta)\cdot n^{1.0001}$, which is close to linear in the number of variables n. By observing that this marking/unmarking of variables was, in fact a specialization in the Boolean case of compressing variables' states, this Markov chain based fast sampling approach was generalized in [10] to CSPs beyond the Boolean domain, specifically, to all almost *atomic* CSPs (which we will explain later), assuming an LLL condition $p\Delta^{350} \lesssim 1$. This bound was remarkably improved to $p\Delta^{7.04} \lesssim 1$ in another work of Jain, Pham and Vuong [11] through a clever witness-tree-like information percolation analysis of the mixing time, which was also used later to support a perfect sampler through the coupling from the past (CFTP) in [13] with a further improved condition $p\Delta^{5.71} \lesssim 1$.

All these fast algorithms for sampling LLL are restricted to the (almost) atomic CSPs, in which each constraint c is violated by exactly one (or very few) forbidden assignment(s) on $\mathsf{vbl}(c)$.

Challenges for general CSP. New techniques are needed for fast sampling LLL for general CSPs. 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. In order to efficiently simulate the random walk on the projected space and to recover a random solution from a random image, one would hope that the CSP formula were well "factorized" into small clusters most of the time because many constraints had already been satisfied for sure given the current image, which was indeed the case for fast sampling LLL for atomic CSPs [9]–[11], [13]. But for general non-atomic CSPs, it may no longer be the case, because now a bad event (violation of a constraint) may be highly non-elementary, and hence is no longer that easy to avoid cleanly after projection, which breaks the factorization.

It is possible that the non-atomicity of general CSPs might have imposed greater challenges to the sampling LLL than to its constructive counterpart. To see this, note that general CSPs can be simulated by atomic ones: by replacing each general constraint c having N forbidden assignments on $\mathsf{vbl}(c)$, with N atomic constraints on the same $\mathsf{vbl}(c)$ each forbidding one assignment. Such simulation would increase the constraint degree Δ by a factor of at most N and also decrease the violation probability p by a factor of N. For the classic LLL condition (1) where p and Δ are homogeneous, this would not change the LLL condition; but the regime for the sampling LLL captured by (2) would be significantly reduced, since there p and Δ are necessarily not homogeneous due to the lower bounds in [14], [18]. This situation seems to suggest that the non-atomicity of general CSPs might impose bigger challenges to the sampling LLL than to the existential/constructive LLL.

Indeed, prior to our work, it was not known for general CSPs with unbounded width k and degree Δ, whether the sampling problem is polynomial-time tractable under an LLL condition like (2).

A. Our results

In this paper, we answer the above open question positively. We give a new algorithm that departs from all prior fast samplers based on Markov chains and achieves, for the first time, a fast sampling of almost uniform satisfying solutions for general CSPs in a local lemma regime.

As in the case of algorithmic LLL [3], [19], 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. Specifically, we assume the following evaluation oracle for checking whether a constraint is already satisfied by a partially specified assignment.

Assumption 1 (**evaluation oracle**)**.** There is an *evaluation oracle* for $\Phi = (V, \mathcal{Q}, C)$ such that given any constraint $c \in C$, any assignment $\sigma \in \mathcal{Q}_\Lambda \triangleq \bigotimes_{v\in\Lambda} \mathcal{Q}_v$ specified on a subset $\Lambda \subseteq \mathsf{vbl}(c)$ of variables, the oracle answers whether c is already satisfied by σ, i.e. $c(\tau) = \texttt{True}$ for all $\tau \in \mathcal{Q}_{\mathsf{vbl}(c)}$ that $\tau_\Lambda = \sigma_\Lambda$.

For specific classes of CSPs, e.g. k-CNF or hypergraph coloring, such an oracle is easy to realize.

Assuming such an oracle for constraint evaluations, we give the following fast, almost uniform sampler for general CSPs in a local lemma regime. Recall the parameters q, k, p, Δ of a CSP formula Φ.

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

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

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.

The formal statement of the theorem is in Theorem V.1 (for termination and correctness of sampling) and in Theorem VI.3 (for efficiency of sampling). The condition in (3) becomes $p\Delta^{7+o(1)} \lesssim 1$ when $p \le (qk)^{-\omega(1)}$, while a typical case is usually given by a much smaller $p \le q^{-\Omega(k)}$. The previous best bound for sampling general CSP solutions was that $q^3kp\Delta^7 < c$ for a small constant c, achieved by the deterministic approximate counting based algorithm in [12] whose running time was $(n/\varepsilon)^{\mathrm{poly}(k,\Delta,\log q)}$.

Let Z be the total number of satisfying assignments for Φ. A $\hat{Z}$ is called an *ε-approximation* of Z if $(1-\varepsilon)Z \le \hat{Z} \le (1+\varepsilon)Z$. By routinely going through the non-adaptive annealing process in [9], the approximate sampler in Theorem I.1 can be used as a black-box to give for any $\varepsilon \in (0,1)$ an ε-approximation of Z in time $\mathrm{poly}(q,k,\Delta)\cdot\tilde{O}\left(n^2\varepsilon^{-2}\right)$ with high probability.

1) Perfect sampler: The evaluation oracle in Assumption 1 in fact checks the sign of $\mathbb{P}[\neg c \mid \sigma]$, the probability that a constraint c is violated given a partially specified assignment σ. If further this probability can be estimated efficiently, then the sampling in Theorem I.1 can be made perfect, where the output sample follows exactly the target distribution.

Theorem I.2 (informal)**.** *For the input class of CSPs, if there is such an FPTAS for violation probability:*

- *for any constraint $c \in C$, any assignment $\sigma \in \mathcal{Q}_\Lambda$ specified on a subset $\Lambda \subseteq \mathsf{vbl}(c)$, and $0 < \varepsilon < 1$, an ε-approximation of $\mathbb{P}[\neg c \mid \sigma]$ is returned deterministically within $\mathrm{poly}(q,k,1/\varepsilon)$ time,*

then the algorithm in Theorem I.1 returns a perfect sample of uniform satisfying assignment within $\mathrm{poly}(q,k,\Delta)\cdot n$ time in expectation under the same condition (3)*.*

The formal statement of the theorem is in Theorem V.1 (for termination and correctness of sampling) and in Theorem VI.1 (for efficiency of sampling). In fact, we prove this perfect sampler first, and then realize the FPTAS assumed in Theorem I.2 using Monte Carlo experiments, which introduces a bounded bias to the sampling and gives us the approximate sampler claimed in Theorem I.1.

For concrete classes of CSPs defined by simple local constraints, it is no surprise to see that the probability $\mathbb{P}[\neg c \mid \sigma]$ almost always has an easy-to-compute closed-form expression, in which case we have a perfect sampler without assuming the oracles in Assumption 1 and in Theorem I.2.

The followings are two examples of non-atomic CSPs which admit linear-time perfect samplers.

Example I.3 (δ-robust k-SAT)**.** *The n variables are Boolean, each clause contains exactly k literals, and a clause is satisfied if and only if at least δk of its literals have the outcome* `True`*.*

- *For δ-robust k-SAT with variable degree d (each variable appears in at most d clauses) satisfying*

$$0 < \delta < \frac{1}{2}, \quad k \ge \frac{32\ln k + 28\ln d + 40}{(1-2\delta)^2},$$

a perfect sample of uniform satisfying solutions is returned within expected time $\mathrm{poly}(k,d)\cdot n$.

