Parameter Estimation for Gibbs Distributions

David G. Harris
Department of Computer Science, University of Maryland, College Park, MD, USA

Vladimir Kolmogorov
Institute of Science and Technology Austria, Klosterneuburg, Austria

Abstract

A central problem in computational statistics is to convert a procedure for *sampling* combinatorial objects into a procedure for *counting* those objects, and vice versa. We will consider sampling problems which come from *Gibbs distributions*, which are families of probability distributions over a discrete space Ω with probability mass function of the form $\mu_\beta^\Omega(\omega) \propto e^{\beta H(\omega)}$ for β in an interval $[\beta_{\min}, \beta_{\max}]$ and $H(\omega) \in \{0\} \cup [1, n]$.

The *partition function* is the normalization factor $Z(\beta) = \sum_{\omega \in \Omega} e^{\beta H(\omega)}$, and the *log partition ratio* is defined as $q = \frac{\log Z(\beta_{\max})}{Z(\beta_{\min})}$

We develop a number of algorithms to estimate the counts c_x using roughly $\tilde{O}(\frac{q}{\varepsilon^2})$ samples for general Gibbs distributions and $\tilde{O}(\frac{n^2}{\varepsilon^2})$ samples for integer-valued distributions (ignoring some second-order terms and parameters), We show this is optimal up to logarithmic factors. We illustrate with improved algorithms for counting connected subgraphs and perfect matchings in a graph.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Applied computing → Physics

Keywords and phrases Gibbs distribution, sampling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.72

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2007.10824`

Acknowledgements We thank Heng Guo for helpful explanations of algorithms for sampling connected subgraphs and matchings, Maksym Serbyn for bringing to our attention the Wang-Landau algorithm and its use in physics.

1 Introduction

A central problem in computational statistics is to convert a procedure for *sampling* combinatorial objects into a procedure for *counting* those objects, and vice versa. We will consider sampling algorithms for Gibbs distributions. Formally, given a real-valued function $H(\cdot)$ over a finite set Ω, the *Gibbs distribution* is defined as a family of distributions μ_β^Ω over Ω, parameterized by β, of the form

$$\mu_\beta^\Omega(\omega) = \frac{e^{\beta H(\omega)}}{Z(\beta)}$$

These distributions occur in a number of sampling algorithms, as we describe shortly; they also frequently occur in physics, where the parameter $-\beta$ corresponds to the inverse temperature, the function $H(\omega)$ is called the *Hamiltonian* of the system, and the normalizing constant $Z(\beta) = \sum_{\omega \in \Omega} e^{\beta H(\omega)}$ is called the *partition function*.

Suppose we have access to an oracle which return a sample from μ_β^Ω for any chosen query value $\beta \in [\beta_{\min}, \beta_{\max}]$. We will seek to estimate the vector of counts (also known as the *(discrete) density of states (DOS)*), defined as

$$c_x = |H^{-1}(x)|, \qquad x \geq 0$$

In statistical physics, for instance, this essentially gives full information about the system and physically relevant quantities such as entropy, free energy, etc. Another parameter, whose role is less intuitive, is the partition ratio function

$$Q(\beta) = \frac{Z(\beta)}{Z(\beta_{\min})},$$

and in particular the value $Q(\beta_{\max}) = \frac{Z(\beta_{\max})}{Z(\beta_{\min})}$. This can interpreted as a measure of the "diversity" of the distribution as β varies.

As is common in this setting, we assume that (after rescaling if necessary) we are given known parameters n, q with

$$\log Q(\beta_{\max}) \leq q, \qquad H(\Omega) \subseteq \mathcal{F} \stackrel{\text{def}}{=} \{0\} \cup [1, n]$$

In some cases, the domain is integer-valued, i.e. $H(\Omega) \subseteq \mathcal{H} \stackrel{\text{def}}{=} \mathcal{F} \cap \mathbb{Z} = \{0, 1, \ldots, n\}$ for integer n. We call this the *general integer setting*. A special case of the integer setting, which we call the *log-concave setting*, is when the counts $c_0, c_1, c_2, \ldots, c_{n-1}, c_n$ are non-zero and satisfy $c_k/c_{k-1} \geq c_{k+1}/c_k$ for $k = 1, \ldots, n-1$. The general case, where $H(\omega)$ takes values in $\mathcal{F}$, is called the *continuous setting*.[1]

There is an associated probability distribution we call the *gross Gibbs distribution* $\mu_\beta(x)$ over $\mathcal{F}$ given by

$$\mu_\beta(x) = \frac{c_x e^{\beta x}}{Z(\beta)}, \qquad Z(\beta) = \sum_x c_x e^{\beta x}$$

We will only require oracle access to μ_β, for any chosen query value $\beta \in [\beta_{\min}, \beta_{\max}]$; this is provided automatically given access to μ_β^Ω. We let γ denote the target failure probability and ε the target accuracy of our algorithms, i.e. with probability at least $1-\gamma$, the algorithms should return estimates within a factor of $[e^{-\varepsilon}, e^{\varepsilon}]$ of the correct value. Throughout, "sample complexity" refers to the number of calls to the oracle; for brevity, we also define the *cost* of a sampling algorithm to be its *expected sample complexity*.

To avoid degenerate cases, we assume $n, q \geq 2$ and $\varepsilon, \gamma \in (0, \frac{1}{2})$ throughout. If upper bounds n and/or q are not available directly, they can often be estimated by simple algorithms (up to constant factors), or can be guessed by exponential back-off strategies.

1.1 Algorithmic sampling-to-counting

To make our problem setting more concrete, consider the following scenario: we have a combinatorial system, where the objects have a "weight", and we have an algorithm to sample objects from the corresponding Gibbs distribution for any parameter β. This may be an exact sampler, or it may be an approximate sampler such as a Markov chain whose stationary distribution is the Gibbs distribution. The runtime (e.g. the mixing time of the Markov chain) may depend on β. As a few prominent examples:

[1] The log-concave algorithms still work if some of the counts c_i are equal to zero; in this case, the non-zero counts must form a discrete interval $\{i_0, i_0+1, \ldots, i_1-1, i_1\}$ and the required bound must hold for $k = i_0+1, \ldots, i_1-1$.

1. Connected subgraphs of a given graph; the weight of a subgraph is its cardinality [10].
2. Matchings of a given graph; the weight of a matching, again, is its cardinality [15, 13].
3. Independent sets in bounded-degree graphs; the weight is the size of the independent set [7, 13].
4. Assignments to a given k-SAT instance; the weight is the number of unsatisfied clauses [8].
5. Vertex cuts for the ferromagnetic Ising model; the weight is the imbalance of the cut [5].

We may wish to know the number of objects of a given weight class, e.g. connected subgraphs of a given size. This can be viewed in terms of estimating the counts c_i. In a number of these applications, such as connected subgraphs and matchings, the count sequence is further known to be log-concave.

Our estimation algorithms can be combined with these prior sampling algorithms to yield improved algorithmic results, essentially for free. As some examples, we will show the following:

▶ **Theorem 1.** *Let $G = (V, E)$ be a connected graph, and for each $i = 0, \dots, |E| - |V| + 1$ let c_i be the number of connected subgraphs with $|E| - i$ edges. There is an fully-polynomial randomized approximation scheme (FPRAS) to estimate all values c_i in time complexity $\tilde{O}(|E|^3|V|/\varepsilon^2)$.*

▶ **Theorem 2.** *Let $G = (V, E)$ be a graph of maximum degree D and for each $i = 0, \dots, |V|$ let c_i be the number of independent sets of size i. For any constant $\xi > 0$ there is an FPRAS with runtime $\tilde{O}(|V|^2/\varepsilon^2)$ to simultaneously estimate all values $c_0, \dots, c_t$ for $t = (\alpha_c - \xi)|V|$, where α_c is the computational hardness threshold shown in [7].*

▶ **Theorem 3.** *Let $G = (V, E)$ be a graph with $|V| = 2v$ and for each $i = 0, \dots, v$ let c_i be the number of matchings in G with i edges. Suppose $c_v > 0$ and $c_{v-1}/c_v \le f$ for a known parameter f. There is an FPRAS for all c_i running in time $\tilde{O}(|E||V|^3 f/\varepsilon^2)$. In particular, if G has minimum degree at least $|V|/2$, the time complexity is $\tilde{O}(|V|^7/\varepsilon^2)$.*

Theorem 1 improves by a factor of $|E|$ over the algorithm in [11]. Similarly, Theorem 3 improves by a factor of $|V|$ compared to the FPRAS for counting matchings in [15]. Theorem 2 matches the runtime of an FPRAS for a *single* value i_k given in [13].

There are two minor technical issues we should clarify here. First, to obtain a randomized estimation algorithm, we must also bound the computational complexity of our procedures in addition to the number of oracle calls. In all the algorithms we develop, the computational complexity is a small logarithmic factor times the query complexity. The computational complexity of the oracle is typically much larger than this overhead. Thus, our sampling procedures translate directly into efficient randomized algorithms, whose runtime is the expected sample complexity multiplied by the oracle's computational complexity. We will not comment on computational issues henceforth.

Second, we may only have access to some approximate oracle $\tilde{\mu}_\beta$ that is close to μ_β in terms of total variation distance (e.g. by running an MCMC sampler). By a standard coupling argument (see e.g. [19, Remark 5.9]), our results remain valid if exact oracles are replaced with sufficiently close approximate oracles.

1.2 Our contributions

Before we can formally describe our algorithm for count estimation, we need to clear up two technical issues. The first is that counts can only be recovered up to scaling, so some (arbitrary) normalization must be chosen. For sake of consistency with other algorithms, we use the parameter $\pi(x)$ defined as:

$$\pi(x) \stackrel{\text{def}}{=} \mu_{\beta_{\min}}(x) = \frac{c_x e^{\beta_{\min} x}}{Z(\beta_{\min})}$$

The second, much trickier, issue is that if a count c_x is relatively small, then it is inherently hard to estimate accurately. To explain this, suppose that $\max_{\beta \in [\beta_{\min}, \beta_{\max}]} \mu_\beta(x) = \mu_*$. In this case, $\Omega(1/\mu_*)$ samples are clearly needed to distinguish between $c_x = 0$ and $c_x > 0$; with fewer samples, we will never draw x from the oracle. Moreover, $\Omega(\frac{1}{\mu_* \varepsilon^2})$ samples are needed to estimate c_x to relative error ε. Since we can vary β, the complexity of estimating c_x must depend on the *best case* $\mu_\beta(x)$, over all allowed values of β. This gives rise to the parameter $\Delta(x)$ defined as

$$\Delta(x) \stackrel{\text{def}}{=} \max_{\beta \in [\beta_{\min}, \beta_{\max}]} \mu_\beta(x)$$

With these two provisos, let us define the problem $P^{\delta,\varepsilon}_{\texttt{count}}$ for parameters $\delta, \varepsilon \in (0,1)$ as follows. We seek to obtain a pair of vectors $(\hat{\pi}, u)$, to satisfy two properties:

- **(i)** for all $x \in \mathcal{F}$ with $c_x \neq 0$, there holds $|\hat{\pi}(x) - \pi(x)| \leq u(x) \leq \varepsilon \pi(x)(1 + \delta/\Delta(x))$.
- **(ii)** for all $x \in \mathcal{F}$ with $c_x = 0$, there holds $\hat{\pi}(x) = 0$, and $u(x)$ can be set to an arbitrary value.

