LOCAL GIBBS SAMPLING BEYOND LOCAL UNIFORMITY

HONGYANG LIU, CHUNYANG WANG, YITONG YIN

Abstract. Local samplers are algorithms that generate random samples based on local queries to high-dimensional distributions, ensuring the samples follow the correct induced distributions while maintaining time complexity that scales locally with the query size. These samplers have broad applications, including deterministic approximate counting [HWY23, FGW+23], sampling from infinite or high-dimensional Gibbs distributions [AJ22, HWY22], and providing local access to large random objects [BRY20].

In this work, we present local samplers for Gibbs distributions of spin systems. Specifically, we design linear-time local samplers for:

- spin systems with soft constraints, including the first local sampler for near-critical Ising models;
- truly repulsive spin systems, represented by the first local sampler for uniform proper q-colorings, with $q = O(\Delta)$ colors on graphs with maximum degree Δ.

These local samplers are efficient beyond the "local uniformity" threshold, which imposes unconditional marginal lower bounds — a key assumption required by all prior local samplers. Our results show that, in general, local sampling is not significantly harder than global sampling for spin systems. As an application, our results also imply local algorithms for probabilistic inference in the same near-critical regimes.

1. Introduction

Spin systems, which originated in statistical physics, are stochastic models characterized by local interactions. These models have not only advanced our understanding of physical phenomena, such as phase transitions and criticality, but have also become central to machine learning and theoretical computer science, particularly in the study of sampling and inference problems in complex distributions.

Let $q \geq 2$ be an integer. A q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda} = (\lambda_v)_{v\in V}, \boldsymbol{A} = (A_e)_{e\in E})$ is defined on a finite graph $G = (V, E)$, where each vertex is associated with an *external field* $\lambda_v \in \mathbb{R}^q_{\geq 0}$, and each edge $e \in E$ is associated with an *interaction matrix* $A_e \in \mathbb{R}^{q\times q}_{\geq 0}$. A configuration $\sigma \in [q]^V$ assigns a spin state from $[q]$ to each vertex $v \in V$.

The *Gibbs distribution* $\mu = \mu^{\mathcal{S}}$ over all configurations $\sigma \in [q]^V$ is given by:

$$\mu(\sigma) \triangleq \frac{w(\sigma)}{Z}, \qquad w(\sigma) \triangleq \prod_{v\in V} \lambda_v(\sigma(v)) \prod_{e=(u,v)\in E} A_e(\sigma(u), \sigma(v)),$$

where the the normalizing factor $Z = \sum_{\sigma\in[q]^V} w(\sigma)$ is the *partition function*.

A central question in the study of spin systems is sampling from their associated Gibbs distributions. For well-known models such as the hardcore model and the Ising model, a critical threshold determined by the system's parameters has been identified, beyond which sampling from the Gibbs distribution becomes **NP**-hard [SS14, GŠV16]. Recent breakthroughs have shown that the Glauber dynamics, a widely-used Markov chain, mixes rapidly up to these critical thresholds for specific spin systems, including the hardcore model and the Ising model [ALO20, CLV20, CLV21, CFYZ21, AJK+22, CE22, CFYZ22]. These results provide a comprehensive characterization of the computational phase transition inherent in sampling from the Gibbs distributions of such spin systems.

1.1. Local sampling and local uniformity.

Recent research has increasingly focused on *local* sampling techniques for high-dimensional Gibbs distributions [AJ22, AGPP23, FGW+23]. Rather than directly drawing a global sample from the Gibbs distribution μ, such algorithms aim to answer on-demand *local* queries on a small subset of vertices $\Lambda \subseteq V$, and returns a sample approximately distributed according to the marginal distribution of μ induced on Λ, at a *local* cost that depends only on the size

State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, 210023, China. E-mail: liuhongyang@smail.nju.edu.cn, wcysai@smail.nju.edu.cn, yinyt@nju.edu.cn

of the query set $|\Lambda|$ (and not on the total size $|V|$ of the spin system). For a subset of vertices $\Lambda \subseteq V$, the marginal distribution μ_Λ is defined as:

$$\forall \tau \in [q]^\Lambda, \quad \mu_\Lambda(\tau) \triangleq \sum_{\sigma \in [q]^V : \sigma_\Lambda = \tau} \mu(\sigma).$$

Then, the local sampling problem is defined as follows:

The local sampling problem

Input: A spin system $\mathcal{S} = (G, \boldsymbol{\lambda}, \boldsymbol{A})$, where $G = (V, E)$, and a subset of vertices $\Lambda \subseteq V$;
Goal: Generate a sample $X \sim \mu_\Lambda$ in time that scales near-linearly in $|\Lambda|$.

Local sampling can, of course, solve global sampling by simply querying all vertices or using the auto-regressive sampler for self-reducible problems, as in [AJ22, HWY22]. Beyond this, these local samplers offer the ability to "scale down" the sampling process, addressing the challenge of providing local access to large random objects [BRY20], where sublinear computational costs are required for sublinear-size queries. For applications, efficient local samplers directly imply efficient algorithms for probabilistic inference for self-reducible problems, and can possibly lead to efficient approximate counting algorithms [HWY23, FGW+23, AFF+24, AFG+25].

However, existing local samplers for spin systems [AJ22, AGPP23, FGW+23] rely on the assumption of *unconditional marginal lower bounds*, also known as the "*local uniformity*" property. This assumption requires that the marginal distribution of each vertex remains nearly identical across all neighboring configurations, which may be excessively restrictive for various problems.

Consider, for example, the Ising model with edge activity $\beta > 0$ and an arbitrary external field. The Ising model, introduced by Ising and Lenz [Isi25], has been extensively studied in various fields. Formally, it is a 2-spin system with $A_e = \begin{pmatrix} \beta & 1 \\ 1 & \beta \end{pmatrix}$ at each edge $e \in E$ and arbitrary λ_v at each $v \in V$. Sampling from its Gibbs distribution can be achieved through Glauber dynamics, which mixes rapidly under the well-known "uniqueness condition":

$$\beta \in \left(\frac{\Delta - 2}{\Delta}, \frac{\Delta}{\Delta - 2} \right), \tag{1}$$

where Δ is the maximum degree of the underlying graph. Beyond this, either Glauber dynamics becomes torpidly mixing, or the sampling problem itself becomes intractable. In contrast, the requirement of unconditional marginal lower bounds for such models imposes a significantly stricter condition:

$$\beta \in \left(\left(\frac{\Delta - 1}{\Delta + 1} \right)^{1/\Delta}, \left(\frac{\Delta + 1}{\Delta - 1} \right)^{1/\Delta} \right) = \left(1 - \frac{1}{\Theta(\Delta^2)}, 1 + \frac{1}{\Theta(\Delta^2)} \right). \tag{2}$$

Much greater challenges arise in "truly repulsive" spin systems — most notably, in sampling uniform proper q-colorings. Given a graph $G = (V, E)$, a proper q-coloring is an assignment $\sigma : V \to [q]$ such that $\sigma(u) \neq \sigma(v)$ for all $(u, v) \in E$. This is one of the most extensively studied sampling problems. (Global) sampling algorithms have gradually lowered the tractability threshold for proper q-colorings to $q > 1.809\Delta$ [Vig99, CDM+19, CV25], where Δ is the maximum degree of the graph, while the uniqueness condition for proper q-colorings is given by $q \geq \Delta + 1$. On the other hand, the truly repulsive nature of proper colorings precludes local uniformity: the marginal probability of a color at a vertex can drop to zero when a neighbor is assigned that color, so any method that relies on an unconditional lower bound on the marginals fails. Consequently, to this day, no local sampler is known for uniform proper q-colorings.

This stark discrepancy raises a fundamental question: Do local samplers exist for such models in near-critical regimes? Or does local sampling inherently require a significantly more stringent critical condition compared to global sampling?

1.2. **Our results.** In this paper, we address the aforementioned open question by designing new linear-time local samplers for two fundamental classes of spin systems under near-critical conditions: models with soft constraints, including the Ising model, and repulsive models, represented by proper q-colorings, showing that local sampling remains feasible near the global threshold for these models.

Our main contributions, both the first of their kind, are:

- a local sampler for the Ising model in near-critical regimes;
- a local sampler for uniform proper q-colorings using $q = O(\Delta)$ colors.

Specifically, our local samplers assume the following natural access model for spin systems.

Assumption 1 (probe access)**.** Let $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$ be a q-spin system. We assume:

- For each $v \in V$, each neighbor $u \in N(v)$ can be accessed in $O(1)$ time.
- Each entry in every λ_v and A_e can be retrieved in $O(1)$ time.

These can be achieved by storing G as an adjacency list and representing λ_v and A_e as arrays.

1.2.1. *Local sampler for spin systems with soft constraints.* Our first general result provides a linear-time local sampler for q-spin systems that satisfy the following sufficient condition.

Condition 1.1 (tractable regime for spin systems with soft constraints)**.** *Let $\delta > 0$ be a parameter, and $\mathcal{S} = (G, \boldsymbol{\lambda}, \boldsymbol{A})$ be a q-spin system on a graph $G = (V, E)$ with maximum degree $\Delta \geq 1$. The following condition holds:*

- *(**Normalized**) All λ_v and A_e are normalized, i.e.,*
$$\forall v \in V, \quad \sum_{c \in [q]} \lambda_v(c) = 1 \quad \text{and} \quad \forall e \in E, \quad \max_{i,j \in [q]} A_e(i, j) = 1.$$
This normalization can be enforced without altering the Gibbs distribution.
- *(**Soft constraints**) For every edge $e = (u, v) \in E$ and every pair of spin values $c_1, c_2 \in [q]$,*
$$A_e(c_1, c_2) \geq C(\Delta, \delta) \triangleq 1 - \frac{1 - \delta}{2\Delta}.$$

The following theorem presents our local sampler for spin systems with soft constraints.

Theorem 1.2 (local sampler for spin systems with soft constraints)**.** *There exists an algorithm that, given access (as in Assumption 1) to a q-spin system $\mathcal{S} = (G, \boldsymbol{\lambda}, \boldsymbol{A})$ satisfying Condition 1.1, with Gibbs distribution $\mu = \mu^{\mathcal{S}}$, and given a subset of vertices $\Lambda \subseteq V$, outputs a perfect sample $X \sim \mu_\Lambda$ in expected time $O(\Delta \log q \cdot |\Lambda|)$.*

The local sampler in Theorem 1.2 is perfect and terminates in time linear in $|\Lambda|$ in expectation.

Next, we apply Theorem 1.2 to one of the most important spin systems with soft constraints: the Ising model. Recall the definition of the Ising model, which is a 2-spin system with an interaction matrix

$$A_e = \begin{pmatrix} \beta & 1 \\ 1 & \beta \end{pmatrix}$$

at each edge $e \in E$ and an arbitrary external field λ_v at each vertex $v \in V$. Note that this standard definition of the Ising model does not satisfy the normalization condition in Condition 1.1 when $\beta > 1$. However, we can transform such a (ferromagnetic) Ising model to satisfy this normalization condition, by using A_e/β as the interaction matrix, without altering its Gibbs distribution.

Applying Theorem 1.2 gives the following corollary, where Assumption 1 is implicitly assumed.

Corollary 1.3 (local Ising sampler)**.** *There exists an algorithm that, given a Ising model with Gibbs distribution μ on a graph $G = (V, E)$ with maximum degree $\Delta \geq 1$, arbitrary external fields λ_v at each $v \in V$, and edge activity β satisfying*

$$\beta \in \left(\frac{\Delta - 0.5}{\Delta}, \frac{\Delta}{\Delta - 0.5} \right), \tag{3}$$

and given a subset of vertices $\Lambda \subseteq V$, outputs a perfect sample $X \sim \mu_\Lambda$ in expected time $O(\Delta \cdot |\Lambda|)$.

The condition in (3) falls within the same regime of $\left(1-\Theta\left(\frac{1}{\Delta}\right), 1+\Theta\left(\frac{1}{\Delta}\right)\right)$ as the uniqueness condition in (1), substantially improving upon the local uniformity condition in (2).

1.2.2. *Local sampler for proper q-colorings.* Our next result establishes a linear-time local sampler for one of the most fundamental repulsive spin systems: the uniform proper q-coloring model. Given a graph $G=(V,E)$, a proper q-coloring is an assignment $\sigma: V \to [q]$ such that $\sigma(u) \neq \sigma(v)$ for every edge $(u,v) \in E$. This classical combinatorial model can be viewed as a q-spin system with truly repulsive hard constraints: the interaction matrix assigns zero weight to configurations where adjacent vertices share the same color (i.e., zeros on the diagonal) and unit weight otherwise (i.e., ones off the diagonal).

Uniform sampling of proper q-colorings has long been a central problem in the study of algorithmic sampling and counting. In a seminal work, Jerrum [Jer95] established optimal mixing of the Glauber dynamics for proper q-colorings under the condition $q > 2\Delta$, where Δ denotes the maximum degree of the graph. This threshold was later improved by Vigoda [Vig99], who showed that the flip dynamics mixes in $O(n \log n)$ time when $q > \frac{11}{6}\Delta$, which in turn implied an $O(n^2)$ mixing time for the standard Glauber dynamics. More recently, the threshold has been further lowered to $q > 1.809\Delta$ through a sequence of advances [CDM+19, CV25].

Despite this progress, efficient local samplers have remained elusive for proper q-colorings, primarily due to the lack of unconditional marginal bounds, as discussed earlier. Previously, as noted in [FGW+23, Section 8], a major obstacle to designing a local sampler for q-colorings has been overcoming the threshold $q = \Omega(\Delta^2)$, which corresponds to Huber's bounding chains [Hub98].

Our result overcomes this obstacle and provides the first local sampling algorithm for proper q-colorings in the near-critical regime $q = O(\Delta)$.

Theorem 1.4 (local sampler for proper q-colorings)**.** *There exists an algorithm that, given a graph $G=(V,E)$ with maximum degree $\Delta \geq 1$, an integer q satisfying*

$$q \geq 65\Delta, \tag{4}$$

and a subset of vertices $\Lambda \subseteq V$, outputs a perfect sample $X \sim \mu_\Lambda$, where μ denotes the uniform distribution over all proper q-colorings of G, in expected time $O\left(\Delta^2 q \cdot |\Lambda|\right)$.

Remark 1.5. If the local computation cost is relaxed to be sublinear in the size of the input graph, as in the local computation algorithm (LCA) model, a better bound of $q \geq 9\Delta$ was obtained in [BRY20]. Specifically, given any subset of vertices $\Lambda \subseteq V$, their algorithm outputs an approximate sample X from μ_Λ within ε total variation distance, in time $\tilde{O}((|V|/\varepsilon)^{0.68}\Delta\,|\Lambda|)$. Their approach is based on simulating distributed local Markov chains and is unlikely to yield a local sampler in the sense of Theorem 1.4.

