Fast and perfect sampling of subgraphs and polymer systems

Antonio Blanca* Sarah Cannon† Will Perkins‡

November 17, 2023

Abstract

We give an efficient perfect sampling algorithm for weighted, connected induced subgraphs (or *graphlets*) of rooted, bounded degree graphs. Our algorithm utilizes a vertex-percolation process with a carefully chosen rejection filter and works under a percolation subcriticality condition. We show that this condition is optimal in the sense that the task of (approximately) sampling weighted rooted graphlets becomes impossible in finite expected time for infinite graphs and intractable for finite graphs when the condition does not hold. We apply our sampling algorithm as a subroutine to give near linear-time perfect sampling algorithms for polymer models and weighted non-rooted graphlets in finite graphs, two widely studied yet very different problems. This new perfect sampling algorithm for polymer models gives improved sampling algorithms for spin systems at low temperatures on expander graphs and unbalanced bipartite graphs, among other applications.

1 Introduction

Sampling is a fundamental computational task: given a specification of a probability distribution on a (large) set of combinatorial objects, output a random object with the specified distribution or with a distribution close to the specified distribution. This task becomes challenging when the specification of the distribution is much more succinct than the set of objects, and one wants to sample using time and space commensurate with the specification. Fundamental examples include sampling from Markov random fields and probabilistic graphical models and sampling substructures of graphs. We will address both of these examples here and connect them in a new way.

We consider a natural sampling problem: given a bounded-degree graph G, sample a *graphlet* (a connected, vertex-induced subgraph) of G containing a fixed vertex r with probability proportional to an exponential in the size of the subgraph. That is, sample a graphlet S containing vertex r with probability proportional to $\lambda^{|S|}$, where $\lambda > 0$ is a distribution parameter and $|S|$ denotes the number of vertices in S. In the paper we are concerned with small values of λ, where the expected size of a sampled graphlet is much smaller than the size of the graph.

Sampling graphlets is an important task in data science, network analysis, bioinformatics, and sociology, as it allows us to gain information about massive graphs from small sections of it; see, e.g., [KIMA04, GK07, LB12, BGP07]. A number of variants of the problem have consequently been studied, including sampling graphlets of a given size uniformly at random or sampling weighted graphlets of all sizes [SVP+09, BRRAH12, JSP15, CLWL16, BCK+17, BCK+18, PSS19, ABH19, MG20, RMŠ21, BLP21, Bre21]. The variant we consider here, i.e., sample a graphlet S with

*Department of Computer Science and Engineering, Pennsylvania State University. ablanca@cse.psu.edu.
†Department of Mathematical Sciences, Claremont McKenna College. scannon@cmc.edu.
‡Department of Mathematics, Statistics, and Computer Science, UIC. math@willperkins.org.

probability proportional to $\lambda^{|S|}$, arises as a key subroutine in recent sampling algorithms for spin systems (hard-core model, Ising model, Potts model, etc.) in the regime of strong interactions via *polymer models* described below in Section 1.2; see [HPR20, CP20, LLLM19, GGS22, BCH+20, JPP22, JKP20, CDK20a, CGŠV22].

One major limitation of previous sampling algorithms for graphlets and polymer models (those in, e.g., [HPR20, LLLM19, RMŠ21, CGG+21, GGS21], among others) is the use of exhaustive enumeration of graphlets of a given size; this requires restrictive parameter regimes or large polynomial running times, with the logarithm of the maximum degree Δ of the graph appearing in the exponent of the polynomial. Here we design a fast perfect sampling algorithm for weighted graphlets based on a vertex percolation process combined with a rejection filter. This method bypasses the enumeration barrier and allows us to design perfect sampling algorithms for a number of applications, substantially improving upon existing algorithms in three ways: 1) our algorithms have considerably faster running times, with no dependence on Δ in the exponent; 2) our algorithms return perfect, rather than approximate, samples from the desired distributions; and 3) our algorithms are conceptually simple and practical to implement.

Our algorithm proceeds as follows. First, run a vertex percolation process on the graph G beginning at vertex r in a breadth-first search manner, repeatedly adding each adjacent vertex to the graphlet with a carefully-chosen probability p. Once the percolation process terminates, the graphlet is accepted as the random sample with a certain probability that depends on the graphlet and rejected otherwise; if the graphlet is rejected, the algorithm restarts another percolation process from r. Because of the careful way we choose the percolation and rejection probabilities, we can prove the final accepted sample is drawn exactly from the desired distribution and the expected running time is bounded by a constant that depends only on λ and the maximum degree Δ.

For our applications to polymer models, we use this graphlet sampling algorithm as a subroutine to implement a Markov chain on polymer configurations. We then use this Markov chain to devise a perfect sampling algorithm for polymer models based on the coupling from the past method from [PW96] and the notion of bounding chains from [Hub04].

1.1 Sampling rooted graphlets

Our key contribution is a new algorithm for perfectly sampling weighted graphlets containing a given vertex r. We start by fixing the model of computation we work with throughout the paper. We assume a model that allows for querying the adjacency list of a given vertex in a bounded degree graph in constant time, including in a rooted infinite graph. We also assume that in a finite graph we can query a uniformly random vertex in constant time. This a standard model used in the study of sublinear algorithms [GR97]. We also assume access to a stream of perfectly random real numbers in $[0, 1]$. The model of computation is fixed for consistency; in particular, our methods extend to other models, only requiring to adjust the running time to account for any additional computational overhead.

Let $G = (V, E)$ be a finite or infinite graph of maximum degree Δ. For $r \in V$, let $\mathcal{S}(G, r)$ be the set of all connected, vertex-induced subgraphs of G containing r. (The subgraph induced by $U \subseteq V$ has vertex set U and includes all the edges of G with both endpoints in U.) We call r the root of G and the elements of $\mathcal{S}(G, r)$ graphlets rooted at r. For $\lambda > 0$ define the probability distribution $\nu_{G,r,\lambda}$ on $\mathcal{S}(G, r)$ by

$$\nu_{G,r,\lambda}(S) = \frac{\lambda^{|S|}}{Z_{G,r,\lambda}}, \quad \text{where} \quad Z_{G,r,\lambda} = \sum_{\hat{S} \in \mathcal{S}(G,r)} \lambda^{|\hat{S}|}. \tag{1}$$

The distribution is well defined when the normalizing constant $Z_{G,r,\lambda}$, known as the partition function, is finite. This is the case for every graph of maximum degree Δ and every r when λ is below the critical threshold:

$$\lambda_*(\Delta) = \frac{(\Delta-2)^{\Delta-2}}{(\Delta-1)^{\Delta-1}}; \tag{2}$$

see Lemma 1.3 below. This threshold was already considered in [RMŠ21], who provided an ε-approximate sampling algorithm for $\nu_{G,r,\lambda}$ for the class of maximum-degree Δ graphs when $\lambda < \lambda_*(\Delta)$ with running time $\mathrm{poly}(\varepsilon^{-1})$. We give a perfect sampling algorithm for $\nu_{G,r,\lambda}$ for $\lambda < \lambda_*(\Delta)$ with constant expected running time.

Theorem 1.1. *Fix $\Delta \geq 3$ and let $\lambda < \lambda_*(\Delta)$. There is a randomized algorithm that for any graph $G = (V, E)$ of maximum degree Δ and any $r \in V$ outputs a graphlet distributed according to $\nu_{G,r,\lambda}$ with expected running time bounded by a constant that depends only on Δ and λ.*

Previous algorithms to generate ε-approximate samples from $\nu_{G,r,\lambda}$ (e.g., those in [HPR20, RMŠ21, CGG+21, GGS21]) exhaustively enumerate all graphlets of size $\leq k$, for some k that depends on the error parameter ε that describes how accurate the sample must be. This results in algorithms with $(1/\varepsilon)^{O(\log \Delta)}$ running times. Applications such as sampling from polymer models require multiple samples from $\nu_{G,r,\lambda}$ and have small error tolerance per sample; in particular, they require $\varepsilon \ll 1/n$, which results in inefficient algorithms with overall running time $n^{O(\log \Delta)}$. The algorithm in Theorem 1.1, on the other hand, is an exact sampler whose expected running time depends only on Δ and λ and thus provides a significant advantage in applications as we detail below.

We also show that Theorem 1.1 is sharp in two ways. First, we establish that there is no polynomial-time approximate sampling algorithm for $\nu_{G,r,\lambda}$ when $\lambda \in (\lambda_*(\Delta), 1)$ for the class of finite graphs of maximum degree at most Δ unless RP=NP; see Definition 1 for the precise definition of a polynomial-time approximate sampler. (We note that a similar hardness result was proved in [RMŠ21] for the related problem of sampling "unrooted graphlets"; we provide more details about this in Section 1.4.) Second, in the infinite setting, the normalizing constant $Z_{G,r,\lambda}$ may diverge (and consequently the distribution $\nu_{G,r,\lambda}$ is not be well-defined) when $\lambda > \lambda_*(\Delta)$; conversely, we prove that $Z_{G,r,\lambda}$ is finite on every graph of maximum degree Δ when $\lambda \leq \lambda_*(\Delta)$.

Lemma 1.2. *Fix $\Delta \geq 3$ and $\lambda \in (\lambda_*(\Delta), 1)$. If there is a polynomial-time approximate sampler for $\nu_{G,r,\lambda}$ for finite graphs $G = (V, E)$ of maximum degree Δ and each $r \in V$, then RP=NP.*

Lemma 1.3. *The partition function $Z_{G,r,\lambda}$ is finite for every (possibly infinite) graph $G = (V, E)$ of maximum degree Δ and every $r \in V$ if and only if $\lambda \leq \lambda_*(\Delta)$.*

Finally, we mention that the algorithmic result in Theorem 1.1 cannot be extended even to the case $\lambda = \lambda_*(\Delta)$: for the infinite Δ-regular tree, we can show that the expected size of a graphlet sampled from $\nu_{G,r,\lambda}$ is infinite when $\lambda = \lambda_*(\Delta)$, and so it is impossible to have sampling algorithms with finite expected running time. In summary, the algorithm in Theorem 1.1 for $\lambda < \lambda_*(\Delta)$, combined with the hardness/impossibility results in Lemmas 1.2 and 1.3 for $\lambda > \lambda_*(\Delta)$, provide a resolution to the computational problem of sampling from $\nu_{G,r,\lambda}$ on graphs of maximum degree at most Δ.

As mentioned, our sampling algorithm is based on exploring the connected component of r in a vertex-percolation process. We carefully choose a specific percolation parameter $p \in (0, 1)$ as a function of λ and Δ (see Lemma 2.2). We then perform breadth-first search (BFS) from r, labeling each new vertex encountered 'active' with probability p and 'inactive' with probability $1 - p$ independently over all vertices; we continue the BFS exploring only unexplored neighbors of active vertices. In this way we uncover the 'active' component of r, call it γ. We then accept

γ with a given probability depending on λ, Δ, $|\gamma|$ and $|\partial\gamma|$, where $\partial\gamma$ denotes the set of vertices outside of γ that are adjacent to γ. If we reject γ, we begin again with a new percolation process. We note that only when $\lambda < \lambda_*(\Delta)$ there exists a suitable percolation probability p that results in a subcritical percolation process, so that the size of the active component has finite expectation and exponential tails. The weighted model we sample from is particularly well suited to this type of exploration algorithm because of the direct connection to a subcritical percolation process.

Random exploration and rejection sampling have been used previously to sample graphlets and other structures, most notably in the recent work of Bressan [Bre21] who uses a novel bucketing scheme in combination with rejection sampling to perfectly sample uniformly random graphlets of size k from a graph, as well as studying the mixing time of the random walk on the set of all such graphlets. See also [AJ22] in which a random growth process and rejection sampling are used to perfectly sample spin configurations.

We prove a more general version of Theorem 1.1 in Section 2, allowing for vertex-labeled graphlets and modifications of the weights by multiplication by a non-negative function bounded by 1. These generalizations are needed for the application to polymer models in Section 1.2.

1.2 Sampling from polymer models

We use our algorithm for sampling weighted rooted graphlets to design fast and perfect samplers for polymer models. Polymer models are systems of interacting geometric objects representing defects from pure ground states (i.e., most likely configurations) in spin systems on graphs in classical statistical physics [GK71, KP86, FV17]. These geometric objects are most often represented by vertex-labeled graphlets from a given host graph. Recently, polymer models have found application as an algorithmic tool to sample from spin systems on various classes of graphs in strong interaction regimes; see, e.g., [HPR20, CP20, LLLM19, CGG$^+$21, HJP22, GGS21, GGS22, BCH$^+$20, JPP22, JKP20, CDK20a, CDK$^+$20b, CGŠV22]. In these applications, the problem of sampling weighted vertex-labeled rooted graphlets emerged as a significant computational barrier.

We will work with *subset polymer models* in which all polymers are vertex-labeled graphlets from a host graph $G = (V, E)$. These models were defined in [GK71] and generalized in [KP86]. Such a polymer model consists of:

- A set $\mathcal{C} = \mathcal{C}(G)$ of polymers, each of which is a graphlet in G with the vertices of the graphlet labeled with colors from a set Σ of size q.
- Weights $w_\gamma \geq 0$ for each $\gamma \in \mathcal{C}$. We assume without loss of generality that all vertex-labeled graphlets of G, including each individual vertex $v \in V$, are elements of $\mathcal{C}$, by setting $w_\gamma = 0$ when necessary.
- An incompatibility relation $\nsim$ defined by connectivity. We say two polymers $\gamma, \gamma' \in \mathcal{C}$ are *incompatible* and write $\gamma \nsim \gamma'$ if the union of their corresponding vertices induces a connected subgraph in G. Otherwise they are *compatible* and we write $\gamma \sim \gamma'$.

Let $\Omega(\mathcal{C})$ denote the set of all sets of pairwise compatible polymers from $\mathcal{C}$. The polymer model is the Gibbs distribution μ on $\Omega(\mathcal{C})$ defined by

$$\mu(X) = \frac{\prod_{\gamma \in X} w_\gamma}{Z(\mathcal{C})}, \quad \text{where} \quad Z(\mathcal{C}) = \sum_{X \in \Omega(\mathcal{C})} \prod_{\gamma \in X} w_\gamma$$

is the polymer model partition function. The size $|\gamma|$ of a polymer γ is the number of vertices in the corresponding graphlet. We let $\mathcal{C}_v$ be all polymers containing vertex v.

Condition	Bound on exponential decay of weights	Running Time	Precision
Kotecký–Preiss [HPR20, JKP20, CP20]	$w_\gamma \le (e^2 q\Delta)^{-\lvert\gamma\rvert}$	$n^{O(\log \Delta)}$	approximate
Polymer Sampling [CGG+21, GGS21]	$w_\gamma \le (e^5 q^3 \Delta^3)^{-\lvert\gamma\rvert}$	$O(n \log n)$	approximate
Clique dynamics [FGKP23]	$w_\gamma \le (eq\Delta)^{-\lvert\gamma\rvert}$	$n^{O(\log \Delta)}$	approximate
This work (Theorem 1.4)	$w_\gamma \le (eq\Delta)^{-\lvert\gamma\rvert}$	$O(n \log n)$	perfect

Table 1: Comparison of conditions and running times of known polymer sampling algorithms.

We will often work with a family of polymer models corresponding to an infinite family of host graphs G. We say the weights of a family of polymer models are *computationally feasible* if w_γ can be computed in time polynomial in $|\gamma|$ uniformly over the polymer models in the family.

Algorithms for sampling polymer models fall into two classes: those based on truncating the cluster expansion of a polymer model to approximate a partition function and using self-reducibility to sample, and those based on Markov chains on the set of collections of compatible polymers. The cluster expansion approach, while giving polynomial-time algorithms, is relatively inefficient in general, with the degree of the polynomial bound on the running time growing with the degree of the underlying graph; e.g., running time $n^{O(\log \Delta)}$ for n-vertex graph of maximum degree Δ. A Markov chain approach based on adding and remove polymers from a polymer configuration in principle can be much faster (near linear time in the size of the graph) but runs into one hitch: a much stricter condition on the parameters of the model is needed to perform one update step of the Markov chain efficiently (the "polymer sampling condition" in [CGG+21, GGS21]). We solve this problem by adapting our rooted graphlet sampler to sample polymers models, leading to a near linear-time perfect sampling algorithm for polymer models under the least restrictive conditions known (see Table 1).

