TOWARDS DERANDOMISING MARKOV CHAIN MONTE CARLO

WEIMING FENG, HENG GUO, CHUNYANG WANG, JIAHENG WANG, YITONG YIN

ABSTRACT. We present a new framework to derandomise certain Markov chain Monte Carlo (MCMC) algorithms. As in MCMC, we first reduce counting problems to sampling from a sequence of marginal distributions. For the latter task, we introduce a method called *coupling towards the past* that can, in logarithmic time, evaluate one or a constant number of variables from a stationary Markov chain state. Since there are at most logarithmic random choices, this leads to very simple derandomisation. We provide two applications of this framework, namely efficient deterministic approximate counting algorithms for hypergraph independent sets and hypergraph colourings, under local lemma type conditions matching, up to lower order factors, their state-of-the-art randomised counterparts.

CONTENTS

1. INTRODUCTION

It is a central question in the theory of computing to understand the power of randomness. Indeed, randomisation has been shown to be extremely useful in designing efficient algorithms. One early and surprising illustration of its power is through estimating the volume of a convex body. Deterministic algorithms cannot achieve good approximation ratios through membership queries [Ele86, BF87], and yet Dyer, Frieze, and Kannan [DFK91] discovered an efficient randomised approximation algorithm under the same model. Their driving force is the celebrated Markov chain Monte Carlo (MCMC) method, which has been studied since the origin of electronic computers. In MCMC, one reduces the counting problem (such as computing the volume) to (usually a sequence of) related sampling problems [JVV86], and the latter is solved using Markov chains. This powerful method has lead to many great achievements, ranging from the early results of approximating the partition function of ferromagnetic Ising models [JS93] and the permanent of non-negative matrices [JSV04], to more recent developments such as counting the number of bases in matroids [ALOV19, CGM21], and estimating partition functions of spin systems up to critical thresholds [ALO20, CLV20, CLV21, CFYZ21, AJK+22, CE22, CFYZ22].

While randomness remains an indispensable ingredient to the MCMC method, the belief that randomness is essential to efficient approximate counting is seriously challenged over the past two decades. This was initiated by a highly influential result of Weitz [Wei06], which gave the first deterministic fully

(Weiming Feng, Heng Guo, Jiaheng Wang) SCHOOL OF INFORMATICS, UNIVERSITY OF EDINBURGH, INFORMATICS FORUM, EDINBURGH, EH8 9AB, UNITED KINGDOM. E-mail: wfeng@ed.ac.uk, hguo@inf.ed.ac.uk, jiaheng.wang@ed.ac.uk.

(Chunyang Wang, Yitong Yin) STATE KEY LABORATORY FOR NOVEL SOFTWARE TECHNOLOGY, NANJING UNIVERSITY, 163 XIANLIN AVENUE, NANJING, JIANGSU PROVINCE, 210023, CHINA. E-mail: wcysai@smail.nju.edu.cn, yinyt@nju.edu.cn

polynomial-time approximation scheme (FPTAS) for a #**P**-hard problem. Since then, deterministic algorithms are gradually catching up with their random counterparts on many fronts. A plethora of techniques have been introduced and developed for deterministic approximate counting, including: decay of correlation [Wei06, BGK⁺07, GK07], zero-freeness of polynomials [Bar16, PR17, LSS22], linear programming based methods [Moi19, GLLZ19, JPV21b], and various statistical physics related techniques [HPR19, JPP22, JPSS22]. Curiously, although these algorithms are usually guided by probabilistic intuitions related to the problem, they work very differently from typical randomised algorithms.

All these developments beg one question: can we derandomise MCMC more directly? The benefit of a positive answer would be two folds. It can help us to better understand the role of randomisation in MCMC and in approximate counting. It might also lead to easier ways of designing deterministic counting algorithms thanks to the plentifulness of MCMC algorithms. However, this task is not easy, as Markov chains require at least a linear amount of random bits to approach their stationary distribution. Short of a breakthrough in pseudo-random number generators, it seems impossible to fully derandomise Markov chains.

In this paper we make a substantial step towards turning rapid mixing Markov chains into deterministic approximate counting algorithms. The saving grace to the issue mentioned above is that the Monte Carlo step of MCMC does not require fully simulating the Markov chains anyways! Usually, only the marginal probability of a single variable needs to be evaluated, rather than the whole state. We show that for certain MCMC algorithms, one or a constant number of variables can be evaluated in their stationary state within only logarithmic time. This leads to some very simple brute force enumeration derandomisation. To illustrate the power of this new method, we obtain efficient deterministic approximate counting algorithms for hypergraph independent sets and hypergraph colourings. Our algorithms match their currently best randomised counterparts under local lemma type conditions up to log factors. We describe our results in more details next.

1.1. **Our contributions.** We give deterministic approximate counting algorithms by derandomising certain MCMC algorithms. In fact, we venture the idea that efficient deterministic approximate counting follows from "highly efficient" randomised algorithms. We first explain what these are and how we obtain them.

1.1.1. *Coupling towards the past.* We consider a class of generic Markov chains known as *systematic scan Glauber dynamics*,[1] which are single-site Glauber dynamics (a.k.a. *Gibbs sampling, heat-bath dynamics*) with a fixed scan order. In classical MCMC, a Markov chain $(X_t)_{t \geq 0}$ is simulated chronologically for long enough to draw a sample according to the stationary distribution $X_\infty \sim \mu$. Due to a lower bound of Hayes and Sinclair [HS07], which applies to a wide range of models, this will cost $\Omega(n \log n)$ random bits if $X_t = (X_t(v))_{v \in [n]}$ consists of n variables. However, this simulation seems very wasteful when we are only interested in evaluating $X_\infty(v)$ for some particular $v \in [n]$. This reflects a more general question: can we calculate the fixed point of a dynamical system without simulating the dynamical system until convergence?

We introduce a new method for evaluating the states of variables in a stationary Markov chain without simulating the entire chain. We call this method *coupling towards the past* (CTTP).[2] Consider a mixed chain $(X_t)_{t \leq 0}$ that runs from the infinite past to now. Our goal is to evaluate $X_0(v)$ for $v \in [n]$, which follows the marginal distribution, denoted μ_v. It suffices to simulate the last update for v, and the key observation here is that updates of Glauber dynamics may depend only on a small amount of information. In particular, when the marginal distributions are properly lower bounded, there is a positive probability of determining the update without any information on the current state. Thus, we can deduce $X_0(v)$ by recursively revealing only the necessary randomness backwards in time. Alternatively, this can be viewed as a grand coupling constructed towards the past. Each random bit revealed is used for all possible chains, and every information successfully deduced is the same for all

---

[1]Instead of random update Markov chains, we choose systematic scan chains to illustrate our ideas, as the analysis is much cleaner and the result is no worse. Our method applies to random update Markov chains as well, but with some additional effort. See Section 1.3 for a brief discussion and Section 7 for the details.

[2]The name resembles the coupling *from* the past (CFTP) method by Propp and Wilson [PW96]. However, our method has several key differences from CFTP. See Section 1.4 for a comparison.

chains as well. When this process terminates at time $-t$ for some $t \geq 0$, no matter what the state before $-t$ is, it evolves into the same $X_0(v)$. This implies that $X_0(v)$ follows μ_v as desired.

When the CTTP process terminates in logarithmic steps with high probability, it yields a *marginal sampler* that draws, approximately, from the marginal distribution with $O(\log n)$ cost in both time and random bits. The error in total variation distance is the failure probability of CTTP. Such marginal samplers can be straightforwardly derandomised by enumerating all possible random choices in polynomial time to deterministically estimate the marginal probabilities, which implies FPTASes via standard self-reductions [JVV86].

We will apply CTTP to the uniform distribution of hypergraph independent sets and a projected distribution induced by uniformly at random hypergraph colourings. In both applications, the conditions we obtain to guarantee the $O(\log n)$ run-time of CTTP match those for $O(n \log n)$ mixing time bounds of Glauber dynamics. From a complexity-theory point of view, our construction of marginal samplers from the original Markov chain algorithm has a certain *direct-sum* flavour: we transform a protocol for sampling n variables, to a new protocol for sampling single variables, using only $O(1/n)$ fraction of the original cost each.

We remark that these low-cost marginal samplers have significance beyond deterministic approximate counting. For example, in probabilistic inferences, it is often the end goal to estimate marginal probabilities. Thus, our marginal samplers are substantially faster in such contexts than standard MCMC. As an instance of such, the perfect marginal sampler we obtain for hypergraph independent sets terminates in $O(\log 1/\gamma)$ time with probability at least $1 - \gamma$ in the regime of parameters we consider (see Remark 5.4). In comparison, other standard methods for such purpose require $\Omega(n)$ running time where n is the number of vertices. See Section 1.4 for a more detailed elaboration on this.

1.1.2. *Counting hypergraph independent sets.* The first testing field of our framework is to approximately count the number of hypergraph independent sets (equivalent to satisfying assignments of monotone CNF formulas). Let $H = (V, \mathcal{E})$ be a hypergraph. A set $S \subseteq V$ is a (weak) *independent set* if $S \cap e \neq e$ for all $e \in \mathcal{E}$. This problem is naturally parameterised by k and Δ, which denote the (uniform) hyperedge size and the maximum vertex degree of H, respectively. There was an exponential gap between the parameters for which efficient randomised and deterministic algorithms exist, and our work closes this gap.

Estimating the number of hypergraph independent sets was first studied by Borderwich, Dyer, and Karpinski [BDK06, BDK08] using Markov chains. They used path coupling to show that the straightforward Glauber dynamics mixes in $O(n \log n)$ time if $\Delta \leq k - 2$. The mixing time analysis was later improved by Hermon, Sly, and Zhang [HSZ19] to extend the condition exponentially to $\Delta \leq c2^{k/2}$ for some absolute constant $c > 0$ via an information percolation argument. This threshold is tight up to constants due to a hardness result in [BGG+19]. Very recently, Qiu, Wang, and Zhang [QWZ22] gave a perfect sampler under similar conditions.

On the other hand, deterministic counting algorithms have been lagging behind for this problem. Even with significant adjustment, the correlation decay method works only if $\Delta \leq k$ [BGG+19]. One may also put it under the Lovász local lemma framework, and apply a more general algorithm by He, Wang, and Yin [HWY23] to obtain an efficient algorithm assuming $\Delta \lesssim 2^{k/5}$. The symbol $\lesssim$ suppresses lower-order items such as $\text{poly}(k)$.[3] The conditions of the latter algorithm still has a gap exponential in k compared to the randomised algorithm or the hardness threshold.

We close this $\exp(k)$ gap using our new framework. The result is summarised in Theorem 1.1, matching the previously mentioned randomised algorithm and hardness threshold up to a factor of $O(k^2)$. We note that one caveat of our algorithm, similar to all other deterministic approximate counting algorithms we are aware of, is that the exponent of its running time depends on Δ and k, rather than an absolute constant as in the case of randomised algorithms. A detailed running time bound of our algorithm is given in Section 5. Comparisons with previous works are summarised in Table 1.

Theorem 1.1. *Let $k \geq 2$ and $\Delta \geq 2$ be two constants satisfying $\Delta \leq \frac{1}{\sqrt{8}ek^2} \cdot 2^{\frac{k}{2}}$. There is an FPTAS for the number of independent sets in k-uniform hypergraphs with maximum degree Δ.*

---

[3] We make sure to suppress same order terms when comparing to other works in Table 1 and Table 2.

Hypergraph independent sets	Reference	Bound	Running time
Randomised counting / sampling	[BDK08, BDK06]	$\Delta \leq k - 2$	$\tilde{O}(n^2) \,/\, O(n \log n)$
	[HSZ19, QWZ22]	$\mathbf{\Delta \lesssim 2^{k/2}}$	$\tilde{O}(n^2) \,/\, O(n \log n)$
Deterministic counting	[BGG$^+$19]	$\Delta \leq k$	$n^{O(\log(k\Delta))}$
	[JPV21b]	$\Delta \lesssim 2^{k/7}$	$n^{\mathrm{poly}(k,\Delta)}$
	[HWY23]	$\Delta \lesssim 2^{k/5}$	$n^{\mathrm{poly}(k,\Delta)}$
	This work	$\mathbf{\Delta \lesssim 2^{k/2}}$	$n^{\mathrm{poly}(k,\Delta)}$
Hardness	[BGG$^+$19]	$\Delta \geq 5 \cdot 2^{k/2}$ assuming $\mathbf{P \neq NP}$	

TABLE 1. Algorithms and hardness results for hypergraph independent sets

This is a relatively straightforward application, since we just apply CTTP to the uniform distribution over hypergraph independent sets. Our run-time analysis incorporates various techniques developed in the local lemma context. The crucial part is to show that when CTTP runs for too long, there must be many independent unlikely events happening.

Comparing to the previous best deterministic algorithm by He, Wang, and Yin [HWY23], our algorithm is both simpler and stronger, thanks to the new CTTP technique. The previous algorithm is a derandomisation of a recursive marginal sampler by Anand and Jerrum [AJ22], which is also a major inspiration for our algorithm. We will discuss it in more detail in Section 1.2. Derandomising the Anand-Jerrum algorithm requires sophisticated dynamic programming, and one needs to carefully control how conditioning goes in the recursive calls, leading to some complicated analysis and worse conditions. In contrast, as CTTP always draws fresh random variable, there is no such issue.

The regime of parameters in which our technique applies go further if the hypergraph is *linear*; namely, any two hyperedges intersect on at most one vertex. In this case our result also almost matches not only the state-of-the-art randomised algorithms [HSZ19], where they require $\Delta \leq c2^k/k^2$ for some absolute constant $c > 0$, but also the hardness result that approximate counting is intractable when $\Delta > 2.5 \cdot 2^k$ [QW22]. The improvement for linear hypergraphs is achieved by adapting a technique introduced in [FGW22].

Theorem 1.2. *For any real number $\delta > 0$, let $k \geq \frac{25(1+\delta)^2}{\delta^2}$ and $\Delta \geq 2$ be two integers such that $\Delta \leq \frac{1}{100k^3}2^{k/(1+\delta)}$. There is an FPTAS for the number of independent sets in k-uniform linear hypergraphs with maximum degree Δ.*

1.1.3. *Counting hypergraph colourings.* The other application we give is to approximately count the number of hypergraph (proper) colourings. Again let $H = (V, \mathcal{E})$ be a hypergraph, and a q-colouring $\sigma \in [q]^V$ is called *proper* if no hyperedge is monochromatic under σ. Similar to hypergraph independent sets, our work also closes the exponential gap that previously existed between the range of parameters for which efficient randomised and deterministic algorithms exist.

This problem was also studied first in [BDK08], where they show that Glauber dynamics mixes in $O(n \log n)$ time if $\Delta < q - 1$ and $k > 4$. However, to go beyond $\Delta = \Theta(q)$, one starts to encounter the so-called frozen barrier. As observed in [FM11], Glauber dynamics can no longer be connected when $\Delta > cq$ for sufficiently large c.

The key to bypassing the frozen barrier lies in a classical combinatorial result, the Lovász local lemma. Introduced by Erdős and Lovász [EL75], the local lemma was originally used to show that hypergraph colourings exist when $\Delta \leq q^{k-1}/e$. In fact, it gives more information about the distribution of uniform colourings than their mere existence [HSS11]. Especially if we assume similar but stronger conditions than the local lemma condition, the distribution enjoys many nice properties. This enables a projection approach [FGYZ21a, FHY21] where one finds a properly projected distribution to run

Glauber dynamics on. The projected distribution has better connectivity than the original state space, yet transitions between projected states can be efficiently implemented. With this approach, the state-of-the-art randomised algorithm [JPV21a] (and the related perfect sampling algorithm [HSW21]) is shown to be efficient if $\Delta \lesssim q^{k/3}$.

On the other hand, in fact, efficient deterministic algorithms were obtained even before randomised algorithms in this setting [GLLZ19] under similar local lemma type conditions, based on Moitra's linear programming based approach [Moi19]. Their algorithm was later improved by Jain, Pham, and Vuong [JPV21b], and then the aforementioned work of He, Wang, and Yin [HWY23] introduced an alternative method achieving $\Delta \lesssim q^{k/5}$, which was state-of-the-art before our work. Note, once again, the exponential in k gap between the range of parameters for efficient randomised and deterministic algorithms.

We also close this gap. Our result is summarised in Theorem 1.3. However, in this setting, there is still an exponential in k gap between the algorithmic threshold and the hardness threshold of [GGW23]. A detailed running time bound is given in Section 6. Previous works and hardness results are summarised in Table 2. Once again, for a detailed discussion and comparison with the previous best algorithm by He, Wang, and Yin [HWY23], see Section 1.2.

Theorem 1.3. *Let $k \geq 20$, $\Delta \geq 2$, and q be three integers such that $\Delta \leq \left(\frac{q}{64}\right)^{\frac{k-5}{3}}$. There is an FPTAS for the number of proper q-colourings in k-uniform hypergraphs with maximum degree Δ.*

Hypergraph colourings	Reference	Bound	Running time
Randomised counting / sampling	[BDK08]	$\Delta \leq q - 1$	$\tilde{O}(n^2) \,/\, O(n \log n)$
	[FHY21]	$\Delta \lesssim q^{k/9}$	$\tilde{O}(n^{2.0001}) \,/\, \tilde{O}(n^{1.0001})$
	[JPV21a, HSW21]	$\boldsymbol{\Delta \lesssim q^{k/3}}$	$\tilde{O}(n^{2.0001}) \,/\, \tilde{O}(n^{1.0001})$
Deterministic counting	[GLLZ19]	$\Delta \lesssim q^{k/14}$	$n^{\mathrm{poly}(k,\Delta,\log q)}$
	[JPV21b]	$\Delta \lesssim q^{k/7}$	$n^{\mathrm{poly}(k,\Delta,\log q)}$
	[HWY23]	$\Delta \lesssim q^{k/5}$	$n^{\mathrm{poly}(k,\Delta,\log q)}$
	This work	$\boldsymbol{\Delta \lesssim q^{k/3}}$	$n^{\mathrm{poly}(k,\Delta,\log q)}$
Hardness	[GGW23]	for even q, $\Delta \geq 5q^{k/2}$ assuming $\mathbf{P} \neq \mathbf{NP}$	

TABLE 2. Algorithms and hardness results for hypergraph colourings

In this setting, we apply CTTP to the projected distribution, and there are two main differences from hypergraph independent sets. First, in each step of the self-reduction, instead of a single variable, we need to evaluate a set of variables of size k. A natural attempt to address this is to run CTTP for each variable, and there is a further complication that we need to maintain consistency of the randomness used amongst different runs. Moreover, as we are sampling from the projected distribution, we still need to sample a colouring conditioned on the projected sample. This is addressed by noticing that CTTP not only returns the final state of the variables, it also gives us enough information to perform the last update for them. This information is indeed also enough to sample a colouring from the marginal distribution conditioned on the projected sample.

Similar to the case of hypergraph independent sets, we also have improved results for linear hypergraphs, almost matching the state-of-the-art randomised algorithm [FGW22].

Theorem 1.4. *For any real number $\delta > 0$, let $k \geq \frac{50(1+\delta)^2}{\delta^2}$, $\Delta \geq 2$, and q be three integers such that $\Delta \leq \left(\frac{q}{50}\right)^{\frac{k-3}{2+\delta}}$. There is an FPTAS for the number of proper q-colourings in k-uniform linear hypergraphs with maximum degree Δ.*

1.2. **Connection and comparisons with the Anand-Jerrum algorithm.** As mentioned before, the core of our derandomisation method is a logarithmic-cost marginal sampler, which may have independent interests. Our main source of inspiration, and also the first such marginal sampler, is the recent recursive algorithm by Anand and Jerrum [AJ22] for perfect sampling in infinite spin systems. Although the implementation of our marginal sampler also has a recursive structure, there are some quite noticeable distinctions. For one thing, our algorithm can be derived from Glauber dynamics, whereas the AJ algorithm relies on spatial mixing properties and does not seem to directly correspond to any Markov chain. Moreover, as the recursive call goes deeper and deeper, AJ's algorithm would pin more and more variables. In contrast, our sampler resembles Markov chains. No variable is permanently fixed, and all variables can be refreshed if we go back long enough. This last point actually provides us with a technical edge in the analysis, which we will discuss later.

The Anand-Jerrum algorithm soon found applications for almost uniform sampling general constraint satisfaction solutions in the local lemma regime [HWY22]. It was later derandomised [HWY23], which leads to the previous best deterministic approximate counting algorithms for the two applications we consider. This is a very recent and rare exception to the common paradigm that deterministic approximate counting algorithms require drastically different techniques from randomised algorithms. However, the Anand-Jerrum algorithm encounters some difficulty in matching the state-of-art bounds for randomised algorithms, particularly in the two applications we consider – hypergraph independent sets and hypergraph colourings. From a technical point of view, the difficulty lies in the lack of control for the aforementioned pinning of partial configurations in deep recursive calls of AJ's algorithm. This makes it hard to analyse the time-space structure derived from the algorithm's execution history, which is a key feature in ours and previous information percolation arguments [HSZ19, QWZ22, JPV21a, HSW21] to achieve the state-of-the-art bounds. We leave it as an interesting direction whether there is a refined analysis of the Anand-Jerrum algorithm matching other methods in these contexts.

In any case, we remark that the Anand-Jerrum algorithm does also lead to deterministic counting algorithms, especially for spin systems with strong spatial mixing on graphs with subexponential growth. The only thing missing from [AJ22] is tail bounds for their algorithm's running time. We provide such analysis and collate the implications in Appendix B.

1.3. **Derandomising random scans.** We choose to consider systematic scan to illustrate our idea of using CTTP for derandomisation, but the technique can be applied to random update Markov chains as well. In Section 7, we show how to use CTTP to derandomise random scan Glauber dynamics for Gibbs distributions.

A vanilla attempt is to do extra enumerations over possible scan sequences. However, even with a more careful argument like considering only the visited vertices, it still costs superpolynomial running time, assuming $O(\log n)$ so many times of backward deductions (which is also optimal for the truncation error we need). Instead, we take the advantage of the local structure of visiting patterns by introducing a "witness tree" structure that captures all information needed to recover the CTTP process. Enumerating all possible witness trees leads to efficient derandomisation for a random scan order.

We remark that our witness tree for Markov chains is reminiscent of the witness tree appeared in the analysis of the celebrated Moser-Tardos algorithm for algorithmic Lovász Local Lemma [MT10], which may be of independent interest.

1.4. **Other related work.** Our coupling towards the past (CTTP) marginal sampler is also inspired by the celebrated "coupling *from* the past" (CFTP) by Propp and Wilson [PW96] for perfect simulation of Markov chains. Both methods share some similarities, such as running backwards in time and having underlying grand couplings. However, they are also very different in several aspects. The main difference is that CFTP needs to sequentially simulate the evolution of the whole state, which, even with some optimisation (such as using bounding chains [Hub98]) and under favourable conditions, would still require at least linear time. This makes it unsuitable for our derandomisation needs. Our CTTP, on the other hand, only guarantees the value of a single variable to be coupled from all possible starting configurations. Even if we want to couple only a single variable in CFTP, it would be impossible

to determine what variables to simulate a priori. Our backwards deduction approach is an adaptive solution to this problem, namely, the information revealed so far determines what variables to be revealed next. This constitutes a big difference in implementing the two methods.

Our CTTP marginal sampler is also related to providing *local access to huge random objects* [BRY20]. For example, consider the uniform distribution μ of independent sets in a huge hypergraph $H = (V, \mathcal{E})$. Upon being queried at a vertex $v \in V$, our algorithm returns an approximate sample X_v from the marginal distribution μ_v using a sub-linear number of local neighbourhood probes. By using public random bits, our algorithm could guarantee that the answers for different queries are consistent with each other, which means for different vertices v, all the answers X_v come from the same X such that X is an approximate random sample of μ. An interesting open problem in this direction is to give (consistent or not) sub-linear time marginal samplers for general Gibbs distributions whenever an efficient global sampling algorithm exists.

There have been various works aiming to find deterministic approximations for Markov chains, especially for random walks on graphs [CS06, CDFS10, SYKY17, SYKY18, MRSV21, PV22]. However, these results do not seem to have meaningful consequences for MCMC, where the Markov chains are essentially high-dimensional random walks. The state space and the underlying (implicit) graph are exponentially large, making those aforementioned results difficult to apply.

Our work focuses on approximate counting via derandomising certain Markov Chain Monte Carlo samplers, which falls under the broader category of derandomising Monte Carlo methods. See [LV91, LVW93, GMR13, DS14, ST19] for several examples of the latter. Usually, in their contexts, it is sufficient to consider additive errors, and as a consequence, the randomised algorithms are relatively simple to get. The emphasis is on finding techniques to derandomise them. In contrast, our derandomisation actually comes from designing and analysing more involved randomised algorithms.

2. Preliminaries

2.1. **Markov chain and Glauber dynamics.** Let Ω be a (finite) state space. Let $(X_t)_{t=1}^{\infty}$ be a Markov chain over the space Ω with transition matrix P. We often use P to refer to the corresponding Markov chain. 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 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 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

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

where the *total variation distance* is defined by

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

In this paper, we consider two fundamental Markov chains on discrete state space. Let μ be a distribution over $[q]^V$. We assume $V = \{v_1, v_2, \ldots, v_n\}$. The *Glauber dynamics* starts from an arbitrary $X_0 \in [q]^V$ with $\mu(X_0) > 0$. For the t-th transition step, the Glauber dynamics does as follows

- pick a variable $v \in V$ uniformly at random and let $X_t(u) = X_{t-1}(u)$ for all $u \neq v$;
- sample $X_t(v)$ from the distribution $\mu_v^{X_{t-1}(V \setminus \{v\})}$.

The *systematic scan Glauber dynamics* starts from an arbitrary $X_0 \in [q]^V$ with $\mu(X_0) > 0$. For the t-th transition step, the systematic scan Glauber dynamics does as follows

- let $i(t) = (t \mod n) + 1$, pick the variable $v = v_{i(t)}$, and let $X_t(u) = X_{t-1}(u)$ for all $u \neq v$;
- sample $X_t(v)$ from the distribution $\mu_v^{X_{t-1}(V \setminus \{v\})}$.

The only difference between the above two Markov chains is the way they pick variables. The Glauber dynamics is an aperiodic and reversible Markov chain. 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.

Theorem 2.1 ([LP17]). *Let μ be a distribution with support $\Omega \subseteq [q]^V$. Let $(X_t)_{t=0}^{\infty}$ denote the Glauber dynamics or the systematic scan Glauber dynamics on μ. If $(X_t)_{t=0}^{\infty}$ is irreducible over Ω, it holds that*

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

2.2. Lovász local lemma.

We introduce the setting of the (variable framework) Lovász Local Lemma. Let $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ be a set of mutually independent random variables. We let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a set of "bad events" that only depends on $\mathcal{X}$. For each event A (not necessarily one of the bad events in $\mathcal{B}$), we let $\mathsf{vbl}(A) \subseteq \mathcal{X}$ be the set of variables in $\mathcal{X}$ that A depends on. Moreover, we let $\Gamma(A) = \{B \in \mathcal{B} \mid B \neq A \wedge \mathsf{vbl}(A) \cap \mathsf{vbl}(B) \neq \varnothing\}$. The celebrated Lovász Local Lemma states that when certain conditions are met, the probability that no bad events occur is nonzero:

Lemma 2.2 ([EL75]). *If there exists a function $x : \mathcal{B} \to [0, 1]$ such that*

$$(1) \qquad \forall B \in \mathcal{B} : \quad \mathbf{Pr}[B] \le x(B) \prod_{B' \in \Gamma(B)} (1 - x(B')),$$

then

$$\mathbf{Pr}\left[\bigwedge_{B \in \mathcal{B}} \overline{B} \right] \ge \prod_{B \in \mathcal{B}} (1 - x(B)) > 0,$$

When the condition (1) is satisfied, as observed in [HSS11], the probability that any event happens, conditioning on no bad events occurs is also bounded:

Lemma 2.3 ([HSS11]). *If (1) holds, then for any event A,*

$$\mathbf{Pr}\left[A \mid \bigwedge_{B \in \mathcal{B}} \overline{B} \right] \le \mathbf{Pr}[A] \prod_{B \in \Gamma(A)} (1 - x(B))^{-1},$$

Note that for a k-uniform hypergraph $H = (V, \mathcal{E})$ with $\Delta \ge 2$ and $q^{k-1} > \mathrm{e}\Delta$, by setting the bad events B_e being "e is not monochromatic" for each $e \in \mathcal{E}$, we then have $\mathbf{Pr}[B_e] = q^{1-k}$ for each $e \in \mathcal{E}$. Setting $x(B_e) = \frac{1}{\Delta}$ for each $e \in \mathcal{E}$ we have the condition in (1) is satisfied, and therefore Lemma 6.1 is a direct corollary from Lemma 2.3.

2.3. Counting to sampling reductions.

The classical result of Jerrum, Valiant and Vazirani [JVV86] showed that sampling and (randomised) approximate counting can be reduced to each other in polynomial time for "self-reducible" functions. We do not need that level of generality, and will describe the next reductions from counting to sampling for the following *graphic model*. Let $H = (V, \mathcal{E})$ be a hypergraph, where each vertex $v \in V$ represents a random variable that takes its value from a finite domain $[q] = \{1, 2, \ldots, q\}$ and each hyperedge $e \in \mathcal{E}$ represents a local constraint on the variable set $e \subseteq V$. For each $v \in V$, there is an "external field" function $\phi_v : [q] \to \mathbb{R}_{\ge 0}$, and for each $e \in \mathcal{E}$, there is a "constraint" function $\phi_e : [q]^{|e|} \to \mathbb{R}_{\ge 0}$. A graphical model is specified by the tuple $\mathcal{G} = (H, (\phi_v)_{v \in V}, (\phi_e)_{e \in \mathcal{E}})$, namely the hypergraph associate with the family of "external field" and "constraint" functions. For each configuration $\sigma \in [q]^V$, define its weight by

$$(2) \qquad w(\sigma) := \prod_{v \in V} \phi_v(\sigma_v) \prod_{e \in \mathcal{E}} \phi_e(\sigma_e).$$

The Gibbs distribution μ defined by the graphical model satisfies

$$(3) \qquad \forall \sigma \in [q]^V, \quad \mu(\sigma) := \frac{w(\sigma)}{Z},$$

where the partition function Z is given by

$$(4) \qquad Z := \sum_{\sigma \in [q]^V} w(\sigma).$$

In our applications, both counting problems for hypergraph independent sets and the hypergraph colourings can be expressed as graphic models, by setting the external field ϕ_v on each vertex to constant 1, and the constraint function ϕ_e on each hyperedge to the indicator function that the respective constraint is not violated.

At the core of [JVV86] is the decomposition of the partition function into products of more tractable quantities in a telescoping manner. Typically, these quantities are probabilities or expectations related to a sequence of distributions induced by the original model. In our paper, such decomposition comes in two manners: the vertex decomposition and the (hyper)edge decomposition. The vertex decomposition is simpler and more common, but it does not always work. The edge decomposition works more generally with the sacrifice of simplicity. We will apply the vertex decomposition to hypergraph independent sets and the edge decomposition to hypergraph colourings.

Vertex decomposition. We assume that $V = \{v_1, \cdots, v_n\}$. Let $\sigma \in [q]^V$ be a feasible configuration that $w(\sigma) > 0$. We call σ the *scheme* of vertex decomposition. By the chain rule of the Gibbs distribution, the following product holds

$$(5) \qquad Z = \frac{w(\sigma)}{\mu(\sigma)} = w(\sigma) \prod_{i=1}^n \frac{1}{\mu_{v_i}^{\sigma_{<i}}(\sigma_{v_i})},$$

where we use $\sigma_{<i}$ to denote the partial configuration of σ restricted on vertex set $\{v_1, v_2, \ldots, v_{i-1}\}$, and $\mu_{v_i}^{\sigma_{<i}}$ denote the marginal distribution on v_i induced from μ conditional on $\sigma_{<i}$. Therefore, to approximate the partition function Z, it suffices to approximate each marginal probability $\mu_{v_i}^{\sigma_{<i}}(\sigma_{v_i})$, since the weight function $w(\sigma)$ is easy to compute. The scheme σ is said to be *b-bounded* if

$$\forall i \in [n], \quad \mu_{v_i}^{\sigma_{<i}}(\sigma_{v_i}) \geq b.$$

Edge decomposition. Let $H = (V, \mathcal{E})$ be a hypergraph. We assume $V = \{v_1, \cdots, v_n\}$ and $\mathcal{E} = \{e_1, \cdots, e_m\}$. In the edge decomposition, the original hypergraph H is decomposed into a sequence of hypergraphs $H_0, H_1, \cdots, H_m$, given by $H_i := \{V, \{e_1, \cdots, e_i\}\}$. In other words, the H_i sequence is obtained by, starting from independent vertices, adding one edge from the original hypergraph in each step. Let w_i, Z_i and μ_i be the weight function, the partition function and the Gibbs distribution induced by the hypergraph H_i respectively. Then

$$
\begin{aligned}
Z = Z_m = Z_0 \prod_{i=1}^m \frac{Z_i}{Z_{i-1}} &= Z_0 \prod_{i=1}^m \sum_{\tau \in [q]^V} \frac{\mu_{i-1}(\tau) w_i(\tau)}{w_{i-1}(\tau)} \\
&\overset{(\star)}{=} Z_0 \prod_{i=1}^m \sum_{\tau_{e_i} \in [q]^{e_i}} \mu_{i-1, e_i}(\tau_{e_i}) \phi_{e_i}(\tau_{e_i}) = Z_0 \prod_{i=1}^m \mathbf{E}_{z \sim \mu_{i-1, e_i}}[\phi_{e_i}(z)],
\end{aligned}
$$

(6)

where $\mu_{i-1, e_i}(\cdot)$ denotes the marginal distribution on e_i projected from μ_{i-1}, and thus $(\star)$ holds because $\frac{w_i(\tau)}{w_{i-1}(\tau)} = \frac{w_i(\tau_{e_i})}{w_{i-1}(\tau_{e_{i-1}})}$. Therefore, it suffices to approximate the expectation $\mathbf{E}_{z \sim \mu_{i-1, e_i}}[\phi_{e_i}(z)]$ in order to approximate the partition function. Again, the edge decomposition is called *b-bounded* if

$$\forall 1 \leq i \leq m, \quad \frac{\mathbf{E}_{z \sim \mu_{i-1, e_i}}[\phi_{e_i}(z)]}{\max_{y \in [q]^{e_i}} \phi_{e_i}(y)} \geq b.$$

3. Derandomisation for deterministic counting

Our idea for deterministic counting is very simple — we just enumerate all possible random choices. In this section, we give a quick formalisation of this idea, and in subsequent sections, we tackle the main challenge of finding algorithms with logarithmic random choices. We consider randomised algorithms whose whole randomness comes from drawing random variables from discrete distributions, such as lines of the form: "draw $r \sim D$", where D is a probability distribution over a finite sample space Ω of constant size. The specification of D is also computed by the algorithm if necessary. Aside from these

samples, there is no randomness involved in the algorithm. This motivates us to consider the following random oracle model.

Random oracle model. We consider randomised algorithms that are deterministic algorithms with access to a random oracle $\mathrm{Draw}(\cdot)$, which, given as input the description of a distribution D, returns an independent random value $r \in \Omega$ distributed according to D.

This model allows us to quantify the number of random choices made in algorithms, as given in the next definition.

Definition 3.1. Let $t, r : \mathbb{N} \to \mathbb{N}$ be two nondecreasing functions and let $c \geq 2$ be a constant. We say that a randomised algorithm $\mathcal{A}$ has *time cost* $t(n)$ and *draws at most* $r(n)$ *random variables over domains of sizes at most* c, if for any $n \in \mathbb{N}$, in the worst case of the inputs of size n and all possible random choices, the algorithm $\mathcal{A}$ terminates within $t(n)$ steps of computation, and accesses the random oracle $\mathrm{Draw}(\cdot)$ for at most $r(n)$ times such that each time it draws from a sample space of size at most c.

We are interested in those randomised algorithms that have $\mathrm{poly}(n)$ time cost and draws at most $O(\log n)$ random variables over constant-sized domains, because such randomised algorithms can be transformed into polynomial-time deterministic algorithms for computing the output distributions, due to a standard routine for derandomisation by enumerating all random choices.

Proposition 3.2. *Let $\mathcal{A}$ be a randomised algorithm with time cost $t(n)$ and draws at most $r(n)$ random variables over domains of sizes at most c. There is a deterministic algorithm $\mathcal{B}$ that, on any input Π of size n, outputs the distribution of $\mathcal{A}(\Pi)$ in time $O(t(n)c^{r(n)})$.*

Proof. Consider the decision tree $\mathcal{T} = \mathcal{T}(\Pi)$ for adaptively querying the random oracle $\mathrm{Draw}(\cdot)$ by the algorithm $\mathcal{A}$ on an input Π of size n. Since the algorithm $\mathcal{A}$ draws at most $r(n)$ random variables over domains of sizes at most c in the worst case of inputs and random choices, the decision tree has a branching number at most c and depth at most $r(n)$. Therefore, there are at most $c^{r(n)}$ leaves in $\mathcal{T}$. Since $\mathcal{A}$ has a time cost $t(n)$ in the worst case of inputs and random choices, the computation cost for the path from the root to each leaf in $\mathcal{T}$ is bounded by $t(n)$. Hence, the entire tree $\mathcal{T}$ can be computed in $O(t(n)c^{r(n)})$ time. Note that each leaf in $\mathcal{T}$ corresponds to a possible output value for $\mathcal{A}(\Pi)$, whose probability is given by that of the random choices along the path. Therefore, the distribution of the output $\mathcal{A}(\Pi)$ can be computed within $O(t(n)c^{r(n)})$ time by aggregating over all leaves in $\mathcal{T}$. $\qquad\square$

Implications to approximate counting. Recall the self-reductions using vertex/edge decompositions defined in Section 2.3 and the corresponding marginal distributions $\mu_{v_i}^{\sigma_{<i}}$ in (5) and μ_{i-1,e_i} in (6). Then a straightforward consequence to Proposition 3.2 is that the partition functions can be approximated deterministically in polynomial-time as long as one can sample approximately from the marginal distribution in polynomial-time drawing $O(\log n)$ random variables whose domain sizes are upper bounded by a constant. This is formally stated below.