Example I.4 (δ-robust hypergraphs q-coloring)**.** *Each vertex is colored with one of the q colors, each hyperedge*

is k-uniform and is satisfied if and only if there are no $(1-\delta)k$ vertices with the same color.

- *For k-uniform hypergraphs on n vertices with maximum vertex degree d satisfying*

$$(1-\delta)k \geq 15, \quad q \geq \frac{7d^{\frac{7}{(1-\delta)k-3}} \cdot (6.1)^{\frac{1}{(1-\delta)}}}{(1-\delta)^{1.25}},$$

a perfect sample of uniform satisfying coloring is returned within expected time $\mathrm{poly}(q,k,d) \cdot n$.

2) Marginal sampler: The core component of our sampling algorithm is a *marginal sampler* for drawing from marginal distributions. Let $\mu = \mu_\Phi$ denote the uniform distribution over all satisfying assignments for Φ, and for each $v \in V$, let μ_v denote the marginal distribution at v induced by μ.

Theorem I.5 (informal)**.** *There is an algorithm such that given as input any $\varepsilon \in (0,1)$, any CSP formula $\Phi = (V, \mathcal{Q}, C)$ satisfying* (3)*, 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.*

This marginal sampler is also perfect under the same assumption as in Theorem I.2. Another byproduct of this marginal sampler is the following algorithm for probabilistic inference.

Theorem I.6 (informal)**.** *There is an algorithm such that given as input any $\varepsilon, \delta \in (0,1)$, any CSP formula $\Phi = (V, \mathcal{Q}, C)$ satisfying* (3)*, 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 $1-\delta$.*

The above two theorems are formally restated and proved in [20, Section 6].

By a self-reduction, the sampling and inference algorithms in Theorems I.5 and I.6 remain to hold for the marginal distributions μ_v^σ conditional on a feasible partially specified assignment σ, as long as the LLL condition (3) is satisfied by the new instance Φ^σ obtained from pinning σ onto Φ.

Both the above algorithms for marginal sampling and probabilistic inference are **local algorithms whose costs are independent of** n. Previously, in order to simulate or estimate the marginal distribution of a variable, it was often necessary to generate a full assignment on all n variables, or at least pay no less than that. One might have asked the following natural question:

Can these locally defined sampling or inference problems be solved at a local cost?

However, decades have passed, and only recently has such a novel local algorithm been discovered for marginal distributions in infinite spin systems [21], which is also our main source of inspiration.

B. Technique overview

As we have explained before, non-atomicity of constraints causes a barrier for the current Markov chain based algorithms [9]–[11], [13], [15]. There is another family of sampling algorithms, which we call "resampling based" algorithms [4], [22]–[25]. These algorithms use resampling of variables to fix the assignment until it follows the right distribution, morally like the Moser-Tardos algorithm, and they are not as affected by disconnectivity of solution space as Markov chains, but here a principle to ensure the correct sampling is to resample the variables that the algorithm has observed and conditioned on, which also causes trouble on non-atomic constraints, because to ensure such constraints are satisfied, the algorithm has to observe too many variables, whose resampling would cancel the progress of the algorithm.

We adopt a new idea of sampling, which we call the *recursive marginal sampler*. It is somehow closer to the resampling based algorithms than to the Markov chains, but thanks to its recursive nature, the algorithm avoids excessive resampling. This algorithm is inspired by a recent novel algorithm of Anand and Jerrum [21] for perfectly sampling in infinite spin systems, where a core component is such a marginal sampler that can draw a spin according to its marginal distribution.

Now let us consider the uniform distribution μ over all satisfying assignments of a CSP formula $\Phi = (V, \mathcal{Q}, C)$, and its marginal distribution μ_v at a variable $v \in V$, say over domain $Q_v = [q]$. To sample from this μ_v over $[q]$, an idea is to exploit the so-called "local uniformity" property [26], which basically says that μ_v should not be far from a uniform distribution over $[q]$ in total variation distance when Φ satisfies some local lemma condition. Therefore, a uniform sample from $[q]$ already gives a coarse sample of μ_v. It remains to boost such a coarse sampler to a sampler with arbitrary precision.

By the local uniformity, there exists a $\theta < \frac{1}{q}$ close enough to $\frac{1}{q}$, such that

$$\forall x \in [q], \quad \mu_v(x) \geq \theta. \tag{4}$$

The marginal distribution μ_v can then be divided as

$$q\theta \cdot \mathcal{U} + (1 - q\theta) \cdot \mathcal{D},$$

where $\mathcal{U}$ is the uniform distribution over $[q]$ and $\mathcal{D}$ gives a distribution of "overflow" mass such that:

$$\forall x \in [q], \quad \mathcal{D}(x) = \frac{\mu_v(x) - \theta}{1 - q\theta}.$$

Sampling from μ_v then can follow this strategy: with probability $q\theta$, the algorithm falls into the "zone of local uniformity" and returns a uniform sample from $\mathcal{U}$; with probability $1 - q\theta$, the algorithm falls into the "zone of indecision" and has to draw a sample from this overflow distribution $\mathcal{D}$, which can be done by constructing a Bernoulli factory that accesses μ_v as an oracle. But wait, if we had such an oracle for μ_v in the first place, why would sampling from μ_v even be a problem?

The above "chicken or egg" paradox is somehow resolved by a simple observation: if enough many other variables had already been sampled correctly, say with outcome X, then assuming a strong enough LLL condition, there is a good chance that the resulting formula Φ^X was "factorized" into small clusters, from where a standard rejection sampling on Φ^X would be efficient for sampling from μ_v^X, and overall from μ_v. Therefore, the sampling strategy is now corrected as: after falling into the "zone of indecision" and before trying to draw from the overflow distribution $\mathcal{D}$, the algorithm picks another variable u whose successful sampling might help factorize Φ, and recursively apply the marginal sampler at u to draw from u's current marginal distribution first. The only loose end now is that the LLL condition is not self-reducible, meaning it is not invariant under arbitrary pinning. We adopt the idea of "freezing" constraints used in [12] to guide the algorithm to pick variables for sampling. The LLL condition is replaced by a more refined invariant condition that guarantees for each variable picked for sampling, the same local uniformity as in (4) to persist throughout the algorithm, and also guarantees a good chance of factorization while there are no other variables to pick. To bound the fast convergence of the recursive sampler, in [21] the strategy was to show that the branching process given by the recursion tree always has decaying offspring number in expectation given the worst-case boundary condition, which is not true here. Instead, we apply a more average-case style analysis. Interestingly, some of our analysis and the LLL condition resemble those in [12] for a deterministic approximate counting algorithm with time complexity $n^{\mathrm{poly}(k,\Delta,\log q)}$.

II. Notations for CSP

We recall the definition of CSP formula $\Phi = (V, \mathcal{Q}, C)$ in Section I. We use $\Omega = \Omega_\Phi$ to denote the set of all satisfying assignments of Φ, and use $\mu = \mu_\Phi$ to denote the uniform distribution over Ω. Recall that $\mathbb{P}$ denotes the law for the uniform product distribution over $\mathcal{Q}$. For $C \subseteq C$, denote $\mathsf{vbl}(C) \triangleq \bigcup_{c \in C} \mathsf{vbl}(c)$; and for $\Lambda \subseteq V$, denote $\mathcal{Q}_\Lambda \triangleq \bigotimes_{v \in \Lambda} Q_v$. We introduce a notation for partial assignments.