Theorem 1.4. *Fix $\Delta \ge 3$, $q \ge 1$, $\theta \in (0,1)$, and $\lambda < \lambda_*(\Delta, q) := \frac{(\Delta-2)^{\Delta-2}}{q(\Delta-1)^{\Delta-1}}$. There is a perfect sampling algorithm for μ with expected running time $O(|V| \log |V|)$ for any family of subset polymer models on maximum degree Δ graphs $G = (V, E)$ with computationally feasible weights satisfying:*

$$w_\gamma \le \lambda^{|\gamma|} \text{ for all } \gamma \in \mathcal{C}; \text{ and} \tag{3}$$

$$\sum\nolimits_{\gamma \not\sim v} |\gamma| w_\gamma \le \theta \text{ for all } v \in V. \tag{4}$$

The threshold defined in (3) is the generalization of the critical threshold for rooted graphlet sampling to the labeled case (taking $q = 1$ recovers the definition in (2)). Theorem 1.4 improves upon the known results for sampling from polymer models in two ways. For a very general class of polymer models, our algorithm simultaneously provides *perfect sampling* with *near-linear running time* under weak conditions on the polymer weights. We now review previous works to illustrate these improvements; see the accompanying Table 1.

A number of conditions on polymer weights have been used to provide efficient sampling algorithms. The first papers in this direction (including [HPR20, JKP20, CP20]) used the Kotecký–Preiss condition for convergence of the cluster expansion of the polymer model partition function [KP86]: $\sum_{\gamma' \not\sim \gamma} w_{\gamma'} e^{|\gamma'|} \le |\gamma| \quad \forall \gamma \in \mathcal{C}$. This condition is typically verified by ensuring that:

$$\sum\nolimits_{\gamma \not\sim v} w_\gamma e^{|\gamma|} \le 1 \quad \forall v \in V\,. \tag{5}$$

Since the number of vertex-labeled rooted graphlets of size k in a maximum degree Δ graph grows roughly like $(eq\Delta)^{k-1}$ (see [BCKL13]), weights of polymers of size k must decay roughly like $(e^2q\Delta)^{-k}$ for the polymer model to satisfy (5), with the extra factor of e coming from the exponential in the left hand side of the condition (5).

The major downside to the algorithms based on the cluster expansion, i.e., those using (5) or the Kotecký–Preiss condition, is that the running times obtained are of the form $n^{O(\log \Delta)}$. Subsequent works, namely [CGG+21, GGS21], addressed this downside but at the cost of a significantly stricter condition on the polymer weights.

In [CGG+21], the authors devised a new Markov chain algorithm for sampling from polymer models. The condition on the polymer weights for rapid mixing of this chain is somewhat less restrictive than the Kotecký–Preiss condition; it is the Polymer Mixing condition:

$$\sum\nolimits_{\gamma' \nsim \gamma} |\gamma'| w_{\gamma'} \leq \theta |\gamma| \quad \forall \gamma \in \mathcal{C} \text{ for some } \theta \in (0,1)\,. \tag{6}$$

This requires weights of polymers of size k to decay like $(eq\Delta)^{-k}$, a savings of a factor e in the base of the exponent over (5). However, to implement a single step of this Markov chain in constant expected time, a stronger condition (the Polymer Sampling condition) was required:

$$w_\gamma \leq \left(e^5\Delta^3 q^3\right)^{-|\gamma|}\,. \tag{7}$$

This is a significant loss of a factor $e^3\Delta^2q^2$ in the base of the exponent compared to (5), but the resulting sampling algorithm does run in near linear time.

In [FGKP23], the authors use a different Markov chain condition, the Clique Dynamics condition, similar to (6), which requires weights of polymers of size k to decay like $(eq\Delta)^{-k}$, saving the same factor e over (5). Their running times, though, are again of the form $n^{O(\log \Delta)}$ since implementing one step of their Markov chain involves enumerating rooted polymers of size $O(\log n)$.

Our results are a "best-of-both-worlds" for polymer sampling: under the conditions (3) and (4) that both require polymer weights to decay like $(eq\Delta)^{-k}$ (this is shown later; see, e.g., the proof of Corollary 1.6), we obtain a near linear time algorithm. Moreover, unlike any of the previous results, our algorithm is a perfect sampler.

To conclude this section, we comment briefly on the algorithm we design to sample from μ. Our starting point is the polymer dynamics Markov chain from [CGG+21]. We use it to implement a Coupling from the Past (CFTP) algorithm (see [PW96]). To do so efficiently (in terms of the number of steps of the Markov chain), we design a new "bounding Markov chain" for the polymer dynamics, a method pioneered in [Hub04, HN99], and to implement each step of the Markov chain efficiently, we turn to our sampler for weighted rooted graphlets from Theorem 1.1.

1.3 Applications to spin systems

Our new algorithm for sampling subset polymer models can be used as a subroutine in essentially all previous applications of polymer models for spin system sampling at low temperatures, including those in [JKP20, CP20, LLLM19, CGG+21, HJP22, GGS21, GGS22, CDK20a, CDK+20b, CGŠV22]. This results in faster sampling algorithms under less restrictive conditions on model parameters in all those settings. As examples, we fleshed out here the details in two of these applications; more details are provided in Section 4.

Hard-core model on bipartite graphs. The hard-core model on a graph G is the probability distribution μ_G^{hc} on $\mathcal{I}(G)$, the set of all independent sets of G, with

$$\mu_G^{hc}(I) = \frac{\lambda^{|I|}}{Z_G^{hc}(\lambda)}, \quad \textit{where} \quad Z_G^{hc}(\lambda) = \sum_{I \in \mathcal{I}(G)} \lambda^{|I|}\,. \tag{8}$$

The complexity of approximate counting and sampling from μ_G^{hc} on bounded-degree graphs is well understood: there is a computational threshold at some $\lambda_c(\Delta)$, with efficient algorithms for $\lambda < \lambda_c(\Delta)$ [Wei06, ALOG21, CLV20, CLV21] and hardness above the threshold (no polynomial-time algorithms unless NP=RP) [Sly10, GŠV16, SS12]. However, on bipartite graphs, the complexity of these problems is unresolved and is captured by the class #BIS (approximately counting independent sets in bipartite graphs) defined by Dyer, Goldberg, Greenhill, and Jerrum [DGGJ04].

Theorem 1.4 implies the existence of a fast perfect sampling algorithm for the hard-core model in a certain class of bipartite graphs called *unbalanced bipartite graphs*, considered in [CP20, FGKP23].

Corollary 1.5. *There is a perfect sampling algorithm for μ_G^{hc} running in expected time $O(n \log n)$ for n-vertex bipartite graphs G with bipartition (L, R), with maximum degree Δ_L in L, maximum degree Δ_R in R, and minimum degree δ_R in R if*

$$\lambda(1 + (1+e)(\Delta_L - 1)\Delta_R) < (1+\lambda)^{\delta_R/\Delta_L}. \tag{9}$$

Approximate sampling algorithms with large polynomial run times were previously given for this problem when $6\lambda\Delta_L\Delta_R < (1+\lambda)^{\delta_R/\Delta_L}$ in [CP20] and when $3.3353\lambda\Delta_L\Delta_R < (1+\lambda)^{\delta_R/\Delta_L}$ in [FGKP23]. Our result applies to a comparable parameter range: inequality (9) holds, for instance, when $(1+e)\lambda\Delta_L\Delta_R < (1+\lambda)^{\delta_R/\Delta_L}$, or when $3\lambda\Delta_L\Delta_R < (1+\lambda)^{\delta_R/\Delta_L}$ and $\Delta_L < 6$. More importantly, our algorithm is the first to achieve perfect sampling and near-linear running time.

Potts model on expander graphs. The Q-color ferromagnetic Potts model on a graph $G = (V, E)$ is the probability distribution μ_G^{Potts} on the set of Q-colorings of the vertices of G; i.e., $\{1, \dots, Q\}^V$. Each Q-coloring σ is assigned probability $\mu_G^{\text{Potts}}(\sigma) \propto e^{\beta m(G,\sigma)}$, where $m(G, \sigma)$ is the number of monochromatic edges of G under the coloring σ and $\beta > 0$ is a model parameter. When the parameter β is large, and G has some structure (e.g., G is an expander graph), typical configurations drawn from μ_G^{Potts} are dominated by one of the Q colors; that is, there is phase coexistence in the model. This enables sampling using subset polymer models.

Recall that an n-vertex graph $G = (V, E)$ is an α-expander if for all subsets $S \subseteq V$ with $|S| \le n/2$, the number of edges in E with exactly one endpoint in S is at least $\alpha|S|$.

Corollary 1.6. *Consider the Q-color ferromagnetic Potts model on an α-expander n-vertex graph of maximum degree Δ. Suppose*

$$\beta \ge \frac{1 + \log\left(\frac{\Delta+1}{e\Delta} + 1\right) + \log((Q-1)\Delta)}{\alpha}. \tag{10}$$

Then there is a sampling algorithm with expected running time $O(n \log n)$ that outputs a sample σ with distribution $\hat{\mu}$ so that $\|\hat{\mu} - \mu_G^{\text{Potts}}\|_{\text{TV}} \le e^{-\Omega(n)}$.

Previously, [CGG+21] provided a ε-approximate sample for μ_G^{Potts} in $O(n \log(n/\varepsilon) \log(1/\varepsilon))$ time whenever $\beta \ge \frac{5+3\log((Q-1)\Delta)}{\alpha}$. Condition (10) holds when $\beta \ge \frac{1.2+\log((Q-1)\Delta)}{\alpha}$, so our algorithm applies to a larger range of parameters and removes the dependence on ε from the running time. We do not achieve perfect sampling in this application only because the subset polymer models used give approximations of μ_G^{Potts} rather than describing μ_G^{Potts} exactly.

1.4 Sampling unrooted graphlets in finite graphs

As another application of our algorithm for sampling weighted rooted graphlets, we consider next the problem of sampling weighted *unrooted* graphlets in a finite graph. Given a finite graph G, let

$\mathcal{S}(G)$ be the set of all graphlets of G. Define the distribution $\nu_{G,\lambda}$ on $\mathcal{S}(G)$ by

$$\nu_{G,\lambda}(\gamma) = \frac{\lambda^{|\gamma|}}{Z_{G,\lambda}}, \quad \text{where } Z_{G,\lambda} = \sum\nolimits_{\gamma \in \mathcal{S}(G)} \lambda^{|\gamma|}.$$

Read-McFarland and Štefankovič [RMŠ21] gave a polynomial-time approximate sampling algorithm for $\nu_{G,\lambda}$ for the class of maximum-degree Δ graphs when $\lambda < \lambda_*(\Delta)$ and prove that there is no such algorithm for $\lambda \in (\lambda_*(\Delta), 1)$ unless NP=RP[1]. We give a new algorithm for this problem, covering the entire $\lambda < \lambda_*(\Delta)$ regime, and improving on the result of [RMŠ21] in two ways: (i) our running time is constant in expectation (with no dependence on n), while the running time of the ε-approximate sampler in [RMŠ21] is $n \cdot (1/\varepsilon)^{O(\log \Delta)}$; and (ii) our algorithm outputs a perfect sample instead of an approximate one (and thus the running time has no dependence on any approximation parameter).

Theorem 1.7. *Fix $\Delta \geq 3$ and let $\lambda < \lambda_*(\Delta)$. Then for the class of finite graphs of maximum degree Δ there is a randomized algorithm running in constant expected time that outputs a perfect sample from $\nu_{G,\lambda}$. The expected running time is bounded as a function of Δ and λ.*

The algorithm we use for this theorem is a modification of the one for sampling rooted graphlets. We pick a uniformly random $v \in V$, run the same BFS percolation exploration, and accept the connected component of v with an adjusted probability (to account for the fact that a graphlet can be generated from any of its vertices). The acceptance probability is bounded away from 0 and so the algorithm runs in constant expected time. As mentioned earlier, the ε-approximate sampling algorithm from [RMŠ21] is based on the exhaustive enumeration of all subgraphs of size $\leq k$, for some k that depends on ε. Our new algorithm entirely bypasses this enumeration barrier.

2 Graphlet sampling: algorithms

In this section we present our efficient perfect sampling algorithm for weighted, vertex-labeled graphlets containing a fixed vertex r from a maximum degree Δ graph; in particular, in Section 2.1, we prove a generalized version of Theorem 1.1 from the introduction. We also provide in Section 2.2 our algorithm for sampling weighted graphlets (i.e., the unrooted, unlabeled case) and establish Theorem 1.7. Our hardness results, that is Lemmas 1.2 and 1.3, are proved later in Section 5.

2.1 Sampling rooted vertex-labeled graphlets

Let $G = (V, E)$ be a (possibly infinite) graph of maximum degree Δ. For $U \subseteq V$, let $G[U]$ denote the corresponding vertex-induced subgraph of G; specifically, $G[U] = (U, E(U))$, where $E(U) \subseteq E$ is the set of edges of G with both endpoints in U. A vertex-induced subgraph is a *graphlet* if it is connected. For $r \in V$, let $\mathcal{S}(G, r)$ be the set of all graphlets of G that contain vertex r. We call the graphlets in $\mathcal{S}(G, r)$ the graphlets rooted at r.

Let $\Sigma = \{1, \ldots, q\}$ be a set of vertex labels or colors, and let $\mathcal{S}(G, r, q) = \bigcup_{(S,E(S)) \in \mathcal{S}(G,r)} \Sigma^S$ be the set of all vertex-labeled graphlets rooted at r. Given a real parameter $\lambda > 0$, we assign to each rooted vertex-labeled graphlet $\gamma \in \mathcal{S}(G, r, q) \cup \{\emptyset\}$ with $|\gamma|$ vertices the weight $w_\gamma = \lambda^{|\gamma|} f(\gamma)$, where $f : \mathcal{S}(G, r, q) \cup \{\emptyset\} \to [0, 1]$. Note that $0 \leq w_\gamma \leq \lambda^{|\gamma|}$, which will be important for later analysis.

[1] In [RMŠ21], the threshold is incorrectly stated as $\lambda < \lambda_*(\Delta + 1)$; this is due to a minor error interchanging the infinite Δ-regular tree with the infinite Δ-ary tree; with this small correction their analysis goes through with the bound on λ as stated here.

Define the probability distribution $\nu_{G,r,\lambda}$ on $\mathcal{S}(G,r,q)\cup\{\emptyset\}$ by setting

$$\nu_{G,r,\lambda}(\gamma) = \frac{w_\gamma}{Z(G,r,\lambda)}, \tag{11}$$

where $Z(G,r,\lambda) = \sum_{\gamma'\in\mathcal{S}(G,r,q)\cup\{\emptyset\}} w_{\gamma'}$. We assume that G, f, q and λ are such that $Z(G,r,\lambda)$ is finite, so that this distribution is well defined. When $q=1$ and $f(\gamma)=\mathbb{1}(\gamma\neq\emptyset)$, $\nu_{G,r,\lambda}$ corresponds exactly to the distribution defined in (1) over the unlabeled graphlets of G rooted at r.

We consider the problem of sampling from $\nu_{G,r,\lambda}$; this more general version of the sampling problem is later used as a subroutine for sampling polymer systems in Section 3. Let

$$\lambda_*(\Delta,q) := \frac{(\Delta-2)^{\Delta-2}}{q(\Delta-1)^{\Delta-1}};$$

cf., (2). Our main algorithmic result for sampling colored rooted graphlets is the following.

Theorem 2.1. *Fix $\Delta\geq 3$, $q\geq 1$, and suppose $0<\lambda<\lambda_*(\Delta,q)$. There is a randomized algorithm to exactly sample from $\nu_{G,r,\lambda}$ for graphs G of maximum degree Δ and functions $f:\mathcal{S}(G,r,q)\cup\{\emptyset\}\to[0,1]$ where $f(\gamma)$ is computable in time polynomial in $|\gamma|$; this randomized algorithm has expected running time $O\big(Z_{G,r,\lambda}^{-1}\big)$.*

Theorem 1.1 from the introduction corresponds to the special case when $q=1$ and $f(\gamma)=\mathbb{1}(\gamma\neq\emptyset)$ (in this case $Z_{G,r,\lambda}\geq\lambda$). Other mild assumptions on the function f, e.g., $f(\emptyset)=1$ or $f(r)=1$, ensure that $Z_{G,r,\lambda}$ is bounded away from 0 and, consequently, that the sampling algorithm in the theorem also has constant expected running time in those cases.