Corollary 3.3. *Let $\mathcal{G}$ be a class of graphical models, where each instance $\mathcal{I} \in \mathcal{G}$ is provided with a b-bounded vertex decomposition (or a b-bounded edge decomposition). If for every $\varepsilon \in (0, 1)$ there exists a randomised algorithm $\mathcal{A}$ such that for every instance $\mathcal{I} \in \mathcal{G}$ of n vertices and m edges, and every possible marginal distribution $\mu_{v_i}^{\sigma_{<i}}$ used in the vertex decomposition (or μ_{i-1,e_i} in the edge decomposition), the algorithm $\mathcal{A}$ returns a Y_i within time $t(\varepsilon, n)$, by drawing at most $r(\varepsilon, n)$ random variables of domain sizes at most c, such that*

$$d_{\mathrm{TV}}\left(Y_i, \mu_{v_i}^{\sigma_{<i}}\right) \leq \frac{b\varepsilon}{10n}, \qquad \left(\text{or } d_{\mathrm{TV}}\left(Y_i, \mu_{i-1,e_i}\right) \leq \frac{b\varepsilon}{10m} \text{ for the edge decomposition,}\right)$$

then there exists a deterministic algorithm $\mathcal{B}$ that for every $\varepsilon \in (0, 1)$ and every instance $\mathcal{I} \in \mathcal{G}$, returns an ε-approximation of the partition function of $\mathcal{I}$ within time $O((m + n)t(\varepsilon, n)c^{r(\varepsilon, n)})$.

The proof of Corollary 3.3 is rather straightforward. The only slight complication is to convert additive errors into relative errors, which is made possible by our b bounded assumptions.

4. Coupling towards the past

In this section, we introduce the coupling towards the past idea and present a marginal sampler by evaluating the state of a single variable in stationary Markov chains. Suppose the chain has evolved sufficiently long. Our goal is to evaluate the current state by revealing as little randomness as possible. It suffices to perform the last update, and the update function is determined by both a fresh random variable and the states of other variables. The fresh random variable may allow us to determine the state of the target variable directly without knowing any state of the rest. This is possible when marginal probability lower bounds are available. If we cannot make such a quick decision, we recursively reveal the necessary information on other variables required for this update. In the case of Gibbs distributions, this step amounts to finding out the states of the neighbours of the target variable. However, in the application of hypergraph colourings, to overcome the irreducibility barrier, we need to sample from a "projected distribution" instead of the uniform distribution over all proper colourings. This "projected distribution" we sample from is no longer a Gibbs distribution, and the revealing step becomes more complicated. Finally, we will truncate once we have revealed too much information to ensure that this process is efficient on randomness.

There is an implicit grand coupling (i.e. a coupling for chains starting from all possible initial configurations) underlying the construction above. We may consider all random variables drawn beforehand, and then all chains use the same values. The approach above is just delayed revelation, or alternatively constructing the grand coupling recursively towards the past. If the CTTP process terminates at time $-t$, then it means that under these random choices, all chains starting from a time $T < -t$, no matter what the initial configurations are, lead to the same value at time 0 for the target variable.

To carry out the plan, we first introduce an implementation of the standard systematic scan Glauber dynamics which utilises the lower bound information as much as possible. This is in Section 4.1. Then in Section 4.2, we flip the order of evaluation and deduce the state of a single variable backwards. The algorithm is given in Algorithm 1 and its correctness is shown in Theorem 4.6. Finally, in Section 4.3, we give the truncated algorithm, Algorithm 3, and bound its error in Theorem 4.9.

4.1. Simulating Glauber dynamics assuming marginal lower bound.
Let (X_t) be a convergent Markov chain on the state space $[q]^V$ with its stationary distribution μ. In particular, consider the chain $(X_t)_{-\infty < t \leq 0}$ running from time $-\infty$ to time 0. Drawing a sample from the marginal distribution μ_v for an arbitrary $v \in V$ can be realised by evaluating $X_0(v)$.

For technical reasons, we consider a class of chains known as *systematic scan Glauber dynamics*. Enumerate the variables as $V = \{v_1, v_2, \ldots, v_n\}$, and for any $t \in \mathbb{Z}$, define:

$$(7) \qquad i(t) := (t \bmod n) + 1.$$

The rule for the t-th transition $(X_{t-1} \rightarrow X_t)$ is:

- pick the variable $v = v_{i(t)}$ where $i(t)$ is defined in (7);
- let $X_t \in [q]^V$ be constructed as that $X_t(u) = X_{t-1}(u)$ for all $u \neq v$, and $X_t(v)$ is drawn independently according to the marginal distribution $\mu_v^{X_{t-1}(V \setminus \{v\})}$.

This chain converges to the (unique) stationary distribution μ when it is irreducible.

Remark 4.1 (choice of systematic scan Glauber dynamics). The CTTP is not restricted to systematic scan Glauber dynamics. Assuming a fixed scan order as in the systematic scan simplifies our expositions. But such simplification does not change the nature of the derandomisation problem. Later in Section 7, we show that with some extra efforts, the CTTP can be applied to derandomise the standard Glauber dynamics with random scan order.

To perform the update, it is usually not necessary to know all of $X_{t-1}(V \setminus \{v\})$. Typically there is a subset $\Lambda \subseteq V \setminus \{v\}$ such that $\mu_v^{X_{t-1}(\Lambda)} = \mu_v^{X_{t-1}(V \setminus \{v\})}$. Let $\sigma_\Lambda = X_{t-1}(\Lambda)$. The marginal distribution $\mu_v^{\sigma_\Lambda}$ can be decomposed as follows if it is suitably lower bounded.

Definition 4.2. Let μ be a distribution over $[q]^V$. Let $\boldsymbol{b} = (b_1, b_2, \ldots, b_q) \in [0, 1]^q$.

- **Marginal lower bound**: μ is said to be $\boldsymbol{b}$-*marginally lower bounded* if for any $v \in V$, $\Lambda \subseteq V \setminus \{v\}$ and any feasible $\sigma_\Lambda \in [q]^\Lambda$, it holds that $\mu_v^{\sigma_\Lambda}(j) \geq b_j$ for all $j \in [q]$.

For a b-marginally lower bounded distribution μ over $[q]^V$, for each $v \in V$, we define the following distributions (we follow the convention $0/0 = 0$):

- **Lower bound distribution** μ_v^{LB} **over** $\{\perp\} \cup [q]$: $\mu_v^{\mathrm{LB}} = \mu^{\mathrm{LB}}$ for all $v \in V$ such that

$$\mu^{\mathrm{LB}}(\perp) := 1 - \sum_{i=1}^q b_i \quad \text{and} \quad \forall j \in [q], \quad \mu^{\mathrm{LB}}(j) := \frac{b_j}{\sum_{i=1}^q b_i}.$$

- **Padding distribution** $\mu_v^{\mathrm{pad}, \sigma_\Lambda}$ **over** $[q]$: for $\Lambda \subseteq V \setminus \{v\}$ and feasible $\sigma_\Lambda \in [q]^\Lambda$,

$$\forall j \in [q], \quad \mu_v^{\mathrm{pad}, \sigma_\Lambda}(j) := \frac{\mu_v^{\sigma_\Lambda}(j) - b_j}{1 - \sum_{i=1}^q b_i}.$$

With above definitions, drawing a sample $c \sim \mu_v^{\sigma_\Lambda}$ according to the marginal distribution $\mu_v^{\sigma_\Lambda}$ can be simulated by the following two steps:

(1) draw $c \sim \mu_v^{\mathrm{LB}}$;
(2) if $c = \perp$, override c by drawing $c \sim \mu_v^{\mathrm{pad}, \sigma_\Lambda}$.

Assume that μ has suitable marginal lower bounds. Fix an integer $T \geq 0$. The systematic scan Glauber dynamics $(X_t)_{-T \leq t \leq 0}$ from time $-T$ to 0 can be generated by the following process $\mathcal{P}(T)$. We use the convention that for any $v \in V$ and $t < -T$, $X_t(v) = X_{-T}(v)$.

The systematic scan Glauber dynamics $\mathcal{P}(T)$

- Initialize $X_{-T} \in [q]^V$ as an arbitrary feasible configuration.
- For $t = -T + 1, -T + 2, \ldots, 0$, the configuration X_t is constructed as follows:
  (a) pick $v = v_{i(t)}$, where $i(t)$ is defined in (7), and let $X_t(u) \leftarrow X_{t-1}(u)$ for all $u \neq v$;
  (b) draw $r_t \sim \mu^{\mathrm{LB}}$ independently, and let $X_t(v) \leftarrow r_t$ if $r_t \neq \perp$; otherwise,

  (8) $\qquad \sigma_\Lambda \leftarrow \mathrm{Boundary}(t), \quad$ (by accessing X_{t-1} and $\mathcal{R}_{t-1} := (r_s)_{-T < s < t}$)

  and draw $X_t(v) \sim \mu_v^{\mathrm{pad}, \sigma_\Lambda}$ independently.

In (8), we use a subroutine $\mathrm{Boundary}(t)$ satisfying the following condition, which allows us to perform the update without revealing the whole X_{t-1}.

Condition 4.3. *The procedure* $\mathrm{Boundary}(t)$ *always terminates and returns* $\sigma_\Lambda \in [q]^\Lambda$ *satisfying that* $\mu_v^{\sigma_\Lambda} = \mu_v^{X_{t-1}(V \setminus \{v\})}$ *for* $v = v_{i(t)}$.

The implementation of $\mathrm{Boundary}(t)$ will be application specific. It is a deterministic procedure with oracle access to previous random variables $\mathcal{R}_{t-1} = (r_s)_{-T < s < t}$ and to the current configuration X_{t-1} generated in the process $\mathcal{P}(T)$. These accesses are provided by the following two oracles:

- **lower bound oracle** $\mathcal{B}(s)$: given any $s < t$, returns r_s if $s > -T$ and $\perp$ otherwise;
- **configuration oracle** $C(u)$: given $u \in V$, returns $X_{t-1}(u)$.

Typically, we will query $\mathcal{B}(s)$ as much as possible, and default to $C(u)$ only if $\mathcal{B}(s)$ returns $\perp$. It is even possible, in some situations, not needing to query $C(u)$ after $\mathcal{B}(u)$ returns $\perp$. This is because not all variables queried in $\mathrm{Boundary}(t)$ are necessary to determine Λ and σ_Λ, and this phenomenon will become self-evident in the applications later. Although using only $C(u)$ is sufficient to achieve Condition 4.3, the use of $\mathcal{B}(s)$ is crucial and allows us to reduce the number of random variables used for the backward deduction in the next subsection.

To better understand $\mathrm{Boundary}(t)$, take a Gibbs distribution μ as an example. In Gibbs distributions, each variable $v \in V$ has a neighbourhood $N(v) \subseteq V \setminus \{v\}$ conditioned on which v is independent from the rest of the variables, namely its non-neighbours. Consequently, for any feasible $X_{t-1} \in [q]^V$,

$$\mu_v^{X_{t-1}(V \setminus \{v\})} = \mu_v^{X_{t-1}(N(v))}.$$

Then a straightforward implementation of $\mathsf{Boundary}(t)$ is to return the configuration $\sigma_\Lambda \leftarrow X_{t-1}(N(v))$ by retrieving it from X_{t-1} via the oracle $C(\cdot)$. However, as we want to reduce the number of accesses to $C(\cdot)$, we instead try to infer $X_{t-1}(N(v))$ from already drawn samples $\mathcal{R}_{t-1} = (r_s)_{-T < s < t}$.

The following gives an alternative implementation of $\mathsf{Boundary}(t)$ for the Gibbs distribution μ. For any $u \in V$ and integer t, denote by $\mathrm{pred}_u(t)$ the last time before t at which u is updated, i.e.

(9) $$\mathrm{pred}_u(t) := \max\{s \le t \mid v_{i(s)} = u\},$$

where $i(s)$ is specified by the scan order defined in (7).

An implementation of $\mathsf{Boundary}(t)$ for Gibbs distribution μ

- Let $\Lambda \leftarrow N(v)$, where $v = v_{i(t)}$ and $i(t)$ is defined as (7).
- For each $u \in \Lambda$:
  - (1) let $s = \mathrm{pred}_u(t)$, if $s > -T$ and $r_s \neq \perp$, then $\sigma(u) \leftarrow r_s$;   (*oracle query* $\mathcal{B}(s)$)
  - (2) otherwise, $\sigma(u) \leftarrow X_{t-1}(u)$;   (*oracle query* $C(u)$)
- return σ_Λ.

This implementation satisfies Condition 4.3 due to the conditional independence property mentioned earlier for Gibbs distributions.

In general, μ can be an arbitrary distribution over $[q]^V$ and the subroutine $\mathsf{Boundary}(t)$ will be implemented specifically depending on μ. Nevertheless, the following proposition is easy to verify.

Proposition 4.4. *As long as the subroutine* $\mathsf{Boundary}(t)$ *satisfies Condition 4.3, the process* $\mathcal{P}(T)$ *generates a faithful copy of the systematic scan Glauber dynamics* $(X_t)_{-T \le t \le 0}$ *for* μ.

4.2. Evaluating stationary states via backward deductions.
Fix an arbitrary integer $T \ge 0$. Consider the Markov chain $(X_t)_{-T \le t \le 0}$ generated by the process $\mathcal{P}(T)$. We present an algorithm that outputs the random variable $X_0(v)$ for $v \in V$. Instead of simulating the process $\mathcal{P}(T)$ chronologically from time $-T$ to 0, our algorithm uses a backward deduction which tries to infer the correct value of $X_0(v)$ by accessing as few random variables as possible for resolving $X_0(v)$.

The algorithm is described in Algorithm 1. It is a recursive algorithm that has an input argument $t \le 0$ and maintains two global data structures M and R, initialized respectively as $M_0 = \perp^{\mathbb{Z}}$ and $R_0 = \varnothing$. Since all recursive calls access and update the same M and R, we sometimes write $\mathsf{Resolve}_T(t) = \mathsf{Resolve}_T(t; M, R)$ for short.

For $-T < t \le 0$, $\mathsf{Resolve}_T(t)$ tries to calculate the result of the update at time t, which is $X_t(v_{i(t)})$. In particular, for $-n < t < 0$, we have $X_t(v_{i(t)}) = X_0(v_{(t \bmod n)+1}) = X_0(v_{t+n+1})$, and $X_0(v_{i(0)}) = X_0(v_1)$.

Algorithm 1: $\mathsf{Resolve}_T(t; M, R)$

Input: an integer $t \le 0$;
Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\perp\}$ and a set R;
Output: a value $X_t(v_{i(t)}) \in [q]$;

1   **if** $t \le -T$ **then return** $X_{-T}(v_{i(t)})$;
2   **if** $M(t) \neq \perp$ **then return** $M(t)$;
3   $M(t) \leftarrow \mathsf{LB\text{-}Sample}(t; R)$ **and if** $M(t) \neq \perp$ **then return** $M(t)$;
4   $\sigma_\Lambda \leftarrow \mathsf{Boundary}(t)$, with the oracle queries being replaced by

- **upon** querying $\mathcal{B}(s)$: **if** $s > -T$, **then return** $\mathsf{LB\text{-}Sample}(s; R)$; **else return** $\perp$;
- **upon** querying $C(u)$: **return** $\mathsf{Resolve}_T(\mathrm{pred}_u(t); M, R)$, where $\mathrm{pred}_u(t)$ is defined in (9);

5   $M(t) \leftarrow$ a random value drawn independently according to $\mu_v^{\mathrm{pad}, \sigma_\Lambda}$;
6   **return** $M(t)$;

The algorithm recursively deduces the outcome of the update at time $t \le 0$. The data structure $M(t)$ stores the resolved outcomes of the updates at time t, and the set R stores the generated samples from the lower bound distribution μ^{LB}. Each value in M will only be updated at most once, and the set R

will never remove elements. Moreover, to implement the subroutine Boundary(t), each query to the lower bound oracle $\mathcal{B}(s)$ is replaced by accessing the sample r_s from the lower bound distribution μ^{LB}, realized by Algorithm 2; each query to the configuration oracle $C(u)$ is replaced by a recursive call to Resolve$_T(\mathrm{pred}_u(t); M, R)$ for evaluating $X_{t-1}(u)$. Here, the *principle of deferred decision* is applied, so that the decision of the random choices of the $(r_s)_{-T < s < t}$ in $\mathcal{P}(T)$ is deferred to the moments when they are accessed in the backward deduction.

Algorithm 2: LB-Sample($t; R$)

    Input: an integer $t \le 0$;
    Global variables: a set R of pairs $(s, r_s) \in \mathbb{Z} \times ([q] \cup \{\bot\})$;
    Output: a random value in $[q] \cup \{\bot\}$ distributed as μ^{LB}.

1 **if** $(t, r) \in R$ **then return** r;
2 **else**
3   draw $r_t \sim \mu^{\mathrm{LB}}$;
4      $R \leftarrow R \cup \{(t, r_t)\}$;
5   **return** r_t;

When T is set to ∞, the program Resolve$_\infty(t)$ is still well-defined, as the only difference is that $t \le -T$ never triggers. Indeed Resolve$_\infty(t)$ is what we call the coupling towards the past process. It tries to calculate the result of the update at time $t \le 0$ in a chain running from the infinite past to time 0. By Theorem 2.1, if the systematic scan Glauber dynamics $\mathcal{P}(T)$ is irreducible, then the distribution of X_0 converges to μ as $T \to \infty$, regardless of the initial state. We give a sufficient condition for both the convergence of the forward process $\mathcal{P}(T)$ and the termination of the backward program Resolve$_\infty(t)$.

Condition 4.5. *The lower bound distribution μ^{LB} in Definition 4.2 satisfies that $\mu^{\mathrm{LB}}(\bot) < 1$.*

The next theorem states that for any finite $T \ge 0$, Algorithm 1 always correctly evaluates the chain; and by setting $T = \infty$, it returns a sample from the marginal distribution μ_v with probability 1 if Condition 4.5 holds.

Theorem 4.6. *Let μ be a distribution over $[q]^V$, $T \ge 0$ be an integer, and $(X_t)_{-T \le t \le 0}$ be generated by the process $\mathcal{P}(T)$ whose Boundary(t) subroutine satisfies Condition 4.3. For any $-T \le t \le 0$, the followings hold:*

- *Resolve$_T(t)$ terminates in finite steps and returns a sample identically distributed as $X_t(v_{i(t)})$;*
- *if further Condition 4.5 holds, then Resolve$_\infty(t)$ terminates with probability 1, and returns a sample distributed as μ_v where $v = v_{i(t)}$.*

Proof. Fix a finite $T \ge 0$. It is straightforward to see that Resolve$_T(t)$ terminates in finite steps since the time t decreases in the recursive calls to Resolve$_T$ and the procedure terminates once $t \le -T$.

We then show that the output of Resolve$_T(t)$ is identically distributed as $X_t(v_{i(t)})$ by a coupling between the process $\mathcal{P}(T)$ and Resolve$_T(t)$ for $-T \le t \le 0$. Consider the following implementations of $\mathcal{P}(T)$ and Resolve$_T(t)$. For each $-T < \ell \le 0$, let $U_\ell \in [0, 1)$ be a real number sampled uniformly and independently at random. Whenever the process $\mathcal{P}(T)$ tries to draw an $X_\ell(v_{i(\ell)})$ according to the padding distribution $\mu^{\mathrm{pad}, \sigma_\Lambda}_{v_{i(\ell)}}$, we simulate this by assigning $X_\ell(v_{i(\ell)})$ the value $c \in [q]$ satisfying

$$(10) \qquad U_\ell \in \left[\sum_{j=1}^{c-1} \mu^{\mathrm{pad}, \sigma_\Lambda}_{v_{i(\ell)}}(j), \ \sum_{j=1}^{c} \mu^{\mathrm{pad}, \sigma_\Lambda}_{v_{i(\ell)}}(j) \right).$$

Similarly, in the procedure Resolve$_T(t)$, when a recursion Resolve$_T(\ell; M, R)$ tries to draw an $M(\ell)$ according to $\mu^{\mathrm{pad}, \sigma_\Lambda}_{v_{i(\ell)}}$ in Line 5 of Algorithm 1, we simulate this by assigning $M(\ell)$ the same $c \in [q]$ satisfying (10). It is easy to see that such implementations faithfully simulate the original processes $\mathcal{P}(T)$ and Resolve$_T(t)$ respectively.

Besides the random choices for sampling from the padding distributions, which are provided by the sequence $(U_\ell)_{-T < \ell \le 0}$, the only remaining random choices in the two processes Resolve$_T(t)$ and $\mathcal{P}(T)$ are the outcomes $(r_\ell)_{-T < \ell \le 0}$ for sampling from the lower bound distribution μ^{LB}.

14

We define the following coupling between $\text{Resolve}_T(t)$ and $\mathcal{P}(T)$:

- the two processes use the same random choices for $(r_\ell)_{-T < \ell \le 0}$ and $(U_\ell)_{-T < \ell \le 0}$.

Fix any evaluation of the random choices $\mathbf{r} = (r_\ell)_{-T < \ell \le 0}$ and $\mathbf{U} = (U_\ell)_{-T < \ell \le 0}$. Both $\text{Resolve}_T(t)$ and the $(X_t)_{-T < t \le 0}$ generated according to $\mathcal{P}(T)$ are fully deterministic given $(\mathbf{r}, \mathbf{U})$. Furthermore, for any mapping $M : \mathbb{Z} \to [q] \cup \{\perp\}$ and any set R of $(s, r_s') \in \mathbb{Z} \times ([q] \times \{\perp\})$ pairs, we say that (M, R) are consistent with $(\mathbf{r}, \mathbf{U})$, if the followings hold

- $\forall -T \le \ell \le 0 : \quad M(\ell) \ne \perp \implies M(\ell) = X_\ell(v_{i(\ell)})$;
- $\forall -T < \ell \le 0 : \quad (\ell, r_\ell') \in R \implies r_\ell' = r_\ell$.

Under the coupling above, by an induction on t from $-T$ to 0, one can routinely verify that for any (M, R) consistent with $(\mathbf{r}, \mathbf{U})$, the output of $\text{Resolve}_T(t; M, R)$ is precisely $X_t(v_{i(t)})$ generated according to $\mathcal{P}(T)$ using the same random choices $(\mathbf{r}, \mathbf{U})$, and the states of (M, R) during the execution of $\text{Resolve}_T(t; M, R)$ remain consistent with $(\mathbf{r}, \mathbf{U})$. This shows that $\text{Resolve}_T(t)$ is identically distributed as $X_t(v_{i(t)})$ for any finite $T \ge 0$ and any $-T \le t \le 0$. In other words,

$$(11) \qquad d_{\text{TV}}(X_t(v), \text{Resolve}_T(t)) = 0.$$

Next, we deal with the infinite case. We claim that Condition 4.5 implies both the irreducibility of $\mathcal{P}(T)$ and the termination of $\text{Resolve}_\infty(t)$ with probability 1.

For the irreducibility of $\mathcal{P}(T)$, by Condition 4.5, there must exist $c_0 \in [q]$ such that $\mu^{\text{LB}}(c_0) > 0$. Let $\sigma \in [q]^V$ be the constant configuration such that $\sigma(v) = c_0$ for all $v \in V$. Then by the definition of μ^{LB} it follows that σ is feasible and also can be reached via transitions of $\mathcal{P}(T)$ from all feasible configurations. It is also straightforward to verify that for any two feasible configurations $\tau, \tau' \in [q]^V$ and any $-T < t \le 0$, if $P_t(\tau, \tau') > 0$ then $P_t(\tau, \tau') > 0$, where P_t denotes the one-step transition matrix of $\mathcal{P}(T)$ at time t. Therefore any feasible configuration is also reachable from σ. This shows the irreducibility of $\mathcal{P}(T)$.

Then we show the termination of $\text{Resolve}_\infty(t_0)$ for any $t_0 \le 0$. For each $t \le t_0$, define the event:

$$\mathcal{B}_t : r_s \ne \perp \text{ for all } s \in [t - n + 1, t].$$

We claim that if $\mathcal{B}_t$ happens for some $t \le t_0$, then no recursive calls of $\text{Resolve}_\infty(s; M, R)$ would be incurred for any $s \le t - n$. Assume for the sake of contradiction that there exists a maximum $s^* \le t - n$ such that $\text{Resolve}_\infty(s^*; M, R)$ is called. As $s^* \le t - n < t_0$, $\text{Resolve}_\infty(s^*; M, R)$ must be recursively called directly within another instance of $\text{Resolve}_\infty(s'; M, R)$ such that $s^* < s'$. Note that by Line 4 of Algorithm 1 and (9) we also have $s^* > s' + n$. We then have two cases:

(1) $s' \le t - n$, this contradicts the maximality assumption for s^*.
(2) Otherwise $s' > t - n$. By $s^* \le t - n$ and $s^* > s' + n$ we have $s' \in [t - n + 1, t]$. Also by the assumption that $\mathcal{B}_t$ happens, we have $r_{s'} \ne \perp$, therefore $\text{Resolve}_\infty(s'; M, R)$ would have terminated at Line 3 of Algorithm 1 without incurring any recursive call. This also leads to a contradiction and thus proves the claim.

Note that by Condition 4.5, for any $t \le t_0$, we have $\mathbf{Pr}[\mathcal{B}_t] \ge p := (1 - \mu^{\text{LB}}(\perp))^n > 0$. For any $L > 0$, let $\mathcal{E}_L$ be the event that there is a recursive call to $\text{Resolve}_\infty(t^*; M, R)$ where $t^* \le t_0 - Ln$. By the claim above,

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

where the equality is due to independence of $(r_t)_{t \le t_0}$. Thus, with probability 1 there is only a finite number of recursive calls, namely $\text{Resolve}_\infty(t_0)$ terminates with probability 1.

For any $t \le 0$, since $\text{Resolve}_\infty(t)$ terminates with probability 1, its output distribution is well defined. For any $\varepsilon > 0$, consider a sufficiently large L such that $(1 - p)^L \le \varepsilon$. For any $T \ge Ln - t$, we couple $\text{Resolve}_\infty(t)$ with $\text{Resolve}_T(t)$ using the same random variables drawn. As the coupling fails only if $\mathcal{E}_L$ happens, by the coupling lemma,

$$d_{\text{TV}}(\text{Resolve}_T(t), \text{Resolve}_\infty(t)) \le \mathbf{Pr}[\mathcal{E}_L] \le \varepsilon.$$

This implies

$$\lim_{T \to \infty} d_{\mathrm{TV}} \left(\mathrm{Resolve}_T(t), \mathrm{Resolve}_\infty(t) \right) = 0. \tag{12}$$

The irreducibility of $\mathcal{P}(T)$ and Theorem 2.1 implies that

$$\lim_{T \to \infty} d_{\mathrm{TV}} \left(\mu_v, X_{T,t}(v) \right) = 0, \tag{13}$$

where $v = v_{i(t)}$ and $X_{T,t}$ is the state of $\mathcal{P}(T)$ at time $t \geq T$. Moreover, for any $T > 0$,

$$d_{\mathrm{TV}} \left(\mu_v, \mathrm{Resolve}_\infty(t) \right) \leq d_{\mathrm{TV}} \left(\mu_v, \mathrm{Resolve}_T(t) \right) + d_{\mathrm{TV}} \left(\mathrm{Resolve}_T(t), \mathrm{Resolve}_\infty(t) \right). \tag{14}$$

Combining the above, we have

$$d_{\mathrm{TV}} \left(\mu_v, \mathrm{Resolve}_\infty(t) \right)$$

(by (14)) $\quad \leq \quad \limsup\limits_{T \to \infty} d_{\mathrm{TV}} \left(\mu_v, \mathrm{Resolve}_T(t) \right) + \limsup\limits_{T \to \infty} d_{\mathrm{TV}} \left(\mathrm{Resolve}_T(t), \mathrm{Resolve}_\infty(t) \right)$

(by (12)) $\quad = \quad \limsup\limits_{T \to \infty} d_{\mathrm{TV}} \left(\mu_v, \mathrm{Resolve}_T(t) \right)$

(by (13) and (11)) $\quad \leq \quad \limsup\limits_{T \to \infty} d_{\mathrm{TV}} \left(\mu_v, X_{T,t}(v) \right) + \limsup\limits_{T \to \infty} d_{\mathrm{TV}} \left(X_{T,t}(v), \mathrm{Resolve}_T(t) \right) = 0,$

and the theorem follows. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Note that in the proof above, L has to be at least $\Omega(p^{-1})$ to guarantee a constant probability upper bound for the event $\mathcal{E}_L$. Since p is an exponentially small quantity, this argument usually does not yield a useful mixing time bound. This is mainly a proof for convergence and correctness.

Remark 4.7. In fact, if $\mathrm{Resolve}_\infty(t)$ terminates with probability 1 for any $-n < t \leq 0$, then the Glauber dynamics has to be irreducible. This is because, fix an arbitrary set of random choices, and run $\mathrm{Resolve}_\infty(t)$ for all $-n < t \leq 0$. Because of the termination assumption, there is a finite $T < 0$ beyond which all of them terminates, and we obtain a configuration σ such that $\sigma(v_i) = \mathrm{Resolve}_\infty(\mathrm{pred}_i(0))$. It is then straightforward to see that if we run $P(T)$ under the same random choices, no matter what the initial configuration is, the configuration at time 0 is exactly σ. This implies that all feasible configurations can reach σ and vice verse, and thus the Glauber dynamics is irreducible.

On the other hand, $\mathrm{Resolve}_\infty(t)$ terminating with probability 1 for any $-n < t \leq 0$ is not that far away from Condition 4.5. Consider a distribution μ. It is possible for $\mathrm{Resolve}_\infty(t)$ to terminate with probability 1 if there is some v such that $\mu_v^{\mathrm{LB}}(\bot) = 1$. However, as soon as there are two variables u and v such that $\mu_v^{\mathrm{LB}}(\bot) = \mu_u^{\mathrm{LB}}(\bot) = 1$, and one cannot deduce the value of u (or v) without knowing the value of v (or u), $\mathrm{Resolve}_\infty$ will not terminate when resolving either u or v.

4.3. Truncated simulation using bounded randomness.
Theorem 4.6 implies that under Condition 4.5, the coupling towards the past process, $\mathrm{Resolve}_\infty(t)$, is a perfect sampler for the marginal distribution of μ_v. In our typical applications, the expected running time of this algorithm is often a constant. However, since our end goal is derandomisation, we need an algorithm that draws no more than logarithmic random variables in the worst case. This leads to the truncated version in Algorithm 3. Similar to Algorithm 1, we may sometimes drop M and R from the input as all recursive calls access the same data structures.

Algorithm 3: $\mathrm{ApproxResolve}(t, K; M, R)$

   Input: integers $t \leq 0$ and $K \geq 0$
   Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\bot\}$ and a set R;
   Output: a random value in $[q] \cup \{\bot\}$;

1   initialize $M \leftarrow \bot^{\mathbb{Z}}$ and $R \leftarrow \varnothing$;

2   **try :**

3 | **return** $\mathrm{Resolve}_\infty(t; M, R)$;

4   **catch** $|R| \geq K$ **:**

5 | **return** $\bot$;

The algorithm $\mathsf{ApproxResolve}(t, K)$ simulates $\mathsf{Resolve}_\infty(t)$ using a set R of size bounded by K. Recall that the set R is used to store the generated samples $(r_s)_{s \leq t}$ from the lower bound distribution μ_v^{LB}. Moreover, since a sample from the padding distribution $\mu_v^{\mathrm{pad}, \cdot}$ is drawn only after a corresponding sample from the lower bound distribution has been drawn, we have the following observation.

Observation 4.8. *In Algorithm 3, at most K samples are drawn from the lower bound distribution μ_v^{LB} and at most K samples are drawn from the padding distributions $\mu_v^{\mathrm{pad}, \cdot}$.*

Denote by $\mathcal{E}_{\mathrm{trun}}(K)$ the event $\mathsf{ApproxResolve}(t, K) = \perp$, i.e. the exception at Line 4 occurs. The following theorem says that this is precisely the error for $\mathsf{ApproxResolve}(t, K)$ sampling from $\mu_{v_{i(t)}}$.

Theorem 4.9. *Let μ be a distribution over $[q]^V$. Assume Condition 4.3 and Condition 4.5. For $-n < t \leq 0$, $K \geq 0$, and $Y = \mathsf{ApproxResolve}(t, K)$, it holds that $d_{\mathrm{TV}}(Y, \mu_v) = \mathbf{Pr}\left[\mathcal{E}_{\mathrm{trun}}(K)\right]$, where $v = v_{i(t)}$.*

Proof. Suppose that sampling from the padding distribution $\mu_{v_{i(\ell)}}^{\mathrm{pad}, \sigma_\Lambda}$ in Line 5 of Algorithm 1 is realised in the same way as in (10), using a sequence of real numbers $U_\ell \in [0, 1)$ chosen uniformly and independently at random for each $\ell \leq 0$.

We apply the following coupling between $\mathsf{Resolve}_\infty(t)$ and $\mathsf{ApproxResolve}(t, K)$:

- the two processes use the same random choices for $(r_\ell)_{\ell \leq 0}$ and $(U_\ell)_{\ell \leq 0}$.

One can verify that if $\mathcal{E}_{\mathrm{trun}}(K)$ does not occur, then the two processes $\mathsf{Resolve}_\infty(t)$ and $\mathsf{ApproxResolve}(t, K)$ are coupled perfectly. By the coupling lemma and Theorem 4.6, we have

$$d_{\mathrm{TV}}(Y, \mu_v) \leq \mathbf{Pr}\left[\mathcal{E}_{\mathrm{trun}}(K)\right].$$

Note that $\mathbf{Pr}\left[Y = \perp\right] = \mathbf{Pr}\left[\mathcal{E}_{\mathrm{trun}}(K)\right]$ and $\mu_v(\perp) = 0$. Therefore, we have

$$d_{\mathrm{TV}}(Y, \mu_v) \geq \mathbf{Pr}\left[Y = \perp\right] - \mu_v(\perp) = \mathbf{Pr}\left[\mathcal{E}_{\mathrm{trun}}(K)\right].$$

Combining the two inequalities proves the theorem. □

A well-known limitation of various perfect sampling approaches, including the coupling from the past (CFTP), is that the sampling procedure is "non-interruptible", which means that forced early termination would introduce a bias in sampling. The same also holds for the coupling towards the past (CTTP) process $\mathsf{Resolve}_\infty(t)$. However, Theorem 4.9 guarantees that this bias due to the interruption is bounded by the truncation probability. In particular, when applying the $\mathsf{ApproxResolve}(t, K)$ to draw approximate samples from the marginal distributions, we are especially interested in the cases where $\mathbf{Pr}\left[\mathcal{E}_{\mathrm{trun}}(K)\right] \leq 1/\mathrm{poly}(n)$ is achieved by a $K = O(\log n)$.

Finally, the "convergence rate" of the $\mathsf{Resolve}_\infty(t)$ procedure, represented by the upper bound for $\mathbf{Pr}\left[\mathcal{E}_{\mathrm{trun}}(K)\right]$, depends very much on how the $\mathsf{Boundary}(t)$ subroutine is implemented, which may vary on concrete models. This is also quite similar to the case of CFTP, where the analyses of convergence rates are also highly dependent on its implementations. Therefore, in the current section, we do not give a generic analysis of the convergence rate of the CTTP procedure. Instead, in the next two sections, we will analyse the efficiencies of CTTP for hypergraph independent set and hypergraph colouring, where the Markov chains with concrete implementations of the $\mathsf{Boundary}(t)$ subroutine are specified.

5. Hypergraph independent set

In this section, we give FPTASes for counting hypergraph independent sets and prove Theorem 1.1 and Theorem 1.2. We first introduce some notations. Given a k-uniform hypergraph $H = (V, \mathcal{E})$ with maximum degree Δ, denote by $\Omega_H \subseteq \{0, 1\}^V$ the set of all independent sets of H, and $Z_H = |\Omega_H|$ its size. Let $\mu = \mu_H$ be the uniform distribution over Ω_H. We identify a subset $S \subseteq V$ with an assignment $\tau_S \in \{0, 1\}^V$ by $\tau_S(v) = 1$ if and only if $v \in S$, for any $v \in V$. Recall that $\tau \in \{0, 1\}^V$ is a hypergraph independent set if every hyperedge $e \in \mathcal{E}$ contains at least one vertex $v \in e$ with $\tau_v = 0$.

Assume $V = \{v_1, v_2, \ldots, v_n\}$ where $n = |V|$. To approximate the partition function Z_H, namely the number of independent sets of H, we use the vertex decomposition defined in (5) with the decomposition scheme $\sigma = \mathbf{0}$ (all-zero vector). This reduces the task to approximating $\mu_{v_i}^{\sigma_{<i}}(0)$ for each $i \in [n]$, the probability that v_i takes the value 0 conditional on all of v_j take 0 where $j < i$. One reason for

choosing the all-zero scheme $\sigma = \mathbf{0}$ is the following marginal lower bound (recall Definition 4.2) by the nature of independent sets.

Observation 5.1. *For any $\Lambda \subseteq V$, any $\sigma_\Lambda \in \{0,1\}^\Lambda$ and any $v \in V \setminus \Lambda$, it holds that*

$$\mu_v^{\sigma_\Lambda}(0) \geq \frac{1}{2} \text{ and } \mu_v^{\sigma_\Lambda}(1) \geq 0.$$

And therefore, $\sigma = \mathbf{0}$ is a $\frac{1}{2}$-bounded vertex decomposition scheme.

The lower bound for $\mu_v^{\sigma_\Lambda}(1)$ can be improved slightly. It will not have any substantial improvement, so for clarity we do not pursue that.

