Near-Linear Time Samplers for Matroid Independent Sets with Applications

Xiaoyu Chen
State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China

Heng Guo
School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, UK

Xinyuan Zhang
State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China

Zongrui Zou
State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China

Abstract

We give a $\widetilde{O}(n)$ time almost uniform sampler for independent sets of a matroid, whose ground set has n elements and is given by an independence oracle. As a consequence, one can sample connected spanning subgraphs of a given graph $G = (V, E)$ in $\widetilde{O}(|E|)$ time, whereas the previous best algorithm takes $O(|E|\,|V|)$ time. This improvement, in turn, leads to a faster running time on estimating all-terminal network reliability. Furthermore, we generalise this near-linear time sampler to the random cluster model with $q \leq 1$.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains

Keywords and phrases Network reliability, Random cluster modek, Matroid, Bases-exchange walk

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.32

Category RANDOM

Funding *Heng Guo*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 947778).

Acknowledgements We would like to thank the hospitality of NII Shonan meeting No. 186, where some of the discussion took place.

1 Introduction

Let $\mathcal{M} = ([n], \mathcal{I})$ be a matroid of rank r and $\boldsymbol{\lambda} \in \mathbb{R}^n_{>0}$ be the external fields (namely weights for the ground set elements). Denote its set of bases by $\mathcal{B} = \mathcal{B}(\mathcal{M})$, and by $\mathcal{I} = \mathcal{I}(\mathcal{M})$ the set of independent sets. Suppose that we want to sample a random base $B \in \mathcal{B}$ from the following distribution:

$$\forall B \in \mathcal{B}, \quad \mu_{\mathcal{B},\boldsymbol{\lambda}}(B) \propto \prod_{i \in B} \lambda_i.$$

There is a natural Markov chain, namely the bases-exchange walk (also known as the down-up walk) [8], that converges to the distribution above. Anari, Liu, Oveis Gharan, and Vinzant [2] showed that this chain mixes in polynomial time. Subsequently, Cryan, Guo, and Mousa [7] and a follow up work by Anari, Liu, Oveis Gharan, Vinzant, and Vuong [3] refined the mixing time to the optimal $O(r \log r)$.

In this work, we focus on another important distribution associated with the matroid $\mathcal{M}$, namely, the distribution $\mu_{\mathcal{M},\boldsymbol{\lambda}}$ over the independent sets of $\mathcal{M}$:

$$\forall S \in \mathcal{I}, \quad \mu_{\mathcal{M},\boldsymbol{\lambda}}(S) \propto \prod_{i \in S} \lambda_i. \tag{1}$$

As suggested by the previous work [3], in order to sample from $\mu_{\mathcal{M},\boldsymbol{\lambda}}$, we may construct another matroid $M_{\mathcal{I}}$ so that there is a one-to-one correspondence between the bases of $\mathcal{M}_{\mathcal{I}}$ and the independent sets of $\mathcal{M}$. Therefore, we may use the bases-exchange walk on $\mathcal{M}_{\mathcal{I}}$ to approximately generate samples from $\mu_{\mathcal{M},\boldsymbol{\lambda}}$ within $O(n \log n)$ steps.

However, an efficient implementation of the bases-exchange walk on $\mathcal{M}_{\mathcal{I}}$ is far from trivial. Note that the input matroid is usually given succinctly. For example, for a graphic matroid, it is sufficient to input the associated graph, rather than a list of all the bases or independent sets, which can be exponentially large in the size of the graph. The most common and standard way is to give the matroid by an independence oracle $\mathcal{O}_I$. Upon receiving a subset of elements, the independence oracle $\mathcal{O}_I$ returns whether or not the set is independent. Given $\mathcal{O}_I$, the naïve implementation of the bases-exchange walk requires $O(n)$ oracle calls per step. Depending on the application, there may be even further cost of implementing the oracle. This prevents us from getting a near-linear time sampler for many potential applications.

To get a faster algorithm, Anari, Liu, Oveis Gharan, Vinzant, and Vuong considered a different oracle $\mathcal{O}'$ [3]. One can query $\mathcal{O}'$ a subset $S \subseteq [n]$ with the promise that S contains at most one circuit. If a circuit exists, $\mathcal{O}'$ will output a uniformly random element of the circuit. In [3], it is showed that there is a sampling algorithm for $\mu_{\mathcal{M},\boldsymbol{\lambda}}$ using $O(n \log n)$ queries in total of $\mathcal{O}'$. However the downside is that this oracle $\mathcal{O}'$ is typically more difficult to implement than the independence oracle $\mathcal{O}_I$. In our applications to be discussed later (network reliability and the random cluster model), the independence oracle $\mathcal{O}_I$ can be implemented in logarithmic time (at least in an amortized sense), whereas it appears to require at least linear time to implement $\mathcal{O}'$. A straightforward implementation of $\mathcal{O}'$ requires $O(r)$ calls to $\mathcal{O}_I$, where r is the rank of the matroid $\mathcal{M}$. This leads to a sampling algorithm using $O(rn \log n)$ oracle calls to $\mathcal{O}_I$. Note that the rank r is often not a constant and can be as large as $\Omega(n)$.

We give a sampler which requires $O((1+\lambda_{\max})n \log(n/\varepsilon))$ oracle calls in expectation to $\mathcal{O}_I$, where $\lambda_{\max} := \max_{i \in [n]} \lambda_i$ and ε is the sampling error. This improves the previously best $O(rn \log n)$ running time [3]. We use the *total variation distance* (TV distance) to measure the distance between two distributions μ and ν over a finite space Ω, defined by $\mathrm{D}_{\mathrm{TV}}(\mu, \nu) := \frac{1}{2} \sum_{X \in \Omega} |\mu(X) - \nu(X)|$.