Definition II.1 (partial assignment)**.** Given a CSP formula $\Phi = (V, \mathcal{Q}, C)$, define:

$$\mathcal{Q}^* \triangleq \bigotimes_{v \in V} (Q_v \cup \{\star, \star\}),$$

where $\star$ and $\star$ are two special symbols not in any Q_v. Each $\sigma \in \mathcal{Q}^*$ is called a *partial assignment.*

In a partial assignment $\sigma \in \mathcal{Q}^*$, each variable $v \in V$ is classified as follows:

- $\sigma(v) \in Q_v$ means that v is *accessed* by the algorithm and *assigned* with the value $\sigma(v) \in Q_v$;
- $\sigma(v) = \star$ means that v is just *accessed* by the algorithm but *unassigned* yet with a value in Q_v;
- $\sigma(v) = \star$ means that v is *unaccessed* by the algorithm and hence *unassigned* with any value.

Furthermore, we use $\Lambda(\sigma) \subseteq V$ and $\Lambda^+(\sigma) \subseteq V$ to respectively denote the sets of assigned and accessed variables in a partial assignment $\sigma \in \mathcal{Q}^*$, that is:

$$\Lambda(\sigma) \triangleq \{v \in V \mid \sigma(v) \in Q_v\}, \Lambda^+(\sigma) \triangleq \{v \in V \mid \sigma(v) \neq \star\}.$$

Given any partial assignment $\sigma \in \mathcal{Q}^*$ and variable $v \in V$, we further denote by $\sigma_{v \leftarrow x}$ the partial assignment obtained from modifying σ by replacing $\sigma(v)$ with $x \in Q_v \cup \{\star, \star\}$.

A partial assignment $\tau \in \mathcal{Q}^*$ is said to *extend* a partial assignment $\sigma \in \mathcal{Q}^*$ if $\Lambda(\sigma) \subseteq \Lambda(\tau)$, $\Lambda^+(\sigma) \subseteq \Lambda^+(\tau)$, and σ, τ agree with each other over all variables in $\Lambda(\sigma)$. A partial assignment $\sigma \in \mathcal{Q}^*$ is said to satisfy a constraint $c \in C$ if c is satisfied by all full assignments $\tau \in \mathcal{Q}$ that extend σ. A partial assignment $\sigma \in \mathcal{Q}^*$ is called *feasible* if there is a satisfying assignment $\tau \in \Omega$ that extends σ.

Given any feasible partial assignment $\sigma \in \mathcal{Q}^*$ and any $S \subseteq V$, we use σ_S to denote $\bigotimes_{v \in S} \sigma(v)$ and μ_S^σ to denote the marginal distribution induced by μ on S conditional on σ. For each $\tau \in \mathcal{Q}_S$, we have $\mu_S^\sigma(\tau) = \Pr_{X \sim \mu}[X_S = \tau \mid \forall v \in \Lambda(\sigma), X(v) = \sigma(v)]$. We further write $\mu_v^\sigma = \mu_{\{v\}}^\sigma$. Similar notation is used for the law $\mathbb{P}$ for the uniform product distribution over $\mathcal{Q}$. For $\sigma \in \mathcal{Q}^*$ and any event $A \subseteq \mathcal{Q}$, we have $\mathbb{P}[A \mid \sigma] = \Pr_{X \in \mathcal{Q}}[X \in A \mid \forall v \in \Lambda(\sigma), X(v) = \sigma(v)]$.

III. The Sampling Algorithm

We give our main algorithm for sampling almost uniform satisfying assignments for a CSP formula. Our presentation uses notations defined in Section II.

A. The main sampling algorithm

Our main sampling algorithm takes as input a CSP formula $\Phi = (V, \mathcal{Q}, C)$ with domain size $q = q_\Phi$, width $k = k_\Phi$, constraint degree $\Delta = \Delta_\Phi$, and violation probability $p = p_\Phi$, where the meaning of these parameters are as defined in Section I.

We suppose that the $n = |V|$ variables are enumerated as $V = \{v_1, v_2, \ldots, v_n\}$ in an arbitrary order. The CSP formula $\Phi = (V, \mathcal{Q}, C)$ is presented to the algorithm by the evaluation oracle in Assumption 1. Also assume that given any $c \in C$ (or $v \in V$), the $\mathsf{vbl}(c)$ (or $\{c \in C \mid v \in \mathsf{vbl}(c)\}$) can be retrieved.

The main algorithm (Algorithm 1) is the same as the main sampling frameworks in [6], [12]. A partial assignment $X \in \mathcal{Q}^*$ is maintained, initially as the empty assignment $X = \star^V$.

1) In the 1st phase, at each step it adaptively picks (in a predetermined order) a variable v that has enough "freedom" because it is not involved in any easy-to-violate constraint given the current X, and replaces $X(v)$ with a random value drawn by a subroutine MarginSample according to the correct marginal distribution μ_v^X.
2) When no such variable with enough freedom remains, the formula is supposed to be "factorized" enough into small clusters and the algorithm enters the 2nd phase, from where the partial assignment constructed in the 1st phase is completed to a uniform random satisfying assignment by a standard RejectionSampling subroutine.

A key threshold p' for the violation probability is fixed as below:

$$p' = \left(18\mathrm{e}^2 q^2 k \Delta^4\right)^{-1}, \tag{5}$$

which satisfies $p' > p = p_\Phi$, assuming the LLL condition in (3).

For the ease of exposition, we assume an oracle for approximately deciding whether a constraint becomes too easy to violate given the current partial assignment.

Assumption 2. There is an oracle such that given any partial assignment $\sigma \in \mathcal{Q}^*$ and any constraint $c \in C$, the oracle distinguishes between the two cases: $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$, and answers arbitrarily and consistently if otherwise. That is the answer to the undefined case $\mathbb{P}[\neg c \mid \sigma] \in [0.99p', p']$ can be either "yes" or "no" but remains the same for the same $\sigma_{\mathsf{vbl}(c)}$.

Such an oracle is clearly implied by the FPTAS for violation probability assumed in Theorem I.2 and will be explicitly realized in Section VI. For now, with respect to such an oracle, the classes of easy-to-violate constraints and their involved variables are defined as follows.

Definition III.1 (frozen and fixed)**.** Assume Assumption 2. Let $\sigma \in \mathcal{Q}^*$ be a partial assignment.

- A constraint $c \in C$ is called *σ-frozen* if it is reported $\mathbb{P}[\neg c \mid \sigma] > p'$ by the oracle in Assumption 2. Denote by $C^\sigma_{\mathsf{frozen}}$ the set of all σ-frozen constraints:
$$C^\sigma_{\mathsf{frozen}} \triangleq \{c \in C \mid c \text{ is reported by the oracle to satisfy } \mathbb{P}[\neg c \mid \sigma] > p'\}.$$
- A variable $v \in V$ is called *σ-fixed* if v is accessed in σ or is involved in some σ-frozen constraint. Denote by V^σ_{fix} the set of all σ-fixed variables:
$$V^\sigma_{\mathsf{fix}} \triangleq \Lambda^+(\sigma) \cup \bigcup_{c \in C^\sigma_{\mathsf{frozen}}} \mathsf{vbl}(c).$$

Similar ideas of freezing appeared in previous works on sampling and algorithmic LLL [12], [27]

Remark III.2 (one-sided error for frozen/fixed decision)**.** By the property of the oracle in Assumption 2, any constraint $c \in C$ with $\mathbb{P}[\neg c \mid \sigma] > p'$ must be in $C^\sigma_{\mathsf{frozen}}$, and any variable $v \in V$ involved in such a constraint must be in V^σ_{fix}; conversely, any σ-frozen constraint $c \in C^\sigma_{\mathsf{frozen}}$ must have $\mathbb{P}[\neg c \mid \sigma] \geq 0.99p'$ and any unaccessed σ-fixed variable $v \in V^\sigma_{\mathsf{fix}}$ must be involved in at least one of such constraints.