Next, consider the distribution $\mu_{v_i}^{\sigma_{<i}}$ obtained by pinning the partial configuration $\sigma_{<i}$ on the hypergraph H. As another reason of choosing $\sigma = \mathbf{0}$, observe that for each vertex v, if the value of v is fixed to be 0, then all the constraints arising from hyperedges incident to v get immediately satisfied. Therefore, these hyperedges can be safely pruned away from the subinstance. Suppose we are given the partial configuration $\sigma_{<i}$ at some stage during the computation of the product in (5). All vertices with index at most $i - 1$, together with hyperedges incident to *any* of them, can be safely removed. This gives the hypergraph $H_i = (V_i, \mathcal{E}_i)$ where $V_i = \{v_i, v_{i+1}, \ldots, v_n\}$ and $\mathcal{E}_i = \{e \in \mathcal{E} \mid e \subseteq V_i\}$. It is straightforward to verify that

- $\mu_{v_i}^{\sigma_{<i}} = \mu'_{v_i}$, where μ' is the uniform distribution over all independent sets in H_i;
- H_i is a k-uniform hypergraph;
- the maximum degree of H_i is at most that of H;
- H_i is a linear hypergraph if H is a linear hypergraph.

Hence, sampling from the distribution $\mu_{v_i}^{\sigma_{<i}}$ is equivalent to sampling from μ'_{v_i}. Moreover, if H satisfies the condition of Theorem 1.1 (or the condition of Theorem 1.2), then so does H_i. Using the "reduction" from single-site samplers to deterministic counting algorithms as per Corollary 3.3, it suffices to design single-site samplers satisfying the conditions therein. We summarise this into the following two lemmata, one for the general case and the other for the linear case.

Lemma 5.2. *Let $k, \Delta \geq 2$ be two integers satisfying $2^{\frac{k}{2}} \geq \sqrt{8e}k^2\Delta$. There exists an algorithm that given as inputs a k-uniform hypergraph $H = (V, \mathcal{E})$ with maximum degree at most Δ, a vertex $v \in V$ and a parameter $\gamma > 0$, outputs a random $Y_v \in \{0,1\}$ such that $d_{\mathrm{TV}}(Y_v, \mu_v) \leq \gamma$, where μ is the uniform distribution over all independent sets in H. The algorithm runs in time $O(\Delta^3 k^5 \log \frac{1}{\gamma})$ and draws at most $3\Delta^2 k^4 \lceil \log \frac{1}{\gamma} \rceil$ Boolean random variables.*

Lemma 5.3. *Let $\delta > 0$. Let $k \geq \frac{25(1+\delta)^2}{\delta^2}$ and $\Delta \geq 2$ be two integers satisfying $2^k \geq (100k^3\Delta)^{1+\delta}$. There exists an algorithm that given as inputs a k-uniform linear hypergraph $H = (V, \mathcal{E})$ with maximum degree at most Δ, a vertex $v \in V$ and a parameter $\gamma > 0$, outputs a random $Y_v \in \{0,1\}$ such that $d_{\mathrm{TV}}(Y_v, \mu_v) \leq \gamma$, where μ is the uniform distribution over all independent sets in H. The algorithm runs in time $O((\frac{1+\delta}{\delta})^2\Delta^4 k^{10} \log \frac{1}{\gamma})$ and draws at most $10^4(\frac{1+\delta}{\delta})^2\Delta^3 k^9 \lceil \log \frac{1}{\gamma} \rceil$ Boolean random variables.*

Remark 5.4 (perfect marginal samplers). Applying coupling towards the past (Algorithm 1 with $T = \infty$) to the uniform distribution of hypergraph independent sets yields a perfect marginal sampler for the marginal distributions μ_v that outputs Y_v^* distributed exactly as μ_v upon termination. Indeed, the algorithms stated in Lemma 5.2 and Lemma 5.3 are obtained from truncating it. For any $\gamma > 0$, with probability at least $1 - \gamma$, the perfect marginal samplers terminate:

- (on k-uniform hypergraphs) within $O\left(\Delta^3 k^5 \log \frac{1}{\gamma}\right)$ time for computation, while drawing at most $3\Delta^2 k^4 \lceil \log \frac{1}{\gamma} \rceil$ Boolean random variables;

- (on k-uniform linear hypergraphs) within $O\left((\frac{1+\delta}{\delta})^2\Delta^4 k^{10} \log \frac{1}{\gamma}\right)$ time for computation, while drawing at most $10^4(\frac{1+\delta}{\delta})^2\Delta^3 k^9 \lceil \log \frac{1}{\gamma} \rceil$ Boolean random variables;

under the same conditions as stated in Lemma 5.2 and Lemma 5.3, respectively. Similar to CFTP, these perfect samplers are "non-interruptible" in the sense that truncations may bias the sample. Nevertheless, due to the tail bounds above, the truncating error introduced is bounded by γ in the total variation distance.

Theorem 1.1 and Theorem 1.2 are straightforward consequences of Lemma 5.2 and Lemma 5.3 respectively. We first show Lemma 5.2 in Section 5.1, and then adapt the proof to the linear case and show Lemma 5.3 in Section 5.3.

Proofs of Theorem 1.1 and Theorem 1.2. Theorem 1.1 follows from Lemma 5.2 with $\gamma = \frac{\varepsilon}{20n}$, Corollary 3.3 with the vertex decomposition scheme $\sigma = \mathbf{0}$ and Observation 5.1. The running time of the approximate counting algorithm is

$$T = O\left(\left(n + \frac{\Delta n}{k}\right) \cdot \left(\Delta^3 k^5 \log \frac{20n}{\varepsilon}\right) \times 2^{3\Delta^2 k^4 \lceil \log \frac{20n}{\varepsilon} \rceil}\right) = \operatorname{poly}(\Delta k) \left(\frac{n}{\varepsilon}\right)^{O(\Delta^2 k^4)}.$$

Theorem 1.2 is proved the same way but with Lemma 5.3 invoked. The running time of the deterministic approximate counting algorithm is

$$T = O\left(\frac{\Delta n}{k} \cdot \left(\left(\frac{1+\delta}{\delta}\right)^2 \Delta^4 k^{10} \log \frac{20n}{\varepsilon}\right) \cdot 2^{10^4 \left(\frac{1+\delta}{\delta}\right)^2 k^9 \Delta^3 \lceil \log \frac{20n}{\varepsilon} \rceil}\right) = \operatorname{poly}\left(\frac{\Delta k (1+\delta)}{\delta}\right) \left(\frac{n}{\varepsilon}\right)^{O\left(\frac{\Delta^3 k^9 (1+\delta)^2}{\delta^2}\right)}. \quad \square$$

5.1. **Systematic scan Glauber dynamics for hypergraph independent sets.** To apply the general algorithm framework as in Algorithm 3, we first identify the lower bound and marginal distribution in Definition 4.2 using Observation 5.1.

- The lower bound distribution μ^{LB} is given by

$$\mu^{\mathrm{LB}}(0) = \mu^{\mathrm{LB}}(\perp) = \frac{1}{2}, \quad \mu^{\mathrm{LB}}(1) = 0.$$

  Hence, we can assume that μ^{LB} is defined over $\{0, \perp\}$ such that $\mu^{\mathrm{LB}}(0) = \mu^{\mathrm{LB}}(\perp) = \frac{1}{2}$.
- The padding distribution $\mu_v^{\mathrm{pad}, \sigma_\Lambda}$, given any $\Lambda \subseteq V$, any $\sigma_\Lambda \in \{0, 1\}^\Lambda$ and any $v \in V \setminus \Lambda$, is defined by

$$\mu_v^{\mathrm{pad}, \sigma_\Lambda}(0) = 2\mu_v^{\sigma_\Lambda}(0) - 1 \text{ and } \mu_v^{\mathrm{pad}, \sigma_\Lambda}(1) = 2\mu_v^{\sigma_\Lambda}(1).$$

Apparently, the Glauber dynamics for μ satisfies Condition 4.5. We then use Algorithm 3 on the distribution $\mu = \mu_H$ to prove Lemma 5.2 and Lemma 5.3, where the subroutine $\mathsf{Boundary}(t)$ is defined in Algorithm 4. Recall that two oracles $\mathcal{B}$ and $\mathcal{C}$ in Algorithm 4 are defined in Section 4.1.

Algorithm 4: $\mathsf{Boundary}(t)$ for hypergraph independent sets

Input: hypergraph $H = (V, \mathcal{E})$ specifying the distribution μ and an integer $t \leq 0$;
Output: a partial configuration $\sigma_\Lambda \in \{0, 1\}^\Lambda$ over some $\Lambda \subseteq V \setminus \{v_{i(t)}\}$;

1   $v \leftarrow v_{i(t)}, \Lambda \leftarrow \varnothing, \sigma_\Lambda \leftarrow \varnothing$;
2   **forall** $e \in \mathcal{E}$ *s.t.* $v \in e$ **do**
3   **if** *for all* $u \in e \setminus \{v\}$, $\mathcal{B}(\mathrm{pred}_u(t)) = \perp$ **then**
4   **forall** $u \in e \setminus \{v\}$ **do**
5              $\Lambda \leftarrow \Lambda \cup \{u\}$;
6              $\sigma_\Lambda(u) \leftarrow \mathcal{C}(u)$;
7   **if** *for all* $u \in e \setminus \{v\}$, $\sigma_\Lambda(u) = 1$ **then**
8   **return** σ_Λ;
9   **else**
10   **forall** $u \in e \setminus \{v\}$ **do**
11   **if** $\mathcal{B}(\mathrm{pred}_u(t)) \neq \perp$ **then**
12               $\Lambda \leftarrow \Lambda \cup \{u\}$;
13               $\sigma_\Lambda(u) \leftarrow 0$;

14 **return** σ_Λ;

Lemma 5.5. $\mathsf{Boundary}(t)$ *in Algorithm 4 satisfies Condition 4.3.*

Proof. Let X_{t-1} and $(r_j)_{-T < j < t}$ be as defined in $\mathcal{P}(T)$. Then

$$(15) \qquad \forall u \neq v, \qquad \mathcal{B}(\mathrm{pred}_u(t)) \neq\, \perp \quad\Longrightarrow\quad X_{t-1}(u) = \mathcal{B}(\mathrm{pred}_u(t)) = 0,$$

where the first equality is due to the construction of X_{t-1} as in $\mathcal{P}(T)$, and the second one is due to the definition of the lower bound oracle $\mathcal{B}$ and $\mu^{\mathrm{LB}}(1) = 0$.

It is straightforward to see that Algorithm 4 is guaranteed to terminate. We then show that the output σ_Λ of Algorithm 4 satisfies the conditional independence property in Condition 4.3. Evidently $\sigma_\Lambda = X_{t-1}(\Lambda)$: if for some u it holds that $\mathcal{B}(\mathrm{pred}_u(t)) \neq\, \perp$ then it must be assigned 0 in σ_Λ in Line 13 in Algorithm 4, which is consistent with $X_{t-1}(u)$ because of (15); otherwise, its value is assigned using the configuration oracle $C(u)$ in Line 6 in consistency with $X_{t-1}(u)$ by the definition of C. Then verify $\mu_v^{\sigma_\Lambda} = \mu_v^{X_{t-1}(V \setminus \{v\})}$ as below.

- Suppose σ_Λ is returned in Line 8. There exists $e \in \mathcal{E}$ with $v \in e$ such that $e \setminus \{v\} \subseteq \Lambda$ and $X_{t-1}(u) = 1$ for all $u \in e \setminus \{v\}$. Conditional on such σ_Λ, v must take the value 0, the same as conditioning on $X_{t-1}(V \setminus \{v\})$.
- Suppose σ_Λ is returned in Line 14. This implies that the condition in Line 7 has never been satisfied. Hence, for any $e \in \mathcal{E}$ with $v \in e$, we either update σ_Λ in Line 6 or in Line 13. Fix an $e \in \mathcal{E}$ with $v \in e$. If σ_Λ is updated in Line 6, then there is a vertex $u \in e \setminus \{v\}$ such that $\sigma_\Lambda(u) = X_{t-1}(u) = C(u) = 0$. Otherwise, σ_Λ is updated in Line 13. Then there is a vertex $u \in e \setminus \{v\}$ such that $\mathcal{B}(\mathrm{pred}_u(t)) \neq\, \perp$, and by (15), it must take $\sigma_\Lambda(u) = X_{t-1}(u) = 0$. In all, the constraints on all hyperedges incident to v, conditioned on either σ_Λ or $X_{t-1}(V \setminus \{v\})$, are all satisfied, and thus v take the value $\{0, 1\}$ uniformly at random.

In both cases $\mu_v^{\sigma_\Lambda} = \mu_v^{X_{t-1}(V \setminus \{v\})}$. $\qquad\square$

The above argument also indicates the following observation:

Observation 5.6. *If σ_Λ is returned in Line 8, then the padding distribution $\mu_v^{\mathrm{pad}, \sigma_\Lambda}$ takes value 0 with probability 1; if σ_Λ is returned in Line 13, then the padding distribution $\mu_v^{\mathrm{pad}, \sigma_\Lambda}$ takes value 1 with probability 1.*

By adapting the subroutine $\mathsf{Boundary}(t)$ (Algorithm 4) to $\mathsf{ApproxResolve}$ (Algorithm 3) and setting the threshold K therein to $3\Delta^2 k^4 \lceil \log \frac{1}{\gamma} \rceil$, we get an algorithm that approximately samples from the marginal distributions of μ, where μ is uniform over all independent sets in k-uniform hypergraphs with the maximum degree at most Δ. Clearing up this algorithm a bit gives Algorithm 5.

The algorithm maintains two global data structures M and R that might be updated as the algorithm proceeds. Upon the first invocation of this algorithm, these two global variables are initialised as $M = M_0 =\, \perp^{\mathbb{Z}}$ and $R = R_0 = \varnothing$. For simplicity of notations, we may drop both M and R from the list of parameters so long as the references to these two global structures are clear from the context. We may also drop the hypergraph H from the list of parameters as it does not change throughout the process. Recall that the subroutine $\mathsf{LB\text{-}Sample}$ in the algorithm is given in Algorithm 2 with the lower bound distribution μ^{LB} given by $\mu^{\mathrm{LB}}(0) = \mu^{\mathrm{LB}}(\perp) = \frac{1}{2}$.

The following lemma bounds the running time of Algorithm 5.

Lemma 5.7. *The running time of $\mathsf{ApproxMarginIndSet}(t, \gamma; M_0, R_0)$ (Algorithm 5) is $O(\Delta^3 k^5 \log \frac{1}{\gamma})$.*

Proof. Consider the recursion tree generated by executing $\mathsf{ApproxMarginIndSet}(t, \gamma; M_0, R_0)$. Let S be the set of all t_0 such that $\mathsf{ApproxMarginIndSet}(t_0, \gamma; M, R)$ is executed at least once. For any such $t_0 \in S$, one of the following two events must happen: (1) the whole algorithm terminates in Line 11 upon attempting to call $\mathsf{LB\text{-}Sample}$ with time t_0; (2) $(t_0, r_{t_0}) \in R$. Moreover, only one $t^* \in S$ triggers the first case, and hence $|S| \leq |R| + 1$. For any $t_0 \in S$, any execution of $\mathsf{ApproxMarginIndSet}(t_0, \gamma; M, R)$ beyond the first run returns on Line 2. Therefore, the total running time is bounded by $O(T^*|S|)$ where T^* is the run time bound if we assume the recursive calls return in $O(1)$ time. It is easy to see that $T^* = O(\Delta k)$. Thus, the total running time is at most

$$O(\Delta k |S|) = O(\Delta k (|R| + 1)) = O\left(\Delta^3 k^5 \log \frac{1}{\gamma} \right). \qquad\square$$

Algorithm 5: ApproxMarginIndSet$(t, \gamma; M, R)$

Input: a hypergraph $H = (V, \mathcal{E})$, an integer $t \leq 0$ and a real number $0 < \gamma < 1$;
Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\perp\}$ and a set R;
Output: a random value in $\{0, 1\}$

1 **try :**
2   **if** $M(t) \neq \perp$ **then return** $M(t)$;
3   **if** LB-Sample$(t; R) = 0$ **then** $M(t) \leftarrow 0$ and **return** 0;
4     $v \leftarrow v_{i(t)}$;
5   **forall** $e \in \mathcal{E}$ s.t. $v \in e$ **do**
6   **if** *for all* $u \in e \setminus \{v\}$, LB-Sample$(\text{pred}_u(t); R) = \perp$ **then**
7   **if** *for all* $u \in e \setminus \{v\}$, ApproxMarginIndSet$(\text{pred}_u(t), \gamma; M, R) = 1$ **then**
8             $M(t) \leftarrow 0$ and **return** 0;

9     $M(t) \leftarrow 1$ and **return** 1;
10 **catch** $|R| \geq 3\Delta^2 k^4 \lceil \log \frac{1}{\gamma} \rceil$ **:**
11   **return** 0;

The correctness of Algorithm 5 is due to the following lemma. Its proof is given in Section 5.2.

Lemma 5.8. *Let $0 < \gamma < 1$. Let $H = (V, \mathcal{E})$ be a k-uniform hypergraph with maximum degree at most Δ such that $2^{\frac{k}{2}} \geq \sqrt{8e}k^2\Delta$, and μ be the uniform distribution over all the independent sets in H. Then for any vertex $v \in V$, the output distribution Y of ApproxMarginIndSet$(\text{pred}_v(0), \gamma)$ satisfies $d_{\text{TV}}(Y, \mu_v) \leq \gamma$.*

Assuming the lemma above for now, we finish off Lemma 5.2.

Proof of Lemma 5.2. By Lemma 5.8, for any $v \in V$, the algorithm ApproxMarginIndSet$(\text{pred}_v(0), \gamma)$ generates a distribution that is γ-close to μ_v. Its running time is given in Lemma 5.7. As each entry in R must have been fetched from the random oracle exactly once, the number of calls to the random oracle equals $|R|$, which is bounded from above by $3\Delta^2 k^4 \lceil \log \frac{1}{\gamma} \rceil$. The random variables returned by the random oracle lie in $\{0, \perp\}$ because $\mu^{\text{LB}}(1) = 0$. $\qquad\square$

5.2. **Analysis of the truncation error.** Unless stated otherwise, the hypergraph $H = (V, \mathcal{E})$ is fixed and satisfies the same condition as in Lemma 5.8 throughout this section. Let $t^* \leq 0$ be the timestamp of the initial call of ApproxMarginIndSet.

For any $\gamma > 0$, consider the event that ApproxMarginIndSet(t^*, γ) terminates by Line 11. Alternatively, it is equivalent to that the size of R reaches $3\Delta^2 k^4 \lceil \log \frac{1}{\gamma} \rceil$ midst the execution of the algorithm (even if we assume the algorithm did not truncate). By Theorem 4.9, it suffices to bound the probability of this event in order to bound the bias of the output distribution from the uniform distribution. This is given by the following lemma.

Lemma 5.9. *Let $\gamma > 0$ be a real, and $H = (V, \mathcal{E})$ be a k-uniform hypergraph with maximum degree at most Δ such that $2^{\frac{k}{2}} \geq \sqrt{8e}k^2\Delta$. Let $\eta = \lceil \log \frac{1}{\gamma} \rceil$. Upon the termination of ApproxMarginIndSet(t^*, γ), the size of R satisfies*

$$(16) \qquad\qquad \mathbf{Pr}\left[|R| \geq 3\Delta^2 k^4 \cdot \eta\right] \leq 2^{-\eta}.$$

Lemma 5.8 follows directly from the above lemma, Lemma 5.5 and Theorem 4.9.

We now prove Lemma 5.9. For any $e \in \mathcal{E}$ and $t \in \mathbb{Z}_{\leq 0}$, let

$$(17) \qquad\qquad \text{TS}(e, t) := \{\text{pred}_v(t) \mid v \in e\}$$

be the set of timestamps of the latest consecutive updates of vertices in e up to time t. Given a hypergraph $H = (V, \mathcal{E})$, we introduce the following definition of the witness graph $G_H = (V_H, E_H)$.

Definition 5.10 (witness graph). For a hypergraph $H = (V, \mathcal{E})$, the *witness graph* $G_H = (V_H, E_H)$ is an infinite graph with a vertex set

$$V_H = \{\mathsf{TS}(e, t) \mid e \in \mathcal{E}, t \in \mathbb{Z}_{\leq 0}\},$$

and there is an undirected edge between $x, y \in V_H$ in G_H if and only if $x \neq y$ and $x \cap y \neq \varnothing$.

We remark that the vertex set V_H is not a multiset, i.e., it does not allow multiple instances of the same element. For example, it is possible that for some $t \neq t'$, $\mathsf{TS}(e, t) = \mathsf{TS}(e, t')$. In this case, the two correspond to the same vertex even if $t \neq t'$. In addition, the witness graph G_H is decided only by the hypergraph H as $\mathrm{pred}_x(t)$ is a deterministic function. For any vertex $x \in V_H$ in the witness graph, let $e_H(x) \in \mathcal{E}$ be the edge in the original hypergraph that corresponds to x.

We first bound the degree of the witness graph. Define the following neighbourhoods of x in the witness graph G_H:

(18)
$$
\begin{aligned}
N_{\mathrm{self}}(x) &:= \{y \in V_H \mid \{x, y\} \in E_H \text{ and } e_H(x) = e_H(y)\}, \\
N_{\mathrm{out}}(x) &:= \{y \in V_H \mid \{x, y\} \in E_H \text{ and } e_H(x) \neq e_H(y)\}, \\
N(x) &:= N_{\mathrm{self}}(x) \uplus N_{\mathrm{out}}(x).
\end{aligned}
$$

In other words, we partition the neighbourhood $N(x)$ of x in the witness graph G_H into two parts, one containing those corresponding to the same edge in the original hypergraph, and the other one collecting the rest.

Lemma 5.11. *For any vertex $x \in V_H$, it holds that $|N_{\mathrm{self}}(x)| \leq 2k - 2$ and $|N_{\mathrm{out}}(x)| \leq (2k-1)k(\Delta - 1)$, and therefore, the maximum degree of G_H is bounded by $2\Delta k^2 - 2$.*

Proof. Fix arbitrarily an edge $e \in \mathcal{E}$ in the original hypergraph H and a time $t \in \mathbb{Z}_{\leq 0}$. This induces a vertex x in the witness graph G_H. We claim that for any $e' \in \mathcal{E}$ such that $e \cap e' \neq \varnothing$, it holds that

$$|\{\mathsf{TS}(e', t') \mid t' \in \mathbb{Z}_{\geq 0} \text{ and } \mathsf{TS}(e', t') \cap \mathsf{TS}(e, t) \neq \varnothing\}| \leq 2k - 1.$$

This claim implies the lemma because

- by taking $e' = e$, we have $|N_{\mathrm{self}}(x)| \leq 2k - 2$ (we need to exclude x itself); and
- by taking e' such that $e' \neq e$ and $e' \cap e \neq \varnothing$, we have $|N_{\mathrm{out}}(x)| \leq (2k-1)k(\Delta - 1)$, since there are at most $(\Delta - 1)k$ choices for such e'.

We then verify the claim. Assume $\mathsf{TS}(e, t) = (t_1, t_2, \ldots, t_k)$ where $t_1 < t_2 < \cdots < t_k < t_1 + n$, and $\mathsf{TS}(e', t') = (t_1', t_2', \ldots, t_k')$ where $t_1' < t_2' < \cdots < t_k' < t_1' + n$. Observe that $\mathsf{TS}(e', t') \cap \mathsf{TS}(e, t) \neq \varnothing$ implies $e \cap e' \neq \varnothing$. This means the following two quantities

$$
\begin{aligned}
j_{\min} &:= \min\left\{j \in [k] \mid v_{i(t_j)} \in e'\right\}, \quad \text{and} \\
j_{\max} &:= \max\left\{j \in [k] \mid v_{i(t_j)} \in e'\right\}
\end{aligned}
$$

are well defined.

Observe that $t_k' \geq t_{j_{\min}}$ and $t_1' \leq t_{j_{\max}}$. This can be argued as follows. Assume towards contradiction that $t_k' < t_{j_{\min}}$. As $\mathsf{TS}(e', t') \cap \mathsf{TS}(e, t) \neq \varnothing$, there must be such j and j' that $t_j = t_{j'}' \in \mathsf{TS}(e', t') \cap \mathsf{TS}(e, t)$. Note that $t_j = t_{j'}' < t_{j_{\min}}$ because of the ordering $t_{j'}' \leq t_k'$ and the assumption. This contradicts with the choice of $j_{\min}$. The same argument goes for the other inequality.

The two inequalities above, together with $t_k' < t_1' + n$, implies $t_1' \in (t_{j_{\min}} - n, t_{j_{\max}}]$. Note further that in the interval $[t_{j_{\min}} - n, t_{j_{\max}}]$, there are at most $2k$ timestamps when there is a vertex in e' that gets updated: k from $[t_{j_{\min}} - n, t_{j_{\min}})$ and at most k from $[t_{j_{\min}}, t_{j_{\max}}]$ By definition of $j_{\min}$, the vertex that gets updated at time $t_{j_{\min}}$ must come from e'. Therefore, there are at most $2k - 1$ timestamps in the interval $(t_{j_{\min}} - n, t_{j_{\max}}]$ that update vertices in e', and hence at most $2k - 1$ choices for t_1'. The claim then follows by observing that the sequence $\mathsf{TS}(e', t') = (t_1', t_2', \cdots, t_k')$ is uniquely determined by t_1'. $\qquad \square$

We analyse how ApproxMarginIndSet branches. For $t_0 < t_1 \leq t^*$, if ApproxMarginIndSet(t_0, γ) originates from ApproxMarginIndSet(t_1, γ), then we say the recursion ApproxMarginIndSet(t_0, γ) is triggered by $x = \mathsf{TS}(e, t_1) \in V_H$, where e is the edge that the caller is handling. Also note that the same

t_0 might be triggered by various different t_1's, but only the first recursion ApproxMarginIndSet(t_0, γ) can trigger more recursive calls. Let

(19)
$$V_{\mathcal{B}} := \{x \in V_H \mid x \text{ triggers recursive calls } \}$$

be a random subset of V_H. In fact, its randomness only comes from that used by the algorithm. Upon the termination of the algorithm, it generates the set R of randomness used over time, and defines the set $V_{\mathcal{B}}$. The size of R can be upper bounded in terms of $|V_{\mathcal{B}}|$.

Lemma 5.12. $|R| \le (|V_{\mathcal{B}}| + 1)k^2\Delta$.

Proof. Arbitrarily fix the values of all random bits $(r_t)_{t \le 0}$. Given this, ApproxMarginIndSet is deterministic. We show that the lemma always holds.

A timestamp t_0 is said to be active if ApproxMarginIndSet(t_0, γ) is invoked. Let A be the set of all active timestamps except the initial call of ApproxMarginIndSet. For any $t_0 \in A$, consider the first invocation of ApproxMarginIndSet(t_0, γ). It invokes the subroutine LB-Sample in Line 3 and Line 6. The number of invocations in Line 3 is 1, and the number of invocations in Line 6 is bounded by $k\Delta - 1$, as each vertex is incident to at most Δ hyperedges and each hyperedge contains k vertices. For any call of ApproxMarginIndSet(t_0, γ) beyond the first run, the algorithm directly returns a value in Line 2 and does not invoke LB-Sample. Together with the initial call at time t^*, we have

$$|R| \le (|A| + 1)k\Delta.$$

For each $t_0 \in A$, consider the first invocation of ApproxMarginIndSet(t_0, γ) and suppose it is invoked by ApproxMarginIndSet(t_1, γ) when it is processing the hyperedge e. This actually defines a map $f : A \to V_{\mathcal{B}}$ where $f(t_0) = \mathsf{TS}(e, t_1)$. By definition, for any t' such that $f(t') = x'$, it must hold that $t' \in x'$ and $|x'| = k$. This implies that for each $x' \in V_{\mathcal{B}}$, there are at most k different $t' \in A$ such that $f(t') = x'$. This gives

(20)
$$|A| \le k|V_{\mathcal{B}}|.$$

Hence $|R| \le (|A| + 1)k\Delta \le (|V_{\mathcal{B}}| + 1)k^2\Delta$. $\qquad\square$

Lemma 5.13. *The subgraph of G_H induced by $V_{\mathcal{B}}$ is connected. Furthermore, if $V_{\mathcal{B}} \ne \varnothing$, then there exists $x \in V_{\mathcal{B}}$ such that $t^* \in x$.*

Proof. Again, arbitrarily fix the values of all random bits $(r_t)_{t \le 0}$. Given this, ApproxMarginIndSet is deterministic. We show that the lemma always holds. We may also assume $V_{\mathcal{B}} \ne \varnothing$ or otherwise the lemma trivially holds.

Order the set $V_{\mathcal{B}} = \{x^{(1)}, x^{(2)}, \ldots, x^{(\ell)}\}$ by the time a vertex joins this set, i.e., triggers a recursion. Here $x^{(i)}$ is the i-th vertex that triggers a recursion. The last index ℓ is finite because ApproxMarginIndSet(t, γ) terminates within a finite number of steps. We claim that for any $i \in [\ell]$, there exists a path $P = (y_1, y_2, \ldots, y_m)$ in G_H such that

- $t^* \in y_1$ and $x^{(i)} = y_m$;
- for all $j \in [m]$, $y_j \in V_{\mathcal{B}}$.

This immediately proves the lemma. We prove the claim by induction on index i.

Base case. Consider the initial invocation ApproxMarginIndSet(t^*, γ). It must trigger some recursion, or otherwise $V_{\mathcal{B}} = \varnothing$, violating our assumption. Therefore, $t^* \in x^{(1)}$, and we can simply take the path $P = (x^{(1)})$.

Induction step. Fix an integer $1 < j \le \ell$. Suppose the claim holds for all $x^{(i)}$ for $i < j$. We prove this for $x^{(j)}$. If $x^{(j)}$ joins $V_{\mathcal{B}}$ whilst ApproxMarginIndSet(t^*, γ) executes, then we can simply take $P = (x^{(j)})$. Otherwise, suppose $x^{(j)}$ joins $V_{\mathcal{B}}$ whilst ApproxMarginIndSet(t_0, γ) executes for some $t_0 < t^*$, and this call is invoked by another ApproxMarginIndSet(t_1, γ) where $t_0 < t_1 \le t^*$ when the caller is processing some hyperedge $e \in \mathcal{E}$ in Line 7. Then there exists $i < j$ such that $x^{(i)} = \mathsf{TS}(e, t_1)$. By induction hypothesis, there is a path $P = (y_1, y_2, \ldots, y_{m'})$ for $x^{(i)}$ with $y_{m'} = x^{(i)}$. Note that $t_0 \in x^{(i)}$ because $x^{(i)}$ triggers a recursive call of ApproxMarginIndSet(t_0, γ), and $t_0 \in x^{(j)}$ because v_j joins $V_{\mathcal{B}}$ whilst ApproxMarginIndSet(t_0, γ). Thus the edge $(x^{(i)}, x^{(j)})$ exists in the witness graph, and we can construct the new path as $P' = (P, x^{(j)})$ to meet the conditions. $\qquad\square$

Lemma 5.14. *For all $x \in V_{\mathcal{B}}$ and all $t' \in x$, it holds that $r_{t'} = \perp$.*

Proof. Fix a $x = \mathsf{TS}(e, t_0) \in V_{\mathcal{B}}$. As x triggers another subroutine in Line 7, the current hyperedge e must satisfy the condition in Line 6. This means $r_{t'} = \perp$ for all $t' \in x \setminus \{t_0\}$ (recall the definition in (17)). Moreover, $r_{t_0} = \perp$, as otherwise $r_{t_0} = 0$ (recall the lower bound distribution μ^{LB}), in which case $\mathsf{ApproxMarginIndSet}(t_0, \gamma)$ must have terminated in Line 3 before reaching Line 7. $\qquad\square$

To prove Lemma 5.9, we need the notion of 2-trees [Alo91]. Given a graph $G = (V, E)$, its power graph $G^2 = (V, E_2)$ has the same vertex set, while an edge $(u, v) \in E_2$ if and only if $1 \le \mathrm{dist}_G(u, v) \le 2$.

Definition 5.15 (2-tree). Let $G = (V, E)$ be a graph. A set of vertices $T \subseteq V$ is called a 2-*tree* of G, if
- for any $u, v \in T$, $\mathrm{dist}_G(u, v) \ge 2$, and
- T is connected on G^2.

Intuitively, a 2-tree is an independent set that does not spread far away. The next lemma bounds the number of 2-trees of a certain size containing a given vertex.

Lemma 5.16 ([FGYZ21a, Corollary 5.7]). *Let $G = (V, E)$ be a graph with maximum degree D, and $v \in V$ be a vertex. The number of 2-trees in G of size $\ell \ge 2$ containing v is at most $\frac{(\mathrm{e}D^2)^{\ell-1}}{2}$.*

The next lemma shows the existence of a large 2-tree.

Lemma 5.17 ([FGYZ21a, Observation 5.5] and [JPV21a, Lemma 4.5]). *Let $G = (V, E)$ be a graph of maximum degree D, $H = (V(H), E(H))$ be a connected finite subgraph of G, and $v \in V(H)$ a vertex in H. Then there exists a 2-tree T of H containing v such that $|T| = \lfloor |V(H)|/(D+1) \rfloor$.*

Proof of Lemma 5.9. By Lemma 5.12, it suffices to show $\mathbf{Pr}\left[|V_{\mathcal{B}}| \ge 2\Delta k^2 \cdot \eta\right] \le 2^{-\eta}$ for any integer $\eta \ge 1$, as

$$\mathbf{Pr}\left[|R| \ge 3\Delta^2 k^4 \cdot \eta\right] \le \mathbf{Pr}\left[(|V_{\mathcal{B}}| + 1)\Delta k^2 \ge 3\Delta^2 k^4 \cdot \eta\right] \le \mathbf{Pr}\left[|V_{\mathcal{B}}| \ge 2\Delta k^2 \cdot \eta\right] \le \left(\frac{1}{2}\right)^{\eta}.$$

Fix an integer $\eta \ge 1$ and assume $|V_{\mathcal{B}}| \ge 2\Delta k^2 \cdot \eta$. Recall that $V_{\mathcal{B}}$ is finite because $\mathsf{ApproxMarginIndSet}$ terminates within a finite number of steps. By Lemma 5.13, there exists $y \in V_{\mathcal{B}}$ such that $t^* \in y$. Also by Lemma 5.11 and Lemma 5.17, there exists a 2-tree $T \subseteq V_{\mathcal{B}}$ of size i such that $y \in T$. For $x \in V_H$, let $\mathcal{T}_x^{\eta}$ denote the set of 2-trees of V_H of size η containing x. Then by Lemma 5.13, Lemma 5.17, and a union bound over all 2-trees, we have

$$\mathbf{Pr}\left[|V_{\mathcal{B}}| \ge 2\Delta k^2 \cdot \eta\right] \le \sum_{T \in \mathcal{T}_x^{\eta}} \mathbf{Pr}\left[T \subseteq V_{\mathcal{B}}\right].$$

By Lemma 5.14, the event $T \subseteq V_{\mathcal{B}}$ implies $r_t = \perp$ for all timestamps t involved in T. Also note that, any pair x, y of vertices in T do not share timestamps. Thus, we have $\mathbf{Pr}\left[T \subseteq V_{\mathcal{B}}\right] \le 2^{-k \cdot |T|}$. Using Lemma 5.11 and Lemma 5.16, it holds that

$$\sum_{T \in \mathcal{T}_x^{\eta}} \mathbf{Pr}\left[T \subseteq V_{\mathcal{B}}\right] \le 2\Delta k^2 \cdot (4\mathrm{e}\Delta^2 k^4)^{\eta-1} \cdot \left(\frac{1}{2}\right)^{k\eta} \le \left(\frac{4\mathrm{e}\Delta^2 k^4}{2^k}\right)^{\eta} \le 2^{-\eta},$$

where the last inequality is due to $2^k \ge 8\mathrm{e}\Delta^2 k^4$. $\qquad\square$

5.3. Improved bounds for linear hypergraphs.

We now give a marginal sampler for linear hypergraphs and prove Lemma 5.3. Let $\delta > 0$ be a constant. Let $k \ge \frac{25(1+\delta)^2}{\delta^2}$ and $\Delta \ge 2$ be two integers satisfying $2^k \ge (100k^3\Delta)^{1+\delta}$. Given as inputs a linear k-uniform hypergraph $H = (V, \mathcal{E})$ with maximum degree Δ and a parameter $\gamma > 0$, the algorithm is the same as Algorithm 5, except that the truncation condition in Line 10 is changed from $|R| \ge 3\Delta^2 k^4 \left\lceil \log \frac{1}{\gamma} \right\rceil$ to:

$$(21) \qquad\qquad |R| \ge 10^4 \left(\frac{1+\delta}{\delta}\right)^2 \Delta^3 k^9 \left\lceil \log \frac{1}{\gamma} \right\rceil.$$

Much of the analysis for the general case can be applied to the linear case, with a much refined condition on the maximum degree.

The running time of the modified algorithm can be bounded in the same way as Lemma 5.7.

Lemma 5.18. *The running time of the modified algorithm is $O\left((\frac{1+\delta}{\delta})^2 \Delta^4 k^{10} \log \frac{1}{\gamma}\right)$.*

Next, we bound the truncation error.

Lemma 5.19. *Denote $\eta = \lceil \log \frac{1}{\gamma} \rceil$. Upon the termination of the modified algorithm, the size of R satisfies*

$$(22) \qquad \mathbf{Pr}\left[|R| \geq 10^4 \left(\frac{1+\delta}{\delta}\right)^2 \Delta^3 k^9 \cdot \eta\right] \leq 2^{-\eta}.$$

Lemma 5.3 then follows by Lemma 5.18, Lemma 5.19, and Theorem 4.9. The rest of this section is dedicated to the proof of Lemma 5.19.