1.2.3. *Local algorithms for probabilistic inference.* An important application of efficient local samplers lies in their connection to local counting for self-reducible problems, where they directly yield efficient algorithms for probabilistic inference. In the (Bayesian) probabilistic inference problem, the goal is typically to estimate how the marginal probability of a specific vertex changes under certain conditions or observations. This task is fundamental to many areas and is particularly well-motivated in machine learning and statistics, where inference plays a central role in prediction, decision-making, and learning [DL93, DL97].

For a partial configuration $\sigma \in [q]^\Lambda$ over a subset of vertices $\Lambda \subset V$ with $\mu_\Lambda(\sigma) > 0$, and a vertex $v \in V \setminus \Lambda$, the conditional marginal distribution μ_v^σ is defined as:

$$\forall c \in [q], \quad \mu_v^\sigma(c) \triangleq \frac{\sum_{\tau \in [q]^V : \tau_\Lambda = \sigma, \tau_v = c} \mu(\tau)}{\mu_\Lambda(\sigma)} = \mathop{\mathbf{Pr}}_{\tau \sim \mu}\left[\tau_v = c \mid \tau_\Lambda = \sigma\right].$$

Specifically, we obtain the following local algorithms for probabilistic inference in spin systems with soft constraints and for proper q-colorings.

Theorem 1.6 (probabilistic inference in spin systems with soft constraints)**.** *There exists an algorithm that, given access (as in Assumption 1) to a q-spin system $\mathcal{S} = (G, \boldsymbol{\lambda}, \boldsymbol{A})$ with Gibbs distribution $\mu = \mu^{\mathcal{S}}$*

satisfying Condition 1.1, and given a subset of vertices $\Lambda \subset V$, a partial configuration $\sigma \in [q]^\Lambda$ with $\mu_\Lambda(\sigma) > 0$, a vertex $v \in V \setminus \Lambda$, and parameters $\varepsilon, \delta \in (0,1)$, outputs an estimate $\hat{\mu}_v^\sigma$ such that

$$\mathbf{Pr}\left[\forall c \in [q] : (1-\varepsilon)\mu_v^\sigma(c) \le \hat{\mu}_v^\sigma(c) \le (1+\varepsilon)\mu_v^\sigma(c)\right] \ge 1-\delta,$$

in expected time $O\left(\varepsilon^{-2}\delta^{-1}\Delta q^2 \log q \cdot |\Lambda|\right)$.

Theorem 1.7 (probabilistic inference for proper q-colorings)**.** *There exists an algorithm that, given a graph $G = (V, E)$ with maximum degree Δ and $q \ge 65\Delta$, a partial proper q-coloring $\sigma \in [q]^\Lambda$ of a subset of vertices $\Lambda \subset V$, a vertex $v \in V \setminus \Lambda$, and parameters $\varepsilon, \delta \in (0,1)$, outputs an estimate $\hat{\mu}_v^\sigma$ such that*

$$\mathbf{Pr}\left[\forall c \in [q] : (1-\varepsilon)\mu_v^\sigma(c) \le \hat{\mu}_v^\sigma(c) \le (1+\varepsilon)\mu_v^\sigma(c)\right] \ge 1-\delta,$$

where μ denotes the uniform distribution over all proper q-colorings of G, in expected time $O\left(\varepsilon^{-2}\delta^{-1}\Delta^2 q^3 |\Lambda|\right)$.

1.3. **Technique overview.** Previous works on local samplers include [AJ22] and [FGW+23], both of which rely on unconditional marginal lower bounds, i.e., the local uniformity property. The work of [AJ22] introduced a novel local sampler called "*lazy depth-first search*" (a.k.a. the A-J algorithm). To sample the spin of a vertex according to its correct marginal distribution, the algorithm first draws a random spin according to the unconditional marginal lower bounds, and with the remaining probability, it recursively samples the spins of all neighboring vertices. The algorithm in [FGW+23] takes a different approach, employing a backward deduction framework for Markov chains, referred to as "*coupling towards the past*" (CTTP). Their method uses systematic Glauber dynamics combined with a grand coupling based on unconditional marginal lower bounds, allowing the spin of a vertex to be inferred via a convergent information-percolation process. Despite their differences, both approaches rely crucially on unconditional marginal lower bounds (implied by local uniformity) to prevent excessive backtracking and thus ensure the efficiency of the sampling procedure. For a more detailed comparison of the two algorithms, we refer the reader to [FGW+23, Section 1.2].

We introduce key innovations that eliminate the reliance on local uniformity for local sampling. While the high-level ideas are broadly applicable, we present our new local samplers within the coupling towards the past (CTTP) framework for local Markov chains. Unlike the original CTTP algorithm of [FGW+23], which depends on a default grand coupling derived from unconditional marginal lower bounds, our approach introduces several new ideas to adapt the grand coupling, enabling efficient local samplers without assuming unconditional marginal lower bounds.

To design local samplers for systems with soft constraints that lack local uniformity, we first generalize the CTTP framework via an abstract notion of *marginal sampling oracles*: procedures that sample from conditional marginal distributions given oracle access to the neighborhood configuration. This abstraction allows each implementation of a marginal sampling oracle to correspond to a specific simulation of Glauber dynamics — or more precisely, to a particular grand coupling of the chain. We then implement the marginal sampling oracle via rejection sampling, which leverages the softness of local constraints rather than relying on unconditional lower bounds on marginal probabilities. This yields efficient local samplers for spin systems with soft constraints beyond local uniformity.

The case of truly repulsive spin systems is much more challenging, as no marginal lower bound exists. Consequently, it is impossible to determine the outcome of an update at a given time with positive probability without additional information. To address this, we further extend the CTTP framework to:

- allow *partial information* (rather than the full outcome) to be resolved at a given timestamp;
- allow the grand coupling strategy at timestamp t to depend on earlier timestamps $t' < t$, introducing *adaptivity* into the grand couplings.

Leveraging these new ideas, we obtain the first local sampler for q-colorings with $q = O(\Delta)$ colors. We note that similar ideas have appeared in Coupling From The Past (CFTP), which yields (global) perfect samplers for q-colorings with $q = O(\Delta)$ colors [Hub98, BC20, JSS21]. Our technical contributions regarding q-colorings can thus be viewed as local counterparts of these CFTP-based global samplers.

For the analyses of our local samplers, correctness follows from the validity of the underlying grand coupling in each construction. For efficiency, we employ different approaches for spin systems with soft and hard constraints. In the case of spin systems with soft constraints, the algorithm's behavior is relatively straightforward: we demonstrate that it is stochastically dominated by a subcritical branching

process, which directly implies its efficiency. In contrast, the q-coloring case exhibits more intricate behavior, rendering the previous analysis inapplicable. To address this, we introduce a carefully designed *potential function* that reflects the state of the algorithm and drops to zero upon termination. We prove that this potential function evolves as a supermartingale with bounded differences throughout the execution of the algorithm, thereby establishing efficiency.

1.4. **Related topics.** Our local sampler is built upon the Coupling Towards The Past (CTTP) framework introduced in [FGW+23], which bears resemblance to the celebrated Coupling From The Past (CFTP) method by Propp and Wilson [PW96] for perfect sampling from Markov chains, as both approaches utilize the idea of grand coupling. (See Section 1.4 of [FGW+23] for a detailed comparison between the two frameworks.) Notably, the CTTP framework is more restrictive than CFTP: an efficient local sampler within the CTTP framework implies the existence of an efficient perfect sampler under CFTP, but the converse does not hold. This asymmetry arises because CTTP aims to produce not only a perfect sample but also a *local* one, whereas existing CFTP constructions typically rely on global knowledge in the analysis [Hub98, BC20, JSS21].

The backward deduction of Markov chain states in the CTTP framework also bears resemblance to the analysis of the cutoff phenomenon via the method of *information percolation* [LS16, LS17]. In particular, [LS17] shows that Glauber dynamics for the ferromagnetic Ising model exhibits a cutoff phenomenon in the near-critical regime $\beta \le 1 + \frac{1}{O(\Delta)}$. Despite these structural similarities, the goals of the two frameworks differ fundamentally: CTTP is designed for constructing local samplers, while the information percolation approach is aimed at analyzing mixing times. Furthermore, our technique for obtaining near-critical local samplers for the Ising model differs significantly from that of [LS17]: our grand coupling at each time step is constructed using rejection sampling, whereas theirs is based on discrete Fourier expansion. Additionally, the bounds we obtain are tighter than those in [LS17].

Our local sampler also falls into the category of providing *local access to large random objects* [BRY20, BPR22, MSW22]. Given a q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$ and public random bits, our algorithm can generate consistent samples X_Λ such that $X \sim \mu = \mu^{\mathcal{S}}$ upon multiple queries of any subset of vertices $\Lambda \subseteq V$, using only a local number of probes for public random bits.

1.5. **Organization.** The paper is organized as follows:

- In Section 2, we introduce the necessary preliminaries.
- In Section 3, we present a generalized CTTP framework, with an abstract notion of "marginal sampling oracles", and show how to utilize this abstraction to yield local samplers beyond local uniformity.
- In Section 4, we design a new marginal sampling oracle and apply it to obtain our local sampler for spin systems with soft constraints, proving Theorems 1.2 and 1.6.
- In Section 5, we further extend the CTTP framework to design a local sampler for q-colorings, proving Theorems 1.4 and 1.7.
- In Section 6, we summarize our contributions and outline potential future directions.

2. Preliminaries

2.1. **Markov chain basics.** Let Ω be a (finite) state space. Let $(X_t)_{t=1}^{\infty}$ be a Markov chain over the state space Ω with transition matrix P. A distribution π over Ω is a *stationary distribution* of P if $\pi = \pi P$. The Markov chain P is *irreducible* if for any $x, y \in \Omega$, there exists a timestamp t such that $P^t(x, y) > 0$. The Markov chain P is *aperiodic* if for any $x \in \Omega$, $\gcd\{t \mid P^t(x, x) > 0\} = 1$. If the Markov chain P is both irreducible and aperiodic, then it has a unique stationary distribution. The Markov chain P is *reversible* with respect to the distribution π if the following *detailed balance equation* holds.

$$\forall x, y \in \Omega, \quad \pi(x)P(x, y) = \pi(y)P(y, x),$$

which implies π is a stationary distribution of P. The *mixing time* of the Markov chain P is defined by

$$\forall \varepsilon > 0, \quad T(P, \varepsilon) \triangleq \max_{X_0 \in \Omega} \max\{t \mid d_{\mathrm{TV}}\left(P^t(X_0, \cdot), \pi\right) \le \varepsilon\},$$

where the *total variation distance* is defined by

$$d_{\mathrm{TV}}\left(P^t(X_0,\cdot),\pi\right) \triangleq \frac{1}{2}\sum_{y\in\Omega}\left|P^t(X_0,y)-\pi(y)\right|.$$

2.2. **Systematic scan Glauber dynamics.** The *systematic scan* Glauber dynamics is a generic way to sample from Gibbs distributions defined by spin systems. Given a q-spin system $\mathcal{S}=(G=(V,E),\boldsymbol{\lambda},\boldsymbol{A})$. Let $n=|V|$ and assume an arbitrary ordering $V=\{v_0,v_1,\ldots,v_{n-1}\}$, and let $T>0$ be some finite integer, the T-step systematic scan Glauber dynamics $\mathcal{P}(T)=\mathcal{P}^{\mathcal{S}}(T)$

(1) starts with an arbitrary configuration $X_{-T}\in[q]^V$ satisfying $\mu(X_{-T})>0$ at time $t=-T$;
(2) at each time $-T<t\le 0$,
   (a) picks the vertex $v=v_{i(t)}$ where $i(t)\triangleq t \mod n$, let $X_t(u)=X_{t-1}(u)$ for every $u\in V\setminus\{v\}$;
   (b) resample $X_t(v)$ from the marginal distribution $\mu_v^{X_{t-1}}$ on v conditioning on X_{t-1} where

$$\forall c\in[q],\quad \mu_v^{X_{t-1}}(c)=\mu_v^{X_{t-1}(N(v))}(c)\propto\lambda_v(c)\prod_{e=(u,v)\in E}A_e(\sigma(u),c).$$

Here, the first equality is due to the *conditional independence* property of Gibbs distributions.

The systematic scan Glauber dynamics is not a time-homogeneous Markov chain. However, by bundling n consecutive updates together, we can obtain a time-homogeneous Markov chain, which is aperiodic and reversible, which is sufficient for us to apply the following theorem.

Theorem 2.1 ([LPW17]). *Let μ be a distribution with support $\Omega\subseteq[q]^V$. Let $(X_t)_{t=0}^{\infty}$ denote the systematic scan Glauber dynamics on μ. If $(X_t)_{t=0}^{\infty}$ is irreducible over Ω, it holds that*

$$\forall X_0\in\Omega,\quad \lim_{t\to\infty} d_{\mathrm{TV}}\left(X_t,\mu\right)=0.$$

3. Coupling towards the past without marginal lower bounds

Our local sampler is based on the Coupling Towards The Past (CTTP) framework recently introduced in [FGW+23], which constructs a local sampler by evaluating multiple spin states from stationary Markov chains through backward deduction. Our framework generalizes the CTTP framework by replacing the default grand coupling, which uses unconditional marginal lower bounds, with grand couplings defined by arbitrary "marginal sampling oracles". This generalization allows us to design a specific marginal sampling oracle that leads to a local sampler beyond the regime of local uniformity.

3.1. **Marginal sampling oracles.** Before introducing the CTTP framework, we first define *marginal sampling oracles*. Due to the conditional independence property of Gibbs distributions, to sample from the (conditional) marginal distribution μ_v^{σ} for a vertex $v\in V$ and a configuration $\sigma\in[q]^{V\setminus v}$, it suffices to retrieve the spins of all neighbors, $\sigma(N(v))$. A *marginal sampling oracle* generalizes this concept by producing a marginal sample, given oracle access to the spin $\sigma(u)$ of each neighbor $u\in N(v)$.

Definition 3.1 (marginal sampling oracle)**.** Let μ be a distribution over $[q]^V$. For a variable $v\in V$, we define $\mathsf{Evaluate}^{\mathcal{O}}(v)$ as a procedure that makes oracle queries to $\mathcal{O}(u)$, which consistently returns a value $c_u\in[q]$ for each $u\in N(v)$.

We say that $\mathsf{Evaluate}^{\mathcal{O}}(v)$ is a *marginal sampling oracle* at v (with respect to μ) if:

- for each $\sigma\in[q]^{N(v)}$, assuming $\mathcal{O}(u)$ consistently returns $\sigma(u)$ for each $u\in N(v)$, the output of $\mathsf{Evaluate}^{\mathcal{O}}(v)$ is distributed exactly as μ_v^{σ}.

Recall the definition of systematic scan Glauber dynamics $\mathcal{P}(T)$ in Section 2.2. Using a marginal sampling oracle, the systematic scan Glauber dynamics can be simulated as follows.

