



Perfect Simulation of Las Vegas Algorithms via Local Computation

Xinyu Fu  

State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory,
Nanjing University, China

Yonggang Jiang  

MPI-INF, Saarbrücken, Germany
Saarland University, Saarbrücken, Germany

Yitong Yin  

State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory,
Nanjing University, China

Abstract

The notion of Las Vegas algorithms was introduced by Babai (1979) and can be defined in two ways:

- In Babai’s original definition, a randomized algorithm is called Las Vegas if it has a finitely bounded running time and certifiable random failure.
- Another definition widely accepted today is that Las Vegas algorithms refer to zero-error randomized algorithms with random running times.

The equivalence between the two definitions is straightforward. Specifically, for randomized algorithms with certifiable failures, repeatedly running the algorithm until no failure is encountered allows for faithful simulation of the correct output when it executes successfully.

We show that a similar perfect simulation can also be achieved in distributed local computation. Specifically, in the LOCAL model, with a polylogarithmic overhead in time complexity, any Las Vegas algorithm with finitely bounded running time and locally certifiable failures can be converted to a zero error Las Vegas algorithm. This transformed algorithm faithfully reproduces the correct output of the original algorithm in successful executions. This is achieved by a reduction to a distributed sampling problem under the Lovász Local Lemma (LLL), where the objective is to sample from the joint distribution of random variables avoiding all bad events. We then design the first efficient algorithm to solve this sampling problem in the LOCAL model.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Las Vegas algorithms, perfect simulation, Lovász Local Lemma, sampling

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.63

Related Version *Full Version:* <https://arxiv.org/abs/2311.11679>

1 Introduction

The Las Vegas algorithm, introduced by Babai [2], stands as a cornerstone in the theory of computing, defining the **ZPP** class for decision problems efficiently solvable by Las Vegas algorithms. Beyond decision problems, Las Vegas algorithms are pivotal in tackling optimization [24, 6], searching [28, 31], or sampling [34, 16] problems.

Las Vegas algorithms can be defined in two distinct yet related ways. In Babai’s original work [2], Las Vegas algorithms are defined as randomized algorithms whose failures are *certifiable*:

- A Las Vegas algorithm produces the correct output or reports failure within a finite bounded time.



© Xinyu Fu, Yonggang Jiang, and Yitong Yin;
licensed under Creative Commons License CC-BY 4.0
17th Innovations in Theoretical Computer Science Conference (ITCS 2026).

Editor: Shubhangi Saraf; Article No. 63; pp. 63:1–63:22



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Another definition of the Las Vegas algorithm, widely accepted today and utilized in various contexts, e.g. in [27, 32, 29], defines Las Vegas algorithms as *zero-error* randomized algorithms:

- A Las Vegas algorithm may exhibit a random running time but always produces the correct output.

The equivalence between these two definitions is evident. Through truncation, we can transform a zero-error Las Vegas algorithm with random running time into a Las Vegas algorithm with bounded running time and certifiable failure. Conversely, restarting the algorithm upon failure allows us to convert a Las Vegas algorithm with certifiable failure into a zero-error Las Vegas algorithm. This strategy, retrying with independent random choice until success, defines a *rejection sampling* procedure that faithfully simulates the random output of the Las Vegas algorithms, provided that they return successfully without failure.

This strategy for perfectly simulating the Las Vegas algorithm relies on a global coordination machinery: Every part of the algorithm must be notified if a failure occurs anywhere. However, our focus is on exploring how this could be accomplished through local computations, without the need for global coordination.

The LOCAL model. Local computations are formally characterized by the LOCAL model [25, 33]. An instance consists of a network $G = (V, E)$, which is an undirected graph, and a vector $\mathbf{x} = (x_v)_{v \in V}$ of local inputs. Each node $v \in V$ receives x_v and $n = |V|$ as input and can access private random bits. Communications are synchronized and proceed in rounds. In each round, each node may perform arbitrary local computation based on all information collected so far and send messages of arbitrary sizes to all its neighbors. This gives a LOCAL algorithm. The time complexity is measured by the number of rounds that the algorithm spends until all nodes are terminated. A LOCAL algorithm is said to be a $t(n)$ -round LOCAL algorithm on a class of instances, if it always terminates within $t(n)$ rounds on every instance from that class, where n represents the number of nodes of the instance.

The (Babai's) Las Vegas algorithm can be defined in the LOCAL mode with *locally certifiable failures*. A $t(n)$ -round LOCAL algorithm is called *Las Vegas* if each node $v \in V$ returns a pair (Y_v, F_v) , where Y_v stands for the local output at v , and $F_v \in \{0, 1\}$ indicates whether the algorithm failed locally at v . The algorithm successfully returns if none of the nodes fails. Furthermore, it is guaranteed that $\sum_{v \in V} \mathbb{E}[F_v] < 1$. This notion of Las Vegas LOCAL algorithm was formulated in [14].

In this paper, we try to answer the following fundamental question:

Can we faithfully simulate the correct output avoiding all local failures via local computation?

Specifically, we wonder whether a fixed-round Las Vegas LOCAL algorithm with locally certifiable failures, can be converted into a zero-error Las Vegas LOCAL algorithm that produces the correct output $(Y_v)_{v \in V}$ conditioned on $\sum_{v \in V} F_v = 0$, where the distribution of the correct output is faithfully preserved. This can be viewed as the analog of rejection sampling, carried out without any form of global coordination.

In this paper, for the first time, we answer this question affirmatively. We prove the following result for the perfect simulation of Las Vegas algorithms via local computation.

► **Theorem 1** (main theorem, informal). *Any $t(n)$ -round Las Vegas LOCAL algorithm can be converted to a zero-error Las Vegas LOCAL algorithm, which terminates within $t(n) \cdot \text{polylog}(n)$ rounds with probability $1 - n^{-O(1)}$, and returns the output of the $t(n)$ -round Las Vegas LOCAL algorithm conditioned on no failure.*

In the above theorem, the output of the zero-error Las Vegas LOCAL algorithm is identically distributed as the output of the $t(n)$ -round Las Vegas LOCAL algorithm conditioned on that none of the nodes fails, i.e. the zero-error algorithm perfectly simulates a successful running of the algorithm that may locally fail.

To see how nontrivial this is, consider a weakened task: to generate an assignment of random bits so that under this random choice the algorithm terminates without failure. One may think of this as “solving” for the random bits under which the algorithm successfully returns, which is weaker than our goal, where the generated random bits are further required to follow the correct distribution. However, for local computation, just solving the feasible random bits without changing their distribution is already highly nontrivial.

In a seminal work [14], Ghaffari, Harris, and Kuhn gave a systematic approach to solve the good random bits under which a Las Vegas LOCAL algorithm successfully returns. Their derandomization based approach preserves the *support* of the distribution, hence was more suitable for the *distributed graph problems* for constructing feasible solutions on graphs. Specifically, polylog(n)-round reductions were established between the two types of Las Vegas LOCAL algorithms for such problems.

In contrast, the perfect simulation guaranteed in Theorem 1 preserves the distribution, therefore, the result can apply to problems beyond constructing feasible solutions, for example, the sampling problems.

Consider the *Gibbs distributions* defined on the network $G = (V, E)$. Each node v corresponds to a variable with finite domain Σ . Let \mathcal{F} be a class of *constraints*, where each $f \in \mathcal{F}$ is a nonnegative-valued function $f : \Sigma^{\text{vbl}(f)} \rightarrow \mathbb{R}_{\geq 0}$ defined on the variables in $\text{vbl}(f) \subseteq V$. This defines a Gibbs distribution μ over all assignments $\sigma \in \Sigma^V$ by:

$$\mu(\sigma) \propto \prod_{f \in \mathcal{F}} f(\sigma_{\text{vbl}(f)}).$$

Such Gibbs distribution μ is said to be *local*, if: (1) for any $f \in \mathcal{F}$, the diameter of $\text{vbl}(f)$ in graph G is bounded by a constant; and (2) for any partial assignment $\sigma \in \Sigma^\Lambda$ specified on $\Lambda \subseteq V$, if σ is locally feasible, i.e. if $f(\sigma_{\text{vbl}(f)}) > 0$ for all $f \in \mathcal{F}$ with $\text{vbl}(f) \subseteq \Lambda$, then σ is (globally) feasible, which means that σ can be extended to a feasible full assignment $\tau \in \Sigma^V$ such that $\tau_\Lambda = \sigma$ and $\mu(\tau) > 0$.¹

A Gibbs distribution μ is said to have *strong spatial mixing with exponential decay* if the discrepancy (measured in the total variation distance) between the marginal distributions μ_v^σ, μ_v^τ at any $v \in V$ given the respective feasible boundary conditions $\sigma, \tau \in \Sigma^\Lambda$ on $\Lambda \subseteq V$ that differ over an arbitrary $\Delta \subseteq \Lambda$, satisfies:

$$d_{\text{TV}}(\mu_v^\sigma, \mu_v^\tau) \leq |V|^{O(1)} \cdot \exp(-\Omega(\text{dist}_G(v, \Delta))).$$

The strong spatial mixing is a key property for sampling algorithms. Its implication for efficient sampling from general Gibbs distributions is a major open problem. In [13, Corollary 5.3], a polylog(n)-round Las Vegas LOCAL algorithm with bounded local failures was given, for perfect sampling from local Gibbs distributions that have strong spatial mixing with exponential decay. By Theorem 1, this immediately implies the following result for the perfect simulation of Gibbs distributions via local computation.

¹ In [13], this property of local feasibility implying global feasibility in Gibbs distribution was called “locally admissible”.

► **Corollary 2.** *For any class of local Gibbs distributions that have strong spatial mixing with exponential decay, there is a LOCAL algorithm for perfect sampling from the Gibbs distribution, which terminates within $\text{polylog}(n)$ rounds with probability $1 - n^{-\Omega(1)}$.*

► **Remark 3.** The computational tractability of sampling from general Gibbs distributions with strong spatial mixing remains unresolved, even in the sequential and centralized setting. Corollary 2 certainly does not resolve the computational complexity of sampling. This is because the LOCAL model does not impose any bounds on the computational costs, and generic reductions such as the one stated in Theorem 1 must apply to all LOCAL algorithms. Such a lack of limitation on computational power was indeed common in general (black-box) reductions in the LOCAL model and may be necessary. For instance, the derandomization of general LOCAL algorithms in [14] also relied on a local exhaustive search of random bits. Here, our objective is even more ambitious: to preserve the distribution of the random bits of a successful run.