As a warm-up, let us consider first our algorithm for sampling labeled rooted graphlets on a finite graph $G=(V,E)$ with $f=1$, and purposely omit certain non-essential implementation details for clarity. First, we find $p\in(0,1)$ such that $\frac{p}{q}(1-p)^{\Delta-2}=\lambda$; this choice of p will be justified in what follows. The algorithm then repeats the following process until a vertex-labeled graphlet is accepted:

1. Each vertex of the graph is independently assigned with probability p a uniform random color from $\{1,\dots,q\}$, or it is marked as "not colored" with the probability $1-p$.
2. Let $\tilde{\gamma}$ be the vertex-labeled graphlet from $\mathcal{S}(G,r,q)\cup\{\emptyset\}$ corresponding the colored connected component of r; i.e., the set of vertices connected to r by at least one path of colored vertices.
3. Observe that the probability that $\tilde{\gamma}=\gamma$ is $(p/q)^{|\gamma|}(1-p)^{|\partial\gamma|}$, where $\partial\gamma$ denotes to set of vertices in G that are not in γ but are adjacent to a vertex in γ (with a slight abuse of notation, we let $|\partial\emptyset|=1$). When $\tilde{\gamma}=\gamma$, our aim is to output γ with probability $\propto\lambda^{|\gamma|}$ which has no dependence on $\partial\gamma$. Therefore, we use a "rejection filter" and only accept γ with probability $(1-p)^{(\Delta-2)|\gamma|+2-|\partial\gamma|}$, so that the probability that γ is the output becomes:

$$\Big(\frac{p}{q}\Big)^{|\gamma|}(1-p)^{|\partial\gamma|}(1-p)^{(\Delta-2)|\gamma|+2-|\partial\gamma|} = (1-p)^2\Big(\frac{p}{q}(1-p)^{\Delta-2}\Big)^{|\gamma|} = (1-p)^2\lambda^{|\gamma|}. \tag{12}$$

From (12), the choice of p such that $\frac{p}{q}(1-p)^{\Delta-2}=\lambda$ is apparent. We will prove that only when $\lambda<\lambda_*(\Delta,q)$ there exists $p\in(0,1)$ such that $\frac{p}{q}(1-p)^{\Delta-2}=\lambda$. In the actual implementation of the algorithm, it will in fact suffice to find an approximation for p.

We comment briefly on the intuition for the rejection filter. The acceptance probability must include a factor of $(1-p)^{-|\partial\gamma|}$, so that the final acceptance probability depends on $|\gamma|$ but not on $|\partial\gamma|$. However, $(1-p)^{-|\partial\gamma|}>1$ is not a valid probability, so we use instead $(1-p)^{(\Delta-2)|\gamma|+2-|\partial\gamma|}$,

which is at most 1 since $(\Delta-2)|\gamma|+2 \geq |\partial\gamma|$. This bound on $|\partial\gamma|$ is best possible since it is tight for the Δ-regular tree. We note that using looser bounds for $|\partial\gamma|$ affects the range of the parameter λ for which we can find $p \in (0,1)$ so that $\frac{p}{q}(1-p)^{\Delta-2} = \lambda$.

Finally, we mention that the algorithm as described requires $\Omega(|V|)$ time per iteration and cannot be extended to infinite graphs. This is easily corrected by assigning colors starting from r and revealing only the colored component of r in a breadth-first fashion. The threshold $\lambda_*(\Delta, q)$ is sharp in the sense that only when $\lambda < \lambda_*(\Delta, q)$ is the value of p such that the revealing process is a sub-critical process that creates a small component with high probability. This ensures the algorithm can be implemented efficiently. In particular, we stress that our algorithm avoids exhaustively enumerating labeled graphlets, as done in previous methods [CGG+21].

Before giving the implementation details of our algorithm and proving Theorem 2.1, we consider the problem of finding $p \in (0,1)$ such that $\frac{p}{q}(1-p)^{\Delta-2} = \lambda$. For $\Delta \geq 3$ and $q \geq 1$, consider the real function $g(x) = \frac{x}{q}(1-x)^{\Delta-2}$. It can be readily checked that the function g is continuous and differentiable in $[0,1]$, has a unique maximum at $x = \frac{1}{\Delta-1}$ with $g(\frac{1}{\Delta-1}) = \lambda_*(\Delta, q)$, is increasing in $[0, \frac{1}{\Delta-1}]$, and decreasing in $[\frac{1}{\Delta-1}, 1]$. This implies that only when $\lambda < \lambda_*(\Delta, q)$, there exists a value of $p \in [0, \frac{1}{\Delta-1})$ such that $g(p) = \lambda$. In particular, when $\lambda > \lambda_*(\Delta, q)$, there is no value of p for which $g(p) = \lambda$ and when $\lambda = \lambda_*(\Delta, q)$, the only possible value is $p = \frac{1}{\Delta-1}$. The latter case would result in a *critical* percolation process, corresponding to the fact that the expected size of a graphlet from $\nu_{G,r,\lambda}$ has no uniform upper bound in the class of graphs of maximum degree Δ; in fact, it is infinite on the Δ-regular tree. We can find a suitable approximation for p when $\lambda < \lambda_*(\Delta, q)$ via a simple (binary search) procedure.

Lemma 2.2. *For any $\lambda \in [0, \lambda_*(\Delta, q))$ we can find rational numbers $\hat{p} \in [0, \frac{1}{\Delta-1})$ and $\hat{\lambda} \in [\lambda, \lambda_*(\Delta, q)]$ such that $g(\hat{p}) = \hat{\lambda}$ in $O(|\log \frac{1}{\Delta q(\lambda_* - \lambda)}|)$ time.*

The proof of this lemma appears after the proof of Theorem 2.1. We now prove Theorem 2.1, including giving a more detailed version of the algorithm outlined above that includes the previously omitted implementation details and allows for general functions $f : \mathcal{S}(G, r, q) \cup \{\emptyset\} \to [0, 1]$.

Proof of Theorem 2.1. For ease of notation, let $\lambda_* = \lambda_*(\Delta, q)$. Our algorithm to sample from $\nu_{G,r,\lambda}$ when $\lambda < \lambda_*$ explores from r in a breadth-first manner and stops once it has revealed the colored connected component of r. It proceeds as follows:

1. Find $\hat{p} \in [0, \frac{1}{\Delta-1})$ and $\hat{\lambda} \in [\lambda, \lambda_*)$ such that $g(\hat{p}) = \hat{\lambda}$. This can be done in $O(|\log \frac{1}{\Delta q(\lambda_* - \lambda)}|)$ time per Lemma 2.2.
2. Let Q be a queue. With probability $1 - \hat{p}$ do not add r to Q; otherwise, assign r a color uniformly at random from $\{1, \ldots, q\}$ and add r to Q. Mark r as explored.
3. While $Q \neq \emptyset$, repeat the following:
   - 3.1) Pop a vertex v from Q.
   - 3.2) For each unexplored neighbor w of v, with probability $1-\hat{p}$ do not add w to Q; otherwise, assign w a color uniformly at random from $\{1, \ldots, q\}$ and add w to Q. Mark w as explored (regardless of whether it was added to Q or not).
4. Let γ be the vertex-labeled graphlet from $\mathcal{S}(G, r, q) \cup \{\emptyset\}$ corresponding the colored connected component of r. Accept γ with probability:

$$f(\gamma) \cdot (1-\hat{p})^{(\Delta-2)|\gamma|+2-|\partial\gamma|} \left(\frac{\lambda}{\hat{\lambda}}\right)^{|\gamma|}.$$

5. If γ is rejected, go to Step 2 and repeat.

The probability of obtaining $\gamma \in \mathcal{S}(G,r,q) \cup \{\emptyset\}$ in an iteration of the algorithm is:

$$\left(\frac{\hat{p}}{q}\right)^{|\gamma|}(1-\hat{p})^{|\partial\gamma|} \cdot f(\gamma)(1-\hat{p})^{(\Delta-2)|\gamma|+2-|\partial\gamma|}\left(\frac{\lambda}{\hat{\lambda}}\right)^{|\gamma|} = (1-\hat{p})^2 f(\gamma)\,\lambda^{|\gamma|} = (1-\hat{p})^2 w_\gamma,$$

and thus the overall acceptance probability in an iteration is:

$$\rho := (1-\hat{p})^2 \sum_{\gamma \in \mathcal{S}(G,r,q)\cup\{\emptyset\}} w_\gamma = (1-\hat{p})^2 Z_{G,r,\lambda}.$$

Then,

$$\Pr[\gamma \in \mathcal{S}(G,r,q) \cup \{\emptyset\} \text{ is the output}] = \sum\nolimits_{t\geq 1}(1-\hat{p})^2 w_\gamma (1-\rho)^{t-1} = \frac{(1-\hat{p})^2 w_\gamma}{\rho} = \nu_{G,r,\lambda}(\gamma).$$

We next bound the expected running time of the algorithm. We claim first that expected running per iteration is at most a constant that depends only on a, Δ and q. If γ is the configuration generated in an iteration, it is discovered in $O(|\gamma|+|\partial\gamma|) = O(|\gamma|)$ time and, by assumption, $f(\gamma)$ can be computed in at most $O(|\gamma|^a)$ time, for suitable a constant $a > 0$. Let $\hat{\mu}$ the output distribution of Step 3 of the algorithm. Then, there exists a constant $C = C(q,\Delta) > 0$ such that the expected running time of each iteration is at most:

$$C \sum_{\gamma \in \mathcal{S}(G,r,q)\cup\emptyset} |\gamma|^{\max\{a,1\}} \Pr_{\hat{\mu}}[\gamma] = C \cdot \mathbb{E}_{\hat{\mu}}[|\gamma|^{\max\{1,a\}}]. \tag{13}$$

We show next that $|\gamma|$ (under $\hat{\mu}$) is stochastically dominated by a random variable $W = X + Y$ (i.e., $|\gamma| \prec W$), where X and Y are i.i.d. random variables corresponding to the cluster size of a homogeneous Galton-Watson process with offspring distribution $Bin(\Delta-1,\hat{p})$. We recall that this is the branching process that starting from a single vertex (or individual) N_0, adds $Z_1 \sim Bin(\Delta-1,\hat{p})$ descendants to N_0. The process is then repeated for each new descendant. The process can either die out or go on forever; the cluster size is the number of descendants of N_0. When different offspring of N_0 use different distributions to generate its descendants, the process is called heterogeneous (see, e.g., [JRL11] for additional background).

To see that $|\gamma| \prec W = X + Y$, first note that $|\gamma| \prec L$, where L is the cluster size of a heterogeneous Galton-Watson process, in which the root vertex has offspring distribution $Bin(\Delta,\hat{p})$ and every other vertex has offspring distribution $Bin(\Delta-1,\hat{p})$. This is because the branching process generating γ includes the root only with probability $\hat{p}$ (the root is always present in the Galton-Watson process), and, in addition, it considers at most Δ (from the root) or $\Delta-1$ (from any other vertex) potential branches (or descendants). In turn, we can bound the cluster size L by $L \prec X + Y$, since, in the branching process corresponding to L, we can couple the first $\Delta-1$ branches of the root N_0 with X (starting at the root) and the remaining branch with Y (starting at the child of the root not coupled with X).

It is well-known that X and Y have finite moments when $(\Delta-1)\hat{p} < 1$ (see, e.g., [JRL11]). In particular, there exists a constant $A = A(a,\Delta,\hat{p}) > 0$ such that

$$\mathbb{E}_{\hat{\mu}}[|\gamma|^a] \leq \mathbb{E}[L^a] \leq \mathbb{E}[(X+Y)^a] \leq 2^a(\mathbb{E}[X^a] + \mathbb{E}[Y^a]) \leq A. \tag{14}$$

This together with (13) shows that the expected running time in each iteration of the algorithm is bounded by $C \cdot A$.

Now, let R be the number of times Steps 2–5 are repeated, let T be the overall running time of the algorithm. Then:

$$\mathbb{E}[T] = \sum_{t\geq 1} \mathbb{E}[T \mid R = t] \Pr[R = t] \leq C \cdot A \cdot \sum_{t\geq 1} t(1-\rho)^{t-1}\rho \leq \frac{CA}{\rho}, \tag{15}$$

and the result follows. □

We conclude this section with the proof of Lemma 2.2.

Proof of Lemma 2.2. It suffices to find $\hat{p} \in [p, \frac{1}{\Delta-1}]$. This can be done via binary search in t steps, provided $t \geq 0$ is such that $\frac{1}{\Delta-1} \cdot \frac{1}{2^t} \leq \frac{1}{\Delta-1} - p$. Since $g' \leq \frac{1}{q}$, it follows from the mean value theorem that $q(\lambda_* - \lambda) \leq \frac{1}{\Delta-1} - p$. Thus for the binary search to require at most t steps it is sufficient to pick t so that $\frac{1}{\Delta-1} \cdot \frac{1}{2^t} \leq q(\lambda_* - \lambda)$, and the result follows. □

2.2 Sampling unrooted graphlets

We consider next the problem of sampling weighted graphlets from a finite graph $G = (V, E)$ of maximum degree Δ; specifically, in this variant of the sampling problem we consider unrooted, unlabeled, weighted graphlets of G. Let $\mathcal{S}(G)$ be the set of all graphlets of G. We define the probability distribution $\nu_{G,\lambda}$ on $\mathcal{S}(G)$ by setting

$$\nu_{G,\lambda}(S) = \frac{\lambda^{|S|}}{Z_{G,\lambda}},$$

where $Z_{G,\lambda} = \sum_{S' \in \mathcal{S}(G)\cup\{\emptyset\}} \lambda^{|S'|}$. The problem of (approximately) sampling from $\nu_{G,\lambda}$ is quite natural. In [RMŠ21], it was established that this problem is computationally hard when $\lambda > \lambda_*(\Delta) = \frac{(\Delta-2)^{\Delta-2}}{(\Delta-1)^{\Delta-1}}$; an ε-approximate sampling algorithm was also given in [RMŠ21] for the case when $\lambda < \lambda_*(\Delta)$ with running time $n \cdot (1/\varepsilon)^{O(\log \Delta)}$. We establish the following:

Theorem 2.3. *Suppose $\Delta \geq 3$ and $\lambda > 0$ are such that $\lambda < \lambda_*(\Delta)$. There is a randomized algorithm to exactly sample from $\nu_{G,\lambda}$ with $O(1)$ expected running time for finite graphs G of maximum degree Δ.*

Proof. For ease of notation, we set $\lambda_* = \lambda_*(\Delta)$ throughout this proof. Our algorithm to sample from $\nu_{G,\lambda}$ is based on the algorithm to sample from $\nu_{G,r,\lambda}$ (the rooted, vertex-labeled, weighted case). The idea is to pick a root uniformly at random and run the algorithm for the rooted case from this random vertex with the rejection filter adjusted to account for the fact that a graphlet can be generated from any of its vertices. It proceeds as follows:

1. Find $\hat{p} \in [0, \frac{1}{\Delta-1})$ and $\hat{\lambda} \in [\lambda, \lambda_*)$ such that $g(\hat{p}) = \hat{\lambda}$ using the method from Lemma 2.2.
2. Pick a vertex $r \in V$ uniformly at random.
3. Let Q be a queue. With probability $\hat{p}$ add r to Q and mark it as colored. Mark r as explored.
4. While $Q \neq \emptyset$, repeat the following:
   - 3.1) Pop a vertex v from Q.
   - 3.2) For each unexplored neighbor w of v, with probability $\hat{p}$ add w to Q and mark w as colored. Mark w as explored.