Definition 3.2 (simulation of systematic scan Glauber dynamics via a marginal sampling oracle)**.** The systematic scan Glauber dynamics $\mathcal{P}(T)$ with respect to μ is simulated as:

(1) start with an arbitrary configuration $X_{-T}\in[q]^V$ satisfying $\mu(X_{-T})>0$ at time $t=-T$;
(2) at each time $-T<t\le 0$,
   (a) pick the vertex $v=v_{i(t)}$ where $i(t)\triangleq t \mod n$, let $X_t(u)=X_{t-1}(u)$ for every $u\in V\setminus\{v\}$;

(b) let $\mathsf{Evaluate}^{O}(v)$ be a marginal sampling oracle (w.r.t μ) at v where the oracle accesses $O(u)$ are replaced with $X_{t-1}(u)$ for each $u \in N(v)$, update $X_t(v) \leftarrow \mathsf{Evaluate}^{O}(v)$.

Remark 3.3 (grand coupling)**.** In Definition 3.2, the only randomness involved is within the subroutine $\mathsf{Evaluate}^{O}(v)$. Notably, for any implementation of a marginal sampling oracle, Definition 3.2 specifies a simulation of systematic scan Glauber dynamics, and implicitly defines a *grand coupling* that couples the Markov chain across all possible initial configurations. To see this, consider pre-sampling all random variables used within $\mathsf{Evaluate}^{O}(v_{i(t)})$ for each timestamp t, thereby defining the grand coupling.

3.2. **Simulating stationary Markov chains using backward deduction.** We present the CTTP framework for constructing the local sampler, which is a backward deduction of the forward simulation described in Definition 3.2 (or equivalently, the grand coupling constructed in Remark 3.3).

Consider the systematic scan Glauber dynamics running from the infinite past toward time 0, which is the limiting process of $\mathcal{P}(T)$ as $T \to \infty$, denoted by $\mathcal{P}(\infty)$. By Theorem 2.1, when $\mathcal{P}(T)$ is irreducible, the state X_0 of this process is distributed exactly according to μ. Our local sampler is then constructed by resolving the outcome $X_0(\Lambda)$, where Λ is the queried set of vertices. For any $t \le 0$ and $u \in V$, define

$$\mathsf{pred}_t(u) \triangleq \max\{t' \mid t' \le t, v_{i(t')} = u\} \tag{5}$$

as the last time, up to t, that vertex u was updated. The local sampler is formally presented in Algorithm 1.

Algorithm 1: $\mathsf{LocalSample}(\Lambda; M)$

Input: A q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, a subset of variables $\Lambda \subseteq V$.
Output: A random configuration $X \in [q]^{\Lambda}$.
Global variables: A mapping $M : \mathbb{Z} \to [q] \cup \{\perp\}$.

1 $X \leftarrow \varnothing, M \leftarrow \perp^{\mathbb{Z}}$;
2 **forall** $v \in \Lambda$ **do**
3 $\quad X(v) \leftarrow \mathsf{Resolve}(\mathsf{pred}_0(v); M)$;
4 **return** X;

Line 3 of Algorithm 1 utilizes a procedure Resolve, formally presented as Algorithm 2, which takes as input a timestamp $t \le 0$, and determines the outcome of the update at time t of $\mathcal{P}(\infty)$.

Algorithm 2: $\mathsf{Resolve}(t; M)$

Input: A q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, a timestamp $t \le 0$.
Output: A random value $x \in [q]$.
Global variables: A mapping $M : \mathbb{Z} \to [q] \cup \{\perp\}$.

1 **if** $M(t) \neq \perp$ **then return** $M(t)$; `// check if the outcome is already resolved`
2 $M(t) \leftarrow \mathsf{Evaluate}^{O}(v_{i(t)})$, with $O(u)$ replaced by $\mathsf{Resolve}(\mathsf{pred}_t(u); M)$ for each $u \in N(v_{i(t)})$;
3 **return** $M(t)$;

A global data structure M is maintained within Algorithm 1, storing the resolved values $M(t)$ for updates at each time t. It is initialized as $M = \perp^{\mathbb{Z}}$ in Line 1. This data structure M is introduced to facilitate memoization: the outcome of $\mathsf{Resolve}(t)$ is evaluated only once, ensuring consistency across multiple calls for the same t. For simplicity, we omit explicit references to M and write $\mathsf{LocalSample}(\Lambda)$ and $\mathsf{Resolve}(t)$ instead of $\mathsf{LocalSample}(\Lambda; M)$ and $\mathsf{Resolve}(t; M)$.

Line 2 of Algorithm 2 invokes a procedure $\mathsf{Evaluate}^{O}(v_{i(t)})$, which is abstractly defined in Definition 3.1 but is not yet fully implemented. Recall that the purpose of $\mathsf{Evaluate}^{O}(v_{i(t)})$ is to infer the value to which the vertex $v = v_{i(t)}$ is updated in $\mathcal{P}(\infty)$ at time t, which is distributed as $\mu_v^{X_{t-1}(N(v))}$ given access to $X_{t-1}(N(v))$. However, since Algorithm 2 implements a backward deduction (as opposed to a forward simulation) of the chain, the neighborhood configuration $X_{t-1}(N(v))$ at time t is not available directly. To address this, the algorithm recursively applies Algorithm 2 to infer the last updated value of each neighbor $u \in N(v)$ before time t (as specified in Line 2 of Algorithm 2).

Formally, the subroutine $\mathsf{Evaluate}(t)$ must satisfy the following local correctness condition:

Condition 3.4 (local correctness of $\mathsf{Evaluate}^{O}(v)$)**.** *For each $v \in V$, the procedure $\mathsf{Evaluate}^{O}(v)$ is a marginal sampling oracle at v, satisfying the requirement of Definition 3.1.*

Recall that Algorithm 2 is designed to resolve the outcome of $\mathcal{P}(\infty)$ at time 0. However, the limiting process $\mathcal{P}(\infty)$ is well-defined only if $\mathcal{P}(T)$ is irreducible. Additionally, we note that Algorithm 2 does not necessarily terminate. Nonetheless, we provide a sufficient condition that ensures both the irreducibility of $\mathcal{P}(T)$ and the termination of Algorithm 2.

Condition 3.5 (immediate termination of $\mathsf{Evaluate}^{O}(v)$)**.** *For each $v \in V$, let $\mathcal{E}_v$ be the event that $\mathsf{Evaluate}^{O}(v)$ terminates without making any calls to O. Then, the following must hold:*

$$\mathbf{Pr}\left[\mathcal{E}_v\right] > 0.$$

We now establish the correctness of Algorithm 1, assuming Conditions 3.4 and 3.5.

Lemma 3.6 (conditional correctness of Algorithm 1)**.** *Assume that Conditions 3.4 and 3.5 hold for $\mathsf{Evaluate}^{O}(v)$. Then, for any $\Lambda \subseteq V$, Algorithm 1 terminates with probability 1 and returns a random value $X \in [q]^{\Lambda}$ distributed according to μ_Λ upon termination.*

Lemma 3.6 is proved later in Section 3.3. It guarantees the termination and correctness of Algorithm 1, without addressing its efficiency. Next, we provide a sufficient condition for the efficiency of Algorithm 1.

Condition 3.7 (condition for fast termination of $\mathsf{Evaluate}^{O}(v)$)**.** *Let $\delta > 0$ be a parameter. For each $v \in V$, and each $\sigma \in [q]^{N(v)}$ such that $O(u)$ consistently returns $\sigma(u)$ for all $u \in N(v)$, let $\mathcal{T}_v^{\sigma}$ denote the total number of calls to $O(u)$ over all $u \in N(v)$. Then, the following holds:*

$$\mathbb{E}\left[\mathcal{T}_v^{\sigma}\right] \leq 1 - \delta.$$

We conclude this subsection with the following lemma, which establishes the efficiency of our local sampler under the assumption of Condition 3.7. Notably, it provides an upper bound on the total number of recursive Resolve calls, rather than simply counting the number of initial calls for each $t \leq 0$.

Lemma 3.8 (conditional efficiency of Algorithm 1)**.** *Assuming Condition 3.7 holds, the expected total number of calls to $\mathsf{Resolve}(t)$ within $\mathsf{LocalSample}(\Lambda)$ is $O(|\Lambda|)$.*

Proof. The introduction of the map M in Line 1 of $\mathsf{Resolve}(t)$ is for memoization and only reduces the number of recursive calls. As a result, the expected running time of $\mathsf{LocalSample}(\Lambda)$ can be upper-bounded by the sum of the expected running times of $\mathsf{Resolve}(\mathsf{pred}_0(v))$ for each $v \in \Lambda$. It remains to show that the expected running time of $\mathsf{Resolve}(\mathsf{pred}_0(v))$ is $O(1)$ for each $v \in V$.

As the mapping M only reduces the number of recursive calls, the behavior of $\mathsf{Resolve}(\mathsf{pred}_0(v))$ can be stochastically dominated by the following multitype Galton-Watson branching process:

- Start with a root node labeled with $\mathsf{pred}_0(v)$ at depth 0.
- For each $i = 0, 1, \ldots$: for all current leaves labeled with some timestamp t at depth i:
  - Perform an independent run of $\mathsf{Resolve}(t)$, and for each timestamp $t' < t$ such that $\mathsf{Resolve}(t)$ is directly recursively called, add a new node labeled with t' as a child of t.

By Condition 3.7, for any timestamp $t \leq 0$, the expected number of offspring of a node labeled t is at most $1 - \delta$. Thus, applying the theory of branching processes, the expected number of nodes generated by this process is at most $\delta^{-1} = O(1)$. Therefore, the expected number of $\mathsf{Resolve}(t)$ calls within $\mathsf{LocalSample}(\Lambda)$ is $O(|\Lambda|)$, completing the proof of the lemma. □

3.3. **Conditional correctness of the local sampler.** We will prove Lemma 3.6, which addresses the conditional correctness of the local sampler (Algorithm 1). At a high level, the proof follows the same structure of the proof in [FGW+23].

First, we need to establish some basic components.

Lemma 3.9. *Assume that both Conditions 3.4 and 3.5 hold for $\mathsf{Evaluate}^{O}(v)$, then $\mathcal{P}(T)$ is irreducible.*

Proof. Recall that in Condition 3.5, for any $v \in V$ and $\sigma \in [q]^{N(v)}$, $\mathcal{E}_v^{\sigma}$ denotes the event that $\mathsf{Evaluate}^{O}(v)$ terminates without any calls to O, assuming that $O(u)$ consistently returns $\sigma(u)$ for each $u \in N(v)$. For any $v \in V$, let $c_v \in [q]$ be an arbitrary possible outcome of $\mathsf{Evaluate}^{O}(v)$, conditioning

on $\mathcal{E}_v$ happens. Note that such c_v always exists by Condition 3.5. Then combining with Condition 3.4, we have

$$\min_{\sigma\in[q]^{N(v)}} \mu_v^{\sigma}(c_v) > 0. \tag{6}$$

Let $\tau \in [q]^V$ be the constant configuration where $\tau(v) = c_v$ for $t = \mathrm{pred}_0(v)$. By (6) and the chain rule, we see that $\mu(\tau) > 0$. Also following (6), any $\sigma \in [q]^V$ such that $\mu(\sigma) > 0$ can reach τ through Glauber moves by changing some $\sigma(u)$ to $\tau(u)$ one at a time. Note that $\mathcal{P}(T)$ is reversible, therefore, any $\sigma \in [q]^V$ such that $\mu(\sigma) > 0$ can also be reached from τ and hence $\mathcal{P}(T)$ is irreducible. □

For any finite $T > 0$, we introduce the following finite-time version of Algorithm 1, presented as Algorithm 3, which locally resolves the final state X_0 of $\mathcal{P}(T)$. Note that the only difference between Algorithms 1 and 3 is the different initialization of the map M.

Algorithm 3: $\mathsf{LocalSample}_T(\Lambda)$

Input: a q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, a subset of variables $\Lambda \subseteq V$
Output: A random configuration $X \in [q]^\Lambda$
Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\perp\}$

```
X ← ∅, M(t) ← X_{-T}(v_{i(t)}) for each t ≤ -T, M(t) ←⊥ for each t > -T;
forall v ∈ Λ do
    X(v) ← Resolve(pred_0(v));
return X;
```

We then have the following lemma.

Lemma 3.10. *Assume that both Conditions 3.4 and 3.5 hold for* $\mathsf{Evaluate}^{O}(v)$. *Then,*

(1) $\mathsf{LocalSample}(\Lambda)$ *terminates with probability* 1*;*
(2) For any initial state X_{-T}*, it holds that* $\lim_{T\to\infty} d_{\mathrm{TV}}\left(\mathsf{LocalSample}(\Lambda), \mathsf{LocalSample}_T(\Lambda)\right) = 0.$

Proof. We start with proving Item 1. It suffices to show the termination of $\mathsf{Resolve}(t_0)$ for any $t_0 \le 0$. Recall the event $\mathcal{E}_v$ in Condition 3.5. For each $t \le 0$, we similarly let $\mathcal{E}_t$ denote the event that $\mathsf{Evaluate}^{O}(v_{i(t)})$ within $\mathsf{Resolve}(t)$ terminates without making any calls to O. We also define the event:

$$\mathcal{B}_t : \mathcal{E}_{t'} \text{ happens for all } t' \in [t - n + 1, t].$$

We claim that if $\mathcal{B}_t$ happens for some $t \le t_0$, then no recursive calls to $\mathsf{Resolve}(t')$ would be incurred for any $t' \le t - n$ within $\mathsf{LocalSample}(\Lambda)$. For the sake of contradiction, assume that a maximum $t^* \le t - n$ exists such that $\mathsf{Resolve}(t^*)$ is called. As $t^* \le t - n < t_0$, $\mathsf{Resolve}(t^*)$ must be recursively called directly within another instance of $\mathsf{Resolve}(t')$ (through $\mathsf{Evaluate}^{O}(v_{i(t')})$) such that $t^* < t'$. Note that by Algorithm 2, the fact that $\mathsf{Evaluate}^{O}(v_{i(t)})$ only make recursive calls to $\mathsf{Resolve}(\mathrm{pred}_t(u))$ for some $u \in N(v_{i(t)})$ and (5) we also have $t^* > t' - n$. We then have two cases:

(1) $t' \le t - n$, this contradicts the maximality assumption for t^*.
(2) Otherwise $t' > t - n$. By $t^* \le t - n$ and $t' < t^* + n$ we have $t' \in [t - n + 1, t]$. Also, by the assumption that $\mathcal{B}_t$ happens, we have $\mathcal{E}_{t'}$ happens; therefore, $\mathsf{Resolve}(t')$ would have directly terminated without incurring any recursive call. This also leads to a contradiction and thus proves the claim.