On a positive note, if it is known that the correct output has strong spatial mixing, as in the case of Corollary 2, then the perfect simulation in Theorem 1 can be simplified (using Algorithm 3 instead of Algorithm 4), avoiding exhaustive enumerations. Consequently, the computational cost of the algorithm in Corollary 2 is dominated by approximately sampling from the marginal distributions μ_v^σ and estimating the marginal probabilities within an inverse-polynomial accuracy. Indeed, the non-trivial computation costs in both the sampling algorithm in [13] and our perfect simulation algorithm when there is a strong spatial mixing, are dominated by these tasks which are computationally equivalent to sampling.

In summary, the focus of Corollary 2 is the locality of Gibbs sampling. Our proof of it may also imply: locality of computation imposes no additional significant barrier to the task of Gibbs sampling with strong spatial mixing, provided the sampling problem is computationally tractable.

1.1 A sampling Lovász Local Lemma in the LOCAL model

The perfect simulation of Las Vegas LOCAL algorithms stated in Theorem 1 is achieved by resolving a more general problem, namely, generating a random sample avoiding all bad events. This problem is formulated as a natural sampling problem in the variable-framework of the Lovász Local Lemma.

An instance for the variable-framework Lovász Local Lemma (LLL) is given by $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$, where $\{X_i\}_{i \in U}$ is a set of mutually independent random variables, such that each X_i follows a distribution ν_i over a finite domain Σ_i ; and $\{A_v\}_{v \in V}$ is a set of *bad events*, such that for each $v \in V$, the occurrence of A_v is determined by the evaluation of $X_{\text{vbl}(v)} = (X_i)_{i \in \text{vbl}(v)}$, where $\text{vbl}(v) \subseteq U$ denotes the subset of variables on which A_v is defined. The LLL instance I defines a *dependency graph* $D = D_I = (V, E)$, such that each vertex $v \in V$ represents a bad event A_v and each $\{u, v\} \in E$ iff $\text{vbl}(v) \cap \text{vbl}(u) \neq \emptyset$.

An LLL instance I is said to be γ -*satisfiable*, if the probability avoiding all bad events is bounded as:

$$\Pr \left(\bigcap_{v \in V} \overline{A_v} \right) \geq \gamma. \quad (1)$$

The Lovász Local Lemma [8] states a sufficient condition on the dependency graph for γ to be positive.

A satisfiable LLL instance I gives rise to a natural probability distribution over satisfying assignments. Let $\mu = \mu_I$ denote the distribution of the random vector $\mathbf{X} = (X_i)_{i \in U}$ conditioning on that none of the bad events $\{A_v\}_{v \in V}$ occurs. Formally, denote by $\Omega =$

$\Omega_I \triangleq \{\sigma \in \bigotimes_{i \in U} \Sigma_i \mid \sigma \text{ avoids } A_v \text{ for all } v \in V\}$ the space of all satisfying assignments, and $\nu = \nu_I \triangleq \prod_{i \in U} \nu_i$ the product measure. Here, the symbol \otimes represents the Cartesian product over sets. Then

$$\forall \sigma \in \Omega, \quad \mu(\sigma) \triangleq \Pr(\mathbf{X} = \sigma \mid \mathbf{X} \in \Omega) = \frac{\nu(\sigma)}{\nu(\Omega)}. \quad (2)$$

This distribution $\mu = \mu_I$ of random satisfying assignment was referred to as the *LLL distribution* in [17, 18]. It is a Gibbs distribution defined by hard constraints.

The following defines a computational problem to generate a sample according to the distribution μ .

Sampling Satisfying Solution of Lovász Local Lemma

Input: a γ -satisfiable LLL instance I with dependency graph $D_I = (V, E)$;

Output: a random satisfying assignment $\mathbf{X}^* = (X_i^*)_{i \in U}$ distributed as μ_I .

When the problem is solved in the LOCAL model, the input is presented to the algorithm as follows:

1. The network G of the LOCAL model is just the dependency graph D_I .
2. Each node $v \in V$ receives as input the values of $n = |V|$ and γ , along with the definition of the bad event A_v , and the distributions ν_i of the random variables $\{X_i \mid i \in \text{vbl}(v)\}$ on which A_v is defined so that it can locally draw independent evaluations of the random variables $\{X_i \mid i \in \text{vbl}(v)\}$ or check the occurrence of A_v on such evaluation.

Our main technical result is an efficient LOCAL algorithm for sampling satisfying solution according to the LLL distribution, for any LLL instance that is not prohibitively hard to satisfy (i.e., γ is not too small). We call this result a “LOCAL sampling lemma” since it uses the local lemma framework to give a LOCAL sampling algorithm.

► **Theorem 4** (LOCAL sampling lemma). *There is a randomized LOCAL algorithm such that for any LLL instance I with n bad events, if I is γ -satisfiable, then the algorithm returns a random satisfying assignment drawn from μ_I , within $\tilde{O}\left(\log^6 n \cdot \log^4 \frac{1}{\gamma}\right)$ rounds in expectation, and within $\tilde{O}\left(\log^6 n \cdot \log^4 \frac{1}{\gamma} \cdot \log^6 \frac{1}{\epsilon}\right)$ rounds with probability at least $1 - \epsilon$ for any $0 < \epsilon < 1$.*

► **Remark 5.** The sampling variant of LLL has recently drawn considerable attention [35, 20, 21, 22, 11, 30], which aims to generate a satisfying solution distributed as μ_I . This is harder than just finding a satisfying solution. It requires a stronger LLL condition (with polynomials of dependency degree) to ensure that the sampling is tractable in polynomial time [3, 16, 18].

Alternatively, in our setting, we do not impose any LLL-like conditions, but instead only assume that the LLL instance is γ -satisfiable for some suitably large γ (e.g. $\gamma = 1/\text{poly}(n)$ or $\gamma = \Omega(1)$). This is because the LLL instance here arises from a Las Vegas algorithm which succeeds with probability γ . In centralized model, such assumption may trivialize the problem because a sample can be drawn using $O(1/\gamma)$ trials of rejection sampling. However, the sampling problem remains highly nontrivial for local computation.

1.2 Perfect simulation via LOCAL sampling lemma

Recall that for LOCAL algorithms, an instance $\mathcal{I} = (G, \mathbf{x})$ consists of a network $G = (V, E)$ and a vector $\mathbf{x} = (x_v)_{v \in V}$ specifying the local input x_v to each node $v \in V$. Let \mathfrak{C} be a class of instances. A $t(n)$ -round Las Vegas LOCAL algorithm with success probability $\gamma(n)$ on instance class \mathfrak{C} , is a randomized LOCAL algorithm such that on every instance $\mathcal{I} = (G, \mathbf{x}) \in \mathfrak{C}$, where G is a network with $n = |V|$ nodes, at every node $v \in V$ the algorithm terminates within $t(n)$ rounds and outputs a pair (Y_v, F_v) where $F_v \in \{0, 1\}$ indicates whether the algorithm failed locally at v , and the probability that the algorithm succeeds is

$$\Pr(\forall v \in V : F_v = 0) \geq \gamma(n).$$

Denote by $(\mathbf{Y}_{\mathcal{I}}, \mathbf{F}_{\mathcal{I}}) = \langle (Y_v)_{v \in V}, (F_v)_{v \in V} \rangle$ the output of the Las Vegas LOCAL algorithm on instance $\mathcal{I} = (G, \mathbf{x})$ with network $G = (V, E)$. The following theorem is a formal restatement of Theorem 1, which gives a zero-error Las Vegas LOCAL algorithm that perfectly simulates the good output $(\mathbf{Y}_{\mathcal{I}} \mid \mathbf{F}_{\mathcal{I}} = \mathbf{0})$ when there is no failure everywhere in the network.

► **Theorem 6** (formal restatement of Theorem 1). *Let $t : \mathbb{N} \rightarrow \mathbb{N}$ and $\gamma : \mathbb{N} \rightarrow [0, 1]$ be two functions. Let \mathcal{A} be a $t(n)$ -round Las Vegas LOCAL algorithm with success probability $\gamma(n)$ on instance class \mathfrak{C} . There is a LOCAL algorithm \mathcal{B} such that on every instance $\mathcal{I} \in \mathfrak{C}$ of n nodes, for any $0 < \epsilon < 1$,*

- \mathcal{B} terminates in $t(n) \cdot \tilde{O}\left(\log^6 n \cdot \log^4 \frac{1}{\gamma(n)} \cdot \log^6 \frac{1}{\epsilon}\right)$ rounds with probability at least $1 - \epsilon$;
- upon termination, \mathcal{B} returns an output $\mathbf{Y}_{\mathcal{I}}^{\mathcal{B}}$ that is identically distributed as $(\mathbf{Y}_{\mathcal{I}}^{\mathcal{A}} \mid \mathbf{F}_{\mathcal{I}}^{\mathcal{A}} = \mathbf{0})$, which stands for the output of \mathcal{A} on the same instance \mathcal{I} conditioned on that none of the nodes fails.

Proof. Let $\mathcal{I} = (G, \mathbf{x}) \in \mathfrak{C}$ be the instance of the LOCAL algorithm, where $G = (V, E)$ is a network with $n = |V|$ nodes and the vector $\mathbf{x} = (x_v)_{v \in V}$ specifies the local inputs. For each $v \in V$, let X_v denote the local random bits at node v used by algorithm \mathcal{A} .

Since \mathcal{A} is a $t(n)$ -round Las Vegas LOCAL algorithm, at any $v \in V$, the algorithm \mathcal{A} deterministically maps the inputs $\mathbf{x}_B = (x_u)_{u \in B}$ and the random bits $\mathbf{X}_B = (X_u)_{u \in B}$ within the $t(n)$ -ball $B = B_{t(n)}(v)$, to the local output $(Y_v^{\mathcal{A}}, F_v^{\mathcal{A}})$, where $F_v^{\mathcal{A}} \in \{0, 1\}$ indicates the failure at v . This defines a bad event A_v for every $v \in V$, on the random variables X_u for $u \in B_{t(n)}(v)$, i.e. $\text{vbl}(v) = B_{t(n)}(v)$, by

$$A_v : F_v^{\mathcal{A}}(\mathbf{X}_{\text{vbl}(v)}) = 1.$$

Together, this defines an LLL instance $I = (\{X_v\}_{v \in V}, \{A_v\}_{v \in V})$, which is $\gamma(n)$ -satisfiable because the probability that \mathcal{A} has no failure everywhere is at least $\gamma(n)$.