▶ **Theorem 1.** *Equipped with the independence oracle $\mathcal{O}_I$ of a matroid $\mathcal{M} = ([n], \mathcal{I})$, there exists an algorithm that takes external fields $\boldsymbol{\lambda} \in \mathbb{R}^n_{>0}$ and $\varepsilon \in (0,1)$ as inputs, and outputs a random set $S \in \mathcal{I}$ satisfying $\mathrm{D}_{\mathrm{TV}}(\mu_{\mathcal{M},\boldsymbol{\lambda}}, S) \le \varepsilon$. In expectation, it runs in $O((1+\lambda_{\max})n \log(n/\varepsilon)(\log n + t_{\mathcal{O}_I}))$ time, where $t_{\mathcal{O}_I}$ is the time to answer a query by the independence oracle $\mathcal{O}_I$ and $\lambda_{\max} := \max_{i \in [n]} \lambda_i$.*

The proof of Theorem 1 is given in Section 3.

▶ Remark 2. In particular, instead of the independence oracle $\mathcal{O}_I$, for our algorithm in Theorem 1, it suffices to have a data structure maintaining a set $S \subseteq [n]$ which supports:

- to *insert* an element to S;
- to *delete* an element from S;
- and to *query* if $S \in \mathcal{I}$.

Given such a data structure, $t_{\mathcal{O}_I}$ in Theorem 1 can be substituted by the worst case or amortized running time of these operations.

The crux of Theorem 1 is a fast implementation of the transition step of the bases-exchange chain for $\mathcal{M}_{\mathcal{I}}$. Note that the transition of a Markov chain is in itself yet another sampling problem. We design a rejection sampling procedure for this latter sampling task. There is a constant upper bound for the rejection probability (see Lemma 14), guaranteeing a success with high probability in logarithmic trials.

A consequence of our algorithm is a faster approximation algorithm for the *all-terminal network reliability*. The problem is defined as follows. Given a connected undirected graph $G = (V, E)$ and failure probabilities $\boldsymbol{p} \in \mathbb{R}^E_{>0}$, the all-terminal network reliability $Z_{\mathrm{rel}}(G, \boldsymbol{p})$ is the probability that the graph is connected if each edge e fails (i.e. is removed) independently with probability p_e. Formally, for $S \subseteq E$, let

$$\mathrm{wt}(S) := \mathbf{1}[G[E \setminus S] \text{ is connected}] \cdot \prod_{e \in S} p_e \prod_{f \in E \setminus S} (1 - p_f), \tag{2}$$

where $G[E \setminus S]$ is the spanning subgraph of G on $E \setminus S$. Then the reliability of the network is

$$Z_{\mathrm{rel}}(G, \boldsymbol{p}) := \sum_{S \subseteq E} \mathrm{wt}(S).$$

By standard techniques [16, 20, 17], estimating $Z_{\mathrm{rel}}(G, \boldsymbol{p})$ can be reduced to approximate sampling of the (weighted) distribution of connected spanning subgraphs:

$$\forall S \subseteq E, \quad \mu^{\mathrm{NR}}_{G,\boldsymbol{p}}(S) \propto \mathrm{wt}(S). \tag{3}$$

The study of the computational complexity of network reliability was initiated by Valiant [21]. Exact evaluation of the all-terminal version is known to be #**P**-hard [15, 19]. Guo and Jerrum [11] gave the first fully polynomial-time randomized approximate scheme (FPRAS) using the *partial rejection sampling* framework [12]. This algorithm samples from $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}$ in $O(|E| + \frac{p_{\max}|V||E|}{1-p_{\max}})$ time in expectation [10] where $p_{\max} := \max_i p_i$ is the maximum failure probability, and this bound is tight for the technique. It is also worth mentioning that, using the result in [3] directly, it is possible to get an $\widetilde{O}(|V|\,|E|)$ time sampler, whose running time is of roughly the same order as the partial rejection sampling algorithm.

Using Theorem 1, we obtain an $\widetilde{O}(|V|)$ speed-up to sample from $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}$. This gives the first near-linear time sampler for connected spanning subgraphs.

▶ **Corollary 3.** *Let $G = (V, E)$ be a connected graph with n vertices and m edges. Let $\boldsymbol{p} \in (0, 1)^E$ be the failure probabilities for edges. There is an algorithm that takes G, $\boldsymbol{p}$ and $\varepsilon \in (0, 1)$ as input, and outputs a random subset $S \subseteq E$ such that $\mathrm{D_{TV}}\left(\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}, S\right) \le \varepsilon$ in $O\left(\frac{m(\log^3 n + \log \frac{1}{\varepsilon})}{1-p_{\max}}\right)$ time in expectation, where $p_{\max} := \max_{e \in E} p_e$.*

Proof. Let $\Omega := \{S \subseteq E \mid \mu^{\mathrm{NR}}_{G,\boldsymbol{p}}(S) > 0\}$ be the support of $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}$. Recall that $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}(S) > 0$ if and only if $G[E \setminus S]$ is connected. This means that there is a spanning tree T of G contained in $E \setminus S$. Note that spanning trees are bases of the graphic matroid $\mathcal{M}_G$ of a graph G. Hence, let $\mathcal{M}_{\mathrm{NR}} := (E, \Omega)$, it holds that $\mathcal{M}_{\mathrm{NR}}$ is the dual matroid of $\mathcal{M}_G$, namely the co-graphic matroid. This also means that $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}} = \mu_{\mathcal{M},\boldsymbol{\lambda}}$ for $\mathcal{M} = \mathcal{M}_{\mathrm{NR}}$ and $\lambda_e = \frac{p_e}{1-p_e}, \forall e \in E$ as defined in (1).