Let $p \triangleq \min_t \mathbf{Pr}[\mathcal{E}_t]$, then $p > 0$ by Condition 3.5. Note that by Condition 3.5, for any $t \le t_0$, we have

$$\mathbf{Pr}[\mathcal{B}_t] = \prod_{t'=t-n+1}^{t} \mathbf{Pr}[\mathcal{E}_{t'}] \ge (1-p)^n > 0,$$

where the first equality is by $\mathcal{E}_t$ only depends on the randomness of procedure Evaluate, therefore all $\mathcal{E}_{t'}$ are independent.

For any $L > 0$, let $\mathcal{E}_L$ be the event that there is a recursive call to $\mathsf{Resolve}(t^*)$ where $t^* \le t_0 - Ln$. By the claim above,

$$\mathbf{Pr}\left[\mathcal{E}_L\right] \le \mathbf{Pr}\left[\bigwedge_{j=0}^{L-1}\left(\neg\mathcal{B}_{t_0-jn}\right)\right] = \prod_{j=0}^{L-1}\mathbf{Pr}\left[\neg\mathcal{B}_{t_0-jn}\right] \le (1-p)^L,$$

where the equality is again due to independence of $(\mathcal{E}_t)_{t\le t_0}$. Consequently, with probability 1, there is only a finite number of recursive calls, meaning that $\mathsf{LocalSample}(\Lambda)$ terminates with probability 1. This establishes Item 1.

For any $t \le 0$, since $\mathsf{Resolve}(t)$ terminates with probability 1, its output distribution is well-defined. Therefore, the output distribution of $\mathsf{LocalSample}(\Lambda)$ is well-defined. For any $\varepsilon > 0$, we choose a sufficiently large L such that $(1-p)^L \le \varepsilon$. For any $T \ge Ln - t_0$, we couple $\mathsf{LocalSample}(\Lambda)$ with $\mathsf{LocalSample}_T(\Lambda)$ by pre-sampling all random variables used in $\mathsf{Evaluate}^O(v_{i(t)})$ within $\mathsf{Resolve}(t)$ for each $t \le 0$. Here by Condition 3.4, the coupling fails if and only if $\mathsf{Resolve}(t')$ is recursively called within $\mathsf{LocalSample}(\Lambda)$ for some $t' \le -T$, that is, $\mathcal{E}_L$ happens. By the coupling lemma, we have

$$d_{\mathrm{TV}}\left(\mathsf{LocalSample}(\Lambda), \mathsf{LocalSample}_T(\Lambda)\right) \le \mathbf{Pr}\left[\mathcal{E}_L\right] \le (1-p)^L \le \varepsilon,$$

which proves Item 2 as we take $T \to \infty$. □

For any finite $T > 0$ and $-T \le t \le 0$, we let $X_{T,t}$ be the state of X_t in $\mathcal{P}(T)$. The following lemma shows $\mathsf{LocalSample}_T$ indeed simulates $\mathcal{P}(T)$.

Lemma 3.11. *Assume that Conditions 3.4 and 3.5 hold for* $\mathsf{Evaluate}^O(v)$. *Then for any* $\Lambda \subseteq V$, *the value returned by* $\mathsf{LocalSample}_T(\Lambda)$ *is identically distributed as* $X_{T,0}(\Lambda)$.

Proof. We maximally couple the value returned by each $\mathsf{Resolve}(t)$ and $X_t(v_{i(t)})$ in $\mathcal{P}(T)$ for each $t \le T$ and claim that in this case, the value returned by each $\mathsf{Resolve}(t)$ is exactly the same as $X_t(v_{i(t)})$ in $\mathcal{P}(T)$; hence the lemma holds by the definition of $\mathsf{LocalSample}_T$ and (5).

We prove the claim by induction from time $-T$ to 0. For each $-T \le t < 0$, let $v = v_{i(t)}$ and consider the value returned by $\mathsf{Resolve}(\mathsf{pred}_t(u))$ for each $u \in N(v)$:

- If $\mathsf{pred}_t(u) < -T$, then by (5), the value of u is not updated up to time t in $\mathcal{P}(T)$, hence $X_t(u) = X_{-T}(u)$ and the value returned by $\mathsf{Resolve}(\mathsf{pred}_t(u))$ is $X_{-T}(u) = X_t(u)$ by the initialization of M in Algorithm 3.
- Otherwise, $-T \le \mathsf{pred}_t(u) < t$ by (5) and $u \in N(v)$, so the value returned by $\mathsf{Resolve}(\mathsf{pred}_t(u))$ is $X_{\mathsf{pred}_t(u)} = X_t(u)$ by the induction hypothesis. Hence by Line 2 of Algorithm 2 and Condition 3.4, both the distribution of $\mathsf{Resolve}(t)$ and $X_t(v_{i(t)})$ is $\mu^{X_{t-1}(N(v))}(v)$ and hence can be perfectly coupled.

Hence, the claim holds, and the lemma is proved. □

We are now ready to prove Lemma 3.6.

Proof of Lemma 3.6. By Item 1 of Lemma 3.10, we have $\mathsf{LocalSample}(\Lambda)$ terminates with probability 1 and its output distribution is well-defined. It remains to prove that the output distribution of $\mathsf{LocalSample}(\Lambda)$ is exactly μ_Λ.

By Lemma 3.9, we have that $\mathcal{P}(T)$ is irreducible. Then, Theorem 2.1 implies that

$$\lim_{T\to\infty} d_{\mathrm{TV}}\left(\mu_\Lambda, X_{T,0}(\Lambda)\right) = 0, \tag{7}$$

For any $T \ge 0$, by the triangle inequality, we have

$$\begin{aligned} d_{\mathrm{TV}}\left(\mu_\Lambda, \mathsf{LocalSample}(\Lambda)\right) \le & d_{\mathrm{TV}}\left(\mu_\Lambda, \mathsf{LocalSample}_T(\Lambda)\right) \\ & + d_{\mathrm{TV}}\left(\mathsf{LocalSample}(\Lambda), \mathsf{LocalSample}_T(\Lambda)\right). \end{aligned} \tag{8}$$

Altogether, the theorem follows from

$$\begin{aligned}
& d_{\mathrm{TV}}\left(\mu_\Lambda, \mathsf{LocalSample}(\Lambda)\right) \\
(\text{by (8)}) \quad \leq\ & \limsup_{T\to\infty} d_{\mathrm{TV}}\left(\mu_\Lambda, \mathsf{LocalSample}_T(\Lambda)\right) \\
& + \limsup_{T\to\infty} d_{\mathrm{TV}}\left(\mathsf{LocalSample}_T(\Lambda), \mathsf{LocalSample}(\Lambda)\right) \\
(\text{by Lemma 3.10}) \quad =\ & \limsup_{T\to\infty} d_{\mathrm{TV}}\left(\mu_\Lambda, \mathsf{LocalSample}_T(\Lambda)\right) \\
(\text{by (8)}) \quad \leq\ & \limsup_{T\to\infty} d_{\mathrm{TV}}\left(\mu_\Lambda, X_{T,0}(\Lambda)\right) + \limsup_{T\to\infty} d_{\mathrm{TV}}\left(X_{T,0}(\Lambda), \mathsf{LocalSample}_T(\Lambda)\right) \\
(\text{by Lemma 3.11}) \quad =\ & \limsup_{T\to\infty} d_{\mathrm{TV}}\left(\mu_\Lambda, X_{T,0}(\Lambda)\right) \\
(\text{by (7)}) \quad =\ & 0.
\end{aligned}$$

□

4. Application: a local sampler for spin systems with soft constraints

In this section, we construct a new marginal sampling oracle for q-spin systems with soft constraints. We will use this oracle to build our local sampler and prove Theorems 1.2 and 1.6. Our construction is inspired by a simple rejection sampling procedure for sampling from μ_v^σ, given the neighborhood configuration $\sigma \in [q]^{N(v)}$ for some $v \in V$.

This rejection sampling procedure is described as follows:

(1) Propose a random value $c \in [q]$ distributed according to λ_v;
(2) With probability $\prod_{e=(u,v)\in E} A_e(\sigma(u), c)$, accept the proposal and return c as the final outcome; Otherwise, reject the proposal and go to Step (1).

Note that the well-definedness of the above procedure follows from Condition 1.1, which ensures that all λ_v and A_e are normalized. Given a q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, for any vertex $v \in V$, we can then define a marginal sampling oracle at v based on the rejection sampling procedure.

Algorithm 4: a marginal sampling oracle for q-spin systems with soft constraints

Input: A q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, a vertex $v \in V$.
Output: A value $X \in [q]$.
Oracle access: $\mathcal{O}(u)$ for each $u \in N(v)$.

1 Sample an infinite long sequence of i.i.d. tuples $\{(c_i, (r_{i,u})_{u\in N(v)})\}_{1\leq i<\infty}$ where each $c_i \in [q]$ is distributed as λ_v and each $r_{i,u}$ is chosen uniformly from $[0, 1]$;
2 $i^* \leftarrow \min\{i \mid \forall e = (u, v) \in E, r_{i,u} < A_e(\mathcal{O}(u), c_i)\}$;
3 **return** c_{i^*};

We remark that in Line 2 of Algorithm 4, such an i^* always exists because A_e satisfies Condition 1.1. We now present the following lemma.

Lemma 4.1. *Suppose that the input q-spin system $\mathcal{S}$ satisfies Condition 1.1. Then, Algorithm 4 implements a marginal sampling oracle at v.*

Proof. Given a q-spin system $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, in Algorithm 4, for each $i \geq 1$ and each $u \in N(v)$, recall that each c_i is chosen distributed as $p \in \Delta_q$ where $p(x) \propto \lambda_v(x)$ and each $r_{i,u}$ is independently chosen uniformly from $[0, 1]$. Let $\mathcal{D}_i$ be the event that

$$\mathcal{D}_i : \forall e = (u, v) \in E, r_{i,u} < A_e(\mathcal{O}(u), c_i),$$

then for any $x \in [q]$, note that $O(u_j) = \sigma(u_j)$ under assumption, we have

$$\begin{aligned}\mathbf{Pr}\left[c_i = x \mid \mathcal{D}_i\right] &= \frac{\mathbf{Pr}\left[c_i = x \wedge \mathcal{D}_i\right]}{\mathbf{Pr}\left[\mathcal{D}_i\right]} \\ &= \frac{\lambda_v(x) \prod\limits_{e=(u,v)\in E} A_e(\sigma(u), x)}{\sum\limits_{c\in[q]}\left(\lambda_v(c) \prod\limits_{e=(u,v)\in E} A_e(\sigma(u), c)\right)} = \mu_v^{\sigma}(x).\end{aligned}$$

Let i^* be the smallest index chosen in Line 2 of Algorithm 4, i.e., $i^* = \min\{i \mid \mathcal{D}_i\}$. the output of Algorithm 4 follows the distribution of c_{i^*} conditioning on $\mathcal{D}_{i^*}$, concluding the proof of the lemma. □

The marginal sampling oracle in Algorithm 4 as originally designed would require a significant number of oracle calls in Line 2, potentially violating the efficiency condition outlined in Condition 3.7. The key optimization is to invoke the oracle $O(u)$ only when necessary for each neighbor $u \in N(v)$, rather than for every iteration in Algorithm 4. Formally, assuming Condition 1.1 holds, in Line 2 of Algorithm 4, if $r_{i,u} < C$, where $C = C(\Delta, \delta) \triangleq 1 - \frac{1-\delta}{2\Delta}$, then the inequality $r_{i,u} < A_e(O(u), c_i)$ will hold true regardless of the value of $O(u)$. This is because the term C is chosen such that $r_{i,u}$ is sufficiently small to ensure success in the comparison without needing the actual value of $O(u)$. Consequently, it becomes unnecessary to call $O(u)$ when $r_{i,u} < C$. With this idea of optimization, we propose the following implementation of Evaluate$^O(v)$, presented in Algorithm 5, which builds upon the above idea to efficiently sample without violating the fast termination condition.

Algorithm 5: Evaluate$^O(v)$

Input: A q-spin systems $\mathcal{S} = (G = (V, E), \boldsymbol{\lambda}, \boldsymbol{A})$, a vertex $v \in V$.
Output: A value $c \in [q]$.

1 Sample an infinite long sequence of i.i.d. tuples $\{(c_i, (r_{i,u})_{u\in N(v)})\}_{1\le i<\infty}$, where each $c_i \in [q]$ is distributed as λ_v and each $r_{i,u}$ is chosen uniformly from $[0, 1]$;
2 **for** $i = 1, 2, \ldots$ **do**
3 $\mathit{flag} \leftarrow 1$;
4 **for** $e = (u, v) \in E$ **do**
5 **if** $r_{i,u} \ge C$ **then**
6 **if** $r_{i,u} \ge A_e(O(u), c_i)$ **then** $\mathit{flag} \leftarrow 0$;
7 **if** $\mathit{flag} = 1$ **then return** c_i;

Remark 4.2 (principle of deferred decision)**.** In Line 1 of Algorithm 5, we are required to sample an infinitely long sequence $\{(c_i, \{r_{i,u}\}_{u\in N(v)})\}_{1\le i<\infty}$. Obviously, it is not feasible to directly sample an infinite number of random variables for implementation. Instead, we adopt the principle of deferred decision: each c_i and $r_{i,u}$ is generated only when they are accessed in Lines 4 and 6 of Algorithm 5.

Next, we show that the marginal sampling oracle Evaluate$^O(v)$ in Algorithm 5 satisfies the conditions for both correctness and efficiency.

Lemma 4.3. *Suppose that the input q-spin system $\mathcal{S}$ satisfies Condition 1.1. Then, the marginal sampling oracle* Evaluate$^O(v)$ *implemented as in Algorithm 5 satisfies both Condition 3.4 and Condition 3.5.*

Proof. Condition 3.4 can be verified directly by Lemma 4.1 and comparing Algorithms 4 and 5.