In other words, $[\hat{\pi}(x) - u(x), \hat{\pi}(x) + u(x)]$ should be a confidence interval for $\pi(x)$. In particular, if $\Delta(x) \geq \delta$, then $P^{\delta,\varepsilon}_{\texttt{count}}$ provides a $(1 \pm O(\varepsilon))$ relative approximation to $\pi(x)$. When $\Delta(x) \ll \delta$, then it still provides meaningful approximation guarantees which are critical in some of our other algorithms.

We develop three main algorithmic results:

▶ **Theorem 4.** *$P^{\delta,\varepsilon}_{\texttt{count}}$ can be solved with the following complexities:*

- *In the continuous setting, with cost $O\Big(\frac{q \log n + \sqrt{q \log n}/\delta}{\varepsilon^2} \log \frac{q}{\delta\gamma}\Big)$.*
- *In the general integer setting, with cost $O\Big(\frac{n^2 + n/\delta}{\varepsilon^2} \log^2 \frac{nq}{\gamma}\Big)$.*
- *In the log-concave setting, with cost $O\Big(\frac{\min\{(q+n)\log n, n^2\} + 1/\delta}{\varepsilon^2} \log \frac{nq}{\gamma}\Big)$.*

where recall that cost refers to the expected number of queries to the oracle.

Our full results are somewhat more precise, see Theorems 13, 20 and 21 for more details.

We also show lower bounds for $P^{\delta,\varepsilon}_{\texttt{count}}$; we summarize these results here as follows:

▶ **Theorem 5.** *Let $n \geq n_0, q \geq q_0, \varepsilon < \varepsilon_0, \delta < \delta_0, \gamma < 1/4$ for certain absolute constants $n_0, q_0, \varepsilon_0, \delta_0$. There are problem instances μ which satisfy the given bounds n and q such that:*

- **(a)** *$P^{\delta,\varepsilon}_{\texttt{count}}$ requires cost $\Omega(\frac{(q + \sqrt{q}/\delta)\log\frac{1}{\gamma}}{\varepsilon^2})$.*
- **(b)** *$P^{\delta,\varepsilon}_{\texttt{count}}$ requires cost $\Omega(\frac{\min\{q + \sqrt{q}/\delta, n^2 + n/\delta\}\log\frac{1}{\gamma}}{\varepsilon^2})$, and μ is integer-valued.*
- **(c)** *$P^{\delta,\varepsilon}_{\texttt{count}}$ requires cost $\Omega(\frac{(1/\delta + \min\{q, n^2\})\log\frac{1}{\gamma}}{\varepsilon^2})$, and μ is log-concave.*

These first two results match Theorem 4 up to logarithmic factors in n and q. The result for the log-concave setting has an additive discrepancy $\tilde{O}(\frac{n}{\varepsilon^2})$ in the regime when $1/\delta + q = o(n)$. (Throughout, we use the notation $\tilde{O}(x) = x \operatorname{polylog}(x)$.) See Theorem 36 for a more precise and general statement of these bounds. We emphasize that these lower bounds only apply to estimation algorithms which make use of the Gibbs oracle in a black-box way.

Some count-estimation algorithms have been considered for specific problems, e.g. in [14] for counting matchings or in [7] for counting independent sets. These procedures depended on specific properties of the Gibbs distribution, e.g. log-concavity. In addition, the algorithm in [14] was roughly worse by a factor of n compared to Theorem 4. By swapping in our new algorithm for $P_{\texttt{count}}^{\delta,\varepsilon}$, we will immediately obtain simpler, and more efficient, algorithms for these problems.

The general problem $P_{\texttt{count}}$ has not been theoretically analyzed, to our knowledge. In practice, the *Wang-Landau (WL)* algorithm [20] is a popular heuristic to estimate counts in physical applications. This uses a completely different methodology from our algorithm, based on a random walk on $\mathcal{F}$ with a running count estimate $\hat{c}$. As discussed in [18], there are more than 1500 papers on the WL algorithm as well as variants such as the $1/t$-WL algorithm [3]. These algorithms are not well understood; some variants are guaranteed to converge asymptotically [9], but bounds on convergence rate or accuracy seem to be lacking. For a representative example, see for example [17], which describes a Gibbs distribution model of protein folding, and uses the WL algorithm to determine relevant properties.

Estimating partition ratio

As a key building block, we develop new subroutines to estimate partition ratios. Formally, let us define the problem $P_{\texttt{ratio}}^{\texttt{all}}$ to compute a data structure $\mathcal{D}$ with an associated *deterministic* function $\hat{Q}(\alpha|\mathcal{D})$ satisfying the property

$$|\log \hat{Q}(\alpha|\mathcal{D}) - \log Q(\alpha)| \leq \varepsilon \qquad \text{for all } \alpha \in (\beta_{\min}, \beta_{\max}]$$

We say in this case that $\mathcal{D}$ is ε-*close*. We emphasize that, although generating $\mathcal{D}$ will require sampling from the Gibbs distribution, using it will not. Our main result here will be the following:

▶ **Theorem 6.** *$P_{\texttt{ratio}}^{\texttt{all}}$ can be solved with the following complexities:*

- *In the continuous setting, with cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$.*
- *In the general integer setting, with cost $O\big(\frac{n^2 \log n}{\varepsilon^2} \log \frac{1}{\gamma} + n \log q\big)$.*
- *In the log-concave integer setting, with cost $O\big(\frac{n^2}{\varepsilon^2} \log \frac{1}{\gamma} + n \log q\big)$.*

A number of algorithms have been developed for *pointwise* estimation of $Q(\beta_{\max})$, with steadily improving sample complexities [4, 19, 12]. We denote this problem by $P_{\texttt{ratio}}^{\texttt{point}}$. The best prior algorithm for $P_{\texttt{ratio}}^{\texttt{point}}$ in the continuous setting [16] had cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$ (matching our algorithm for $P_{\texttt{ratio}}^{\texttt{all}}$). No specialized algorithms were known for the integer setting. We also show matching lower bounds:

▶ **Theorem 7.** *Let $n \geq n_0, q \geq q_0, \varepsilon < \varepsilon_0, \delta < \delta_0, \gamma < 1/4$ for certain absolute constants $n_0, q_0, \varepsilon_0, \delta_0$. There are problem instances μ which satisfy the given bounds n and q such that:*

(a) *$P_{\texttt{ratio}}^{\texttt{point}}$ requires cost $\Omega(\frac{q \log \frac{1}{\gamma}}{\varepsilon^2})$.*

(b) *$P_{\texttt{ratio}}^{\texttt{point}}$ requires cost $\Omega(\frac{\min\{q, n^2\} \log \frac{1}{\gamma}}{\varepsilon^2})$. and μ is log-concave*

Thus, Theorem 6 is optimal up to logarithmic factors; this essentially settles the complexity of $P_{\texttt{ratio}}$ as functions of n and q.

The first algorithm of Theorem 6, for the continuous setting, is similar to the pointwise algorithm in [16]; we defer it the to the full version of the paper.

1.3 Overview

We will develop two, quite distinct, types of algorithms: the first uses "cooling schedules" similar to [12, 16], and the second is based on a new type of "covering schedule" for the integer setting. In Section 6, we use these algorithms for approximate counting of matching and connected subgraphs. In Section 7, we show the lower bounds for the problems $P_{\texttt{ratio}}$ and $P_{\texttt{count}}$.

We remark that when $q \leq n^2$ in the integer setting, general continuous algorithms may be more efficient than the specialized integer algorithms for some tasks. These will be used for our algorithms to count independent sets and connected subgraphs, for instance.

Before the technical details, let us provide a high-level roadmap. For simplicity, we assume that tasks need to be solved with constant success probability.

The continuous setting

We can use a variant on an algorithm of [16] to solve $P^{\texttt{all}}_{\texttt{ratio}}$. Assuming this problem can be solved, let us examine the problem $P^{\delta,\varepsilon}_{\texttt{count}}$. As a starting point, consider the identity

$$\pi(x) = e^{-(\beta-\beta_{\min})x} \cdot \mu_\beta(x) \cdot Q(\beta) \qquad \text{for all } x \in \mathcal{F}, \beta \in [\beta_{\min}, \beta_{\max}]. \tag{1}$$

For any value β, we can estimate $Q(\beta)$ using our algorithm for $P^{\texttt{all}}_{\texttt{ratio}}$, and we can estimate $\mu_\beta(x)$ by drawing $\Theta(\frac{1}{\mu_\beta(x)\varepsilon^2})$ samples from μ_β. We then make use of the following important result: if $\mu_\beta([0,x])$ and $\mu_\beta([x,n])$ are both bounded below by constants, then $\mu_\beta(x) \geq \Omega(\Delta(x))$.

Therefore, we do the following: (i) use binary search to find value β with $\mu_\beta([0,x]) \approx \mu_\beta([x,n])$; and (ii) estimate $\mu_\beta(x)$ using $O(\frac{1}{\delta\varepsilon^2})$ samples; (iii) use Eq. (1) to determine $\pi(x)$. From standard concentration bounds, this satisfies the conditions of $P^{\delta,\varepsilon}_{\texttt{count}}$; for example, if $\Delta(x) \geq \delta$, then $\mu_\beta(x)$, and hence $\pi(x)$, is estimated within relative error ε.

To estimate all the counts, we find cut-points $y_1, \ldots, y_t$, where each interval $[y_i, y_{i+1}]$ has a corresponding value β_i with $\mu_{\beta_i}([y_i, n]) \geq \Omega(1)$ and $\mu_{\beta_i}([0, y_{i+1}]) \geq \Omega(1)$. Any $x \in [y_{i+1}, y_i]$ then has $\mu_{\beta_i}(x) \geq \Omega(\Delta(x))$, so we can use samples from μ_{β_i} to estimate c_x simultaneously for all $x \in [y_i, y_{i+1}]$. We show that only $t = O(\sqrt{q \log n})$ distinct intervals are needed, leading to a cost of $O(\frac{\sqrt{q \log n}}{\delta\varepsilon^2})$ plus the cost of solving $P^{\texttt{all}}_{\texttt{ratio}}$. The formal analysis appears in Section 3.

The integer setting

To solve $P^{\delta,\varepsilon}_{\texttt{count}}$, we develop a new data structure we call a *covering schedule*. This consists of a sequence $\beta_{\min} = \beta_0, \beta_1, \ldots, \beta_t = \beta_{\max}$ and corresponding values $k_1, \ldots, k_t$ so that $\mu_{\beta_i}(k_i)$ and $\mu_{\beta_i}(k_{i+1})$ are large for all i. (The definition is adjusted slightly for the endpoints $i = 0$ and $i = t$). Define $w_i = \min\{\mu_{\beta_i}(k_i), \mu_{\beta_i}(k_{i+1})\}$ ("weight" of i). If we take $\Omega(1/w_i)$ samples from μ_{β_i}, we can accurately estimate the quantities $\mu_{\beta_i}(k_i), \mu_{\beta_i}(k_{i+1})$, in turn allowing us to estimate

$$\frac{Q(\beta_i)}{Q(\beta_{i-1})} = e^{(\beta_i - \beta_{i-1})k_i} \frac{\mu_{\beta_{i-1}}(k_i)}{\mu_{\beta_i}(k_i)}$$

By telescoping products, this in turn allows us to estimate every value $Q(\beta_i)$.