It remains to implement the independence oracle efficiently. For this we use dynamic data structures for connectivity of graphs, which is a topic that has been extensively studied. For $\mathcal{M}_{\mathrm{NR}}$, as in Remark 2, we implement the independence oracle $\mathcal{O}_I$ with amortized cost $t_{\mathcal{O}_I} = O(\log^2 n)$ by using the data structure in [22, Section 3] directly. The corollary follows by combining this with Theorem 1. ◀

As we can see in the proof, the advantage of our algorithm is that the independence oracle can be implemented in amortized logarithmic time. In contrast, it requires at least linear time to implement the oracle $\mathcal{O}'$ of Anari, Liu, Oveis Gharan, Vinzant, and Vuong [3]. Our sampling algorithm in Theorem 1 calls the independence oracle only a near-linear number of times, which is crucial to obtain the overall near-linear running time.

Using the counting to sampling reduction in [10], Corollary 3 implies an FPRAS that outputs an $(1 \pm \varepsilon)$-approximation of $Z_{\mathrm{rel}}(G, \boldsymbol{p})$ in time $O\left(\frac{mn \log^4(n)}{\varepsilon^2(1-p_{\max})} \log \frac{1}{1-p_{\max}}\right)$. As before, this improves the previous best running time by a factor of $\widetilde{O}(n)$. We also note that the running time in Corollary 3 is linear in $(1-p_{\max})^{-1}$. This factor comes from our rejection sampling implementation of the down-up walk, whereas the naïve implementation of the down-up walk has logarithmic dependence. On the other hand, in most applications, $1-p_{\max} = \Omega(1)$. For example, for the uniform distribution over connected spanning subgraphs, $p_e = 1/2$ for all e. Moreover, in the simulated annealing counting to sampling reduction [10], we do not need to consider p_e larger than $p_{\max}$. Thus, in this reduction, there is no extra slowdown due to this linear dependence on $1 - p_{\max}$. In any case, we leave improving this dependence for near-linear time samplers as an open problem.

The distribution $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}$ in (3) is a special case of the random cluster model on the graph G with parameter $q = 0$ [9]. More generally, for a matroid $\mathcal{M} = ([n], \mathcal{I})$ with a rank function $\mathrm{rk}(\cdot)$, the random cluster model with parameter $q \geq 0$ and external fields $\boldsymbol{\lambda} \in \mathbb{R}^n_{>0}$ is defined as follows: for $S \subseteq X$,

$$\pi_{RC,q,\boldsymbol{\lambda}}(S) \propto q^{-\mathrm{rk}(S)} \prod_{x_i \in S} \lambda_i. \tag{4}$$

For $q = 0$, the support of the distribution in (4) must have the highest rank. For a graphic matroid over a graph G, this means that $G[S]$ must be connected, namely S in (4) corresponds to $E \setminus S$ in $\mu^{\mathrm{NR}}_{G,\boldsymbol{p}}$.

We also extend our near-linear time sampler to random cluster models with $q \leq 1$. Note that the rank of S plays an important role in the distribution in $\pi_{RC,q,\boldsymbol{\lambda}}$ (4). Thus, instead of the independence oracle, here we use a rank oracle, which upon a query S returns the rank of S.

▶ **Theorem 4.** *Let $0 \leq q \leq 1$ be a parameter. Equipped with the rank oracle $\mathcal{O}_r$ of a matroid $\mathcal{M} = ([n], \mathcal{I})$, there exists an algorithm that takes external fields $\boldsymbol{\lambda} \in \mathbb{R}^n_{>0}$ and $\varepsilon \in (0,1)$ as inputs, and outputs a random set S such that $\mathrm{D_{TV}}(\pi_{RC,q}, S) \leq \varepsilon$. It runs in $O\left((1 + \lambda_{\min}^{-1}) n \log(n/\varepsilon)(\log n + t_{\mathcal{O}_r})\right)$ time in expectation, where $t_{\mathcal{O}_r}$ is the time to answer a query by the rank oracle $\mathcal{O}_r$ and $\lambda_{\min} := \min_{i \in [n]} \lambda_i$.*

Theorem 4 is proved in Section 4.

Similar to Remark 2, it suffices to replace the rank oracle by a data structure that supports insertion/deletion of elements and query if the rank changes after removing an element. For graphs, the rank oracle, once again, can be implemented using the data structure in [22]. This is because $\mathrm{rk}(S) = |V| - 1 + \kappa(E) - \kappa(S)$, where $\kappa(S)$ is the number of connected components in $G[S]$. Thus the rank change query in graphs is exactly the same as asking if u and v are connected after removing an edge (u, v). The amortized cost of using this data structure is $O(\log^2 n)$.

▶ **Corollary 5.** *Let $G = (V, E)$ be a graph with n vertices and m edges. Let $q \leq 1$ be a parameter and $\boldsymbol{\lambda} \in \mathbb{R}^E_{>0}$ be the external fields on edges. There is an algorithm that takes G, $\boldsymbol{\lambda}$ and $\varepsilon \in (0,1)$ as input, and outputs a random subset $S \subseteq E$ such that $\mathrm{D_{TV}}(\pi_{RC,q}, S) \leq \varepsilon$ in $O\left((1 + \lambda_{\min}^{-1}) m (\log^3 n + \log \frac{1}{\varepsilon})\right)$ time in expectation, where $\lambda_{\min} := \min_{i \in [n]} \lambda_i$.*

2 Preliminaries

2.1 Matroid

Matroid is an abstract combinatorial structure that generalizes the notion of linear independence. It is usually specified by a pair $\mathcal{M} = (U, \mathcal{I})$ where U is a ground set and $\mathcal{I} \subseteq 2^U$ is a collection of subsets of U. The subsets in $\mathcal{I}$ are known as the independent sets of the matroid and satisfy the following axioms:

- $\varnothing \in \mathcal{I}$;
- if $S \in \mathcal{I}$, $T \subseteq S$, then $T \in \mathcal{I}$;
- if $S, T \in \mathcal{I}$ and $|S| > |T|$, then there is an element $i \in S \setminus T$ such that $T \cup \{i\} \in \mathcal{I}$.