For Condition 3.5, recall that $\mathcal{E}_v$ is the event that Evaluate$^O(v)$ in Algorithm 5 terminates without any calls to O. Note that $\mathcal{E}_v$ occurs if and only if Algorithm 5 terminates within 1 round of the loop at Line 2 and $r_{1,u} < C$ holds for each $u \in N(v)$. Since c_1 and each $r_{1,u}$ are independent, assuming the q-spin system $\mathcal{S}$ satisfies Condition 1.1 with constant $\delta > 0$, we immediately have

$$\mathbf{Pr}\left[\mathcal{E}_t\right] \ge C^{\Delta} > 0.$$

It verifies that Evaluate$^O(v)$ satisfies Condition 3.5 assuming Condition 1.1 holds. □

Lemma 4.4. *Suppose that the input q-spin system $\mathcal{S}$ satisfies Condition 1.1. Then, the marginal sampling oracle* $\mathsf{Evaluate}^{O}(v)$ *implemented as in Algorithm 5 satisfies Condition 3.7.*

Proof. Fix some $v \in V$. Also fix some $\sigma \in [q]^{N(v)}$ and assume that $O(u)$ returns $\sigma(u)$ within $\mathsf{Evaluate}^{O}(v)$. Within each round of the outer for loop at Line 2 of Algorithm 5, the expected total number of oracle calls to $O(u)$ for any $u \in N(v)$ is given by:

$$\mathbb{E}\left[\text{number of calls to } O(\cdot) \text{ in one iteration}\right] = \sum_{c\in[q]}\left(\lambda_v(c)\sum_{e=(u,v)\in E}(1-C)\right). \tag{9}$$

Let $\mathcal{I}_v^{\sigma}$ be the number of executions of the outer for loop at Line 2 in $\mathsf{Evaluate}^{O}(v)$. Note that $\mathcal{I}_v^{\sigma}$ corresponds exactly to the number of executions of Item 1 in the rejection sampling. Therefore,

$$\mathbb{E}\left[\mathcal{I}_v^{\sigma}\right] = \frac{1}{\sum_{c\in[q]}\left(\lambda_v(c)\prod_{e=(u,v)\in E}A_e(\sigma(u),c)\right)}. \tag{10}$$

Note that the number of oracle calls in each iteration are i.i.d. random variables with finite mean. Applying Wald's equation, we have:

$$\mathbb{E}\left[\mathcal{T}_v^{\sigma}\right] = \mathbb{E}\left[\mathcal{I}_v^{\sigma}\right]\cdot\mathbb{E}\left[\text{number of calls to } O(\cdot) \text{ in one iteration}\right].$$

It then follows from (9) and (10) that:

$$\begin{aligned}
\mathbb{E}\left[\mathcal{T}_v^{\sigma}\right] &= \frac{\sum_{c\in[q]}\left(\lambda_v(c)\sum_{e=(u,v)\in E}(1-C)\right)}{\sum_{c\in[q]}\left(\lambda_v(c)\prod_{e=(u,v)\in E}A_e(\sigma(u),c)\right)}\\
&\le \frac{\sum_{c\in[q]}\left(\lambda_v(c)\cdot(1-C)\cdot|N(v)|\right)}{\sum_{c\in[q]}\left(\lambda_v(c)\cdot C^{\Delta}\right)} && \text{(by Condition 1.1)}\\
&\le \frac{\sum_{c\in[q]}(\lambda_v(c)(1-\delta)/2)}{\sum_{c\in[q]}(\lambda_v(c)(1+\delta)/2)} && \left(\text{by } C = 1-\frac{\delta}{2\Delta}\right)\\
&< 1.
\end{aligned}$$

It proves that $\mathsf{Evaluate}^{O}(v)$ satisfies Condition 3.7 assuming Condition 1.1 holds. □

We are now ready to prove Theorems 1.2 and 1.6.

Proof of Theorem 1.2. We use Algorithm 1 as our local sampler, where the subroutine $\mathsf{Evaluate}^{O}(v)$ is implemented by Algorithm 5 (using the principle of deferred decision as explained in Remark 4.2).

Here, the correctness of sampling follows from Lemmas 3.6 and 4.3.

For efficiency, by Lemmas 3.8 and 4.4, we have that the expected number of Resolve calls is $O(|\Lambda|)$. Also, note that each outer loop either terminates directly or results in at least one call to Resolve. Hence, the overall running time is bounded by $\Delta\log q$ times the total number of Resolve calls, which is $O(|\Lambda|\Delta\log q)$ in expectation. □

Proof of Theorem 1.6. Note that by the self-reducibility of spin systems with soft constraints (i.e., Condition 1.1 holds under arbitrary pinning), we have the uniform lower bound:

$$\forall v\in V, c\in[q],\quad \mu_v^{\sigma}(c) \ge \frac{C(\Delta,\delta)^{\Delta}}{q} \ge \frac{1}{2q}.$$

This ensures that the marginal probabilities $\mu_v^\sigma(x)$ are all bounded away from zero. Therefore, by the Chernoff bound, for each $x \in Q_v$, the value $\mu_v^\sigma(x)$ can be estimated to within a multiplicative error of $(1 \pm \varepsilon)$ with probability at least 0.9 using $O(q/\varepsilon^2)$ approximate samples.

According to Theorem 1.2, each such sample can be generated in expected time $O(|\Lambda|\Delta \log q)$, where we easily handle the conditioning in σ by setting $M(t) = \sigma_{(v_{i(t)})}$ whenever Resolve(t) is called for some $v_{i(t)} \in \Lambda$, so the total expected cost to estimate all $\mu_v^\sigma(x)$ for $x \in Q_v$ is bounded by

$$O\left(|\Lambda| q^2 \log q \Delta \varepsilon^{-2}\right). \tag{11}$$

Applying Markov's inequality, the probability that the total cost exceeds this bound is at most 0.1. Therefore, by truncating the algorithm's running time to (11), we obtain a bounded-cost algorithm which, with probability at least $0.9 - 0.1 = 0.8$, returns estimates of all $\mu_v^\sigma(x)$ within a multiplicative factor of $(1 \pm \varepsilon)$ for all $x \in Q_v$.

Finally, to boost the success probability to at least $1 - \delta/q$, we repeat this procedure independently $O(\log(q/\delta))$ times and take the median of the resulting estimates. This yields the desired algorithm claimed in the theorem by applying Chernoff's bound again. □

5. Extension: a local sampler for q-colorings

In this section, we show how to extend our CTTP framework to design local samplers for a prototypical spin system with truly repulsive hard constraints: the proper q-colorings.

A key challenge in applying the CTTP framework to problems with hard constraints lies in the absence of unconditional marginal lower bounds. In the systematic scan Glauber dynamics $\mathcal{P}(T) = (X_t)_{t=-T}^0$ for q-colorings, determining the outcome of an update at node $v = v_{i(t)}$ at time t requires knowledge of the color set S_t, defined as the set of colors assigned to the neighbors of v in the configuration σ_t:

$$S_t \triangleq \{X_t(u) \mid u \in N(v)\}. \tag{12}$$

To correctly perform the update at time t, one needs to sample uniformly from $[q] \setminus S_t$. However, if for any neighbor $u \in N(v)$, the outcome of its most recent update occurring at time $\mathrm{pred}_t(u)$ is unknown, then the update at time t cannot be resolved with positive probability. In other words, the immediate termination condition (Condition 3.5) does not hold without additional information.

To generate a uniform sample from the set $[q] \setminus S_t$, we could use a rejection sampling procedure similar to that described in the previous section: propose a color c uniformly at random from $[q]$, and accept it if $c \notin S_t$. Since S_t contains at most Δ colors, when $q > C\Delta$ for some sufficiently large constant C, this rejection sampling succeeds with constant probability. However, a key challenge remains: determining whether $c \in S_t$ requires knowledge of $X_t(u)$ for each neighbor $u \in N(v)$, which may lead to endless recursion.

The crucial observation is that for each neighbor $u \in N(v)$, at the time of its last update $t' = \mathrm{pred}_u(t)$, the probability that $X_{t'}(u) = c$ is at most $\frac{1}{|[q] \setminus S_{t'}|}$, which is bounded above by $\frac{2}{q}$ under the assumption $q \geq 2\Delta$. This allows us to implement a probabilistic filter to test whether $X_{t'}(u) \neq c$ as follows:

- With probability $1 - \frac{2}{q}$, we directly *certify* that $X_{t'}(u) \neq c$;
- With the remaining probability $\frac{2}{q}$, we first check whether $c \in S_{t'}$:
  - If $c \in S_{t'}$, we can immediately *certify* that $X_{t'}(u) \neq c$. This condition can be verified recursively by invoking a similar procedure for each neighbor $w \in N(u)$;
  - Otherwise, we set $X_{t'}(u) = c$ with probability $\frac{q/2}{|[q] \setminus S_{t'}|}$, and otherwise *certify* that $X_{t'}(u) \neq c$.

Observe that the filtering procedure terminates immediately with probability at least $1 - \frac{2}{q}$. When q is sufficiently large, this high probability of immediate termination serves as the basis of the recursion, allowing us to avoid infinite recursion. To fully implement this filtering process, two aspects remains to be specified: what it means to "certify" that a color cannot appear at a given timestamp t, and how to simulate a trial with an unknown success probability $\frac{q/2}{|[q] \setminus S_{t'}|}$. We briefly outline these below:

- To certify that a certain color cannot appear at timestamp t, we have an auxiliary mapping $L : \mathbb{Z} \to 2^{[q]}$ maintained globally outside the recursion, where each $L(t)$ is initially set to $[q]$. This data structure records the set of *available colors* at each timestamp. The color assigned at

time t is interpreted as being uniformly sampled from $L(t) \setminus S_t$. Thus, to certify that c cannot be the outcome of the update at time t, we simply remove c from $L(t)$.

- Although we cannot directly compute the probability $\frac{q/2}{|[q]\setminus S_{t'}|}$ since $S_{t'}$ is not explicitly known, we can query the membership of certain colors in $S_{t'}$ through recursive calls. This allows us to employ a *Bernoulli factory algorithm* to simulate a Bernoulli trial with the correct success probability $\frac{q/2}{|[q]\setminus S_{t'}|}$.

Remark 5.1 (adaptivity of the grand coupling)**.** As noted in Remark 3.3, simulating the Markov chain in this manner implicitly defines a grand coupling over all starting configurations. In the procedure described above, this grand coupling is *adaptive*: the coupling decision at a given timestamp t may depend on the outcome of the coupling at an earlier time $t' < t$. This adaptivity is a crucial feature of our local sampler for q-colorings. Later, we will show that it does not compromise the correctness of the Coupling Towards The Past (CTTP) framework. We also note that similar adaptive strategies have been employed in the design of several perfect sampling algorithms based on the Coupling From The Past (CFTP) method [BC20, JSS21]. However, our strategy fundamentally differs from those in the CFTP framework, due to the additional constraints imposed by local samplers.

5.1. **The local sampler.** We now formally present our local sampler for q-colorings. As discussed above, our local sampler is built on the CTTP framework, with a key modification that allows partial information whether an updated outcome equals some particular color to be obtained. The local sampler relies on two core subroutines:

- $\mathsf{Resolve}(t)$, which takes as input a timestamp $t \le 0$, and returns the outcome of the update at time t (i.e., the color to which the vertex is updated);
- $\mathsf{Check}(t, c)$, which takes as input a timestamp $t \le 0$ as well as a color $c \in [q]$, and determines whether the outcome of the update at time t equals c;

Our local sampler for q-colorings is formally described in Algorithm 6.

Algorithm 6: $\mathsf{LocalSample}(\Lambda; M, L)$

Input: A q-coloring instance $G = (V, E)$, a subset of variables $\Lambda \subseteq V$.
Output: A random configuration $X \in [q]^\Lambda$.
Global variables: Two mappings $M : \mathbb{Z} \to [q] \cup \{\perp\}$, $L : \mathbb{Z} \to 2^{[q]}$.

```
X ← ∅, M ← ⊥^ℤ, L ← [q]^ℤ;
forall v ∈ Λ do
    X(v) ← Resolve(pred_0(v); M, L);
return X;
```

In Algorithm 6, two global mappings, M and L, are maintained for memoization to ensure the consistency of the algorithm. Specifically, $M(t)$ stores the resolved values of updates at each time t, and $L(t)$ stores the remaining set of available colors for unresolved updates at each time t.

We now formally present the two core procedures, $\mathsf{Resolve}(t)$ and $\mathsf{Check}(t, c)$, which underpin our construction of the local sampler. Similar to the case of spin systems with soft constraints, these procedures are recursively defined, making calls to themselves on earlier timestamps. To facilitate their definition, we adopt a variant of the abstraction for local sampling procedures, resembling the $\mathsf{Evaluate}^{O}(v)$ introduced in Definition 3.1, extended to allow access to partial information.

Definition 5.2 (local sampling procedures with access to partial information)**.** For each vertex $v \in V$ and color $c \in [q]$, we define the procedures $\mathsf{Evaluate}^{O}(v)$ and $\mathsf{Evaluate}^{O}(v, c)$, which make oracle queries to:

- $O(u)$, which consistently returns a value $c_u \in [q]$ for each neighbor $u \in N(v)$;
- $O(u, c)$, which consistently returns a binary value $x_{u,c} \in \{0, 1\}$ for each $u \in N(v)$ and $c \in [q]$, such that $x_{u,c} = 1$ if and only if $c_u = c$.

In addition, since local sampling steps require knowledge of the set of currently available colors, finite slices of the global mapping L may be passed by reference to $\mathsf{Evaluate}^{O}(v)$ and $\mathsf{Evaluate}^{O}(v, c)$.

We remark that Definition 5.2 differs from the marginal sampling oracles in Definition 3.1 as it imposes no requirement on the distribution of the outcomes produced by $\mathsf{Evaluate}^{\mathcal{O}}(v)$ and $\mathsf{Evaluate}^{\mathcal{O}}(v, c)$.

The procedures $\mathsf{Resolve}(t)$ and $\mathsf{Check}(t, c)$ are detailed in Algorithms 7 and 8, respectively.

The procedure $\mathsf{Resolve}(t)$ first checks whether the update at time t has already been resolved, and if not, invokes $\mathsf{Evaluate}^{\mathcal{O}}(v)$ to resolve it, passing the list of available colors $L(t)$ by reference. Similarly, the procedure $\mathsf{Check}(t, c)$ performs a series of preliminary checks before invoking $\mathsf{Evaluate}^{\mathcal{O}}(v, c)$ determine equality to c, also passing $L(t)$ by reference and potentially updating $L(t)$ at the end.

An important detail occurs in Lines 8 and 9 of $\mathsf{Check}(t, c)$: if the size of the list $L(t)$ falls below 50Δ, the algorithm invokes $\mathsf{Resolve}(t)$ to fully resolve the outcome at time t rather than merely checking equality to c. This mechanism ensures that the list size $|L(t)|$ remains above 50Δ whenever it is passed to either $\mathsf{Evaluate}^{\mathcal{O}}(v)$ or $\mathsf{Evaluate}^{\mathcal{O}}(v, c)$ during the algorithm's execution, assuming $q \geq 50\Delta$.

Algorithm 7: $\mathsf{Resolve}(t; M, L)$

Input: A q-coloring instance $G = (V, E)$, a timestamp $t \in \mathbb{Z}$.
Output: A random configuration $X \in [q]^{\Lambda}$.
Global variables: Two mappings $M : \mathbb{Z} \to [q] \cup \{\perp\}$, $L : \mathbb{Z} \to 2^{[q]}$.