We simulate the sampling algorithm in Theorem 4 (which we call the *LLL sampler*) on this LLL instance I . Rather than executing it on the dependency graph D_I as in Theorem 4, here we simulate the LLL sampler on the network $G = (V, E)$, where each node $v \in V$ holds an independent random variable X_v and a bad event A_v . Note that any 1-round communication in the dependency graph D_I can be simulated by $O(t(n))$ -round communications in this network $G = (V, E)$. Also note that at each $v \in V$, the values of $t(n)$ and $\gamma(n)$ can be computed locally by knowing $n = |V|$ and enumerating all network instances $\mathcal{I} \in \mathfrak{C}$ with n nodes. The LLL sampler can thus be simulated with $O(t(n))$ -multiplicative overhead. In the end it outputs an $\mathbf{X}^* = (X_v^*)_{v \in V} \sim \mu_I$, which is identically distributed as $(\mathbf{X} \mid \mathbf{F}_{\mathcal{I}}^{\mathcal{A}} = \mathbf{0})$, i.e. the random bits used in the algorithm \mathcal{A} conditioned on no failure. The final output $\mathbf{Y}^* = (Y_v^*)_{v \in V}$ is computed by simulating \mathcal{A} within $t(n)$ locality deterministically using $\mathbf{X}^* = (X_v^*)_{v \in V}$ as random bits.

The LLL sampler in Theorem 4 is a Las Vegas algorithm with random terminations. Each node $v \in V$ can continue updating Y_v^* using the current random bits \mathbf{X}_B^* it has collected within its $t(n)$ -local neighborhood $B = B_{t(n)}(v)$. Once the LLL sampler for generating \mathbf{X}^* terminates at all nodes, the updating of \mathbf{Y}^* will stabilize within additional $t(n)$ rounds. And this final \mathbf{Y}^* is identically distributed as $(\mathbf{Y}_T^A \mid \mathbf{F}_T^A = \mathbf{0})$. This gives us the zero-error Las Vegas LOCAL algorithm \mathcal{B} as claimed in Theorem 6. ◀

1.3 Related work

The perfect simulation of Las Vegas algorithms is a fundamental problem. In the celebrated work of Luby, Sinclair, and Zuckerman [27], an optimal strategy was given for speeding up Las Vegas algorithms. Their approach was based on stochastic resetting, which requires global coordination and works for the Las Vegas algorithms with deterministic outputs, or the interruptible random outputs.

The distribution of satisfying solutions of the Lovász local lemma (LLL) has drawn much attention, e.g. in [16, 18]. Its perfect simulation was studied in [16, 20, 19, 9], where several key approaches for perfect sampling were applied, including partial rejection sampling (PRS) [16], “lazy depth-first” method of Anand and Jerrum (*a.k.a.* the AJ algorithm) [1], coupling from the past (CFTP) [34], and coupling towards the past (CTTP) [9].

The connection between LLL and distributed computing is well-established. The foundational work of Chung, Pettie, and Su [5] pioneered the application of distributed LLL for solving Locally Checkable Labeling (LCL) problems in the LOCAL model. Then, Chang and Pettie [4] proposed that any Las Vegas randomized LOCAL algorithm can be represented as an LLL instance. Recently, Davies-Peck [7] present a novel analysis that leads to improved constructive LLL algorithms with variable running times for the LOCAL model, further enriching this domain.

In the LOCAL model, Ghaffari, Harris and Kuhn [14] showed that for distributed graph problems, where the goal is to construct feasible graph configurations, the fixed-round Las Vegas algorithms and the zero-error Las Vegas algorithms are equivalent to polylogarithmic rounds. Their approach was based on a derandomization by conditional expectations, and hence was especially suitable for the tasks where the support of the output distribution, instead of the output distribution itself, is concerned, such as the searching problems for constructing feasible solutions.

The LOCAL algorithms and Gibbs distributions are intrinsically related. For example, the distributions of the random bits on which a fixed-round Las Vegas LOCAL algorithm successfully returns are Gibbs distributions, where the certifiers of local failures are the local constraints defining the Gibbs distribution. In [13], Feng and Yin gave a LOCAL sampler with local failures for the Gibbs distributions with strong spatial mixing by parallelizing the JVV sampler [23] using the network decomposition [26].

1.4 Technique overview

We provide an overview of the main sampling algorithm stated in Theorem 4. The design and analysis of this algorithm will be detailed in subsequent sections.

Recall that the input instance for the algorithm in Theorem 4 is an LLL instance $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ with $n = |V|$ bad events. We assume that I is γ -satisfiable.

Initially, a random assignment \mathbf{Y} is generated according to the product distribution ν , which can be achieved by each node independently drawing its own variable $Y_v \sim \nu_v$. If none of the bad events in $\{A_v\}_{v \in V}$ occurs on this \mathbf{Y} , then \mathbf{Y} is a correct sample that follows the

LLL distribution μ_I . However, in general, some bad events in $\{A_v\}_{v \in V}$ may occur. In this case, the crux of the sampling algorithm lies in how to *locally resample* the independently generated \mathbf{Y} to make it follow the correct joint distribution μ_I .

Locally resample via the Bayes filter. We first propose a resampling procedure under the idealized assumption that there is only one node $v \in V$ with its bad event A_v occurring on \mathbf{Y} . This algorithm serves as the principal subroutine in our main sampling algorithm. To locally resample the random assignment \mathbf{Y} around node v , a natural attempt would be to resample $Y_{\text{vbl}(v)}$ locally according to the marginal distribution induced by μ_I on $\text{vbl}(v)$ conditioned on the outer boundary $Y_{U \setminus \text{vbl}(v)}$. However, this naïve resampling would introduce a bias to the sample, because $Y_{U \setminus \text{vbl}(v)}$ does not follow the marginal distribution induced by μ_I on $U \setminus \text{vbl}(v)$, but rather follows the *conditional* marginal distribution on $U \setminus \text{vbl}(v)$ given $Y_{\text{vbl}(v)}$.

To rectify this, the idea of *Bayes filter* was introduced in [10]. In this approach, a (locally checkable) filter is constructed according to Bayes' law to cancel the bias introduced by the naïve resampling. This new resampling procedure guarantees the production of a correct \mathbf{Y} that follows the desired distribution μ_I upon termination. Moreover, the procedure terminates in finite steps if the distribution μ_I exhibits a strong enough decay of correlation property. The only problem is that such decay of correlation may not necessarily hold for general LLL instances.

De-correlate distant variables by LLL augmentation. The decay of correlation is a key property that is widely assumed by sampling algorithms. However, in our main applications, namely simulating Las Vegas algorithms via local computation, such decay of correlation may not always hold. Instead, we assume that the LLL instance is sufficiently satisfiable (which corresponds to a Las Vegas algorithm that succeeds with a sufficiently large probability).

With this new assumption, we show that it is possible to establish suitably strong correlation decay between far-apart regions of variables by introducing a new bad event on the boundary between these two regions. Intuitively, we show that strong correlation between distant regions can only persist due to some specific low-probability bad blocking assignments on the variables between them. We then de-correlate the regions by augmenting the LLL instance with a new locally constructed bad event that explicitly prohibits these bad blocking assignments. Crucially, because these bad blocking assignments must have small marginal probabilities, the new bad event itself is rare, preserving the overall satisfiability of the instance. A precise example illustrating this mechanism is provided in the full version of the paper for further insight. Furthermore, we show how to utilize this LLL augmentation in a carefully designed resampling procedure to guarantee both the correctness and efficiency of the resampling.

Resample in general case using the SLOCAL paradigm. Finally, to handle the general case where multiple bad events $v \in V$ may occur on \mathbf{Y} , we first cluster the bad events into small enough balls far apart using network decomposition. We then apply the resampling procedure to fix each ball one by one. This final step of sequentially resampling on balls is presented in the sequential local (SLOCAL) paradigm. We introduce the notion of SLOCAL-LV algorithm as an extension of the SLOCAL model, where SLOCAL-LV stands for the term sequential LOCAL Las Vegas. Finally, we show that this SLOCAL-LV algorithm can be simulated in the LOCAL model with bounded time complexity.

1.5 Organization of the paper

We begin with preliminaries in Section 2. Then, in Section 3, we illustrate the core ideas behind our main sampling algorithm (stated in Theorem 4) by analyzing a special case where only a single bad event occurs initially. The algorithm for the general case involving multiple bad events initially, along with the complete proofs, is deferred to the full version of this paper. We conclude with a discussion of our work in Section 4.

2 Preliminary

2.1 Graph and LLL notations

Let $G = (V, E)$ be an undirected graph. The following notations are used throughout.

- *Neighborhoods:* $N_G(v) \triangleq \{u \in V \mid \{u, v\} \in E\}$ and inclusive neighborhood $N_G^+(v) \triangleq N(v) \cup \{v\}$.
- *Distances:* $\text{dist}_G(u, v)$ represents the shortest path distance between u and v in G , and $\text{dist}_G(S, T) \triangleq \min_{u \in S, v \in T} \text{dist}_G(u, v)$. The diameter of $\Lambda \subseteq V$ in G is $\text{diam}_G(\Lambda) \triangleq \max_{u \in \Lambda, v \in \Lambda} \text{dist}_G(u, v)$.
- *Balls:* $B_r^G(v) \triangleq \{u \in V \mid \text{dist}_G(u, v) \leq r\}$ and $B_r^G(\Lambda) \triangleq \{u \in V \mid \text{dist}_G(u, \Lambda) \leq r\}$ for $\Lambda \subseteq V$.
- *Shells/Spheres:* $S_{[\ell, r]}^G(v) \triangleq B_r^G(v) \setminus B_{\ell-1}^G(v)$ and $S_{[\ell, r]}^G(\Lambda) \triangleq B_r^G(\Lambda) \setminus B_{\ell-1}^G(\Lambda)$ for $\Lambda \subseteq V$.

Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be an LLL instance with dependency graph D_I . The following defines a notion of rings of variables enclosing a nonempty subset $\Lambda \subseteq V$ of bad events, where $\text{vbl}(\Lambda) \triangleq \bigcup_{v \in \Lambda} \text{vbl}(v)$.

- *Rings:* $R_r^I(\Lambda) \triangleq \text{vbl}(B_r^{D_I}(\Lambda)) \setminus \text{vbl}(B_{r-1}^{D_I}(\Lambda))$ and in particular, $R_0^I(\Lambda) \triangleq \text{vbl}(\Lambda)$. Furthermore, we define

$$R_{[i, j]}^I(\Lambda) \triangleq \bigcup_{i \leq r \leq j} R_r^I(\Lambda). \quad (3)$$

We also define the rings of bad events intersected or contained by a ring of variables:

$$V_{[i, j]}^I(\Lambda) \triangleq \left\{ v \in V \mid \text{vbl}(v) \cap R_{[i, j]}^I(\Lambda) \neq \emptyset \right\}, \quad V_{(i, j)}^I(\Lambda) \triangleq \left\{ v \in V \mid \text{vbl}(v) \subseteq R_{[i, j]}^I(\Lambda) \right\}. \quad (4)$$

In all the above notations, we omit the graph G or the LLL instance I if they are clear in the context.