Next, for each index $x \in \mathcal{H}$, we use binary search to find α with $\mu_\alpha([0,x]) \approx \mu_\alpha([x,n])$ and then estimate $\mu_\alpha(x)$ by taking $O(\frac{1}{\delta\varepsilon^2})$ samples. If α lies in interval $[\beta_i, \beta_{i+1}]$ of the covering schedule, we can use the estimates for $Q(\beta_i)$ and $Q(\beta_{i+1})$ to estimate $Q(\alpha)$ and hence $\pi(x)$. Since we do this for each $x \in \mathcal{H}$, the overall cost of this second phase is roughly $O(\frac{n}{\delta\varepsilon^2})$.

There is a more efficient algorithm for $P^{\delta,\varepsilon}_{\texttt{count}}$ for log-concave counts. In this case, for a fixed β and $x \in [\sigma^-, \sigma^+]$ we have $\mu_\beta(x) \geq \min\{\mu_\beta(\sigma^-), \mu_\beta(\sigma^+)\}$. Thus, a single value β_i in a covering schedule "covers" the interval $[k_i, k_{i+1}]$. We can solve $P^{\delta,\varepsilon}_{\texttt{count}}$ with $O(\frac{1}{\delta\varepsilon^2} + \sum_i \frac{1}{w_i\varepsilon^2})$ samples, by drawing $\Theta(\frac{1}{w_i\varepsilon^2})$ samples at β_i and $\Theta(\frac{1}{\delta\varepsilon^2})$ samples at $\beta_{\min}$ and $\beta_{\max}$.

After solving $P^{\delta,\varepsilon}_{\texttt{count}}$, we can then solve $P^{\texttt{all}}_{\texttt{ratio}}$ essentially for free, by estimating $\hat{Q}(\alpha \mid \mathcal{D}) = \sum_i e^{(\alpha-\beta_{\min})i}\hat{\pi}(i)$. So $P^{\texttt{all}}_{\texttt{ratio}}$ in the integer setting reduces to a special case of $P_{\texttt{count}}$. (Interestingly, the continuous-case algorithm works very differently – there, $P^{\texttt{all}}_{\texttt{ratio}}$ is a subroutine used to solve $P_{\texttt{count}}$.)

Obtaining a covering schedule

The general $P_{\texttt{count}}$ algorithm described above uses $O(\sum_i \frac{n}{w_i\varepsilon^2})$ samples to estimate the values $Q(\beta_i)$, and similarly the log-concave algorithm uses $O(\frac{1}{\delta\varepsilon^2} + \sum_i \frac{1}{w_i\varepsilon^2})$ samples. We thus refer to the quantity $\sum_i \frac{1}{w_i}$ as the *inverse weight* of the schedule. In the most technically involved part of the paper, we produce a covering schedule with inverse weight $O(n \log n)$ (or $O(n)$ in the log-concave setting). Here we just sketch some key ideas.

First, we construct a "preschedule" where each interval can choose two different indices σ_i^-, σ_i^+ instead of a single index k_i, with the indices interleaving as $\sigma_i^- \leq \sigma_{i+1}^- \leq \sigma_i^+ \leq \sigma_{i+1}^+$. The algorithm repeatedly fill gaps: if some half-integer $\ell + 1/2$ is not currently covered, then we can select a value β with $\mu_\beta([0,\ell]) \approx \mu_\beta([\ell+1, n])$. For this β, there is a value $\sigma^+ \in [\ell+1, n]$ with $\mu_\beta(\sigma^+) \cdot (\sigma^+ - \ell) \geq \Omega(\frac{1}{\log n})$, and similarly a value $\sigma^- \in [0, k]$ with $\mu_\beta(\sigma^-) \cdot (\ell - \sigma^- + 1) \geq \Omega(\frac{1}{\log n})$. The interval $[\sigma^-, \sigma^+]$ then fills the gap and also has weight $w \geq \Omega(\frac{1}{(\sigma^+ - \sigma^-)\log n})$.

At the end of the process, we throw away redundant intervals so each x is covered by at most two intervals, and "uncross" them into a schedule with $k_i \in \{\sigma^+_{i-1}, \sigma^-_i\}$. Since $\frac{1}{w_i} \leq O((\sigma_i^+ - \sigma_i^-)\log n)$ for each i, this gives an $O(n \log n)$ bound of the inverse weight of the schedule.

2 Preliminaries

Define $z(\beta) = \log Z(\beta)$ and $z(\beta_1, \beta_2) = \log \frac{Z(\beta_2)}{Z(\beta_1)} = \log \frac{Q(\beta_2)}{Q(\beta_1)}$; note that $z(\beta_{\min}, \beta_{\max}) \leq q$ by definition. We write $z'(\beta)$ for the derivative of function z.

Define the Chernoff separation functions $F_+(x,t) = \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^x$ and $F_-(x,t) = \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^x$, where $\delta = t/x$. These are well-known upper bounds on the probability that a binomial random variable with mean x is larger than $x+t$ or smaller than $x-t$, respectively. We also define $F(x,t) = F_+(x,t) + F_-(x,t)$.

For a random variable X, we write $\mathbb{V}(X)$ for the variance of X, and $\mathbb{S}[X] = \frac{\mathbb{E}[X^2]}{(\mathbb{E}[X])^2} - 1 = \frac{\mathbb{V}(X)}{(\mathbb{E}[X])^2}$ for the relative variance of X.

We write $\mu_\beta(x,y), \mu_\beta[x,y)$ instead of $\mu_\beta((x,y)), \mu_\beta([x,y))$, etc. for readability.

2.1 The `Balance` subroutine

Given a target χ, we sometimes need to find a value β with $\mu_\beta[0,\chi] \approx 1/2 \approx \mu_\beta[\chi, n]$. That is, χ is the "balancing point" in the distribution μ_β. Formally, for values $\beta_{\texttt{left}} \leq \beta_{\texttt{right}}$, let us denote by $\Lambda_\tau(\beta_{\texttt{left}}, \beta_{\texttt{right}}, \chi)$ the set of values $\beta \in [\beta_{\texttt{left}}, \beta_{\texttt{right}}]$ which satisfy the following two properties:

- Either $\beta = \beta_{\texttt{left}}$ or $\mu_\beta[0,\chi) \geq \tau$
- Either $\beta = \beta_{\texttt{right}}$ or $\mu_\beta[\chi, n] \geq \tau$

To find this, we use a subroutine `Balance`. To summarize briefly, since $\mu_\beta[0,\chi]$ is a monotonic function of β and can be estimated by sampling, the value β is found via a noisy binary search. Our main result is the following:

▶ **Theorem 8.** *Suppose that τ is an arbitrary constant and $\beta_{\min} \le \beta_{\texttt{left}} < \beta_{\texttt{right}} \le \beta_{\max}$. Then $\beta \leftarrow \texttt{Balance}(\beta_{\texttt{left}}, \beta_{\texttt{right}}, \chi, \gamma, \tau)$ has cost $O(\log \frac{nq}{\gamma})$. With probability at least $1-\gamma$, there holds $\beta \in \Lambda_\tau(\beta_{\texttt{left}}, \beta_{\texttt{right}}, \chi)$ (we say in this case that the call is* good*).*

The details appear in the full paper. The following observation explains the motivation for the definition.

▶ **Proposition 9.** *If $\beta \in \Lambda_\tau(\beta_{\min}, \beta_{\max}, x)$, then $\mu_\beta(x) \ge \tau\Delta(x)$.*

Proof. Consider $\alpha \in [\beta_{\min}, \beta_{\max}]$ with $\mu_\alpha(x) = \Delta(x)$. The result is clear if $\alpha = \beta$. Suppose that $\alpha < \beta$; the case $\alpha > \beta$ is completely analogous. So $\beta > \beta_{\min} = \beta_{\texttt{left}}$, and since $\beta \in \Lambda_\tau(\beta_{\min}, \beta_{\max}, x)$, this implies that $\mu_\beta[0,x] \ge \tau$. We then have:

$$\begin{aligned}\mu_\alpha(x) &= \frac{c_x e^{\alpha x}}{\sum_y c_y e^{\alpha y}} \le \frac{c_x}{\sum_{y\le x} c_y e^{\alpha(y-x)}} \le \frac{c_x}{\sum_{y\le x} c_y e^{\beta(y-x)}} \\ &= \frac{c_x e^{\beta x}}{\sum_{y\le x} c_y e^{\beta y}} = \frac{\mu_\beta(x)}{\mu_\beta[0,x]} \le \frac{\mu_\beta(x)}{\tau}.\end{aligned}$$

◀

2.2 Statistical sampling

We can obtain an unbiased estimator of the probability vector μ_β by computing empirical frequencies $\hat\mu_\beta$ from N independent samples from μ_β; we denote this process as $\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; N)$. We record the following standard concentration bound, which we will use repeatedly:

▶ **Lemma 10.** *For $\varepsilon, \gamma \in (0, \frac12), p_\circ \in (0,1]$, suppose we draw random variable $\hat p \sim \frac1N Binom(N,p)$ where $N \ge \frac{3e^\varepsilon \log(4/\gamma)}{(1-e^{-\varepsilon})^2 p_\circ}$. Then, with probability at least $1-\gamma$, the following two bounds both hold:*

$$|\hat p - p| \le \varepsilon(p + p_\circ), \qquad \text{and} \tag{2}$$

$$\hat p \in \begin{cases} [e^{-\varepsilon}p, e^{\varepsilon}p] & \text{if } p \ge e^{-\varepsilon}p_\circ \\ [0, p_\circ) & \text{if } p < e^{-\varepsilon}p_\circ \end{cases} \tag{3}$$

In particular, if Eq. (3) holds and $\min\{p,\hat p\} \ge p_\circ$, then $|\log\hat p - \log p| \le \varepsilon$.

Proof. See full paper. ◀

Many of our algorithms are based on calling $\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; N)$ and making decisions depending on the values $\hat\mu_\beta(I)$ for certain sets $I \subseteq \mathcal{F}$; they succeed when the estimates $\hat\mu_\beta(I)$ are close to $\mu_\beta(I)$. We say the execution of `Sample` *well-estimates* I if Eqs. (2),(3) hold for $p = \mu_\beta(I)$ and $\hat p = \hat\mu_\beta(I)$; otherwise it *mis-estimates* I. Likewise we say `Sample` well-estimates k if it well-estimates the singleton set $I = \{k\}$. Since this comes up so frequently, we write

$$\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; \varepsilon, \gamma, p_\circ)$$

as shorthand for $\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; \lceil \frac{3e^\varepsilon \log(4/\gamma)}{(1-e^{-\varepsilon})^2 p_\circ} \rceil)$. Note that this has cost $O(\frac{\log(1/\gamma)}{\varepsilon^2 p_\circ})$, and each set I is well-estimated with probability at least $1-\gamma$.

As we have touched upon, our algorithms for $P^{\delta,\varepsilon}_{\texttt{count}}$ estimate each value $\pi(x)$ by sampling $\hat\mu_\alpha(x)$ for a well-chosen value α. We use similar formulas to produce the estimates $\hat\pi(x), u(x)$ in all these cases. We record the following general result:

▶ **Lemma 11.** *Suppose that for $x \in \mathcal{F}$, we are given $\alpha \in [\beta_{\min}, \beta_{\max}]$ and non-negative parameters $\hat{Q}(\alpha), \hat{\mu}_\alpha(x), p_\circ$ (all of which may depend upon x), satisfying the following bounds:*
(A1) $|\log \hat{Q}(\alpha) - \log Q(\alpha)| \le 0.1\varepsilon$.
(A2) $p_\circ \le \mu_\alpha(x)(1 + \delta/\Delta(x))$
(A3) $|\hat{\mu}_\alpha(x) - \mu_\alpha(x)| \le 0.1\varepsilon(\mu_\alpha(x) + p_\circ)$.