1 **if** $M(t) \neq \perp$ **then return** $M(t)$; `// check if the outcome is already resolved`
2 $M(t) \leftarrow \mathsf{Evaluate}^{\mathcal{O}}(v_{i(t)}; L(t))$, with
- $\mathcal{O}(u)$ replaced by $\mathsf{Resolve}(\mathrm{pred}_t(u); M, L)$ for each $u \in N(v_{i(t)})$;
- $\mathcal{O}(u, c)$ replaced by $\mathsf{Check}(\mathrm{pred}_t(u), c; M, L)$ for each $u \in N(v_{i(t)})$;

3 **return** $M(t)$;

Algorithm 8: $\mathsf{Check}(t, c; M, L)$

Input: A q-coloring instance $G = (V, E)$, a timestamp $t \in \mathbb{Z}$, a color $c \in [q]$.
Output: A binary value $x \in \{0, 1\}$.
Global variables: Two mappings $M : \mathbb{Z} \to [q] \cup \{\perp\}$, $L : \mathbb{Z} \to 2^{[q]}$.

1 **if** $M(t) \neq \perp$ **then** `// check if the outcome is already resolved`
2 **if** $M(t) = c$ **then**
3 **return** 1;
4 **else**
5 **return** 0;
6 **if** $c \notin L(t)$ **then** `// check if` c `is already not available`
7 **return** 0;
8 **if** $|L(t)| \leq 50\Delta$ **then** `// resolve the full outcome instead when` $|L(t)| \leq 50\Delta$
9 $\mathsf{Resolve}(t; M, L)$;
10 $x \leftarrow \mathsf{Evaluate}^{\mathcal{O}}(v_{i(t)}, c; L(t))$, with
- $\mathcal{O}(u)$ replaced by $\mathsf{Resolve}(\mathrm{pred}_t(u); M, L)$ for each $u \in N(v_{i(t)})$;
- $\mathcal{O}(u, c)$ replaced by $\mathsf{Check}(\mathrm{pred}_t(u), c; M, L)$ for each $u \in N(v_{i(t)})$;

11 **if** $x = 1$ **then**
12 $M(t) \leftarrow c$;
13 **else**
14 $L(t) \leftarrow L(t) \setminus \{c\}$;
15 **return** x;

We now present our specific construction of local sampling procedures for q-colorings. To fully evaluate a specific outcome of the update and realize $\mathsf{Evaluate}^{\mathcal{O}}(v; L)$, we employ the strategy described earlier. Let $S = \{\mathcal{O}(u) \mid u \in N(v)\}$. To obtain a uniform sample from $L \setminus S$, we repeatedly select a color

$c \in L$ uniformly at random and check whether $c \in S$ by querying the oracle $O(u, c)$ for each $u \in N(v)$. This procedure is detailed in Algorithm 9.

Algorithm 9: Evaluate$^{O}(v; L)$

Input: A q-coloring instance $G = (V, E)$, a variable $v \in V$, and a list of available colors $L \subseteq [q]$, with the promise that $|L| \geq 50\Delta$.
Output: A value $c \in [q]$.
Oracle access: $O(u)$ for each $u \in N(v)$ and $O(u, c)$ for each $u \in N(v)$ and $c \in [q]$.
Global variables: a list of available colors $L \subseteq [q]$, with the promise that $c \notin L$ and $|L| \geq 50\Delta$.

```
while True do
    choose c ∈ L uniformly at random;
    if O(u, c) = 0 for each u ∈ N(v) then
        return c;
    L ← L \ {c};
```

To determine whether the outcome equals a specific color c and realize Evaluate$^{O}(v, c; L)$, we leverage the observation that when the size of the available color list satisfies $|L| \geq 2\Delta \geq 2|S|$, the probability that a uniform sample from $L \setminus S$ equals c is small. This motivates the following strategy to simulate a coin that always returns 0 when $c \in S$, and when $c \notin S$, returns 1 with probability $1/|L \setminus S|$ and 0 otherwise:

- With probability $1 - \frac{1}{|L|/2}$, return 0 directly;
- Otherwise:
  - If $c \in S$, return 0;
  - Otherwise, return 0 with probability $\frac{|L|/2}{|L \setminus S|}$, and return 1 otherwise.

When the first step does not immediately return 0, we query $O(u, c)$ for each $u \in N(v)$ to check whether $c \in S$. If $c \notin S$, a challenge arises: the set S is unknown to the algorithm, so we cannot directly simulate a coin with success probability $\frac{|L|/2}{|L \setminus S|}$. While one could retrieve S by querying $O(u)$ for every neighbor $u \in N(v)$, this incurs a prohibitive number of recursive calls. Fortunately, we can efficiently simulate a coin with success probability $\frac{|L \setminus S|}{|L|}$ using the following procedure:

- Choose a color $c \in L$ uniformly at random.
- Determine whether $c \in S$: query $O(u, c)$ for all $u \in N(v)$; if any query returns 1, output 0; otherwise, output 1.

Moreover, given access to such a $\frac{|L \setminus S|}{|L|}$-coin, we can simulate a coin with success probability $\frac{|L|/2}{|L \setminus S|}$ by solving an instance of the Bernoulli factory problem [VN51]. Specifically, we employ a Bernoulli factory algorithm for division: given access to a coin with success probability $p = \frac{|L \setminus S|}{|L|}$, we simulate a coin with success probability $\frac{1}{2p}$.

Our complete procedure for checking whether the outcome equals a specific color c is described in Algorithm 10, while the subroutine for simulating a $\frac{|L \setminus S|}{|L|}$-probability coin is described in Algorithm 11.

Remark 5.3 (implementation of the algorithm)**.** Note that our local sampler for q-colorings (Algorithms 6–11) involves a considerable number of recursive calls. To implement this algorithm efficiently, all recursive calls to Resolve(t) and Check(t, c) are managed using a stack. At each step, the algorithm pops the call at the top of the stack — either Resolve(t) or Check(t, c) — executes it, updates the values of $L(t)$ and $M(t)$ in the global data structure if needed, and pushes any new recursive calls onto the stack. Since each call to Resolve(t) or Check(t, c) only recurses on strictly smaller timestamps $t' < t$, the recursions will be executed correctly as above.

The Bernoulli factory for division used in Algorithm 10 is achieved by a combination of existing constructions [NP05, Hub16, Mor21]. Here, we present its correctness and efficiency.

Algorithm 10: Evaluate$^{O}(v, c; L)$

Input: A q-coloring instance $G = (V, E)$, a variable $v \in V$, a color $c \in [q]$
Output: A binary value $x \in \{0, 1\}$.
Oracle access: $O(u)$ for each $u \in N(v)$ and $O(u, c)$ for each $u \in N(v)$ and $c \in [q]$.
Global variables: a list of available colors $L \subseteq [q]$, with the promise that $c \notin L$ and $|L| \geq 50\Delta$.

with probability $1 - \frac{1}{|L|/2}$ **return** 0;
forall $u \in N(v)$ **do**
  If $O(u, c) = 1$ **return** 0;
let C be an $\frac{|L\setminus S|}{|L|}$-probability coin where $S = \{O(u) \mid u \in N(v)\}$, generated as in Algorithm 11;
with probability $\frac{|L|/2}{|L\setminus S|}$ return 1, otherwise return 0, where the probability $\frac{|L|/2}{|L\setminus S|}$ is generated using the Bernoulli factory algorithm for division [Mor21], given access to C.

Algorithm 11: access to an $|L \setminus S|/|L|$-probability coin

Input: A q-coloring instance $G = (V, E)$, a variable $v \in V$, a color $c \in [q]$
Output: A binary value $x \in \{0, 1\}$.
Oracle access: $O(u)$ for each $u \in N(v)$ and $O(u, c)$ for each $u \in N(v)$ and $c \in [q]$.
Global variables: A list of available colors $L \subseteq [q]$.

choose $c_0 \in L$ uniformly at random;
forall $u \in N(v)$ **do**
  if $O(u, c_0) = 1$ **then return** 0;
return 1;

Lemma 5.4 (correctness and efficiency of the Bernoulli factory algorithm)**.** *There is a Bernoulli factory algorithm accessing a ξ-coin with the promise that $\frac{1}{2} + \zeta \leq \xi \leq 1$ for some $\zeta > 0$, terminates with probability 1, returns 1 with probability $\frac{1}{2\xi}$ and returns 0 otherwise. Moreover, the expected number of calls to the ξ-coin is at most $9.5\xi^{-1}\zeta^{-1}$.*

The proof of Lemma 5.4 will be presented in Appendix A, where we also present the explicit construction of the Bernoulli factory algorithm for division.

5.2. **Correctness of the local sampler.** We now proceed to prove the correctness of the local sampler, stated in Lemma 5.5. Here, we establish only the *conditional* correctness under the assumption that the local sampler always terminates. The proof of termination, along with the efficiency analysis of the algorithm, is presented in the next subsection.

Lemma 5.5 (conditional correctness of the local sampler for q-colorings)**.** *Suppose that the input q-coloring instance $G = (V, E)$ satisfies $q \geq 65\Delta$ where $\Delta \geq 1$ is the maximum degree of G. For any $\Lambda \subseteq V$, suppose that* LocalSample(Λ) *terminates almost surely, then the output X of* LocalSample(Λ) *follows the law μ_Λ, where μ denotes the uniform distribution over all proper q-colorings of G.*

To prove Lemma 5.5, we first establish the correctness of the local procedures Evaluate$^{O}(v; L)$ and Evaluate$^{O}(v, c; L)$.

Lemma 5.6 (local correctness of Evaluate$^{O}(v; L)$ and Evaluate$^{O}(v, c; L)$)**.** *Given as input a q-coloring instance $G = (V, E)$. For any vertex $v \in V$, color $c \in [q]$ and a list of available colors $L \subseteq [q]$ satisfying $|L| \geq 50\Delta$ where $\Delta \geq 1$ is the maximum degree of G, assuming $O(u)$ for each $u \in N(v)$ and $O(u, c)$ for each $u \in N(v)$ and $c \in [q]$ as specified in Definition 5.2 and let $S = \{O(u) \mid u \in N(v)\}$:*

(1) Evaluate$^{O}(v; L)$ *returns a uniform sample from $L \setminus S$;*
(2) Evaluate$^{O}(v, c; L)$*:*
- *returns 0 if $c \in S$;*
- *returns 1 with probability $\frac{1}{|L\setminus S|}$ and 0 with probability $1 - \frac{1}{|L\setminus S|}$ otherwise.*

Proof. For Evaluate$^{O}(v; L)$, by $|L| \geq 50\Delta$, Lines 2-5 of Algorithm 9, and Definition 5.2, Evaluate$^{O}(v; L)$ always returns a value from $L \setminus S$. Then Item 1 follows from the symmetry of all colors in $L \setminus S$.

It is direct that Algorithm 11 returns 0 when $c \in S$. Also, according to that $|L| \geq 50\Delta$ and the correctness guarantee of the Bernoulli factory algorithm (Lemma 5.4), Algorithm 11 indeed returns 1 with probability $\frac{1}{|L\setminus S|}$ and 0 with the remaining probability when $c \notin S$. Item 2 then directly follows. □

Similar to the proof of correctness for our local sampler for spin systems with soft constraints, we introduce a finite-time variant of Algorithm 6 for any finite $T > 0$, denoted as Algorithm 12. This algorithm locally reconstructs the final state X_0 of $\mathcal{P}(T)$. Still, the only difference between Algorithms 6 and 12 lies in the initialization of the mapping M.

Algorithm 12: LocalSample$_T(\Lambda; M, L)$ (for q-colorings)

Input: a q-coloring instance $G = (V, E)$, a subset of variables $\Lambda \subseteq V$
Output: A random configuration $X \in [q]^{\Lambda}$
Global variables: Two mappings $M : \mathbb{Z} \to [q] \cup \{\perp\}$, $L : \mathbb{Z} \to 2^{[q]}$.
1 $X \leftarrow \varnothing$, $M(t) \leftarrow X_{-T}(v_{i(t)})$ for each $t \leq -T$, $M(t) \leftarrow \perp$ for each $t > -T$, $L \leftarrow [q]^{\mathbb{Z}}$;
2 **forall** $v \in \Lambda$ **do**
3 $\quad X(v) \leftarrow$ Resolve(pred$_0(v)$; M, L);
4 **return** X;

We then present the crucial lemma for the correctness of the finite-time version of the algorithm. For any finite $T > 0$ and $-T \leq t \leq 0$, we let $X_{T,t}$ be the state of X_t in $\mathcal{P}(T)$. The following lemma shows LocalSample$_T$ indeed simulates $\mathcal{P}(T)$ for q-colorings.

Lemma 5.7. *Suppose that the input q-coloring instance $G = (V, E)$ satisfies $q \geq 50\Delta$ where $\Delta \geq 1$ is the maximum degree of G. For any $\Lambda \subseteq V$, the value returned by* LocalSample$_T(\Lambda)$ *is identically distributed as $X_{T,0}(\Lambda)$.*

Proof. We claim that after the initialization step (Line 1) of Algorithm 12, it is possible to couple the randomness of the algorithm and the process $(X_t)_{-T\leq t\leq 0}$ in such a way that the following two invariants hold for any $T \geq 0$:

(1) Whenever Check(t, c) is called for some timestamp $-T \leq t \leq 0$ and some color $c \in [q]$, the outcome is 1 if and only if $X_t(v_{i(t)}) = c$, where we define $X_t = X_{-T}$ for all $t \leq -T$;
(2) Whenever Resolve(t) is called for some timestamp $-T \leq t \leq 0$, its output is $X_t(v_{i(t)})$, where again $X_t = X_{-T}$ for all $t \leq -T$.

Once such a coupling is established, the lemma immediately follows.

We then verify the existence of this coupling by induction on the length of the process, T.

The base case is when $T = 0$, and the invariants hold by construction due to the initialization of the mapping M in Algorithm 6.

Assume the claim holds for some $T - 1 \geq 0$. Consider the case for T. We rely on two facts:

- The process $(X_t)_{-T\leq t\leq 0}$ forms a Markov chain.
- Any recursive call to Resolve(t) or Check(t, c) only involves timestamps $t' < t$.

These ensure that we can apply the induction hypothesis to couple the randomness for all timestamps strictly less than $t = 0$.

We now extend the coupling to the timestamp $t = 0$. Let $v = v_{i(0)}$, and let $S = \bigcup_{u\in N(v)} X_0(v_{i(\text{pred}_u(0))})$, i.e., S is the set of colors assigned to the neighbors of v in the configuration X_0. By the transition rule of the Markov chain, the value $X_0(v)$ is then sampled uniformly from $[q] \setminus S$.

Under the inductive coupling at times $t < 0$, the oracle values $O(u)$ returned in any call to Check$(0, c)$ or Resolve(0) are exactly $X_0(v_{i(\text{pred}_u(0))})$. Thus, we can extend the coupling to $t = 0$ as follows.