Inspired by the induced subgraph, we define the sub-instance of $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ induced by a subset $\Lambda \subseteq V$ of bad events:

$$I(\Lambda) \triangleq (\{X_i\}_{i \in \text{vbl}(\Lambda)}, \{A_v\}_{v \in \Lambda}),$$

where $I(\Lambda)$ is the LLL sub-instance induced by the bad events $\{A_v\}_{v \in \Lambda}$.

2.2 Marginal distribution

Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be a LLL instance, where each random variable X_i follows the distribution ν_i over domain Σ_i . For $\Lambda \subseteq U$, define $\Sigma_\Lambda \triangleq \bigotimes_{i \in \Lambda} \Sigma_i$ and $\nu_\Lambda \triangleq \prod_{i \in \Lambda} \nu_i$, and write $\nu = \nu_U$ and $\Sigma = \Sigma_U$.

For nonempty $\Lambda \subset U$ and $\tau \in \Sigma_\Lambda$, define

$$\Omega_\Lambda^\tau \triangleq \left\{ \sigma \in \Sigma \mid \sigma_\Lambda = \tau \text{ and } \sigma \text{ avoids bad events } A_v \text{ for all } v \in V \text{ s.t. } \text{vbl}(v) \not\subseteq \Lambda \right\}. \quad (5)$$

63:10 Perfect Simulation of Las Vegas Algorithms via Local Computation

For disjoint $S, T \subseteq U$, $\sigma \in \Sigma_S$ and $\tau \in \Sigma_T$, denote by $\sigma \wedge \tau$ the direct concatenation of σ and τ , that is, $\sigma \wedge \tau \in \Sigma_{S \cup T}$ satisfying $(\sigma \wedge \tau)_i = \sigma(i)$ for $i \in S$ and $(\sigma \wedge \tau)_i = \tau(i)$ for $i \in T$.

The following defines a notion of marginal distribution in the LLL instance.

► **Definition 7** (marginal distribution). For $\Lambda \subseteq U$, a $\tau \in \Sigma_\Lambda$ is said to be a feasible boundary condition if $\Omega^\tau \neq \emptyset$, where Ω^τ is defined in (5). Given $\Lambda \subset U$ and feasible boundary condition $\tau \in \Sigma_\Lambda$, for any nonempty $S \subseteq U \setminus \Lambda$, the marginal distribution on S induced by $\mu = \mu_I$ conditioned on τ , denoted by $\mu_S^\tau = \mu_{I,S}^\tau$, is defined as:

$$\forall \sigma \in \Sigma_S, \quad \mu_S^\tau(\sigma) \triangleq \Pr_{\mathbf{X} \sim \nu} (X_S = \sigma \mid \mathbf{X} \in \Omega^\tau).$$

2.3 Decay of correlation in LLL

We consider the following notion of correlation decay in the LLL instance.

► **Definition 8** (ϵ -correlated sets). A pair of disjoint $S, T \subset U$ with $S \cup T \neq U$, is said to be ϵ -correlated, if one of S, T is empty, or for any $\sigma_1, \sigma_2 \in \Sigma_S$ and $\tau_1, \tau_2 \in \Sigma_T$,

$$1/(1+\epsilon) \cdot \nu(\Omega^{\sigma_1 \wedge \tau_2}) \cdot \nu(\Omega^{\sigma_2 \wedge \tau_1}) \leq \nu(\Omega^{\sigma_1 \wedge \tau_1}) \cdot \nu(\Omega^{\sigma_2 \wedge \tau_2}) \leq (1+\epsilon) \cdot \nu(\Omega^{\sigma_1 \wedge \tau_2}) \cdot \nu(\Omega^{\sigma_2 \wedge \tau_1}).$$

To see that this indeed defines a decay of correlation, note that it is equivalent to the following property: For \mathbf{X} drawn according to the product distribution ν that avoids all bad events A_v satisfying $\text{vbl}(v) \not\subseteq S \cup T$,

$$\begin{aligned} & \Pr(X_S = \sigma_1 \wedge X_T = \tau_1) \cdot \Pr(X_S = \sigma_2 \wedge X_T = \tau_2) \\ & \leq (1+\epsilon) \cdot \Pr(X_S = \sigma_1 \wedge X_T = \tau_2) \cdot \Pr(X_S = \sigma_2 \wedge X_T = \tau_1). \end{aligned}$$

Recall that we want to bound the correlation between X_S and X_T in a random \mathbf{X} distributed as $\mu = \mu_I$. Here, Definition 8 is slightly different by ignoring the bad events A_v defined on the variables within $S \cup T$.

2.4 Las Vegas SLOCAL algorithm

Our main sampling algorithm is described in a sequential local (SLOCAL) paradigm. The SLOCAL model introduced by Ghaffari, Kuhn, and Maus [15] captures local computations where the inherent parallelism of distributed systems is intentionally suppressed, allowing nodes to be processed sequentially rather than concurrently. We extend this notion to the algorithms with random locality of computation.

SLOCAL-LV algorithms. Here, SLOCAL-LV stands for the term sequential LOCAL Las Vegas. An N -scan SLOCAL-LV algorithm runs on a network $G = (V, E)$ with a subset $A \subseteq V$ of *active* nodes. Each node $v \in V$ maintains a local memory state M_v , initially storing v 's local input, random bits, its ID, and neighbors' IDs. An arbitrary total order is assumed over V , such that the relative order between any $u, v \in V$ can be deduced from the contents of M_u and M_v .

The algorithm operates in $N \geq 1$ scans. Within each scan, the active nodes in A are processed one after another in the ordering. Upon each node $v \in A$ being processed, for $\ell = 0, 1, 2, \dots$, the node v tries to grow an ℓ -ball $B_\ell(v)$ and update the memory states M_u for all $u \in B_\ell(v)$ based on the information observed so far by v , until some stopping condition has been met by the information within the current ball $B_\ell(v)$. Finally, each $v \in V$ outputs a value based on its memory state M_v .

Compared to the standard (Monte Carlo) SLOCAL algorithm, whose locality is upper bounded by a fixed value, in SLOCAL-LV algorithm, the local neighborhoods are randomly constructed in a sequential and local fashion. The next theorem gives a simulation of SLOCAL-LV algorithms in the LOCAL model.

► **Proposition 9** (simulation of SLOCAL-LV in LOCAL). *Let \mathcal{A} be an N -scan SLOCAL-LV algorithm that assumes an arbitrary ordering of nodes. Then there is a randomized LOCAL algorithm \mathcal{B} , such that starting from the same initial memory states $\mathbf{M} = (M_v)_{v \in V}$, the algorithm \mathcal{B} terminates and returns the same output as \mathcal{A} within $O(|A| \cdot \max_{v \in A, j \in [N]} \ell_{v,j})$ rounds, where A is the set of active nodes and $\ell_{v,j}$ is the radius of the ball accessed by the active node v in the j th scan of algorithm \mathcal{A} , both fully determined by \mathbf{M} .*

Compared to the simulation of SLOCAL Monte Carlo algorithm in the LOCAL model proved in [15], which relies on the network decomposition to parallelize the SLOCAL procedure, Proposition 9 provides a rather straightforward simulation that does not parallelize the local computations. An advantage of such an easy simulation is that it does not require a worst-case complexity upper bound for all scan orders of nodes. This translation from SLOCAL-LV to LOCAL algorithm makes it more convenient to describe LOCAL algorithms where there are multiple randomly growing local neighborhoods interfering with each other.

A formal proof of Proposition 9 is given in the full version of the paper.

3 Exposition of the Algorithm: Special Case

In this section, we expose the key ideas of the main sampling algorithm stated in Theorem 4 within a special case. The detailed and formal construction of the algorithm will be presented in the next section.

As stated in Theorem 4, the algorithm deals with an LLL instance $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ with $n = |V|$ bad events. Assume that I is γ -satisfiable, and let D_I be its dependency graph.

Our sampling algorithm follows a natural two-step framework:

- Initially, the algorithm generates a random assignment \mathbf{Y} according to the product distribution ν . This can be easily achieved by having each node independently generate its own variables.
- If none of the bad events in $\{A_v\}_{v \in V}$ occurs, then \mathbf{Y} follows the correct distribution μ_I . Otherwise, if some bad events A_v have occurred on \mathbf{Y} , the algorithm locally applies some corrections to \mathbf{Y} to ensure that the corrected assignment \mathbf{Y}' follows the correct distribution μ_I .

Obviously, the nontrivial part of the algorithm is the above second step, in which the algorithm locally modifies a random assignment $\mathbf{Y} \sim \nu$ to a new assignment $\mathbf{Y}' \sim \mu_I$ when some bad events A_v occur on \mathbf{Y} .

To simplify our exposition, we start by making the following idealized assumption.

► **Assumption 10.** *Only one node $v \in V$ has its corresponding bad event A_v occurs on \mathbf{Y} .*

Next, we will expose the main idea of our sampling algorithm by explaining how to resolve the sampling problem under this idealized assumption. The rest of this section is organized as follows:

1. We begin by approaching the sampling problem with modest objectives. Our first goal is to sample with a bounded *expected* complexity under Assumption 10, while also assuming a *correlation decay* property. With these assumptions, we present a sampling algorithm. (Section 3.1)

2. The correlation decay assumed above may not always hold. To address this, we introduce an *augmentation* of the LLL instance, inducing desirable correlation decay by properly augmenting the LLL instance. This enables us to remove the correlation decay assumption from the earlier mentioned algorithm, ensuring correct sampling from μ_I despite augmenting the LLL instance I . (Section 3.2)
 3. Finally, we introduce a *recursive* sampling framework, which upgrades the aforementioned sampling algorithms with bounded expected complexity to algorithms with *exponentially convergent* running time. This proves Theorem 4 under Assumption 10. (Section 3.3)
- In the full version of the paper, we eliminate the need for Assumption 10 through the introduction of a new section that addresses the general case involving multiple bad events.

3.1 Warm-up: Sampling with idealized correlation decay

In this part, we assume the correlation decay property as formally defined in Definition 8. This idealized correlation decay property is assumed for exposition purpose, and reliance on it will be eliminated later.

► **Assumption 11.** *Any pair of disjoint $S, T \subseteq U$ are $\frac{1}{2n^3}$ -correlated.*

Let \mathbf{Y} be generated according to the product distribution ν . Under Assumption 10, there is only one node $v \in V$ having A_v occur on \mathbf{Y} . Let $S \triangleq \text{vbl}(v)$ and $T \triangleq U \setminus \text{vbl}(B_1(v))$. Notice that we have

$$Y_T \sim \mu_T^{Y_S}, \quad (6)$$

because \mathbf{Y} is distributed as ν and all bad events except A_v do not occur on \mathbf{Y} .