Then the estimated values

$$\hat{\pi}(x) = \hat{Q}(\alpha)e^{(\beta_{\min}-\alpha)x}\hat{\mu}_\alpha(x), \qquad u(x) = 0.4\hat{Q}(\alpha)e^{(\beta_{\min}-\alpha)x}\varepsilon(\hat{\mu}_\alpha(x) + p_\circ)$$

satisfy the criteria for the problem $P^{\delta,\varepsilon}_{\texttt{count}}$.

Proof. See full paper. ◀

Since this formula comes up so often, we write

$\texttt{EstimatePi}(x, \alpha, p_\circ)$

as shorthand for setting $\hat{\pi}(x), u(x)$ according to the formula in Lemma 11. The values $\hat{Q}(\alpha)$ and $\hat{\mu}_\alpha(x)$ should be clear from the context.

In a number of places, we need to estimate certain telescoping products. Direct Monte Carlo sampling does not give strong tail bounds, so we use a standard method based on median amplification. See the full paper for a description and proof.

▶ **Theorem 12.** *Suppose we can sample non-negative random variables $X_1, \dots, X_N$. The subroutine $\texttt{EstimateProducts}(X, \tau, \varepsilon, \gamma)$ takes input $\varepsilon, \gamma \in (0,1)$ and $\tau > 0$, and returns a vector of estimates $(\hat{X}_1^{\texttt{prod}}, \dots, \hat{X}_N^{\texttt{prod}})$. It uses $O(N(1 + \tau/\varepsilon^2)\log\frac{1}{\gamma})$ total samples of the X variables. If $\tau \ge \sum_{i=1}^N \mathbb{S}[X_i]$, then with probability at least $1-\gamma$, it holds that $\frac{\hat{X}_i^{\texttt{prod}}}{\prod_{j=1}^i \mathbb{E}[X_j]} \in [e^{-\varepsilon}, e^{\varepsilon}]$ for all $i = 1, \dots, N$.*

In this case, it is also convenient to define $\hat{X}_0^{\texttt{prod}} = 1 = \prod_{j=1}^0 \mathbb{E}[X_j]$.

3 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$ in the continuous setting

In this section, we develop Algorithm 1 for $P^{\delta,\varepsilon}_{\texttt{count}}$. Here, we use a general algorithm to solve $P^{\texttt{all}}_{\texttt{ratio}}$ in the continuous setting with cost $O\big(\frac{q\log n}{\varepsilon^2}\log\frac{1}{\gamma}\big)$.

Algorithm 1 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$ for error parameter γ.

call $\mathcal{D} \leftarrow \texttt{PratioAll}(\varepsilon/10, \gamma/4)$.
initialize $x_0 \leftarrow n, \alpha_0 \leftarrow \beta_{\max}$
for $t = 1$ **to** $T = 10\min\{q, \sqrt{q\log n}\}$ **do**
  set $\alpha_t \leftarrow \texttt{Balance}(\beta_{\min}, \alpha_{t-1}, x_{t-1}, \frac{\gamma}{100T}, 1/4)$
  set $\hat{\mu}_{\alpha_t} \leftarrow \texttt{Sample}(\alpha_t; \frac{10^8 \log\frac{50T}{\delta\gamma}}{\delta\varepsilon^2})$
  if $\alpha_t > \beta_{\min}$ **then**
    set x_t to be the minimum value with $\hat{\mu}_{\alpha_t}[0, x_t] \ge 1/100$
    foreach $y \in (x_t, x_{t-1}]$ **do** $\texttt{EstimatePi}(y, \alpha_t, \delta/200)$ with $\hat{Q}(\alpha_t) = \hat{Q}(\alpha_t \mid \mathcal{D})$
  else if $\alpha_t = \beta_{\min}$ **then**
    foreach $y \in [0, x_{t-1}]$ **do** $\texttt{EstimatePi}(y, \alpha_t, \delta/200)$ with $\hat{Q}(\alpha_t) = \hat{Q}(\alpha_t \mid \mathcal{D})$
    return

▶ **Theorem 13.** *Algorithm 1 solves $P^{\delta,\varepsilon}_{\texttt{count}}$ with cost*

$$O\Big(\frac{\min\{q, \sqrt{q \log n}\} \log \frac{q}{\delta\gamma}}{\delta\varepsilon^2} + \frac{q \log n \log \frac{1}{\gamma}}{\varepsilon^2}\Big).$$

The complexity bound follows immediately from specification of suborutines. We next analyze the success probability; this will require a number of intermediate calculations.

▶ **Proposition 14.** *With probability at least $1 - \gamma/10$, the following conditions hold for all iterations t:*

(i) *$\alpha_t \in \Lambda_{1/4}(\beta_{\min}, \alpha_{t-1}, x_{t-1})$ and $x_t < x_{t-1}$ and and $\mu_{\alpha_t}[0, x_t) \le \frac{1}{70}$*
(ii) *$\alpha_t = \beta_{\min}$ or $\mu_{\alpha_t}[0, x_t] \ge \frac{1}{200}$.*

Proof. See full paper. ◀

For the remainder of the analysis, we suppose that the bounds of Proposition 14 hold.

▶ **Proposition 15.** *For all iterations t, we have $\alpha_{t+1} < \alpha_t$ strictly and $\mu_{\alpha_{t+1}}[x_t, n] \ge 1/4$. Furthermore, if $\alpha_{t+1} \ne \beta_{\min}$, then $z(\alpha_{t+1}, \alpha_t) \ge 2 + \frac{x_{t+1}}{x_{t-1} - x_{t+1}}$.*

Proof. See full paper. ◀

▶ **Lemma 16.** *The loop at line 3 terminates before iteration T.*

Proof. Suppose not; by Proposition 14, we have $x_1 > x_2 > \cdots > x_{T-1} > \beta_{\min}$ strictly. Since each x_i comes from $\mathcal{F}$, we must have $x_1 \le n$ and $x_{T-2} \ge 1$. Let $g = T - 4$; note that due to bounds on n, q. we have $g \ge T/2 > 0$. For each $\ell = 1, \dots, g$, consider the non-negative value $a_\ell = \log(\frac{x_{\ell-1}}{x_{\ell+1}})$. We note the following bound:

$$\begin{aligned}\sum_{\ell=1}^{g} a_\ell &= \log\frac{x_1}{x_3} + \log\frac{x_2}{x_4} + \log\frac{x_3}{x_5} + \cdots + \log\frac{x_{g-2}}{x_g} + \log\frac{x_{g-1}}{x_{g+1}} \\ &= \log x_1 + \log x_2 - \log x_g - \log x_{g+1} \qquad \text{by telescoping sums} \\ &\le \log n + \log n - 0 - 0 = 2\log n\end{aligned}$$

By using Proposition 15 for each iteration $1, \dots, g$, we can compute:

$$q \ge z(\beta_{\max}, \beta_{\min}) \ge \sum_{i=1}^{g} z(\alpha_{i+1}, \alpha_i) \ge \sum_{i=1}^{g} 2 + \frac{x_{i+1}}{x_{i-1} - x_{i+1}} = 2g + \sum_{\ell=1}^{g} \frac{1}{e^{a_\ell} - 1}. \tag{4}$$

By Jensen's inequality applied to the concave function $y \mapsto \frac{1}{e^y - 1}$, we have

$$\sum_{\ell=1}^{g} \frac{1}{e^{a_\ell} - 1} \ge \frac{g}{\exp(\frac{1}{g}\sum_{\ell=1}^{g} a_\ell) - 1} \ge \frac{g}{\exp(\frac{2\log n}{g}) - 1}. \tag{5}$$

If $q > 2\log n$, then Eq. (4) shows $g \le q/2$. If $q \ge 2\log n$, then $\exp(\frac{2\log n}{g}) - 1 \le \frac{4e\log n}{g}$, and then Eq. (5) implies $q \ge \frac{g}{(4e\log n)/g} \ge g^2/20$, i.e. $g \le \sqrt{20q\log n}$. Either way, we have $g \ge \min\{\sqrt{20q\log n}, q/2\}$. Since $g \ge T/2$, this is a contradiction to the definition of T. ◀

▶ **Proposition 17.** *With probability at least $1 - \gamma/10$, the preconditions of Lemma 11 for* `EstimatePi` *(with $p_\circ = \delta/200$) hold for all $y \in \mathcal{F}$.*

Proof. See full paper. ◀

Overall, the total failure probability is at most $\gamma/10$ (from Proposition 14) plus $\gamma/10$ (from Proposition 17). This concludes the proof of Theorem 13. It also shows the first part of Theorem 4.

4 Solving $P_{\texttt{count}}$ and $P^{\texttt{all}}_{\texttt{ratio}}$ for integer-valued Gibbs distributions

The algorithms in the integer setting hinge on a data structure called the *covering schedule*. Formally, we define a covering schedule to be a sequence of the form

$$(\beta_0, w_0, k_1, \beta_1, w_1, k_2, \dots, \beta_{t-1}, w_{t-1}, k_t, \beta_t, w_t)$$

which satisfies the following additional constraints:

(i) $\beta_{\min} = \beta_0 < \dots < \beta_t = \beta_{\max}$;
(ii) $k_1 < k_2 < \dots < k_t$
(iii) $w_i \in [0,1]$ for $i = 0, \dots, t$.

Note that $t \le n+1$. We say that $\mathcal{I}$ is *proper* if for all $i = 1, \dots, t$ it satisfies

$$\mu_{\beta_{i-1}}(k_i) \ge w_{i-1} \text{ and } \mu_{\beta_i}(k_i) \ge w_i.$$

We define

$$\texttt{InvWeight}(\mathcal{I}) = \sum_{i=0}^{t} \frac{1}{w_i}.$$

Our algorithm to solve $P_{\texttt{count}}$ will have four stages. As a high-level summary, it proceeds as follows:

1. Construct a suitable covering schedule $\mathcal{I} = (\beta_0, w_0, k_1, \dots, k_t, \beta_t, w_t)$.
2. Estimate the values $Q(\beta_i)$ for $i = 0, \dots, t$.
3. Use these estimates $\hat{Q}(\beta_i)$ to estimate the counts c_i
4. Use the estimated counts $\hat{c}_i$ to estimate the entire function $Q(\beta)$

The first stage is quite involved, so we defer it to Section 5 where we show the following result:

▶ **Theorem 18.** *There is a procedure* $\texttt{FindCoveringSchedule}(\gamma)$ *which produces a covering schedule* $\mathcal{I}$*, which is proper with probability at least* $1-\gamma$*. In the general integer setting, the procedure has cost* $O(n \log^3 n + n \log n \log \frac{1}{\gamma} + n \log q)$ *and has* $\texttt{InvWeight}(\mathcal{I}) \le O(n \log n)$*. In the log-concave setting, the procedure has cost* $O(n \log^2 n + n \log \frac{1}{\gamma} + n \log q)$ *and has* $\texttt{InvWeight}(\mathcal{I}) \le O(n)$*.*

The second stage is summarized in the following result:

▶ **Theorem 19.** *There is an algorithm* $\texttt{PratioCoveringSchedule}(\mathcal{I}, \varepsilon, \gamma)$ *which takes as input a covering schedule* $\mathcal{I} = (\beta_0, w_0, k_1, \dots, k_t, \beta_t, w_t)$ *and produces estimates* $\hat{Q}(\beta_0), \dots, \hat{Q}(\beta_t)$*. The overall algorithm cost is* $O\Big(\frac{\min\{nW, q \log n\} \log \frac{1}{\gamma}}{\varepsilon^2}\Big)$ *where* $W = \texttt{InvWeight}(\mathcal{I})$*. If* $\mathcal{I}$ *is proper, then with probability at least* $1-\gamma$ *it satisfies* $|\log \hat{Q}(\beta_i) - Q(\beta_i)| \le \varepsilon$ *for all* i*. (When this latter condition holds, we say that the call to* $\texttt{PratioCoveringSchedule}$ *is* good*).*

Proof. To get the cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$, we simply run the algorithm $\mathcal{D} \leftarrow \texttt{PratioAll}(\varepsilon, \gamma)$ for the continuous setting as in Theorem 6, and output $\hat{Q}(\beta_i \mid \mathcal{D})$ for all i. To get the other cost bound (in terms of W), we use the following algorithm:

Algorithm 2 Estimating values $Q(\beta_i)$ via `EstimateProducts`.