Maintain a local list L', initially equal to $[q]$, and ensure that L' and the global list $L(0)$ used in the algorithm remain synchronized throughout (meaning the invariant $L' = L(0)$ always holds):

- Whenever Resolve(0) is called:

  - If $M(0) \neq \perp$, the output is deterministic and no further randomness needs to be coupled;
  - Otherwise, by $q \geq 50\Delta$ and Lines 8–9 of $\mathsf{Check}(t, c)$, we have $|L(0)| \geq 50\Delta$. By Item 1 of Lemma 5.6, we can couple the randomness so that the outcome of $\mathsf{Resolve}(0)$ is exactly $X_0(v_{i(\mathsf{pred}_u(0))})$.
- Whenever $\mathsf{Check}(0, c)$ is called for some $c \in [q]$:
  - If $M(0) \neq \perp$ or $c \notin L(0)$, the output is deterministic and no further randomness needs to be coupled;
  - Otherwise, if $|L(0)| \leq 50\Delta$, $\mathsf{Resolve}(0)$ is called instead, and this reduces to the case we already proved;
  - Otherwise we have $|L(0)| \geq 50\Delta$. By Item 2 of Lemma 5.6, we can couple the randomness so that:
    * If $\mathsf{Check}(0, c)$ returns 1, let $X_0(v_{i(\mathsf{pred}_u(0))}) = c$ (both happens with probability $\frac{1}{|L' \setminus S|} = \frac{1}{|L(0) \setminus S|}$);
    * Otherwise, remove c from L' (both happens with probability $1 - \frac{1}{|L' \setminus S|} = 1 - \frac{1}{|L(0) \setminus S|}$). As c is also removed from $L(0)$ in this case, L' and $L(0)$ stay the same.

This completes the construction of the coupling, thereby proving the lemma. □

Now, we are finally ready to prove Lemma 5.5.

Proof of Lemma 5.5. For any $T \geq 0$, we couple the randomness of $\mathsf{LocalSample}(\Lambda)$ with $\mathsf{LocalSample}_T(\Lambda)$ by pre-sampling all random variables used within $\mathsf{Resolve}(t)$ and $\mathsf{Check}(t, c)$ for each $t \leq 0$. Here, the coupling fails if and only if recursion reaches some timestamp $t' \leq -T$. Note that when $\mathsf{LocalSample}_T(\Lambda)$ terminates almost surely, the probability of any recursion within $\mathsf{LocalSample}_T(\Lambda)$ reaching some timestamp $t' \leq -T$ must tend to 0 as T tends to infinity, meaning the coupling above gives us

$$\lim_{T \to \infty} d_{\mathrm{TV}} \left(\mathsf{LocalSample}(\Lambda), \mathsf{LocalSample}_T(\Lambda) \right) = 0. \tag{13}$$

Note that when $q \geq \Delta + 1$, we have $\mathcal{P}(T)$ is irreducible and Theorem 2.1 implies that

$$\lim_{T \to \infty} d_{\mathrm{TV}} \left(\mu_\Lambda, X_{T,0}(\Lambda) \right) = 0, \tag{14}$$

Therefore,

$$\begin{aligned}
& d_{\mathrm{TV}} \left(\mu_\Lambda, \mathsf{LocalSample}(\Lambda) \right) \\
\text{(by triangle inequality)} \quad & \leq \limsup_{T \to \infty} d_{\mathrm{TV}} \left(\mu_\Lambda, \mathsf{LocalSample}_T(\Lambda) \right) \\
& + \limsup_{T \to \infty} d_{\mathrm{TV}} \left(\mathsf{LocalSample}_T(\Lambda), \mathsf{LocalSample}(\Lambda) \right) \\
\text{(by (13))} \quad & = \limsup_{T \to \infty} d_{\mathrm{TV}} \left(\mu_\Lambda, \mathsf{LocalSample}_T(\Lambda) \right) \\
\text{(by triangle inequality)} \quad & \leq \limsup_{T \to \infty} d_{\mathrm{TV}} \left(\mu_\Lambda, X_{T,0}(\Lambda) \right) + \limsup_{T \to \infty} d_{\mathrm{TV}} \left(X_{T,0}(\Lambda), \mathsf{LocalSample}_T(\Lambda) \right) \\
\text{(by Lemma 5.7)} \quad & = \limsup_{T \to \infty} d_{\mathrm{TV}} \left(\mu_\Lambda, X_{T,0}(\Lambda) \right) \\
\text{(by (14))} \quad & = 0,
\end{aligned}$$

concluding the proof of the lemma. □

5.3. **Efficiency of the local sampler.** Now, we proceed to proving the efficiency of our local sampler for q-colorings. Our proof is done by designing a carefully-chosen potential function that relates to the state of the algorithm, and showing that such a potential function decays in expectation at each step as the algorithm evolves.

Lemma 5.8 (efficiency of the local sampler for q-colorings)**.** *Suppose that the input q-coloring instance $G = (V, E)$ satisfies $q \geq 65\Delta$ where $\Delta \geq 1$ is the maximum degree of G. Then for any subset $\Lambda \subseteq V$,* $\mathsf{LocalSample}(\Lambda)$ *terminates almost surely within expected time $|\Lambda| \cdot \Delta^2 q$.*

Proof. Following the proof of Lemma 5.7, we can couple the randomness of the algorithm and $(X_t)_{t \leq 0}$ in such a way that the following two invariants hold:

(1) Whenever $\mathsf{Check}(t, c)$ is called for some timestamp $t \le 0$, some color $c \in [q]$ and *terminates*, the outcome is 1 if and only if $X_t(v_{i(t)}) = c$;
(2) Whenever $\mathsf{Resolve}(t)$ is called for some timestamp $t \le 0$ and *terminates*, its output is $X_t(v_{i(t)})$.

Note that here $(X_t)_{t\le 0}$ is viewed as the limiting process of $(X_t)_{-T\le t\le 0}$ as $T \to \infty$. When $q \ge \Delta + 1$, the Markov chain $\mathcal{P}(T)$ is ergodic, and the distribution specified by $(X_t)_{t\le 0}$ projecting onto any finite subset of timestamps is always well-defined.

We then strengthen the lemma and prove that under any realization of the outcome of updates in $(X_t)_{t\le 0}$, the desired result that for any $\Lambda \subseteq V$, $\mathsf{LocalSample}(\Lambda)$ terminates almost surely within expected time $|\Lambda| \cdot \Delta^2 q$ always holds.

Note that the state Π of the algorithm $\mathsf{LocalSample}(\Lambda)$ can be defined by:

- the state of the global data structures M and L, and
- the current stack of recursive calls to $\mathsf{Resolve}(t)$ and $\mathsf{Check}(t, c)$.

We introduce a potential function to track the state of the algorithm.

Definition 5.9 (potential function associated with the state of the algorithm)**.** Let $\Phi = \Phi(\Pi) \in \mathbb{R}$ be the potential function, defined as follows. Initially, $\Phi = 0$.

- For each *distinct* $\mathsf{Resolve}(t)$ call such that $M(t) = \perp$ in the current stack of the algorithm, update $\Phi \leftarrow \Phi + C_1\Delta$.
- For each *distinct* $\mathsf{Check}(t, c)$ call such that $M(t) = \perp$ and $c \in L(t)$ in the current stack of the algorithm, update $\Phi \leftarrow \Phi + C_2$.
- For each $t \in \mathbb{Z}$ such that $M(t) = \perp$ and *there is currently no* $\mathsf{Resolve}(t)$ *call in the stack*, update $\Phi \leftarrow \Phi + C_3 \cdot (65\Delta - |L|)$; that is, add C_3 to the potential function for each removed color in the list of an yet undetermined timestamp.

Here, C_1, C_2, C_3 are universal constants to be determined later in this proof.

At the start of the algorithm, for each $v \in \Lambda$, the only call is $\mathsf{Resolve}(\mathsf{pred}_0(v))$. Thus, the potential function is initially set to $\Phi = |\Lambda| \cdot C_1\Delta$, as given by Definition 5.9. We will show that Φ contracts in expectation throughout the algorithm's execution.

Without loss of generality, we only consider *active* calls of the form:

- $\mathsf{Resolve}(t)$ such that $M(t) = \perp$;
- $\mathsf{Check}(t, c)$ such that $M(t) = \perp$ and $c \in L(t)$,

as the remaining calls directly terminate by memoization, and we attribute the running time of such calls to the procedure that incurred them. The algorithm terminates when there are no active $\mathsf{Resolve}(t)$ or $\mathsf{Check}(t, c)$ calls remaining in the stack.

We also decompose the execution of the algorithm into *steps*, where in each step, we reveal certain random choices made by the algorithm, and the order of these random choices may differ from the execution order of the algorithm. We analyze the expected change of Φ in several cases.

- Suppose the top of the stack is a $\mathsf{Resolve}(t)$ call for some $t \in \mathbb{Z}$ such that $M(t) = \perp$. This call proceeds by calling $\mathsf{Evaluate}^{\mathcal{O}}(v_{i(t)}; L(t))$. We define each iteration of the **while** loop in $\mathsf{Evaluate}^{\mathcal{O}}(v_{i(t)}; L(t))$ as a single *step*, and we track the expected change in Φ after each step.

  Since $|L| = |L(t)| \ge 50\Delta$ at the beginning, and $|S_t|$ has size at most Δ, we always have $|L| \ge 49\Delta$ at the start of any iteration. Therefore, the condition in Line 3 of $\mathsf{Evaluate}^{\mathcal{O}}(v_{i(t)}; L(t))$ is satisfied with probability at least $\frac{49}{50}$, regardless of the realization of S_t. This allows the number of iterations to be stochastically dominated by $\mathrm{Geo}(\frac{49}{50})$.

  Hence, in each step:
  - Δ calls to $\mathsf{Check}(t, c)$ are added to the stack, increasing the potential function by $C_2\Delta$.
  - With probability at least $\frac{49}{50}$, the $\mathsf{Resolve}(t)$ call is removed from the stack, decreasing the potential function by $C_1\Delta$.

  We remark that during these steps, the potential function is not affected by the change in the size of $L(t)$ as $\mathsf{Resolve}(t)$ is already in the stack of the algorithm.

Therefore, the expected change in the potential function per step is at most:

$$C_2\Delta - \frac{49}{50} \cdot C_1\Delta.$$

- Suppose the top of the stack is a $\mathsf{Check}(t, c)$ call for some $t \in \mathbb{Z}$ and $c \in [q]$ such that $M(t) = \perp$ and $c \in L(t)$.
  - If $|L(t)| = 50\Delta$, then the $\mathsf{Check}(t, c)$ call is removed from the stack, decreasing the potential function by C_2.
    - If there currently is a $\mathsf{Resolve}(t)$ call in the stack, then the total change in the potential function is simply $-C_2$.
    - Otherwise, a new $\mathsf{Resolve}(t)$ call is added to the stack, increasing the potential function by $C_1\Delta$. Additionally, since the removed color in the list $L(t)$ no longer contributes to the potential function when this new $\mathsf{Resolve}(t)$ call is added, the potential function further decreases by $C_3 \cdot (65\Delta - 50\Delta) = 15\Delta \cdot C_3$. Therefore, the total change in the potential function is:

$$C_1\Delta - C_2 - 15\Delta \cdot C_3.$$

  - Otherwise, we assume $M(t) \neq \perp, c \in L(t)$ and that $|L(t)| > 50\Delta$. This call proceeds by calling $\mathsf{Evaluate}^O(v_{i(t)}, c; L(t))$. According to $\mathsf{Evaluate}^O(v_{i(t)}, c; L(t))$:
    - With probability $1 - \frac{2}{|L|} \geq 1 - \frac{1}{25\Delta}$, $\mathsf{Evaluate}^O(v_{i(t)}, c; L(t))$ directly terminates and returns 0. In this case, the $\mathsf{Check}(t, c)$ call is removed from the stack, decreasing the potential function by C_2. Also, due to Line 14 of $\mathsf{Check}(t, c)$, c is removed from $L(t)$, increasing the potential function by at most C_3. Hence, in this case, the change in the potential function is at most $C_3 - C_2$.
    - With probability $\frac{2}{|L|} \leq \frac{1}{25\Delta}$, $\mathsf{Evaluate}^O(v_{i(t)}, c; L(t))$ first adds at most Δ recursive calls of the form $\mathsf{Check}(t', c')$, then uses a Bernoulli factory algorithm for division to return 1 with probability $\frac{|L|/2}{|L \setminus S_t|}$, accessing Algorithm 11 as an $|L \setminus S_t|/|L|$-probability coin, where S_t is defined in (12). Note that after fixing the realization of $(X_t)_{t \leq 0}$, the outcome of Algorithm 11 is solely determined by Line 1, which are clearly independent of the randomness within recursive calls to Line 1. Therefore, we can first determine the randomness within this Bernoulli factory algorithm and consider this as a *step*. According to Lemma 5.4, the expected number of calls to Algorithm 11 is at most 21. Each such call may in turn generate up to Δ recursive calls of the form $\mathsf{Check}(t', c')$, with each recursive call contributing at most C_2 to the potential function. At the same time, the original $\mathsf{Check}(t, c)$ call is removed from the stack, decreasing the potential function by C_2; and, due to Line 14, the color c is removed from $L(t)$, increasing the potential function by C_3. Thus, the total expected change in the potential function in this case is at most:

$$(22\Delta - 1)C_2 + C_3.$$

      Moreover, since at most Δq distinct recursive calls of the form $\mathsf{Check}(t', c')$ can be generated where $t' = \mathsf{pred}_u(t)$ for some $u \in N(v_{i(t)})$, the maximum change in the potential function is at most:

$$(\Delta q - 1)C_2 + C_3.$$

    Summarizing, in this case, the expected change in the potential function per step is at most:

$$\left(1 - \frac{1}{25\Delta}\right) \cdot (C_3 - C_2) + \frac{1}{25\Delta} \cdot ((22\Delta - 1)C_2 + C_3) = C_3 - \frac{3C_2}{25}.$$

We then set the constants as follows:

$$C_1 = 30, \quad C_2 = 25, \quad C_3 = 2.$$

In this case, it is easy to see that the expected change of the potential function after each step is at most -1. Let $\Phi_0, \Phi_1, \ldots$ be the random sequence of the potential function after the i-th step of the

algorithm. Define $\Psi_i = \Phi_i + i$ for each $i \geq 0$. By the previous analysis, we have $\{\Psi_i\}_{i\geq 0}$ forms a supermartingale with each absolute increment $|\Psi_{i+1} - \Psi_i|$ upper bounded by some $K = \mathrm{poly}(q, \Delta)$. Let $\tau = \min\{i > 0 \mid \Phi_i = 0\} = \min\{i > 0 \mid \Psi_i = i\}$ be a stopping time. By the Azuma-Hoeffding inequality, we have for each fixed $\varepsilon > 0$,

$$\mathbf{Pr}\left[\tau > (1+\varepsilon)30|\Lambda|\cdot\Delta\right] \leq \mathbf{Pr}\left[\Psi_{(1+\varepsilon)30|\Lambda|\cdot\Delta} > \Psi_0 + 30\varepsilon|\Lambda|\cdot\Delta\right] \leq \exp\left(\frac{-30\varepsilon^2|\Lambda|\Delta}{(1+\varepsilon)K^2}\right),$$

from which we can conclude that $\mathbb{E}[\tau] < \infty$, allowing us to apply Doob's optional stopping theorem:

$$30|\Lambda|\cdot\Delta = \mathbb{E}[\Psi_0] \geq \mathbb{E}[\Psi_\tau] = \mathbb{E}[\tau],$$

meaning that the algorithm executes at most $30|\Lambda|\cdot\Delta$ steps in expectation. Consequently, by Lemma 5.4, the expected running time of the algorithm is bounded by $O(|\Lambda|\cdot\Delta q^2)$. □

Lemmas 5.6 and 5.8 together establish Theorem 1.4. For Theorem 1.7, observe that all algorithms and proofs in this section naturally extend to the setting of list colorings, where each vertex $v \in V$ has its own color list Q_v satisfying $|Q_v| \geq 65\Delta$. By applying the same argument as in the proof of Theorem 1.6, Theorem 1.7 follows.