5. Let $S \in \mathcal{S}(G)$ be the graphlet corresponding to the colored connected component of v. Accept S with probability:
$$\frac{1}{|S|} \cdot (1-\hat{p})^{(\Delta-2)|S|+2-|\partial S|}\Big(\frac{\lambda}{\hat{\lambda}}\Big)^{|S|}.$$

6. If S is rejected, go back to Step 2 and repeat.

The analysis of this algorithm is similar to that in the proof of Theorem 2.1. Let $n = |V|$. The probability that the algorithm outputs S in an iteration is:
$$\sum_{v \in S} \frac{1}{n} \cdot \hat{p}^{|S|}(1-\hat{p})^{|\partial S|} \cdot \frac{1}{|S|} \cdot (1-\hat{p})^{(\Delta-2)|S|+2-|\partial S|}\Big(\frac{\lambda}{\hat{\lambda}}\Big)^{|S|} = \frac{(1-\hat{p})^2\lambda^{|S|}}{n}. \tag{16}$$

Hence, conditioned on acceptance, the probability of obtaining $S \in \mathcal{S}(G)$ is thus $\nu_{G,\lambda}(S)$, and so the output distribution of the algorithm is $\nu_{G,\lambda}$.

For the running time of the algorithm, we note that Step 4 of the algorithm is analogous to Step 3 of the algorithm in the proof of Theorem 2.1, and so the expected running time of each round is also bounded by a constant $C = C(\Delta, \hat{p}) > 0$. Let T be the overall the running time of the algorithm. From (16), we have that the overall acceptance probability in a round is $\rho = \frac{(1-\hat{p})^2 Z(G,\lambda)}{n}$. Then, as in (15), we deduce that $\mathbb{E}[T] = O(nZ(G,\lambda)^{-1})$. Since $Z(G,\lambda) \geq n\lambda$, we have $\mathbb{E}[T] = O(1)$. □

3 Applications to Polymer Models

In this section, we show how to use our algorithm for sampling rooted vertex-labeled graphlets from Section 2 to sample from subset polymer models and prove Theorem 1.4.

Consider a subset polymer model on an n-vertex graph $G = (V, E)$; see Section 1.2 for the definition. Recall that we use $\mathcal{C}_v$ for the set of all polymers containing vertex $v \in V$, and let $\gamma_\emptyset$ denote the empty polymer. Define the distribution ν_v on $\mathcal{C}_v \cup \{\gamma_\emptyset\}$ where $\nu_v(\gamma) = \frac{w_\gamma}{40}$ and
$$\nu_v(\gamma_\emptyset) = 1 - \frac{1}{40}\sum_{\hat{\gamma} \in \mathcal{C}_v \cup \{\gamma_\emptyset\}} w_{\hat{\gamma}}.$$

We note that ν_v is well-defined when condition (3) holds, since under this condition we have $\sum_{\hat{\gamma} \in \mathcal{C}_v \cup \{\gamma_\emptyset\}} w_{\hat{\gamma}} \leq 40$; this is proved later in Lemma 5.1. The following Markov chain on $\Omega(\mathcal{C})$ is similar to the one introduced in [CGG+21].

Polymer dynamics. Given a configuration $X_t \in \Omega(\mathcal{C})$, form X_{t+1} as follows:

1. Pick $v \in V$ uniformly at random and let $S_v = \{\gamma \in X_t : v \in \gamma\}$ (note that S_v is either empty or contains 1 polymer).

2. With probability 1/41, let $X_{t+1} = X_t \setminus S_v$.

3. Otherwise, with the remaining probability 40/41, sample γ from ν_v. Let $X_{t+1} = X_t \cup \{\gamma\}$ if $X_t \cup \{\gamma\} \in \Omega(\mathcal{C})$ and let $X_{t+1} = X_t$ otherwise.

We first note that this Markov chain is irreducible (every configuration can reach and can be reached from the empty set of polymers with positive probability), and aperiodic (there is positive probability of remaining on the same state). It is also reversible with respect to μ: letting P be

the transition matrix for this chain, $\Gamma \in \Omega(\mathcal{C})$, and a polymer $\gamma \notin \Gamma$ be such that $\Gamma \cup \{\gamma\} \in \Omega(\mathcal{C})$, we have $\mu(\Gamma \cup \{\gamma\})/\mu(\Gamma) = w_\gamma$ and for $\gamma \neq \gamma_\emptyset$:

$$\frac{P(\Gamma, \Gamma \cup \{\gamma\})}{P(\Gamma \cup \{\gamma\}, \Gamma)} = \frac{\sum_{v \in \gamma} \frac{40}{41n} \nu_v(\gamma)}{\frac{|\gamma|}{41n}} = w_\gamma.$$

Now, to implement a single update step of the polymer dynamics, one must sample from ν_v in Step 3. We give a fast perfect sampler for ν_v.

Theorem 3.1. *Fix $\Delta \geq 3, q \geq 1$, and let $\lambda < \lambda_*(\Delta, q)$. Consider a family of subset polymer models defined on graphs of maximum degree Δ with computationally feasible weights that satisfy $w_\gamma \leq \lambda^{|\gamma|}$. There is a randomized algorithm to sample perfectly from ν_v for any $v \in V(G)$ with a constant expected running time.*

Proof. For each vertex v, let $\hat{\nu}_v$ be the distribution over $\mathcal{C}_v \cup \{\gamma_\emptyset\}$ where $\hat{\nu}_v(\gamma) = w_\gamma / Z_v$ with $Z_v = \sum_{\hat{\gamma} \in \mathcal{C}_v \cup \{\gamma_\emptyset\}} w_{\hat{\gamma}}$ and $w_{\gamma_\emptyset} = 1$. By Theorem 2.1, we can perfectly from $\hat{\nu}_v$ in constant expected time, with the constant depending λ, Δ, and q. Our algorithm to sample from ν_v draws a perfect sample from $\hat{\nu}_v$ first and then outputs the sample with probability $Z_v/40$; otherwise it outputs the empty polymer $\gamma_\emptyset$. The output distribution of this algorithm is exactly ν_v, and so all that remains is for us to show how to sample exactly from a Bernoulli distribution with parameter $Z_v/40$, denoted $\mathrm{Ber}(Z_v/40)$.

We can sample exactly from $\mathrm{Ber}(1/Z_v)$ in constant expected time by drawing an exact sample from $\hat{\nu}_v$ (using the algorithm from Theorem 2.1) and outputting 1 if the sample is $\gamma_\emptyset$ and 0 otherwise. With this, we can then use a "Bernoulli factory" to obtain a perfect sample from $\mathrm{Ber}(Z_v/40)$. Specifically, we know how to simulate a Bernoulli coin with parameter $p = 1/Z_v$ and would like to obtain a Bernoulli coin with parameter $f(p) = 1/(40p) = Z_v/40$. This is possible in constant expected time (independent of p) since $1 + \lambda \leq Z_v \leq 40$, and the function $f(p) = 1/(40p)$ is real analytic in the closed interval $[1/40, 1 + \lambda]$; see Theorem 2 from [SY05]. □

Note that Bernoulli factories have been used in a similar fashion to design perfect sampling algorithms for CSP solutions and spin models in [HWY22, HWY23, AGPP23].

Using this theorem, we give next a perfect sampling algorithm that works whenever a new condition (4) is satisfied (our algorithm also requires the assumptions in Theorem 3.1).

3.1 Perfect Sampling for polymer systems: Proof of Theorem 1.4

We propose here an algorithm to output a perfect sample from μ. Our algorithm is based on the polymer dynamics and the coupling from the past method [PW96], using the notion of bounding Markov chains [Hub04, HN99] to efficiently implement it.

We proceed with the proof of Theorem 1.4. We start with the description of a grand coupling for the polymer dynamics, which is then used to implement a coupling from the past algorithm. For an n-vertex graph $G = (V, E)$, let $\{X_t^\Gamma\}$ denote an instance of the polymer dynamics started from the polymer configuration $\Gamma \in \Omega(\mathcal{C})$. For all $\Gamma \in \Omega(\mathcal{C})$, the chains $\{X_t^\Gamma\}$ are coupled by choosing the same uniform random random vertex $v \in V$, the same polymer γ sampled from ν_v, and the same uniform random number in $[0, 1]$ to decide whether to remove S_v (Step 2) or to add γ (Step 3). A coupling from the past algorithm will find a time $-T$ such the grand coupling started from all possible states at time $-T$ coalesces to a single state by time 0. This guarantees that the output of the algorithm, that is the state at time 0, has distribution μ (see Theorem 1 from [PW96]). Such a T can be found with a binary search procedure. Unfortunately, implementing the coupling

from the past algorithm in this manner for the polymer dynamics Markov chain is infeasible, since it requires simulating an exponential number of copies of the polymer dynamics, one from each $\Gamma \in \Omega(\mathcal{C})$.

To work around this, we consider a bounding Markov chain for the polymer dynamics rather than the polymer dynamics chain itself. Bounding Markov chains were pioneered in [Hub04, HN99] as a method for efficiently implementing coupling from the past. The bounding chain for the polymer dynamics has state space $\Omega(\mathcal{C}) \times 2^{\mathcal{C}}$ and will be denoted by $\{B_t, D_t\}$, where $B_t \in \Omega(\mathcal{C})$ and $D_t \subseteq \mathcal{C}$ are sets of polymers. The chain will maintain throughout that all polymers in B_t are compatible and that every polymer in B_t is compatible with every polymer in D_t. The polymers in D_t do *not* need to be compatible with each other. A step of the bounding Markov chain is defined next.

Polymer Dynamics Bounding Chain. Given $\{B_t, D_t\}$, the chain generates $\{B_{t+1}, D_{t+1}\}$ by:

1. Uniformly at random, select $v \in V$.
2. With probability 1/41, remove all polymers containing v by setting $B_{t+1} = B_t \setminus \mathcal{C}_v$ and $D_{t+1} = D_t \setminus \mathcal{C}_v$.
3. With the remaining probability 40/41, draw a sample γ according to ν_v and:
   (a) If γ is compatible with B_t and γ is compatible with $D_t \setminus \{\gamma\}$, let $B_{t+1} = B_t \cup \{\gamma\}$ and let $D_{t+1} = D_t \setminus \{\gamma\}$.
   (b) Else if γ is compatible with B_t but γ is not compatible with D_t, let $B_{t+1} = B_t$ and let $D_{t+1} = D_t \cup \{\gamma\}$.
   (c) If γ is incompatible with B_t, do nothing: $B_{t+1} = B_t$ and $D_{t+1} = D_t$.

Observe that polymers are only added to B_t if they are compatible with all other polymers in B_t; hence, if B_0 is a valid polymer configuration, so is B_t for all $t \geq 0$.

To implement a step of the polymer dynamics bounding chain it suffices to pick a vertex $v \in V$ uniformly at random, a uniform random number in $[0, 1]$, and a polymer γ from ν_v, just like for the polymer dynamics. Hence, we can couple the evolution of $\{B_t, D_t\}$ with the grand coupling of the polymer dynamics described earlier. If we set $B_0 = \emptyset$ and $D_0 = \mathcal{C}$, it can be checked that for all $\Gamma \in \Omega(\mathcal{C})$ and all $t \geq 0$:

$$B_t \subseteq X_t^\Gamma \subseteq B_t \cup D_t.$$

Indeed, this holds initially for $t = 0$, and the grand coupling ensures that whenever a polymer is removed from X_t^Γ it is also removed from B_t, and whenever a polymer is added to X_t^Γ it is also added to B_t or D_t. Consequently, $\{B_t, D_t\}$ is a bounding chain for the polymer dynamics. In particular, the first time $B_t = B_t \cup D_t$, all instances X_t^Γ have necessarily coalesced to the same configuration. This bounding chain allows us to implement the coupling from the past algorithm efficiently, as follows.

Coupling from the Past. Set $k = 1$.

(A) For $t = -2^k, -2^k + 1, \ldots, -2^{k-1}$ generate $\rho_t = (v_t, \gamma_t, r_t)$ by choosing $v_t \in V$ uniformly at random, $r_t \in [0, 1]$ uniformly at random, and by sampling $\gamma_t \in \mathcal{C}_{v_t}$ from ν_{v_t}.

(B) Set $B_{-2^k} = \emptyset$ and $D_{-2^k} = \mathcal{C}$.

(C) Simulate the polymer dynamics bounding chain from time -2^k to time 0 using $\rho_{-2^k}, \ldots, \rho_{-1}$.

(D) If $B_0 = B_0 \cup D_0$, then output B_0; otherwise set $k \to 2k$ and repeat the process from Step (A).

This implementation of the coupling from the past algorithm provides a perfect sample from μ; see [PW96]. It remains for us to show that it is efficient. For this, we show first that the expected number of steps of the polymer dynamics bounding chain throughout the execution of the algorithm is $O(n \log n)$. Afterwards, we will show how to implement steps so that they can be executed in amortized constant expected time.

Lemma 3.2. *Suppose a subset polymer model on an n-vertex graph satisfies condition (4). Then, the expected number of steps of the polymer dynamics bounding chain in the coupling from past algorithm is $O(n \log n)$.*

Proof. Let us first consider an evolution of the bounding chain $\{B_t, D_t\}$ from the initial state $\{B_0, D_0\} = \{\emptyset, \mathcal{C}\}$. Let T be the first time $B_t = B_t \cup D_t$; that is, the time until $D_t = \emptyset$. We bound first the expected value of T. Let $\phi_t = |(\cup_{\gamma \in D_t} \gamma) \cap V|$, the number of vertices of G that are included in at least one polymer of D_t. Note that $\phi_t \leq n$. We analyze the expected value of $\phi_{t+1} - \phi_t$.

With probability $\phi_t/(41n)$, a vertex in a polymer of D_t is picked and all polymers in D_t containing that vertex are removed, in which case $\phi_{t+1} \leq \phi_t - 1$. (A polymer may also be removed from D_t in Step 3(a), but this case is omitted from our calculations as it never increases ϕ_t.)

The potential ϕ_t may increase in Step 3(b), when a polymer γ may be added to D_t. Such a polymer is only added if it is incompatible with D_t. The probability this occurs is at most:

$$\sum_{v \in D_t} \frac{40}{41n} \sum_{\gamma \not\sim v} \nu_v(\gamma)$$

For each such polymer, the expected increase in the size of ϕ_t is at most $|\gamma|$. We can calculate:

$$\mathbb{E}[\phi_{t+1} - \phi_t \mid B_t, D_t] \leq -\frac{\phi_t}{41n} + \sum_{v \in D_t} \frac{40}{41n} \sum_{\gamma \not\sim v} |\gamma| \nu_v(\gamma) = -\frac{\phi_t}{41n} + \frac{\phi_t}{41n} \sum_{\gamma \not\sim v} |\gamma| w_\gamma.$$

By (4), there exists a constant $\theta < 1$ such that $\sum_{\gamma \not\sim v} |\gamma| w_\gamma \leq \theta$. It follows that

$$\mathbb{E}[\phi_{t+1} - \phi_t \mid B_t, D_t] \leq -\frac{\phi_t}{41n}(1 - \theta).$$

This shows that $\{\phi_t\}$ is a stochastic process with variable multiplicative drift. Since T is also the first time $\phi_t < 1$, using standard hitting time estimates (see, e.g., Theorem 10 in [KK19]), we get:

$$\mathbb{E}[T] \leq \frac{41n}{1-\theta} + \frac{41n}{1-\theta} \int_1^n \frac{1}{z} dz = O(n \log n).$$

Now, let $-M$ be the first time in past such that, after setting $B_{-M} = \emptyset$ and $D_{-M} = \mathcal{C}$, we have $B_0 = B_0 \cup D_0$. The coupling from the past algorithm simulates at most $2M + M + M/2 + \cdots + 1 = O(M)$ steps of the bounding chain. The result follows by noting that $\mathbb{E}[M] = \mathbb{E}[T] = O(n \log n)$. □

It remains to consider how to efficiently implement the steps of the polymer dynamics bounding chain. This is subtle because D_t may initially contain an exponentially large number of polymers, and care is thus needed in how D_t is represented and stored. We describe and analyze efficient data structures for B_t and D_t in the following lemma.