An attempt to correct resampling. Our goal is to locally fix the random assignment \mathbf{Y} around v to make it distributed as μ_I . A natural attempt is to resample evaluation of $Y_{U \setminus T}$ according to the correct marginal distribution $\mu_{U \setminus T}^{Y_T}$. This would produce a new assignment \mathbf{Y}' which is distributed as:

$$\begin{aligned} \forall \sigma \in \Sigma, \quad \Pr[\mathbf{Y}' = \sigma] &= \Pr[\mathbf{Y}'_T = \sigma_T] \cdot \Pr[\mathbf{Y}'_{U \setminus T} = \sigma_{U \setminus T} \mid \mathbf{Y}'_T = \sigma_T] \\ &= \mu_T^{Y_S}(\sigma_T) \cdot \mu_{U \setminus T}^{\sigma_T}(\sigma_{U \setminus T}); \end{aligned}$$

whereas, our goal is that each $\sigma \in \Sigma$ is sampled with probability $\mu_I(\sigma) = \mu_T(\sigma_T) \cdot \mu_{U \setminus T}^{\sigma_T}(\sigma_{U \setminus T})$.

To remedy this, we apply the *Bayes filters* introduced in [10]. A Bayes filter $\mathcal{F} = \mathcal{F}(\mathbf{Y}, S, T)$ is a trial whose success or failure is determined by \mathbf{Y}, S, T . The probability that \mathcal{F} succeeds satisfies

$$\Pr[\mathcal{F} \text{ succeeds} \mid \mathbf{Y} = \sigma] \propto \frac{\mu_T(\sigma_T)}{\mu_T^{Y_S}(\sigma_T)}, \quad (7)$$

where \propto are taken over all possible $\sigma \in \Sigma$ with $\mu_T^{Y_S}(\sigma_T) > 0$.

Now given a Bayes filter \mathcal{F} satisfying (7), we conduct an experiment of \mathcal{F} , and resample $Y_{U \setminus T} \sim \mu_{U \setminus T}^{Y_T}$ if \mathcal{F} succeeds. This will produce a $\mathbf{Y}' \sim \mu_I$ due to the Bayes law derived as follows:

$$\begin{aligned} \Pr[\mathbf{Y}' = \sigma \mid \mathcal{F} \text{ succeeds}] &= \frac{\Pr[\mathbf{Y}' = \sigma] \cdot \Pr[\mathcal{F} \text{ succeeds} \mid \mathbf{Y} = \sigma]}{\Pr[\mathcal{F} \text{ succeeds}]} \\ &\propto \left(\mu_T^{Y_S}(\sigma_T) \cdot \mu_{U \setminus T}^{\sigma_T}(\sigma_{U \setminus T}) \right) \cdot \left(\frac{\mu_T(\sigma_T)}{\mu_T^{Y_S}(\sigma_T)} \right) \propto \mu_I(\sigma). \end{aligned} \quad (8)$$

Otherwise, if \mathcal{F} fails, we trivially resample the entire $\mathbf{Y}' \sim \mu_I$, which still ensures the correctness of sampling but uses global information. Overall, the above procedure correctly produces a $\mathbf{Y}' \sim \mu_I$.

It remains to construct a good Bayes filter using local generation and succeeding with high probability.

Construction of the Bayes filter \mathcal{F} . Condition (7) of the Bayes filter can be expressed as

$$\begin{aligned} \Pr[\mathcal{F} \text{ succeeds} \mid \mathbf{Y} = \sigma] &\propto \frac{\mu_T(\sigma_T)}{\mu_T^{Y_S}(\sigma_T)} \\ \text{(the Bayes law)} &= \frac{\nu(\Omega^{\sigma_T}) \cdot \nu(\Omega^{Y_S})}{\nu(\Omega) \cdot \nu(\Omega^{Y_S \wedge \sigma_T})} \\ &\propto \frac{\nu(\Omega^{\sigma_T})}{\nu(\Omega^{Y_S \wedge \sigma_T})} \\ &\triangleq f(\sigma_T). \end{aligned} \tag{9}$$

Note that $f(\sigma_T) \triangleq \frac{\nu(\Omega^{\sigma_T})}{\nu(\Omega^{Y_S \wedge \sigma_T})}$ can be computed locally from $B_2(v)$. It is thus natural to define \mathcal{F} as

$$\Pr[\mathcal{F} \text{ succeeds}] = \frac{f(Y_T)}{\max f},$$

where $\max f$ denotes the maximum value of $f(\sigma_T)$ taken over all all possible $\sigma_T \in \Sigma_T$ with $\mu_T^{Y_S}(\sigma_T) > 0$.

It is obvious to see that for the Bayes filter \mathcal{F} constructed as above, an experiment of \mathcal{F} can be conducted and observed locally from $B_2(v)$. Furthermore, due to the correlation decay assumed in Assumption 11, \mathcal{F} succeeds with high probability. Let Σ'_S be the set of possible assignments on the set S of variables:

$$\Sigma'_S \triangleq \{\rho \in \Sigma_S \mid \rho \text{ avoids bad events } A_v \text{ for all } v \in V \text{ s.t. } \text{vbl}(v) \subseteq S\}.$$

Let τ^* be the assignment on T that achieve the maximum $f(\tau^*)$. It holds that

$$\begin{aligned} \Pr[\mathcal{F} \text{ succeeds} \mid \mathbf{Y} = \sigma] &= \frac{f(\sigma_T)}{f(\tau^*)} = \frac{\nu(\Omega^{\sigma_T}) \cdot \nu(\Omega^{Y_S \wedge \tau^*})}{\nu(\Omega^{\tau^*}) \cdot \nu(\Omega^{Y_S \wedge \sigma_T})} \\ &= \frac{\sum_{\rho \in \Sigma'_S} \nu(\Omega^{\rho \wedge \sigma_T}) \cdot \nu(\Omega^{Y_S \wedge \tau^*})}{\sum_{\rho \in \Sigma'_S} \nu(\Omega^{Y_S \wedge \sigma_T}) \cdot \nu(\Omega^{\rho \wedge \tau^*})} \geq 1 - \frac{1}{2n^3}, \end{aligned} \tag{10}$$

where the last inequality is derived from that S and T are $\frac{1}{2n^3}$ -correlated, guaranteed by Assumption 11.

This simple sampling algorithm under Assumption 10 and Assumption 11 is described in Algorithm 1. By (10) and the locality of the Bayes filter, the algorithm terminates within $O(1)$ rounds in expectation.

3.2 Sampling without correlation decay via LLL augmentation

The correlation decay asked by Assumption 11 may not always hold for general γ -satisfiable LLL instances. A key idea is then to properly “augment” the LLL instance by including new bad events to enforce desirable decay of correlation. This is highlighted by the following LLL augmentation lemma.

Recall the notions of balls $B(\cdot)$, shells $S_{[\cdot, \cdot]}(\cdot)$, and rings $R_{[\cdot, \cdot]}(\cdot)$ formally defined in Section 2.1.

■ **Algorithm 1** *Sampling-with-decay*($\mathbf{Y}; I, v$).

Input : LLL instance $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$, node $v \subseteq V$;
Data : assignment $\mathbf{Y} = (Y_i)_{i \in U}$ stored globally that can be updated by the algorithm;

- 1 define $S \triangleq \text{vbl}(v)$ and $T \triangleq U \setminus \text{vbl}(B_1(v))$;
- 2 **with probability** $\frac{f(Y_T)}{\max f}$, where f is defined as in (9) **do**
- 3 $\left[\begin{array}{l} \text{update } \mathbf{Y} \text{ by resampling } Y_{U \setminus T} \sim \mu_{U \setminus T}^{\mathbf{Y}}; \\ // \frac{f(Y_T)}{\max f} \text{ and } \mu_{U \setminus T}^{\mathbf{Y}} \text{ can be evaluated locally within } B_2(v). \end{array} \right.$
- 4 **else**
- 5 $\left[\begin{array}{l} \text{resample } \mathbf{Y} \sim \mu_I; \\ // \text{Evaluating } \mu_I \text{ requires global information.} \end{array} \right.$
- 6 **return;**

► **Lemma 12** (LLL augmentation). *Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be a LLL instance. Let $\epsilon, \gamma \in (0, 1)$, $\delta \in (0, \frac{\gamma}{2})$ and $\ell \geq \ell_0(\epsilon, \gamma, \delta)$ where $\ell_0(\epsilon, \gamma, \delta) = \tilde{O}(\log \frac{1}{\epsilon} \log \frac{1}{\gamma} \log \frac{1}{\delta})$. For any nonempty $\Lambda \subseteq V$, if the sub-instance $I(S_{[1, \ell]}(\Lambda))$ is γ -satisfiable, then a new bad event A_λ with $\lambda \notin V$ can be constructed such that:*

1. A_λ is defined on $\text{vbl}(\lambda) = R_{[1, \ell]}(\Lambda)$, and is constructed using local information on $B_{\ell+1}(\Lambda)$;
2. A_λ has probability at most δ on independent random variables $\{X_i\}_{i \in U}$, i.e. $\nu(A_\lambda) \leq \delta$;
3. the variable sets $S = \text{vbl}(\Lambda)$ and $T = U \setminus \text{vbl}(B_\ell(\Lambda))$ are ϵ -correlated after including A_λ into I .

► **Definition 13** (the augmenting event). *Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be a LLL instance. Let $\epsilon, \gamma \in (0, 1)$, $\delta \in (0, \frac{\gamma}{2})$, and $\ell = \ell_0(\epsilon, \gamma, \delta)$. Fix any nonempty $\Lambda \subseteq V$ that $I(S_{[1, \ell]}(\Lambda))$ is γ -satisfiable. We use $A_{\lambda(\Lambda, \epsilon, \gamma, \delta)}^I$ to denote the new bad event A_λ constructed in Lemma 12.*

► **Remark 14.** Lemma 12 constitutes a critical component of our sampling approach. Beyond the algorithmic implications, it provides new insights into correlation between variables constrained by local constraints.

This lemma aims to establish a decay of correlation between two distant regions $S = \text{vbl}(\Lambda) = R_0(\Lambda)$ and $T = U \setminus \text{vbl}(B_\ell(\Lambda)) = R_{[\ell, \infty]}(\Lambda)$. However, it only assumes that all local constraints sandwiched between S and T are collectively easy to satisfy. Merely knowing this fact does not prevent S and T from being strongly correlated. Remarkably, Lemma 12 shows that the only obstacle to achieving correlation decay between S and T in this case lies in rarely occurring “bad” assignments between them. Consequently, S and T can be effectively de-correlated by introducing a new locally defined bad event over the region between S and T to prohibit those bad assignments that cause significant correlation between them.