As the condition of maximum degree is much more refined, we need to rely heavily on the property that the original hypergraph H is linear. Recall that in the analysis in Section 5.2, we were actually dealing with the witness graph G_H (Definition 5.10). However, for some pair of adjacent vertices in the witness graph G_H, the overlap $|x \cap y|$ is not necessarily 1 even the underlying hypergraph H is linear. To see this, recall the partition of neighbourhood of $x \in V_H$ constructed in (18). For any neighbour y that corresponds to a different edge in H, i.e., $y \in N_{\text{out}}(x)$, it indeed holds that $|x \cap y| = 1$; however, for those $u \in N_{\text{self}}(v)$, there is no guarantee on the size of the intersection $|x \cap y|$. To handle such kind of "neighbours" that are actually doppelgängers, we introduce the following new notion of the *self-neighbourhood powered witness graph*.

Definition 5.20. Let $G_H = (V_H, E_H)$ be the witness graph as in Definition 5.10. The self-neighbourhood powered witness graph $G_H^{\text{self}} = (V_H, E_H^{\text{self}})$ is defined on the same vertex set as G_H. Its edge set $E_H^{\text{self}} = E_H \cup E'$ is given by

$$E' = \{\{x, y\} \mid (\exists w \in V_H \text{ s.t. } w \in N_{\text{self}}(y) \wedge w \in N(x)) \vee (\exists w \in V_H \text{ s.t. } w \in N(y) \wedge w \in N_{\text{self}}(x))\},$$

where $N_{\text{self}}(x)$ and $N(x)$ are defined as in (18).

We first bound the maximum degree of G_H^{self}.

Lemma 5.21. *The maximum degree of G_H^{self} is at most $10k^3 \Delta - 1$.*

Proof. For any $x \in V_H$, by Lemma 5.11, $|N_{\text{self}}(x)| \leq 2k$ and $N(x) \leq 2\Delta k^2 - 1$. Hence, the maximum degree of G_H^{self} is at most $2\Delta k^2 - 1 + 2 \times 2k \times (2\Delta k^2 - 1) \leq 10k^3 \Delta - 1$. □

Recall from the proof of general case that we only need to analyse the size of a connected vertex set $V_{\mathcal{B}}$ in G_H, as Lemma 5.12 still applies. To employ linearity at some point, we instead work on G_H^{self}. As there are more edges in G_H^{self} than in G_H, small components are more likely to emerge. Therefore, for any connected component $V_{\mathcal{B}}$ in G_H, we can find a subset $V_{\mathcal{B}}^{\text{lin}} \subseteq V_{\mathcal{B}}$ that is connected in G_H^{self}, fulfilling linearity. This is formally described as the following lemma.

Lemma 5.22. *Given a k-uniform linear hypergraph $H = (V, \mathcal{E})$ with the maximum degree Δ, let $G_H = (V_H, E_H)$ be its witness graph. Fix a vertex $x \in V_H$, and let $V_{\mathcal{B}} \subseteq V_H$ be a finite subset containing x and connected in G_H. Then, there exists $V_{\mathcal{B}}^{\text{lin}} \subseteq V_{\mathcal{B}}$ such that*

(L1) $x \in V_{\mathcal{B}}^{\text{lin}}$ and $|V_{\mathcal{B}}^{\text{lin}}| \geq \lfloor \frac{|V_{\mathcal{B}}|}{2k+1} \rfloor$,
(L2) the induced subgraph $G_H^{\text{self}}[V_{\mathcal{B}}^{\text{lin}}]$ is connected, and
(L3) for any two distinct vertices $x_1, x_2 \in V_{\mathcal{B}}^{\text{lin}}$, it holds that $|x_1 \cap x_2| \leq 1$.

Proof. We construct one such $V_{\mathcal{B}}^{\text{lin}}$ explicitly. Let $G = G_H[V_{\mathcal{B}}]$. The condition in the lemma says G is connected. For any vertex $y \in V_B$, denote by $\Gamma_G(y)$ the neighbourhood of y in G, and for any subset $\Lambda \subseteq V_B$, define

$$\Gamma_G(\Lambda) := \{y \in V_{\mathcal{B}} \setminus \Lambda \mid \exists w \in \Lambda \text{ s.t. } y \in \Gamma_G(w)\}.$$

The set $V_{\mathcal{B}}^{\text{lin}}$ is constructed by the following algorithm.

- Initialise $V_{\mathcal{B}}^{\lin} \leftarrow \varnothing$, $\Lambda \leftarrow \varnothing$ and $V \leftarrow V_{\mathcal{B}}$.
- Repeat the following until $V = \varnothing$:
  (1) if $\Lambda = \varnothing$, let $y \leftarrow x$; otherwise, let y be an arbitrary vertex in $\Gamma_G(\Lambda)$;
  (2) $V_{\mathcal{B}}^{\lin} \leftarrow V_{\mathcal{B}}^{\lin} \cup \{y\}$;
  (3) $\Lambda \leftarrow \Lambda \cup \{y\} \cup (V_{\mathcal{B}} \cap N_{\self}(y))$.
  (4) $V \leftarrow V_{\mathcal{B}} \setminus \Lambda$.

The algorithm above is well defined and terminates in finite steps. First, the vertex y in Line (1) always exists. To see this, we only need to consider the case $\Lambda \neq \varnothing$. Then $V \neq \varnothing$ because the algorithm has not yet terminated. Note that $\Lambda \uplus V = V_{\mathcal{B}}$ upon entering Line (1), which implies $\Lambda \neq V_{\mathcal{B}}$. In addition, G is connected, and hence $\Gamma_G(\Lambda) \neq \varnothing$, implying the existence of such y. Secondly, the algorithm will eventually terminate because the set V is initialised to be $V_{\mathcal{B}}$ which is finite, and the size of V decreases by at least 1 after each iteration.

We then verify that the output $V_{\mathcal{B}}^{\lin}$ satisfies the conditions in the lemma.

(L1): Apparently $x \in V_{\mathcal{B}}^{\lin}$. By Lemma 5.11, we add at most $2k + 1$ vertices into Λ in Line (3), thus remove at most $2k + 1$ vertices from $|V|$ in each iteration. This implies $|V_{\mathcal{B}}^{\lin}| \geq \lfloor \frac{|V_{\mathcal{B}}|}{2k+1} \rfloor$.

(L2): We show by a simple induction that $G_H^{\self}[V_{\mathcal{B}}^{\lin}]$ is connected during the whole process. As a base case, $V_{\mathcal{B}}^{\lin} = \{x\}$ after the first iteration, and hence the claim holds. For the upcoming iterations, we always pick $y \in \Gamma_G(\Lambda)$, thus there exists $w \in \Lambda$ such that y and w are adjacent in G, and thus adjacent in G_H. There are two cases for w joining Λ in Line (3) in earlier iterations, that either (a) $w \in V_{\mathcal{B}}^{\lin}$, or (b) $w \in N_{\self}(w')$ for some $w' \in V_{\mathcal{B}}^{\lin}$. In Case (a), y and w are adjacent in G_H, and thus they are adjacent $G_H^{\self}$. In Case (b), y and w' are adjacent $G_H^{\self}$. The claim then follows by the principle of induction.

(L3): Fix $x_1, x_2 \in V_{\mathcal{B}}^{\lin}$. Suppose the algorithm first adds x_1 into $V_{\mathcal{B}}^{\lin}$ and then x_2. After x_1 gets added, the algorithm puts all $(V_{\mathcal{B}} \cap N_{\self}(x_1))$ into Λ, implying $x_2 \notin N_{\self}(x_1)$. Hence, there are only two cases for x_2: either (a) $x_2 \notin N(x_1)$, and thus $x_1 \cap x_2 = \varnothing$; or (b) $x_2 \in N_{\out}(x_1)$, in which case, since the hypergraph H is linear, $|x_1 \cap x_2| = 1$. □

To make the most of linearity, we use the following 2-block trees, first introduced in [FGW22], instead of 2-trees as in the general case.

Definition 5.23 (2-block-tree). Let $\theta, \ell \geq 1$ be two integer parameters, and $G = (V, E)$ be a graph. We call a collection of vertex sets $\{C_1, C_2, \cdots, C_\ell\}$ a *2-block-tree* of block size θ and tree size ℓ in G, if

- for any $1 \leq i \leq \ell$, the set $C_i \subset V$ has size θ, and the induced subgraph $G[C_i]$ is connected;
- $\dist_G(C_i, C_j) := \min_{u \in C_i, v \in C_j} \dist_G(u, v) \geq 2$ for any distinct $1 \leq i, j \leq \ell$;
- $\bigcup_{i=1}^{\ell} C_i$ is connected on G^2, where u and v are adjacent in G^2 if and only if $1 \leq \dist_G(u, v) \leq 2$.

In fact, a 2-tree is a 2-block-tree but with $\theta = 1$. As we see from last section, the point of using 2-trees is to secure a bound on the independent vertices in the original hypergraph. Therefore, we only look at independent hyperedges, by which we may drop too much. The observation is that, if the hypergraph is simple, then a block, as defined above, has a much better lower bound on the number of distinct vertices in the original hypergraph than just dropping dependent hyperedges, so long as $\theta \ll k$.

We can construct a 2-block-tree out of a connected induced subgraph.

Lemma 5.24. *Let $\theta \geq 1$ be an integer. Let $G = (V, E)$ be a graph with maximum degree Δ. Given a finite subset $C \subseteq V$ such that the subgraph $G[C]$ induced by C is connected, and a vertex $v \in C$, then there exists a 2-block-tree $\{C_1, C_2, \ldots, C_\ell\}$ in G with block size θ and tree size $\ell = \lfloor |C|/(\theta^2 \Delta^2) \rfloor$ such that $v \in C_1$ and $C_i \subseteq C$ for all $i \in [\ell]$.*

The above lemma is proved by the technique in [FGW22, Proposition 16], which we shall defer to Appendix A for completeness. We comment that [FGW22, Proposition 16] actually proves some additional results, but here in Lemma 5.24, we only requires $\ell = \Omega_{\theta, \Delta}(|C|)$ as this is enough for our application.

We also need a bound on the number of 2-block-trees.

Lemma 5.25 ([FGW22, Lemma 20]). *Let $\theta \geq 1$ be an integer. Let $G = (V, E)$ be a graph with maximum degree Δ. For any integer $\ell \geq 1$, any vertex $v \in V$, the number of 2-block-trees $\{C_1, C_2, \ldots, C_\ell\}$ with block size θ and tree size ℓ such $v \in \cup_{i=1}^{\ell} C_i$ is at most $(\theta e^\theta \Delta^{\theta+1})^\ell$.*

Now we prove Lemma 5.19.

Proof of Lemma 5.19. Let $V_{\mathcal{B}}$ be the set generated by the modified algorithm (see (21)) as defined in (19). Again, $V_{\mathcal{B}} \subseteq V_H$ is a finite subset because the algorithm terminates after a finite number of steps. Choose the parameter

$$\theta := \left\lceil \frac{4(1+\delta)}{\delta} \right\rceil,$$

and by this choice, $\mathbf{Pr}\left[|R| \geq 10^4 (\frac{1+\delta}{\delta})^2 k^9 \Delta^3 \eta\right] \leq \mathbf{Pr}\left[|R| \geq 400\theta^2 k^9 \Delta^3 \eta\right]$. Using Lemma 5.12, it holds for any positive integer η that

$$\mathbf{Pr}\left[|R| \geq 400\theta^2 k^9 \Delta^3 \eta\right] \leq \mathbf{Pr}\left[(|V_{\mathcal{B}}| + 1)k^2 \Delta \geq 400\theta^2 k^9 \Delta^3 \eta\right] \leq \mathbf{Pr}\left[|V_{\mathcal{B}}| \geq 300\theta^2 k^7 \Delta^2 \eta\right].$$

Hence, it suffices to show

$$\mathbf{Pr}\left[|V_{\mathcal{B}}| \geq 300\theta^2 k^7 \Delta^2 \eta\right] \leq \left(\frac{1}{2}\right)^\eta$$

for any integer $\eta \geq 1$.

Fix an integer $\eta \geq 1$ and assume $|V_{\mathcal{B}}| \geq 300\theta^2 k^7 \Delta^2 \eta$. As the set $V_{\mathcal{B}}$ is non-empty, by Lemma 5.13, there exists $x \in V_{\mathcal{B}}$ such that $t^* \in x$. Fix such a vertex x. By Lemma 5.22, we can find the set $V_{\mathcal{B}}^{\text{lin}} \subseteq V_{\mathcal{B}}$ with size $|V_{\mathcal{B}}^{\text{lin}}| \geq \lfloor \frac{|V_{\mathcal{B}}|}{2k+1} \rfloor \geq \lfloor \frac{|V_{\mathcal{B}}|}{3k} \rfloor \geq \theta^2 (10k^3 \Delta)^2 \eta$ containing x and fulfilling the conditions in Lemma 5.22, and it is straightforward to find a subset $U \subseteq V_{\mathcal{B}}^{\text{lin}}$ with size *exactly* $|U| = \theta^2 (10k^3 \Delta)^2 \eta$ such that $x \in U$ and the rest of Lemma 5.22 are satisfied by U. By Lemma 5.21, the maximum degree of $G_H^{\text{self}}[U]$ is at most $10k^3 \Delta$. Since $G_H^{\text{self}}[U]$ is a finite connected subgraph in G_H^{self}, by Lemma 5.24, we can find a 2-block-tree $\{C_1, C_2, \ldots, C_\eta\}$ in G_H^{self} with block size θ and tree size η such that $x \in C_1$ and $C_j \subseteq U \subseteq V_{\mathcal{B}}$ for all $j \in [\eta]$. By (L3) in Lemma 5.22, for any distinct $x_1, x_2 \in \cup_{j=1}^{\eta} C_j$, it holds that $|x_1 \cap x_2| \leq 1$.

The discussion above motivates the following notation. For any x, let $\mathcal{T}_x^{\eta,\theta}$ be the set of all 2-block-trees $\{C_1, C_2, \cdots, C_\eta\}$ with block size θ and tree size η in G_H^{self} such that

- $x \in C_1$;
- let $C = \cup_{j=1}^{\eta} C_j$, then for any $w_1, w_2 \in C$, $|w_1 \cap w_2| \leq 1$.

Hence, if $|V_{\mathcal{B}}| \geq 300\theta^2 k^7 \Delta^2 \eta$, then there exists a vertex $x \in V_H$ satisfying $t^* \in v$ together with a 2-block-tree $\{C_1, C_2, \ldots, C_\eta\} \in \mathcal{T}_x^{\eta,\theta}$ such that $C_j \subseteq V_{\mathcal{B}}$ for all $j \in [\eta]$. By a union bound over all possible vertices x and all 2-block-trees in $\mathcal{T}_x^{\eta,\theta}$, we have

$$\mathbf{Pr}\left[|V_{\mathcal{B}}| \geq 3\theta^2 k \Delta^2 \eta\right] \leq \sum_{x \in V_H : t \in x} \sum_{\{C_1, \ldots, C_\eta\} \in \mathcal{T}_x^{\eta,\theta}} \mathbf{Pr}\left[\forall j \in [\eta], C_j \subseteq V_{\mathcal{B}}\right].$$

Fix a 2-block-tree $\{C_1, \ldots, C_\eta\} \in \mathcal{T}_x^{\eta,\theta}$. By definition, for any j and ℓ that $j \neq \ell$, we have $\text{dist}_{G_H^{\text{self}}}(C_j, C_\ell) \geq 2$, and thus for any $x_j \in C_j$ and $x_\ell \in C_\ell$, it holds that $x_j \cap x_\ell = \varnothing$. For any $j \in [\eta]$, and any two $w_1, w_2 \in C_j$, it holds that $|w_1 \cap w_2| \leq 1$ by the definition of $\mathcal{T}_x^{\eta,\theta}$. This implies that C_j contains at least $\theta(k - \theta)$ distinct timestamps, and hence the 2-block-tree $\{C_1, \cdots, C_\eta\}$ contains at least $\theta(k-\theta)\eta$ distinct timestamps. Since the 2-block-tree is a subset of $V_{\mathcal{B}}$, by Lemma 5.14, every timestamp t in the 2-block-tree must take the corresponding random oracle output $r_t = \perp$. This gives

$$\mathbf{Pr}\left[\forall j \in [\eta], C_j \subseteq V_{\mathcal{B}}\right] \leq \left(\frac{1}{2}\right)^{\theta(k-\theta)\eta}.$$

Next, we count the number of possible 2-block-trees in $\mathcal{T}_x^{\eta,\theta}$, which can be upper bound by the number of all 2-block-trees $\{C_1, C_2, \ldots, C_\eta\}$ with block size θ and tree size η in G_H^{self} such that $x \in C_1$. By

Lemma 5.25 and Lemma 5.21, we have

$$\left| \mathcal{T}_x^{\eta,\theta}(U) \right| \leq (\theta \mathrm{e}^\theta (10k^3\Delta)^{\theta+1})^\eta.$$

Hence, we only need to prove that

$$(\theta \mathrm{e}^\theta (10k^3\Delta)^{\theta+1})^\eta \left(\frac{1}{2}\right)^{\theta(k-\theta)\eta} \leq \left(\frac{1}{2}\right)^\eta,$$

which is equivalent to $2^{k-\theta} \geq (2\theta)^{1/\theta} \mathrm{e}(10k^3\Delta)^{(1+\theta)/\theta}$. We derive this as follows. Observe the following inequalities

$$(23) \qquad\qquad \delta \geq \frac{2}{\frac{4}{\delta}+3} \geq \frac{2}{\left\lceil\frac{4}{\delta}+3\right\rceil} = \frac{2}{\theta-1};$$

$$(24) \qquad\qquad k \geq \frac{25(1+\delta)^2}{\delta^2} \geq \theta^2 \implies \frac{k-\theta}{k} \geq \frac{\theta-1}{\theta}.$$

Then we have

(By condition)
$$2^{k-\theta} = (2^k)^{\frac{k-\theta}{k}} \geq \left((100k^3\Delta)^{1+\delta}\right)^{\frac{k-\theta}{k}}$$

(By (23))
$$\geq \left(10^{1+\frac{2}{\theta-1}}(10k^3\Delta)^{1+\frac{2}{\theta-1}}\right)^{\frac{k-\theta}{k}}$$

$$\geq \left(5^{\frac{\theta}{\theta-1}}(10k^3\Delta)^{1+\frac{2}{\theta-1}}\right)^{\frac{k-\theta}{k}}$$

(By (24))
$$\geq 5(10k^3\Delta)^{\frac{1+\theta}{\theta}}$$

(By $\theta \geq 4$)
$$\geq (2\theta)^{1/\theta}\mathrm{e}(10k^3\Delta)^{\frac{1+\theta}{\theta}}.$$

This finishes the proof. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

6. Hypergraph colouring

In this section, we give FPTASes for the number of proper colourings in a hypergraph in the local lemma regime and prove Theorem 1.3 and Theorem 1.4. Given a k-uniform hypergraph $H = (V, \mathcal{E})$ with maximum degree Δ, we use $\Omega_H \subseteq [q]^V$ to denote the set of all proper q-colourings of H, and $Z = |\Omega_H|$ to denote its size. Our goal is to approximate Z.

We use the edge decomposition scheme in (6) to count the number of hypergraph colourings. Assume $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$. Let $\mathcal{E}_i = \{e_1, e_2, \ldots, e_i\}$, $H_i = (V, \mathcal{E}_i)$. Let μ_i denote the uniform distribution over all proper q-colourings of $H_i = (V, \mathcal{E}_i)$ for each $0 \leq i \leq m$, where $\mathcal{E}_i = \{e_1, e_2, \ldots, e_i\}$. According to (6), approximating Z_H boils down to approximating $\mathbf{E}_{z \sim \mu_{i-1}, e_i}[\phi_{e_i}(z)]$ for each $1 \leq i \leq m$, where $\phi_{e_i}(z) = 0$ if and only if all $v \in e_i$ take the same value in z. Hence,

$$(25) \qquad\qquad \mathbf{E}_{z \sim \mu_{i-1}, e_i}[\phi_{e_i}(z)] = \Pr_{X \sim \mu_{i-1}}[e_i \text{ is not monochromatic in } X].$$

Next, for each $0 \leq i \leq m$, it is straightforward to verify the following properties:

- H_i is a k-uniform hypergraph if H is a k-uniform hypergraph;
- the maximum degree of H_i is at most the maximum degree of H;
- H_i is a linear hypergraph if H is a linear hypergraph.

When the Lovász local lemma applies, any edge decomposition scheme is suitably lower bounded, as shown by the next lemma.

Lemma 6.1. *If $\Delta \geq 2$ and $q > (\mathrm{e}\Delta k)^{\frac{1}{k-1}}$, any edge decomposition scheme for $H = (V, \mathcal{E})$ is $\frac{1}{2}$-bounded.*

Proof. For any ordering of the hyperedges of H, fix a hypergraph $H_{i-1} = (V, \mathcal{E}_{i-1})$. Note that the maximum degree of H_{i-1} is at most Δ. Suppose each vertex takes a colour from $[q]$ uniformly and independently. For each hyperedge $e \in \mathcal{E}_{i-1}$, define B_e as the event that e is monochromatic. Let

$x(B_e) = \frac{1}{\Delta k}$ for all $e \in \mathcal{E}_{i-1}$. Then (1) in Lemma 2.2 is satisfied. Let A denote the event that all vertices in set e_i are monochromatic. By Lemma 2.3,

$$\Pr_{X \sim \mu_{i-1}} [e_i \text{ is monochromatic in } X] \leq \left(\frac{1}{q}\right)^{k-1} \left(1 - \frac{1}{\Delta k}\right)^{-(\Delta-1)k} \leq \frac{1}{\Delta k} \leq \frac{1}{2},$$

and thus the probability that e_i is not monochromatic is at least $\frac{1}{2}$. □

By Corollary 3.3, it suffices to give the following sampling algorithms for general hypergraphs and linear hypergraphs, respectively.

Lemma 6.2. *Let $\Delta \geq 2$, $k \geq 20$ and $q \geq 64\Delta^{\frac{3}{k-5}}$ be three integers and $0 < \gamma < 1$ be a real number. There exists an algorithm such that given any k-uniform hypergraph $H = (V, \mathcal{E})$ with the maximum degree at most Δ, any subset of vertices $S \subseteq V$ such that $|S| \leq k$, outputs a random $Y \in [q]^S$ such that $d_{\mathrm{TV}}(Y, \mu_S) \leq \gamma$, where μ is the uniform distribution over all proper q-colourings in H. The time cost of this algorithm is $O(\Delta^6 k^{11} (\log^2 \frac{1}{\gamma}) \cdot q^{8\Delta^3 k^6 \log \frac{1}{\gamma}})$. It draws at most $8\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil + 1$ random variables over a size-$(q+1)$ domain.*

Lemma 6.3. *Let $\delta > 0$ and $0 < \gamma < 1$ be two real numbers. Let $k \geq \frac{50(1+\delta)^2}{\delta^2}$, $\Delta \geq 2$ and q be integers satisfying $q \geq 50\Delta^{\frac{2+\delta}{k-3}}$. There exists an algorithm such that given any k-uniform linear hypergraph $H = (V, \mathcal{E})$ with the maximum degree at most Δ, any subset of vertices $S \subseteq V$ such that $|S| \leq k$, outputs a random $Y \in [q]^S$ such that $d_{\mathrm{TV}}(Y, \mu_S) \leq \gamma$, where μ is the uniform distribution over all proper q-colourings in H. The time cost of this algorithm is $O\left(\left(\frac{1+\delta}{\delta}\right)^4 \Delta^8 k^{21} (\log^2 \frac{1}{\gamma}) q^{6\cdot10^4 (\frac{1+\delta}{\delta})^2 \Delta^4 k^{11} \log \frac{1}{\gamma}}\right)$. It draws at most $6 \cdot 10^4 (\frac{1+\delta}{\delta})^2 \Delta^3 k^{10} \lceil \log \frac{1}{\gamma} \rceil + 1$ random variables over a size-$(q+1)$ domain.*

Remark 6.4. Unlike in Remark 5.4, algorithms in Lemma 6.2 and Lemma 6.3 are not direct truncations of perfect marginal samplers. In fact, we will apply CTTP to a projected distribution, rather than to the original uniform distribution over all proper hypergraph colourings. There are some extra steps after truncating CTTP, which will be explained in more details in Section 6.2.

Theorem 1.3 and Theorem 1.4 are direct consequences of Lemma 6.2 and Lemma 6.3, respectively.

Proof of Theorem 1.3 and Theorem 1.4. By Lemma 6.1, any edge decomposition scheme is $\frac{1}{2}$-bounded under the conditions of Theorem 1.3 and Theorem 1.4. Plugging $\gamma = \frac{\varepsilon}{20m}$ into Lemma 6.2, where $m \leq \Delta n$ is the number of hyperedges, we have an algorithm that draws at most $8\Delta^2 k^5 \lceil \log \frac{20m}{\varepsilon} \rceil + 1$ random variables over a size-$(q+1)$ domain with time cost $O(\Delta^6 k^{12} \log^2 \frac{20m}{\varepsilon} \cdot q^{8\Delta^3 k^6 \log \frac{20m}{\varepsilon}})$ for sampling from the conditional distribution within $\frac{\varepsilon}{20m}$ total variation distance. By Corollary 3.3, we have a deterministic approximate counting algorithm with running time

$$T = O\left(\Delta n \cdot \Delta^6 k^{11} \log^2 \frac{20\Delta n}{\varepsilon} \cdot q^{8\Delta^3 k^6 \log \frac{20\Delta n}{\varepsilon}} \cdot (q+1)^{9\Delta^2 k^5 \log \frac{20\Delta n}{\varepsilon}}\right) = \text{poly}(\Delta k) \left(\frac{\Delta n}{\varepsilon}\right)^{O(\Delta^3 k^6 \log q)}.$$

This proves Theorem 1.3. Theorem 1.4 is proved the same way but with Lemma 6.3 invoked. The running time of the deterministic approximate counting algorithm is

$$T = O\left(\Delta n \cdot \left(\left(\frac{1+\delta}{\delta}\right)^4 \Delta^8 k^{21} \log^2 \frac{20\Delta n}{\varepsilon}\right) \cdot q^{6\cdot10^4 (\frac{1+\delta}{\delta})^2 \Delta^4 k^{11} \log \frac{20\Delta n}{\varepsilon}} \cdot (q+1)^{7\cdot10^4 (\frac{1+\delta}{\delta})^2 \Delta^3 k^{10} \log \frac{20\Delta n}{\varepsilon}}\right)$$

$$= \text{poly}\left(\frac{(1+\delta)\Delta k}{\delta}\right) \left(\frac{\Delta n}{\varepsilon}\right)^{O\left(\frac{(1+\delta)^2 \Delta^4 k^{11} \log q}{\delta^2}\right)}.$$

This proves Theorem 1.4. □

6.1. Projection schemes and projected distribution.

Unlike in the case of hypergraph independent sets, the single-site Glauber dynamics for hypergraph colouring is not necessarily irreducible. We use the following "projection scheme" introduced in [FHY21] to resolve this issue.

Definition 6.5 (projection scheme for hypergraph colourings [FHY21]). Let $1 \leq s \leq q$ be an integer. A (balanced) *projection scheme* $h : [q] \to [s]$ satisfies for any $i \in [s]$, $\left|h^{-1}(i)\right| = \lfloor \frac{q}{s} \rfloor$ or $\left|h^{-1}(i)\right| = \lceil \frac{q}{s} \rceil$.

We extend h to colourings of V as well. For any $X \in [q]^V$, we use $Y = h(X)$ to denote the *projected image*:

$$\forall v \in V, \quad Y_v = h(X_v),$$

i.e., the colouring is projected independently for each vertex. Also for any subset of vertices $\Lambda \subseteq V$, we will use a similar notation $Y_\Lambda = h(X_\Lambda) = (h(X_v))_{v \in \Lambda}$.

Consider the projection scheme $h(\cdot)$ defined in Definition 6.5, where the integer parameter $1 \leq s \leq q$ will be fixed later. We can naturally associate it with the following projected distribution.

Definition 6.6 (projected distribution). The projected distribution $\psi : [s]^V \to [0, 1]$ is the distribution $Y = h(X)$ where $X \sim \mu$, where μ is the uniform distribution over all proper hypergraph q-colourings of $H = (V, \mathcal{E})$. Formally,

$$\forall \sigma \in [s]^V, \quad \psi(\sigma) = \sum_{\tau \in h^{-1}(\sigma)} \mu(\tau).$$

For any $\Lambda \subseteq V$ and $\sigma_\Lambda \in [s]^\Lambda$, we let ψ^{σ_Λ} denote the distribution over $[s]^V$ obtained from ψ conditional on σ_Λ. For any $S \subseteq V$, let $\psi_S^{\sigma_\Lambda}$ denote the marginal distribution on S projected from ψ^{σ_Λ}. If $S = \{v\}$ only contains a single vertex, we write $\psi_v^{\sigma_\Lambda}$ for simplicity.

We also slightly abuse the notation: for any $\Lambda \subseteq V$ and $\sigma_\Lambda \in [s]^\Lambda$, we let μ^{σ_Λ} denote the distribution of $X \sim \mu$ conditional on $h(X(v)) = \sigma_\Lambda(v)$ for each $v \in \Lambda$. Note that this conditional distribution differs from the usual conditional distribution μ^{σ_Λ} when $\sigma_\Lambda \in [q]^\Lambda$ is a partial colouring. We shall explain the meaning of σ_Λ when using μ^{σ_Λ}. Also, for any $S \subseteq V$, let $\mu_S^{\sigma_\Lambda}$ denote the marginal distribution on S projected from μ^{σ_Λ}. If $S = \{v\}$ only contains a single vertex, we write $\mu_v^{\sigma_\Lambda}$ for simplicity.

An important corollary from Lemma 2.3 is the "local uniformity" property for the projected distribution ψ in Definition 6.6, which states that for any $\Lambda \subseteq V$, any $v \in V \setminus \Lambda$, conditional on any partial configuration $\sigma \in [s]^\Lambda$, ψ_v^σ is close to the uniform distribution over $[s]$.

Lemma 6.7 (local uniformity). *Let $1 \leq s \leq q$. If $\lfloor q/s \rfloor^k \geq 4\mathrm{e}qs\Delta k$, then for any $v \in V$, any subset $\Lambda \subseteq V \setminus \{v\}$ and partial configuration $\sigma_\Lambda \in [s]^\Lambda$, it follows that*

$$\forall j \in [s], \quad \frac{\left|h^{-1}(j)\right|}{q}\left(1 - \frac{1}{4s}\right) \leq \psi_v^{\sigma_\Lambda}(j) \leq \frac{\left|h^{-1}(j)\right|}{q}\left(1 + \frac{1}{s}\right).$$

To prove Lemma 6.7, we need the following lemma for the general list hypergraph colouring problem. Let $H = (V, \mathcal{E})$ be a k-uniform hypergraph with maximum degree at most Δ. Let $\{Q_v\}_{v \in V}$ be a set of colour lists. We say $X \in \otimes_{v \in V} Q_v$ is a proper list colouring if no hyperedge in $\mathcal{E}$ is monochromatic with respect to X. Let μ denote the uniform distribution of all proper list hypergraph colourings of H.

Lemma 6.8 ([GLLZ19, Lemma 7] and [FGW22, Lemma 6]). *Suppose $q_0 \leq |Q_v| \leq q_1$ for any $v \in V$. For any $r \geq k \geq 2$, if $q_0^k \geq \mathrm{e}q_1 r\Delta$, then for any $v \in V$ and any colour $c \in Q_v$,*

$$\frac{1}{|Q_v|}\left(1 - \frac{1}{r}\right) \leq \mu_v(c) \leq \frac{1}{|Q_v|}\left(1 + \frac{4}{r}\right).$$

We are now ready to prove Lemma 6.7 using Lemma 6.8.

Proof. Note that $\psi_v^{\sigma_\Lambda}$ is the marginal distribution induced by a list hypergraph colouring instance with hypergraph H and colour lists satisfying

$$\forall u \in V, \quad Q_u = \begin{cases} h^{-1}(\sigma_\Lambda(u)) & u \in \Lambda \\ [q] & u \notin \Lambda \end{cases}.$$

Note that the size of the colour list Q_u of each vertex $u \in V$ satisfies

$$\lfloor q/s \rfloor \leq \left|h^{-1}(\sigma_\Lambda(u))\right| \leq |Q_v| \leq q$$

By setting $q_0 = \lfloor q/s \rfloor$, $q_1 = q$ and $r = 4sk$ in Lemma 6.8, we have

$$\forall j \in [s], \quad \frac{\left|h^{-1}(j)\right|}{q}\left(1 - \frac{1}{4s}\right) \leq \psi_v^{\sigma_\Lambda}(j) \leq \frac{\left|h^{-1}(j)\right|}{q}\left(1 + \frac{1}{s}\right). \qquad \square$$

6.2. **Marginal sampler for the projected distribution.** Here, we give the marginal sampler for the projected distribution of hypergraph colourings (Algorithm 7). We want to apply the truncated resolve algorithm, Algorithm 3. One requirement for the correctness of Algorithm 3 is the irreducibility of systematic scan Glauber dynamics, and we achieve this by running it on the projected distribution ψ, with a suitably chosen s. Furthermore, we will specify suitable marginal lower bounds for ψ, and specify the Boundary(t) subroutine.

Formally, we use the systematic scan Glauber dynamics $(Y_t)_{-T \leq t \leq 0}$ on the projected distribution ψ. By Lemma 6.7, for any $\Lambda \subseteq V$, any $\sigma_\Lambda \in [s]^\Lambda$ and any $v \in V \setminus \Lambda$, it holds that

$$\forall j \in [s], \quad \psi_v^{\sigma_\Lambda}(j) \geq \rho_j := \frac{\left|h^{-1}(j)\right|}{q}\left(1 - \frac{1}{4s}\right).$$

Hence, the distribution ψ has the $(\rho_1, \rho_2, \cdots, \rho_s)$-marginal lower bound. Note that this immediately shows that the systematic scan Glauber dynamics satisfies Condition 4.5 as $\rho_j > 0$ for all $j \in [s]$. Define the lower bound distribution for (projected) hypergraph colouring instances ψ^{LB} by

$$(26) \qquad \psi^{\mathrm{LB}}(\bot) = 1 - \sum_{1 \leq i \leq s} \rho_i = \frac{1}{4s} \qquad \text{and} \qquad \forall j \in [s], \quad \psi^{\mathrm{LB}}(j) = \rho_j.$$

For any $\Lambda \subseteq V$, any $\sigma_\Lambda \in [s]^\Lambda$ and any $v \in V \setminus \Lambda$, define the padding distribution $\psi_v^{\mathrm{pad}, \sigma_\Lambda}$ over $[s]$ by

$$\forall j \in [s], \quad \psi_v^{\mathrm{pad}, \sigma_\Lambda}(j) = 4s(\psi_v^{\sigma_\Lambda}(j) - \rho_j).$$

In order to apply Algorithm 3, we still need to specify the Boundary(t) subroutine for ψ. Note that the projected distribution ψ is no longer a Gibbs distribution, but we can still get the conditional independence property by constructing σ_Λ via a breadth-first search (BFS). Given a projected configuration $\sigma_\Lambda \in [s]^\Lambda$ on a subset of vertices $\Lambda \subseteq V$, we say a hyperedge $e \in \mathcal{E}$ is *satisfied* by σ_Λ if there are $u, v \in e \cap \Lambda$ such that $\sigma_\Lambda(u) \neq \sigma_\Lambda(v)$. If e is satisfied by σ_Λ, for all $X \in [q]^V$ such that $\sigma_\Lambda = h(X_\Lambda)$, the hyperedge e cannot be monochromatic with respect to X. Let $H^{\sigma_\Lambda} = (V, \mathcal{E}')$ be the hypergraph obtained from H after removing all hyperedges in H satisfied by σ_Λ. Our subroutine Boundary(t) uses BFS to find the connected component V' in $H^{Y_{t-1}(V \setminus \{v_{i(t)}\})}$ containing the target variable $v = v_{i(t)}$. Once the component is found, it also finds $\sigma_\Lambda = Y_{t-1}(\Lambda)$ on some $\Lambda \supset V'$ such that V' is disconnected from the rest in H^{σ_Λ}. We show in Lemma 6.9 that this σ_Λ satisfies Condition 4.3.

The pseudocode for Boundary(t) is given in Algorithm 6. Recall that it requires two oracles $\mathcal{B}$ and C in Section 4.1.

Lemma 6.9. Boundary(t) *in Algorithm 6 satisfies Condition 4.3.*

Proof. The assumptions on the oracles $\mathcal{B}$ and C ensure that

$$(27) \qquad \forall u \neq v, \qquad \mathcal{B}(\mathrm{pred}_u(t)) \neq \bot \implies Y_{t-1}(u) = \mathcal{B}(\mathrm{pred}_u(t)),$$

and $C(u) = Y_{t-1}(u)$. We then verify the termination and conditional independence property in Condition 4.3. Note that in the while loop in Lines 2-11, if the condition in Line 4 is satisfied, then e is added into V' in Line 7; otherwise e is satisfied by σ_Λ after the loop by Line 11. This shows each $e \in \mathcal{E}$ can be chosen in Line 3 at most once, and therefore the while loop eventually terminates. After the while loop in Lines 2-11 terminates, the following holds:

    (1) $v \in V'$;
    (2) $V' \subseteq \Lambda \cup \{v\}$;
    (3) for all $e \in \mathcal{E}$ s.t. $e \cap V' \neq \varnothing$ and $e \cap (V \setminus V') \neq \varnothing$, e is satisfied by σ_Λ.

Property (1) holds because v is added to V' in the initialisation, and V' never removes vertices. Property (2) holds because if vertices in some $e \in \mathcal{E}$ are added to V' in Line 7, then all vertices in $e \setminus \{v\}$ are added into Λ in Line 6. As the while loop terminates, the condition in Line 2 no long holds, which is exactly Property (3).