1 **for** $\boldsymbol{i = 1, \ldots t}$ form random variables $X_i \sim \text{Bernoulli}(\mu_{\beta_{i-1}}(k_i))$ and $Y_i \sim \text{Bernoulli}(\mu_{\beta_i}(k_i))$
2 set $\hat{X}^{\texttt{prod}} \leftarrow \texttt{EstimateProducts}(X, W, \varepsilon/2, \gamma/4)$
3 set $\hat{Y}^{\texttt{prod}} \leftarrow \texttt{EstimateProducts}(Y, W, \varepsilon/2, \gamma/4)$
4 for $i = 0, \ldots, t$ set $\hat{Q}(\beta_i) = \exp\left(\sum_{j=1}^{i}(\beta_j - \beta_{j-1})k_j\right) \cdot \hat{X}_i^{\texttt{prod}} / \hat{Y}_i^{\texttt{prod}}$

Here, assuming that $\mathcal{I}$ is proper, we have $\mathbb{S}[X_i] = \frac{1}{\mu_{\beta_{i-1}}(k_i)-1} \leq \frac{1}{w_{i-1}}$ for each i, so $\sum_i \mathbb{S}[X_i] \leq W$. Likewise $\sum_i \mathbb{S}[Y_i] \leq W$. So with probability at least $1 - \gamma/2$ the estimates $\hat{X}_i^{\texttt{prod}}, \hat{Y}_i^{\texttt{prod}}$ are all within $e^{\pm\varepsilon/2}$ of $\prod_{j=1}^{i}\mathbb{E}[X_j], \prod_{j=1}^{i}\mathbb{E}[Y_j]$ respectively. Observe that

$$\frac{\mathbb{E}[\prod_{j=1}^{i} X_j]}{\mathbb{E}[\prod_{j=1}^{i} Y_j]} = \prod_{j=1}^{i-1} \frac{\mu_{\beta_{j-1}}(k_j)}{\mu_{\beta_j}(k_j)} = \prod_j e^{(\beta_{j-1}-\beta_j)k_j} \frac{Z(\beta_j)}{Z(\beta_{j-1})} = \frac{Z(\beta_i)}{Z(\beta_0)} \cdot \exp\Big(\sum_{j=1}^{i}(\beta_{j-1} - \beta_j)k_j\Big)$$

so in that case, the values $\hat{Q}(\beta_i)$ are also within $e^{\pm\varepsilon}$ of $Z(\beta_i)/Z(\beta_0) = Q(\beta_i)$ as required. ◀

4.1 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$

We now move on to the third stage, of using the covering schedule to solve $P_{\texttt{count}}$. There are two quite distinct algorithms here: one for generic integer-valued distributions, and a specialized algorithm for log-concave distributions. We begin with the following algorithm for general integer distributions:

Algorithm 3 Solving problem $P^{\delta,\varepsilon}_{\texttt{count}}$.

1 set $\mathcal{I} = (\beta_0, w_0, k_1, \ldots, k_t, \beta_t, w_t) \leftarrow \texttt{FindCoveringSchedule}(\gamma/10)$
2 set $(\hat{Q}(\beta_0), \ldots, \hat{Q}(\beta_t)) \leftarrow \texttt{PratioCoveringSchedule}(\mathcal{I}, \varepsilon/100, \gamma/10)$
3 **for** $i = 0, \ldots, t$ **do** let $\hat{\mu}_{\beta_i} \leftarrow \texttt{Sample}(\beta_i; \varepsilon/100, \frac{\gamma}{10(n+1)^2}, w_i)$
4 **for** $j \in \mathcal{H}$ **do**
5 set $\alpha \leftarrow \texttt{Balance}(\beta_{\min}, \beta_{\max}, j, \frac{\gamma}{10(n+1)^2}, 1/4)$
6 find index $i < t$ with $\alpha \in [\beta_i, \beta_{i+1}]$
7 let $\hat{\mu}_\alpha \leftarrow \texttt{Sample}(\alpha; \varepsilon/100, \frac{\gamma}{10(n+1)^2}, \delta/4)$
8 **if** $\boldsymbol{\hat{\mu}_\alpha(k_{i+1}) \geq \delta}$ **then** $\texttt{EstimatePi}(j, \alpha, \delta/4)$ where $\hat{Q}(\alpha) = \frac{\hat{\mu}_{\beta_i}(k_{i+1})}{\hat{\mu}_\alpha(k_{i+1})} e^{(\alpha-\beta_i)k_{i+1}} \hat{Q}(\beta_i)$
9 **else if** $\boldsymbol{j \geq k_{i+1}}$ **then** $\texttt{EstimatePi}(j, \beta_{i+1}, w_{i+1}/8)$ where $\hat{Q}(\beta_{i+1})$ is set at line 2.
10 **else if** $\boldsymbol{j < k_{i+1}}$ **then** $\texttt{EstimatePi}(j, \beta_i, w_i/8)$ where $\hat{Q}(\beta_i)$ is set at line 2.

▶ **Theorem 20.** *Algorithm 3 solves $P^{\delta,\varepsilon}_{\texttt{count}}$ with cost $O\Big(\frac{(n/\delta)\log\frac{n}{\gamma} + n^2 \log n \log\frac{1}{\gamma}}{\varepsilon^2} + n \log q\Big)$.*

Proof. See full paper. ◀

This gives the second part of Theorem 4. As we have mentioned, there is an alternative algorithm to estimate counts in the log-concave setting:

Algorithm 4 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$ in the log-concave setting.

set $\mathcal{I} = (\beta_0, w_0, k_1, \dots, k_t, \beta_t, w_t) \leftarrow \texttt{FindCoveringSchedule}(\gamma/10)$
set $(\hat{Q}(\beta_0), \dots, \hat{Q}(\beta_t)) \leftarrow \texttt{PratioCoveringSchedule}(\mathcal{I}, 0.1\varepsilon, \gamma/6)$
update $\delta \leftarrow \min\{\delta, 1/n, 1/\texttt{InvWeight}(\mathcal{I})\}$.
for $i = 1, \dots t-1$ **do**
  let $\hat{\mu}_{\beta_i} \leftarrow \texttt{Sample}(\beta_i; 0.01\varepsilon, \frac{\gamma}{6(n+1)}, w_i)$
  foreach $j \in \{k_i + 1, k_i + 2, \dots, k_{i+1}\}$ **do** $\texttt{EstimatePi}(j, \beta_i, \delta/4)$
let $\hat{\mu}_{\beta_{\min}} \leftarrow \texttt{Sample}(\beta_{\min}; 0.01\varepsilon, \frac{\gamma}{6(n+1)}, w'_0)$ for $w'_0 = \min\{w_0, \delta/2\}$
foreach $j \in \{0, 1, \dots, k_1\}$ **do** $\texttt{EstimatePi}(j, \beta_{\min}, w'_0)$.
let $\hat{\mu}_{\beta_{\max}} \leftarrow \texttt{Sample}(\beta_{\max}; 0.01\varepsilon, \frac{\gamma}{6(n+1)}, w'_t)$ for $w'_t = \min\{w_t, \delta/2\}$)
foreach $j \in \{k_t + 1, k_t + 2, \dots, n\}$ **do** $\texttt{EstimatePi}(j, \beta_{\max}, w'_t)$.

▶ **Theorem 21.** *In the log-concave setting, Algorithm 4 solves $P^{\delta,\varepsilon}_{\texttt{count}}$ with cost*

$$O\Big(n \log^2 n + n \log q + \frac{\min\{n^2, q \log n\} \log \frac{1}{\gamma} + (n + 1/\delta) \log \frac{n}{\gamma}}{\varepsilon^2}\Big)$$

Proof. See full paper. ◀

Again, with some simplification of parameters, this gives the third part of Theorem 4.

4.2 Solving $P^{\texttt{all}}_{\texttt{ratio}}$

Finally, having estimated the counts, we can proceed to use these estimates to fill in the entire function $Q(\beta)$. This is a black-box reduction from $P_{\texttt{count}}$ to $P^{\texttt{all}}_{\texttt{ratio}}$.

▶ **Theorem 22.** *Given a solution $(\hat{\pi}, u)$ for $P^{1/n, 0.1\varepsilon}_{\texttt{count}}$ in the integer setting, we can solve $P^{\texttt{all}}_{\texttt{ratio}}$ with probability one and no additional queries to the oracle.*

Proof. The data structure $\mathcal{D}$ is the vector $\hat{\pi}$, and for a query value α we set $\hat{Q}(\alpha \mid \mathcal{D}) = \sum_{i \in \mathcal{H}} \hat{\pi}(i) e^{(\alpha - \beta_{\min})i}$. See full paper for proof details. ◀

Our $P^{\delta,\varepsilon}_{\texttt{count}}$ algorithms thus solve $P^{\texttt{all}}_{\texttt{ratio}}$ with cost $O\big(\frac{n^2 \log n \log \frac{1}{\gamma}}{\varepsilon^2} + n \log q\big)$ in the general integer setting, and $O\big(\frac{n^2 \log \frac{1}{\gamma}}{\varepsilon^2} + n \log q\big)$ in the log-concave setting. This shows the two bounds of Theorem 6.

5 Constructing a covering schedule

In the full paper, we show that any non-negative log-concave sequence $a_1, \dots, a_m$ satisfying $a_k \le \frac{1}{k}$ for each $k \in [m]$ satisfies $a_1 + \dots + a_m \le e$. Without the log-concavity assumption we would have $a_1 + \dots + a_m \le \sum_{k=1}^m \frac{1}{k} \le 1 + \log m$ (by a well-known inequality for the harmonic series). Motivated by these facts, we define the following parameter in this section:

$$\rho \stackrel{\text{def}}{=} \begin{cases} 1 + \log(n+1) & \text{in the general integer setting} \\ e & \text{in the log-concave setting} \end{cases}$$

We will show the following more precise bound on the weight of the schedule.