The first axiom ensures that $\mathcal{I}$ is non-empty. The second shows that $\mathcal{I}$ is downward closed, and the third implies that the cardinality of maximal independent sets are the same. This maximum cardinality is known as the *rank* of the matroid $\mathcal{M}$. The set of *bases* $\mathcal{B} = \mathcal{B}(\mathcal{M})$ is the collection of independent sets of maximum cardinality. The rank also extends as a function to subsets of U. For $S \subseteq U$, $\mathrm{rk}(S)$ is defined as the size of the maximum independent sets contained in S.

For a matroid $\mathcal{M} = (U, \mathcal{I})$, its *dual matroid* $M^\star = (U, \mathcal{I}^\star)$ has the same ground set U with the collection of independent sets $\mathcal{I}^\star := \{S \subseteq U \mid \exists B \in \mathcal{B}(\mathcal{M}), B \subseteq U \setminus S\}$. By definition, every base $B^\star$ of $M^\star$ is the complement of the base $B = U \setminus B^\star$ of $\mathcal{M}$ and vice versa.

2.2 Strongly log-concave polynomial

Let $f \in \mathbb{R}[x_1, x_2, \ldots, x_n]$ be a polynomial with non-negative coefficients. f is called

- *r-homogeneous* if the degree of every monomial in f is r;
- *multiaffine* if every variable appears with degree no more than 1;
- *log-concave over the first orthant* (or *log-concave* for short) if $\log f$ is concave over $\mathbb{R}^n_{>0}$, i.e., for $x, y \in \mathbb{R}^n_{>0}$ and $\lambda \in (0, 1)$,

$$f(\lambda x + (1-\lambda)y) \geq f(x)^\lambda f(y)^{1-\lambda};$$

- *strongly log-concave* if f is either vanishes or log-concave after taking any sequence of partial derivatives.

The notion of strong log-concavity is introduced by Gurvitz [14, 13]. For a homogeneous polynomial, it turns out to be equivalent to related notions of *complete log-concavity* by Anari, Oveis Gharan, and Vinzant [4] and *Lorentzian* by Brändén and Huh [6]. See [6, Theorem 2.30]. For simplicity, we will not define the latter two.

A well known fact is that affine transform $T : \mathbb{R}^m \to \mathbb{R}^n$ (i.e., $T(\boldsymbol{y}) = A\boldsymbol{y} + \boldsymbol{b}$ for some $A \in \mathbb{R}^{n \times m}$ and $\boldsymbol{b} \in \mathbb{R}^n$) preserves log-concavity.

▶ **Lemma 6** ([4, Lemma 2.1]). *If $f \in \mathbb{R}[x_1, \cdots, x_n]$ is log-concave and $T : R^m \to R^n$ is an affine transformation such that $T(R^m_{>0}) \subseteq R^n_{>0}$, then $f(T(y_1, \cdots, y_m)) \in \mathbb{R}[y_1, \cdots y_m]$ is log-concave.*

As observed in [3], for a multiaffine polynomial f, its partial derivative is given by

$$\partial_1 f = \lim_{c \to \infty} \frac{f(c, x_2, \cdots, x_n)}{c},$$

which means the derivatives of a multiaffine log-concave polynomial are limits of log-concave polynomials, which are also log-concave by definition. It implies that a multiaffine log-concave polynomial is automatically strongly log-concave.

▶ **Fact 7** ([3]). *If polynomial f is multiaffine and log-concave, then f is strongly log-concave.*

We note that most polynomials we consider are multiaffine, which means that log-concavity and strong log-concavity are equivalent within the scope of this work.

2.3 Polynomial and distribution

Let μ be a distribution over $2^{[n]}$. The generating function of μ is given by

$$g_\mu = \sum_{S \subseteq [n]} \mu(S) \prod_{i \in S} x_i.$$

It is known that multiaffine polynomials are closely related to generating polynomials of distribution. Let f be a multiaffine polynomial $f \in \mathbb{R}[x_1, x_2, \ldots, x_n]$ with non-negative coefficients. If $f \neq 0$, there exists a distribution μ over $2^{[n]}$ such that its generating polynomial is identical to f up to a scaling factor. Hence, we may also say f is the generating polynomial of μ.

Furthermore, if the generating function g_μ of μ is r-homogeneous and log-concave, then the support of μ must be the set of bases of a matroid [6].

2.4 Down-up walk

Let μ be a distribution over $\binom{[n]}{r}$. Let $\Omega(\mu) := \{S \subseteq [n] \mid \mu(S) > 0\}$ be the support of μ. A classical method for sampling from this homogeneous distribution is the down-up walk, described below.

▶ **Definition 8.** *For a distribution μ over $\binom{[n]}{r}$, the* down-up walk *P updates a configuration $S \in \binom{[n]}{r}$ according to the following rule:*

1. *select a subset $T \subseteq S$ of size $r-1$ uniformly at random;*
2. *update S to S' by selecting $S' \supseteq T$ with probability proportional to $\mu(S')$.*

When the support of μ is the set of bases of a matroid, this walk is also known as the *bases-exchange walk*.