Algorithm 6: Boundary(t) for (projected) hypergraph colourings

---

Input: a hypergraph $H = (V, \mathcal{E})$, a set of colours $[q]$, a projection scheme $h : [q] \to [s]$ that defines the projected distribution ψ, and an integer $t \leq 0$;

Output: a partial configuration $\sigma_\Lambda \in [s]^\Lambda$ over some $\Lambda \subseteq V \setminus \{v_{i(t)}\}$;

1   $v \leftarrow v_{i(t)}, \Lambda \leftarrow \varnothing, \sigma_\Lambda \leftarrow \varnothing, V' \leftarrow \{v\}$;

2   **while** $\exists e \in \mathcal{E}$ *s.t.* $e \cap V' \neq \varnothing$, $e \cap (V \setminus V') \neq \varnothing$ *and* e *is not satisfied by* σ_Λ **do**

3   choose such e with the lowest index;

4   **if** $\exists j \in [s]$ *s.t.* $\forall u \in e \setminus \{v\}, \mathcal{B}(\mathrm{pred}_u(t)) \in \{\bot, j\}$ **then**

5   **forall** $u \in e \setminus \{v\}$ **do**

6              $\Lambda \leftarrow \Lambda \cup \{u\}$ and $\sigma_\Lambda(u) \leftarrow C(u)$;

7          $V' \leftarrow V' \cup e$;

8   **else**

9   **forall** $u \in e \setminus \{v\}$ **do**

10   **if** $\mathcal{B}(\mathrm{pred}_u(t)) \neq \bot$ **then**

11                $\Lambda \leftarrow \Lambda \cup \{u\}$ and $\sigma_\Lambda(u) \leftarrow \mathcal{B}(\mathrm{pred}_u(t))$;

12 **return** σ_Λ;

Since σ_Λ is a restriction of $Y_{t-1}(V \setminus \{v\})$ on Λ, Property (1) and Property (3) imply that conditioned on either σ_Λ or $Y_{t-1}(V \setminus \{v\})$, the marginal distribution for v is unchanged if we remove all hyperedges crossing both V' and $V \setminus V'$. Thus, under both conditioning, the marginal distribution for v is the same as its marginal distribution on $H[V']$ given σ_Λ and $Y_{t-1}(V \setminus \{v\})$ restricted to $V' \setminus \{v\}$, respectively. By Property (2), the latter two conditionings restricted to $V' \setminus \{v\}$ are the same. Thus, $\mu_v^{\sigma_\Lambda} = \mu_v^{Y_{t-1}(V \setminus \{v\})}$. $\qquad \square$

Plug Boundary(t) of Algorithm 6 into ApproxResolve of Algorithm 3 and set the threshold K to be $4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil$ This gives Algorithm 7 for approximately sampling from the marginal distributions on one vertex projected from ψ. Recall that as in Algorithm 1, when plugging Boundary(t), we need to replace the oracles $\mathcal{B}$ and C with suitable calls to LB-Sample and recursive calls, respectively.

Algorithm 7 takes a hypergraph $H = (V, \mathcal{E})$, a colour set $[q]$, a projection scheme $h : [q] \to [s]$, an integer $t < 0$, and a parameter γ as the input. To avoid notation cluttering, we drop $(H, [q], h)$ from the input, since these are the same throughout all recursive calls. We denote the algorithm by ApproxMarginColouring($t, \gamma; M, R$), where M and R are two global data structures maintained by the algorithm. When approximately sampling from ψ_v, find an integer t such that $-n < t \leq 0$ and $v = v_{i(t)}$, and then evoke ApproxMarginColouring($t, \gamma; M_0, R_0$), where $M_0 = \bot^{\mathbb{Z}}$ and $R_0 = \varnothing$. Recall that the subroutine LB-Sample in the algorithm is given in Algorithm 2 with the lower bound distribution μ^{LB} given by (26).

The correctness of Algorithm 7 is justified by Theorem 4.9 and Lemma 6.9. In Line 16 of Algorithm 7, we sample from the padding distribution $\psi_v^{\mathrm{pad}, \sigma_\Lambda}$. To implement it, we enumerate all colourings $X \in \otimes_{u \in \Lambda} h^{-1}(\sigma_u)$ on subset Λ to compute $\mu_v^{\sigma_\Lambda}$ and then compute $\psi_v^{\mathrm{pad}, \sigma_\Lambda}$ by

$$(28) \qquad \forall j \in [s], \quad \psi_v^{\mathrm{pad}, \sigma_\Lambda}(j) = 4s \left(\left(\sum_{c \in h^{-1}(j)} \mu_v^{\sigma_\Lambda}(c) \right) - \rho_j \right).$$

Algorithm 7 is not the algorithm stated in Lemma 6.2 and Lemma 6.3 as it only samples from the projected value on a single vertex. In the upcoming Section 6.3, we show how to use Algorithm 7 as a subroutine to build the algorithm for Lemma 6.2 and Lemma 6.3.

6.3. **Estimating the marginal distribution on a subset.** We now describe an algorithm such that given a hypergraph $H = (V, \mathcal{E})$ and a subset $S \subseteq V$, it approximately samples from μ_S, where μ_S is the marginal distribution on S induced by the uniform distribution μ of all proper q-colourings of H.

Algorithm 7: ApproxMarginColouring$(t, \gamma; M, R)$

Input: a hypergraph $H = (V, \mathcal{E})$, a set of colours $[q]$, a projection scheme $h : [q] \to [s]$ that
defines the projected distribution ψ, an integer $t \leq 0$ and a real number $0 < \gamma < 1$;

Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\bot\}$ and a set R;

Output: a random value in $[s]$;

1 try :

2   **if** $M(t) \neq \bot$ **then return** $M(t)$;

3   **if** LB-Sample$(R, t) \neq \bot$ **then** $M(t) \leftarrow$ LB-Sample(R, t) **and return** $M(t)$;

4 $v \leftarrow v_{i(t)}, \Lambda \leftarrow \varnothing, \sigma_\Lambda \leftarrow \varnothing, V' \leftarrow \{v\}$;

5   **while** $\exists e \in \mathcal{E}$ *s.t.* $e \cap V' \neq \varnothing$, $e \cap (V \setminus V') \neq \varnothing$ *and* e *is not satisfied by* σ_Λ **do**

6   choose such e with the lowest index;

7   **if** $\exists j \in [s]$ *s.t.* $\forall u \in e \setminus \{v\}$, LB-Sample$(\text{pred}_u(t); R) \in \{\bot, j\}$ **then**

8   **forall** $u \in e \setminus \{v\}$ **do**

9 $\Lambda \leftarrow \Lambda \cup \{u\}$ and $\sigma_\Lambda(u) \leftarrow$ ApproxMarginColouring$(\text{pred}_u(t), \gamma; M, R)$;

10 $V' \leftarrow V' \cup e$;

11   **else**

12 $U \leftarrow \{u \in e \setminus \{v\} \mid$ LB-Sample$(\text{pred}_u(t); R) \neq \bot\}$;

13   **forall** $u \in U$ **do**

14 $\Lambda \leftarrow \Lambda \cup \{u\}$ and $\sigma_\Lambda(u) \leftarrow$ LB-Sample$(\text{pred}_u(t); R)$;

15   enumerate all $X \in \otimes_{u \in \Lambda} h^{-1}(\sigma_u)$ to compute $\mu_v^{\sigma_\Lambda}$ and then compute $\psi_v^{\text{pad}, \sigma_\Lambda}$ using (28);

16   sample $c \sim \psi_v^{\text{pad}, \sigma_\Lambda}$;

17 $M(t) \leftarrow c$ **and return** $M(t)$;

18 catch $|R| \geq 4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil$ **:**

19   **return** 1;

We then use this algorithm to prove Lemma 6.2 and Lemma 6.3. Our algorithm builds on Algorithm 7, which draws approximate samples from the projected marginal distribution ψ_v. Suppose we want to sample from the marginal distribution μ_S. One straightforward approach is that

- sample $Y \sim \psi$;
- sample $X \sim \mu$ conditioning on $h(X) = Y$ and output X_S.

However, the approach above uses too much randomness. Our sampling algorithm essentially follows it but without sampling the full configurations of X and Y. Note that the process above specifies a joint distribution $(Y, X) \sim \pi$ such that $Y \sim \psi$ and $X \sim \mu^Y$. Consider the following computational problem. Suppose there is a random sample $Y \sim \psi$ together with an oracle $\mathcal{Y}$ such that given any $v \in V$, $\mathcal{Y}$ returns the value of Y_v. The algorithm needs to draw a random sample X_S using as few oracle queries as possible. We use the idea in Algorithm 6. Start from S and use BFS to find a subset $\Lambda \supseteq S$ together with a partial configuration Y_Λ such that X_S is independent from $Y_{V \setminus \Lambda}$ in π conditional on Y_Λ. In other words, the partial configuration Y_Λ gives enough information to compute X_S. The algorithm for sampling $X_S \sim \mu_S$ is presented in Algorithm 8, where the oracle $\mathcal{Y}$ mentioned above is implemented by ApproxMarginColouring (Algorithm 7).

ApproxMarginColouring-Set (Algorithm 8) together with subroutines ApproxMarginColouring and LB-Sample maintain two global data structures M and R, which are initialised as $M = M_0 = \bot^{\mathbb{Z}}$ and $R = R_0 = \varnothing$ in Line 1 of ApproxMarginColouring-Set. Once $|R| \geq 4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil$, ApproxMarginColouring-Set stops immediately and outputs the all-one configuration on S.

Since we truncate the algorithm when $|R| \geq 4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil$, we have the following lemma that bounds the running time of Algorithm 8.

Lemma 6.10. *The running time of Algorithm 8 is* $O(\Delta^6 k^{11} \log^2 \frac{1}{\gamma} \cdot q^{8\Delta^3 k^6 \log \frac{1}{\gamma}})$.

Algorithm 8: ApproxMarginColouring-Set(S, γ)

Input: a hypergraph $H = (V, \mathcal{E})$, a set of colours $[q]$, a projection scheme $h : [q] \to [s]$ that defines the projected distribution ψ, a subset of $S \subseteq V$ and a real number $0 < \gamma < 1$;
Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\bot\}$ and a set R;
Output: a random assignment in $\tau \in [q]^S$

1   $M \leftarrow \bot^{\mathbb{Z}}$ and $R \leftarrow \varnothing$;
2   **try :**
3     $\sigma \leftarrow \varnothing, V' \leftarrow S$;
4   **forall** $u \in S$ **do**
5       $\sigma(u) \leftarrow$ ApproxMarginColouring$(\text{pred}_u(0), \gamma; M, R)$;
6   **while** $\exists e \in \mathcal{E}$ *s.t.* $e \cap V' \neq \varnothing$, $e \cap (V \setminus V') \neq \varnothing$ *and e is not satisfied by σ* **do**
7   choose such e with the lowest index;
8   **if** $\exists j \in [s]$ *s.t.* $\forall u \in e$, LB-Sample$(\text{pred}_u(0); R) \in \{\bot, j\}$ **then**
9   **forall** $u \in e$ **do**
10           $\sigma(u) \leftarrow$ ApproxMarginColouring$(\text{pred}_u(0), \gamma; M, R)$;
11         $V' \leftarrow V' \cup e$;
12   **else**
13         $U \leftarrow \{u \in e \mid$ LB-Sample$(\text{pred}_u(0); R) \neq \bot\}$;
14   **forall** $u \in U$ **do**
15           $\sigma(u) \leftarrow$ LB-Sample$(\text{pred}_u(0); R)$;
16   enumerate all colourings $X \in \otimes_{u \in V'} h^{-1}(\sigma_u)$ on V' to compute the marginal distribution of S on the sub-hypergraph $H[V']$, which is equivalent to μ_S^{σ};
17   Sample $\tau \sim \mu_S^{\sigma}$;
18   **return** τ;
19   **catch** $|R| \geq 4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil$ **:**
20   **return** 1^S;

To prove Lemma 6.10, we need the following lemma.

Lemma 6.11. *For any hypergraph k-uniform $H = (V, \mathcal{E})$ with the maximum degree at most Δ, any integer $t < 0$ and any $0 < \gamma \leq \frac{1}{2}$, any $M : \mathbb{Z} \to [q] \cup \{\bot\}$ and any set R, let Λ be either*

- *the set of assigned variables in σ in Line 15 of the execution ApproxMarginColouring$(t, \gamma; M, R)$;*
- *the set of assigned variables in σ in Line 16 of the execution of ApproxMarginColouring-Set(S, γ),*

it holds that

$$|\Lambda| \leq k\Delta \cdot |V'| \leq 4\Delta^3 k^6 \left\lceil \log \frac{1}{\gamma} \right\rceil.$$

Proof. We only show the case for Algorithm 7 and the case for Algorithm 8 holds through a similar argument. The size of the set V' increases from during the while loop. By the condition in Line 5, we have for all $u \in \Lambda$, there exists $e \in \mathcal{E}$ and $w \in V'$ such that $u, w \in e$. This shows that

$$|\Lambda| \leq k\Delta \cdot |V'|.$$

Moreover, note that by Line 3 and Line 7 we have $|R| \geq |V'|$. Therefore by the truncation condition at Line 18 we have

$$|V'| \leq 4\Delta^2 k^5 \left\lceil \log \frac{1}{\gamma} \right\rceil.$$

Combining the two inequalities proves the lemma.   □

Proof of Lemma 6.10. Consider the whole process of ApproxMarginColouring-Set(S, γ). Let A be the set of all t_0 such that ApproxMarginColouring$(t_0, \gamma; M, R)$ (for some M and R) is executed at least

once. For any $t_0 \in A$, one of the following two events must happen (1) the whole algorithm terminates in Line 18; or (2) $(t_0, r_{t_0}) \in R_F$, where R_F is the final R when whole algorithm terminates. Moreover, there is at most one $t^* \in A$ making the first event happen. Hence $|A| \le |R_F| + 1 = O\left(\Delta^2 k^5 \log \frac{1}{\gamma}\right)$. For any $t_0 \in A$, once ApproxMarginColouring$(t_0, \gamma; M, R)$ returns, $M(t_0) \ne \bot$. Any further calls to ApproxMarginColouring$(t_0, \gamma; M, R)$ would not execute beyond Line 2 of Algorithm 7 and would have cost $O(1)$ time. Consider the recursion tree of the whole execution. All subsequent recursive calls are leaf nodes. We attribute the time cost of the subsequent recursive calls to its parent, and this way the total time cost becomes the total cost of all first calls. Assuming the cost of each recursive call is $O(1)$, let T^* be the upper bound of the running time of both

- ApproxMarginColouring$(t, \gamma; M, R)$ of Algorithm 7, and
- ApproxMarginColouring-Set(S, γ) of Algorithm 8.

Then the total running time of can be bounded by $(|A| + 1)T^*$. We then analyze the time cost of the first item for Algorithm 7.

Note that the cost of the while loop in Lines 5-14 of Algorithm 7 is at most a constant multiple of the number of executions of Line 9 and Line 14. If the condition in Line 7 is satisfied, then e is added into V' in Line 10; otherwise e is satisfied by σ after the loop by Line 14. This shows that each $e \in \mathcal{E}$ can be chosen in Line 6 at most once. Each time Line 9 or Line 14 executes, some u is added to Λ due to an hyperedge $e \ni u$. Thus it happens at most Δ times for each $u \in \Lambda$. By Lemma 6.11, $|\Lambda| \le O\left(\Delta^3 k^6 \log \frac{1}{\gamma}\right)$. Thus the total cost of the while loop is at most $O\left(\Delta^4 k^6 \log \frac{1}{\gamma}\right)$. Moreover, in Line 15, we enumerate $q^{|\Lambda|+1}$ possible configurations, each of which takes $O(|\Lambda| \Delta)$ time to check. The total time cost of Line 15 is

$$O(|\Lambda| \Delta q^{|\Lambda|+1}) = O\left(\Delta^4 k^6 \log \frac{1}{\gamma} \cdot q^{8\Delta^3 k^6 \log \frac{1}{\gamma}}\right).$$

The same time cost bound holds for Algorithm 8 by the same argument. Therefore it suffices to take

$$T^* = O\left(\Delta^4 k^6 \log \frac{1}{\gamma} \cdot q^{8\Delta^3 k^6 \log \frac{1}{\gamma}}\right).$$

Since $|A| = O(\Delta^2 k^5 \log \frac{1}{\gamma})$, the total running time of ApproxMarginColouring-Set(S, γ) is at most $O(\Delta^6 k^{11} \log^2 \frac{1}{\gamma} \cdot q^{8\Delta^3 k^6 \log \frac{1}{\gamma}})$. The lemma follows. $\qquad\square$

The next lemma shows the correctness of Algorithm 8. Let $\mathcal{E}_{\text{trun}} = \mathcal{E}_{\text{trun}}(4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil)$ be the event that the truncation in Line 20 of Algorithm 8 occurs. Similar to Theorem 4.9, we bound the total variation distance between the output of Algorithm 7 and the target distribution μ_S by $\mathbf{Pr}[\mathcal{E}_{\text{trun}}]$.

Lemma 6.12. *Let $H = (V, \mathcal{E})$ be a k-uniform hypergraph with the maximum degree at most Δ, $[q]$ be a set of colours, and $h : [q] \to [s]$ be a projection scheme that defines the projected distribution ψ. Let $S \subseteq V$ be a subset of vertices and $\gamma > 0$ be a real number. Let X_S be the output of ApproxMarginColouring-Set(S, γ). If $\lfloor q/s \rfloor^k \ge 4eqs\Delta k$, it holds that $d_{\text{TV}}(X_S, \mu_S) \le \mathbf{Pr}[\mathcal{E}_{\text{trun}}]$, where $\mathcal{E}_{\text{trun}} = \mathcal{E}_{\text{trun}}(4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil)$ and μ is the uniform distribution over all proper q-colourings of H.*

The proof of Lemma 6.12 is similar to the proof of Theorem 4.9. The main difference is that the equivalent to Algorithm 3, ApproxMarginColouring, is called multiple times in Algorithm 8. Intuitively, we use ApproxMarginColouring to access a full configuration $Y \in [s]^V$, where Y approximately follows the projected distribution ψ. We may query the values of Y_u for different vertices u. The answers returned by ApproxMarginColouring for different u should be consistent with each other. Since this proof is somewhat repetitive to that of Theorem 4.9, we delay the proof of Lemma 6.12 to Section 6.6.

We also have the following lemma that bounds the probability that the size of R becomes too large.

Lemma 6.13. *Let $H = (V, \mathcal{E})$ be a k-uniform hypergraph with the maximum degree at most Δ, $[q]$ be a set of colours, and $h : [q] \to [s]$ be a projection scheme that defines the projected distribution ψ. Let $S \subseteq V$ be a subset of vertices and $\gamma > 0$ be a real number. If $k \ge 20$, $q \ge 4s$, $\lfloor q/s \rfloor^k \ge 4eqs\Delta k$, $s \ge 6\Delta^{\frac{2}{k-2}}$*

and $|S| \leq k$, *then upon the termination of* ApproxMarginColouring-Set(S, γ), *it holds that*

$$(29) \qquad\qquad \mathbf{Pr}\left[|R| \geq 4\Delta^2 k^5 \cdot \eta\right] \leq \cdot 2^{-\eta}$$

where $\eta = \lceil \log \frac{1}{\gamma} \rceil$. *In particular, this shows that*

$$\mathbf{Pr}\left[\mathcal{E}_{\text{trun}}\right] \leq \gamma.$$

Lemma 6.13 is proved in Section 6.4. We are now ready to prove Lemma 6.2.

Proof of Lemma 6.2. Let $s = \left\lfloor q^{\frac{2}{3}} \right\rfloor$. As $q \geq 64\Delta^{\frac{3}{k-5}}$, we have

$$q \geq 4s, \qquad\qquad\qquad s \geq 16\Delta^{\frac{2}{k-5}} - 1 \geq 6\Delta^{\frac{2}{k-2}},$$

and by $k \geq 20$,

$$\left\lfloor \frac{q}{s} \right\rfloor \geq 4\Delta^{\frac{1}{k-5}} - 1 \geq 2(k\Delta)^{\frac{1}{k-5}}.$$

Therefore

$$\left\lfloor \frac{q}{s} \right\rfloor^k \geq 2^{k-5}(k\Delta)\left\lfloor \frac{q}{s} \right\rfloor^5 \geq 2^{11} \cdot 4ek\Delta \cdot \left\lfloor \frac{q}{s} \right\rfloor^5 \geq 4eqs\Delta k,$$

where the second inequality is by $k \geq 20$, and the last inequality is by $8\left\lfloor \frac{q}{s} \right\rfloor^2 \geq s$ and $8\left\lfloor \frac{q}{s} \right\rfloor^3 \geq q$ from $s = \left\lfloor q^{\frac{2}{3}} \right\rfloor$. Therefore the conditions in Lemma 6.13 are met. We use ApproxMarginColouring-Set(S, γ) to sample from the distribution μ_S. By Lemma 6.10, the running time is $O(\Delta^6 k^{11} \log^2 \frac{1}{\gamma} \cdot q^{8\Delta^3 k^6 \log \frac{1}{\gamma}})$. It is straightforward to verify that Algorithm 8 uses at most $8\Delta^2 k^5 \left\lceil \log \frac{1}{\gamma} \right\rceil + 1$ random variables with domain size at most $q + 1$. By combining Lemma 6.12 and Lemma 6.13, the output is a random sample that is γ-close to μ_v in the total variation distance. This proves the lemma. $\qquad\square$

6.4. Truncation error of hypergraph colouring.
We now prove Lemma 6.13.

We need some notations from Section 5. Recall TS(e, t) in (17) for each $e \in \mathcal{E}$ and $t \leq 0$. Also recall the witness graph $G_H = (V_H, E_H)$ in Definition 5.10. Note that the maximum degree of G_H is bounded by $2\Delta k^2 - 2$ by Lemma 5.11.

Fix a k-uniform hypergraph $H = (V, \mathcal{E})$ with the maximum degree at most Δ together with a set $[q]$ of colours, a projection scheme with parameter s, a subset of vertices $S \subseteq V$, and a real number $\gamma > 0$ satisfying the condition in Lemma 6.13. We analyse the algorithm ApproxMarginColouring-Set(S, γ).

As we want to resolve the marginal distribution for a set S of vertices, we modify the definition of G_H to obtain another graph $G_H^S = (V_H^S, E_H^S)$ by adding a new vertex representing the last update times of all $v \in S$ before 0 and corresponding edges.

Definition 6.14 (modified witness graph)**.** For a hypergraph $H = (V, \mathcal{E})$ and a subset of vertices $S \subseteq V$, if $S \in \mathcal{E}$, define the modified witness graph $G_H^S = G_H$; otherwise, let TS$(S, 0) = \{\text{pred}_u(0) \mid u \in S\}$, and define the modified witness graph $G_H^S = (V_H^S, E_H^S)$ with respect to S as follows:

- $V_H^S = V_H \cup \{\text{TS}(S, 0)\}$,
- $E_H^S = E_H \cup \{(x, \text{TS}(S, 0)) \mid x \in V_H \wedge x \cap \text{TS}(S, 0) \neq \varnothing\}$,

where $G_H = (V_H, E_H)$ is defined in Definition 5.10.

Note that in the modified graph G_H^S, there exists a vertex TS$(S, 0)$ for the subset S. Let $e_H(\text{TS}(S, 0)) = S$. We can then define $N_{\text{self}}(\text{TS}(S, 0))$, $N_{\text{out}}(\text{TS}(S, 0))$ and $N(\text{TS}(S, 0))$ as in (18). That is, $N_{\text{self}}(\text{TS}(S, 0)) = \varnothing$ and $N_{\text{out}}(\text{TS}(S, 0)) = N(\text{TS}(S, 0)) = \{w \in V_H^S \mid (w, \text{TS}(S, 0)) \in E_H^S\}$; And for all $w \in N(\text{TS}(S, 0))$, it holds that TS$(S, 0) \in N_{\text{out}}(w)$.

The next result follows from the proof of Lemma 5.11.

Corollary 6.15. *If* $|S| \leq k$, *then the maximum degree of* G_H^S *is bounded by* $2\Delta k^2 - 1$.

Proof. Note that compared to G_H, the degree of all vertices $x \in V_H = V_H^S \setminus \{\text{TS}(S, 0)\}$ increases by at most one and is therefore bounded by $2\Delta k^2 - 1$ by Lemma 5.11.

It then remains to bound the degree of TS$(S, 0)$. We have two cases:

- If $S = e$ for some $e \in \mathcal{E}$, then following the proof of Lemma 5.11, $|N_{\text{out}}(x)| = |N_{\text{out}}(\mathsf{TS}(e, 0))| \leq (2k-1)(\Delta-1)k$ and $|N_{\text{self}}(x)| \leq 2k-1$. This together shows $|N(x)| \leq 2k^2(\Delta-1) + 2k \leq 2k^2\Delta - 1$.
- Otherwise, $|N_{\text{self}}(x)| = 0$. Following the proof of Lemma 5.11, we have $|N_{\text{out}}(x)| \leq k\Delta \cdot (2k-1) \leq 2k^2\Delta - 1$. This shows $|N(x)| \leq 2k^2\Delta - 1$.

Combining the above proves the lemma. $\qquad\square$

Fix an integer $t_0 \leq 0$. ApproxMarginColouring$(t_0, \gamma; M, R)$ can only be evoked by:

(1) ApproxMarginColouring-Set(S, γ) in Line 5 when it processes S, and in this case, it holds that $t_0 = \text{pred}_u(0)$ for some $u \in S$; or

(2) ApproxMarginColouring-Set(S, γ) in Line 10 when it processes $e \in \mathcal{E}$, and in this case, it holds that $t_0 = \text{pred}_u(0)$ for some $u \in e$; or

(3) ApproxMarginColouring$(t_1, \gamma; M, R)$ in Line 9 when it processes $e \in \mathcal{E}$, and in this case, it holds that $t_0 < t_1 \leq 0$ and $t_0 = \text{pred}_u(t_1)$ for some $u \in e$.

In the first case, we say an instance of ApproxMarginColouring$(t_0, \gamma; M, R)$ is triggered by $\mathsf{TS}(S, 0) \in V_H^S$. In the second case, we say such an instance is triggered by $\mathsf{TS}(e, 0) \in V_H^S$. In the last case, we say such an instance is triggered by $\mathsf{TS}(e, t_1) \in V_H^S$. Conversely, for any $x \in V_H^S$ in the witness graph, we say x triggers some instance of ApproxMarginColouring if any of the above happens with the trigger being x during the whole process.

Define the following random subset of vertices in G_H^S:

(30) $$V_{\mathcal{B}}^{\text{col}} = \{x \in V_H^S \mid x \text{ triggers some instance of ApproxMarginColouring}\},$$

where the randomness of $V_{\mathcal{B}}^{\text{col}}$ comes from the random variables $\{r_t\}_{t \leq 0}$ and the samples drawn from padding distribution in Line 16 of ApproxMarginColouring. Since Line 5 always happens, it holds that

$$\mathsf{TS}(S, 0) \in V_{\mathcal{B}}^{\text{col}}.$$

After Algorithm 8 terminates, it generates sets $V_{\mathcal{B}}^{\text{col}}$ and R. Similar to Lemma 5.12, the following lemma shows that the size of R can be upper bounded in terms of $|V_{\mathcal{B}}^{\text{col}}|$.

Lemma 6.16. *If $|S| \leq k$, then $|R| \leq 2\Delta k^3 |V_{\mathcal{B}}^{\text{col}}|$.*

Proof. We first arbitrarily fix values of all random variables $(r_t)_{t \leq 0}$ and the outcomes of samples drawn from the padding distribution in Line 16 in ApproxMarginColouring. Then, the whole algorithm is deterministic. We show that the lemma always holds.

We claim that for each $(t, r_t) \in R$, there exist $x, y \in V_H^S$ (possibly $x = y$) such that $t \in x$, $y \in V_{\mathcal{B}}^{\text{col}}$ and $x \cap y \neq \varnothing$. Namely either $x = y$ or $(x, y) \in E_H^S$. For any $(t, r_t) \in R$, we define a map $f(t, r_t) = x \in V_{\mathcal{B}}^{\text{col}}$, where x is chosen arbitrarily from those satisfying the claim, such as the lexicographically first. Since $|S| \leq k$, by Corollary 6.15, the maximum degree of G_H^S is at most $2\Delta k^2 - 1$. As $|x| \leq k$ there are at most $2\Delta k^3$ different $(t, r_t) \in R$ such that $f(t, r_t)$ maps to the same y. Hence,

$$|R| \leq |V_{\mathcal{B}}^{\text{col}}| 2\Delta k^3.$$

To prove the claim, we first show the following result for ApproxMarginColouring-Set (Algorithm 8) and ApproxMarginColouring (Algorithm 7):

(31) $$\forall u \in V', \exists x \in V_{\mathcal{B}}^{\text{col}} \text{ s.t. } \text{pred}_u(t_0) \in x,$$

where $t_0 = 0$ for ApproxMarginColouring-Set(S, γ) and $t_0 = t$ for ApproxMarginColouring$(t, \gamma; M, R)$.

- For ApproxMarginColouring-Set(S, γ), If $u \in S$, we can take $x = \mathsf{TS}(S, 0)$. If $u \in V' \setminus S$, then u must be added into V' when the algorithm is processing $e \in \mathcal{E}$ in Line 11. Then $\mathsf{TS}(e, 0) \in V_{\mathcal{B}}^{\text{col}}$ by Line 10 and $\text{pred}_u(0) \in \mathsf{TS}(e, 0)$.
- For ApproxMarginColouring$(t, \gamma; M, R)$. Note that ApproxMarginColouring$(t, \gamma; M, R)$ itself must be triggered by some $x^* \in V_{\mathcal{B}}^{\text{col}}$ and $t = \text{pred}_{v_{i(t)}}(t) \in x^*$. If $u = v_{i(t)}$, (31) holds for $x = x^*$. If $u \in V' \setminus \{v_{i(t)}\}$, then u must be added into V' when the algorithm is processing $e \in \mathcal{E}$ in Line 10. Then $\mathsf{TS}(e, t) \in V_{\mathcal{B}}^{\text{col}}$ by Line 9 and $\text{pred}_u(t) \in \mathsf{TS}(e, t)$.

Now we turn to show $x, y \in V_H^S$ in the claim exist. A pair (t, r_t) can be added into R only through the call of LB-Sample(R, t). There are 3 cases.

- It happens in Line 8 of Algorithm 8 when some edge $e \in \mathcal{E}$ is processed and for some $u \in e$, $t = \text{pred}_u(0)$. It holds that $e \cap V' \neq \emptyset$. Let $x = \text{TS}(e, 0)$ so that $t \in x$. Fix an arbitrary $w \in e \cap V'$. By (31), there is $y \in V_\mathcal{B}^{\text{col}}$ such that $\text{pred}_w(0) \in y$. Since $\text{pred}_w(0) \in x$, $x \cap y \neq \emptyset$.
- It happens in Line 3 of Algorithm 7 in ApproxMarginColouring$(t, \gamma; M, R)$. Note that $v_{i(t)} \in V'$ in ApproxMarginColouring$(t, \gamma; M, R)$. Hence, by (31), there is $y \in V_\mathcal{B}^{\text{col}}$ such that $t = \text{pred}_{v_{i(t)}}(t) \in y$. We can take $x = y$.
- It happens in Line 7 of Algorithm 7 when some edge $e \in \mathcal{E}$ is processed in an instance of ApproxMarginColouring$(t_0, \gamma; M, R)$ for some $t_0 \geq t$. It holds that $e \cap V' \neq \emptyset$. Let $x = \text{TS}(e, t_0)$ so that $t \in x$. Fix an arbitrary $w \in e \cap V'$. By (31), there is $y \in V_\mathcal{B}^{\text{col}}$ such that $\text{pred}_w(t_0) \in y$. Since $\text{pred}_w(t_0) \in x$, $x \cap y \neq \emptyset$.

Finally, we remark that we do not need to consider Line 13 / Line 15 in Algorithm 8 nor Line 12 / Line 14 in Algorithm 7, because if LB-Sample$(t; R)$ is evoked in these lines, it must have been evoked in Line 8 of Algorithm 8 or Line 7 of Algorithm 7 already. This finishes the proof. □

Lemma 6.16 shows that to upper bound the probability that $|R|$ is large, it suffices to upper bound the probability that $|V_\mathcal{B}^{\text{col}}|$ is large. The following lemma says that $V_\mathcal{B}^{\text{col}}$ is connected on G_H^S, and the proof is also similar to the proof of Lemma 5.13.

Lemma 6.17. $\text{TS}(S, 0) \subseteq V_\mathcal{B}^{\text{col}}$ and the subgraph in G_H^S induced by $V_\mathcal{B}^{\text{col}}$ is connected.

Proof. Fix arbitrary random choices of all random variables $(r_t)_{t \leq 0}$ and outcomes of samples drawn from the padding distribution in Line 16 in ApproxMarginColouring. Then the algorithm is deterministic. We prove that the lemma always holds. Note that $\text{TS}(S, 0) \in V_\mathcal{B}^{\text{col}}$ is trivial and we only need to prove the second property.

Consider the whole execution of ApproxMarginColouring-Set(S, γ). During the algorithm, we say $x \in V_H^S$ joins $V_\mathcal{B}^{\text{col}}$ once the condition in (30) is met by x. Note that all vertices join $V_\mathcal{B}^{\text{col}}$ in order during the execution of the algorithm. Let $V_\mathcal{B}^{\text{col}} = \{x^{(1)}, x^{(2)}, \ldots, x^{(\ell)}\}$, where $x^{(i)}$ is the i-th vertex that joins the set $V_\mathcal{B}^{\text{col}}$. Note that $x^{(1)} = \text{TS}(S, 0)$. This ℓ is finite because ApproxMarginColouring-Set(S, γ) terminates within a finite number of steps. We show that for any $i \in [\ell]$, there exists a path $P = (y_1, y_2, \ldots, y_m)$ in G_H^S such that

- $y_1 = \text{TS}(S, 0)$ and $y_m = x^{(i)}$;
- for all $1 \leq j \leq m$, $y_j \in V_\mathcal{B}^{\text{col}}$.

This result immediately proves the lemma.

We prove the result by induction on index i. The base case trivially holds by taking $P = (\text{TS}(S, 0))$.

Fix an integer $1 < k \leq \ell$. Suppose the result holds for all $x^{(i)}$ for $i < k$. We prove the result for $x^{(k)}$. We have the following cases:

(1) If $x^{(k)}$ joins $V_\mathcal{B}^{\text{col}}$ in ApproxMarginColouring-Set$(S, \gamma; M, R)$ when some edge $e \in \mathcal{E}$ is processed, then we have $x^{(k)} = \text{TS}(e, 0)$. By the condition in Line 6, there must exist some $w \in V' \cap e$. The set V' is initially set as S in Line 3 and is updated only in Line 11 after recursive calls. We have two further cases:
   (a) If $w \in S$, it suffices to take $P = (x^{(1)}, x^{(k)})$.
   (b) Otherwise w must have been added into V' in Line 11 after some instance is triggered in Line 10, which implies that there exists $e' \in \mathcal{E}$ and $i < k$ such that $w \in e'$ and $x^{(i)} = \text{TS}(e', 0)$. By the induction hypothesis, there is a path $P = (y_1, y_2, \ldots, y_{m'})$ for $x^{(i)}$ with $y_{m'} = x^{(i)}$. Note that $\text{pred}_w(0) \in x^{(k)} = \text{TS}(e, 0)$ and $\text{pred}_w(0) \in x^{(i)} = \text{TS}(e', 0)$. It suffices to take the path $P' = (P, x^{(k)})$.

(2) If $x^{(k)}$ joins $V_\mathcal{B}^{\text{col}}$ in ApproxMarginColouring$(t_0, \gamma; M, R)$ when some edge $e \in \mathcal{E}$ is processed, then we have $x^{(k)} = \text{TS}(e, t_0)$. By the condition in Line 5, there must exist some $w \in V' \cap e$. Here the set V' is initially set as $\{v_{i(t_0)}\}$ in Line 4 and is updated only in Line 10 after some recursive calls are triggered in Line 9. We have two further cases:

38

(a) If $w = v_{i(t_0)}$, we consider what parent evoked ApproxMarginColouring$(t_0, \gamma; M, R)$.

    (i) It is evoked by ApproxMarginColouring-Set(S, γ) when the latter processes some $e' \in \mathcal{E}$ or $e' = S$. Then $w \in e'$ and there exists $i < k$ such that $x^{(i)} = \mathsf{TS}(e', 0)$. By the induction hypothesis, there is a path $P = (y_1, y_2, \ldots, y_{m'})$ for $x^{(i)}$ with $y_{m'} = x^{(i)}$. Because $x^{(i)}$ triggers a recursive call of ApproxMarginColouring$(t_0, \gamma; M, R)$, $t_0 \in x^{(i)}$. Also, $t_0 = \mathrm{pred}_w(t_0) \in x^{(k)} = \mathsf{TS}(e, t_0)$. It suffices to take the path $P' = (P, x^{(k)})$.

    (ii) It is evoked by ApproxMarginColouring$(t_1, \gamma; M, R)$ for some $t_1 < t_0$ when the latter processes some $e' \in \mathcal{E}$. Then $w \in e'$ and there exists $i < k$ such that and $x^{(i)} = \mathsf{TS}(e', t_1)$. By the induction hypothesis, there is a path $P = (y_1, y_2, \ldots, y_{m'})$ for $x^{(i)}$ with $y_{m'} = x^{(i)}$. Note that $t_0 \in x^{(i)}$ because $x^{(i)}$ triggers a recursive call of ApproxMarginColouring$(t_0, \gamma; M, R)$ and $t_0 = \mathrm{pred}_w(t_1) \in x^{(k)} = \mathsf{TS}(e, t_0)$. It suffices to take the path $P' = (P, x^{(k)})$.