Lemma 3.3. *Suppose a subset polymer model on an n-vertex graph of maximum degree $\Delta \geq 3$ with computationally feasible weights that satisfy $w_\gamma \leq \lambda^{|\gamma|}$ for some $\lambda < \lambda_*(\Delta, q)$. There exists a compact representation of B_t and D_t that uses $O(n+t)$ space in expectation. Using this representation, each iteration of the Polymer Dynamics Bounding Chain can be executed in amortized constant expected time, and the termination condition $B_t = B_t \cup D_t$ can be checked in constant time.*

Proof. First, by Lemma 3.1, there is a randomized algorithm to sample perfectly from ν_v for any $v \in V$ with constant expected running time. Therefore, the expected size of the polymer γ produced by the sampling procedure is constant. This will imply many of our data structure operations, which take (amortized) time $O(|\gamma|)$ for some γ sampled from ν_v, take constant expected time.

For simplicity, we assume time begins at $t = 0$ and at some point in the future we wish to check whether $B_t = B_t \cup D_t$. Shifting the indices appropriately, this data structure can be applied in each round of coupling from the past from each negative starting time. Note because all polymers in B_t are compatible with all polymers in D_t, the condition $B_t = B_t \cup D_t$ is equivalent to the condition $D_t = \emptyset$.

Data Structure for B_t: Note B_t is a collection of compatible polymers. The operations to be performed on B_t include addition of a polymer (Step 3a), deletion of any polymer containing a particular vertex (Step 2), and comparison of a polymer to B_t to determine whether it is compatible with B_t (Step 3). B_t will be stored as a length n array, and we use $\overline{B}_t$ to denote the state of the array at time t. Initially, $\overline{B}_0$ has only 0 entries. Each entry may also have a pointer to a node of a doubly linked list. Each polymer in B_t is stored as a doubly linked list. This requires $O(n)$ space.

When γ is added to B_t, for each $v \in \gamma$ we set $\overline{B}_t(v) = 1$. We also create a doubly linked list $\mathcal{L}^\gamma$, which has a node for each $v \in \gamma$ that notes the color v is assigned in γ and contains a forward pointer, a backward pointer, and a pointer back to $\overline{B}_t(v)$. Each entry of $\overline{B}_t$ for $v \in \gamma$ also points to the corresponding node of $\mathcal{L}^\gamma$. Doing these updates to the data structure for B_t takes time $O(|\gamma|)$.

The polymer removal step only occurs in Step 2 when any polymers containing v are removed from B_t. There is at most one such polymer γ in B_t containing v. The other vertices in γ can quickly be found using the doubly linked list $\mathcal{L}^\gamma$ and its pointers back to $\overline{B}_t$. We can set $B_t(w) = 0$ for each $w \in \gamma$ and remove $\mathcal{L}^\gamma$. This removal of γ from B_t occurs in time $O(|\gamma|)$.

To compare a polymer γ to B_t to determine whether it is compatible with B_t, one looks up every vertex w that is in or adjacent to γ in $\overline{B}_t$ to see if it is part of a polymer of B_t, that is, if $\overline{B}_t(w) = 1$. If at least one of these vertices w has $\overline{B}_t(w) = 1$, γ is incompatible with B_t; if all these w have $\overline{B}_t(w) = 0$, then γ is compatible with B_t. This takes time $O(\Delta|\gamma|)$, which because we assume Δ is constant is $O(|\gamma|)$.

Note each of these operations can be performed in time $O(|\gamma|)$, where γ is a polymer that was at some (possibly earlier) time step drawn from ν_v for some v. When each polymer γ is sampled in Step 3 in some iteration of the Polymer Dynamics Bounding Chain, all $O(|\gamma|)$ operations that can potentially be performed on it in the future are immediately charged to it. As argued in the first paragraph above, the expected size of a polymer γ drawn in Step 3 is constant, so the amount $O(|\gamma|)$ charged to each sampled polymer γ is constant in expectation. We conclude updating B_t takes expected amortized constant time. This data structure requires $O(n)$ space.

Data Structure for D_t: Much more care is needed with how D_t is stored and accessed. Initially $D_0 = \mathcal{C}$, which contains an exponential number of polymers. The operations we need to perform on D_t include deleting all polymers containing a single vertex v (Step 2), checking whether a polymer is compatible with D_t (Step 3b), adding a polymer to D_t (Step 3b), and checking whether a polymer γ is compatible with $D_t \setminus \{\gamma\}$ and if so deleting γ from D_t (Step 3b).

The key observation is that once a particular vertex v of G is selected in Step1 and the coin

flip is such that Step 2 is performed, no polymers containing v remain in $\mathcal{C}$ and the size of D_t has been reduced dramatically. Because of this, it makes sense to keep track of which vertices in D_t have been the subject of a deletion in Step 2 at least once. We let D_t^* be (the state at time t) of an array of length n with an entry corresponding to each vertex v of G, where $D_t^*(v) = 0$ if Step 2, deleting $\mathcal{C}_v$ from D_t, has been performed for v at least once, and $D_t^*(v) = 1$ if it has not. We also let N_t^* be the number of 1's in D_t^*. Initially, $D_0^*(v) = 1$ for all v and $N_0^* = n$. At any time step, we know any polymer γ where $D_t^*(v) = 1$ for all $v \in \gamma$ is in D_t. Because of this, initially D_0^* completely describes D_0.

Amidst these deletions, polymers are also added to D_t, and a polymer may contain both vertices where $D_t^*(v) = 0$ and vertices where $D_t^*(v) = 1$. All polymers added to D_t in Step 3b will be stored separately from D_t^*. The data structure used here will have the same idea as that for B_t. There will be an array $\overline{D}_t$ with an entry for each vertex where $\overline{D}_t(v)$ gives the number of polymers containing v that are currently stored in D_t. We also keep track of $\overline{N}_t$, the total number of polymers that are currently in D_t. Adding a polymer γ to D_t involves incrementing N_t, incrementing the corresponding entries in $\overline{D}_t$, and adding a doubly linked list $\mathcal{L}^\gamma$ connecting the vertices of the polymer together. However, because polymers in D_t need not be compatible, there may be more than one polymer in D_t containing a given vertex v. Because of this, instead of each vertex being in at most one doubly linked list $\mathcal{L}^\gamma$, it may be in many such doubly linked lists. To maintain all such pointers for vertex v (two for each $\mathcal{L}^\gamma$ where $v \in \gamma$), these pointers are themselves stored in another doubly linked list $\mathcal{L}^v$.

The necessary operations that must be performed on D_t can be implemented as follows. Note that initially $D_0^*(v) = 1$ for all v, $N_0^* = n$, $\overline{D}_0(v) = 0$ for all v, $\overline{N}_0 = 0$, there are no doubly linked lists $\mathcal{L}^\gamma$, and each doubly linked list $\mathcal{L}^v$ is empty. These steps are presented in a logical order for explaining the data structures for D_t, rather than in the order in which they occur in the description of the Polymer Dynamics Bounding Chain.

- *Adding polymer γ to D_t (Step 3b)*: For each v in γ, increment $\overline{D}_t(v)$. Increment $\overline{N}_t$. Create a doubly linked list $\mathcal{L}^\gamma$, which has a node for each $v \in \gamma$ that notes the color v is assigned in γ and containing a forward pointer and a backward pointer. For each v, this node containing the pointers of $\mathcal{L}^\gamma$ is inserted at the front of the doubly linked list $\mathcal{L}^v$. This takes time $O(|\gamma|)$.

- *Deleting all polymers containing v from D_t (Step 2)*: If $D_t^*(v) = 1$, decrement N_t^* and set $D_t^*(v) = 0$. If $\overline{D}_t(v) > 0$, set $\overline{N}_t \leftarrow \overline{N}_t - \overline{D}_t(v)$ and set $\overline{D}_t(v) = 0$. Additionally, for each node in $\mathcal{L}^v$, we explore the corresponding linked list $\mathcal{L}^\gamma$ using the pointers in the node. We delete every other node of $\mathcal{L}^\gamma$, possible in time $O(|\gamma|)$ because for list $\mathcal{L}^w$ also containing a node of $\mathcal{L}^\gamma$, the nodes before and after this node in $\mathcal{L}^w$ can easily be connected to each other because $\mathcal{L}^w$ is doubly linked. For each node in $\mathcal{L}^w$ that is deleted, we also decrement $\overline{D}_t(w)$. This takes time $O(\sum_{\gamma \ni v} |\gamma|)$.

- *Checking if polymer γ is compatible with D_t (Step 3b)*: For each w that is in or adjacent to γ, check whether $D_t^*(w) = 0$ and $\overline{D}_t(w) = 0$. If both are 0 for all such w, then γ is compatible with D_t. If at least one is nonzero, then γ is not compatible with D_t. This takes time $O(\Delta|\gamma|)$.

- *Checking whether a polymer γ is compatible with $D_t \setminus \{\gamma\}$ and if so, deleting γ from D_t (Step 3a)*: We only need to check this case when γ is compatible with B_t. We first check whether γ is compatible with D_t, as described above. If it is, we add γ to B_t. If γ is compatible with D_t then it cannot be in D_t, so D_t and $D_t \setminus \{\gamma\}$ are the same and no further steps are needed.

If γ is incompatible with D_t, the next step is to check if it is also incompatible with $D_t \setminus \{\gamma\}$. We check if $D^*(v) = 0$ for all v in or adjacent to γ, $\overline{D}_t(v) = 0$ for all v adjacent to γ, and $\overline{D}_t(v) = 1$ for all $v \in \gamma$; if at least one of these does not hold, γ cannot be compatible with $D_t \setminus \{\gamma\}$ and so we are not in Step 3a and we proceed to Step 3b. If all these hold, we check further to see whether all $\overline{D}_t(v) = 1$ for $v \in \gamma$ because of the presence of the single polymer γ or due to the presence of other polymers, which we can detect by looking at the doubly linked list(s) connecting the vertices in γ, including the colors assigned to each vertex. If there are polymers other than γ here, either multiple smaller polymers or a polymer with the same vertices as γ but differently assigned colors, then γ is incompatible with $D_t \setminus \{\gamma\}$ and we are not in Step 3a. If instead we find exactly polymer γ, then γ is compatible with $D_t \setminus \{\gamma\}$ and we add γ to B_t and delete it from D_t as previously described. This takes time $O(\Delta|\gamma|)$.

Because any polymer added to D_t was drawn from the distribution ν_v for some v, as argued above its expected size is constant. Therefore all of the implementations above, with the possible exception of deleting all polymers containing a single vertex, takes expected constant time. Setting $D_t^*(v) = 0$ takes constant time, but it may take longer to remove any polymers that were added as linked lists $\mathcal{L}^\gamma$. However, we can amortize the cost of deletion if we pay for the cost of deleting a polymer when we add it, as all doubly linked lists $\mathcal{L}^\gamma$ must be added before they are deleted. Doing this amortization makes the cost of adding a polymer to D_t be $O(|\gamma|) + O(|\gamma|) = O(|\gamma|)$, while the amortized cost of deleting all polymers containing a vertex v is now $O(1)$. Thus all necessary operations for D_t can be performed in amortized expected constant time.

Finally, we note that the termination condition $B_t = B_t \cup D_t$, equivalent to $D_t = \emptyset$, can be checked in constant time by verifying $N_t^* = 0$ and $\overline{N}_t = 0$. The first condition $N_t^* = 0$ verifies that every vertex v has had all polymers containing v deleted at least once, and the second condition $\overline{N}_t = 0$ verifies that there are currently no additional polymers in D_t.

The total space used for these data structures after t steps of the algorithm is in expectation at most $O(n + t)$: $O(n)$ for $\overline{B}_t$, D_t^* and $\overline{D}_t$, and $O(|\gamma|)$ for each polymer γ added to D_t, which is constant in expectation. There are at most t polymers in D_t, one added each step, so this is $O(t)$ space at most in expectation. □

Together, Theorem 3.1, Lemma 3.2, and Lemma 3.3 imply Theorem 1.4.

4 Applications to low-temperature spin systems

We present in this section the details of the applications spin system sampling given in Section 1.3.

4.1 The hard-core model on unbalanced bipartite graphs

We start by proving Corollary 1.5 for which we use Theorem 1.4.

Proof of Corollary 1.5. Let G be an n-vertex bipartite graph with partite sets L and R. Suppose that the vertices in L have maximum degree Δ_L, and that the vertices in R have maximum degree Δ_R and minimum degree δ_R.

We can sample from the hardcore model distribution μ_G^{hc} (see (8) for its the definition) by considering an auxiliary distribution $\overline{\mu}$, which is a distribution on subsets $S \subseteq R$. For a subset $S \subseteq R$ with neighbors $N(S)$ in L, this distribution is given by

$$\overline{\mu}(S) = \frac{(1+\lambda)^{|L \setminus N(S)|} \cdot \lambda^{|S|}}{Z},$$

where $Z = \sum_{S \subseteq R} (1+\lambda)^{|L \setminus N(S)|} \cdot \lambda^{|S|}$. One can sample from μ_G^{hc} by first sampling $S \subseteq R$ according to $\overline{\mu}$, adding S to the independent set, and then for each vertex $v \in L \setminus N(S)$ include v in the independent set with probability $\lambda/(1+\lambda)$ independently. This results in exactly the desired distribution μ_G^{hc}.

To sample from $\overline{\mu}$ we use a polymer model, which we define next. Let R^2 be the graph whose vertices are the vertices of R, where two vertices are adjacent if they are at distance two in G. Note the maximum degree in R^2 is $\Delta_R(\Delta_L - 1)$. We define a polymer γ to be a connected subset of R^2, its neighborhood $N(\gamma)$ to be all vertices in L adjacent to a vertex in γ, and its weight as

$$w_\gamma = \frac{\lambda^{|\gamma|}}{(1+\lambda)^{|N(\gamma)|}} \leq \left(\frac{\lambda}{(1+\lambda)^{\delta_R/\Delta_L}} \right)^{|\gamma|}.$$

Polymers will not be labelled; that is, $q = 1$. Compatible polymer configurations in R^2 correspond exactly to subsets $S \subseteq R$, and the weights w_γ mean a compatible polymer configuration corresponding to subset S has probability exactly $\overline{\mu}(S)$.

To sample from this subset polymer model in R^2, we show that when condition (9) holds, the conditions (3) and (4) of Theorem 1.4 hold, implying the existence of a perfect sampling algorithm with expected running time $O(n \log n)$.

First, when (9) holds, it follows that $e\lambda(\Delta_L - 1)\Delta_R \leq (1+\lambda)^{\delta_R/\Delta_L}$. Then

$$w_\gamma \leq \left(\frac{\lambda}{(1+\lambda)^{\delta_R/\Delta_L}} \right)^{|\gamma|} \leq \left(\frac{1}{e(\Delta_L - 1)\Delta_R} \right)^{|\gamma|},$$

and since $\Delta_R(\Delta_L - 1)$ is the maximum degree in the host graph R^2, condition (3) holds.