If the down-up walk P connects $\Omega(\mu)$, then μ is its unique stationary distribution. Its *mixing time* is defined by

$$t_{\mathrm{mix}}(\varepsilon) := \min\left\{ t \,\middle|\, \max_{S \in \Omega(\mu)} \mathrm{D_{TV}}\left(P^t(S, \cdot), \mu\right) \leq \varepsilon \right\}.$$

The down-up walk mixes rapidly if g_μ is (strongly) log-concave [7, 3].

▶ **Proposition 9** ([3, Theorem 1]). *If g_μ is r-homogeneous and log-concave, the mixing time of the down-up walk can be bounded by $t_{\mathrm{mix}}(\varepsilon) = O(r \log(r/\varepsilon))$.*

3 Our algorithm

In this section, we prove Theorem 1. Our main tool is the down-up walk in Definition 8. As the uniform distribution over independent sets is not homogeneous, in Section 3.1 we first consider a standard homogenization, namely its polarized version. Then standard results imply that the down-up walk for the polarized homogeneous distribution mixes in time $O(n \log n)$. In Section 3.2, we show how to implement the down-up walk with $\widetilde{O}(1)$ cost. With these ingredients, the proof of Theorem 1 is given at the end of this section.

3.1 Down-up walk for polarized polynomial

Let $\mathcal{M} = ([n], \mathcal{I})$ be a rank-r matroid and $\boldsymbol{\lambda} \in \mathbb{R}^n_{>0}$ be the external fields. Consider

$$g(x_1, \cdots, x_n) := \sum_{S \in \mathcal{I}} \prod_{i \in S} x_i.$$

It is straightforward to verify that $g(\lambda_1 x_1, \cdots, \lambda_n x_n)$ is the generating polynomial of $\mu_{\mathcal{M},\boldsymbol{\lambda}}$.

Note that g is not homogeneous, which means that we may not directly employ the down-up walk to sample from the distribution $\mu_{\mathcal{M},\boldsymbol{\lambda}}$. However, there is a homogeneous variant of g,

$$g_h(y, x_1, \cdots, x_n) := \sum_{S \in \mathcal{I}} y^{n-|S|} \prod_{i \in S} x_i. \tag{5}$$

As a key step in the proof of Mason's ultra-log-concavity conjecture for independent sets of matroid [1, 6], the following result is proved.

▶ **Lemma 10** ([1, Theorem 4.1]). *The polynomial g_h in (5) is strongly log-concave.*

However, g_h is not multiaffine, which means that it is not a generating polynomial of any distribution. Instead, we consider the polarized version of g_h.

▶ **Lemma 11.** *If g_h in (5) is strongly log-concave, then the following polynomial is also strongly log-concave:*

$$g_p(x_1, \cdots, x_n, y_1, \cdots, y_n) = \sum_{S \in \mathcal{I}} \frac{e_{n-|S|}(\boldsymbol{y})}{\binom{n}{|S|}} \prod_{i \in S} x_i, \tag{6}$$

where $e_k(\boldsymbol{y}) = \sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} \prod_{j=1}^{k} y_{i_j}$ is the k-th elementary symmetric polynomial.

We remark that Lemma 11 is a special case of [6, Proposition 3.1], and it is more explicitly derived in [18, Section 6.6].

The polarized polynomial in (6) corresponds back to a distribution. Let $X := \{x_1, \cdots, x_n\}$ denote elements in $\mathcal{M}$ and $Y := \{y_1, \cdots, y_n\}$ denote the auxiliary variables introduced by polarization. Let π be the distribution over subsets of $X \cup Y$ corresponding to the generating polynomial $g_p(\lambda_1 x_1, \ldots, \lambda_n x_n, y_1, \ldots, y_n)$. Then the support of π is given by

$$\Omega(\pi) = \{A \cup B \mid A \in \mathcal{I}, B \subseteq Y, |A| + |B| = n\}.$$

Furthermore, for every $S = A \cup B \in \Omega(\pi)$, it holds that

$$\pi(S) \propto \frac{1}{\binom{n}{|A|}} \prod_{x_i \in A} \lambda_i,$$

and $\sum_{S : S \cap X = A} \pi(S) = \mu_{\mathcal{M},\boldsymbol{\lambda}}(A)$.

Therefore, to sample from $\mu_{\mathcal{M},\boldsymbol{\lambda}}$ within a TV distance of ε, it suffices to sample $S \in \Omega(\pi)$ such that $\mathrm{D}_{\mathrm{TV}}(S, \pi) \le \varepsilon$, and then return $S \cap X$ as the result.

Note that g_p is homogeneous and multiaffine. Moreover, according to Lemma 10 and Lemma 11, g_p is strongly log-concave. Thus, the polynomial $g_p(\lambda_1 x_1, \cdots, \lambda_n x_n, y_1, \cdots, y_n)$ is also log-concave by Lemma 6. Hence, by Proposition 9, we have the following result, which gives a powerful framework to build fast sampling algorithm for independent set of matroid.

▶ **Lemma 12.** *The down-up walk P of π mixes in time $O(n \log(n/\varepsilon))$.*

Algorithm 1 a step of down-up walk P on π.

1 select a subset $T \subseteq S$ of size $n-1$ uniformly at random;
2 update S to S' by selecting random $S' \supseteq T$ according to the following law:

$$\mathbf{Pr}\left[S'\right] \propto \mathbf{1}[S' \cap X \in \mathcal{I}] \times \begin{cases} \frac{1}{\binom{n}{|T\cap X|}}, & S' \setminus T \in Y; \\ \frac{\lambda_i}{\binom{n}{|T\cap X|+1}}, & S' \setminus T = \{x_i\} \text{ for some } x_i \in X. \end{cases} \tag{7}$$

We note that the mixing time $O(n \log(n/\varepsilon))$ does not readily imply a sampler which runs in $O(n \log(n/\varepsilon))$ time, as it may take $\omega(1)$ time to implement a single transition step of P. According to Definition 8, in each step, P updates a state $S \in \Omega(\pi)$ as in Algorithm 1.