Algorithm 1: The sampling algorithm

Input: a CSP formula $\Phi = (V, \mathcal{Q}, C)$;
Output: a uniform random satisfying assignment $X \in \Omega_\Phi$;

$X \leftarrow \star^V$;
for $i = 1$ *to* n **do**
  if v_i *is not* X*-fixed* **then**
    $X(v_i) \leftarrow \mathsf{MarginSample}(\Phi, X, v_i)$;
$X_{V \setminus \Lambda(X)} \leftarrow \mathsf{RejectionSampling}(\Phi, X, V \setminus \Lambda(X))$;
return X;

The following invariant is satisfied in the **for** loop in Algorithm 1 (formally proved in Lemma V.3). The correctness of the MarginSample subroutine is guaranteed by this invariant.

Condition III.3 (invariant for MarginSample)**.** *The following holds for the input tuple (Φ, σ, v):*

- *$\Phi = (V, \mathcal{Q}, C)$ is a CSP formula, $\sigma \in \mathcal{Q}^*$ is a feasible partial assignment, and $v \in V$ is a variable;*

- *v is not σ-fixed and $\sigma(v) = \star$, and for all $u \in V$, $\sigma(u) \in Q_u \cup \{\star\}$;*
- *$\mathbb{P}[\neg c \mid \sigma] \le p'q$ for all $c \in C$.*

The correctness of Algorithm 1 follows from the correctness of MarginSample and RejectionSampling for sampling from the correct marginal distributions, which is formally proved in Theorem V.1.

In fact, the sampling in Algorithm 1 is *perfect*. It will only become approximate after the oracle in Assumption 2 realized by a Monte Carlo program that may bias the sampling.

B. The rejection sampling

We first introduce RejectionSampling, which is a standard procedure. Our rejection sampling takes advantages of simplification and decomposition of a CSP formula.

A simplification of $\Phi = (V, \mathcal{Q}, C)$ under partial assignment $\sigma \in \mathcal{Q}^*$, denoted by $\Phi^\sigma = (V^\sigma, \mathcal{Q}^\sigma, C^\sigma)$, is a new CSP formula such that $V^\sigma = V \setminus \Lambda(\sigma)$ and $\mathcal{Q}^\sigma = \mathcal{Q}_{V\setminus\Lambda(\sigma)}$, and the C^σ is obtained from C by:

1) removing all the constraints that have already been satisfied by σ;
2) for the remaining constraints, replacing the variables $v \in \Lambda(\sigma)$ with their values $\sigma(v)$.

It is easy to see that $\mu_{\Phi^\sigma} = \mu^\sigma_{V\setminus\Lambda(\sigma)}$ for the uniform distribution μ_{Φ^σ} over satisfying assignments of Φ^σ.

A CSP formula $\Phi = (V, \mathcal{Q}, C)$ can be naturally represented as a (multi-)hypergraph H_Φ, where each variable $v \in V$ corresponds to a vertex in H_Φ and each constraint $c \in C$ corresponds to a hyperedge $\mathsf{vbl}(c)$ in H_Φ. We slightly abuse the notation and write $H_\Phi = (V, C)$.

Let $H_i = (V_i, C_i)$ for $1 \le i \le K$ denote all $K \ge 1$ connected components in H_Φ, and $\Phi_i = (V_i, \mathcal{Q}_{V_i}, C_i)$ their formulas. Obviously $\Phi = \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_K$ with disjoint Φ_i, and μ_Φ is the product of all μ_{Φ_i}. Also μ_S on a subset of variables $S \subseteq V$ is determined only by those components with V_i intersecting S.

Our rejection sampling algorithm for drawing from a marginal distribution μ^σ_S is given in Algorithm 2. The correctness of this algorithm is folklore. We state without proof.

Algorithm 2: RejectionSampling(Φ, σ, S)

Input: a CSP formula $\Phi = (V, \mathcal{Q}, C)$, a feasible partial assignment $\sigma \in \mathcal{Q}^*$, and a subset $S \subseteq V \setminus \Lambda(\sigma)$ of unassigned variables in σ;
Output: an assignment $X_S \in \mathcal{Q}_S$ distributed as μ^σ_S;

find $\{H^\sigma_i = (V^\sigma_i, C^\sigma_i)\} \mid 1 \le i \le K, H^\sigma_i \cap S \ne \emptyset\}$, i.e., all the connected components in H_{Φ^σ} which intersect S where Φ^σ denotes the simplification of Φ under σ;
for $1 \le i \le K$ **do**
  repeat
    generate $X_{V^\sigma_i} \in \mathcal{Q}_{V^\sigma_i}$ uniformly and independently at random;
  until *all the constraints in C^σ_i are satisfied*;
return X_S where X is the concatenation of all $X_{V^\sigma_i}$;

Theorem III.4. *On any input (Φ, σ, S) as specified in Algorithm 2,* RejectionSampling *terminates with probability* 1*, and upon termination it returns an assignment $X_S \in \mathcal{Q}_S$ distributed as μ^σ_S.*

C. The marginal sampler

We now introduce the the core part of our sampling algorithm, the MarginSample subroutine. This procedure is a "marginal sampler": it can draw a random value for a variable $v \in V$ according to its marginal distribution μ^σ_v. Our marginal sampling algorithm is inspired by a recent novel sampling algorithm of Anand and Jerrum for infinite spin systems [21].

For each variable $v \in V$, we suppose that an arbitrary order is assumed over all values in Q_v; we use $q_v \triangleq |Q_v|$ to denote the domain size of v and fix the parameters θ_v, θ as $\theta_v \triangleq \frac{1}{q_v} - \eta - \zeta$ and $\theta \triangleq \frac{1}{q} - \eta - \zeta$ where

$$\begin{cases} \eta = (1 - \mathrm{e}p'q)^{-\Delta} - 1 \\ \zeta = \left(8\mathrm{e}qk\Delta^3\right)^{-1} \end{cases} \tag{6}$$

Note that $\zeta < \frac{1}{q} - \eta$ is guaranteed by the LLL condition in (3), and hence $\theta_v, \theta > 0$.

The MarginSample subroutine for drawing from a marginal distribution μ^σ_v is given in Algorithm 3.

Algorithm 3: MarginSample(Φ, σ, v)

Input: a CSP formula $\Phi = (V, \mathcal{Q}, C)$, a feasible partial assignment $\sigma \in \mathcal{Q}^*$, and a variable $v \in V$;
Output: a random $x \in Q_v$ distributed as μ^σ_v;

choose $r \in [0, 1)$ uniformly at random;
if $r < q_v \cdot \theta_v$ **then** `// r falls into the zone of local uniformity`
  return the $\lceil r/\theta_v \rceil$-th value in Q_v;
else `// r falls into the zone of indecision`
  return MarginOverflow$(\Phi, \sigma_{v\leftarrow\star}, v)$;

An invariant satisfied by Algorithm 3 guarantees that θ_v always lower bounds the marginal probability with

gap ζ. This is formally proved in Section IV by a "local uniformity" property (Corollary IV.3).