From (9) it also follows that

$$\frac{e\lambda\Delta_R(\Delta_L - 1)}{(1+\lambda)^{\delta_R/\Delta_L}} < \frac{e\Delta_R(\Delta_L - 1)}{1 + (1+e)\Delta_R(\Delta_L - 1)}.$$

Because all quantities in this equation are constants that do not depend on the size of the graph, we conclude there exists $\theta \in (0,1)$ such that

$$\frac{e\lambda\Delta_R(\Delta_L - 1)}{(1+\lambda)^{\delta_R/\Delta_L}} \leq \theta \frac{e\Delta_R(\Delta_L - 1)}{1 + (1+e)\Delta_R(\Delta_L - 1)}. \tag{17}$$

Using Lemma 5.2 (which precisely counts the number of rooted subtrees of size k of the Δ-regular tree), the number of polymers of size k that are incompatible with v is at most $(\Delta_R(\Delta_L - 1) + 1)(e\Delta_R(\Delta_L - 1))^{k-1}/k$; recall that $\Delta_R(\Delta_L - 1)$ is the maximum degree of R^2. It follows from this and (17) that

$$\begin{aligned}
\sum_{\gamma \not\sim v} |\gamma| w_\gamma &\leq \sum_{k=1}^{\infty} k \cdot \frac{(\Delta_R(\Delta_L - 1) + 1)(e\Delta_R(\Delta_L - 1))^{k-1}}{k} \cdot \left(\frac{\lambda}{(1+\lambda)^{\delta_R/\Delta_L}} \right)^k \\
&= \frac{\Delta_R(\Delta_L - 1) + 1}{e\Delta_R(\Delta_L - 1)} \sum_{k=1}^{\infty} \left(\frac{e\lambda\Delta_R(\Delta_L - 1)}{(1+\lambda)^{\delta_R/\Delta_L}} \right)^k \\
&\leq \frac{\Delta_R(\Delta_L - 1) + 1}{e\Delta_R(\Delta_L - 1)} \sum_{k=1}^{\infty} \theta \left(\frac{e\Delta_R(\Delta_L - 1)}{1 + (1+e)\Delta_R(\Delta_L - 1)} \right)^k \\
&= \theta \cdot \frac{\Delta_R(\Delta_L - 1) + 1}{e\Delta_R(\Delta_L - 1)} \frac{\frac{e\Delta_R(\Delta_L - 1)}{1+(1+e)\Delta_R(\Delta_L - 1)}}{1 - \frac{e\Delta_R(\Delta_L - 1)}{1+(1+e)\Delta_R(\Delta_L - 1)}} \\
&= \theta.
\end{aligned}$$

Thus, condition (4) also holds, and Theorem 1.4 supplies the perfect sampling algorithm with the desired running time. □

4.2 Potts Model on Expander Graphs

The Q-color Potts model on G at inverse temperature β is a distribution over all (not necessarily proper) Q-colorings of V. Let $\Omega_{G,Q}$ be all colorings $\omega : V \to [Q]$. For a coloring $\omega \in \Omega_{G,Q}$ with $m(G,\omega)$ monochromatic edges, this distribution has

$$\mu_G^{\text{Potts}}(\omega) = \frac{e^{\beta m(G,\omega)}}{Z},$$

where $Z = \sum_{\Omega_{G,q}} e^{\beta m(G,\omega)}$.

For $j \in [Q]$, let $\Omega_{G,Q,j}$ be all colorings in $\Omega_{G,Q}$ such that strictly more than $n/2$ vertices are assigned color j. Let $\overline{\Omega}_{G,Q} := \bigcup_{j\in[Q]} \Omega_{G,Q,j}$ and consider the distribution $\overline{\mu}^{\text{Potts}}$ given by:

$$\overline{\mu}^{\text{Potts}}(\omega) = \frac{e^{\beta m(G,\omega)}}{\hat{Z}} \mathbb{1}(\omega \in \overline{\Omega}_{G,Q}), \text{ where } \hat{Z} = \sum_{\omega\in\overline{\Omega}_{G,Q}} e^{\beta m(G,\omega)}.$$

It follows from [JKP20] that, under suitable conditions, (as we detail next) a perfect sample from $\overline{\mu}^{\text{Potts}}$ is an e^{-n}-approximate sample from μ^{Potts}.

Let $\overline{\mu}_j^{\text{Potts}}$ be the distribution $\overline{\mu}^{\text{Potts}}$ conditioned on being close to the ground state that is entirely color j, that is, conditioned on being in $\Omega_{G,Q,j}$. Because all Q ground states are symmetric, one can sample from $\overline{\mu}^{\text{Potts}}$ by first picking a uniformly random $j \in [Q]$, and then sampling from $\overline{\mu}_j^{\text{Potts}}$. Sampling from $\overline{\mu}_j^{\text{Potts}}$ can be done using the polymer model we define next.

For $\omega \in \Omega_{G,Q,j}$, let $\Gamma(\omega) = \{v \in V : \omega(v) \neq j\}$. Consider a subset polymer model whose host graph is G where a polymer is a graphlet of G whose vertices have colors other than j (there are thus $Q-1$ colors available to color the vertices of a polymer). For a polymer γ, we let $w_\gamma = \exp(-\beta B(\gamma))$ where $B(\gamma)$ is the number of bichromatic edges of the colored subgraph γ plus the number of boundary edges of γ.

There is a bijection between Potts configurations in $\Omega_{G,Q,j}$ and compatible polymer configurations consisting of at most $n/2$ total vertices, where vertices assigned a color other than j are identified. A sampling algorithm for this polymer model can give a sampling algorithm for $\overline{\mu}_j^{\text{Potts}}$, where if the polymer configuration produced has more than $n/2$ total vertices, that configuration is rejected and resampling occurs. As we will see, the probability of needing to resample is small.

Using this polymer model representation, [JKP20] gives an efficient ε-approximate sampling algorithm with for μ_G^{Potts} , derived from an approximate sampling algorithm for $\overline{\mu}_j^{\text{Potts}}$. This algorithm applies to all α-expanding graphs G with maximum degree Δ whenever $\alpha > 0$, $\Delta \geq 3$, $Q \geq 2$, and $\beta > 4\log((Q-1)\Delta)/\alpha$. However, it involves a polymer enumeration step, and so its runtime is (omitting the dependence on other parameters) of the form $n^{O(\log \Delta)}$.

In [CGG+21], the authors take a Markov chain approach and give an ε-approximate sampling algorithm for μ^{Potts}, again via approximate sampling from $\overline{\mu}_j^{\text{Potts}}$, on α-expanding graphs G with maximum degree Δ whenever $\beta \geq \frac{5+3\log((Q-1)\Delta)}{\alpha}$. This has running time $O(n\log(n/\varepsilon)\log(1/\varepsilon))$. In both of these prior works, it must hold that $\varepsilon \geq Qe^{-n}$.

Our algorithm gives an even larger range of β in which $O(n\log n)$ sampling is possible, and removes the dependence on ε. It produces an exact sample for $\overline{\mu}^{\text{Potts}}$ rather than an approximate sample, although this sample is still only an e^{-n}-approximate sample for μ_G^{Potts}.

Proof of Corollary 1.6. From the discussion above, it suffices to generate a perfect sample from $\overline{\mu}_j^{\text{Potts}}$ for any $j \in [Q]$ using the subset polymer model described above. We will show that conditions (3) and (4) of Theorem 1.4 hold, implying the existence of a perfect sampling algorithm for $\overline{\mu}_j^{\text{Potts}}$ with expected running time $O(n \log n)$.

When $\beta \geq \frac{1+\log((Q-1)\Delta)}{\alpha}$, since G is an α expander, $w_\gamma = \exp(-\beta B(\gamma)) \leq \left(\frac{1}{e\Delta(Q-1)}\right)^{|\gamma|} \leq \lambda^{|\gamma|}$ for a suitable $\lambda < \lambda_*(\Delta, Q-1)$, and thus condition (3) is met. Moreover, when (10) holds, then rearranging terms shows

$$e(Q-1)\Delta e^{-\alpha\beta} < \frac{e\Delta}{e\Delta + \Delta + 1}.$$

Because all quantities in this equation are constants that do not depend on the size of the graph, we conclude there exists $\theta \in (0,1)$ such that

$$e(Q-1)\Delta e^{-\alpha\beta} \leq \theta \cdot \frac{e\Delta}{e\Delta + \Delta + 1}.$$

Using the bound $(e\Delta)^{k-1}/k$ for the number of graphlets of size k containing a given vertex, we deduce that the number of polymers incompatible with v of size k is at most $(\Delta+1)(e\Delta)^{k-1}(Q-1)^k/k$. (This bound for the number of graphlets appears in [BI89] but can also be deduced by direct computation from Lemma 5.2 below which provides a tighter bound.) Therefore, we get

$$\begin{aligned}\sum_{\gamma \not\sim v} |\gamma| w_\gamma &\leq \sum_{k=1}^{\infty} k \cdot \frac{(\Delta+1)(e\Delta)^{k-1}(Q-1)^k}{k} e^{-\beta\alpha k} = \frac{\Delta+1}{e\Delta} \sum_{k=1}^{\infty} \left(e\Delta(Q-1)e^{-\beta\alpha}\right)^k \\ &\leq \frac{\Delta+1}{e\Delta} \sum_{k=1}^{\infty} \theta \cdot \left(\frac{e\Delta}{e\Delta+\Delta+1}\right)^k = \theta \cdot \frac{\Delta+1}{e\Delta} \left(\frac{\frac{e\Delta}{e\Delta+\Delta+1}}{1-\frac{e\Delta}{e\Delta+\Delta+1}}\right) = \theta.\end{aligned}$$

Thus, condition (4) holds, and Theorem 1.4 provides a perfect sampling algorithm with the desired running time. □

5 Graphlet sampling: hardness

In this section we establish the sharpness of the threshold $\lambda_*(\Delta, q)$. In particular, we prove Lemmas 1.2 and 1.3 from the introduction. We first prove the following more general variant of Lemma 1.3, which corresponds to the $q = 1$ case.

Lemma 5.1. *The partition function $Z_{G,r,\lambda}$ is finite for every (possibly infinite) graph $G = (V, E)$ of maximum degree Δ and every $r \in V$ if and only if $\lambda \leq \lambda_*(\Delta, q)$. Moreover, $Z_{G,r,\lambda} \leq 40$ when $\lambda \leq \lambda_*(\Delta, q)$.*

The following combinatorial fact will be used in the proof of Lemma 5.1.

Lemma 5.2. *Consider the infinite Δ-regular tree rooted at ρ, and let T_k denote the number of subtrees of size k rooted at ρ. Then for $k \geq 1$:*

$$T_k = \frac{\Delta}{(\Delta-1)k+1} \binom{(\Delta-1)k+1}{k-1}.$$

Proof. For positive integers a and b, let

$$A_k(a,b) = \frac{a}{bk+a}\binom{bk+a}{k}.$$

The numbers $A_k(a,b)$ are a generalization of the Catalan numbers and satisfy the recurrence

$$A_k(a+c,b) = \sum_{i=0}^{k} A_i(a,b)A_{k-i}(c,b); \tag{18}$$

see [Kah13]. It is known (see, e.g., Lemma 1 in [RMŠ21]) that the number of subtrees of size k containing the root of the infinite $(\Delta-1)$-ary tree is

$$\frac{1}{(\Delta-2)k+1}\binom{(\Delta-1)k}{k} = \frac{1}{(\Delta-1)k+1}\binom{(\Delta-1)k+1}{k},$$

which is also equal to $A_k(1,\Delta-1)$. Hence, for $k \geq 2$, we have

$$T_k = \sum_{k_1=0}^{k-1}\sum_{k_2=0}^{k-1-k_1}\cdots\sum_{k_{\Delta-1}=0}^{k-1-(k_1+\cdots+k_{\Delta-2})} A_{k-1-(k_1+\cdots+k_{\Delta-1})}(1,\Delta-1)\cdot\prod_{i=1}^{\Delta-1}A_{k_i}(1,\Delta-1).$$

Using (18), we see that

$$\sum_{k_{\Delta-1}=0}^{k-1-(k_1+\cdots+k_{\Delta-2})} A_{k-1-(k_1+\cdots+k_{\Delta-1})}(1,\Delta-1)A_{k_{\Delta-1}}(1,\Delta-1) = A_{k-1-(k_1+\cdots+k_{\Delta-2})}(2,\Delta-1),$$

and using (18) repeatedly we get for $k \geq 2$

$$\begin{aligned} T_k = A_{k-1}(\Delta,\Delta-1) &= \frac{\Delta}{(\Delta-1)(k-1)+\Delta}\binom{(\Delta-1)(k-1)+\Delta}{k-1} \\ &= \frac{\Delta}{(\Delta-1)k+1}\binom{(\Delta-1)k+1}{k-1}. \end{aligned}$$

The claim also holds trivially for $k=1$ and the result follows. □

We are now ready to provide the proof of Lemma 5.1.

Proof of Lemma 5.1. Let $C_{v,k}$ be the number of graphlets of G with k vertices that contain v. Then, since $f \leq 1$ by assumption, we have

$$Z_{G,r,\lambda} = \sum_{\gamma\in\mathcal{S}(G,r,q)\cup\{\emptyset\}}\lambda^{|\gamma|}f(\gamma) \leq \sum_{k\geq 0}\sum_{\gamma\in\mathcal{S}(G,r,q)\cup\{\emptyset\}:|\gamma|=k}\lambda^k = \sum_{k\geq 0}C_{v,k}\cdot\lambda^k q^k.$$

Consider the infinite Δ-regular tree rooted at ρ, and let T_k be the number of subtrees of size k rooted at ρ so that $C_{v,k} \leq T_k$. (The latter bound captures the key idea in the proof: the infinite Δ-regular tree is the worst case among graphs of maximum degree Δ and the the threshold $\lambda_*(\Delta,q)$ arises as the threshold at which there is a change in the number of fixed points of the corresponding tree recurrence.)

Then, it follows from Lemma 5.2 that

$$Z_{G,r,\lambda} \le 1 + \sum_{k\ge 1} \frac{\Delta}{(\Delta-1)k+1} \binom{(\Delta-1)k+1}{k-1} \lambda^k q^k.$$

We apply the ratio test for the summation on the right hand side and consider the limit:

$$\begin{aligned} L &:= \lim_{k\to\infty} \frac{\frac{\Delta}{(\Delta-1)(k+1)+1}\binom{(\Delta-1)(k+1)+1}{k} q^{k+1}\lambda^{k+1}}{\frac{\Delta}{(\Delta-1)k+1}\binom{(\Delta-1)k+1}{k-1} q^k\lambda^k} \\ &= q\lambda \lim_{k\to\infty} \frac{(\Delta-1)k+1}{(\Delta-1)(k+1)+1} \frac{\binom{(\Delta-1)(k+1)+1}{k}}{\binom{(\Delta-1)k+1}{k-1}}. \end{aligned}$$

We have that

$$A_k := \frac{\binom{(\Delta-1)(k+1)+1}{k}}{\binom{(\Delta-1)k+1}{k-1}} = \frac{1}{k} \cdot \frac{[(\Delta-1)(k+1)+1]!\,[(\Delta-2)k+2]!}{[(\Delta-2)k+\Delta]!\,[(\Delta-1)k+1]!}.$$

Using the inequality $h(n)e^{1/(12n+1)} \le n! \le h(n)e^{1/12n}$ where $h(n) = \sqrt{2\pi}n^{n+1/2}e^{-n}$, and setting

$$B_k = \frac{h((\Delta-1)(k+1)+1)h((\Delta-2)k+2)}{h((\Delta-2)k+\Delta)h((\Delta-1)k+1)}$$

we get

$$\frac{B_k}{k} \frac{e^{\frac{1}{12[(\Delta-1)(k+1)+1]+1}+\frac{1}{12[(\Delta-2)k+2]+1}}}{e^{\frac{1}{12[(\Delta-2)k+\Delta]}+\frac{1}{12[(\Delta-1)k+1]}}} \le A_k \le \frac{B_k}{k} \cdot \frac{e^{\frac{1}{12[(\Delta-1)(k+1)+1]}+\frac{1}{12[(\Delta-2)k+2]}}}{e^{\frac{1}{12[(\Delta-2)k+\Delta]+1}+\frac{1}{12[(\Delta-1)k+1]+1}}}. \tag{19}$$

From a direct calculation, it can be checked that

$$\begin{aligned} B_k =& \frac{1}{e}\Big(1+\frac{\Delta-1}{(\Delta-1)k+1}\Big)^{(\Delta-1)k+1}\Big(1-\frac{\Delta-2}{(\Delta-2)k+\Delta}\Big)^{(\Delta-2)k+\Delta} \\ &\times \frac{[(\Delta-1)(k+1)+1]^{\Delta-1}}{[(\Delta-2)k+2]^{\Delta-2}} \frac{[(\Delta-2)k+2]^{1/2}}{[(\Delta-1)k+1]^{1/2}} \frac{[(\Delta-1)(k+1)+1]^{1/2}}{[(\Delta-2)k+\Delta]^{1/2}}. \end{aligned}$$

Hence,

$$\lim_{k\to\infty} \frac{B_k}{k} = \frac{1}{e}e^{\Delta-1}\frac{1}{e^{\Delta-2}}\frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta-2}} = \frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta-2}}.$$

From this and (19) we deduce that $\lim_{k\to\infty} A_k = \frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta-2}}$, which implies that

$$L = q\lambda\frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta-2}}.$$

From the ratio test we can then conclude that the series converges when $\lambda < \lambda_*(\Delta, q)$ and diverges when $\lambda > \lambda_*(\Delta, q)$.