The main obstacle is to implement the second step in Algorithm 1 efficiently. A naïve approach checks whether $(\{x_i\} \cup T) \cap X \in \mathcal{I}$ for each $x_i \in X \setminus T$ by calling the independence oracle $\mathcal{O}_I$, and then generates a random sample from all "feasible" x_i and together with all of $y_i \in Y \setminus T$ according to the desired distribution in (7). In the worst case, this gives an $O(n)$ overhead and the running time of the sampling algorithm becomes $O(n^2 \log n)$.

3.2 A fast implementation of the down-up walk

Our main contribution is an efficient implementation of the down-up walk P on π, where each step of P takes constant time in expectation given an independence oracle $\mathcal{O}_I$. In fact, the implementation task is yet another sampling problem from the distribution in (7), and we do so by rejection sampling, described in Algorithm 2.

Algorithm 2 implementation for the second step of the down-up walk P.

input : a subset $T \subseteq X \cup Y$ of size $n-1$ such that $T \cap X \in \mathcal{I}$
output : a random configuration S according to the distribution defined in (7)
1 **while** *true* **do**
2 propose an element $e \in (X \cup Y) \setminus T$ according to the following distribution ν:

$$\forall e \in (X \cup Y) \setminus T, \quad \nu(e) \propto \begin{cases} \lambda_i, & e = x_i \in X \setminus T; \\ \frac{n-|T\cap X|}{1+|T\cap X|}, & e \in Y \setminus T. \end{cases} \tag{8}$$

3 **if** $(T \cup \{e\}) \cap X \in \mathcal{I}$ **then**
4 **return** $S = T \cup \{e\}$;

The correctness of Algorithm 2 is straightforward.

▶ **Fact 13.** *The state S produced by Algorithm 2 follows the distribution defined in* (7).

In terms of efficiency, the **while** loop in Algorithm 2 is anticipated to execute for a constant number of rounds in expectation. This is because the rejection probability is upper bounded by a constant, as shown by the next lemma.

▶ **Lemma 14.** *It holds that* $\mathbf{Pr}_{e\sim\nu}\left[(T \cup \{e\}) \cap X \notin \mathcal{I}\right] \leq \frac{\lambda_{\max}}{1+\lambda_{\max}}$, *where* $\lambda_{\max} = \max_{i\in[n]} \lambda_i$.

Proof. Suppose $|T \cap X| = k$. Note that if $e \in Y \setminus T$, then $(T \cup \{e\}) \cap X \in \mathcal{I}$. This means

$$\begin{aligned}\mathbf{Pr}_{e\sim\nu}\left[(T\cup\{e\})\cap X \notin \mathcal{I}\right] &\le \sum_{x_i\in X\setminus T} \frac{\lambda_i}{\sum_{x_i\in X\setminus T}\lambda_i + \sum_{y\in Y\setminus T}\frac{n-k}{1+k}} \\ &= \frac{\sum_{x_i\in X\setminus T}\lambda_i}{\sum_{x_i\in X\setminus T}(1+\lambda_i)} \le \frac{\lambda_{\max}}{1+\lambda_{\max}},\end{aligned}$$

where the equality is due to $|Y \cap T| + k = n - 1$. ◀

Now, we are ready to prove Theorem 1.

Proof of Theorem 1. Our algorithm is just running Algorithm 1 for $O(n\log(n/\varepsilon))$ steps and then output $S \cap X$. Line 2 of Algorithm 1 is implemented by Algorithm 2, to get a random state S. By Lemma 12 and Fact 13, it holds that $\mathrm{D}_{\mathrm{TV}}(S\cap X, \mu) \le \varepsilon$.

To implement (8), we maintain two balanced binary search trees T_X and T_Y that keep track of the weight of each node and the sum of weights in each subtree. The first tree T_X maintains elements $x_i \in X \setminus T$ each assigned with weight λ_i, and the second tree T_Y maintains elements $y_i \in Y \setminus T$ with weight 1.

To produce an $e \sim \nu$, first choose tree $T_Z \in \{T_X, T_Y\}$ randomly according to the law:

$$T_Z = \begin{cases} T_X & \text{with prob.} \propto \sum_{x_i\in X\setminus T}\lambda_i \\ T_Y & \text{with prob.} \propto n - |T\cap X| \end{cases},$$

where we note that $\sum_{x_i\in X\setminus T}\lambda_i$, and $|T \cap X|$ could be obtained by a constant number of queries via the binary search trees. To sample an element $e \in T_Z$ according to the weights of each element, we may consider a binary search algorithm on T_Z that runs in $O(\log n)$ time. We initialize a variable e with the root of T_Z, and then repeat the following procedure:

1. Let L be the sum of weights in the left subtrees of e, R be the sum of weights in the right subtrees, and W be the weights of e;
2. Sample a real number $0 < x < L + R + W$ uniformly at random. If $x < W$, return e; else if $x < W + L$, update e to the left child of e; otherwise, update e to the right child of e.