Proposition III.5. *Assuming Condition III.3 for the input* (Φ, σ, v)*, it holds that* $\min_{x \in Q_v} \mu_v^\sigma(x) \geq \theta_v + \zeta$.

Therefore, the function $\mathcal{D}$ constructed below is a well-defined distribution over Q_v:

$$\forall x \in Q_v, \qquad \mathcal{D}(x) \triangleq \frac{\mu_v^\sigma(x) - \theta_v}{1 - q_v \cdot \theta_v}. \tag{7}$$

Consider the following thought experiment. Partition $[0, 1)$ into $(q_v + 1)$ intervals $I_1, I_2 \ldots, I_{q_v}$ and I', where $I_i = [(i-1)\theta_v, i\theta_v)$ for $1 \leq i \leq q_v$ are of equal size θ_v, and $I' = [q_v\theta_v, 1)$ is the remaining part. We call $\bigcup_{i=1}^{q_v} I_i = [0, q_v\theta_v)$ the "zone of local uniformity" and $I' = [q_v\theta_v, 1)$ the "zone of indecision".

Drawing from μ_v^σ can then be simulated as: first drawing a uniform random $r \in [0, 1)$, if $r < q_v\theta_v$, i.e. it falls into the "zone of local uniformity", then returning the i-th value in Q_v if $r \in I_i$; if otherwise $r \in I'$, i.e. it falls into the "zone of indecision", then returning a random value drawn from the above $\mathcal{D}$. It is easy to verify that the generated sample is distributed as μ_v^σ. And this is exactly what Algorithm 3 is doing, assuming that the subroutine $\mathsf{MarginOverflow}(\Phi, \sigma_{v \leftarrow \star}, v)$ correctly draws from $\mathcal{D}$.

D. Recursive sampling for margin overflow

The goal of the MarginOverflow subroutine is to draw from the distribution $\mathcal{D}$ which is computed from the marginal distribution μ_v^σ as defined in (7).

Now suppose that we are given access to an oracle for drawing from μ_v^σ (such an oracle can be realized by $\mathsf{RejectionSampling}(\Phi, \sigma, \{v\})$ in Algorithm 2). Then, drawing from $\mathcal{D}$ that is a linear function of μ_v^σ, by accessing an oracle for drawing from μ_v^σ, can be resolved using existing approaches of *Bernoulli factory* [28]–[30].

This sounds silly because if such oracle for μ_v^σ were efficient we would have been using it to output a sample for μ_v^σ in the first place, which is exactly the reason why we ended up trying to draw from $\mathcal{D}$.

Nevertheless, such Bernoulli factory for sampling from $\mathcal{D}$ may serve as the basis of a recursion, where sufficiently many variables with enough "freedom" would have been sampled successfully in their zones of local uniformity during the recursion, and hence the remaining CSP formula would have been "factorized" into small connected components, in which case an oracle for μ_v^σ would be efficient to realize by the $\mathsf{RejectionSampling}(\Phi, \sigma, \{v\})$, and the Bernoulli factory for $\mathcal{D}$ could apply.

We define a class of variables that are candidates for sampling with priority in the recursion.

Definition III.6 ($\star$-influenced variables)**.** Let $\sigma \in Q^*$ be a partial assignment. Let $H^\sigma = H_{\Phi^\sigma} = (V^\sigma, C^\sigma)$ be the hypergraph for simplification Φ^σ. Let H^σ_{fix} be the sub-hypergraph of H^σ induced by $V^\sigma \cap V^\sigma_{\mathsf{fix}}$.

- Let $V^\sigma_\star \subseteq V^\sigma \cap V^\sigma_{\mathsf{fix}}$ be the set of vertices belong to the connected components in H^σ_{fix} that contain any v with $\sigma(v) = \star$.
- Let $V^\sigma_{\star\text{-inf}}$ be the vertex boundary of $V^\sigma_\star$ in H^σ where $V^\sigma_{\star\text{-inf}} \triangleq \left\{u \in V^\sigma \setminus V^\sigma_\star \mid \exists c \in C^\sigma, v \in V^\sigma_\star : u, v \in \mathsf{vbl}(c)\right\}$.
- Define $\mathsf{NextVar}(\sigma)$ as

$$\begin{cases} v_i \in V^\sigma_{\star\text{-inf}} \text{ with smallest } i & \text{if } V^\sigma_{\star\text{-inf}} \neq \emptyset, \\ \perp & \text{otherwise.} \end{cases} \tag{8}$$

With this construction of $\mathsf{NextVar}(\cdot)$, we describe the MarginOverflow subroutine in Algorithm 4.

Algorithm 4: $\mathsf{MarginOverflow}(\Phi, \sigma, v)$

Input: a CSP formula $\Phi = (V, Q, C)$, a feasible partial assignment $\sigma \in Q^*$, and a variable $v \in V$;
Output: a random $x \in Q_v$ distributed as $\mathcal{D} \triangleq \frac{1}{(1-q_v\cdot\theta_v)}(\mu_v^\sigma - \theta_v)$;

```
1  u ← NextVar(σ) where NextVar(σ) is defined as in (8);
2  if u ≠ ⊥ then
3      choose r ∈ [0, 1) uniformly at random;
4      if r < q_u · θ_u then      // r falls into the zone of local uniformity
5          σ(u) ← the ⌈r/θ_u⌉-th value in Q_u;
6      else                       // r falls into the zone of indecision
7          σ(u) ← MarginOverflow(Φ, σ_{u←⋆}, u);
       /* Lines 4 to 7 together draw σ(u) according to μ_u^{σ_{u←⋆}}  */
8      return MarginOverflow(Φ, σ, v);
9  else
10     sample a random x ∈ Q_v according to D ≜ 1/(1−q_v·θ_v) (μ_v^σ − θ_v) using a Bernoulli factory that accesses RejectionSampling(Φ, σ, {v}) as an oracle;
11     return x;
```

Basically, a variable u is a good candidate for sampling if it currently has enough "freedom" (since u is not σ-fixed) and can "influence" the variables that we are

trying to sample in the recursion (which are marked by $\star$) through a chain of constraints in the simplification of Φ under the current σ. Such variables are enumerated by $\mathsf{NextVar}(\sigma)$.

The idea of Algorithm 4 is simple. In order to draw from the overflow distribution $\mathcal{D}$ for a variable $v \in V$: if there is another candidate variable $u = \mathsf{NextVar}(\sigma)$ that still has enough freedom so its sampling might be easy, and also is relevant to the sampling at v or its ancestors, we try to sample u's marginal value first (hopefully within its zone of local uniformity and compensated by a recursive call for drawing from its margin overflow); and if there is no such candidate variable to sample first, we finally draw from $\mathcal{D}$ using the Bernoulli factory.

The following invariant is satisfied by the MarginOverflow subroutine called within the MarginSample subroutine and the MarginOverflow itself (formally proved in Lemma V.3).