(b) Otherwise w must have been added into V' in Line 10 after some recursive calls, which implies that there exists $e' \in \mathcal{E}$ and $i < k$ such that $w \in e'$ and $x^{(i)} = \mathsf{TS}(e', t_0)$. By the induction hypothesis, there is a path $P = (y_1, y_2, \ldots, y_{m'})$ for $x^{(i)}$ with $y_{m'} = x^{(i)}$. Note that $\mathrm{pred}_w(t_0) \in x^{(k)} = \mathsf{TS}(e, t_0)$ and $\mathrm{pred}_w(t_0) \in x^{(i)} = \mathsf{TS}(e', t_0)$. It suffices to take $P' = (P, x^{(k)})$.

This finishes the induction proof. □

Next, we show the following property for the set $V_{\mathcal{B}}^{\mathrm{col}}$.

Lemma 6.18. *For all $x \in V_{\mathcal{B}}^{\mathrm{col}} \setminus \{\mathsf{TS}(S, 0)\}$, there exists $j \in [s]$ and $t_0 \in x$ such that $r_{t'} = \perp$ or $r_{t'} = j$ for all $t' \in x \setminus \{t_0\}$.*

Proof. Fix $x \in V_{\mathcal{B}}^{\mathrm{col}} \setminus \{\mathsf{TS}(S, 0)\}$. Then x joins $V_{\mathcal{B}}^{\mathrm{col}}$ in the following two cases.

    (1) $x = \mathsf{TS}(e, 0)$ triggers an instance of ApproxMarginColouring$(t_0, \gamma; M, R)$ for some $t_0 \leq 0$ when ApproxMarginColouring-Set(S, γ) processes some hyperedge $e \in \mathcal{E}$ in Line 10. Thus, e must satisfy the condition in Line 8, which shows that there exists $j \in [s]$ such that $r_{t'} = \perp$ or $r_{t'} = j$ for all $t' \in x$.

    (2) $x = \mathsf{TS}(e, t_0)$ triggers an instance of ApproxMarginColouring$(t_1, \gamma; M, R)$ for some $t_1 < t_0$ when ApproxMarginColouring$(t_0, \gamma; M, R)$ processes some hyperedge $e \in \mathcal{E}$ in Line 9. Thus, e must satisfy the condition in Line 7, which shows that there exists some $j \in [s]$ such that $r_{t'} = \perp$ or $r_{t'} = j$ for all $t' \in x \setminus \{t_0\}$.

This finishes the proof. □

Recall the definition of 2-tree in Definition 5.15. We are now ready to prove Lemma 6.13.

Proof of Lemma 6.13. By Lemma 6.16, it suffices to show $\mathbf{Pr}\left[\left|V_{\mathcal{B}}^{\mathrm{col}}\right| \geq 2\Delta k^2 \cdot \eta\right] \leq 2^{-\eta}$ for each $\eta \geq 1$ as

$$\mathbf{Pr}\left[|R| \geq 4\Delta^2 k^5 \cdot \eta\right] \leq \mathbf{Pr}\left[|V_{\mathcal{B}}^{\mathrm{col}}|2\Delta k^3 \geq 4\Delta^2 k^5 \cdot \eta\right] = \mathbf{Pr}\left[|V_{\mathcal{B}}^{\mathrm{col}}| \geq 2\Delta k^2 \cdot \eta\right] \leq \left(\frac{1}{2}\right)^\eta.$$

Fix $\eta \geq 1$. Assume $\left|V_{\mathcal{B}}^{\mathrm{col}}\right| \geq 2\Delta k^2 \cdot \eta$. Note that $V_{\mathcal{B}}^{\mathrm{col}}$ is finite because ApproxMarginColouring terminates within a finite number of steps. By Corollary 6.15, Lemma 6.17, and Lemma 5.17 there exists a 2-tree $T \subseteq V_{\mathcal{B}}^{\mathrm{col}}$ of size i such that $\mathsf{TS}(S, 0) \in T$. For each $\eta \geq 1$, denote by $\mathcal{T}_S^\eta$ the set of 2-trees T in G_H^S of size η such that $\mathsf{TS}(S, 0) \in T$. Then by a union bound, we have

$$\mathbf{Pr}\left[\left|V_{\mathcal{B}}^{\mathrm{col}}\right| \geq 2\Delta k^2 \cdot \eta\right] \leq \sum_{T \in \mathcal{T}_S^\eta} \mathbf{Pr}\left[T \subseteq V_{\mathcal{B}}^{\mathrm{col}}\right].$$

By Lemma 6.18, the event $T \subseteq V_{\mathcal{B}}^{\mathrm{col}}$ implies for each $x \in T \setminus \{\mathsf{TS}(S, 0)\}$, there exists $t_0 \in x$ and $j \in [s]$ such that $r_{t'} = \perp$ or $r_{t'} = j$ for all $t' \in x \setminus \{t_0\}$. Due to $\lfloor q/s \rfloor^k \geq 4\mathrm{e}qs\Delta k$ and Lemma 6.7, for

any $x \in T \setminus \{\mathsf{TS}(S,0)\}$, this happens with probability at most

$$sk \left(\frac{\left\lceil \frac{q}{s} \right\rceil}{q} \left(1 + \frac{1}{s}\right) + \frac{1}{4s} \right)^{k-1} \le sk \left(\frac{5}{4s} \left(1 + \frac{1}{s}\right) + \frac{1}{4s} \right)^{k-1} \le 2k \left(\frac{2}{s} \right)^{k-2},$$

where we obtain the first term by a union bound over all possible $t_0 \in x$ and $j \in [s]$ and noting $q \ge 4s$, and the second inequality is by $s \ge 6\Delta^{\frac{2}{k-2}} \ge 6$.

Since for any 2-tree $T \subseteq V_H^S$, the timestamps in vertices of T are pairwise disjoint, and the event above are all mutually independent. Thus, we have $\mathbf{Pr}\left[T \subseteq V_{\mathcal{B}}^{\mathrm{col}} \right] \le (2k)^{(|T|-1)}(2/s)^{(k-2)\cdot(|T|-1)}$ and by Corollary 6.15 and Lemma 5.16, it holds that

$$\sum_{T \in \mathcal{T}_S^\eta} \mathbf{Pr}\left[T \subseteq V_{\mathcal{B}}^{\mathrm{col}} \right] \le \frac{(4e\Delta^2 k^4)^{\eta-1}}{2} \cdot (2k)^{\eta-1} \cdot \left(\frac{2}{s} \right)^{(k-2)(i-1)}$$

$$\le \frac{1}{2} \left(\frac{8e\Delta^2 k^5}{(s/2)^{k-2}} \right)^{\eta-1}$$

$$\le 2^{-\eta},$$

where the last inequality holds because $s \ge 2 \left(8e\Delta^2 k^5 \right)^{\frac{1}{k-2}}$ from $k \ge 20$ and $s \ge 6\Delta^{\frac{2}{k-2}}$. $\qquad\square$

6.5. **Improved bounds for linear hypergraphs.** We now give a marginal sampler for linear hypergraphs, and prove Lemma 6.3. Let $\delta > 0$ be a constant $k \ge \frac{50(1+\delta)^2}{\delta^2}$, and $q \ge 50\Delta^{\frac{2+\delta}{k-3}}$. Given as inputs a linear k-uniform hypergraph $H = (V, \mathcal{E})$ with the maximum degree Δ, a set of vertices $S \subseteq V$ that $|S| \le k$, and a parameter $\gamma > 0$, The algorithm is almost the same, except that we replace the truncation condition in Line 20 of Algorithm 8 and Line 18 of Algorithm 7 with

$$(32) \qquad |R| \ge 3 \cdot 10^4 \left(\frac{1+\delta}{\delta} \right)^2 \Delta^3 k^{10} \left\lceil \log \frac{1}{\gamma} \right\rceil$$

Like what we have done for linear hypergraph independent sets in Section 5.3, much of the analysis of the general hypergraph colourings can be applied to the linear case. We also reuse some other proved results from Section 5.3.

The running time of the modified algorithm is bounded the same way as Lemma 6.10, whose proof we shall omit again.

Lemma 6.19. *The running time of the modified algorithm is*

$$O\left(\left(\frac{1+\delta}{\delta} \right)^4 \Delta^8 k^{21} \log^2 \frac{1}{\gamma} q^{6\cdot 10^4 (\frac{1+\delta}{\delta})^2 \Delta^4 k^{11} \log \frac{1}{\gamma}} \right).$$

Next, we bound the truncation error.

Lemma 6.20. *Denote $\eta = \lceil \log \frac{1}{\gamma} \rceil$. If the projection scheme $[q] \to [s]$ satisfies (a) $q \ge 4s$, (b) $s \ge 6\Delta^{\frac{1+1/(1+2/\delta)}{k}}$, and (c) $\lfloor q/s \rfloor^k \ge 4eqs\Delta k$, then upon the termination of the modified algorithm, the size of R satisfies*

$$(33) \qquad \mathbf{Pr}\left[|R| \ge 3 \cdot 10^4 \left(\frac{1+\delta}{\delta} \right)^2 \Delta^3 k^{10} \cdot \eta \right] \le 2^{-\eta}.$$

Assuming this for now, we prove Lemma 6.3.

Proof of Lemma 6.3. Choose $s = \lfloor q^{\frac{1}{2}} \rfloor$, and then verify the three conditions of Lemma 6.20.

(a) Because $k \ge 50(1+\delta)^2/\delta^2 \ge 50 > 3$, it holds that $q \ge 50\Delta^{(2+\delta)/(k-3)} \ge 50 > 16$, and hence $q > 4\sqrt{q} \ge 4s$.

(b) This is derived as follows.

$$s \ge \sqrt{q} - 1 > 7\Delta^{\frac{1+\delta/2}{(k-3)}} - 1 \ge 7\Delta^{\frac{1+1/(1+2/\delta)}{k-3}} - 1 \ge 7\Delta^{\frac{1+1/(1+2/\delta)}{k}} - 1 \ge 6\Delta^{\frac{1+1/(1+2/\delta)}{k}}.$$

(c) For all $k \geq 50$, we have

$$k^{\frac{1+1/(1+2/\delta)}{k-3}} \leq k^{\frac{2}{k-3}} < 1.2 < 2.$$

This gives

$$\left\lfloor \frac{q}{s} \right\rfloor \geq 3 \times 2 \times \Delta^{\frac{1+1/(1+2/\delta)}{k-3}} > 3(k\Delta)^{\frac{1+1/(1+2/\delta)}{k-3}},$$

and hence

$$\left\lfloor \frac{q}{s} \right\rfloor^k \geq 3^{k-3}(k\Delta)\left\lfloor \frac{q}{s} \right\rfloor^3 \geq 3^{40} \cdot 4\mathrm{e}k\Delta \cdot \left\lfloor \frac{q}{s} \right\rfloor^3 \geq 4\mathrm{e}qs\Delta k.$$

We use the modified ApproxMarginColouring-Set(S, γ) to sample from the distribution μ_S. By Lemma 6.19, the running time is $O((\frac{1+\delta}{\delta})^4 \Delta^8 k^{21} \log^2 \frac{1}{\gamma} q^{6 \cdot 10^4 (\frac{1+\delta}{\delta})^2 \Delta^4 k^{11} \log \frac{1}{\gamma}})$. It is straightforward to verify that Algorithm 8 uses at most $6 \cdot 10^4 \left(\frac{1+\delta}{\delta}\right)^2 \Delta^3 k^{10} \left\lceil \log \frac{1}{\gamma} \right\rceil + 1$ random variables with domain size at most $q + 1$. By combining Lemma 6.12 and Lemma 6.13, the output distribution is γ-close to μ_S. This proves the lemma. □

The rest of this section is dedicated to the proof of Lemma 6.20. Recall from Section 5.3 that we introduce the self-neighbourhood powered witness graphs to make use of linearity under the hypergraph independent set setting. We provide an analogous definition regarding the modified witness graph (Definition 6.14) whilst dealing with hypergraph colourings.

Definition 6.21. Let $G_H^S = (V_H^S, E_H^S)$ be the modified witness graph in Definition 6.14. The self-neighbourhood powered witness graph $G_H^{S,\mathrm{self}} = (V_H^S, E_H^{S,\mathrm{self}})$ is defined on the same vertex set and the edge set $E_H^{S,\mathrm{self}} = E_H^S \cup E'$ such that

$$E' = \{\{x, y\} \mid (\exists w \in V_H^S \text{ s.t. } w \in N_{\mathrm{self}}(x) \wedge w \in N(y)) \vee (\exists w \in V_H^S \text{ s.t. } w \in N(y) \wedge w \in N_{\mathrm{self}}(x))\}.$$

Observe that $G_H^{S,\mathrm{self}}$ modifies G_H^{self} by only adding the new vertex $\mathsf{TS}(S, 0)$ and some additional edges that connects $\mathsf{TS}(S, 0)$ with several other vertices, if $S \notin \mathcal{E}$.

The next lemma is an analogue of Lemma 5.21.

Lemma 6.22. *If $|S| \leq k$, then the maximum degree of $G_H^{S,\mathrm{self}}$ is at most $10k^3\Delta - 1$.*

Proof. For any $x \in V_H^S$, by Lemma 5.11, $|N_{\mathrm{self}}(x)| \leq 2k$ and by Corollary 6.15, $N(x) \leq 2\Delta k^2 - 1$. Hence, the maximum degree of $G_H^{S,\mathrm{self}}$ is at most $2\Delta k^2 - 1 + 2 \times 2k \times (2\Delta k^2 - 1) \leq 10k^3\Delta - 1$. □

We also have the following lemma as a counterpart of Lemma 5.22.

Lemma 6.23. *Given a k-uniform linear hypergraph $H = (V, \mathcal{E})$ with the maximum degree Δ, and a subset of vertices $S \subseteq V$, let $G_H^S = (V_H^S, E_H^S)$ be the modified witness graph with respect to S. Let $V_{\mathcal{B}}^{\mathrm{col}} \subseteq V_H^S$ be a finite subset containing $\mathsf{TS}(S, 0) \in V_{\mathcal{B}}^{\mathrm{col}}$ and connected in G_H^S. Then, there exists $V_{\mathcal{B}}^{\mathrm{lin}} \subseteq V_{\mathcal{B}}^{\mathrm{col}}$ such that*

(C1) $\mathsf{TS}(S, 0) \in V_{\mathcal{B}}^{\mathrm{lin}}$ and $|V_{\mathcal{B}}^{\mathrm{lin}}| \geq \lfloor \frac{|V_{\mathcal{B}}|}{2k+1} \rfloor$,
(C2) the induced subgraph $G_H^{S,\mathrm{self}}[V_{\mathcal{B}}^{\mathrm{lin}}]$ is connected, and
(C3) for any two distinct vertices $x_1, x_2 \in V_{\mathcal{B}}^{\mathrm{lin}} \setminus \{\mathsf{TS}(S, 0)\}$, it holds that $|x_1 \cap x_2| \leq 1$.

Proof. The proof is almost the same as the proof of Lemma 5.22: we apply the explicit construction algorithm there starting from the vertex $\mathsf{TS}(S, 0)$. To see (C1), note that the number of self-neighbours is at most $2k$ for any vertex in G_H^S. (C2) follows from the construction and the same argument as (L2) in Lemma 5.22. (C3) also follows the same argument as (L3) in Lemma 5.22. We remark that the artifact $\mathsf{TS}(S, 0)$ needs to be excluded as there is no control on how the set S intersects with the hyperedges in H. □

Now we prove Lemma 6.20. Recall from the last section the toolkit of 2-block-trees, including Definition 5.23, Lemma 5.24, and Lemma 5.25.

41

Proof of Lemma 6.20. Let $V_{\mathcal{B}}^{\text{col}}$ be the set generated by the modified algorithm (see (32)) as defined in (30). Again, $V_{\mathcal{B}}^{\text{col}} \subseteq V_H^S$ is a finite subset because the algorithm terminates after a finite number of steps. Define a parameter

$$\theta := \left\lceil \frac{6(1+\delta)}{\delta} \right\rceil,$$

and by this choice, $\mathbf{Pr}\left[|R| \geq 3 \cdot 10^4 (\frac{1+\delta}{\delta})^2 k^{10} \Delta^3 \eta\right] \leq \mathbf{Pr}\left[|R| \geq 600\theta^2 k^{10} \Delta^3 \eta\right]$. Using Lemma 6.16, it holds for any positive integer η that

$$\mathbf{Pr}\left[|R| \geq 600\theta^2 k^{10} \Delta^3 \eta\right] \leq \mathbf{Pr}\left[|V_{\mathcal{B}}^{\text{col}}| 2k^3 \Delta \geq 600\theta^2 k^{10} \Delta^3 \eta\right] \leq \mathbf{Pr}\left[|V_{\mathcal{B}}^{\text{col}}| \geq 300\theta^2 k^7 \Delta^2 \eta\right].$$

Hence, it suffices to show

$$\mathbf{Pr}\left[|V_{\mathcal{B}}^{\text{col}}| \geq 300\theta^2 k^7 \Delta^2 \eta\right] \leq \left(\frac{1}{2}\right)^{\eta}$$

for any integer $\eta \geq 1$.

Fix an integer $\eta \geq 1$, and assume $|V_{\mathcal{B}}^{\text{col}}| \geq 300\theta^2 k^7 \Delta^2 \eta$, by Lemma 6.17 and Lemma 6.23, we can find the set $V_{\mathcal{B}}^{\text{lin}} \subseteq V_{\mathcal{B}}^{\text{col}}$ with size $|V_{\mathcal{B}}^{\text{lin}}| \geq \lfloor \frac{|V_{\mathcal{B}}^{\text{col}}|}{2k+1} \rfloor \geq \lfloor \frac{|V_{\mathcal{B}}^{\text{col}}|}{3k} \rfloor \geq \theta^2 (10k^3 \Delta)^2 \eta$ such that $\text{TS}(S, 0) \in V_{\mathcal{B}}^{\text{lin}}$ and the conditions in Lemma 6.23 get fulfilled. Moreover, it is straightforward to find a subset $U \subseteq V_{\mathcal{B}}^{\text{lin}}$ with size exactly $|U| = \theta^2 (10k^3 \Delta)^2 \eta$ such that $\text{TS}(S, 0) \in U$ and the rest of Lemma 6.23 are satisfied by U. By Lemma 6.22, the maximum degree of $G_H^{S, \text{self}}[U]$ is at most $10k^3 \Delta$. Since $G_H^{S, \text{self}}[U]$ is a finite connected subgraph in $G_H^{S, \text{self}}$, by Lemma 5.24, we can find a 2-block-tree $\{C_1, C_2, \ldots, C_\eta\}$ in $G_H^{S, \text{self}}$ with block size θ and tree size η such that $\text{TS}(S, 0) \in C_1$ and $C_j \subseteq U \subseteq V_{\mathcal{B}}^{\text{col}}$ for all $j \in [\eta]$. By Item (C3) in Lemma 6.23, for any distinct $x_1, x_2 \in (\cup_{j=1}^{\eta} C_j) \setminus \{\text{TS}(S, 0)\}$, it holds that $|x_1 \cap x_2| \leq 1$.

Define by $\mathcal{T}_S^{\eta, \theta}$ the set of all 2-block-trees $\{C_1, C_2, \ldots, C_\eta\}$ with block size θ and tree size η in graph $G_H^{S, \text{self}}$ such that

- $\text{TS}(S, 0) \in C_1$;
- let $C = (\cup_{j=1}^{\eta} C_j) \setminus \{\text{TS}(S, 0)\}$, then for any $w_1, w_2 \in C$, $|w_1 \cap w_2| \leq 1$.

Hence, if $|V_{\mathcal{B}}^{\text{col}}| \geq 300\theta^2 k^7 \Delta^2 \eta$, then there exists a 2-block-tree $\{C_1, C_2, \ldots, C_\eta\} \in \mathcal{T}_S^{\eta, \theta}$ such that $C_j \subseteq V_{\mathcal{B}}^{\text{col}}$ for all $j \in [\eta]$. By a union bound over all 2-block-trees in $\mathcal{T}_S^{\eta, \theta}$, we have

$$\mathbf{Pr}\left[|V_{\mathcal{B}}^{\text{col}}| \geq 3\theta^2 k \Delta^2 \eta\right] \leq \sum_{\{C_1, \ldots, C_\eta\} \in \mathcal{T}_S^{\eta, \theta}} \mathbf{Pr}\left[\forall j \in [\eta], C_j \subseteq V_{\mathcal{B}}^{\text{col}}\right].$$

Fix a 2-block-tree $\{C_1, \ldots, C_\eta\} \in \mathcal{T}_S^{\eta, \theta}$. By definition, for any j and ℓ that $j \neq \ell$, we have $\text{dist}_{G_H^{S, \text{self}}}(C_j, C_\ell) \geq 2$, and thus for any $x_j \in C_j$ and $x_\ell \in C_\ell$, it holds that $x_j \cap x_\ell = \varnothing$. For any $j \in [\eta]$, and any two $w_1, w_2 \in C_j \setminus \{\text{TS}(S, 0)\}$, it holds that $|w_1 \cap w_2| \leq 1$ by the definition of $\mathcal{T}_S^{\eta, \theta}$. Let $C_j = \{e_1^j, e_2^j, \ldots, e_\theta^j\}$. Without loss of generality, assume $\text{TS}(S, 0) = e_\theta^j$ if $\text{TS}(S, 0) \in C_j$. For each $\ell \in [\theta]$, define

(34)

$\mathcal{B}_\ell^j$: There exists $x \in [s]$ and $y \in e_\ell^j$ s.t., for all $t' \in e_\ell^j \setminus (\{y\} \cup \{\text{TS}(S, 0)\})$, either $r_{t'} = \perp$ or $r_{t'} = x$.

42

We then have

$$\mathbf{Pr}\left[\forall j \in [\eta], C_j \subseteq V_{\mathcal{B}}^{\mathrm{col}}\right]$$

(By Lemma 6.18 and chain rule) $\displaystyle \leq \prod_{1\leq j\leq \eta}\prod_{1\leq \ell\leq \theta}\mathbf{Pr}\left[\mathcal{B}_\ell^j \mid \bigwedge_{\substack{(j',\ell'):\\ j'<j \vee (j'=j \wedge \ell'<\ell)}}\mathcal{B}_{\ell'}^{j'}\right]$

(By $e_\ell^j \cap e_{\ell'}^{j'} = \varnothing$ for $j \neq j'$) $\displaystyle = \prod_{1\leq j\leq \eta}\prod_{1\leq \ell\leq \theta}\mathbf{Pr}\left[\mathcal{B}_\ell^j \mid \bigwedge_{\ell'<\ell}\mathcal{B}_{\ell'}^{j}\right]$

$(\star)$ $\displaystyle \leq \prod_{1\leq j\leq \eta}\prod_{1\leq \ell\leq \theta-1}\left(sk\cdot\left(\frac{\left\lceil\frac{q}{s}\right\rceil}{q}\left(1+\frac{1}{s}\right)+\frac{1}{4s}\right)^{k-\theta}\right)$

(By $q \geq 4s$ and $s \geq 6$) $\displaystyle \leq (sk)^{(\theta-1)\eta}\left(\frac{2}{s}\right)^{\eta(k-\theta)(\theta-1)} \leq (2k)^{(\theta-1)\eta}\left(\frac{2}{s}\right)^{\eta(k-\theta-1)(\theta-1)}$

The inequality $(\star)$ is due to (1) a union bound according to the definition in (34); (2) the local uniformity property in Lemma 6.7; and (3) (C3) in Lemma 6.23 for each e_ℓ^j where $j \in [\theta-1]$.

Next, we count the number of possible 2-block-trees in $\mathcal{T}_S^{\eta,\theta}$, which can be upper bound by the number of all 2-block-trees $\{C_1, C_2, \ldots, C_\eta\}$ with block size θ and tree size η in $G_H^{S,\mathrm{self}}$ such that $\mathrm{TS}(S,0) \in C_1$. By Lemma 5.25 and Lemma 6.22, we have

$$\left|\mathcal{T}_S^{\eta,\theta}(U)\right| \leq (\theta e^\theta (10k^3\Delta)^{\theta+1})^\eta.$$

Hence, we only need to prove that

$$(\theta e^\theta (20k^4\Delta)^{\theta+1})^\eta\left(\frac{2}{s}\right)^{(k-\theta-1)(\theta-1)\eta} \leq \left(\frac{1}{2}\right)^\eta$$

which is equivalent to

$$\left(\frac{s}{2}\right)^{(k-\theta-1)(\theta-1)} \geq 2\theta e^\theta (20k^4\Delta)^{\theta+1} \iff \left(\frac{s}{2}\right)^{(k-\theta-1)} \geq (2\theta)^{1/(\theta-1)} e^{\theta/(\theta-1)}(20k^4\Delta)^{(\theta+1)/(\theta-1)}.$$

We derive this as follows. Observe the following inequalities

(35) $$k \geq \frac{50(1+\delta)^2}{\delta^2} \geq \theta^2 - 1 \implies k - \theta - 1 \geq (1-1/(\theta-1))k,$$

(36) $$k \geq 50 \implies (120k^4)^{2/k} < 2.3 < 3.$$

Then we have

(By condition) $\displaystyle s \geq 6\Delta^{\frac{1+1/(1+2/\delta)}{k}}$

(By (36)) $\displaystyle \geq 2\left(120k^4\right)^{\frac{2}{k}}\Delta^{\frac{1+1/(1+2/\delta)}{k}}$

$\displaystyle \geq 2\left(120k^4\Delta\right)^{\frac{1+1/(1+2/\delta)}{k}}$

(Use $6+\frac{6}{\delta} \leq \theta \leq 7+\frac{6}{\delta}$) $\displaystyle \geq 2\left(6^{\frac{\theta-1}{k(\theta-2)}}(20k^4\Delta)^{\frac{\theta+1}{k(\theta-2)}}\right)$

(By (35)) $\displaystyle \geq 2\left(6\left(20k^4\right)^{\frac{\theta+1}{\theta-1}}\right)^{\frac{1}{k-\theta-1}}.$

The desired inequality then follows by noticing that $\theta \geq 6$. $\qquad\square$

6.6. **Proof of Lemma 6.12.** Now we prove Lemma 6.12, verifying the correctness of Algorithm 8. We follow the same proof strategy as in Section 4. We will first define a forward version marginal sampler $\mathcal{A}$ in Algorithm 9 for any finite T, and then give its backward counterpart $\mathcal{B}$ in Algorithm 10. Using the same randomness, $\mathcal{A}$ and $\mathcal{B}$ are perfectly coupled. Algorithm 8 is the same as $\mathcal{B}$ taking $T \to \infty$ and with truncation. Lemma 6.12 then follows in a similar manner as in Theorem 4.9.

For the rest of this subsection, we fix a k-uniform hypergraph $H = (V, \mathcal{E})$ with the maximum degree at most Δ together with a set of colours $[q]$, a projection scheme h with parameter s, a subset of vertices $S \subseteq V$ and a real number $\gamma > 0$ satisfying the condition in Lemma 6.12. Fix an integer $T \geq 0$. For the forward marginal sampler, recall that the systematic scan Glauber dynamics on the distribution ψ is defined as follows:

- let $Y_{-T} \in [s]^V$ be an arbitrary feasible configuration;
- for each t from $-T + 1$ to 0, the transition $Y_{t-1} \to Y_t$ is defined as follows:
  (1) let $w = v_{i(t)}$, where $i(t) = (t \bmod n) + 1$, and let $Y_t(u) = Y_{t-1}(u)$ for all $u \neq w$;
  (2) sample $r_t \sim \psi^{\mathrm{LB}}$, if $r_t \neq \perp$, then let $Y_t(w) = r_t$; otherwise, let

$$\sigma_\Lambda = \mathsf{Boundary}(t)$$

   where Boundary is defined in Algorithm 6.
  (3) sample $Y_t(w) \sim \psi_w^{\mathrm{pad}, \sigma_\Lambda}$;
- output Y_0.

Here we sample from the padding distribution by the method in (10) using a sequence of real random variables $\{U_t\}_{-T < t \leq 0}$.

Denote by $Y_0^{\mathcal{P}(T)} \in [s]^V$ the output of the above process. We will always consider $T \geq n$. It holds that for any $u \in V$, $Y_0^{\mathcal{P}(T)}(u) = Y_t(v_{i(t)})$, where $t = \mathrm{pred}_u(0)$.

Given $Y_0^{\mathcal{P}(T)}$ generated by the process $\mathcal{P}(T)$, what we are really interested in is a sample from μ (instead of ψ) conditioned on $Y_0^{\mathcal{P}(T)}$. To this end, we introduce the algorithm $\mathcal{A}(S, Y_0, \{r_t\}_{-T < t \leq 0})$, described in Algorithm 9, that takes a configuration Y_0 and a set $\{r_t\}_{-T < t \leq 0}$ as its input. This algorithm is similar to Algorithm 6 in that it uses BFS to find a boundary so that S is independent from the outside, and in addition it enumerates all possibilities inside to generate a sample. Let $X_S^{(T)} = \mathcal{A}\left(S, Y_0^{\mathcal{P}(T)}, \{r_t\}_{-T < t \leq 0}\right)$, where $Y_0^{\mathcal{P}(T)}$ and $\{r_t\}_{-T < t \leq 0}$ are generated by $\mathcal{P}(T)$. Note that $\left\{X_S^{(T)}\right\}_{T \geq n}$ is an infinite sequence of random variables. The next lemma shows that it converges to the desired marginal distribution.

Lemma 6.24. If $\lfloor q/s \rfloor^k \geq 4eqs\Delta k$, then $d_{\mathrm{TV}}\left(X_S^{(T)}, \mu_S\right) = 0$ as $T \to \infty$.

Proof. Fix $Y = Y_0^{\mathcal{P}(T)} \in [s]^V$ and $\{r_t\}_{-T < t \leq 0}$ generated by $\mathcal{P}(T)$. We first show that the output $X_S = X_S^{(T)}$ follows the distribution μ_S^Y. For each hyperedge $e \in \mathcal{E}$, if e is picked in Line 3 , then after the loop, either $e \subseteq V'$ or e is satisfied by σ, and thus e cannot be picked in Line 3 again. The algorithm $\mathcal{A}$ always terminates. Suppose the final σ is defined on a subset $\Lambda \subseteq V$, namely $\sigma \in [s]^\Lambda$. Note that X is sampled from μ_S^σ. We show that the configuration σ satisfies:

- $\sigma = Y_\Lambda$ and $S \subseteq V' \subseteq \Lambda$;
- for all $e \in \mathcal{E}$ such that $e \cap V' \neq \varnothing$ and $e \cap (V \setminus V') \neq \varnothing$, e is satisfied by σ.

If these two properties are true, then conditioned on σ, all constraints on hyperedges in the boundary of V' are satisfied by σ, which implies $\mu_S^Y = \mu_S^\sigma$. This is similar to the argument in Lemma 6.9.

Note that $S \subseteq V'$ because V' is initialised as S and it does not remove vertices. The fact $V' \subseteq \Lambda$ follows from Line 6 and Line 7. Since both Y and $\{r_t\}$ are generated by $\mathcal{P}(T)$, for any $u \in V$, if $r_{\mathrm{pred}_u(0)} \neq \perp$, then $Y_u = r_{\mathrm{pred}_u(0)}$. Hence, it is straightforward to see $\sigma = Y_\Lambda$. The second property follows from the fact that the while-loop has terminated.

By Lemma 6.9, Boundary (Algorithm 6) satisfies Condition 4.3. The process $\mathcal{P}(T)$ faithfully simulates the systematic scan Glauber dynamics on ψ. By Lemma 6.7, the systematic scan Glauber dynamics on

---

Algorithm 9: $\mathcal{A}\left(S, Y_0, \{r_t\}_{-T < t \leq 0}\right)$

Input: a hypergraph $H = (V, \mathcal{E})$, a set of colours $[q]$, a projection scheme $h : [q] \to [s]$ that defines the projected distribution ψ, a subset $S \subseteq V$, a configuration $Y_0 \in [s]^V$, and random variables $r_t \in [s] \cup \{\bot\}$ for $-T \leq t \leq 0$;

Output: $\sigma \in [s]^\Lambda$ for some $S \subseteq \Lambda \subseteq V$

1   $\sigma \leftarrow Y_0(S)$ and $V' \leftarrow S$;

2   **while** $\exists e \in \mathcal{E}$ s.t. $e \cap V' \neq \varnothing$, $e \cap (V \setminus V') \neq \varnothing$ and e is not satisfied by σ **do**

3   choose such e with the lowest index;

4   **if** $\exists j \in [s]$ s.t. $\forall u \in e$, $r_{\mathrm{pred}_u(0)} \in \{\bot, j\}$ **then**

5   **forall** $u \in e$ **do**

6            $\sigma(u) \leftarrow Y_0(u)$;

7         $V' \leftarrow V' \cup e$;

8   **else**

9         $U \leftarrow \{u \in e \mid r_{\mathrm{pred}_u(0)} \neq \bot\}$;

10   **forall** $u \in U$ **do**

11           $\sigma(u) \leftarrow r_{\mathrm{pred}_u(0)}$;

12   enumerate all colourings $X \in \otimes_{u \in V'} h^{-1}(\sigma_u)$ on V' to compute the marginal distribution of S on the sub-hypergraph $H[V']$, which is equivalent to μ_S^σ;

13   sample $X_S \sim \mu_S^\sigma$;

14   **return** X_S;

ψ is irreducible, which implies that $Y_0^{\mathcal{P}(T)}$ follows the distribution ψ as $T \to \infty$. Since $X_S^{(T)} \sim \mu_S^{Y_0^{\mathcal{P}(T)}}$, $d_{\mathrm{TV}}\left(X_S^{(T)}, \mu_S\right) = 0$ as $T \to \infty$. $\square$

The algorithm $\mathcal{A}$ uses too much randomness and next we give its backwards version $\mathcal{B}$ that achieves the same output distribution. Fix the initial configuration Y_{-T} of $\mathcal{P}(T)$ as an arbitrary feasible configuration, say $Y_{-T}(u) = 1$ for all $u \in V$. Given the description of the distribution ψ, an integer $T \geq n$ and a subset $S \subseteq V$, the algorithm $\mathcal{B}$ returns a random variable that follows the same distribution as $X_S = \mathcal{A}\left(S, Y_0^{\mathcal{P}(T)}, \{r_t\}_{-T < t \leq 0}\right)$.

To construct $\mathcal{B}$, we need yet another algorithm $C = C_T(t; M, R)$, which is to plug Algorithm 6 as Boundary into $\mathrm{Resolve}_T(t; M, R)$ (Algorithm 1) in this context. As before, C maintains two global data structures M and R, initialised as $\bot^{\mathbb{Z}}$ and $\varnothing$ respectively. All recursive calls of C access the same M and R. The algorithm C uses LB-Sample$(t; R)$ to draw random samples from ψ^{LB}, and samples from the padding distribution by the method in (10). Given the random variables $\{r_t, U_t\}_{-T < t \leq 0}$, C and LB-Sample become deterministic. We define $C \in [s]^V$ where $C(u) = C_T(\mathrm{pred}_u(0); M, R)$ for all $u \in V$. Note that C is a function of $\{r_t, U_t\}_{-T < t \leq 0}$, and is thus a random vector.

Lemma 6.25. *Use the same random variables* $\{r_t, U_t\}_{-T < t \leq 0}$ *in* $C_T(\mathrm{pred}_u(0); M, R)$ *for all* $u \in V$ *and in* $\mathcal{P}(T)$ *to generate* C *and* Y_0, *respectively. The distribution of* C *is the same as that of* Y_0.

Proof. By Lemma 6.9, the subroutine Boundary satisfies Condition 4.3. Following the proof of Theorem 4.6, once we used the same random variables $\{r_\ell, U_\ell\}_{-T < t \leq 0}$, C and the $\mathcal{P}(T)$ are perfectly coupled and the lemma follows. $\square$

Lemma 6.25 says that we can use C to simulate Y_0 in $\mathcal{P}(T)$. We may also use LB-Sample to access $\{r_t\}_{-T < t \leq 0}$ "on demand". The algorithm $\mathcal{B} = \mathcal{B}_T(S)$ (Algorithm 10) takes S as an input, and is the same as $\mathcal{A}$ except that any access to Y_0 is replaced by a call to C, and any access to $\{r_t\}_{-T < t \leq 0}$ is replaced by a call to LB-Sample. Similar to Algorithm 8, $\mathcal{B}$ together with its subroutines maintains two global data structures M and R, which are initialised respectively as $M_0 = \bot^{\mathbb{Z}}$ and $R_0 = \varnothing$.