Finally, by Lemma 14, the rejection sampling procedure in Algorithm 2 runs within $O(1+\lambda_{\max})$ rounds in expectation. Also note that in each round of the rejection sampling, we need $t_{\mathcal{O}_I}$ time to query the independence oracle. Together, the algorithm runs in

$$O((1+\lambda_{\max})n\log(n/\varepsilon)(\log n + t_{\mathcal{O}_I}))$$

time in expectation. ◀

4 Random cluster models with $q \le 1$

Once again, let $\mathcal{M} = (X, \mathcal{I})$ be a matroid, equipped with a rank function $\mathrm{rk}(\cdot)$. Let $\boldsymbol{\lambda} = \{\lambda_i\}_{i\in[n]}$, where $\lambda_i > 0$ is the weight or external field of $x_i \in X$. Let $0 \le q \le 1$ be a parameter. Recall the definition of the random cluster model [9] in (4). Note that when $q = 0$, the support of $\pi_{RC,q,\boldsymbol{\lambda}}$ are all subsets of full rank, namely they are the complements of the independent sets of the dual matroid.

Similar to Section 3.1, as the distribution is not homogeneous, we want to polarize it. Let Y be a set of $n = |X|$ auxiliary variables. For $T \subseteq X \cup Y$ such that $|T| = n$, the polarized distribution is given by

$$\widehat{\pi}_{RC,q,\boldsymbol{\lambda}}(T) \propto \frac{q^{-\mathrm{rk}(T\cap X)}\prod_{x_i\in T\cap X}\lambda_i}{\binom{n}{|T\cap Y|}}. \tag{9}$$

Let the right hand side of (9) be $\mathrm{wt}(T)$. Note that the marginal distribution of $\widehat{\pi}_{RC,q,\boldsymbol{\lambda}}$ on X is the same as $\pi_{RC,q,\boldsymbol{\lambda}}$.

Once homogenized, we may consider the up-down walk for $\widehat{\pi}_{RC,q,\boldsymbol{\lambda}}$. For the up step, we uniformly add an element from $(X \cup Y) \setminus T$. For the down step, suppose the current set is T such that $|T| = n+1$. We want to remove an element $e \in T$ with probability proportional to $\mathrm{wt}(T \setminus \{e\})$. Namely, the transition probability $p(e)$ satisfies

$$p(e) \propto \begin{cases} \frac{q^{-\operatorname{rk}(T\cap X)} \prod_{x_i \in T\cap X} \lambda_i}{\binom{n}{|T\cap Y|-1}} & \text{if } e \in T \cap Y; \\ \frac{q^{-\operatorname{rk}(T\cap X)} \prod_{x_i \in T\cap X} \lambda_i}{\lambda_j \binom{n}{|T\cap Y|}} & \text{if } e = x_j \in T \cap X \text{ and } \operatorname{rk}(T \cap X \setminus e) = \operatorname{rk}(T \cap X); \\ \frac{q^{-\operatorname{rk}(T\cap X)+1} \prod_{x_i \in T\cap X} \lambda_i}{\lambda_j \binom{n}{|T\cap Y|}} & \text{if } e = x_j \in T \cap X \text{ and } \operatorname{rk}(T \cap X \setminus e) = \operatorname{rk}(T \cap X) - 1. \end{cases} \tag{10}$$

We may further normalize (10) to get

$$p(e) \propto \begin{cases} \frac{n-|T\cap Y|+1}{|T\cap Y|} = \frac{|T\cap X|}{|T\cap Y|} & \text{if } e \in T \cap Y; \\ \lambda_j^{-1} & \text{if } e = x_j \in T \cap X \text{ and } \operatorname{rk}(T \cap X \setminus e) = \operatorname{rk}(T \cap X); \\ q\lambda_j^{-1} & \text{if } e = x_j \in T \cap X \text{ and } \operatorname{rk}(T \cap X \setminus e) = \operatorname{rk}(T \cap X) - 1. \end{cases} \tag{11}$$

To implement (11), we may first propose $e \sim \nu$ where

$$\nu(e) \propto \begin{cases} \frac{|T\cap X|}{|T\cap Y|} & \text{if } e \in T \cap Y; \\ \lambda_j^{-1} & \text{if } e = x_j \in T \cap X, \end{cases}$$

and then reject $e \in T \cap X$ with probability $1-q$ if $\operatorname{rk}(T \cap X \setminus \{e\}) = \operatorname{rk}(T \cap X) - 1$. Keep doing this until we accept. To see the efficiency of this implementation. Let $\mathcal{E}$ be the event that rejection happens. Then,

$$\begin{aligned} \mathbf{Pr}\left[\mathcal{E}\right] &\le \sum_{x_j \in T\cap X} \frac{(1-q)\lambda_j^{-1}}{\sum_{x_j \in T\cap X} \lambda_j^{-1} + \sum_{e \in T\cap Y} \frac{|T\cap X|}{|T\cap Y|}} \\ &= (1-q) \frac{\sum_{x_j \in T\cap X} \lambda_j^{-1}}{\sum_{x_j \in T\cap X} (\lambda_j^{-1}+1)} \le \frac{1-q}{1+\lambda_{\min}}, \end{aligned}$$

where $\lambda_{\min} := \min_{i\in[n]} \lambda_i$. Thus, in expectation, we will successfully make a transition after $O\left(\frac{\lambda_{\min}+1}{\lambda_{\min}+q}\right) = O(1+\lambda_{\min}^{-1})$ steps.

One more issue with the implementation is that we need to check the rank of $T \cap X$ and $T \cap X \setminus \{e\}$. This is why we need a rank oracle instead of the independence oracle.

The only missing ingredient to get Theorem 4 is the mixing time of the chain. To this end, we claim that the up-down walk here is just the down-up walk of the corresponding random cluster model on the dual matroid. As the homogenized generating polynomial for random cluster models with $q \le 1$ is shown to be strongly log-concave by Brändén and Huh [5], Proposition 9 implies that the mixing time of the up-down walk is $O(n \log(n/\varepsilon))$ as well.