It remains for us to consider the $\lambda = \lambda_*(\Delta, q)$ case, where

$$Z_{G,r,\lambda} \le A + \sum_{k\ge 2} \frac{\Delta}{(\Delta-1)k+1} \binom{(\Delta-1)k+1}{k-1} \frac{(\Delta-2)^{(\Delta-2)k}}{(\Delta-1)^{(\Delta-1)k}}$$

with $A = 1 + \frac{(\Delta-2)^{(\Delta-2)}}{(\Delta-1)^{(\Delta-1)}} \leq 5/4$. We also have for $k \geq 2$ that

$$\begin{aligned}\binom{(\Delta-1)k+1}{k-1} &\leq \frac{h((\Delta-1)k+1)}{h(k-1)h((\Delta-2)k+2)} \times \frac{e^{\frac{1}{12((\Delta-1)k+1)}}}{e^{\frac{1}{12(k-1)+1}} e^{\frac{1}{12((\Delta-2)k+2)+1}}} \\ &\leq \frac{h((\Delta-1)k+1)}{h(k-1)h((\Delta-2)k+2)}.\end{aligned}$$

Therefore,

$$\begin{aligned}Z_{G,r,\lambda} &\leq \frac{5}{4} + \sum_{k\geq 2} \frac{\Delta}{(\Delta-1)k+1} \frac{h((\Delta-1)k+1)}{h(k-1)h((\Delta-2)k+2)} \frac{(\Delta-2)^{(\Delta-2)k}}{(\Delta-1)^{(\Delta-1)k}} \\ &= \frac{5}{4} + \frac{1}{\sqrt{2\pi}} \sum_{k\geq 2} \frac{\Delta}{(\Delta-1)k+1} \frac{[(\Delta-1)k+1]^{(\Delta-1)k+3/2}}{(k-1)^{k-1/2}[(\Delta-2)k+2]^{(\Delta-2)k+2+1/2}} \frac{(\Delta-2)^{(\Delta-2)k}}{(\Delta-1)^{(\Delta-1)k}} \\ &= \frac{5}{4} + \frac{1}{\sqrt{2\pi}} \sum_{k\geq 2} \frac{\Delta[(\Delta-1)k+1]^{1/2}(k-1)^{1/2}}{[(\Delta-2)k+2]^{5/2}} \left[\frac{((\Delta-1)k+1)^{\Delta-1}(\Delta-2)^{\Delta-2}}{(k-1)((\Delta-2)k+2)^{\Delta-2}(\Delta-1)^{\Delta-1}}\right]^k.\end{aligned}$$

Now, letting $L_1(\Delta,k) = \frac{\Delta[(\Delta-1)k+1]^{1/2}(k-1)^{1/2}}{[(\Delta-2)k+2]^{5/2}}$, $L_2(\Delta,k) = \frac{((\Delta-1)k+1)^{\Delta-1}(\Delta-2)^{\Delta-2}}{(k-1)((\Delta-2)k+2)^{\Delta-2}(\Delta-1)^{\Delta-1}}$, and noting that L_1 is a decreasing function of Δ, we have

$$L_1(\Delta,k) \leq \frac{3(2k+1)^{1/2}(k-1)^{1/2}}{(k+2)^{5/2}} \leq \frac{3\sqrt{2}(k+2)^{1/2}(k+2)^{1/2}}{(k+2)^{5/2}} \leq \frac{3\sqrt{2}}{(k+2)^{3/2}},$$

and,

$$\begin{aligned}L_2(\Delta,k) &= \left(1 - \frac{\Delta}{(\Delta-1)(\Delta-2)k + 2(\Delta-1)}\right)^{\Delta-2} \left(1 + \frac{\Delta}{(\Delta-1)(k-1)}\right) \\ &\leq \exp\left[\frac{\Delta}{\Delta-1}\left(-\frac{\Delta-2}{(\Delta-2)k+2} + \frac{1}{k-1}\right)\right] \\ &\leq \exp\left[\frac{\Delta}{(\Delta-1)(k-1)}\right] \leq \exp\left[\frac{3}{2(k-1)}\right].\end{aligned}$$

Hence,

$$Z_{G,r,\lambda} \leq \frac{5}{4} + \frac{3}{\sqrt{\pi}} \sum_{k\geq 2} \frac{1}{(k+2)^{3/2}} \exp\left[\frac{3k}{2(k-1)}\right] \leq \frac{5}{4} + \frac{3e^3}{\sqrt{\pi}} \sum_{k\geq 2} \frac{1}{(k+2)^{3/2}} \leq 40$$

when $\lambda = \lambda_*(\Delta, q)$. □

We proceed next with the proof of Lemma 1.2. Let us first formally define the notion of a polynomial-time approximate sampler.

Definition 1. *An algorithm $\mathcal{A}$ that takes as input a graph $G = (V, E)$, $\lambda > 0$, a vertex $r \in V$, and an accuracy parameter $\varepsilon \in (0, 1]$ is a polynomial-time approximate sampler for $\nu_{G,r,\lambda}$ if it returns a sample from a distribution $\mu_{\mathcal{A}}$ such that $\|\mu_{\mathcal{A}} - \nu_{G,r,\lambda}\|_{\mathrm{TV}} \leq \varepsilon$ and has a running time that is polynomial in $|V|$ and $1/\varepsilon$. A polynomial-time approximate sampler for $\nu_{G,\lambda}$ is defined analogously.*

We will require the following result.

Lemma 5.3. *Fix $\lambda < 1$, and let $q = 1$ and $f = 1$. There is an algorithm with the following guarantees. The algorithm takes as input a finite graph $G = (V, E)$, a vertex $v \in V$, and parameters $\epsilon, \delta \in (0, 1)$. The algorithm has access to an exact sampling algorithm for $\nu_{G,v,\lambda}$. With probability at least $1 - \delta$ the algorithm outputs a value $\tilde{Z}_{G,v,\lambda}$ such that $(1-\varepsilon)\tilde{Z}_{G,v,\lambda} \leq Z_{G,v,\lambda} \leq (1+\varepsilon)\tilde{Z}_{G,v,\lambda}$. The algorithm's sample complexity (queries to $\nu_{G,v,\lambda}$) and running time are* $\mathrm{poly}(|V|, \varepsilon, \log(1/\delta))$.

Proof. Let $u_1, u_2, \ldots, u_{n-1}$ be the vertices in $V \setminus \{v\}$ in a fixed order we define later. Let G_i be the graph that results from removing vertices $u_1, \ldots, u_i$ from G and let $Z_i = Z_{G_i,v,\lambda}$. We let $Z_0 = Z_{G,v,\lambda}$ and note that $Z_{n-1} = 1 + \lambda$. Then:

$$\frac{1+\lambda}{Z_0} = \prod_{i=0}^{n-2} \frac{Z_{i+1}}{Z_i}.$$

Let $p_i = \frac{Z_{i+1}}{Z_i}$. For each p_i, we will find $\tilde{p}_i$ such that with probability at least $1 - \delta/n$:

$$(1 - \frac{\varepsilon}{4n})\tilde{p}_i \leq p_i \leq (1 + \frac{\varepsilon}{4n})\tilde{p}_i. \tag{20}$$

This implies that

$$(1 - \frac{\varepsilon}{2n})\frac{1}{\tilde{p}_i} \leq \frac{1}{p_i} \leq (1 + \frac{\varepsilon}{2n})\frac{1}{\tilde{p}_i},$$

and so letting $\tilde{Z}_{G,v,\lambda} = (1+\lambda)\prod_{i=0}^{n-2} \frac{1}{\tilde{p}_i}$, we get that

$$(1-\varepsilon)\tilde{Z}_{G,v,\lambda} \leq (1 - \frac{\varepsilon}{2n})^n \tilde{Z}_{G,v,\lambda} \leq Z_0 \leq (1 + \frac{\varepsilon}{2n})^n \tilde{Z}_{G,v,\lambda} \leq (1+\varepsilon)\tilde{Z}_{G,v,\lambda},$$

with probability at least $1 - \delta$ by a union bound.

To obtain (20), we define the sequence of vertices $u_1, u_2, \ldots, u_{n-1}$ such that u_i is a leaf in the breadth-first search tree of G_{i-1} rooted at v. This way, we can conveniently claim that $1 \geq p_i \geq \frac{1}{2}$. To see this, note that $S \in \mathcal{S}(G_{i+1})$ if and only if $S \in \mathcal{S}(G_i) : u_{i+1} \notin S$, so

$$Z_{i+1} = \sum_{S \in \mathcal{S}(G_{i+1})} \lambda^{|S|} = \sum_{S \in \mathcal{S}(G_i) : u_{i+1} \notin S} \lambda^{|S|}. \tag{21}$$

Now, if $S \in \mathcal{S}(G_i)$ and S contains u_{i+1}, then, since u_{i+1} is a leaf in the breadth-first search tree rooted at v, $S \setminus \{u_{i+1}\}$ is a connected subgraph that contains v. That is, every $S \in \mathcal{S}(G_i)$ that contains u_{i+1} can be mapped to a unique subgraph in $\mathcal{S}(G_i)$ that does not contain u_{i+1} of size $|S| - 1$. Hence, since $\lambda < 1$,

$$\sum_{S \in \mathcal{S}(G_i) : u_{i+1} \in S} \lambda^{|S|} \leq \sum_{S \in \mathcal{S}(G_i) : u_{i+1} \notin S} \lambda^{|S|},$$

and so

$$Z_i = \sum_{S \in \mathcal{S}(G_i) : u_{i+1} \in S} \lambda^{|S|} + \sum_{S \in \mathcal{S}(G_i) : u_{i+1} \notin S} \lambda^{|S|} \leq 2 \sum_{S \in \mathcal{S}(G_i) : u_{i+1} \notin S} \lambda^{|S|}.$$

Combined with (21), this gives $p_i = \frac{Z_{i+1}}{Z_i} \geq \frac{1}{2}$. Therefore, to deduce (20), it suffices to find $\tilde{p}_i$ such that

$$\tilde{p}_i - \frac{\varepsilon}{16n} \leq p_i \leq \tilde{p}_i + \frac{\varepsilon}{16n}.$$

For this, we draw L samples from $\nu_{G_i,v,\lambda}$ and let X_j be the indicator random variable for the event that the j-th sample contains u_{i+1}. Letting $\tilde{p}_i = \frac{1}{L}\sum_{i=1}^{L} X_j$, we get from a Chernoff bound that

$$\Pr[|\tilde{p}_i - p_i| \geq \rho p_i] \leq 2e^{-\frac{L\rho^2 p_i}{3}}$$

and setting $\rho = \frac{\varepsilon}{16np_i}$ and $L = 384\frac{n^2}{\varepsilon^2}\log(2n/\delta)$:

$$\Pr\left[|\tilde{p}_i - p_i| \geq \frac{\varepsilon}{16n}\right] \leq 2e^{-\frac{L\rho\varepsilon}{24n}} \leq 2e^{-\frac{L\varepsilon^2}{384n^2}} = \frac{\delta}{n}.$$

In summary, we have provided algorithm that computes $\tilde{p}_i$ such that (20) holds with probability at least $1 - \delta/n$. The algorithm has sample complexity and running time $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$. □

We can now provide the proof of Lemma 1.2.

Proof of Lemma 1.2. It suffices to prove the result for $f = 1$. Let $G = (V, E)$ be a finite graph of maximum degree Δ, and for ease of notation let $\lambda_* = \lambda_*(\Delta, 1)$. We show that if there is a polynomial-time approximate sampler for $\nu_{G,v,\lambda}$ for each $v \in V$ (see Definition 1), then there is also a polynomial-time approximate sampler for for $\nu_{G,\lambda}$. In [RMŠ21], it was shown that there is no polynomial-time approximate sampler for for $\nu_{G,\lambda}$ when $\lambda \in (\lambda_*, 1)$ unless NP=RP, and thus our result follows[2].

First, given exact sampling algorithms for $\nu_{G,v,\lambda}$ for each $v \in V$ with $\mathrm{poly}(n)$ running times, and a desired accuracy parameter $\hat{\varepsilon} \in (0, 1)$, we provide a sampling algorithm whose output distribution is within $\hat{\varepsilon}$ total variation distance from $\nu_{G,\lambda}$. The running time of the algorithm will be $\mathrm{poly}(n, 1/\hat{\varepsilon})$. We then modify the algorithm to use instead the polynomial-time approximate samplers for $\nu_{G,v,\lambda}$ (instead of the exact samplers) and show that this has a negligible effect on its running time and output distribution.

The algorithm is the following:

1. Let $\varepsilon = \min\{\frac{1}{8}, \frac{\hat{\varepsilon}}{6} - \frac{1}{4n}\}$. For each $v \in V$, using the exact sampler for $\nu_{G,v,\lambda}$ and the algorithm from Lemma 5.3, obtain $\tilde{Z}_{G,v,\lambda}$ such that, with probability at least $1 - \frac{1}{n^2 2^n}$,
$$(1-\varepsilon)\tilde{Z}_{G,v,\lambda} \leq Z_{G,v,\lambda} \leq (1+\varepsilon)\tilde{Z}_{G,v,\lambda} \tag{22}$$
2. Set $t = 0$ and choose a vertex $v \in V$ with probability $\frac{\tilde{Z}_{G,v,\lambda}}{\sum_w \tilde{Z}_{G,w,\lambda}}$;
3. Using the exact sampler for $\nu_{G,v,\lambda}$, draw a sample S from $\nu_{G,v,\lambda}$;
4. Accept and output S with probability $1/|S|$;
5. If S is rejected, increase t, and if $t \leq 2n^2$ go to Step 2 and repeat; otherwise output $\emptyset$.

First, we note that by Lemma 5.3, Step 1 can be implemented in $\mathrm{poly}(n, 1/\varepsilon)$. Steps 2 to 5 of the algorithm are repeated at most $O(n^2)$ times, and each of those step takes $\mathrm{poly}(n)$ time. Therefore, the overall running time of the algorithm is $\mathrm{poly}(n, 1/\varepsilon) = \mathrm{poly}(n, 1/\hat{\varepsilon})$. Moreover, (22) holds for every $v \in V$ with probability at least $1 - \frac{1}{n2^n}$ by a union bound.

[2] We note that the hardness result in [RMŠ21] is stated for the harder problem of producing approximate samples with a running time depending polynomially on $\log(1/\varepsilon)$, but the proofs in [RMŠ21] extend to our setting without modification.