Algorithm 10: $\mathcal{B}_T(S)$

Input: a hypergraph $H = (V, \mathcal{E})$, a set of colours $[q]$, a projection scheme $h : [q] \to [s]$ that
defines the projected distribution ψ, a subset $S \subseteq V$, an integer $T \leq -n$ and a parameter
$\gamma > 0$;

Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\bot\}$ and a set R;

Output: a random assignment in $\tau \in [q]^S$

1 $M \leftarrow \bot^{\mathbb{Z}}$ and $R \leftarrow \varnothing$;

2 $\sigma \leftarrow \varnothing, V' \leftarrow S$;

3 $\sigma(u) \leftarrow C_T(\text{pred}_u(0); M, R)$ for all $u \in S$;

4 **while** $\exists e \in \mathcal{E}$ s.t. $e \cap V' \neq \varnothing$, $e \cap (V \setminus V') \neq \varnothing$ *and e is not satisfied by σ* **do**

5   choose such e with the lowest index;

6   **if** $\exists j \in [s]$ *s.t.* $\forall u \in e$, LB-Sample$(\text{pred}_u(0); R) \in \{\bot, j\}$ **then**

7   **forall** $u \in e$ **do**

8              $\sigma(u) \leftarrow C_T(\text{pred}_u(0); M, R)$;

9          $V' \leftarrow V' \cup e$;

10   **else**

11          $U \leftarrow \{u \in e \mid$ LB-Sample$(\text{pred}_u(0); R) \neq \bot\}$;

12   **forall** $u \in U$ **do**

13              $\sigma(u) \leftarrow$ LB-Sample$(\text{pred}_u(0); R)$;

14 enumerate all colourings $X \in \otimes_{u \in V'} h^{-1}(\sigma_u)$ on V' to compute the marginal distribution of S
on the sub-hypergraph $H[V']$, which is equivalent to μ_S^σ;

15 sample $X_S \sim \mu_S^\sigma$;

16 **return** X_S;

Denote by $B_T(S)$ to the output of the algorithm $\mathcal{B}_T(S)$. Recall that $X_S^{(T)}$ denotes the output of
$\mathcal{A}\left(S, Y_0^{\mathcal{P}(T)}, \{r_t\}_{-T < t \leq 0}\right)$. Suppose $\lfloor q/s \rfloor^k \geq 4eqs\Delta k$. By Lemma 6.25, it holds that

$$(37) \qquad \forall T \geq n, \quad d_{\text{TV}}\left(B_T(S), X_S^{(T)}\right) = 0.$$

By Lemma 6.7, if $\lfloor q/s \rfloor^k \geq 4eqs\Delta k$, Condition 4.5 is satisfied. By Lemma 6.9, the subroutine Boundary
satisfies Condition 4.3. Following the same proof of Theorem 4.6, one can verify that in this case $\mathcal{B}_\infty(S)$
terminates with probability 1 and its output $B_\infty(S)$ satisfies

$$(38) \qquad \lim_{T \to \infty} d_{\text{TV}}\left(B_\infty(S), B_T(S)\right) = 0.$$

Note that

$$\limsup_{T \to \infty} d_{\text{TV}}(\mu_S, B_T(S)) \leq \limsup_{T \to \infty} d_{\text{TV}}\left(\mu_S, X_S^{(T)}\right) + \limsup_{T \to \infty} d_{\text{TV}}\left(X_S^{(T)}, B_T(S)\right)$$

(by Lemma 6.24 and (37)) $\qquad = 0.$

Combining the above with (38), we have

$$(39) \qquad d_{\text{TV}}\left(B_\infty(S), \mu_S\right) \leq \limsup_{T \to \infty} d_{\text{TV}}\left(B_\infty(S), B_T(S)\right) + \limsup_{T \to \infty} d_{\text{TV}}\left(\mu_S, B_T(S)\right) = 0.$$

Now, we can prove Lemma 6.12.

Proof of Lemma 6.12. Let $K = 4\Delta^2 k^5 \lceil \log \frac{1}{\gamma} \rceil$ be the truncation threshold of $|R|$ in Algorithm 8. We
couple ApproxMarginColouring-Set(S, γ) and the algorithm $\mathcal{B}_\infty(S)$ by using the same random choices
$\{r_\ell, U_\ell\}_{\ell \leq 0}$, where $\{U_\ell\}_{\ell \leq 0}$ are used to realise the padding distribution as in (10). Furthermore, we
use the same uniformly at random real number $U^* \in [0, 1]$ to realise both Line 17 of Algorithm 8 and
Line 15 of $\mathcal{B}_\infty(S)$.

Let X_S be output of Algorithm 8. Similar to the proof of Theorem 4.9, $\mathcal{B}_\infty(S)$ and Algorithm 8 couple perfectly unless truncation happens. Thus,

$$\Pr_{\text{coupling}} [B_\infty(S) \neq X_S] \leq \Pr[\mathcal{E}_{\text{trun}}(K)],$$

where $B_\infty(S)$ denotes the output of the algorithm $\mathcal{B}_\infty(S)$, and the the second $\Pr$ refers to Algorithm 8. The random variable $B_\infty(S)$ is well-defined since $\mathcal{B}_\infty(S)$ terminates with probability 1. Then,

$$
\begin{aligned}
d_{\text{TV}}(X_S, \mu_S) &\leq d_{\text{TV}}(X_S, B_\infty(S)) + d_{\text{TV}}(B_\infty(S), \mu_S) \\
\text{(by (39))} \qquad &= d_{\text{TV}}(X_S, B_\infty(S)) \\
&\leq \Pr_{\text{coupling}}[B_\infty(S) \neq X_S] \leq \Pr[\mathcal{E}_{\text{trun}}(K)]. \qquad \square
\end{aligned}
$$

7. Derandomising random scan Glauber dynamics using CTTP

In this section, we elaborate how one can derandomise the MCMC algorithm based on the random scan Glauber dynamics for Gibbs distributions over bounded-degree graphs.

7.1. **CTTP for random scan Glauber dynamics.** Let $\mathcal{S} : \mathbb{Z} \to V$ be an (infinite) scan sequence and let $T > 0$. We consider the following Glauber dynamics with scan sequence $\mathcal{S}$.

The Glauber dynamics $\mathcal{P}^{\mathcal{S}}(T)$

- Initialize $X_{-T}^{\mathcal{S}} \in [q]^V$ as an arbitrary feasible configuration.
- For $t = -T+1, -T+2, \ldots, 0$, the configuration $X_t^{\mathcal{S}}$ is constructed as follows:
  (a) pick $v = \mathcal{S}(t)$, and let $X_t^{\mathcal{S}}(u) \leftarrow X_{t-1}^{\mathcal{S}}(u)$ for all $u \neq v$;
  (b) draw $r_t \sim \mu^{\text{LB}}$ independently, and let $X_t^{\mathcal{S}}(v) \leftarrow r_t$ if $r_t \neq \perp$; otherwise,

(40) $\qquad \sigma_\Lambda \leftarrow \text{Boundary}^{\mathcal{S}}(t), \quad$ (by accessing $X_{t-1}^{\mathcal{S}}$ and $\mathcal{R}_{t-1} := (r_s)_{-T < s < t}$)

  and draw $X_t^{\mathcal{S}}(v) \sim \mu_v^{\text{pad}, \sigma_\Lambda}$ independently.

Compared with the systematic scan Glauber dynamics $\mathcal{P}(T)$ defined in Section 4, the only difference is that the vertex to update is chosen according to $\mathcal{S}$ rather than (7). Furthermore, for any $u \in V$ and integer t, denote by $\text{pred}_u^{\mathcal{S}}(t)$ the last time in $\mathcal{S}$ before t at which u is updated, i.e.

(41) $$\text{pred}_u^{\mathcal{S}}(t) := \max\{s \leq t \mid \mathcal{S}(s) = u\},$$

If no such s exists, let $\text{pred}_u^{\mathcal{S}}(t) = -T$. With such replacement in mind, we can define $\text{Boundary}^{\mathcal{S}}(t)$, $\text{Resolve}_T^{\mathcal{S}}(t; M, R)$, and $\text{ApproxResolve}^{\mathcal{S}}(v, K; M, R)$, analogously to those in Section 4. An analogue of Condition 4.3 with random or fixed scan is stated as follows.

Condition 7.1. *With probability 1 over the choices of $\mathcal{S}$, the procedure $\text{Boundary}^{\mathcal{S}}(t)$ terminates and returns $\sigma_\Lambda \in [q]^\Lambda$ satisfying that $\mu_v^{\sigma_\Lambda} = \mu_v^{X_{t-1}(V \setminus \{v\})}$ for $v = v_{\mathcal{S}(t)}$.*

The correctness of $\text{Resolve}_T^{\mathcal{S}}$ is due to the next theorem, whose proof is almost identical to that of Theorem 4.6.

Theorem 7.2. *Let μ be a distribution over $[q]^V$, $T \geq 0$ be an integer, $\mathcal{S} : \mathbb{Z} \to V$ be a (fixed or random) scan sequence and $(X_t^{\mathcal{S}})_{-T \leq t \leq 0}$ be generated by the process $\mathcal{P}^{\mathcal{S}}(T)$ whose $\text{Boundary}^{\mathcal{S}}(t)$ subroutine satisfies Condition 7.1. For any $-T \leq t \leq 0$, the followings hold:*

- *For any fixed scan sequence $\mathcal{S} : \mathbb{Z} \to V$, $\text{Resolve}_T^{\mathcal{S}}(t)$ terminates in finite steps and returns a sample identically distributed as $X_t^{\mathcal{S}}(\mathcal{S}(t))$;*
- *Suppose further Condition 4.5 holds. If for randomly generated sequence $\mathcal{S}$ such that $\mathcal{P}^{\mathcal{S}}(T)$ converges to μ as $T \to \infty$, then $\text{Resolve}_\infty^{\mathcal{S}}(t)$ terminates with probability 1, and returns a sample distributed as μ_v where $v = \mathcal{S}(t)$.*

Algorithm 11: ApproxResolve-Random$(v, K; M, R)$

Input: a variable $v \in V$ and $K \geq 0$

Global variables: a map $M : \mathbb{Z} \to [q] \cup \{\bot\}$ and a set R;

Output: a random value in $[q] \cup \{\bot\}$;

1 initialize $M \leftarrow \bot^{\mathbb{Z}}$ and $R \leftarrow \varnothing$;

2 Generate infinite scan sequence $\mathcal{S} : \mathbb{Z} \to V$ where each $\mathcal{S}(i)$ is chosen from V uniformly and independently at random;

3 try :

4   **return** $\text{Resolve}_{\infty}^{\mathcal{S}}(\text{pred}_v^{\mathcal{S}}(0); M, R)$;

5 catch $|R| \geq K$ **:**

6   **return** $\bot$;

On top of this, taking $\mathcal{S}$ as an infinite random sequence whose each entry is chosen from V uniformly and independently at random gives Algorithm 11. This is the random scan analogue of Algorithm 3.

Denote by $\mathcal{E}_{\text{trun}}(K)$ as the event ApproxResolve-Random$(v, K) = \bot$. We have the following theorem that says this is precisely the error for ApproxResolve-Random(v, K) sampling from μ_v. This can be viewed as the random scan Glauber dynamics analogue for Theorem 4.9.

Theorem 7.3. *Let μ be a distribution over $[q]^V$. Assume Condition 4.5 and Condition 7.1. For $v \in V$, $K \geq 0$, and $Y = $ ApproxResolve-Random(v, K), it holds that $d_{\text{TV}}(Y, \mu_v) = \mathbf{Pr}[\mathcal{E}_{\text{trun}}(K)]$.*

Proof. Suppose that sampling from the padding distribution $\mu_{v_{i(\ell)}}^{\text{pad}, \sigma_{\Lambda}}$ in $\text{Resolve}_T^{\mathcal{S}}$ is realised in the same way as in (10), using a sequence of real numbers $U_\ell \in [0, 1)$ chosen uniformly and independently at random for each $\ell \leq 0$.

We apply the following coupling between $\text{Resolve}_{\infty}^{\mathcal{S}}(\text{pred}_v^{\mathcal{S}}(0))$ and ApproxResolve-Random(v, K):

- the two processes use the same random choices for $(r_\ell)_{\ell \leq 0}$ and $(U_\ell)_{\ell \leq 0}$.

One can verify that if $\mathcal{E}_{\text{trun}}(K)$ does not occur, then the two processes $\text{Resolve}_{\infty}^{\mathcal{S}}(t)$ and ApproxResolve-Random(v, K) are coupled perfectly. By the coupling lemma and Theorem 7.2,

$$d_{\text{TV}}(Y, \mu_v) \leq \mathbf{Pr}[\mathcal{E}_{\text{trun}}(K)].$$

Note that $\mathbf{Pr}[Y = \bot] = \mathbf{Pr}[\mathcal{E}_{\text{trun}}(K)]$ and $\mu_v(\bot) = 0$. Therefore, we have

$$d_{\text{TV}}(Y, \mu_v) \geq \mathbf{Pr}[Y = \bot] - \mu_v(\bot) = \mathbf{Pr}[\mathcal{E}_{\text{trun}}(K)].$$

Combining the two inequalities proves the theorem. $\qquad\square$

7.2. Derandomising CTTP for random scan Glauber dynamics. The rest of this section strives to show how one can derandomise Algorithm 11. Specifically,

Theorem 7.4. *Let μ be a q-spin Gibbs distribution with the maximum degree of the underlying graph Δ. The distribution of the output of* ApproxResolve-Random(v, K) *can be deterministically computed in* $\text{poly}(q^K, \Delta^K)$ *time.*

Remark 7.5. The above should be treated as a generic theorem. Depending on the system being studied, one needs to bound the probability of visiting more than K vertices during the backward deduction in order to control the error introduced by truncation. Typically, for fixed q and Δ, if taking $K = O(\log n)$ ensures the desired error bound, then the running time is polynomial in n. This is exactly the error bound we need for derandomising systematic scan Glauber dynamics using CTTP. That being said, for Gibbs distributions, derandomising random scan Glauber dynamics using CTTP is no harder than systematic scan Glauber dynamics.

7.2.1. *Witness tree.* We begin with introducing the structure that we "count" on. Fix the underlying graph $G = (V, E)$ for now.

Definition 7.6 (witness tree). A *witness tree* τ is a finite rooted tree together with, on each of its vertices, a vertex label from V and a random choice label from $[q] \cup \{\bot\}$ where the children of some vertex in τ with vertex label $u \in V$ receive vertex labels from $N(u)$.

Hereinafter, we generally use the letters $u, v, w, \cdots$ to represent the vertices in the original graph, and $\alpha, \beta, \gamma \cdots$ for the vertices in the witness tree (and another combinatorial structure arising from the witness tree later).

Fix an infinite scan sequence $\mathcal{S}$ and the set of random choices $\{r_t\}_{t \in \mathbb{Z}_{\leq 0}}$ (*for the lower bound distribution*) in advance. Without loss of generality, the vertex v we start is at timestamp 0. We present the following algorithm (Algorithm 12) for constructing a witness tree from the scan sequence $\mathcal{S}$.

Algorithm 12: Glauber dynamics witness tree

Input: a graph G, a scan sequence $\mathcal{S}$ and random choices $\{r_t\}_{t \in \mathbb{Z}_{\leq 0}}$, and a size threshold K
Output: an associated witness tree τ

1 construct a single-vertex tree τ with the root labeled (v, r_0);
2 **if** $r_0 \neq \bot$ **then**
3   **return** τ and $\varnothing$;
4 $A \leftarrow N(v)$;
5 $t \leftarrow 0$;
6 **while** $A \neq \varnothing$ *and* $|\tau| < K$ **do**
7   decrease t till $u := \mathcal{S}(t) \in A$;
8      $A \leftarrow A - \{u\}$;
9   find the deepest vertex α' in τ with label $(u', \cdot)$ such that $u \in N(u')$; tie-breaking by
       lexicographical order of the vertex label;
10   insert, as a child of α', a fresh vertex with label (u, r_t);
11   **if** $r_t = \bot$ **then**
12          $A \leftarrow A \cup N(u)$;

13 **return** τ and A;

Remark 7.7 (Compare to the Moser-Tardos witness trees). The witness trees defined in Definition 7.6 and constructed by Algorithm 12 , apart from the additional random choice labels, are the same as the witness trees constructed in the analysis of the Moser-Tardos algorithm for algorithmic Lovász Local Lemma [MT10]. Moreover, Algorithm 12 resembles the procedure for constructing witness trees out of an execution log for the Moser-Tardos algorithm. This is because this specific combinatorial structure succinctly captures a "local total ordering", which is a chronological total ordering between all neighbouring vertices/events encountered in ApproxResolve-Random(v, K) or the Moser-Tardos algorithm.

The construction of the witness tree in Algorithm 12 only depends on the input graph G, the scan sequence $\mathcal{S}$ and the random choices $\{r_t\}_{t \in \mathbb{Z}_{\leq 0}}$. In other words, it is independent of the implementation of the algorithm as long as it simulates the Glauber dynamics. In our setting, the random choices are samples from the lower bound distribution μ^{LB}, where a vertex no more requires any recursion if it samples anything other than $\bot$ from μ^{LB}. The set A collects the vertices that are needed in order to deduce the correct marginal of current unresolved vertices, including the starting vertex. When the construction halts with $A = \varnothing$, the marginal of the root only depends on samples from the padding distribution for those receiving $\bot$ from the lower bound distribution. This motivates the definition of a full witness tree.

Definition 7.8. A witness tree τ constructed in Algorithm 12 is called *full*, if the set A is empty upon the termination of the algorithm.

We then collect some simple properties of the above process. First, inside the **while**-loop, one of the neighbours of the current vertex u needs to be visited in prior so that u can join the set A, and hence:

Observation 7.9. *Algorithm 12 is well-defined. Specifically, such vertex in Line 9 always exists.*

We then argue that there is no need for further tie-breaking among vertices of the same level sharing the same vertex label in Line 9.

Lemma 7.10. *No two distinct vertices α and β of the same level in τ share the same vertex label.*

Proof. Given a vertex α in the tree, let $t(\alpha)$ be the variable t in the algorithm when α is inserted. Suppose α and β receive the same vertex label u and $t(\alpha) < t(\beta) \leq 0$. As soon as β gets added in the tree, the vertex u is no more present in the set A. However, as t decreases, α joins the tree with the same vertex label. This means there must be some time $t(\alpha) < t < t(\beta)$ when the algorithm visits some u' with $u \in N(u')$, detects the random choice being $\perp$, and introduces u into the set A. Let γ be the vertex in the tree created at this step. By definition, γ has label $(u', \perp)$ and must be deeper than β in the tree. Then as the algorithm progresses to time $t(\alpha)$, any parent that α can choose must be either the same or deeper level of γ. This implies that α must be deeper than β in the tree. $\square$

Finally, note that if the tree is full, then the algorithm terminates with empty A. This implies the following observation.

Observation 7.11. *If τ is a full witness tree, then it holds that: for any vertex $v \in V$ in the original graph G, if there exists a vertex $\alpha \in \tau$ labelled $(v, \perp)$, then for every $u \in N(v)$, there must be some other vertex $\beta \in \tau$ with vertex label u that is deeper than α.*

Imagine there are three updates taking place at vertices w, w, z chronologically where $w \in N(z)$. The spin for the first update on w gets overwritten by the second one, which the afterward update on z depends on. This motivates us to extract the actual dependency out from the witness tree, which yields a directed acyclic graph (DAG). We formalise the construction of such dependency DAG as below. (In the algorithm below, as W and τ shares the same vertex set, we may use the same variable to refer to the vertex in either structures, depending on the context.)

Algorithm 13: Witness dependency graph

Input: witness tree τ with a fixed root
Output: a dependency DAG

1   $B \leftarrow \varnothing$;
2   initialise W with τ's vertex set;
3   **foreach** *vertex α labelled (v, r) in the tree such that $r = \perp$* **do**
4   **foreach** $u \in N(v)$ **do**
5   find the shallowest vertex β labelled u that is deeper than α in τ;
6   **if** *such β exists* **then**
7   add an arc $\beta \rightarrow \alpha$ into W;
8   **else**
9            $B \leftarrow B + u$;

10   **return** W and B;

The above process is deterministic; that is, the algorithm outputs a unique DAG for a given witness tree. Observe that Lemma 7.10 implies the β found in Line 5, if exists, is always unique. Moreover, such β always exists if τ is full due to Observation 7.11. This observation can be formalised and generalised as follows.

Lemma 7.12. *Suppose τ is a tree returned by Algorithm 12 with the set A. Then with input τ, the set B returned by Algorithm 13 is the same as A.*

Proof. Proof by induction. If τ is of size 1, then the lemma holds trivially. Assume this lemma holds for all witness trees of size at most $L - 1$. Let τ be a witness tree of size L. Consider the last vertex γ that is added into τ during Algorithm 12, whose label is (v, r). If $r = \perp$, then the addition of γ in Algorithm 12

updates $A \leftarrow A \cup N(v) - \{v\}$. The rule of adding γ into τ ensures that γ is below any vertex with vertex label from γ. Therefore, in Algorithm 13 any α that causes v to join B in the $L-1$ case now finds such γ, and hence v is absent in B. However, any vertex below γ does not have any vertex label in $N(v)$, causing Line 5 fails for all vertices in $N(v)$. This introduces $N(v)$ into B. Hence, A and B agree with each other. The case that $r \neq \perp$ is identical to the argument above but without introducing $N(v)$.

This lemma then follows by the principle of induction. □

A similar induction also shows that the DAG constructed captures the correct dependency introduced during the recursion of the original algorithm. Moreover, it can be shown that

Lemma 7.13. *Suppose τ is a full witness tree, and W is the DAG constructed by Algorithm 13 with input τ. Then any vertex in W with in-degree 0 has a non-$\perp$ random choice label.*

Proof. By τ is a full witness tree, the set A returned by Algorithm 12 is empty. By Lemma 7.12, Line 5 always succeeds for vertices with random choice labels $\perp$. Thus, they have in-degree at least 1. □

7.2.2. *Derandomisation.* Harnessed with the witness tree and dependency DAG, we are now able to carry out the derandomisation efficiently. Assume each vertex in the infinite scan sequence $\mathcal{S}$ is independently drawn uniformly at random, and the random choices are drawn independently subject to the lower bound distribution. Rather than enumerating over all possible scan sequences and the random choices, we turn to enumerating the witness tree defined above in order to derandomise the random scan Glauber dynamics. However, different witness trees may emerge from the process with different probabilities, and this should be coped with first.

Denote the witness tree generated by the process starting at v as τ^v. We provide an algorithm that, given a witness tree τ, computes the probability of a witness tree τ emerging, namely $\mathbf{Pr}\left[\tau^v = \tau\right]$. This can be done by the following dynamic programming.

- Boundary case: If τ consists merely the root, then enumerate over all possible states r of the lower bound distribution (including $\perp$), and set $f(\tau) = \mathbf{Pr}\left[\tau^v = \tau\right] = \mu^{\mathrm{LB}}(r)$.
- Recursion: For each tree τ of size K, suppose the probabilities of all size-$(K-1)$ and degree-Δ trees have been computed. Set $f(\tau) \leftarrow 0$. Enumerate over all its leaves and do the following:
  - Suppose the current leaf is labelled (u, r). Remove the current leaf from τ to obtain τ'.
  - Run Algorithm 13 on τ' to obtain B.
  - If $v \in B$, then accumulate the current probability by

$$f(\tau) \leftarrow f(\tau) + \mathbf{Pr}\left[\tau^v = \tau'\right] \cdot |B|^{-1} \cdot \mu^{\mathrm{LB}}(r).$$

Then let $\mathbf{Pr}\left[\tau^v = \tau\right] = f(\tau)$.

Now fix a *full* witness tree τ. Conditional independence together with Lemma 7.13 implies that conditioned on such tree generated, the marginal of the spin conditioned on such tree generated, the marginal of the spin then depends solely on the DAG generated based on τ. Thus it is left for us to compute $\mathbf{Pr}\left[\sigma(v) = i \mid \tau^v = \tau\right]$.

Recall that the algorithm needs to sample from the padding distribution for every internal vertex of τ, and the dependency is captured by the DAG generated by Algorithm 13. We then enumerate over all possible configurations of the internal nodes other than the root, which receives a fixed spin i, and compute the probability of one such configuration by a simple traversal of the DAG in topological order.

Assembling both parts, the following algorithm computes the output distribution of ApproxResolve-Random(v, K).

- Enumerate, in the order of size, *all* possible trees τ of size $\leq K$ and degree Δ with root labeled v, and for each of them, do the followings:
  - Compute $\mathbf{Pr}\left[\tau^v = \tau\right]$.
  - If τ is full, then for each spin i, compute $\mathbf{Pr}\left[\sigma(v) = i \mid \tau^v = \tau\right]$. Accumulate the marginal of spin i at v by $\mathbf{Pr}\left[\tau^v = \tau\right] \mathbf{Pr}\left[\sigma(v) = i \mid \tau^v = \tau\right]$.

7.2.3. *Running-time analysis.* We finally analyse the time efficiency of the above algorithm, and conclude Theorem 7.4. The key is to bound the number of possible subtrees of a certain size in a given graph. We need the following lemma.

Lemma 7.14 ([BCKL13, Lemma 2.1]). *Let $G = (V, E)$ be a graph with maximum degree Δ and $v \in V$ be a vertex. Then the number of subtrees of G with ℓ nodes containing v is at most $\frac{(e\Delta)^{\ell-1}}{2}$.*

The outer loop of the algorithm, namely enumerating over all possible witness trees of size $\leq K$, is done by noticing that all possible τ satisfy $|V(\tau)| \leq K$. Note that considering only the vertex labels, all possible witness trees exist as a subtree of an infinite rooted tree with root labelled v, and all $|N(u)|$ children of a node labelled u receive distinct labels from the set $N(u)$. Therefore by Lemma 7.14, there are at most $(e\Delta)^K$ possible witness trees (without distinguishing the random choice labels). We can then enumerate all possible random choice labels, resulting in at most $((q+1)e\Delta)^K$ possibilities. Hence, there are $\mathrm{poly}(q^K, \Delta^K)$ possibilities to enumerate.

For each of them, the first step where we compute $\mathbf{Pr}\,[\tau^v = \tau]$, which takes $\mathrm{poly}(q^K, \Delta^K)$ time due to dynamic programming that ranges over all possible smaller witness trees than the one being processed. The second step, computing $\mathbf{Pr}\,[\sigma(v) = i \mid \tau^v = \tau]$, requires us to enumerate all possible spins from the support of padding distribution, and each enumeration requires one topological traversal. This, in total, is also $\mathrm{poly}(q^K, \Delta^K)$. Theorem 7.4 then follows.

8. CONCLUDING REMARKS

In this paper, we propose a new framework for derandomising MCMC algorithms. We introduce a method called coupling towards the past (CTTP) for evaluating a small amount of variables in their stationary states without simulating the entire chain, which gives light-weight samplers that can draw from marginal distributions. Under strong enough marginal lower bound guarantees, CTTP terminates in logarithmic steps with high probability. This provides a direct-sum style decomposition of the randomness used in MCMC sampling: a marginal sampler that can draw one random variable using only $O(\log n)$ random bits is extracted from an existing machinery for generating n jointly distributed random variables using $O(n \log n)$ random bits. A direct consequence to this is that, derandomising such marginal sampler becomes easy in polynomial time by a brute-force enumeration of all possible random choices, which gives efficient deterministic counting algorithms via standard self-reductions.

As concrete applications, we obtain efficient deterministic approximate counting algorithms for hypergraph independent sets and hypergraph colourings, in regimes matching the state-of-the-art achieved by randomised counting/sampling algorithms.

The current work makes the first step towards the goal of derandomising general Markov chain Monte Carlo algorithms. We summarize some challenges to the current framework which may lead to interesting future directions:

- Our current CTTP method relies crucially on the marginal lower bound being non-trivial, namely Condition 4.5. This condition seems necessary for our current implementation (see Remark 4.7), and it restricts problems our method can be applied to. For example, for graph colourings, no such lower bound holds, and yet efficient deterministic algorithms exist [LSS22]. Not only that, the algorithm in [LSS22] requires $q > 2\Delta$, where q is the number of colours and Δ is the maximum degree of the graph. This is close to the condition $q > (11/6 - \varepsilon)\Delta$, where $\varepsilon \approx 10^{-5}$ is a small constant, for the best randomised algorithms [CDM$^+$19]. It would be very interesting to find alternative ways to implement CTTP without marginal lower bounds to match the bounds from other methods. One idea is to expand $\perp$ to a set of possible values, similar to bounding chains [Hub04], and yet for this modified method to be efficient it apparently requires a condition of the order $q = \Omega(\Delta^2)$.

  In fact, our CTTP method implies perfect sampling algorithms, and the bound $q = \Omega(\Delta^2)$ (for the modified implementation) matches other general purpose perfect sampling algorithms [Hub04, FGY22, AJ22] applied to graph colourings. With refined techniques specific to this problem, the bounding chain approach can be made efficient under the condition $q \geq (8/3 +$

$o(1))\Delta$ [BC20, JSS21]. It would be interesting to explore if these refined techniques can help our method as well.

- Even if marginal lower bounds exist, our method still has a constant or lower order term gap in the conditions comparing to randomised algorithms. This is true for the two applications we consider in this paper, as is for problems such as estimating the partition function of the hardcore model. In the latter problem, efficient deterministic algorithms [Wei06] work all the way up to the computational complexity transition threshold, and actually predate randomised counterparts with matching bounds [ALO20, CLV21, CFYZ21, AJK⁺22, CE22, CFYZ22]. However, none of these results use coupling techniques, which our method crucially relies on. An interesting direction is to find CTTP matching these results, and one potential lead is through finding more refined couplings.

- Our current direct-sum style proof for the MCMC sampling, effectively decomposes an $O(n \log n)$-step Markov chain to an $O(\log n)$-step marginal sampler, whose random choices are enumerable in polynomial time. It is then a challenge to derandomise those MCMC algorithms with significantly higher mixing time bounds to have, for example, low-cost marginal samplers for graph matchings or bipartite perfect matchings.

- So far, all known deterministic approximate counting algorithms suffer from running time bounds whose exponents depend on the parameters of the problem and/or the instance. It is still wide open to give a deterministic approximate counting algorithm with a polynomial running time where the exponent of the polynomial is an absolute constant, especially when the instance is close enough to the critical threshold. We hope our framework of derandomising MCMC can be combined with some classical derandomisation techniques, e.g. k-wise independence, or their variants, to make progress towards such a breakthrough.

Acknowledgement

References

[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.

[Alo91]   Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991.

[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.

[ALOV19] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *STOC*, pages 1–12. ACM, 2019.

[AMMB05] Dimitris Achlioptas, Mike Molloy, Cristopher Moore, and Frank Van Bussel. Rapid mixing for lattice colourings with fewer colours. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10012, 2005.

[Ava08]   Jean-Christophe Aval. Multivariate Fuss-Catalan numbers. *Discrete Mathematics*, 308(20):4660–4669, 2008.

[Bar16]   Alexander I. Barvinok. *Combinatorics and Complexity of Partition Functions*, volume 30 of *Algorithms and combinatorics*. Springer, 2016.

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

[BCC⁺22] Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of Markov chains: Coupling, spectral independence, and entropy factorization. In *SODA*, pages 3670–3692. SIAM, 2022.

[BCKL13] Christian Borgs, Jennifer Chayes, Jeff Kahn, and László Lovász. Left and right convergence of graphs with bounded degree. *Random Struct. Algorithms*, 42(1):1–28, 2013.

[BDK06] Magnus Bordewich, Martin E. Dyer, and Marek Karpinski. Stopping times, metrics and approximate counting. In *ICALP*, volume 4051 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2006.

[BDK08] Magnus Bordewich, Martin E. Dyer, and Marek Karpinski. Path coupling using stopping times and counting independent sets and colorings in hypergraphs. *Random Struct. Algorithms*, 32(3):375–399, 2008.

[BF87] Imre Bárány and Zoltán Füredi. Computing the volume is difficult. *Discret. Comput. Geom.*, 2:319–326, 1987.

[BGG⁺19] Ivona Bezáková, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Daniel Štefankovič. Approximation via correlation decay when strong spatial mixing fails. *SIAM J. Comput.*, 48(2):279–349, 2019.

[BGK⁺07] Mohsen Bayati, David Gamarnik, Dimitriy A. Katz, Chandra Nair, and Prasad Tetali. Simple deterministic approximation algorithms for counting matchings. In *STOC*, pages 122–127. ACM, 2007.

[BRY20] Amartya Shankha Biswas, Ronitt Rubinfeld, and Anak Yodpinyanee. Local access to huge random objects through partial sampling. In *ITCS*, volume 151 of *LIPIcs*, pages 27:1–27:65. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[CDFS10] Joshua N. Cooper, Benjamin Doerr, Tobias Friedrich, and Joel Spencer. Deterministic random walks on regular trees. *Random Struct. Algorithms*, 37(3):353–366, 2010.

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

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

[CGM21] Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-Sobolev inequalities for strongly log-concave distributions. *Ann. Probab.*, 49(1):506–525, 2021.

[CGŠV21] Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Rapid mixing for colorings via spectral independence. In *SODA*, pages 1548–1557. SIAM, 2021.

[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.

[CMT15] Pietro Caputo, Georg Menz, and Prasad Tetali. Approximate tensorization of entropy at high temperature. *Ann. Fac. Sci. Toulouse Math. (6)*, 24(4):691–716, 2015.

[CS06] Joshua N. Cooper and Joel Spencer. Simulating a random walk with constant error. *Comb. Probab. Comput.*, 15(6):815–822, 2006.

[DFK91] Martin E. Dyer, Alan M. Frieze, and Ravi Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[DS14] Anindya De and Rocco A. Servedio. Efficient deterministic approximate counting for low-degree polynomial threshold functions. In *STOC*, pages 832–841. ACM, 2014.

[DSVW04] Martin E. Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Struct. Algorithms*, 24(4):461–479, 2004.

[EL75] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, Amsterdam, 1975.

[Ele86] György Elekes. A geometric inequality and the complexity of computing volume. *Discret. Comput. Geom.*, 1(4):289–292, 1986.

[FGW22] Weiming Feng, Heng Guo, and Jiaheng Wang. Improved bounds for randomly colouring simple hypergraphs. In *RANDOM*, volume 245 of *LIPIcs*, pages 25:1–25:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. (full version in arXiv:2202.05554).

[FGY22] Weiming Feng, Heng Guo, and Yitong Yin. Perfect sampling from spatial mixing. *Random Struct. Algorithms*, 61(4):678–709, 2022.

[FGYZ21a] Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Fast sampling and counting k-SAT solutions in the local lemma regime. *J. ACM*, 68(6):Art. 40, 42, 2021.

[FGYZ21b] Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Rapid mixing from spectral independence beyond the Boolean domain. In *SODA*, pages 1558–1577. SIAM, 2021.

[FHY21] Weiming Feng, Kun He, and Yitong Yin. Sampling constraint satisfaction solutions in the local lemma regime. In *STOC*, pages 1565–1578. ACM, 2021.

[FM11] Alan M. Frieze and Páll Melsted. Randomly coloring simple hypergraphs. *Inf. Process. Lett.*, 111(17):848–853, 2011.

[GGW23] Andreas Galanis, Heng Guo, and Jiaheng Wang. Inapproximability of counting hypergraph colourings. *ACM Trans. Comput. Theory*, 14(3–4), 2023.

[GK07] David Gamarnik and Dmitriy Katz. Correlation decay and deterministic FPTAS for counting list-colorings of a graph. In *SODA*, pages 1245–1254. SIAM, 2007.

[GLLZ19] Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang. Counting hypergraph colorings in the local lemma regime. *SIAM J. Comput.*, 48(4):1397–1424, 2019.

[GMP05] Leslie Ann Goldberg, Russell A. Martin, and Mike Paterson. Strong spatial mixing with fewer colors for lattice graphs. *SIAM J. Comput.*, 35(2):486–517, 2005.

[GMR13] Parikshit Gopalan, Raghu Meka, and Omer Reingold. DNF sparsification and a faster deterministic counting algorithm. *Comput. Complex.*, 22(2):275–310, 2013.

[HPR19] Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov-Sinai theory. In *STOC*, pages 1009–1020. ACM, 2019.

[HS07] Thomas P. Hayes and Alistair Sinclair. A general lower bound for mixing of single-site dynamics on graphs. *Ann. Appl. Probab.*, 17(3):931–952, 2007.

[HSS11] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):28, 2011.

[HSW21] Kun He, Xiaoming Sun, and Kewen Wu. Perfect sampling for (atomic) Lovász local lemma. *arXiv*, abs/2107.03932, 2021.

[HSZ19] Jonathan Hermon, Allan Sly, and Yumeng Zhang. Rapid mixing of hypergraph independent sets. *Random Struct. Algorithms*, 54(4):730–767, 2019.

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

[Hub04] Mark Huber. Perfect sampling using bounding chains. *Ann. Appl. Probab.*, 14(2):734–753, 2004.

[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, 2022.

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

[Jal09] Markus Jalsenius. Strong spatial mixing and rapid mixing with five colours for the Kagome lattice. *LMS Journal of Computation and Mathematics*, 12:195–227, 2009.

[JPP22] Matthew Jenssen, Aditya Potukuchi, and Will Perkins. Approximately counting independent sets in bipartite graphs via graph containers. In *SODA*, pages 499–516. SIAM, 2022.

[JPSS22] Vishesh Jain, Will Perkins, Ashwin Sah, and Mehtaab Sawhney. Approximate counting and sampling via local central limit theorems. In *STOC*, pages 1473–1486. ACM, 2022.

[JPV21a] Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. On the sampling Lovász local lemma for atomic constraint satisfaction problems. *arXiv*, abs/2102.08342, 2021.

[JPV21b] Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Towards the sampling lovász local lemma. In *FOCS*, pages 173–183. IEEE, 2021.

[JS93] Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993.

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

[JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[JVV86] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[Liu21] Kuikui Liu. From coupling to spectral independence and blackbox comparison with the down-up walk. In *RANDOM*, volume 207 of *LIPIcs*, pages 32:1–32:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[LP17] David A. Levin and Yuval Peres. *Markov chains and mixing times*. American Mathematical Soc., 2017.

[LSS22] Jingcheng Liu, Alistair Sinclair, and Piyush Srivastava. Correlation decay and partition function zeros: Algorithms and phase transitions. *SIAM J. Comput.*, 2022.

[LV91] Michael Luby and Boban Velickovic. On deterministic approximation of DNF. In *STOC*, pages 430–438. ACM, 1991.

[LVW93] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *ISTCS*, pages 18–24. IEEE, 1993.

[Moi19] Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *J. ACM*, 66(2):10:1–10:25, 2019.

[MRSV21] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil P. Vadhan. Deterministic approximation of random walks in small space. *Theory Comput.*, 17:1–35, 2021.

[MT10] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11, 2010.

[PR17] Viresh Patel and Guus Regts. Deterministic polynomial-time approximation algorithms for partition functions and graph polynomials. *SIAM J. Comput.*, 46(6):1893–1919, 2017.

[PV22] Edward Pyne and Salil P. Vadhan. Deterministic approximation of random walks via queries in graphs of unbounded size. In *SOSA*, pages 57–67. SIAM, 2022.

[PW96] James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Struct. Algorithms*, 9(1-2):223–252, 1996.

[QW22] Guoliang Qiu and Jiaheng Wang. Inapproximability of counting independent sets in linear hypergraphs. *CoRR*, abs/2212.03072, 2022.