This argument will be formalized in the full version of the paper, where a formal restatement of Lemma 12 will be introduced, and the lemma will be rigorously proved.

Sampling using augmented LLL. Now we show how to utilize this LLL augmentation stated in Lemma 12 to get rid of Assumption 11 in Algorithm 1, while still assuming Assumption 10.

As before, let \mathbf{Y} be generated according to the product distribution ν ; and let $S \triangleq \text{vbl}(v)$ and $T \triangleq U \setminus \text{vbl}(B_1(v))$, where $v \in V$ is the only node (Assumption 10) with the bad event A_v occurring on \mathbf{Y} .

Our goal is to utilize Lemma 12 to induce the necessary decay of correlation required for sampling. However, augmenting LLL would inevitably alter the distribution μ_I . The saving grace is the following observation, suggesting that sampling may not be affected by certain local changes to the distribution.

► **Observation 15.** *The sampling in Algorithm 1 is correct and efficient as long as:*

- (correctness) Y_T follows the marginal distribution $\mu_{I,T}^{Y_S}$;
- (efficiency) S and T are $\frac{1}{2n^3}$ -correlated.

Using this observation, we can apply the LLL augmentation described in Lemma 12 to ensure correlation decay between S and T , while preserving the marginal distribution $\mu_T^{Y_S}$. Consequently, the resulting sampling process is both correct and efficient without relying on Assumption 11.

Note the new bad event $A_\lambda = A_{\lambda(\Lambda, \epsilon, \gamma, \delta)}^I$ in Definition 13, and define its complement $A_{\bar{\lambda}}$:

$$A_\lambda \triangleq A_{\lambda(\Lambda, \epsilon, \gamma, \delta)}^I, \text{ where } \lambda \notin V, \quad \text{and} \quad A_{\bar{\lambda}} \triangleq \overline{A_\lambda}, \text{ where } \bar{\lambda} \notin V. \quad (11)$$

Correspondingly, define the following two augmented LLL instances:

$$\hat{I} = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V \cup \{\lambda\}}) \quad \text{and} \quad \hat{I}' = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V \cup \{\bar{\lambda}\}}). \quad (12)$$

Let $0 < \zeta_0 < 1$ be a sufficiently small constant. We define the choice of parameter

$$(\epsilon_0, \gamma_0, \delta_0) \triangleq \left(\frac{1}{2n^3}, \frac{\gamma}{8}, \frac{\zeta_0 \gamma}{24n^3} \right). \quad (13)$$

In the following, let $A_\lambda, A_{\bar{\lambda}}, \hat{I}, \hat{I}'$ be defined as above on $\Lambda = \{v\}$ with parameter $(\epsilon_0, \gamma_0, \delta_0)$. Let $S = \text{vbl}(\Lambda)$ and $T = U \setminus \text{vbl}(B_\ell(\Lambda))$, where $\ell = \ell_0(\epsilon_0, \gamma_0, \delta_0) = \tilde{O}(\log^2 n \log^2 \frac{1}{\gamma})$.

By the same argument for (6), we still have $Y_T \sim \mu_{I,T}^{Y_S}$. Notice that \hat{I} is identical I except for the new bad event A_λ . Then, conditioned on $A_{\bar{\lambda}}$, we have $Y_T \sim \mu_{I,T}^{Y_S}$ in the augmented instance \hat{I} . By Lemma 12 on our choice of parameter, S and T are $\frac{1}{2n^3}$ -correlated in \hat{I} . Thus, a calling to *Sampling-with-decay*($\mathbf{Y}; \hat{I}, v$) (Algorithm 1) will produce a random assignment $\mathbf{Y} \sim \mu_{\hat{I}}$ within $O(1)$ rounds in expectation.

An idealized sampling algorithm. Then an idealized sampling procedure may proceed as follows. Our goal is to modify the input \mathbf{Y} to a new $\mathbf{Y} \sim \mu_I$. This is achieved differently depending on whether the new bad event A_λ occurs on \mathbf{Y} . If A_λ occurs on \mathbf{Y} (which happens with a small probability), we generate an entire new $\mathbf{Y} \sim \mu_I$ using global information. Otherwise, if A_λ does not occur on \mathbf{Y} , we can use the following cleverer approach to produce a $\mathbf{Y} \sim \mu_I$:

- With probability P , generate a $\mathbf{Y} \sim \mu_{\hat{I}}$, where P is defined as

$$P \triangleq \Pr_{\mathbf{X} \sim \mu_I} [\mathbf{X} \text{ avoids } A_\lambda]. \quad (14)$$

Since \mathbf{Y} avoids A_λ , this can be achieved by calling *Sampling-with-decay*($\mathbf{Y}; \hat{I}, v$) to produce $\mathbf{Y} \sim \mu_{\hat{I}}$.

- Otherwise, generate entire $\mathbf{Y} \sim \mu_{\hat{I}}$, using global information.

Overall, this correctly generates a $\mathbf{Y} \sim \mu_I = P \cdot \mu_{\hat{I}} + (1 - P) \cdot \mu_{\hat{I}}$. This procedure for sampling is idealized because it uses a value P as defined in (14) that is not easy to compute locally.

Bootstrapping the unknown threshold P . The probability P in (14) can be lower bounded. By Lemma 12, A_λ occurs with probability at most δ_0 . With the parameter specified in (13), we have

$$P = \frac{\nu(\Omega_{\hat{I}})}{\nu(\Omega_I)} = \frac{\nu(\Omega_I) - \nu(\Omega_{\hat{I}'})}{\nu(\Omega_I)} \geq 1 - \frac{\nu(A_\lambda)}{\nu(\Omega_I)} \geq 1 - \frac{\delta_0}{\gamma_0} \geq 1 - \frac{1}{2n^3}.$$

Therefore, when A_λ does not occur on \mathbf{Y} , the above idealized sampling procedure can be realized as: first call the subroutine *Sampling-with-decay*($\mathbf{Y}; \hat{I}, v$) defined in Algorithm 1 to produce $\mathbf{Y} \sim \mu_{\hat{I}}$, and then with probability $\frac{1}{2n^3}$, compute the value of P (which uses global information) and generate the entire $\mathbf{Y} \sim \mu_{\hat{I}'}$ with probability $2(1 - P) \cdot n^3$. The resulting algorithm is described in Algorithm 2.

■ **Algorithm 2** *Sampling-without-decay*($\mathbf{Y}; I, v$).

Input : LLL instance $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$, $v \subseteq V$;
Data : assignment $\mathbf{Y} = (Y_i)_{i \in U}$ stored globally that can be updated by the algorithm;
// Throughout the algorithm, $(\epsilon, \gamma, \delta)$ are defined as in (13).
// $A_\lambda, A_{\bar{\lambda}}, \hat{I}, \hat{I}'$ are defined as in (11) and (12) with $\Lambda = \{v\}$ and $(\epsilon_0, \gamma_0, \delta_0)$.
1 define $\ell \triangleq \ell_0(\epsilon, \gamma, \delta)$, $S \triangleq \text{vbl}(\Lambda)$ and $T \triangleq U \setminus \text{vbl}(B_\ell(\Lambda))$;
2 **if** \mathbf{Y} avoids A_λ **then**
3 call *Sampling-with-decay*($\mathbf{Y}; \hat{I}, v$), which is defined in Algorithm 1;
 // Now we have $\mathbf{Y} \sim \mu_{\hat{I}}$.
4 **with probability** $\frac{1}{2n^3}$ **do**
5 evaluate $P \triangleq \Pr_{\mathbf{X} \sim \mu_I}[\mathbf{X} \text{ avoids } A_\lambda]$ and resample $\mathbf{Y} \sim \mu_{\hat{I}'}$ with probability
 | $2(1 - P) \cdot n^3$;
 | *// Now we have $\mathbf{Y} \sim \mu_I$.*
6 **else**
7 resample $\mathbf{Y} \sim \mu_I$;
 | *// Evaluating P , μ_I , $\mu_{\hat{I}'}$ requires global information.*
8 **return**;

Algorithm 2 may use global information, particularly in Line 5 with probability $\frac{1}{2n^3}$, and in Line 7 when A_λ occurs on \mathbf{Y} , which happens with probability at most $\delta_0 \leq \frac{1}{n^3}$ due to Lemma 12.² As we discussed, the call to *Sampling-with-decay*($\mathbf{Y}; \hat{I}, v$) at Line 3 returns within $O(1)$ rounds in expectation. At last, the construction of the augmented instance \hat{I} takes $\ell = \tilde{O}(\log^2 n \log^2 \frac{1}{\gamma})$ rounds. Overall, Algorithm 2 returns a $\mathbf{Y} \sim \mu_I$ within $\tilde{O}(\log^2 n \log^2 \frac{1}{\gamma})$ rounds in expectation without relying on Assumption 11.

3.3 Recursive sampling with exponential convergence

Both Algorithm 1 and Algorithm 2 occasionally rely on global information with a probability of $O(1/n^3)$. While the expected time complexity remains well bounded, achieving exponential convergence, as outlined in Theorem 4, would be more desirable in randomized running time.

² Technically, Assumption 10 may bias the probability of A_λ on \mathbf{Y} . However, Algorithm 2 is solely used for exposition purposes, and such bias caused by Assumption 10 will no longer be an issue beyond in Algorithm 2.

We introduce a recursive sampling framework to achieve such exponential convergence in the time complexity. And more importantly, this framework is also crucial for finally getting rid of Assumption 10.

First, we adapt Algorithm 1 to the recursive sampling framework. This adaptation is direct and simple. Then, we achieve the recursive sampling without Assumption 11. Although part of this has already been embodied in Algorithm 2, there are still some substantial difficulties that need to be overcome.

Recursive sampling assuming correlation decay. At first, we still assume Assumption 11. The recursive sampling algorithm in this scenario is described in Algorithm 3. There are two differences between this algorithm and Algorithm 1: First, instead of taking just a node $v \in V$ as input in Algorithm 1, now Algorithm 3 takes a subset $\Lambda \subseteq V$ as input. Second and more importantly, the original step of sampling using global information in Algorithm 1, is now replaced by a recursive call in Algorithm 3.

■ **Algorithm 3** *RecursiveSampling-with-decay*($\mathbf{Y}; I, \Lambda$).