▶ **Theorem 23.** *In the integer setting, the procedure* `FindCoveringSchedule`(γ) *produces a covering schedule* $\mathcal{I}$ *with* `InvWeight`$(\mathcal{I}) \le a(n+1)\rho$ *and* $\mathbb{P}[\mathcal{I}$ *is proper*$] \ge 1-\gamma$*, where* $a > 4$ *is an arbitrary constant. It has cost* $O(n\rho(\log^2 n + \log\frac{1}{\gamma}) + n\log q)$.

This immediately implies Theorem 18. In order to build the covering schedule, we first build an object with relaxed constraints called a *preschedule*, discussed in Sections 5.1. In Section 5.2, we convert this into a schedule.

5.1 Constructing a preschedule

Let us fix constants $\tau \in (0, \frac{1}{2})$, $\lambda \in (0,1)$, and set $\phi = \tau\lambda^3/\rho$. Let us introduce basic terminology and definitions.

An $\mathcal{H}$-interval is a discrete set of points $\{\sigma^-, \sigma^-+1, \dots, \sigma^+-1, \sigma^+\}$, for integers $0 \le \sigma^- \le \sigma^+ \le n$. We also write this more compactly as $\sigma = [\sigma^-, \sigma^+]$. We define $\mathtt{span}(\sigma) = \sigma^+ - \sigma^- + 1$, i.e. the cardinality of σ when viewed as a subset of $\mathcal{H}$.

A *segment* is a tuple $\theta = (\beta, \sigma)$ where $\beta \in [\beta_{\min}, \beta_{\max}]$, and σ is an $\mathcal{H}$-segment. We say θ is ϕ-*proper* (or just proper if ϕ is understood) if it satisfies the following two properties:

- Either $\beta = \beta_{\min}$ or $\mu_\beta(\sigma^-) \ge \phi/\mathtt{span}(\sigma)$
- Either $\beta = \beta_{\max}$ or $\mu_\beta(\sigma^+) \ge \phi/\mathtt{span}(\sigma)$

A *preschedule* is a sequence of distinct segments $\mathcal{J} = ((\beta_0, \sigma_0), \dots, (\beta_t, \sigma_t))$ satisfying the following properties:

(`I0`) $\sigma^-_{i+1} \le \sigma^+_i$ for $i = 0, \dots, t-1$.

(`I1`) $\beta_{\min} = \beta_0 \le \dots \le \beta_t = \beta_{\max}$.

(`I2`) $0 = \sigma^-_0 \le \dots \le \sigma^-_t \le n$ and $0 \le \sigma^+_0 \le \dots \le \sigma^+_t = n$

We say that $\mathcal{I}$ is ϕ-*proper* if all segments θ_i are ϕ-proper.

The main idea of the algorithm is to maintain a sequence of proper segments satisfying properties (`I1`) and (`I2`), and grow it until it satisfies (`I0`). This uses an additional subroutine $\sigma \leftarrow \mathtt{FindInterval}(\beta, \sigma_{\mathtt{left}}, \sigma_{\mathtt{right}})$, where $\beta \in [\beta_{\min}, \beta_{\max}]$, and $\sigma_{\mathtt{left}}, \sigma_{\mathtt{right}}$ are two discrete intervals in $\mathcal{H}$ and the returned interval $\sigma = [\sigma^-, \sigma^+]$ has $\sigma^- \in \sigma_{\mathtt{left}}, \sigma^+ \in \sigma_{\mathtt{right}}$. Deferring for the moment the definition of `FindInterval`, the details are provided below.

Algorithm 5 Computing an initial preschedule.

call $\sigma_{\min} \leftarrow \mathtt{FindInterval}(\beta_{\min}, \{0\}, \mathcal{H})$ and $\sigma_{\max} \leftarrow \mathtt{FindInterval}(\beta_{\max}, \mathcal{H}, \{n\})$
initialize $\mathcal{J}$ to contain the two segments $(\beta_{\min}, \sigma_{\min}), (\beta_{\max}, \sigma_{\max})$
while $\mathcal{J}$ *does not satisfy* (`I0`) **do**
  pick arbitrary consecutive segments $\theta_{\mathtt{left}} = (\beta_{\mathtt{left}}, \sigma_{\mathtt{left}})$ and $\theta_{\mathtt{right}} = (\beta_{\mathtt{right}}, \sigma_{\mathtt{right}})$ in $\mathcal{J}$ with $\sigma^+_{\mathtt{left}} < \sigma^-_{\mathtt{right}}$.
  let $M = \left\lfloor \frac{\sigma^+_{\mathtt{left}} + \sigma^-_{\mathtt{right}}}{2} \right\rfloor + \frac{1}{2}$
  call $\beta \leftarrow \mathtt{Balance}(\beta_{\mathtt{left}}, \beta_{\mathtt{right}}, M, \frac{1}{4n}, \tau)$
  call

$$\sigma \leftarrow \begin{cases} \mathtt{FindInterval}(\beta, [\sigma^-_{\mathtt{left}}, M-\frac{1}{2}], [M+\frac{1}{2}, \sigma^+_{\mathtt{right}}]) & \text{if } \beta_{\mathtt{left}} < \beta < \beta_{\mathtt{right}} \\ \mathtt{FindInterval}(\beta, \{\sigma^-_{\mathtt{left}}\}, [M+\frac{1}{2}, \sigma^+_{\mathtt{right}}]) & \text{if } \beta = \beta_{\mathtt{left}} \\ \mathtt{FindInterval}(\beta, [\sigma^-_{\mathtt{left}}, M-\frac{1}{2}], \{\sigma^+_{\mathtt{right}}\}) & \text{if } \beta = \beta_{\mathtt{right}} \end{cases}$$

  insert (β, σ) into $\mathcal{J}$ between $\theta_{\mathtt{left}}$ and $\theta_{\mathtt{right}}$
return $\mathcal{J}$

Now let us say that a segment (β, σ, w) is *extremal* if it satisfies the following conditions:

$$\mu_\beta(k) \leq \frac{1}{\lambda} \cdot \frac{\texttt{span}(\sigma)}{\texttt{span}(\sigma) + (\sigma^- - k)} \cdot \mu_\beta(\sigma^-) \qquad \forall k \in \{0, \dots, \sigma^- - 1\} \tag{6a}$$

$$\mu_\beta(k) \leq \frac{1}{\lambda} \cdot \frac{\texttt{span}(\sigma)}{\texttt{span}(\sigma) + (k - \sigma^+)} \cdot \mu_\beta(\sigma^+) \qquad \forall k \in \{\sigma^+ + 1, \dots, n\} \tag{6b}$$

There are two additional invariants we hope to maintain in Algorithm 5:
(I3) Each segment θ of $\mathcal{J}$ is ϕ-proper.
(I4) Each segment θ of $\mathcal{J}$ is extremal.

We say the call $\sigma \leftarrow \texttt{FindInterval}(\beta, \sigma_{\texttt{left}}, \sigma_{\texttt{right}})$ is *good* if the segment $\theta = (\beta, \sigma)$ satisfies (I3) and (I4), and we say the call at line 7 is *valid* if $\beta \in \Lambda_\tau(\beta_{\texttt{left}}, \beta_{\texttt{right}}, M)$ and both $\theta_{\texttt{left}}$ and $\theta_{\texttt{right}}$ satisfy (I3), (I4). The calls at line 1 are always valid. The following result summarizes `FindInterval`.

▶ **Theorem 24.** $\texttt{FindInterval}(\beta, \sigma_{\texttt{left}}, \sigma_{\texttt{right}})$ *has cost* $O\big(\rho(\sigma^+_{\texttt{right}} - \sigma^-_{\texttt{left}} + 1) \log n\big)$. *If the call is valid, then the call is good with probability at least* $1 - \frac{1}{4(n+2)}$.

We defer the proof, which is quite technical, the full paper. Putting it aside for the moment, we have the following results:

▶ **Proposition 25.** *Algorithm 5 outputs a preschedule, and it is ϕ-proper with probability at least* $1/2$.

Proof. If all calls to `Balance` and `FindInterval` are good, then $\mathcal{J}$ maintains properties (I3) and (I4), and in particular it is ϕ-proper. The loop in lines 3 – 8 is executed at most n times, since each time it covers a new half-integer value M. So the algorithm calls `FindInterval` at most $n+2$ times and `Balance` at most n times. Since `Balance` or `FindInterval` fail with probability at most $\frac{1}{4n}$ and $\frac{1}{4(n+2)}$ respectively, properties (I3) and (I4) are maintained with probability at least $1/2$. ◀

▶ **Proposition 26.** *Algorithm 5 has cost* $O(n \log q + n\rho \log^2 n)$.

Proof. See full paper. ◀

5.2 Converting the preschedule into a covering schedule

There are two steps to convert the preschedule into a covering schedule. First, we throw away redundant intervals. Second, we "uncross" the adjacent intervals. While we are doing this, we also check if the resulting schedule is proper; if not, we will discard it and generate a new preschedule from scratch.

▶ **Proposition 27.** *Given a preschedule $\mathcal{J}$, there is a procedure* $\texttt{MinimizePreschedule}(\mathcal{J})$, *which has zero sample complexity, to generate a preschedule* $\mathcal{J}' = ((\beta_0, \sigma_0), \dots, (\beta_t, \sigma_t))$ *satisfying the following three properties:*
(J1) $\sigma_i^+ < \sigma_{i+2}^-$ *for* $i = 0, \dots, t-2$.
(J2) $\beta_0 < \beta_1 < \cdots < \beta_t$ *strictly.*
(J3) *For any $k \in \mathcal{H}$, there are at most two segments $\theta_i = (\beta_i, \sigma_i) \in \mathcal{J}'$ with $k \in \sigma_i$.*
Furthermore, if $\mathcal{J}$ is ϕ-proper, then so is $\mathcal{J}'$ with probability one.

Proof. Start with $\mathcal{J}$ and repeatedly apply two operations: (i) discard a segment $i \in \{1, \dots, t-1\}$ if $\sigma_{i+1}^- \leq \sigma_{i-1}^+$ or (ii) merge adjacent segments with $\beta_i = \beta_{i+1}$, namely, replace the two segments $(\beta_i, \sigma_i), (\beta_{i+1}, \sigma_{i+1})$ with a single segment $(\beta_i, [\sigma_i^-, \sigma_{i+1}^+])$. The operations are performed in any order until no further changes are possible; let $\mathcal{J}'$ be the result of this process. ◀

We next describe the procedure to uncross a preschedule. Here $\nu > 0$ is some arbitrary constant.