[QWZ22] Guoliang Qiu, Yanheng Wang, and Chihao Zhang. A perfect sampler for hypergraph independent sets. In *ICALP*, volume 229 of *LIPIcs*, pages 103:1–103:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[ST19] Rocco A. Servedio and Li-Yang Tan. Pseudorandomness for read-k DNF formulas. In *SODA*, pages 621–638. SIAM, 2019.

[SYKY17] Takeharu Shiraga, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Total variation discrepancy of deterministic random walks for ergodic Markov chains. *Theor. Comput. Sci.*, 699:63–74, 2017.

[SYKY18] Takeharu Shiraga, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Deterministic random walks for rapidly mixing chains. *SIAM J. Discret. Math.*, 32(3):2180–2193, 2018.

[Wei06] Dror Weitz. Counting independent sets up to the tree threshold. In *STOC*, pages 140–149. ACM, 2006.

Appendix A. Construction of 2-block-tree

Proof of Lemma 5.24. The lemma is proved by a greedy algorithm similar to Algorithm 4 in [FGW22]. We can assume $|C| \geq \theta$ as otherwise the lemma holds trivially by setting $\ell = 0$. Let $G_C = G[C]$ be the connected subgraph of G induced by C. For any $u \in C$, we use $\Gamma_{G_C}(u)$ to denote the neighbourhood of u in G_C. For any $\Lambda \subseteq C$, define $\Gamma_{G_C}(\Lambda) = \{u \in C \setminus \Lambda \mid \exists w \in \Lambda \text{ s.t. } u \in \Gamma_{G_C}(w)\}$.

Let $\ell = 0$ and $R = C$. The algorithm repeats the following process until $R = \varnothing$:

(1) $\ell \leftarrow \ell + 1$, if $\ell = 1$, let $u = v$, if $\ell > 1$, let u be an arbitrary vertex in $\Gamma_{G_C}(C \setminus R)$;
(2) find an arbitrary connected component C_ℓ in $G[R]$ satisfying $|C_\ell| = \theta$ and $u \in C_\ell$;
(3) $R \leftarrow R \setminus (C_\ell \cup \Gamma_{G_C}(C_\ell))$;
(4) for all connected components $G' = (V', E')$ in $G[R]$ with $|V'| < \theta$, let $R \leftarrow R \setminus V'$.

Output the set $\{C_1, C_2, \ldots, C_\ell\}$.

We first show that the above algorithm is valid. The vertex u in Line (1) can be found because if $\ell = 1$, $u = v$; if $\ell > 1$, we know that $R \neq \varnothing$ and $R \neq C$ (some vertices are deleted in previous steps), since G_C is connected, $\Gamma_{G_C}(C \setminus R)$ is not empty as otherwise $R \neq \varnothing$ and $C \setminus R \neq \varnothing$ are disconnected, which contradicts with the fact G_C is connected. We then prove that the component C_ℓ can be found in Line (2). Note that $u \in R$. If $\ell = 1$, C_ℓ exists because $|C| \geq \theta$ and G_C is connected. If $\ell > 1$, by Line (4), we know that R is a set of connected components, and each component has a size at least θ, and thus C_ℓ exists.

We next show that the output is a 2-block-tree in graph G. It is straightforward to see that each C_i has size θ and is connected in graph G_C, and thus is connected in graph G. Note that after we found C_i, we remove all vertices in $\Gamma_{G_C}(C_i)$ in Line (3). Hence, for any C_i and C_j with $i \neq j$, $\mathrm{dist}_{G_C}(C_i, C_j) \geq 2$. Since $C_i, C_j \subseteq C$, it holds that $\mathrm{dist}_G(C_i, C_j) \geq 2$. Finally, fix $2 \leq i \leq \ell$. Consider the i-th iteration. Let u_i be the vertex picked in Line (1), we show that there exists $w \in C_1 \cup C_2 \cup \ldots \cup C_{i-1}$ such that $\mathrm{dist}_G(w, u_i) = 2$, which implies that $\bigcup_{j=1}^{i} C_i$ is connected on G^2. Let R denote the set R at the beginning of the i-th iteration. Since $u_i \in \Gamma_{G_C}(C \setminus R)$, we know that in graph G_C, u_ℓ has a neighbour in $v' \in C \setminus R$, which is one of the vertices deleted in Line (3) or Line (4). By Line (3), we know that for any $j < i$, when C_j is removed from R, $\Gamma_{G_C}(C_j)$ is also removed, which implies for any $j < i$, $v' \notin C_j$. Next, we show that v' cannot in any component removed in Line (4), which implies v' must in $\Gamma_{G_C}(C_j)$ for some $j < i$. Hence, there is a vertex $w \in C_j$ such that $\mathrm{dist}_{G_C}(w, u_i) = 2$. Note that both $w, u_i \in C$. It holds that $\mathrm{dist}_G(w, u_i) = 2$. Suppose $v' \in V'$ for some $|V'| < \theta$ in Line (4). Note that u_i and v' are adjacent, and $u_i \in R$ belongs to a component with size at least θ, which implies $|V'|$ also belongs to a component with size at least θ, this implies a contradiction. We remark that we proved a stronger result: any prefix $\{C_1, C_2, \ldots, C_i\}$ for $i \in [\ell]$ is a 2-block-tree in G.

Finally, we bound the size of the output 2-block-tree. In Line (3), we remove at most $\theta(\Delta+1)$ vertices. For each V' in Line (4), we claim that there exists $w \in V'$ such that w is adjacent to $\Gamma_{G_C}(C_\ell)$ in Line (3). Before the removal of $C_\ell \cup \Gamma_{G_C}(C_\ell)$, V' belongs to another component $V'' \supset V'$ such that $|V''| \geq \theta$. However, after the removal, $V' \subseteq V''$ becomes a component of size $< \theta$, which implies some vertices adjacent to V' must be removed. This proves the claim. Note that $|V'| \leq \theta - 1$. Hence the number of vertices removed in Line (4) is at most $(\theta - 1) \cdot \theta\Delta^2$. The total number of vertices removed in each iteration is at most $\theta(\Delta+1)+(\theta-1)\cdot\theta\Delta^2 \leq \theta^2\Delta^2$, where the inequality holds because $\theta \geq 1$ and $\Delta \geq 2$. Hence, the algorithm outputs a 2-block-tree with tree size $\ell \geq \lfloor |C|/(\theta^2\Delta^2) \rfloor$. If $\ell > \lfloor |C|/(\theta^2\Delta^2) \rfloor$, we just take the prefix of length $\lfloor |C|/(\theta^2\Delta^2) \rfloor$. □

Appendix B. Deterministic counting via derandomising the AJ algorithm

In this appendix, we apply the generic derandomising argument in Section 3 to the Anand-Jerrum (AJ) algorithm [AJ22], and hence provides deterministic approximate counting algorithms for spin systems on sub-exponential neighbourhood growth graphs. We start with some basic definitions.

A spin system is specified by a tuple $\mathcal{S} = (G = (V, E), [q], \boldsymbol{h}, A)$, where G is a graph, $[q]$ is the spins that each vertex may take, $\boldsymbol{h} \in \mathbb{R}_{\geq 0}^q$ is the *external field* of the system, and $A \in \mathbb{R}_{\geq 0}^{q \times q}$ is the *interaction matrix*. A *configuration* $\sigma \in [q]^V$ assigns each vertex a spin amongst $[q]$. The *weight* of

each configuration σ is defined by

$$w(\sigma) := \prod_{v \in V} \boldsymbol{h}(\sigma_v) \prod_{(u,v) \in E} \boldsymbol{A}(\sigma_u, \sigma_v).$$

The *partition function* of the system $Z(\mathcal{S})$ is the sum of weight over all possible configurations, and the *Gibbs distribution* μ is defined where the probability that each configuration is drawn is proportional to its weight. Namely,

$$Z(\mathcal{S}) := \sum_{\sigma \in [q]^V} w(\sigma) \qquad \text{and} \qquad \mu(\sigma) := \frac{w(\sigma)}{Z(\mathcal{S})}.$$

The spin system captures a lot of other counting problems. For example, when the external field is an all-1 vector and the interaction matrix has 0's on the diagonal and 1's off the diagonal, the partition function counts the number of proper q-colourings of the underlying graph.

It is useful to consider the same spin system but with some vertices pinned to take some certain spins. A configuration $\sigma \in [q]^V$ is called feasible if $w(\sigma) > 0$. For $\Lambda \subset V$, we say a *partial configuration* τ_Λ over the subset Λ is feasible, if it can be extended to a feasible configuration by assigning all other vertices with the spins. Denote by μ^{σ_Λ} the distribution over $[q]^V$ obtained from the Gibbs distribution μ conditional on the partial configuration σ_Λ. For any $S \subseteq V$, denote by $\mu_S^{\sigma_\Lambda}$ the marginal distribution on S projected from μ^{σ_Λ}. If $S = \{v\}$ is a singleton, we abbreviate the notation a bit as $\mu_v^{\sigma_\Lambda}$ for simplicity.

Definition B.1 (marginal lower bound). A distribution is said to take a marginal lower bound b, if for any $\Lambda \subset V$, $v \notin \Lambda$, any feasible partial configuration σ_Λ on Λ, and any spin $i \in [q]$, it holds that

$$\text{either} \qquad \mu_v^{\sigma_\Lambda}(i) \geq b \qquad \text{or} \qquad \mu_v^{\sigma_\Lambda}(i) = 0.$$

To give a precise characterisation where the deterministic counting algorithm works, we need the definition of the *strong spatial mixing*, a notion that is of great interest in the study of spin systems.

Definition B.2 (strong spatial mixing). Let $\delta : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a non-increasing function. A spin system $\mathcal{S} = (G = (V, E), [q], \boldsymbol{h}, \boldsymbol{A})$ is said to exhibits *strong spatial mixing* with rate δ, if for any vertex subset $\Lambda \subseteq V$, any other vertex $v \notin \Lambda$, and any two feasible partial configurations $\sigma, \tau \in [q]^\Lambda$ with $\ell = \min\{\text{dist}_G(u, v) \mid u \in \Lambda \text{ and } \sigma_u \neq \tau_u\}$,

$$d_{\text{TV}}\left(\mu_v^\sigma, \mu_v^\tau\right) \leq \delta(\ell).$$

In particular, if $\delta(\ell) = \alpha \exp(-\beta\ell)$ for some constants $\alpha, \beta > 0$, the spin system $\mathcal{S}$ then is said to exhibit the strong spatial mixing with exponential decay.

In the context of this paper, the decay rate is always exponential. We shall abbreviate the term "strong spatial mixing with exponential decay" just as SSM.

We also need some assumptions on the graph. Let $G = (V, E)$ be a graph. For any vertex $v \in V$, any integer $\ell > 0$, let

$$S_\ell(v) = \{u \in V \mid \text{dist}_G(u, v) = \ell\}$$

be the *sphere* of radius ℓ centred at v, where $\text{dist}_G(u, v)$ denotes the length of the shortest path between u and v in the graph G.

Definition B.3 (sub-exponential neighbourhood growth). Let $s : \mathbb{N} \to \mathbb{N}$ be a sub-exponential growth function, namely, satisfying $s(\ell) = \exp(o(\ell))$.[4] A graph $G = (V, E)$ is said to have *sub-exponential neighbourhood growth* (SENG) within rate s, if $|S_\ell(v)| \leq s(\ell)$ for any vertex v and any integer $\ell > 0$.

We remark that the maximum degree of a SENG graph G is $\Delta \leq s(1) = O(1)$. In statistical mechanism, SENG graphs are ubiquitous; for example, the d-dimensional integer lattice $\mathbb{Z}^d$ is heavily studied. We now state the result for spin systems.

Condition B.4. *A tuple (q, δ, s), where $\delta(\ell)$ and $s(\ell)$ are two functions, is said to satisfy this condition with constant $L = L(q, \delta, s)$, if*

(42) $$2eq(1 + s(\ell))\delta(\ell) \leq 1 \qquad \text{holds for all } \ell \geq L.$$

---

[4]The notation $s(\ell) = \exp(o(\ell))$ means for any $c > 1$, there exists an integer N such that for all $\ell \geq N$, $s(\ell) < c^\ell$.

Theorem B.5. *There exists a deterministic algorithm such that, for any $q \geq 2$, δ and s satisfying Condition B.4 with constant L, and for any q-spin system on SENG graph with growth rate s exhibiting SSM with decay rate δ, the algorithm outputs an estimate to the partition function of the spin system with multiplicative error $(1 \pm \varepsilon)$ in time $O((\frac{n}{b\varepsilon})^{O(s(L)\log q)})$, where b is the marginal lower bound.*

In practice, the interaction matrix and the external field are fixed and considered as constants, so that the actual input only contains the graph itself. With the assumption of SENG graphs, the marginal lower bound is a constant $b = b(\boldsymbol{h}, \boldsymbol{A}, s(1))$ independent of the input graph.

B.1. **Implications of Theorem B.5.** As one of the important corollaries of Theorem B.5, we provide an FPTAS for the number of (proper) colourings in SENG graphs.

Theorem B.6. *Let $q \geq 3$ and $\Delta \geq 3$ be two constants satisfying $q \geq (\frac{11}{6} - \varepsilon_0)\Delta$, where $\varepsilon_0 > 10^{-5}$ is a universal constant. There is an FPTAS for the number of proper q-colourings in SENG graphs with maximum degree at most Δ.*

Prior to this work, an FPTAS for the number of proper q-colourings on a graph of maximum degree Δ is known only when $q \geq 2\Delta$ [LSS22]. This result pushes forward the condition where FPTAS exists to that for FPRAS [CDM+19], albeit only on SENG graphs.

Specialised to colouring lattices graphs, one can prove SSM in a more refined regime than applying general theorems for colourings. The known SSM results, together with our main theorem for spin systems Theorem B.5, imply the following theorem.

Theorem B.7. *There is an FPTAS for the number of proper q-vertex-colourings for any finite subgraph of the following lattice graphs:*

- $\mathbb{Z}^2$ *lattice, if $q \geq 6$;*   *(cf. [AMMB05])*
- $\mathbb{Z}^3$ *lattice, if $q \geq 10$;*   *(cf. [GMP05])*
- *Triangular lattice, if $q \geq 10$;*   *(cf. [GMP05])*
- *Honeycomb lattice, if $q \geq 5$;*   *(cf. [GMP05])*
- *Kagome lattice, if $q \geq 5$.*   *(cf. [Jal09])*

Algorithm 14: SSMS-Truncated$_T$ $(\mathcal{S}, (\Sigma, \sigma), v, \ell)$

Input: a spin system $\mathcal{S} = (G, [q], \boldsymbol{h}, \boldsymbol{A})$, a set of vertices $\Sigma \subseteq V$ with a configuration $\sigma \in \Omega_\Sigma$, a vertex to sample $v \notin \Sigma$, and a distance $\ell \in \mathbb{N}$

Output: the partial configuration passed in with a spin at v: $(\Sigma, \sigma) \oplus (v, i)$ for some $i \in [q]$.

1 try :

2   Decrease the global timer $T \leftarrow T - 1$;

3   **for** $i \in [q]$ **do**

4 $p_v^i \leftarrow \min_{\tau \in \Omega_{S_\ell \setminus \Sigma}} \mu^{\sigma \oplus \tau}(i)$;

5 $p_v^0 \leftarrow 1 - \sum_{i \in [q]} p_v^i$;

6   Sample a random value $X \in \{0, 1, \ldots, q\}$ with $\mathbf{Pr}[X = i] = p_v^i$ for each $0 \leq i \leq q$;

7   **if** $X = 0$ **then**

8   **if** $T > 0$ **then**

9 $(\rho_1, \rho_2, \ldots, \rho_q) \leftarrow$ BD-SPLIT-Truncated$(\mathcal{S}, (\Sigma, \sigma), v, \ell, (p_v^0, p_v^1, p_v^2, \ldots, p_v^q))$;

10   Sample a random value $Y \in [q]$ with $\mathbf{Pr}[Y = i] = \rho_i$ for each $1 \leq i \leq q$;

11   **return** $((\Sigma, \sigma) \oplus (v, Y))$;

12   **else**

13   **return** $((\Sigma, \sigma) \oplus (v, X))$;

14 catch $T < 0$ **:**

15   Terminate all instances of SSMS-Truncated and BD-SPLIT-Truncated routines and set the vertex the first call is on to a random feasible colour;

Algorithm 15: BD-SPLIT-Truncated$(\mathcal{S}, (\Sigma, \sigma), v, \ell, (p_v^0, p_v^1, p_v^2, \ldots, p_v^q))$

Input: a spin system $\mathcal{S} = (G, [q], \boldsymbol{h}, \boldsymbol{A})$, a set of vertices $\Sigma \subseteq V$ with a configuration $\sigma \in \Omega_\Sigma$, a vertex to sample $v \notin \Sigma$, a distance $\ell \in \mathbb{N}$, and a probability distribution $(p_v^0, p_v^1, p_v^2, \ldots, p_v^q)$

Output: a probability distribution $(\rho_1, \rho_2, \ldots, \rho_q)$

1 Give $S_\ell(v) \setminus \Sigma$ an arbitrary ordering $S_\ell(v) \setminus \Sigma = \{w_1, w_2, \ldots, w_m\}$;

2 $(\Sigma', \sigma') \leftarrow (\Sigma, \sigma)$;

3 **for** $1 \le j \le m$ **do**

4     $(\Sigma', \sigma') \leftarrow$ SSMS-Truncated$(\mathcal{S}, (\Sigma', \sigma'), w_j, \ell)$;

5 **for** $i \in [q]$ **do**

6     $\rho_i \leftarrow (\mu_v^{\sigma'}(i) - p_v^i)/p_v^0$;

7 **return** $(\rho_1, \rho_2, \ldots, \rho_q)$;

B.2. **Truncated AJ algorithm: proof of Theorem B.5.** To derandomise the AJ algorithm, we truncate it à la Section 4.3. This results in Algorithm 14 and Algorithm 15. The algorithm maintains a global timer T shared by all copies of the process. We use a subscript to represent what the timer is initially set to, e.g., SSMS-Truncated$_T$ or SSMS-Truncated$_\infty$. For simplicity in notation, as the input spin system $\mathcal{S}$ and the parameter ℓ are never changed throughout the algorithm, we drop it from the list of parameters.

We remark that this is slightly different from the "bounded" variant appeared in [AJ22] as that truncates the algorithm by a certain depth instead of the number of calls. The correctness of the untruncated algorithm is summarised as follows.

Theorem B.8 ([AJ22, Theorem 5.3]). *Let $\mathcal{S}$ be a spin system exhibiting SSM, μ be its Gibbs distribution, and $\ell \ge 1$ be an integer. If the untruncated AJ algorithm, i.e., SSMS-Truncated$_\infty((\Sigma, \sigma), v)$ with the global timer set to ∞, terminates with probability 1, then it generates a spin of v subject to the correct conditional distribution upon terminating, i.e.,*

$$\mathbf{Pr}\left[\text{SSMS-Truncated}_\infty((\Sigma, \sigma), v) = i\right] = \mu_v^\sigma(i),$$

providing the partial configuration (Σ, σ) is feasible.

As a corollary of the above theorem, the truncated variant outputs a spin close to the correct distribution, the distance of which depends on the probability that Line 15 is executed.

Corollary B.9. *Let $\mathcal{S}$ be a spin system exhibiting SSM, μ be its Gibbs distribution, and $T, \ell \ge 1$ be integers. If the untruncated AJ algorithm SSMS-Truncated$_\infty((\Sigma, \sigma), v)$ terminates with probability 1. Then, suppose Y is the distribution of the output of SSMS-Truncated$_T((\Sigma, \sigma), v)$ with the global timer initially set to T, it holds that*

$$d_{\mathrm{TV}}\left(Y, \mu_v^\sigma\right) \le \mathbf{Pr}\left[\text{Line 15 is executed}\right].$$

Proof. For simplicity, let $\mathcal{A}$ be the truncated algorithm SSMS-Truncated$_T((\Sigma, \sigma), v)$ and $\mathcal{A}'$ be the untruncated algorithm SSMS-Truncated$_\infty((\Sigma, \sigma), v)$. Denote the bad event that Line 15 is executed by $\mathcal{B}$. For any spin $j \in [q]$, by Theorem B.8,

$$\mu_v^\sigma(j) = \mathbf{Pr}\left[\mathcal{A}' = j \mid \neg\mathcal{B}\right] \mathbf{Pr}\left[\neg\mathcal{B}\right] + \mathbf{Pr}\left[\mathcal{A}' = j \mid \mathcal{B}\right] \mathbf{Pr}\left[\mathcal{B}\right],$$

and by definition,

$$Y(j) = \mathbf{Pr}\left[\mathcal{A} = j \mid \neg\mathcal{B}\right] \mathbf{Pr}\left[\neg\mathcal{B}\right] + \mathbf{Pr}\left[\mathcal{A} = j \mid \mathcal{B}\right] \mathbf{Pr}\left[\mathcal{B}\right].$$

Naturally, $\mathbf{Pr}\left[\mathcal{A}' = j \mid \neg\mathcal{B}\right] = \mathbf{Pr}\left[\mathcal{A} = j \mid \neg\mathcal{B}\right]$, which gives

$$\left|\mu_v^\sigma(j) - Y(j)\right| = \mathbf{Pr}\left[\mathcal{B}\right] \cdot \left|\mathbf{Pr}\left[\mathcal{A}' = j \mid \mathcal{B}\right] - \mathbf{Pr}\left[\mathcal{A} = j \mid \mathcal{B}\right]\right|$$

$$\le \mathbf{Pr}\left[\mathcal{B}\right] \cdot \left(\mathbf{Pr}\left[\mathcal{A}' = j \mid \mathcal{B}\right] + \mathbf{Pr}\left[\mathcal{A} = j \mid \mathcal{B}\right]\right)$$

Therefore,

$$d_{\text{TV}}\left(Y, \mu_v^\sigma\right) = \frac{1}{2} \sum_{j \in [q]} \left|\mu_v^\sigma(j) - Y(j)\right| \le \mathbf{Pr}\left[\mathcal{B}\right]. \qquad \square$$

To analyse the probability of truncation, we treat the algorithm as a branching process. There are two possible scenarios each time when an iteration of SSMS-Truncated is invoked: either this iteration vanishes without invoking any iteration, or the algorithm calls BD-SPLIT-Truncated which leads to the creation of at most Δ^ℓ new copies of SSMS-Truncated. With the presence of strong spatial mixing, the branching process is likely to terminate in finite time, which is captured by the following lemma.

Lemma B.10. *Suppose* (q, δ, s) *is a tuple satisfying Condition B.4 with constant L. Let* $0 < \varepsilon' < 1$ *be a real, and* $T = \frac{1+s(L)}{\log 2} \cdot \log \frac{2}{\varepsilon'}$. *Given a q-spin system* $\mathcal{S} = (G, [q], \boldsymbol{h}, \boldsymbol{A})$ *as an input, where* $\mathcal{S}$ *exhibits SSM with decay rate* δ *and G is a SENG graph with rate s, the probability that Line 15 in* SSMS-Truncated$_T\left((\Sigma, \sigma), v\right)$ *is executed is bounded from above by* ε' *if the global timer is initially set to T.*

One of the main ingredients for the above argument is to bound the length of ZOI, which is also the probability for branching. This is done by the following lemma [AJ22, Lemma 4.2].

Lemma B.11. *Let* $\mathcal{S}$ *be a q-spin system exhibiting SSM with rate* δ. *In the execution of* SSMS-Truncated$_T\left((\Sigma, \sigma), v\right)$ *where the partial assignment* σ *over* Σ *is feasible, it holds that*

$$p_v^0 \le q \cdot \delta(\ell).$$

Given the exponential tail bound, Theorem B.5 can be proved.

Proof of Theorem B.5. Each random number drawn in the algorithm can take $\{0, 1, \cdots, q\}$, and hence the domain is of size $q + 1$. The worst case running time is $O(qT)$ which is straightforward to show; note that the cost of Line 4 in BD-SPLIT-Truncated is amortised into each subroutine. The number of random numbers drawn throughout the process is at most $2T$. The conditions in Corollary 3.3 get fulfilled if we set $\varepsilon' = \frac{b\varepsilon}{10n}$ where b is the marginal lower bound, which gives $T = \frac{1+s(L)}{\log 2} \cdot \log \frac{20n}{b\varepsilon}$. The theorem then follows by applying Corollary 3.3. $\qquad \square$

B.3. **Exponential tail on running time: proof of Lemma B.10.** In this section we give a proof of Lemma B.10. Each time the algorithm recurses into BD-SPLIT-Truncated, it creates at most $s(\ell)$ new copies of the routine SSMS-Truncated. Such branching happens with probability p_v^0 which is at most $q \cdot \delta(\ell)$ due to Lemma B.11. This leads us to analysing a Markov process that stochastically dominates the actual branching process.

Consider the following discrete Markov chain (X_t) where $X_t \in \mathbb{Z}_{\ge 0}$. At the beginning, $X_0 = 1$. The chain has an absorbing barrier at 0, and for other $X_t > 0$, the transition is given by

$$(43) \qquad\qquad X_{t+1} \leftarrow \begin{cases} X_t + D & \text{with probability } p; \\ X_t - 1 & \text{with probability } 1 - p. \end{cases}$$

Intuitively, if $pD < 0.99$, in expectation the random walk moves towards the absorbing barrier, and the process terminates in constant time. But for analysing the truncated algorithm, a tail bound on the event that the process does not terminate for a long time is required. This is shown as the next lemma.

Lemma B.12. *Suppose* $2e(1 + D)p \le 1$ *and* $0 < \varepsilon' < 1$. *Let* $T = \frac{1+D}{\log 2} \cdot \log \frac{2}{\varepsilon'}$. *With probability at most* ε', *the process* (X_t) *defined by (43) does not terminate in T rounds.*

We need to use the following notion of the generalised Dyck path.

Definition B.13 ($(D + 1)$-Dyck path). A sequence $a_1, a_2, \cdots, a_k \in \{+D, -1\}$ forms a $(D + 1)$-Dyck path of length k, if

$$\forall j, \sum_{i=1}^{j} a_i \ge 0 \qquad \text{and} \qquad \sum_{i=1}^{k} a_i = 0.$$

The number of $(D + 1)$ Dyck path of given length is known to be the Fuss-Catalan number (see, for example, [Ava08]).

Lemma B.14. *The number of $(D + 1)$-Dyck paths of length $(D + 1)N$ is $\frac{1}{DN+1}\binom{(D+1)N}{N}$.*

Proof of Lemma B.12. If the process (X_t) terminates exactly at $(T + 1)$-th round, namely $X_T = 1$ and $X_{T+1} = 0$, then it must hold that the sequence $\{X_{t+1} - X_t\}_{t=0}^{T-1}$ forms a dyke path, and $X_{T+1} = 0$. Obviously T is a multiple of $D + 1$. Let $N := T/(D + 1)$. Then there are N "move-ups", each with probability p, and $DN + 1$ "move-downs", each with probability $1 - p$. The extra plus one is owing to the move $X_T = 1$ to $X_{T+1} = 0$. Using the count on $(D + 1)$-Dyck paths of length $T = (D + 1)N$, the probability that (X_t) terminates at $(T + 1)$-th round is exactly

$$w_N = \frac{1}{DN + 1}\binom{(D + 1)N}{N}p^N(1 - p)^{DN+1}.$$

Using the inequality $\binom{n}{k} \leq (en/k)^k$, we have

$$w_N \leq \frac{1}{DN + 1} \cdot (e(1 + D))^N \cdot p^N(1 - p)^{DN+1} \leq (e(1 + D)p)^N.$$

Let $\gamma := e(1 + D)p$ which is at most $1/2$. The probability that the process does not terminate in $T = (D + 1)N$ rounds is

$$\mathbf{Pr}\left[(X_t) \text{ does not terminate in } (D + 1)N \text{ rounds}\right] = \sum_{i=N}^{\infty} w_i \leq \gamma^N \sum_{j=0}^{\infty} \gamma^j < 2\gamma^N < 2\left(\frac{1}{\varepsilon'}\right)^{\frac{\log(\gamma)}{\log 2}} < \varepsilon'.$$

$\square$

Proof of Lemma B.10. Let $p = q \cdot \delta(\ell)$ and $D = s(\ell)$ Suppose we are now running SSMS-Truncated$_\infty$ with the timer set to infinity. We might alter Line 4 in BD-SPLIT-Truncated a bit, by registering first the m instances of SSMS-Truncated to run, and then setting the appropriate parameter and invoking the routine. We call an instance of SSMS-Truncated *active*, if it is running or has been registered but not yet run. Let Y_t be the number of active instances upon the invocation of the $(t + 1)$-th instance. If there is no such $(t + 1)$-th instance, set $Y_t = 0$. For each instance, it dies out and becomes no more active if it does not recurse into BD-SPLIT-Truncated, and upon the invocation of the next instance, the count goes down by one. Or otherwise, it registers up to $S_\ell(v)$ new instances, so that prior to the next invocation (the current instance has not died out yet) the count goes up by the number of newly-created instances. This happens with probability $p_v^0 \leq p$, the length of ZOI.

Despite the fact that the i-th invocation may change the boundary condition for later invocations, Lemma B.11 holds for all boundary conditions, and because each invocation use fresh random numbers, the probability that x invocations create copies and y invocations do not is bounded by $p^x(1 - p)^y$. Then it is a simple observation that

$$\mathbf{Pr}\left[(Y_t) \text{ does not terminate in } T \text{ rounds}\right] \leq \mathbf{Pr}\left[(X_t) \text{ does not terminate in } T \text{ rounds}\right].$$

This is because each time it branches out $D' \leq D$ new instances. We can just treat this as if it created D copies, but the last $D - D'$ ones were bound to die out. The lemma then follows by applying Lemma B.12 with the aforementioned choices of p, D and T. $\square$

B.4. **FPTAS from optimal temporal mixing.** As a well-known result, the notion of SSM is equivalent to *optimal temporal mixing* of the Glauber dynamics on SENG graphs [DSVW04]. The latter notion receives a glaring attention in recent study of Markov chains.

Fix a subset of vertices $\Lambda \subseteq V$. Define its boundary by

$$\partial\Lambda := \{u \in V \setminus \Lambda \mid \exists w \in \Lambda \text{ s.t. } \{u, w\} \in E\}.$$

Let $(X_t)_{t\geq 0}$ and $(Y_t)_{t\geq 0}$ be two instance of the Glauber dynamics, where $(X_t)_{t\geq 0}$ and $(Y_t)_{t\geq 0}$ may start from different initial configurations X_0 and Y_0. The optimal temporal mixing is defined as follows.

Definition B.15 (optimal temporal mixing). The Glauber dynamics is said to have the optimal temporal mixing under arbitrary pinning, if there exists $\gamma, \zeta > 0$ such that for any vertex set $\Lambda \subseteq V$, any feasible boundary condition $\sigma \in [q]^{\partial\Lambda}$, and any two instances $(X_t)_{t\geq 0}$ and $(Y_t)_{t\geq 0}$ of the Glauber dynamics on Λ with boundary configuration σ, it holds that

$$\forall k \in \mathbb{N}, \quad d_{\mathrm{TV}}(X_{kn}, Y_{kn}) \leq |\Lambda| \gamma \exp(-\zeta k).$$

The above definition also implies $O(n \log n)$ mixing time of the Glauber dynamics, by observing that starting from an arbitrary X_0, $d_{\mathrm{TV}}(X_T, \mu_\Lambda^\sigma) \leq 1/4$ for $T = O(n \log n)$.

One of the main results in [DSVW04] states:

Theorem B.16 ([DSVW04, Theorem 2.3]). *If the spin system poses optimal temporal mixing on SENG graphs, then the system exhibits SSM.*

We remark that the definition of SSM varies slightly across the literature. However, the version in this paper is equivalent to that in [DSVW04] on SENG graphs, and the above theorem still applies in our context. Combining Theorem B.5 and Theorem B.16 yields:

Theorem B.17. *Let $A \in \mathbb{R}_{\geq 0}^{q \times q}$ and $\boldsymbol{h}_{\geq 0}^q$ be an interaction matrix and an external field vector. There is an FPTAS for the partition functions of the spin system defined by A and $\boldsymbol{h}$ on SENG graphs, if the Glauber dynamics on the Gibbs distribution has the optimal temporal mixing under arbitrary pinning.*

B.5. **FPTAS from spectral independence.** To derive optimal temporal mixing and hence apply Theorem B.17, we utilise a powerful tool called the *spectral independence*, first defined by Anari, Liu and Oveis Gharan [ALO20] to obtain rapid mixing, extended to general $[q]$ domains by various authors [CGŠV21, FGYZ21b], and refined by Chen, Liu and Vigoda [CLV21] for optimal mixing of the Glauber dynamics.

The formal definition of spectral independence is given below. For any subset $\Lambda \subset V$, any $\sigma \in \Omega(\mu_\Lambda)$, where $\Omega(\mu_\Lambda)$ denotes the support of the distribution μ_Λ, define

$$\widetilde{V}_\sigma = \left\{ (u, c) \mid u \in V \setminus \Lambda \text{ and } c \in \Omega(\mu_v^\sigma) \right\}.$$

For every pair $(u, i), (v, j) \in \widetilde{V}_\sigma$ with $u \neq v$, define the (signed) influence from (u, i) to (v, j) with respect to the conditional σ by

$$\gamma_\mu^\sigma((u, i), (v, j)) = \mu_v^{\sigma \wedge (u \leftarrow i)}(j) - \mu_v^\sigma(j),$$

and define $\gamma_\mu^\sigma((v, i), (v, j)) = 0$ for all $(v, i), (v, j) \in \widetilde{V}_\sigma$, where $\mu_v^{\sigma \wedge (u \leftarrow i)}$ denotes the marginal distribution on v conditional on σ and the event that u takes the value i.

Definition B.18 (spectral independence). Let $\eta > 0$ be a constant. A distribution μ is η-spectrally independent if for any $\Lambda \subset V$, any $\sigma \in \Omega(\mu_\Lambda)$, the maximum eigenvalue of γ_μ^σ satisfies $\lambda_{\max}(\gamma_\mu^\sigma) \leq \eta$.

The main result of this subsection is the following theorem.

Theorem B.19. *Let $A \in \mathbb{R}_{\geq 0}^{q \times q}$ and $\boldsymbol{h}_{\geq 0}^q$ be an interaction matrix and an external field vector. There is an FPTAS for the partition functions of the spin system defined by A and $\boldsymbol{h}$ on SENG graphs, if the Gibbs distribution of the spin system is η-spectrally independent for some constant η.*

The work [CLV21] establishes that, under some conditions, spectral independence implies approximate tensorisation of entropy, a notion that is used to establish the decay of relative entropy and hence optimal temporal mixing (see, for example, [CMT15]). Connecting the main theorem of [CLV21, Theorem 2.8] (see also [BCC+22, Theorem 1.7]) with [CMT15] yields the following.

Theorem B.20. *Let $\Delta \geq 3$ be an integer and $b, \eta > 0$ be reals. Let μ be the distribution of a q-spin system on a graph $G = (V, E)$ of maximum degree at most Δ. If μ is η-spectrally independent and b-marginally bounded, then the relative entropy of P_{GL} decays with rate $\frac{1}{C|V|}$, i.e.,*

$$D_{\mathrm{KL}}(\nu P_{\mathrm{GL}} \parallel \mu) \leq \left(1 - \frac{1}{C|V|}\right) D_{\mathrm{KL}}(\nu \parallel \mu)$$

holds for any distribution ν over $[q]^V$. Here the constant $C = \left(\frac{\Delta}{b}\right)^{1+2\lceil \frac{\eta}{b} \rceil}$.

Theorem B.19 can be proved by iterating the above theorem. We also remark that spectral independence holds under arbitrary pinning.

Proof of Theorem B.19. Fix the vertex set Λ. Under arbitrary feasible pinning, we apply Theorem B.20 iteratively. The distribution X_{kn} of the Glauber dynamics on Λ after kn rounds then satisfies

$$D_{\text{KL}}\left(X_{kn} \parallel \mu_\Lambda^\sigma\right) \le \left(1 - \frac{1}{C|\Lambda|}\right)^{kn} D_{\text{KL}}\left(X_0 \parallel \mu_\Lambda^\sigma\right)$$

(Using $|\Lambda| \le n$)
$$\le \exp\left\{-\frac{k}{C}\right\} D_{\text{KL}}\left(X_0 \parallel \mu_\Lambda^\sigma\right)$$

(44)
$$\le \exp\left\{-\frac{k}{C}\right\} \log\left(\frac{1}{\mu_{\Lambda,\min}^\sigma}\right)$$

where $\mu_{\Lambda,\min}^\sigma = \min_{\tau \in [q]^\Lambda} \mu_\Lambda^\sigma(\tau)$ is the minimum non-zero probability of the distribution μ_Λ^σ.

We verify optimal temporal mixing as follows.

$$d_{\text{TV}}\left(X_{kn}, Y_{kn}\right) \le d_{\text{TV}}\left(X_{kn}, \mu_\Lambda^\sigma\right) + d_{\text{TV}}\left(Y_{kn}, \mu_\Lambda^\sigma\right)$$

(w.l.o.g.)
$$\le 2 d_{\text{TV}}\left(X_{kn}, \mu_\Lambda^\sigma\right)$$

(Pinsker's Inequality)
$$\le \sqrt{2 D_{\text{KL}}\left(X_{kn} \parallel \mu_\Lambda^\sigma\right)}$$

(By (44))
$$\le \sqrt{2}\sqrt{\log\left(\frac{1}{\mu_{\Lambda,\min}^\sigma}\right) \exp\left\{-\frac{k}{2C}\right\}}$$

$$\le \sqrt{2|\Lambda| \log(1/b)} \exp\left\{-\frac{k}{2C}\right\}.$$

The theorem then follows by invoking Theorem B.17. $\qquad\square$

Finally, Theorem B.6 can be proved.

Proof of Theorem B.6. The uniform distribution over proper q-colourings in the same regime as Theorem B.6 is shown to be η-spectrally independent for some constant $\eta = \eta(q, \Delta)$ [Liu21]. This theorem then follows after Theorem B.19. $\qquad\square$