6. Conclusions and Open Problems

In this paper, we design new local samplers that go beyond the use of the local uniformity property by generalizing and refining the framework of backward deduction of Markov chains, i.e., the "Coupling Towards The Past" (CTTP) method. Specifically, we design the first local samplers for both spin systems with soft constraints in near-critical regimes, and uniform q-coloring under the near-critical condition of $q = O(\Delta)$ where Δ is the maximum degree of the graph. The proposed local samplers are perfect, achieve near-linear runtime, and admit direct applications to local algorithms for probabilistic inference within the same parameter regimes.

We leave the following open problems and directions for future work:

- While our local sampler performs well in near-critical regimes for spin systems using backward deduction of Glauber dynamics, it has been shown that forward simulation of Glauber dynamics mixes rapidly for the Ising model up to the uniqueness threshold. Can we improve the analysis of our current algorithm, design new local samplers that are efficient up to this critical threshold, or prove a lower bound showing that this is intractable for local samplers?
- Both the CTTP (Coupling Towards The Past) and the CFTP (Coupling From The Past) approaches are based on grand coupling and can yield perfect sampling algorithms. However, the local sampler via CTTP requires a more restrictive grand coupling that admits a local implementation. For q-colorings, we show that CTTP with such a local grand coupling can indeed achieve the $q = O(\Delta)$ threshold; in particular, we establish $q \geq 65\Delta$. In contrast, the best known result for CFTP with a global grand coupling is $q \geq (\frac{8}{3} + o(1))\Delta$, as achieved in [JSS21], which attains a better constant factor. This raises a natural question: can one obtain a local sampler that matches or even surpasses the global grand coupling threshold?
- Local samplers, as introduced in [AJ22, FGW+23], have found a wide range of applications, as discussed in the introduction. Our work further shows that local samplers can also imply efficient local algorithms for probabilistic inference. We hope to see more applications of local samplers, particularly in the design of distributed and parallel algorithms.

Acknowledgement

Chunyang would like to thank Jingcheng Liu and Yixiao Yu for pointing out Lubetzky and Sly's result, which analyzes the cutoff phenomenon of the Ising model in near-critical regimes.

References

[AFF+24] Konrad Anand, Weiming Feng, Graham Freifeld, Heng Guo, and Jiaheng Wang. Approximate Counting for Spin Systems in Sub-Quadratic Time. In *ICALP*, volume 297, pages 11:1–11:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

[AFG+25] Konrad Anand, Graham Freifeld, Heng Guo, Chunyang Wang, and Jiaheng Wang. Sink-free orientations: a local sampler with applications, 2025. to appear in RANDOM 2025.

[AGPP23] Konrad Anand, Andreas Göbel, Marcus Pappik, and Will Perkins. Perfect sampling for hard spheres from strong spatial mixing. In *RANDOM*, volume 275 of *LIPIcs*, pages 38:1–38:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[AJ22] Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *SIAM Journal on Computing*, 51(4):1280–1295, 2022.

[AJK+22] Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence: optimal mixing of down-up random walks. In *STOC*, pages 1418–1430. ACM, 2022.

[ALO20] Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *FOCS*, pages 1319–1330. IEEE, 2020.

[BC20] Siddharth Bhandari and Sayantan Chakraborty. Improved bounds for perfect sampling of k-colorings in graphs. In *STOC*, pages 631–642, 2020.

[BPR22] Amartya Shankha Biswas, Edward Pyne, and Ronitt Rubinfeld. Local Access to Random Walks. In *ITCS*, volume 215, pages 24:1–24:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[BRY20] Amartya Shankha Biswas, Ronitt Rubinfeld, and Anak Yodpinyanee. Local access to huge random objects through partial sampling. In *ITCS*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[CDM+19] Sitan Chen, Michelle Delcourt, Ankur Moitra, Guillem Perarnau, and Luke Postle. Improved bounds for randomly sampling colorings via linear programming. In *SODA*, page 2216–2234, USA, 2019. SIAM.

[CE22] Yuansi Chen and Ronen Eldan. Localization schemes: A framework for proving mixing bounds for markov chains (extended abstract). In *FOCS*, pages 110–122. IEEE, 2022.

[CFYZ21] Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Rapid mixing of Glauber dynamics via spectral independence for all degrees. In *FOCS*, pages 137–148, 2021.

[CFYZ22] Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Optimal mixing for two-state anti-ferromagnetic spin systems. In *FOCS*, pages 588–599. IEEE, 2022.

[CLV20] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. In *FOCS*, pages 1307–1318. IEEE, 2020.

[CLV21] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of Glauber dynamics: entropy factorization via high-dimensional expansion. In *STOC*, pages 1537–1550. ACM, 2021.

[CV25] Charlie Carlson and Eric Vigoda. Flip dynamics for sampling colorings: Improving $(11/6 - \varepsilon)$ using a simple metric. In *SODA*, pages 2194–2212, 2025.

[DL93] Paul Dagum and Michael Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artif. Intell.*, 60(1):141–153, 1993.

[DL97] Paul Dagum and Michael Luby. An optimal approximation algorithm for bayesian inference. *Artif. Intell.*, 93(1–2):1–27, 1997.

[FGW+23] Weiming Feng, Heng Guo, Chunyang Wang, Jiaheng Wang, and Yitong Yin. Towards derandomising Markov chain Monte Carlo. In *FOCS*, pages 1963–1990. IEEE, 2023.

[GŠV16] Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *Combinatorics, Probability and Computing*, 25(04):500–559, 2016.

[Hub98] Mark Huber. Exact sampling and approximate counting techniques. In *STOC*, pages 31–40, 1998.

[Hub16] Mark Huber. Nearly optimal Bernoulli factories for linear functions. *Combin. Probab. Comput.*, 25(4):577–591, 2016.

[HWY22] Kun He, Chunyang Wang, and Yitong Yin. Sampling lovász local lemma for general constraint satisfaction solutions in near-linear time. In *FOCS*, pages 147–158. IEEE, 2022.

[HWY23] Kun He, Chunyang Wang, and Yitong Yin. Deterministic counting lovász local lemma beyond linear programming. In *SODA*, pages 3388–3425. SIAM, 2023.

[Isi25] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31:253–258, 1925.

[Jer95] Mark Jerrum. A very simple algorithm for estimating the number of k-colorings of a low-degree graph. *Random Struct. Algorithms*, 7(2):157–165, September 1995.

[JSS21] Vishesh Jain, Ashwin Sah, and Mehtaab Sawhney. Perfectly sampling k¿(8/3+o(1)) δ-colorings in graphs. In *STOC*, pages 1589–1600, 2021.

[LPW17] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, Providence, RI, 2017.

[LS16] Eyal Lubetzky and Allan Sly. Information percolation and cutoff for the stochastic Ising model. *J. Amer. Math. Soc.*, 29(3):729–774, 2016.

[LS17] Eyal Lubetzky and Allan Sly. Universality of cutoff for the Ising model. *The Annals of Probability*, 45(6A):3664 – 3696, 2017.

[Mor21] G. Morina. *Extending the Bernoulli Factory to a Dice Enterprise*. University of Warwick, 2021.

[MSW22] Peter Mörters, Christian Sohler, and Stefan Walzer. A Sublinear Local Access Implementation for the Chinese Restaurant Process. In *RANDOM*, volume 245, pages 28:1–28:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[NP05] Şerban Nacu and Yuval Peres. Fast simulation of new coins from old. *Ann. Appl. Probab.*, 15(1A):93–115, 2005.

[PW96] James Gary Propp and David Bruce Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252, 1996.

[SS14] Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Ann. Probab.*, 42(6):2383–2416, 2014.

[Vig99] Eric Vigoda. Improved bounds for sampling colorings. In *FOCS*, pages 51–59, 1999.

[VN51] John Von Neumann. various techniques used in connection with random digits. *Appl. Math Ser*, 12(36-38):3, 1951.

Appendix A. Bernoulli factory method for simulating probability

In this section, we explicitly provide the construction of the Bernoulli factory for division, and formally prove Lemma 5.4.

For $\xi \in [0,1]$ we denote by $\mathcal{O}_\xi$ a coin (or oracle) that returns 1 (heads) with probability ξ, and 0 (tails) with probability $1-\xi$, independently on each call.

Our construction of the Bernoulli factory for division is based on the method from [Mor21], which in turn builds on the subtraction Bernoulli factory from [NP05]. That construction is itself based on a linear Bernoulli factory, for which we adopt the implementation from [Hub16].

We first introduce the linear Bernoulli factory, which transforms $\mathcal{O}_\xi$ to $\mathcal{O}_{C\xi}$ for a $C > 1$ with the promise that $C\xi \le 1$. We adopt the construction of linear Bernoulli factory in [Hub16] described in Algorithm 13. Its correctness and efficiency is guaranteed as follows.

Algorithm 13: LinearBF$(\mathcal{O}, C, \zeta)$[Hub16]

Input: a coin $\mathcal{O} = \mathcal{O}_\xi$ with unknown ξ, $C > 1$ and a slack $\zeta > 0$, with promise that $C\xi \le 1 - \zeta$;
Output: a random value from $\mathcal{O}_{C\xi}$;

$k \leftarrow 4.6/\zeta, \zeta \leftarrow \min\{\zeta, 0.644\}, i \leftarrow 1$;
repeat
  repeat
    draw $B \leftarrow \mathcal{O}$, $G \leftarrow \mathrm{Geo}\left(\frac{C-1}{C}\right)$;
    // G is drawn according to geometric distribution with parameter $\frac{C-1}{C}$
    $i \leftarrow i - 1 + (1-B)G$;
  until $i = 0$ *or* $i \ge k$;
  if $i \ge k$ **then**
    draw $R \leftarrow \mathrm{Bernoulli}\left((1+\zeta/2)^{-i}\right)$;
    $C \leftarrow C(1+\zeta/2), \zeta \leftarrow \zeta/2, k \leftarrow 2k$;
until $i = 0$ *or* $R = 0$;
return $\mathbb{1}[i = 0]$;

Proposition A.1 ([Hub16, Theorem 1]). *Given access to a coin $\mathcal{O}_\xi$, given as input $C > 1$ and $\zeta > 0$, with promise that $C\xi \le 1 - \zeta$,* LinearBF$(\mathcal{O}, C, \zeta)$ *terminates with probability 1 and returns a draw of $\mathcal{O}_{C\xi}$. Moreover, the expected number of calls to $\mathcal{O}_\xi$ is at most $9.5C\zeta^{-1}$.*

Building upon the linear Bernoulli factory, a subtraction Bernoulli factory SubtractBF$(\mathcal{O}_{\xi_1}, \mathcal{O}_{\xi_2}, \zeta)$ was proposed in [NP05]. It transforms two coins $\mathcal{O}_{\xi_1}$ and $\mathcal{O}_{\xi_2}$—under the promise that $\xi_1 - \xi_2 \ge \zeta > 0$—into a new coin $\mathcal{O}_{\xi_1-\xi_2}$.

We implement this procedure using the linear Bernoulli factory defined above:

- SubtractBF$\left(\mathcal{O}_{\xi_1}, \mathcal{O}_{\xi_2}, \zeta\right) = 1 -$ LinearBF$\left(\mathcal{O}_{(1-\xi_1+\xi_2)/2}, 2, \zeta\right)$,

where the coin $\mathcal{O}_{(1-\xi_1+\xi_2)/2}$ is realized using $\mathcal{O}_{1/2}$, $\mathcal{O}_{\xi_1}$, and $\mathcal{O}_{\xi_2}$ as follows: if $\mathcal{O}_{1/2} = 1$, return $1 - \mathcal{O}_{\xi_1}$; otherwise, return $\mathcal{O}_{\xi_2}$.

The correctness and efficiency of this procedure is guaranteed as follows.

Corollary A.2 (c.f. [NP05, Proposition 14, (iv)]). *Given access to two coins $\mathcal{O}_{\xi_1}$ and $\mathcal{O}_{\xi_2}$, and given as input $\zeta > 0$, with promise that $\xi_1 - \xi_2 \ge \zeta$,* SubtractBF$\left(\mathcal{O}_{\xi_1}, \mathcal{O}_{\xi_2}, \zeta\right)$ *terminates with probability 1 and returns a draw of $\mathcal{O}_{\xi_1-\xi_2}$. Moreover, the expected number of call to $\mathcal{O}_\xi$ and $\mathcal{O}_{\xi_2}$ is at most $9.5\zeta^{-1}$ each.*

Now we can formally give the construction of the Bernoulli factory for division in [Mor21], BernoulliDivision$(\mathcal{O}_\xi, p, \zeta)$, which transforms a coin $\mathcal{O}_\xi$ with the promise that $\xi - p \ge \zeta$ into a new coin $\mathcal{O}_{p/\zeta}$. The construction is given as follows [Mor21, Algorithm 9].

Repeat the following until a value is returned:

- If a draw I from $\mathcal{O}_{1/2}$ is 1: return 1 with probability p.
- Otherwise, return 0 if a draw from $\mathcal{O}_{\xi-p}$ returns 1,

where $O_{\xi-p}$ is implemented using SubtractBF (O_ξ, O_p, ζ).

The correctness and efficiency of this procedure is guaranteed as follows, which directly proves Lemma 5.4.

Corollary A.3 (c.f. [Mor21, Proposition 2.24])**.** *Given access to a coin O_ξ, and given as input $0 \le p \le 1, \zeta > 0$, with promise that $\xi - p \ge \zeta$,* BernoulliDivision (O_ξ, p, ζ) *terminates with probability 1 and returns a draw of $O_{p/\xi}$. Moreover, the expected number of calls to O_ξ is at most $9.5\xi^{-1}\zeta^{-1}$.*