Condition III.7 (invariant for MarginOverflow)**.** *The following holds for the input tuple* (Φ, σ, v)*:*

- $\Phi = (V, \mathcal{Q}, C)$ *is a CSP formula,* $\sigma \in \mathcal{Q}^*$ *is a feasible partial assignment, and* $v \in V$ *is a variable;*
- $\sigma(v) = \star$*;*
- $\mathbb{P}[\neg c \mid \sigma] \le p'q$ *for all* $c \in C$*.*

The following marginal lower bound follows from the "local uniformity" property (Corollary IV.3) in the same way as in Proposition III.5 and is formally proved in Section IV.

Proposition III.8. *Assuming Condition III.7 for the input* (Φ, σ, v)*, it holds that* $\min_{x \in Q_v} \mu_v^\sigma(x) \ge \theta_v + \zeta$ *and for* $u = \mathsf{NextVar}(\sigma)$*, if* $u \neq \perp$ *then it also holds that* $\min_{x \in Q_u} \mu_u^\sigma(x) \ge \theta_u + \zeta$*.*

The Bernoulli factory used in Algorithm 4 is achieved by a combination of existing constructions (to be specific, the Bernoulli factory for subtraction in [28], composed with the linear Bernoulli factory in [29] and the Bernoulli race in [30]), given access to an oracle for drawing from the marginal distribution μ_v^σ, which is realized by $\mathsf{RejectionSampling}(\Phi, \sigma, \{v\})$ in Algorithm 2. The description of the Bernoulli factory and the proof of its correctness and efficiency can be found in [20, Appendix A].

IV. Preliminary on Lovász Local Lemma

The following is the asymmetric Lovász Local Lemma stated in the context of CSP.

Theorem IV.1 ([1])**.** *Given a CSP formula* $\Phi = (V, \mathcal{Q}, C)$*, if there is a function* $x : C \to (0, 1)$ *such that:*

$$\forall c \in C: \quad \mathbb{P}[\neg c] \le x(c) \prod_{\substack{c' \in C \\ \mathsf{vbl}(c) \cap \mathsf{vbl}(c') \neq \emptyset}} (1 - x(c')), \quad (9)$$

then

$$\mathbb{P}\left[\bigwedge_{c \in C} c\right] \ge \prod_{c \in C} (1 - x(c)) > 0.$$

When the condition (9) is satisfied, the probability of any event in the uniform distribution μ over all satisfying assignments can be well approximated by the probability of the event in the product distribution. This was observed in [26]:

Theorem IV.2 ([26])**.** *Given a CSP formula* $\Phi = (V, \mathcal{Q}, C)$*, if* (9) *holds, then for any event* A *that is determined by the assignment on a subset of variables* $\mathsf{vbl}(A) \subseteq V$*,*

$$\Pr_{\mu}[A] = \mathbb{P}\left[A \mid \bigwedge_{c \in C} c\right] \le \mathbb{P}[A] \prod_{\substack{c \in C \\ \mathsf{vbl}(c) \cap \mathsf{vbl}(A) \neq \emptyset}} (1 - x(c))^{-1},$$

where μ *denotes the uniform distribution over all satisfying assignments of* Φ *and* $\mathbb{P}$ *denotes the law of the uniform product distribution over* $\mathcal{Q}$*.*

The following "local uniformity" property is a straightforward corollary to Theorem IV.2 by setting $x(c) = \mathrm{e}p$ for every $c \in C$ (and the lower bound is calculated by $\mu_v(x) = 1 - \sum_{y \in Q_v \setminus \{x\}} \mu_v(y)$).

Corollary IV.3 (local uniformity)**.** *Given a CSP formula* $\Phi = (V, \mathcal{Q}, C)$*, if* $\mathrm{e}p\Delta < 1$*, then for any variable* $v \in V$ *and any value* $x \in Q_v$*, it holds that*

$$\frac{1}{q_v} - \left((1 - \mathrm{e}p)^{-\Delta} - 1\right) \le \mu_v(x) \le \frac{1}{q_v} + \left((1 - \mathrm{e}p)^{-\Delta} - 1\right).$$

Recall p' defined in (5) and θ_v, ζ, η defined in (6). The following corollary implied by the "local uniformity" property simultaneously proves Proposition III.5 and Proposition III.8.

Corollary IV.4. *For any CSP formula* $\Phi = (V, \mathcal{Q}, C)$ *and any partial assignment* $\sigma \in \mathcal{Q}^*$*, if*

$$\forall c \in C, \quad \mathbb{P}[\neg c \mid \sigma] \le p'q,$$

then σ *is feasible, and for any* $v \in V \setminus \Lambda(\sigma)$ *and any* $x \in Q_v$*,*

$$\theta_v + \zeta \le \mu_v^\sigma(x) \le \theta_v + 2\eta + \zeta.$$

V. Correctness of Sampling

In this section, we prove the correctness of Algorithm 1. All theorems in this section assume the setting of parameters in (5) and (6), and the oracles in Assumption 1 and Assumption 2.

Theorem V.1. *On any input CSP formula $\Phi = (V, Q, C)$ satisfying* (3)*, Algorithm 1 terminates with probability* 1*, and returns a uniform random satisfying assignment of Φ upon termination.*

Remark V.2 (perfectness of sampling)**.** Note that the sampling in above theorem is *perfect*: Algorithm 1 returns a sample that is distributed exactly as the uniform distribution μ over all satisfying assignments of Φ. Later in Section VI, the oracle assumed in Assumption 2 will be realized by a Monte Carlo routine, which will further generalize the sampling algorithm to assume nothing beyond an evaluation oracle, in a price of a bounded bias introduced to the sampling.

The following lemma guarantees that the invariants in Condition III.3 and Condition III.7 are satisfied respectively by the inputs to Algorithm 3 and Algorithm 4.

Lemma V.3. *During the execution of Algorithm 1 on a CSP formula $\Phi = (V, Q, C)$ satisfying* (3)*:*

1) *whenever* MarginSample(Φ, X, v) *is called, Condition III.3 is satisfied by its input (Φ, X, v);*
2) *whenever* MarginOverflow(Φ, σ, v) *is called, Condition III.7 is satisfied by its input (Φ, σ, v).*

Before proving this lemma, we show that these invariants can already imply the correctness of MarginSample, which is critical for the correctness of the main sampling algorithm (Algorithm 1).

Theorem V.4. *The following holds for Algorithm 3 and Algorithm 4:*

1) *Assuming Condition III.3,* MarginSample(Φ, σ, v) *terminates with probability* 1*, and it returns a random value $x \in Q_v$ distributed as μ_v^σ upon termination.*
2) *Assuming Condition III.7,* MarginOverflow(Φ, σ, v) *terminates with probability* 1*, and upon termination it returns a random value $x \in Q_v$ distributed as the $\mathcal{D} \triangleq \frac{\mu_v^\sigma - \theta_v}{1 - q_v \cdot \theta_v}$ defined in* (7)*.*

The proof of Theorem V.4 is left to [20, Section 5].

We then verify the invariant conditions claimed in Lemma V.3. Before that, we formally define the sequence of partial assignments that evolve in Algorithm 1.

Definition V.5 (partial assignments in Algorithm 1)**.** Let $X^0, X^1, \ldots, X^n \in Q^*$ denote the sequence of partial assignments, where $X^0 = \star^V$ and for every $1 \le i \le n$, X^i is the partial assignments X in Algorithm 1 after the i-th iteration of the **for** loop in Lines 1-4.