Finally we verify the claim. Let $\overline{\mathcal{M}} = (X, \overline{\mathcal{I}})$ be the dual matroid of $\mathcal{M}$. Consider the random cluster model on $\overline{\mathcal{M}}$ with external fields $q/\boldsymbol{\lambda}$, where $\boldsymbol{\lambda} = \{\lambda_i\}$ is the external field vector of the original model:

$$\forall U \subseteq X, \quad \pi_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}(U) \propto q^{-\operatorname{rk}_{\overline{\mathcal{M}}}(U)} \prod_{x_i \in U} \left(\frac{q}{\lambda_i}\right).$$

This is equivalent to having the field $1/\boldsymbol{\lambda}$, but the extra q simplifies some calculation next.

The rank function of the dual matroid satisfies:

$$\mathrm{rk}_{\overline{\mathcal{M}}}(U) = |U| + \mathrm{rk}_{\mathcal{M}}(X \setminus U) - \mathrm{rk}_{\mathcal{M}}(X).$$

As $\mathrm{rk}_{\mathcal{M}}(X)$ is a constant, we have

$$\forall U \subseteq X, \quad \pi_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}(U) \propto q^{-\mathrm{rk}_{\mathcal{M}}(X\setminus U)} \prod_{x_i \in X\setminus U} \lambda_i.$$

Similarly, the polarized version of $\pi_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}$ satisfies

$$\forall R \in \binom{X \cup Y}{n}, \quad \widehat{\pi}_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}(R) \propto \frac{q^{-\mathrm{rk}_{\mathcal{M}}(X\setminus R)} \prod_{x_i \in X\setminus R} \lambda_i}{\binom{n}{|R\cap Y|}}.$$

Comparing the above with (9), we have that

$$\widehat{\pi}_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}(R) = \widehat{\pi}_{\mathcal{M},q,\boldsymbol{\lambda}}((X \cup Y) \setminus R). \tag{12}$$

The down-up walk (with R being the current state) on $\widehat{\pi}_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}$ is:
- (down) remove an element from R uniformly at random;
- (up) add an element $e \in (X \cup Y) \setminus R$ to R with probability $\propto \widehat{\pi}_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}(R \cup \{e\})$.

As $T = (X \cup Y) \setminus R$, we can rephrase this down-up walk as an up-down walk on the random cluster over $\mathcal{M}$:
- (up) add an element from $(X \cup Y) \setminus T$ to T uniformly at random;
- (down) remove an element $e \in T$ from T with probability $\propto \widehat{\pi}_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}(R \cup \{e\}) = \widehat{\pi}_{\mathcal{M},q,\boldsymbol{\lambda}}(T \setminus \{e\})$, where the equality is by (12).

Thus, we conclude that the down-up walk on $\widehat{\pi}_{\overline{\mathcal{M}},q,q/\boldsymbol{\lambda}}$ is the same as the up-down walk on $\widehat{\pi}_{\mathcal{M},q,\boldsymbol{\lambda}}$ modulo mapping T to $(X \cup Y) \setminus T$. This verifies the claim and finishes the proof of Theorem 4.

References

1. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials III: Mason's ultra-log-concavity conjecture for independent sets of matroids. *arXiv*, abs/1811.01600, 2018. `arXiv:1811.01600`.
2. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *STOC*, pages 1–12. ACM, 2019.
3. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, Cynthia Vinzant, and Thuy-Duong Vuong. Log-concave polynomials IV: approximate exchange, tight mixing times, and near-optimal sampling of forests. In *STOC*, pages 408–420. ACM, 2021. `arXiv:2004.07220v2`.
4. Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *FOCS*, pages 35–46. IEEE, 2018. `arXiv:1807.00929v2`.
5. Petter Brändén and June Huh. Hodge-Riemann relations for Potts model partition functions. *arXiv*, abs/1811.01696, 2018. `arXiv:1811.01696`.
6. Petter Brändén and June Huh. Lorentzian polynomials. *Ann. of Math. (2)*, 192(3):821–891, 2020. `arXiv:1902.03719v6`.
7. Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-Sobolev inequalities for strongly log-concave distributions. *Ann. Probab.*, 49(1):506–525, 2021.
8. Tomás Feder and Milena Mihail. Balanced matroids. In *STOC*, pages 26–38. ACM, 1992.
9. Cornelius M. Fortuin and Pieter W. Kasteleyn. On the random-cluster model. I. Introduction and relation to other models. *Physica*, 57:536–564, 1972.

10 Heng Guo and Kun He. Tight bounds for popping algorithms. *Random Structures Algorithms*, 57(2):371–392, 2020.

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

12 Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *J. ACM*, 66(3):18:1–18:31, 2019.

13 Leonid Gurvits. On multivariate Newton-like inequalities. In *Advances in combinatorial mathematics*, pages 61–78. Springer, Berlin, 2009.

14 Leonid Gurvits. A polynomial-time algorithm to approximate the mixed volume within a simply exponential factor. *Discrete Comput. Geom.*, 41(4):533–555, 2009.

15 Mark Jerrum. *On the complexity of evaluating multivariate polynomials*. PhD thesis, The University of Edinburgh, 1981.

16 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43(2-3):169–188, 1986.

17 Vladimir Kolmogorov. A faster approximation algorithm for the gibbs partition function. In *COLT*, volume 75, pages 228–249. PMLR, 2018.

18 Giorgos Mousa. *Local-to-Global Functional Inequalities in Simplicial Complexes*. PhD thesis, The University of Edinburgh, 2022.

19 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.

20 Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. Adaptive simulated annealing: a near-optimal connection between sampling and counting. *J. ACM*, 56(3):Art. 18, 36, 2009.

21 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

22 Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *SODA*, pages 1757–1769. SIAM, 2013.