Let μ_{ALG} be the output distribution of the algorithm, and let μ^*_{ALG} be the output distribution when (i) Step 1 succeeds in finding the required approximations and (ii) a subgraph is accepted in Step 4 before $t > 2n^2$. We show next that $\|\mu^*_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq \hat{\varepsilon}$. Let $\tilde{Z} = \sum_{v \in V} \tilde{Z}_{G,v,\lambda}$ and let $\mathcal{S} = \mathcal{S}(G) \cup \{\emptyset\}$. The probability that the algorithm outputs $S \in \mathcal{S}$ in Step 4 in a give iteration is:

$$\sum_{v \in S} \frac{\tilde{Z}_{G,v,\lambda}}{\tilde{Z}} \nu_{G,v,\lambda}(S) \frac{1}{|S|} =: \frac{\phi(S)}{\tilde{Z}}. \tag{23}$$

Then, $\mu^*_{\text{ALG}}(S) = \frac{\phi(S)}{\sum_{\hat{S} \in \mathcal{S}} \phi(\hat{S})}$, and

$$\begin{aligned}
\|\mu^*_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} &= \frac{1}{2} \sum_{S \in \mathcal{S}} \Big| \frac{\phi(S)}{\sum_{W \in \mathcal{S}} \phi(W)} - \frac{\lambda^{|S|}}{Z_{G,\lambda}} \Big| = \frac{1}{2} \sum_{S \in \mathcal{S}} \Big| \frac{\sum_{v \in S} \nu_{G,v,\lambda}(S) \tilde{Z}_{G,v,\lambda}}{|S| \sum_{W \in \mathcal{S}} \phi(W)} - \frac{\lambda^{|S|}}{Z_{G,\lambda}} \Big| \\
&\leq \frac{1}{2} \sum_{S \in \mathcal{S}} \sum_{v \in S} \Big| \frac{\nu_{G,v,\lambda}(S) \tilde{Z}_{G,v,\lambda}}{|S| \sum_{W \in \mathcal{S}} \phi(W)} - \frac{\lambda^{|S|}}{|S| Z_{G,\lambda}} \Big| \\
&= \frac{1}{2} \sum_{S \in \mathcal{S}} \sum_{v \in S} \frac{Z_{G,v,\lambda}}{|S| Z_{G,\lambda}} \Big| \frac{\nu_{G,v,\lambda}(S) \cdot \tilde{Z}_{G,v,\lambda}}{Z_{G,v,\lambda}} \cdot \frac{Z_{G,\lambda}}{\sum_{W \in \mathcal{S}} \phi(W)} - \frac{\lambda^{|S|}}{Z_{G,v,\lambda}} \Big| \\
&= \frac{1}{2} \sum_{S \in \mathcal{S}} \sum_{v \in S} \frac{\nu_{G,v,\lambda}(S) \cdot Z_{G,v,\lambda}}{|S| Z_{G,\lambda}} \Big| \frac{\tilde{Z}_{G,v,\lambda}}{Z_{G,v,\lambda}} \cdot \frac{Z_{G,\lambda}}{\sum_{W \in \mathcal{S}} \phi(W)} - 1 \Big|.
\end{aligned} \tag{24}$$

From (22) we get

$$1 - 2\varepsilon \leq \frac{1}{1+\varepsilon} \leq \frac{\tilde{Z}_{G,v,\lambda}}{Z_{G,v,\lambda}} \leq \frac{1}{1-\varepsilon} \leq 1 + 2\varepsilon,$$

since $\varepsilon \leq 1/8$. Moreover,

$$\sum_{W \in \mathcal{S}} \phi(W) = \sum_{W \in \mathcal{S}} \sum_{v \in W} \frac{\nu_{G,v,\lambda}(W) \tilde{Z}_{G,v,\lambda}}{|W|} \leq (1+2\varepsilon) \sum_{W \in \mathcal{S}} \sum_{v \in W} \frac{\lambda^{|W|}}{|W|} = (1+2\varepsilon) Z_{G,\lambda},$$

and similarly we can obtain that $\sum_{W \in \mathcal{S}} \phi(W) \geq (1-2\varepsilon) Z_{G,\lambda}$. Combining these bounds we get:

$$\Big| \frac{\tilde{Z}_{G,v,\lambda}}{Z_{G,v,\lambda}} \cdot \frac{Z_{G,\lambda}}{\sum_{W \in \mathcal{S}} \phi(W)} - 1 \Big| \leq \frac{4\varepsilon}{1-2\varepsilon} \leq 8\varepsilon$$

Plugging this into (24) we deduce that:

$$\|\mu^*_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq 4\varepsilon \sum_{S \in \mathcal{S}} \sum_{v \in S} \frac{\nu_{G,v,\lambda}(S) \cdot Z_{G,v,\lambda}}{|S| Z_{G,\lambda}} = \frac{4\varepsilon}{Z_{G,\lambda}} \sum_{S \in \mathcal{S}} \sum_{v \in S} \frac{\lambda^{|S|}}{|S|} \leq 4\varepsilon. \tag{25}$$

Let $\mathcal{E}_1$ be the event that Step 1 succeeds in finding the approximations, let $\mathcal{E}_2$ the event that the algorithm accepts and outputs a graphlet in Step 4, and let $\mathcal{E}$ be the event that both $\mathcal{E}_1$ and $\mathcal{E}_2$ occur. Then, the triangle inequality and (25) imply:

$$\|\mu_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq \|\mu^*_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} + \|\mu^*_{\text{ALG}} - \mu_{\text{ALG}}\|_{\text{TV}} \leq 4\varepsilon + \|\mu^*_{\text{ALG}} - \mu_{\text{ALG}}\|_{\text{TV}}. \tag{26}$$

We proceed to bound $\|\mu^*_{\text{ALG}} - \mu_{\text{ALG}}\|_{\text{TV}}$.

$$\begin{aligned}
\|\mu^*_{\text{ALG}} - \mu_{\text{ALG}}\|_{\text{TV}} &= \|\mu_{\text{ALG}} - \mu_{\text{ALG}}(\cdot \mid \mathcal{E})\|_{\text{TV}} \leq \mu_{\text{ALG}}(\neg \mathcal{E}) \\
&\leq \mu_{\text{ALG}}(\neg \mathcal{E}_1) + \mu_{\text{ALG}}(\neg \mathcal{E}_2 \mid \mathcal{E}_1) \leq \frac{1}{2n} + \mu_{\text{ALG}}(\neg \mathcal{E}_2 \mid \mathcal{E}_1).
\end{aligned} \tag{27}$$

So it remains for us to bound $\mu_{\text{ALG}}(\neg\mathcal{E}_2 \mid \mathcal{E}_1)$. For this, note that the probability that a subgraph is accepted in Step 4 in an iteration is:

$$\sum_{S\in\mathcal{S}}\sum_{v\in S}\frac{\tilde{Z}_{G,v,\lambda}}{\tilde{Z}}\nu_{G,v,\lambda}(S)\frac{1}{|S|} \geq \frac{1-2\varepsilon}{\tilde{Z}}\sum_{S\in\mathcal{S}}\sum_{v\in S}\frac{\lambda^{|S|}}{|S|} \geq \frac{(1-2\varepsilon)Z_{G,\lambda}}{\tilde{Z}}.$$

Since

$$\tilde{Z} = \sum_{v\in V}\tilde{Z}_{G,v,\lambda} \leq (1+2\varepsilon)\sum_{v\in V}Z_{G,v,\lambda} \leq (1+2\varepsilon)nZ_{G,\lambda},$$

we get the acceptance probability in Step 4 in an iteration is at least

$$\frac{1-2\varepsilon}{(1+2\varepsilon)n} \geq \frac{1-4\varepsilon}{n} \geq \frac{1}{2n},$$

provided $\mathcal{E}_1$ occurs. Therefore, if X is geometric random variable with parameter $1/(2n)$, we have

$$\mu_{\text{ALG}}(\neg\mathcal{E}_2 \mid \mathcal{E}_1) \leq \Pr[X \geq 2n^2] \leq \left(1-\frac{1}{2n}\right)^{2n^2} \leq \frac{1}{e^n},$$

and plugging this bound into (27) and (26) we obtain:

$$\|\mu_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq 4\varepsilon + \frac{3}{2n}.$$

We have established so far the hardness of exactly sampling from $\nu_{G,v,\lambda}$ for $\lambda \in (\lambda_*, 1)$. The same reduction (i.e., algorithm) with minor adjustments works for approximate sampling. Suppose that in Step 3 of the algorithm, we instead generate $T = 2n^2$ samples from a distribution μ_v such that $\|\mu_v - \nu_{G,v,\lambda}\|_{\text{TV}} \leq \frac{\varepsilon}{n^2}$. Let $\mu_v^{\otimes T}$ and $\nu_{G,v,\lambda}^{\otimes T}$ be the product distributions corresponding to T independent samples from μ_v and $\nu_{G,v,\lambda}$ respectively. We have

$$\|\mu_v^{\otimes T} - \nu_{G,v,\lambda}^{\otimes T}\|_{\text{TV}} \leq T\|\mu_v - \nu_{G,v,\lambda}\|_{\text{TV}} \leq \frac{T\varepsilon}{n^2} \leq 2\varepsilon. \tag{28}$$

Let $\tilde{\mu}_{\text{ALG}}$ be the output distribution of the algorithm when using samples from μ_v in Step 3. Then:

$$\|\tilde{\mu}_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq \|\tilde{\mu}_{\text{ALG}} - \mu_{\text{ALG}}\|_{\text{TV}} + \|\mu_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq \|\tilde{\mu}_{\text{ALG}} - \mu_{\text{ALG}}\|_{\text{TV}} + 4\varepsilon + \frac{3}{2n}.$$

Consider the following coupling between $\tilde{\mu}_{\text{ALG}}$ and μ_{ALG}: use the same randomness for Steps 1, 2 and 5 and the optimal coupling for $\mu_v^{\otimes T}$ and $\nu_{G,v,\lambda}^{\otimes T}$ for Step 3. Then, from (28), we get

$$\|\tilde{\mu}_{\text{ALG}} - \nu_{G,\lambda}\|_{\text{TV}} \leq 6\varepsilon + \frac{3}{2n} \leq \hat{\varepsilon}.$$

That is, we obtain a polynomial-time approximate sampler for $\nu_{G,\lambda}$ which completes the reduction. □

Acknowledgements

This work was carried out as part of the AIM SQuaRE workshop "Connections between computational and physical phase transitions." We thank Tyler Helmuth, Alexandre Stauffer, and Izabella Stuhl for many helpful conversations. WP is supported in part by NSF grant DMS-2309958 and CCF-2309708. SC is supported in part by NSF grants DMS-1803325 and CCF-2104795. AB is supported in part by NSF grant CCF-2143762. We also thank the anonymous referees for their many insightful corrections and suggestions.

References

[ABH19] Matteo Agostini, Marco Bressan, and Shahrzad Haddadan. Mixing time bounds for graphlet random walks. *Information Processing Letters*, 152:105851, 2019.

[AGPP23] Konrad Anand, Andreas Göbel, Marcus Pappik, and Will Perkins. Perfect sampling for hard spheres from strong spatial mixing. In *International Conference on Randomization and Computation (RANDOM 2023)*, 2023.

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

[ALOG21] Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *SIAM Journal on Computing*, (0):FOCS20–1, 2021.

[BCH+20] Christian Borgs, Jennifer Chayes, Tyler Helmuth, Will Perkins, and Prasad Tetali. Efficient sampling and counting algorithms for the Potts model on $\mathbb{Z}^d$ at all temperatures. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 738–751, 2020.

[BCK+17] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Counting graphlets: Space vs time. In *Proceedings of the tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 557–566, 2017.

[BCK+18] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Motif counting beyond five nodes. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):1–25, 2018.

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

[BGP07] Kim Baskerville, Peter Grassberger, and Maya Paczuski. Graph animals, subgraph sampling, and motif search in large networks. *Physical Review E*, 76(3):036107, 2007.

[BI89] Christian Borgs and John Z Imbrie. A unified approach to phase diagrams in field theory and statistical mechanics. *Communications in mathematical physics*, 123(2):305–328, 1989.

[BLP21] Marco Bressan, Stefano Leucci, and Alessandro Panconesi. Faster motif counting via succinct color coding and adaptive sampling. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(6):1–27, 2021.

[Bre21] Marco Bressan. Efficient and near-optimal algorithms for sampling connected subgraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1132–1143, 2021.

[BRRAH12] Mansurul A Bhuiyan, Mahmudur Rahman, Mahmuda Rahman, and Mohammad Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *2012 IEEE 12th International Conference on Data Mining*, pages 91–100. IEEE, 2012.

[CDK20a] Charles Carlson, Ewan Davies, and Alexandra Kolla. Efficient algorithms for the Potts model on small-set expanders. *arXiv preprint arXiv:2003.01154*, 2020.

[CDK+20b] Matthew Coulson, Ewan Davies, Alexandra Kolla, Viresh Patel, and Guus Regts. Statistical physics approaches to unique games. In *35th Computational Complexity Conference (CCC)*, 2020.

[CGG+21] Zongchen Chen, Andreas Galanis, Leslie A Goldberg, Will Perkins, James Stewart, and Eric Vigoda. Fast algorithms at low temperatures via Markov chains. *Random Structures & Algorithms*, 58(2):294–321, 2021.

[CGŠV22] Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Sampling colorings and independent sets of random regular bipartite graphs in the non-uniqueness region. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2198–2207. SIAM, 2022.

[CLV20] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (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 *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1537–1550, 2021.

[CLWL16] Xiaowei Chen, Yongkun Li, Pinghui Wang, and John CS Lui. A general framework for estimating graphlet statistics via random walk. *Proceedings of the VLDB Endowment*, 10(3), 2016.

[CP20] Sarah Cannon and Will Perkins. Counting independent sets in unbalanced bipartite graphs. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1456–1466, 2020.

[DGGJ04] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004.

[FGKP23] Tobias Friedrich, Andreas Göbel, Martin S Krejca, and Marcus Pappik. Polymer dynamics via cliques: New conditions for approximations. *Theoretical Computer Science*, 942:230–252, 2023.

[FV17] Sacha Friedli and Yvan Velenik. *Statistical mechanics of lattice systems: a concrete mathematical introduction*. Cambridge University Press, 2017.

[GGS21] Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast algorithms for general spin systems on bipartite expanders. *ACM Transactions on Computation Theory (TOCT)*, 13(4):1–18, 2021.

[GGS22] Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast mixing via polymers for random graphs with unbounded degree. *Information and Computation*, page 104894, 2022.

[GK71] Christian Gruber and Hervé Kunz. General properties of polymer systems. *Communications in Mathematical Physics*, 22(2):133–161, 1971.

[GK07] Joshua A Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.

[GR97] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC)*, pages 406–415, 1997.

[GŠV16] Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic ising and hard-core models. *Combinatorics, Probability and Computing*, 25(4):500–559, 2016.

[HJP22] Tyler Helmuth, Matthew Jenssen, and Will Perkins. Finite-size scaling, phase coexistence, and algorithms for the random cluster model on random graphs. *Annales de l'Institut Henri Poincaré B. To appear*, 2022.

[HN99] Olle Haggstrom and Karin Nelander. On exact simulation of Markov random fields using coupling from the past. *Scandinavian Journal of Statistics*, 26(3):395–411, 1999.

[HPR20] Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov–Sinai theory. *Probability Theory and Related Fields*, 176(3):851–895, 2020.

[Hub04] Mark Huber. Perfect sampling using bounding chains. *The Annals of Applied Probability*, 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 *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 147–158. IEEE, 2022.

[HWY23] Kun He, Kewen Wu, and Kuan Yang. Improved bounds for sampling solutions of random CNF formulas. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3330–3361. SIAM, 2023.

[JKP20] Matthew Jenssen, Peter Keevash, and Will Perkins. Algorithms for #BIS-hard problems on expander graphs. *SIAM Journal on Computing*, 49(4):681–710, 2020.

[JPP22] Matthew Jenssen, Aditya Potukuchi, and Will Perkins. Approximately counting independent sets in bipartite graphs via graph containers. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 499–516. SIAM, 2022.

[JRL11] Svante Janson, Andrzej Rucinski, and Tomasz Luczak. *Random graphs*. John Wiley & Sons, 2011.

[JSP15] Madhav Jha, C Seshadhri, and Ali Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 495–505, 2015.

[Kah13] Reza Kahkeshani. A generalization of the Catalan numbers. *Journal of Integer Sequences*, 16(2):3, 2013.

[KIMA04] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[KK19] Timo Kötzing and Martin S Krejca. First-hitting times under drift. *Theoretical Computer Science*, 796:51–69, 2019.

[KP86] Roman Koteckỳ and David Preiss. Cluster expansion for abstract polymer models. *Communications in Mathematical Physics*, 103(3):491–498, 1986.

[LB12] Xuesong Lu and Stéphane Bressan. Sampling connected induced subgraphs uniformly at random. In *International Conference on Scientific and Statistical Database Management*, pages 195–212. Springer, 2012.

[LLLM19] Chao Liao, Jiabao Lin, Pinyan Lu, and Zhenyu Mao. Counting independent sets and colorings on random regular bipartite graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[MG20] Ryuta Matsuno and Aristides Gionis. Improved mixing time for k-subgraph sampling. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 568–576. SIAM, 2020.

[PSS19] Kirill Paramonov, Dmitry Shemetov, and James Sharpnack. Estimating graphlet statistics via lifting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 587–595, 2019.

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

[RMŠ21] Andrew Read-McFarland and Daniel Štefankovič. The hardness of sampling connected subgraphs. In *Latin American Symposium on Theoretical Informatics*, pages 464–475. Springer, 2021.

[Sly10] Allan Sly. Computational transition at the uniqueness threshold. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 287–296. IEEE, 2010.

[SS12] Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on d-regular graphs. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 361–369. IEEE, 2012.

[SVP+09] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.

[SY05] Nacu Serban and Peres Yuval. Fast simulation of new coins from old. *The Annals of Applied Probability*, 15(1A):93 – 115, 2005.

[Wei06] Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149, 2006.