Lemma V.6. *For the $X^0, X^1, \ldots, X^n$ in Definition V.5, it holds for all $0 \le i \le n$ that X^i is feasible and*

$$\forall c \in C, \qquad \mathbb{P}[\neg c \mid X^i] \le p'q.$$

Lemma V.7. *Assume Condition III.7 for (Φ, σ, v). For any $u \in V$, if u is not σ-fixed, then $(\Phi, \sigma_{u \leftarrow a}, v)$ and $(\Phi, \sigma_{u \leftarrow \star}, u)$ satisfy Condition III.7 for any $a \in Q_u \cup \{\star\}$.*

Both proofs of Lemma V.6 and Lemma V.7 can be found in [20, Section 5]. The invariant of Condition III.3 for MarginSample stated in Lemma V.3-(1) follows easily from Lemma V.6. The invariant of Condition III.7 for MarginOverflow stated in Lemma V.3-(2) follows from Lemmas V.3-(1) and V.7. Because by (Φ, σ, v) satisfies Condition III.3, we have $(\Phi, \sigma_{v \leftarrow \star}, v)$ satisfies Condition III.7. In addition, during the execution of MarginOverflow(Φ, τ, v) where $\tau = \sigma_{v \leftarrow \star}$, the algorithm will only change an input partial assignment τ to $\tau_{u \leftarrow a}$ for those vertices u that are not τ-fixed u and for $a \in Q_u \cup \{\star\}$. Lemma V.3 is proved.

Combining Lemma V.3 and Theorem V.4, we prove the correctness of MarginSample (Algorithm 3), assuming the LLL condition in (3) for the input CSP in the main algorithm (Algorithm 1).

The correctness of RejectionSampling (Algorithm 2) has already been established in Theorem III.4.

The correctness of the main sampling algorithm (Algorithm 1) then follows from the correctness of these two main subroutines, with details left to [20, Section 5].

VI. Efficiency of Sampling

In this section, we show the efficiency of Algorithm 1 under the LLL condition in (3).

Algorithm 1 assumes accesses to the following oracles for a class of constraints C, both of which receive as input a constraint $c \in C$ and a partial assignment $\sigma \in Q^*$ upon queries:

- Eval(c, σ): the evaluation oracle in Assumption 1, which decides whether $\mathbb{P}[c \mid \sigma] = 1$, that is, whether c is already satisfied by σ;
- Frozen(c, σ): the oracle for frozen decision in Assumption 2, which distinguishes between the two cases $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$, where p' is the threshold defined in (5), and answers arbitrarily and consistently if otherwise.

The complexity of our sampling algorithm is measured in terms of the queries to the two oracles Eval$(\cdot)$ and

Frozen(·), and the computation costs. We prove the following theorem.

Theorem VI.1. *Given as input a CSP formula* $\Phi = (V, \mathcal{Q}, C)$ *satisfying* (3)*, Algorithm 1 in expectation costs* $O(q^2k^2\Delta^{10}n)$ *queries to* Eval(·)*,* $O(k\Delta^7 n)$ *queries to* Frozen(·)*, and* $O(q^3k^3\Delta^{10}n)$ *in computation.*

Together with the correctness of Algorithm 1 stated in Theorem V.1, this proves the main theorem for perfect sampling (Theorem I.2), since any query to the oracle Frozen(·) can be resolved in $\mathrm{poly}(q,k)$ time assuming the FPTAS for violation probability in Theorem I.2.

Remark VI.2 (Monte Carlo realization of frozen decision). The oracle Frozen(·) can be realized probabilistically through the Monte Carlo method. Upon each query on a constraint c and a partial assignment σ, the two extreme cases $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$ can be distinguished with high probability $(1-\delta)$ by independently testing for $O(\frac{1}{p'}\log\frac{1}{\delta})$ times whether the constraint c is satisfied by a randomly generated assignment over $\mathsf{vbl}(c)$ consistent with σ. We further apply a memoization to guarantee the consistency of the oracle as required in Assumption 2. The resulting algorithm is called Algorithm 1' and is formally described in Section VI-C.

This Monte Carlo realization of the Frozen(·) oracle introduces a bounded bias to the resulting sample and turns the perfect sampler in Theorem VI.1 to an approximate sampler Algorithm 1', which no longer assumes any nontrivial machinery beyond evaluating constraints.

Theorem VI.3. *Given as input an* $\varepsilon \in (0,1)$ *and a CSP formula* Φ *satisfying* (3)*, Algorithm 1' in expectation costs* $O\left(q^2k^2\Delta^{11}n\log\left(\frac{\Delta n}{\varepsilon}\right)\right)$ *queries to* Eval(·) *and* $O\left(q^3k^3\Delta^{11}n\log\left(\frac{\Delta n}{\varepsilon}\right)\right)$ *in computation, and outputs within* ε *total variation distance from the output of Algorithm 1 on input* Φ*.*

Together with the correctness of Algorithm 1 stated in Theorem V.1, this proves the main theorem (Theorem I.1) of the paper.

A notation for complexity bound: Throughout the section, we adopt the following abstract notation for any complexity bound. A complexity bound is expressed as a formal bi-variate linear function:

$$t(\underline{x}, \underline{y}) = \alpha \cdot \underline{x} + \beta \cdot \underline{y} + \gamma, \tag{10}$$

where α represents the number of queries to Eval(·), β represents the number of queries to Frozen(·), and γ represents the computation costs.

A. Efficiency of MarginSample

We now prove the following upper bound on the expected running time of MarginSample (Algorithm 3), which is expressed in the form of (10)

Let $T_{\mathsf{MS}}(\Phi, \sigma, v)$ be the random variable that represents the complexity of MarginSample(Φ, σ, v) when Condition III.3 is satisfied by (Φ, σ, v).

Theorem VI.4. *Assume* $8ep\Delta^3 \le 0.99p'$ *and let* $X^0, X^1, \ldots, X^n$ *be the random sequence in Definition V.5. Assume the convention that* $T_{\mathsf{MS}}(\Phi, X^{t-1}, v_t) = 0$ *when* v_t *is* X^{t-1}*-fixed. For any* $1 \le t \le n$*, we have* $\mathbb{E}\left[T_{\mathsf{MS}}(\Phi, X^{t-1}, v_t)\right]$ *is no more than*

$$O\left(q^2k^2\Delta^{10}\right)\cdot\underline{x} + 480k\Delta^7\cdot\underline{y} + O\left(q^3k^3\Delta^{10}\right),$$

where expectation is taken over both X^{t-1} *and the randomness of* MarginSample *algorithm.*

The proof of Theorem VI.4 is left to [20, Section 6.5].

B. Efficiency of RejectionSampling

Here, we focus on $X = X^n$, the partial assignment obtained after all n iterations of the **for** loop in Line 2 of Algorithm 1, and passed to the RejectionSampling (Algorithm 2) as input, as formally defined in Definition V.5.

For partial assignment $\sigma \in \mathcal{Q}^*$, let $T_{\mathsf{Rej}}(\sigma)$ be the random variable representing the complexity of RejectionSampling$(\Phi, \sigma, V \setminus \Lambda(\sigma))$ in form of (10).

We show the following bound on the expectation of $T_{\mathsf{Rej}}(X)$ on the random partial assignment $X = X^n$, whose proof can be found in [20, Section 6.6].