Algorithm 6 `UncrossSchedule`$(\mathcal{J}, \gamma)$ for preschedule $\mathcal{J} = ((\beta_0, \sigma_0), \dots, (\beta_t, \sigma_t))$.

for $i = 0, \dots, t$ **do** let $\hat{\mu}_{\beta_i} \leftarrow \texttt{Sample}(\beta_i; \frac{\nu}{2}, \frac{\gamma}{4(t+1)}, e^{-\nu/2} w_i)$ where $w_i = \phi / \texttt{span}(\sigma_i)$
for $i = 1, \dots, t$ **do**
  if $\exists\, k \in \{\sigma_{i-1}^+, \sigma_i^-\}$ *s.t.* $\hat{\mu}_{\beta_{i-1}}(k) \geq e^{-\nu/2} w_{i-1}$ *and* $\hat{\mu}_{\beta_i}(k) \geq e^{-\nu/2} w_i$ **then**
    set $k_i = k$ for arbitrary such k
  else return $\perp$
return *covering schedule* $\mathcal{I} = (\beta_0, e^{-\nu} w_0, k_1, \beta_1, e^{-\nu} w_1, k_2, \dots, k_t, \beta_t, e^{-\nu} w_t)$

▶ **Theorem 28.** *Suppose that preschedule $\mathcal{J}$ satisfies properties* `(J1)`, `(J2)`, `(J3)`. *Then:*
(a) *The output is either $\perp$ or a covering schedule $\mathcal{I}$ with* $\texttt{InvWeight}(\mathcal{I}) \leq \frac{2e^{\nu}(n+1)}{\phi}$.
(b) *It outputs an improper covering schedule with probability at most γ, irrespective of $\mathcal{J}$.*
(c) *If $\mathcal{J}$ is ϕ-proper, then it outputs a proper covering schedule with probability at least $1 - \gamma$.*
(d) *The cost is $O(n\rho \log \frac{n}{\gamma})$.*

Proof. See full paper. ◀

We can finish by combining all the preschedule processing algorithms, as follows:

Algorithm 7 Algorithm `FindCoveringSchedule`(γ).

while `true` **do**
  call Algorithm 5 with appropriate constants ν, λ, τ to compute preschedule $\mathcal{J}$
  call $\mathcal{J}' \leftarrow \texttt{MinimizePreschedule}(\mathcal{J})$
  call $\mathcal{I} \leftarrow \texttt{UncrossSchedule}(\mathcal{J}', \gamma/4)$
  if $\mathcal{I} \neq \perp$ **then return** $\mathcal{I}$

By Proposition 25 and Theorem 28, each iteration of Algorithm 7 terminates with probability at least $\frac{1}{2}(1 - \gamma/4) \geq 3/8$, so there are $O(1)$ expected iterations. Each call to `UncrossSchedule` has cost $O(n\rho \log \frac{n}{\gamma})$. By Proposition 26, each call to Algorithm 5 has cost $O(n \log q + n\rho \log n)$.

By Theorem 28(a), $\texttt{InvWeight}(\mathcal{I}) \leq 2\rho(n+1) \cdot \frac{e^{\nu}}{\tau\lambda^3}$. The term $\frac{e^{\nu}}{\tau\lambda^3}$ gets arbitrarily close to 2 for constants ν, λ, τ sufficiently close to $0, 1, \frac{1}{2}$ respectively.

Finally, by Proposition 28, each iteration of Algorithm 7 returns a non-proper covering schedule with probability at most $\gamma/4$ (irrespective of the choice of $\mathcal{J}$). Thus, the total probability of returning a non-proper covering schedule over all iterations is at most $\sum_{i=0}^{\infty} (3/8)^i \gamma/4 = 2\gamma/5 \leq \gamma$ as desired.

This shows Theorem 23.

6 Combinatorial applications

Consider a combinatorial setting with c_i objects of weights $i = 0, \dots, n$, and we can sample from a Gibbs distribution at rate β (for certain values of β). If we know at least one of the counts, then estimates for $\pi(x)$ directly translate into estimate of c_i. Our usual strategy here will be to solve $P^{\delta,\varepsilon}_{\texttt{count}}$ for $\delta = O(\min_x \Delta(x))$, for chosen boundary parameters $\beta_{\min}, \beta_{\max}$; in this case, it can easily be seen that the resulting estimated counts $\hat{c}_i = c_0 \hat{\pi}(i)/\hat{\pi}(0)$ are accurate within $e^{\pm\varepsilon}$ relative error.

In many of these combinatorial applications, the counts are known to be log-concave; in this case, there are natural choices for algorithm parameters which lead to particularly clean bounds. When counts are not log-concave, more involved properties of the Gibbs distribution (e.g. it approaches a normal distribution) must be used.

▶ **Theorem 29.** *Suppose the counts are log-concave and non-zero. If* $\beta_{\min} \le \log \frac{c_0}{c_1}$ *and* $\beta_{\max} \ge \log \frac{c_{n-1}}{c_n}$*, then* $\Delta(k) \ge \frac{1}{n+1}$ *for all* $k = 0, \dots, n$*, and* $\log Q(\beta_{\max}) \le q := 3n\Gamma$ *where* $\Gamma := \max\{\beta_{\max}, \log \frac{c_1}{c_0}, 1\}$*. In particular, for* $\delta = \frac{1}{n+1}$*, we can solve* $P^{\delta,\varepsilon}_{\texttt{count}}$ *with cost*

$$O\Big(\min\Big\{\frac{n\Gamma \log n \log \frac{1}{\gamma}}{\varepsilon^2}, \frac{n^2 \log \frac{1}{\gamma}}{\varepsilon^2} + n \log \Gamma\Big\}\Big)$$

Proof. See full paper. ◀

Theorem 3 follows directly from Theorem 29 combined with an MCMC sampler for matchings appearing [15]. (See full paper for details).

6.1 Counting connected subgraphs

Consider a connected graph $G = (V, E)$. In [11], Guo & Jerrum described an algorithm to sample a connected subgraph $G' = (V, E')$ with probability proportional to $\prod_{f \in E'}(1 - p(f)) \prod_{f \in E - E'} p(f)$, for any weighting function $p : E \to [0, 1]$. If we set $p(f) = \frac{1}{1+e^\beta}$ for all edges f, then their algorithm samples from the Gibbs distribution with c_i being the number of connected subgraphs of G with $|E| - i$ edges. Guo & He [10] subsequently improved the algorithm runtime; we summarize their result as follows:

▶ **Theorem 30** ([10], Corollary 10)**.** *There is an algorithm to sample from the Gibbs distribution with counts* c_i *for any value of* $\beta > 0$*; the expected runtime is* $O(|E| + |E||V|e^\beta)$*.*

Proof of Theorem 1. The sequence c_i counts the number of independent sets in the co-graphic matroid, where $n = |E| - |V| + 1$. By the result of [1], this sequence c_i is log-concave; also $c_0 = 1$ so it suffices to estimates counts up to any scaling. The ratios c_{n-1}/c_n and c_1/c_0 are both at most $|E|$, since to enumerate a connected graph with $|V|$ edges we may select a spanning tree and any other edge in the graph, and to enumerate a graph with $|E| - 1$ edges we simply select an edge of G to delete.

So we can apply Theorem 29, setting $\beta_{\max} = \log |E| \ge \log \frac{c_{n-1}}{c_n}, \beta_{\min} = -\log |E| \le \log \frac{c_0}{c_1}$ and $\Gamma = \log |E|$. The definition of an FPRAS traditionally sets $\gamma = O(1)$, and here $n = |E|$. So the algorithm uses $O(\frac{|E| \log^2 |E|}{\varepsilon^2})$ samples in expectation. With these parameters, each call to the sampling oracle of Theorem 30 has runtime $O(|E|^2|V|)$. The total runtime is then $O(\frac{|E|^3|V| \log^2 |E|}{\varepsilon^2})$. ◀

The work [11] sketches an FPRAS for this problem as well; the precise complexity is unspecified and appears to be much larger than Theorem 1. We also note that Anari et al. [2] provide a general FPRAS for counting the number of independent sets in arbitrary matroids, which would include the number of connected subgraphs. This uses a very different sampling method, which is not based on the Gibbs distribution. They do not provide concrete complexity estimates for their algorithm.

6.2 Counting independent sets in bounded-degree graphs

For a graph $G = (V, E)$ of maximum degree D, let I_k denote the collection of independent sets of size k for $k = 0, \dots, |V|$. A key problem in statistical physics is to sample efficiently from I_k. Here, there is critical hardness threshold defined by $\lambda_c = \frac{(D-1)^{D-1}}{(D-2)^D} \approx e/D$, such

that, for $\beta > \lambda_c$, it is intractable to sample from the Gibbs distribution at rate λ; on the other, for $\beta < \lambda_c$, there is a polynomial-time sampler for the Gibbs distribution. We quote the following result of [6].

▶ **Theorem 31** ([6]). *Let $D \geq 3$ and $\xi > 0$ be any fixed constants. There is an algorithm to approximately sample from the Gibbs distribution at $\beta \in [0, \lambda_c - \xi]$, up to total variation distance ρ, with runtime $O(n \log n \log(n/\rho))$.*

The related problem of estimating the values i_k was considered in [7]. Based on this sampling result, they identified a related computational threshold for estimating the counts $c_k = |I_k|$. Namely, they define the threshold value $\alpha_c = \frac{\lambda_c}{1+(D+1)\lambda_c}$ and then show that, for $k > \alpha_c|V|$, it is intractable to estimate c_k or to sample approximately uniformly from I_k; on the other hand, for constant $D \geq 3$ and $\xi > 0$ and $k < (\alpha_c - \xi)|V|$, then describe an algorithm to estimate c_k in polynomial time. A follow-up work [13] provided tighter estimates for the Gibbs distribution and improved algorithms; specifically, it showed how to estimate a given count c_i for $i < (\alpha - \xi)|V|$ with runtime $\tilde{O}(n^2/\varepsilon^2)$.

A key analytical technique of [13] was to show that the Gibbs distribution for independent sets closely approximated to a normal distribution, i.e. it obeyed a type of Central Limit Theorem. Using Theorem 3.1 of [13], we have the following crude estimate:

▶ **Lemma 32** ([13]). *Let $D \geq 3$ and $\xi > 0$ be any fixed constants. There is a constant $\xi' > 0$ such that, for any $k \leq (\alpha_c - \xi)|V|$, there is some value $\beta \in [0, \lambda_c - \xi']$ with $\mu_\beta(k) \geq \Omega(1/\sqrt{|V|})$.*

By using Lemma 32, we immediately get the following result:

▶ **Theorem 33.** *Let $D \geq 3$ and $\xi > 0$ be any fixed constants. There is an algorithm to estimate all counts $c_0, \dots c_{\lfloor(\alpha_c - \xi)|V|\rfloor}$ with runtime $\tilde{O}(\frac{n^2 \log(1/\gamma)}{\varepsilon^2})$.*

Proof. We set $\beta_{\min} = 0, \beta_{\max} = \lambda_c - \xi', n = |V|$. Note that the Gibbs distribution is *not* necessarily log-concave. Since $c_0 = 1$ and clearly $c_i \leq 2^n$ for all i, we have $Q(\beta_{\max})/Q(\beta_{\min}) \leq (2^n e^{\beta_{\max} n})/1$; in particular, since $\beta_{\max} = O(1)$ (for fixed D), we have $Q(\beta_{\max})/Q(\beta_{\min}) \leq e^{O(n)}$ and we can take $q = \Theta(n)$. By Lemma 32, we have $\Delta(k) \geq \Omega(1/\sqrt{n})$ for these parameters. Thus, it suffices to solve $P^{\delta, 0.1\varepsilon}_{\texttt{count}}$ for $\delta = \Omega(1/\sqrt{n})$.