Input : LLL instance $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$, subset $\Lambda \subseteq V$;
Data : assignment $\mathbf{Y} = (Y_i)_{i \in U}$ stored globally that can be updated by the algorithm;

- 1 define $S \triangleq \text{vbl}(\Lambda)$ and $T \triangleq U \setminus \text{vbl}(B_1(\Lambda))$;
- 2 **with probability** $\frac{f(\mathbf{Y}_T)}{\max f}$, where f is defined as in (9) **do**
- 3 update \mathbf{Y} by resampling $Y_{U \setminus T} \sim \mu_{U \setminus T}^{\mathbf{Y}_T}$;
 // $\frac{f(\mathbf{Y}_T)}{\max f}$ and $\mu_{U \setminus T}^{\mathbf{Y}_T}$ can be evaluated locally within $B_2(\Lambda)$.
- 4 **else**
- 5 *RecursiveSampling-with-decay*($\mathbf{Y}; I, B_2(\Lambda)$);
 // Recursively sample $\mathbf{Y} \sim \mu_I$.
- 6 **return;**

The correctness of recursive sampling relies on the following *conditional Gibbs* property, which is adapted from the same property introduced in [12, 10] for Gibbs distributions.

► **Definition 16** (conditional Gibbs). *Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be a LLL instance. The random pair $(\mathbf{Y}, \mathcal{R})$, where $\mathbf{Y} \in \Sigma$ is an assignment and $\mathcal{R} \subseteq V$ is a subset of events, is said to satisfy conditional Gibbs property on instance I , if for any $R \subseteq V$ and $\sigma \in \Sigma_{\text{vbl}(\Lambda)}$ with $\Pr[\mathcal{R} = R \wedge Y_{\text{vbl}(R)} = \sigma] > 0$, conditioned on $\mathcal{R} = R \wedge Y_{\text{vbl}(R)} = \sigma$, it holds that $Y_{U \setminus \text{vbl}(R)} \sim \mu_{I, U \setminus \text{vbl}(R)}^\sigma$.*

Intuitively, the random pair $(\mathbf{Y}, \mathcal{R})$ represents a “partially correct” sample \mathbf{Y} with its problematic part covered by \mathcal{R} . The conditional Gibbs property guarantees that except for this problematic part, the sample is always distributed correctly. This gives a key invariant condition for the correctness of recursive sampling.

The following is easy to verify for \mathbf{Y} satisfying Assumption 10 with $\Lambda = \{v\}$.

► **Condition 17.** (\mathbf{Y}, Λ) satisfies conditional Gibbs property on instance I .

The following can be verified routinely by a structural induction: As long as Condition 17 is satisfied by the input, Algorithm 3 terminates with probability 1 and returns a \mathbf{Y} that is distributed as μ_I . This guarantees the correctness of Algorithm 3 as a sampling algorithm.

To further bound the complexity of Algorithm 3, we need to assume Assumption 11. With such assumption, by the same analysis as in (10), the Bayes filter in Line 2 succeeds with probability at least $1 - \frac{1}{2n^3}$. Hence, for any $0 < \epsilon < 1$, Algorithm 3 terminates in $O(\log \frac{1}{\epsilon} / \log n)$ rounds with probability $1 - \epsilon$.

Recursive sampling with LLL augmentation. Our goal here is to adopt Algorithm 2 for sampling without the additional assumption about correlation decay, into the recursive sampling framework. The resulting algorithm will solve the problem in Theorem 4 under Assumption 10.

As in Algorithm 2, we need LLL augmentation. The following notion of *augmented conditional Gibbs property* is a variant of the conditional Gibbs property in Definition 16, tailored with LLL augmentation.

► **Definition 18** (augmented conditional Gibbs). *Let $\epsilon, \gamma \in (0, 1)$, $\delta \in (0, \frac{\gamma}{2})$, and $\ell = \ell_0(\epsilon, \gamma, \delta)$, where $\ell_0(\epsilon, \gamma, \delta) = \tilde{O}(\log \frac{1}{\epsilon} \log \frac{1}{\gamma} \log \frac{1}{\delta})$ is defined as in Lemma 12. Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be a LLL instance. The random pair $(\mathbf{Y}, \mathcal{R})$, where $\mathbf{Y} \in \Sigma$ and $\mathcal{R} \subseteq V$, is said to satisfy augmented conditional Gibbs property on instance I with parameter $(\epsilon, \gamma, \delta)$, if for any $R \subseteq V$ with $\Pr[\mathcal{R} = R] > 0$:*

1. *the sub-instance $I(S_{[1, \ell]}(R))$ is γ -satisfiable;*
2. *for $S \triangleq \text{vbl}(R)$, $T \triangleq U \setminus \text{vbl}(B_\ell(R))$, for any $\sigma \in \Sigma_{\text{vbl}(\Lambda)}$ with $\Pr[\mathcal{R} = R \wedge Y_{\text{vbl}(R)} = \sigma] > 0$, conditioned on that $\mathcal{R} = R \wedge Y_{\text{vbl}(R)} = \sigma$, Y_T follows the distribution $\mu_{I, T}^\sigma$, i.e.,*

$$\forall \tau \in \Sigma_T, \quad \Pr(Y_T = \tau \mid \mathcal{R} = R \wedge Y_S = \sigma) = \mu_{I, T}^\sigma(\tau),$$

where \hat{I} stands for the augmented LLL instance $\hat{I} \triangleq (\{X_i\}_{i \in U}, \{A_v\}_{v \in V} \cup \{A_\lambda^I(\epsilon, \gamma, \delta, R)\})$ with $A_\lambda^I(\epsilon, \gamma, \delta, R)$ as in Definition 13.

Recall the choice of parameter $(\epsilon_0, \gamma_0, \delta_0)$ in (13) and let $\ell \triangleq \ell_0(\epsilon_0, \gamma_0, \delta_0)$. Let $r \in \mathbb{N}$ be the minimal integer that \mathbf{Y} avoids the bad event $A_{\lambda(\epsilon_0, \gamma_0, \delta_0, B_{r \cdot \ell}(v))}^I$. The following can be routinely verified on the instance I , the random assignment \mathbf{Y} , $\Lambda = B_{r \cdot \ell}(v)$, along with the parameter $(\epsilon, \gamma, \delta, \alpha) = (\epsilon_0, \gamma_0, \delta_0, \gamma_0)$.

► **Condition 19.** *The following hold:*

- $0 < \epsilon \leq \frac{1}{2}$, $0 < \alpha \leq \gamma < 1$ and $0 < \delta < \zeta_0 \cdot \alpha$;
- the LLL instance I is α -satisfiable and the sub-instance $I(V \setminus \Lambda)$ is γ -satisfiable;
- (\mathbf{Y}, Λ) satisfies the augmented conditional Gibbs property on instance I with parameter $(\epsilon, \gamma, \delta)$.

Our goal is then to modify the random assignment \mathbf{Y} around the region Λ to make it follow the correct distribution μ_I , as long as Condition 19 is satisfied. Adopting the definitions of $A_\lambda, A_{\bar{\lambda}}, \hat{I}, \hat{I}'$ and P as in Equations (11), (12), and (14) with $\Lambda = B_{r \cdot \ell}(v)$ and $(\epsilon, \gamma, \delta) = (\epsilon_0, \gamma_0, \delta_0)$. By our choice of r , the bad event A_λ may never occur on \mathbf{Y} . Then, Algorithm 2 can be simplified to the case where \mathbf{Y} always avoids A_λ .

Recursive bootstrapping of marginal probabilities. A challenge, as we mentioned in Section 3.2, is that the true value of the marginal probability P as defined in (14) is hard to compute locally. This is now resolved by a more sophisticated bootstrapping of the threshold P , by recursively estimating it within a progressively more accurate interval $[L, R]$. The estimation is formally stated by the following lemma, which is proved in the full version of the paper. It shows that the threshold P can be estimated exponentially more accurately as the radius of local information grows.

► **Lemma 20.** *Let $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$ be a LLL instance. Let $0 < \epsilon < \frac{1}{2}$, $k \in \mathbb{N}^+$, $0 < \alpha_1 \leq \alpha_2 < 1$ and $\ell = \ell_0(\epsilon^k, \alpha_2, \alpha_1 \cdot \epsilon^k)$, where the function ℓ_0 is as defined in Lemma 12. For any nonempty $\Lambda \subseteq V$ and an arbitrary event A_Λ defined on the variables in $\text{vbl}(\Lambda)$, assuming that I is α_1 -satisfiable and $I(V \setminus \Lambda)$ is α_2 -satisfiable, there is a $\hat{P} \in (0, 1)$ determined only by Λ , A_Λ , and $I(B_{\ell+1}(\Lambda))$, such that*

$$P \triangleq \Pr_{\mathbf{X} \sim \mu_I} [\mathbf{X} \text{ avoids } A_\Lambda] \in [\hat{P} - 2\epsilon^k, \hat{P} + 2\epsilon^k].$$

The high-level strategy for sampling can be outlined as follows. Given the current estimate interval $[L, R]$ of the threshold $P \in [L, R]$, as provided by Lemma 20, the algorithm operates as follows:

- With probability L , the algorithm is determined to enter the branch for sampling $\mathbf{Y} \sim \mu_{\hat{I}}$. This can be resolved efficiently as in Algorithm 3, because of the correlation decay in the augmented instance \hat{I} .
- With probability $1 - R$, the algorithm is determined to enter the branch for sampling $\mathbf{Y} \sim \mu_{\hat{I}'}$. This can be resolved recursively with an appropriately enlarged neighborhood containing $\text{vbl}(\Lambda)$, ensuring that Condition 19 is still satisfied invariantly.
- Otherwise, the algorithm enters the so-called “zone of indecision”, where it enlarges the local radius to gather more information in order to obtain a more accurate estimate $[L, R]$ of the threshold P .

Overall, the above procedure generates a random assignment $\mathbf{Y} \sim \mu_I$. Throughout the recursive calls, we ensure to maintain the invariant Condition 19. The procedure *RecursiveSampling*($\mathbf{Y}; I, \Lambda, \epsilon, \gamma, \delta, \alpha$) is detailed in Algorithm 4.

Our use of “zones of indecision” for recursive bootstrapping of marginal probabilities is inspired from the Anand-Jerrum algorithm introduced in [1] for solving perfect simulation of Gibbs distributions.

The correctness of Algorithm 4 follows from a structural induction. The complexity of Algorithm 4 is challenging to analyze due to the random recursion, nevertheless, we apply a potential method to bound it.

► **Lemma 21.** *Assume the input of *RecursiveSampling*($\mathbf{Y}; I, \Lambda, \epsilon, \gamma, \delta, \alpha$) satisfy Condition 19.*

1. *After *RecursiveSampling*($\mathbf{Y}; I, \Lambda, \epsilon, \gamma, \delta, \alpha$) returns, \mathbf{Y} follows the distribution μ_I .*
2. *For any $0 < \eta < 1$, with probability $1 - \eta$, *RecursiveSampling*($\mathbf{Y}; I, \Lambda, \epsilon, \gamma, \delta, \alpha$) accesses $I(B_r(\Lambda))$ and updates $Y_{\text{vbl}(B_r(\Lambda))}$, where*

$$r = \ell_0(\epsilon, \gamma, \delta) + \tilde{O} \left(\log \frac{1}{\gamma} \cdot \log^4 \frac{1}{\eta} + \log \frac{1}{\gamma} \cdot \log^2 \frac{1}{\eta} \cdot \log \frac{1}{\alpha} \right).$$

► **Remark 22.** It is important to note that Assumption 10 is not assumed in the statement of Lemma 21. Instead, the lemma holds as long as that Condition 19 is satisfied. The scenario described in Assumption 10 can in fact be incorporated into Condition 19 as a special case where $\Lambda = B_{r,\ell}(v)$.