Theorem VI.5. *Assume* $8ep\Delta^3 \le 0.99p'$*,where* p' *is fixed as in* (5)*.*

$$\mathbb{E}\left[T_{\mathsf{Rej}}(X)\right] \le 10\Delta^5 n\cdot\underline{x} + O(qk\Delta^5 n),$$

where expectation is taken over both X *and the randomness of* RejectionSampling *algorithm.*

C. Efficiency of the main sampling algorithm

The proof of Theorem VI.1 then follows from combining Theorem VI.4 and Theorem VI.5. The proof of Theorem VI.3 follows by setting up Algorithm 1' that does the followings. Set the parameters as:

$$\delta = 0.005 \quad \text{and} \quad N = \left\lceil \frac{\ln\left(4\times 10^3 k\Delta^7 n\varepsilon^{-2}\right)}{0.33p'\delta^2} \right\rceil.$$

Algorithm 1' simply executes Algorithm 1 on input Φ, with the oracle Frozen(·) replaced by the following explicitly implemented Monte Carlo subroutine:

- Given as input any constraint $c \in C$ and any partial assignment $\sigma \in Q^*$, repeat for N times:
  - generate an assignment $Y \in Q_{\mathsf{vbl}(c)}$ on $\mathsf{vbl}(c)$ uniformly at random consistent with σ;
  - check whether $c(Y) = \texttt{True}$ by querying $\mathsf{Eval}(c, Y)$;
- let Z be the number of times within N trials that $c(Y) = \texttt{False}$, and return $I[Z/N > 0.995p']$.

The detailed proof can be found in [20, Section 6.7].

Acknowledgment

We thank Weiming Feng and Kewen Wu for helpful discussions. Kun He wants to thank Xiaoming Sun for his support in doing this work. Yitong Yin wants to thank Vishesh Jain for pointing to the notion of robust CSPs.

References

[1] P. Erdős and L. Lovász, "Problems and results on 3-chromatic hypergraphs and some related questions," *Infinite and finite sets, volume 10 of Colloquia Mathematica Societatis János Bolyai*, pp. 609–628, 1975.

[2] J. B. Shearer, "On a problem of Spencer," *Combinatorica*, vol. 5, no. 3, pp. 241–245, 1985.

[3] R. A. Moser and G. Tardos, "A constructive proof of the general Lovász local lemma," *J. ACM*, vol. 57, no. 2, p. 11, 2010.

[4] H. Guo, M. Jerrum, and J. Liu, "Uniform sampling through the Lovász local lemma," *J. ACM*, vol. 66, no. 3, pp. Art. 18, 31, 2019.

[5] A. Moitra, "Approximate counting, the Lovász local lemma, and inference in graphical models," *J. ACM*, vol. 66, no. 2, pp. 10:1–10:25, 2019. (Conference version in *STOC*'17).

[6] H. Guo, C. Liao, P. Lu, and C. Zhang, "Counting hypergraph colorings in the local lemma regime," *SIAM Journal on Computing*, vol. 48, no. 4, pp. 1397–1424, 2019.

[7] A. Galanis, L. A. Goldberg, H. Guo, and K. Yang, "Counting solutions to random CNF formulas," in *ICALP*, vol. 168 of *LIPIcs*, pp. 53:1–53:14, 2020.

[8] D. G. Harris, "New bounds for the moser-tardos distribution," *Random Structures & Algorithms*, vol. 57, no. 1, pp. 97–131, 2020.

[9] W. Feng, H. Guo, Y. Yin, and C. Zhang, "Fast sampling and counting k-sat solutions in the local lemma regime," *Journal of the ACM (JACM)*, vol. 68, no. 6, pp. 1–42, 2021.

[10] W. Feng, K. He, and Y. Yin, "Sampling constraint satisfaction solutions in the local lemma regime," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1565–1578, 2021.

[11] V. Jain, H. T. Pham, and T. D. Vuong, "On the sampling lovász local lemma for atomic constraint satisfaction problems," *CoRR*, vol. abs/2102.08342, 2021.

[12] V. Jain, H. T. Pham, and T. D. 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*, pp. 173–183, IEEE, 2021.

[13] K. He, X. Sun, and K. Wu, "Perfect sampling for (atomic) lovász local lemma," *CoRR*, vol. abs/2107.03932, 2021.

[14] A. Galanis, H. Guo, and J. Wang, "Inapproximability of counting hypergraph colourings," *arXiv preprint arXiv:2107.05486*, 2021.

[15] W. Feng, H. Guo, and J. Wang, "Improved bounds for randomly colouring simple hypergraphs," *arXiv preprint arXiv:2202.05554*, 2022.

[16] H. Guo and M. Jerrum, "A polynomial-time approximation algorithm for all-terminal network reliability," *SIAM J. Comput.*, vol. 48, no. 3, pp. 964–978, 2019.

[17] H. Guo and K. He, "Tight bounds for popping algorithms," *Random Structures Algorithms*, vol. 57, no. 2, pp. 371–392, 2020.

[18] I. Bezáková, A. Galanis, L. A. Goldberg, H. Guo, and D. Štefankovič, "Approximation via correlation decay when strong spatial mixing fails," *SIAM J. Comput.*, vol. 48, no. 2, pp. 279–349, 2019.

[19] N. J. A. Harvey and J. Vondrák, "An algorithmic proof of the Lovász local lemma via resampling oracles," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 1327–1345, IEEE Computer Soc., Los Alamitos, CA, 2015.

[20] K. He, C. Wang, and Y. Yin, "Sampling lovász local lemma for general constraint satisfaction solutions in near-linear time." arXiv preprint arXiv:2204.01520, 2022.

[21] K. Anand and M. Jerrum, "Perfect sampling in infinite spin systems via strong spatial mixing," *CoRR*, vol. abs/2106.15992, 2021.

[22] J. A. Fill and M. Huber, "The randomness recycler: a new technique for perfect sampling," in *FOCS*, pp. 503–511, IEEE, 2000.

[23] W. Feng, N. K. Vishnoi, and Y. Yin, "Dynamic sampling from graphical models," in *STOC*, pp. 1070–1081, ACM, 2019.

[24] M. Jerrum, "Fundamentals of partial rejection sampling," *arXiv preprint arXiv:2106.07744*, 2021.

[25] W. Feng, H. Guo, and Y. Yin, "Perfect sampling from spatial mixing," *Random Structures and Algorithms*, 2022.

[26] B. Haeupler, B. Saha, and A. Srinivasan, "New constructive aspects of the Lovász local lemma," *J. ACM*, vol. 58, no. 6, p. 28, 2011. (Conference version in *FOCS*'10).

[27] J. Beck, "An algorithmic approach to the Lovász local lemma.," *Random Struct. Algorithms*, vol. 2, no. 4, pp. 343–365, 1991.

[28] c. Nacu and Y. Peres, "Fast simulation of new coins from old," *Ann. Appl. Probab.*, vol. 15, no. 1A, pp. 93–115, 2005.

[29] M. Huber, "Nearly optimal Bernoulli factories for linear functions," *Combin. Probab. Comput.*, vol. 25, no. 4, pp. 577–591, 2016.

[30] S. Dughmi, J. D. Hartline, R. Kleinberg, and R. Niazadeh, "Bernoulli factories and black-box reductions in mechanism design," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 158–169, ACM, New York, 2017.