For this purpose, we will actually use the continuous-setting algorithm – it is more efficient than the general integer-setting algorithm. By Theorem 13, this algorithm has cost

$$O\Big(\frac{\min\{q, \sqrt{q \log n}\} \log \frac{q}{\delta\gamma}}{\delta\varepsilon^2} + \frac{q \log n \log \frac{1}{\gamma}}{\varepsilon^2}\Big) = O\Big(\frac{n \log^{3/2} n + n \log n \log \frac{1}{\gamma}}{\varepsilon^2}\Big).$$

Accordingly, we need to run the approximate sampler of Theorem 31 with $\rho = \text{poly}(n, 1/\varepsilon, \log \frac{1}{\gamma})$ leading to a computational complexity of $O(n \log n \log(\frac{n \log 1/\gamma}{\varepsilon}))$. With some simplification of parameters, the overall runtime becomes

$$O\Big(\frac{n^2 \log^{5/2} n \log(n/\varepsilon) + n^2 \log^2 n \log \frac{1}{\gamma} \log(\frac{n \log 1/\gamma}{\varepsilon})}{\varepsilon^2}\Big) = \tilde{O}\Big(\frac{n^2 \log \frac{1}{\gamma}}{\varepsilon^2}\Big). \qquad \blacktriangleleft$$

We note that the algorithm in [13] has this same runtime, but only estimates a *single* count c_i; our algorithm simultaneously produces estimates for all values c_i up to the threshold value $i < (\alpha_c - \xi)|V|$ with the same runtime. It also does not depend on the precise distributional properties of the Gibbs distribution; it only requires the much cruder estimate in Lemma 32.

7 Lower bounds on sample complexity

Following [16], our strategy is to construct a target instance $c^{(0)}$ surrounded by an envelope of d alternate instances $c^{(1)}, \dots, c^{(d)}$, such that solving $P^{\texttt{point}}_{\texttt{ratio}}$ or $P_{\texttt{count}}$ on an unknown instance $c^{(r)}$ distinguishes between the cases $r = 0$ and $r > 0$. On the other hand, an "indistinguishability lemma" gives a lower bound on the sample complexity of any such procedure to distinguish the distributions.

Define $\mu^{(r)}_\beta$ to be the Gibbs distribution with parameter β for instance $c^{(r)}$, and $Z^{(r)}(\beta)$ to be its partition function, and $z^{(r)} = \log Z^{(r)}$, and $\Delta^{(r)}(x) = \max_{\beta \in [\beta_{\min}, \beta_{\max}]} \mu^{(r)}_\beta(x)$. We will require that the instances are *balanced*, namely, that they satisfy the property $\prod_{r=1}^d c^{(r)}_x = (c^{(0)}_x)^d$ for all $x \in \mathcal{F}$. We also define parameters

$$U(\beta) = \prod_{r=1}^{d} \frac{Z^{(r)}(\beta)}{Z^{(0)}(\beta)}, \qquad \Psi = \max_{\beta \in [\beta_{\min}, \beta_{\max}]} \log U(\beta).$$

▶ **Lemma 34** ([16]). *Let $\mathfrak{A}$ be an algorithm which generates queries $\beta_1, \dots, \beta_T \in [\beta_{\min}, \beta_{\max}]$ and receives values $x_1, \dots, x_T$, where each x_i is drawn from μ_{β_i}. At some point the procedure stops and outputs TRUE or FALSE. The queries β_i and the stopping time T may be adaptive and may be randomized.*

Suppose that $\mathfrak{A}$ outputs TRUE on input $c^{(0)}$ with probability at least $1 - \gamma$ and outputs FALSE on inputs $c^{(1)}, \dots, c^{(d)}$ with probability at least $1 - \gamma$, for some parameter $\gamma < 1/4$.

If the instances are balanced, then the cost of $\mathfrak{A}$ on instance $c^{(0)}$ is $\Omega(\frac{d \log(1/\gamma)}{\Psi})$.

To get more general lower bounds for count estimation, we consider a problem variant called $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$, namely, to compute a vector $\hat{c} \in (\mathbb{R}_{>0} \cup \{?\})^{\mathcal{F}}$ satisfying the following two properties:

- **(i)** for all pairs x, y with $\hat{c}_x, \hat{c}_y \neq ?$, there holds $|\log \frac{\hat{c}_x}{\hat{c}_y} - \log \frac{c_x}{c_y}| \leq \varepsilon$
- **(ii)** for all x with $\Delta(x) \geq \delta$ there holds $\hat{c}_x \neq ?$.

Given a solution $(\hat{\pi}, u)$ to $P^{\delta, 0.1\varepsilon}_{\texttt{count}}$, we can solve $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ with zero sample complexity and probability one (see full paper for details). Problem $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ is easier than $P^{\delta,\varepsilon}_{\texttt{count}}$ (up to constant factors in parameters), in two ways: first, it does not require any specific normalization of the counts, only pairwise consistency. Second, it only provides approximation guarantees for c_x if $\Delta(x) \geq \delta$, while $P^{\delta,\varepsilon}_{\texttt{count}}$ provides meaningful bounds over a wide range of scales.

▶ **Corollary 35.**

- **(a)** *Suppose that $|z^{(0)}(\beta_{\min}, \beta_{\max}) - z^{(r)}(\beta_{\min}, \beta_{\max})| > 2\varepsilon$ for all $r = 1, \dots, d$. Then any algorithm for $P^{\texttt{point}}_{\texttt{ratio}}$ must have cost $\Omega(\frac{d \log(1/\gamma)}{\Psi})$ on instance $c^{(0)}$.*
- **(b)** *Suppose that for each $r = 1, \dots, d$ there are x, y with $\Delta^{(0)}(x), \Delta^{(0)}(y) \geq \delta$, and $|\log(c^{(0)}_x / c^{(0)}_y) - \log(c^{(r)}_x / c^{(r)}_y)| > 2\varepsilon$. (We refer to the values x, y as the* witnesses *for r.) Then any algorithm for $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ must have cost $\Omega(\frac{d \log(1/\gamma)}{\Psi})$ on instance $c^{(0)}$.*

Proof. We show how to convert these algorithms into procedures distinguishing $c^{(0)}$ from $c^{(1)}, \dots, c^{(d)}$:

- **(a)** Given a solution $\hat{Q}(\beta_{\max})$ to $P^{\texttt{point}}_{\texttt{ratio}}$, output TRUE if $|\log \hat{Q}(\beta_{\max}) - z^{(0)}(\beta_{\min}, \beta_{\max})| \leq \varepsilon$, else output FALSE.
- **(b)** Given a solution $\hat{c}$ to $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$, output TRUE if $\hat{c} \neq ?$ for all x with $\Delta^{(0)}(x) \geq \delta$, and every pair x, y with $\Delta(x), \Delta(y) \geq \delta$ satisfy $|\log(\hat{c}_x / \hat{c}_y) - \log(c^{(0)}_x / c^{(0)}_y)| \leq \varepsilon$, else output FALSE. ◀

By applying Corollary 35 to carefully constructed instances, we will show the following:

▶ **Theorem 36.** *Let $n \geq n_0, q \geq q_0, \varepsilon < \varepsilon_0, \delta < \delta_0, \gamma < 1/4$ for certain absolute constants $n_0, q_0, \varepsilon_0, \delta_0$. There are problem instances μ which satisfy the given bounds n and q such that:*

(a) $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ *requires cost* $\Omega(\frac{\min\{q+\sqrt{q}/\delta, n^2+n/\delta\}\log\frac{1}{\gamma}}{\varepsilon^2})$*, and μ is integer-valued.*

(b) $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ *requires cost* $\Omega(\frac{(1/\delta+\min\{q,n^2\})\log\frac{1}{\gamma}}{\varepsilon^2})$*, and μ is log-concave.*

(c) $P^{\texttt{point}}_{\texttt{ratio}}$ *requires cost* $\Omega(\frac{\min\{q,n^2\}\log\frac{1}{\gamma}}{\varepsilon^2})$*, and μ is log-concave.*

(d) $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ *requires cost* $\Omega(\frac{(q+\sqrt{q}/\delta)\log\frac{1}{\gamma}}{\varepsilon^2})$.

(e) $P^{\texttt{point}}_{\texttt{ratio}}$ *requires cost* $\Omega(\frac{q\log\frac{1}{\gamma}}{\varepsilon^2})$.

The lower bounds on $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ immediately imply lower bounds on $P^{\delta,\varepsilon}_{\texttt{count}}$, in particular, they give Theorems 5 and 7. Result (e) was already shown in [16], but we include it here since it is a corollary of other results.

The proofs and constructions appear in the full paper.

References

1 K. Adiprasito, J. Huh, and E. Katz. Hodge theory for combinatorial geometries. *Annals of Mathematics*, 188(2):381–452, 2018.

2 N. Anari, K. Liu, S. O. Gharan, and C. Vinzant. Log-concave polynomials II: High-dimensional walks and an FPRAS for counting bases of a matroid. In *Proc. 51st annual ACM Symposium on Theory of Computing (STOC)*, pages 1–12, 2019.

3 R. E. Belardinelli and V. D. Pereyra. Wang-Landau algorithm: A theoretical analysis of the saturation of the error. *The Journal of Chemical Physics*, 127(18):184105, 2007.

4 I. Bezáková, D. Štefankovič, V. V. Vazirani, and E. Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. *SIAM J. Comput.*, 37:1429–1454, 2008.

5 Charlie Carlson, Ewan Davies, Alexandra Kolla, and Will Perkins. Computational thresholds for the fixed-magnetization ising model. In *Proc. 54th annual ACM Symposium on Theory of Computing (STOC)*, pages 1459–1472, 2022.

6 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of Glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proc. 53rd annual ACM Symposium on Theory of Computing (STOC)*, pages 1537–1550, 2021.

7 Ewan Davies and Will Perkins. Approximately counting independent sets of a given size in bounded-degree graphs. In *Proc. 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 62:1–62:18, 2021.

8 W. Feng, H. Guo, Y. Yin, and C. Zhang. Fast sampling and counting k-SAT solutions in the local lemma regime. In *Proc. 52nd annual ACM Symposium on Theory of Computing (STOC)*, pages 854–867, 2020.

9 G. Fort, B. Jourdain, E. Kuhn, T. Leliévre, and G. Stoltz. Convergence of the Wang-Landau algorithm. *Mathematics of Computation*, 84(295):2297–2327, 2015.

10 H. Guo and K. He. Tight bounds for popping algorithms. *Random Struct. Algorithms*, 57(2):371–392, 2020.

11 H. Guo and M. Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019.

12 M. Huber. Approximation algorithms for the normalizing constant of Gibbs distributions. *The Annals of Applied Probability*, 25(2):974–985, 2015.

13 Vishesh Jain, Will Perkins, Ashwin Sah, and Mehtaab Sawhney. Approximate counting and sampling via local central limit theorems. In *Proc. 54th annual ACM Symposium on Theory of Computing (STOC)*, pages 1473–1486, 2022.

14 M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.

15 M. Jerrum and A. Sinclair. The Markov Chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.

16 V. Kolmogorov. A faster approximation algorithm for the Gibbs partition function. *Proceedings of Machine Learning Research*, 75:228–249, 2018.

17 Pedro Ojeda, Martin E Garcia, Aurora Londoño, and Nan-Yow Chen. Monte Carlo simulations of proteins in cages: influence of confinement on the stability of intermediate states. *Biophysical journal*, 96(3):1076–1082, 2009.

18 L. N. Shchur. On properties of the Wang-Landau algorithm. *Journal of Physics: Conference Series*, 1252, 2019.

19 D. Štefankovič, S. Vempala, and E. Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *J. of the ACM*, 56(3) Article #18, 2009.

20 F. Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, 86(10):2050–2053, 2001.