4 Conclusion

In this paper, we show that for local computation, the successful output of any fixed-round Las Vegas computation, where failures are reported locally, can be perfectly simulated with polylogarithmic overheads.

As by-products, this gives perfect simulations, via efficient (polylogarithmic-round) local computation, for several fundamental classes of high-dimensional joint distributions, including:

Algorithm 4 *RecursiveSampling*($\mathbf{Y}; I, \Lambda, \epsilon, \gamma, \delta, \alpha$).

Input : LLL instance $I = (\{X_i\}_{i \in U}, \{A_v\}_{v \in V})$, subset $\Lambda \subseteq V$, parameter $(\epsilon, \gamma, \delta, \alpha)$;
Data : assignment $\mathbf{Y} = (Y_i)_{i \in U}$ stored globally that can be updated by the algorithm;

// Throughout the algorithm, $A_\lambda, A_{\hat{\lambda}}, \hat{I}, \hat{I}'$ are defined as in (11) and (12).

- 1 initialize $i \leftarrow 1$, and define $\ell_0 \triangleq \ell_0(\epsilon, \gamma, \delta)$;
- 2 draw $\rho \in [0, 1)$ uniformly at random;
- 3 **while true do**
- 4 $\ell_i \leftarrow \ell_0(\zeta_0^i, \gamma, \alpha \cdot \zeta_0^i)$;
- 5 compute the smallest interval $[L, R]$ containing $P \triangleq \Pr_{\mathbf{X} \sim \mu_I}[\mathbf{X} \text{ avoids } A_\lambda]$ based on $\Lambda, A_\lambda, I(B_{\ell_0 + \ell_i + 1}(\Lambda))$, assuming that I is α -satisfiable and $I(V \setminus B_{\ell_0 + \ell_i + 1}(\Lambda))$ is γ -satisfiable;
// By Lemma 20, such interval $[L, R]$ exists and satisfies $R - L \leq 4\zeta_0^i$.
- 6 **if** $\rho < L$ **then**
- 7 // Enters the zone $[0, L] \subseteq [0, P)$ for generating $\mathbf{Y} \sim \mu_{\hat{I}}$.
define $T \triangleq U \setminus \text{vbl}(B_{\ell_0}(\Lambda))$;
- 8 Define $f(\tau) \triangleq \frac{\nu(\Omega_{\hat{I}}^\tau)}{\nu(\Omega_{\hat{I}}^{Y_S \wedge \tau})}$ for all $\tau \in \Sigma_T$ with $\nu(\Omega_{\hat{I}}^{Y_S \wedge \tau}) > 0$, similar to (9);
- 9 **with probability** $\frac{f(Y_T)}{\max f}$ **do**
- 10 update \mathbf{Y} by redrawing $Y_{U \setminus T} \sim \mu_{\hat{I}, U \setminus T}^{Y_T}$;
// $\frac{f(Y_T)}{\max f}$ and $\mu_{\hat{I}, U \setminus T}^{Y_T}$ can be evaluated locally within $B_{\ell_0 + 1}(\Lambda)$.
- 11 **else**
- 12 initialize $r \leftarrow \ell_0 + 1$;
- 13 **while** \mathbf{Y} does not avoid the bad event $A_{\hat{\lambda}}^{\hat{I}}(\lambda(B_r(\Lambda)), 1/2, \gamma, \zeta_0 \alpha / 2)$ **do**
- 14 grow the ball: $r \leftarrow r + \ell_0\left(\frac{1}{2}, \gamma, \frac{\zeta_0 \alpha}{2}\right)$;
- 15 $\text{RecursiveSampling}(\mathbf{Y}; \hat{I}, B_r(\Lambda) \cup \{\lambda\}, \frac{1}{2}, \gamma, \frac{\zeta_0 \alpha}{2}, \frac{\alpha}{2})$;
- 16 **return**;
- 17 **else if** $\rho \geq R$ **then**
- 18 // Enters the zone $[R, 1] \subseteq [P, 1)$ for generating $\mathbf{Y} \sim \mu_{\hat{I}'}$.
initialize $s \leftarrow \ell_0 + 1$;
- 19 **while** \mathbf{Y} does not avoid the bad event $A_{\hat{\lambda}'}^{\hat{I}'}(\lambda(B_s(\Lambda)), 1/2, \gamma, \zeta_0 \alpha (1-R)/2)$ **do**
- 20 grow the ball: $s \leftarrow s + \ell_0\left(\frac{1}{2}, \gamma, \frac{\zeta_0 \alpha (1-R)}{2}\right)$;
- 21 $\text{RecursiveSampling}(\mathbf{Y}; \hat{I}', B_s(\Lambda) \cup \{\bar{\lambda}\}, \frac{1}{2}, \gamma, \frac{\zeta_0 \alpha (1-R)}{2}, \alpha(1-R))$;
- 22 **return**;
- 23 **else**
- 24 // Enters the zone $[L, R]$ of indecision.
enter the next iteration (and refine the estimation of P): $i \leftarrow i + 1$;

- random satisfying solutions of Lovász local lemma with non-negligible satisfiability;
- uniform locally checkable labelings (LCLs) with non-negligible feasibility;
- Gibbs distributions satisfying the strong spatial mixing with exponential decay.

We develop a novel approach for augmenting Lovász local lemma (LLL) instances by introducing locally-defined new bad events, to create the desirable decay of correlation. We also give a recursive local sampling procedure, which utilizes the correlation decay to accelerate the sampling process, and meanwhile still keeps the sampling result correct, without being biased by the change to the LLL instance.

At first glance, it almost looks like we are creating mixing conditions out of nothing. In particular, the approach seems to bypass the local-lemma-type conditions for sampling (e.g. the one assumed in [20]). But indeed, our augmentation of LLL instances relies on that the LLL instances are fairly satisfiable (or at least the separator between the regions that we want to de-correlate should be enough satisfiable). Sampling in such instances might already be tractable in conventional computation models, e.g. in polynomial-time Turing machine, but the problem remains highly nontrivial for local computation.

This new approach for perfect simulation works especially well in the models where the locality is the sole concern. A fundamental question is how this could be extended to the models where the computation and/or communication costs are also concerned, e.g. CONGEST model or PRAM model.

References

- 1 Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *SIAM Journal on Computing*, 51(4):1280–1295, 2022. doi:10.1137/21M1437433.
- 2 László Babai. Monte-carlo algorithms in graph isomorphism testing. *Université tde Montréal Technical Report, DMS*, (79-10), 1979.
- 3 Ivona Bezáková, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Daniel Stefankovic. Approximation via correlation decay when strong spatial mixing fails. *SIAM Journal on Computing*, 48(2):279–349, 2019. doi:10.1137/16M1083906.
- 4 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the local model. *SIAM J. Comput.*, 48(1):33–69, January 2019. doi:10.1137/17M1157957.
- 5 Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. In *Proceedings of the 33th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 134–143, 2014. doi:10.1145/2611462.2611465.
- 6 Kenneth L Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995. doi:10.1145/201019.201036.
- 7 Peter Davies-Peck. On the locality of the lovász local lemma. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC '25*, pages 1271–1282, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3717823.3718103.
- 8 Paul Erdos and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–627, 1975.
- 9 Weiming Feng, Heng Guo, Chunyang Wang, Jiaheng Wang, and Yitong Yin. Towards derandomising markov chain monte carlo. In *FOCS*, 2023.
- 10 Weiming Feng, Heng Guo, and Yitong Yin. Perfect sampling from spatial mixing. *Random Structures & Algorithms*, 61(4):678–709, 2022. doi:10.1002/RSA.21079.
- 11 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. doi:10.1145/3469832.
- 12 Weiming Feng, Nisheeth K Vishnoi, and Yitong Yin. Dynamic sampling from graphical models. *SIAM Journal on Computing*, 50(2):350–381, 2021. doi:10.1137/20M1315099.

- 13 Weiming Feng and Yitong Yin. On local distributed sampling and counting. In *PODC*, 2018.
- 14 Mohsen Ghaffari, David G Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *FOCS*, 2018.
- 15 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *STOC*, 2017.
- 16 Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *Journal of the ACM (JACM)*, 66(3):18:1–18:31, 2019. doi:10.1145/3310131.
- 17 Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *Journal of the ACM (JACM)*, 58(6):28, 2011.
- 18 David G Harris. New bounds for the Moser-Tardos distribution. *Random Structures & Algorithms*, 57(1):97–131, 2020. doi:10.1002/rsa.20914.
- 19 Kun He, Xiaoming Sun, and Kewen Wu. Perfect sampling for (atomic) Lovász local lemma. *arXiv preprint arXiv:2107.03932*, 2021.
- 20 Kun He, Chunyang Wang, and Yitong Yin. Sampling Lovász local lemma for general constraint satisfaction solutions in near-linear time. In *FOCS*, 2022.
- 21 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. On the sampling Lovász local lemma for atomic constraint satisfaction problems. *arXiv*, abs/2102.08342, 2021. arXiv:2107.03932.
- 22 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Towards the sampling Lovász local lemma. In *FOCS*, 2021.
- 23 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X.
- 24 Gil Kalai. A subexponential randomized simplex algorithm. In *STOC*, 1992.
- 25 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing (SICOMP)*, 21(1):193–201, 1992. doi:10.1137/0221015.
- 26 Nathan Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993. doi:10.1007/BF01303516.
- 27 Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47(4):173–180, 1993. doi:10.1016/0020-0190(93)90029-9.
- 28 D Mitchell, B Selman, and H Leveque. A new method for solving hard satisfiability problems. In *AAAI*, 1992.
- 29 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi:10.1017/CB09780511813603.
- 30 Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *Journal of the ACM (JACM)*, 66(2):1–25, 2019. doi:10.1145/3268930.
- 31 Robin A Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.
- 32 Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995. doi:10.1017/CB09780511814075.
- 33 David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.
- 34 James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms*, 9(1-2):223–252, 1996. doi:10.1002/(SICI)1098-2418(199608/09)9:1/2<3C223::AID-RSA14%3E3.0.CO;2-0.
- 35 Chunyang Wang and Yitong Yin. A sampling Lovász local lemma for large domain sizes. In *FOCS*, 2024.