APPROXIMATELY COUNTING BASES OF BICIRCULAR MATROIDS

HENG GUO AND MARK JERRUM

Abstract. We give a fully polynomial-time randomised approximation scheme (FPRAS) for the number of bases in a bicircular matroids. This is a natural class of matroids for which counting bases exactly is #**P**-hard and yet approximate counting can be done efficiently.

1. Introduction

This note has two aims. One is to introduce a new application of the "popping" paradigm that that has been used to design efficient perfect samplers for a number of combinatorial structures. Existing examples are cycle popping [Wil96, PW98], sink popping [CPP02] and cluster popping [GP14, GJ18] that, respectively, produce uniformly distributed spanning trees, sink-free orientations in undirected graphs, and root-connected subgraphs in directed graphs (and, as a consequence, connected subgraphs of an undirected graph). The second aim is to provide an example of a natural class of matroids for which the bases-counting problem is hard (#**P**-complete) to solve exactly, but which is polynomial time to solve approximately in the sense of Fully Polynomial-time Randomised Approximation Schemes (or FPRAS). For basic definitions connected with the complexity of counting problems refer to [MR95] or [Jer03].

Towards these aims we introduce "bicycle popping" as a means to sample, uniformly at random, bases of a bicircular matroid.[1] Bicircular matroids are associated with undirected graphs and will be defined in the next section. Note that the main result and its proof can be understood in graph-theoretic terms, and no knowledge of matroid theory is needed beyond the exchange axiom.

The computational complexity of counting bases of a matroid is still only partially understood. According to the class of matroids under consideration, the exact counting problem may be polynomial time, #**P**-complete or unresolved. Counting bases of a graphic matroid (i.e., counting spanning trees of a graph) is a classical problem and is well solved by Kirchhoff's matrix-tree theorem. This method extends fairly directly to the wider class or regular matroids [Mau76]. The bases-counting problem for bicircular matroids, a restriction of the class of transversal matroids, was shown to be #**P**-complete by Giménez and Noy [GN06]. The status of the important case of binary matroids appears to be open [Sno12].

The situation with respect to approximation algorithms is even more tantalising. If the so-called bases-exchange graph of a matroid is always a good expander, then there is a simple FPRAS that solves the problem on every reasonable class of matroids via Markov Chain Monte Carlo (MCMC). "Reasonable" here just means that there is an efficient algorithm for deciding whether a given subset of the ground set is a basis (so-called "independence oracle"). Although the bases-exchange graph is widely conjectured to be an expander (a special case of the "zero-one polytope conjecture" of Mihail and Vazirani [Mih92, Kai04]), there has not been much progress on this conjecture, as far as we are aware. However, Feder and Mihail [FM92] have shown that the class of so-called "balanced matroids", a strict superset of the class of regular matroids, is known to satisfy the expansion conjecture. (See also [JSTV04] for improvements and simplifications.) Furthermore, all paving matroids admit an FPRAS for the number of their bases, as shown by Chávez Lomelí and Welsh [CW96], through the straightforward Monte-Carlo method, since bases in these matroids are sufficiently dense.

The work described here was supported by the EPSRC research grant EP/N004221/1 "Algorithms that Count".

[1] "Bicycle popping" has the advantage of being easy to remember, but it is important to note that the term is unconnected with the concept of bicycle space of a graph.

Jerrum [Jer06] showed that it is #**P**-hard to exactly count bases of certain sparse paving matroids. Combined with the approximation algorithm of Chávez Lomelí and Welsh, this result highlights a (presumably) exponential gap between exact and approximate counting. However, it could be said that this example is not particularly natural. Piff and Welsh [PW71] demonstrated that the number of paving matroids on a ground set of n elements is doubly exponential in n, so even representing the problem instance raises significant issues.

Here we give an efficient popping-style perfect sampler for bases of a bicircular matroid, which implies an FPRAS vis standard self-reductions. Combined with the completeness result of Giménez and Noy [GN06], this provides a more convincing and natural demonstration of the gap between exact and approximate counting for matroid bases. On the other hand, it is not clear whether the same goal can be achieved via MCMC. In particular, it is not known whether bicircular matroids are balanced.

2. Bicycle-popping

For a graph $G = (V, E)$, let $n = |V|$ and $m = |E|$. When $m \geq n$ and G is connected, we associate a *bicircular* matroid $B(G)$ with G. The ground set is E, and a subset $R \subseteq E$ is independent if every connected component of (V, R) has at most one cycle. Thus, the set of bases of $B(G)$ is

$$\mathcal{B} = \{R \mid \text{every connected component of } (V, R) \text{ is unicyclic}\}.$$

In particular, if $R \in \mathcal{B}$, then $|R| = n$. Denote by $\pi_{\mathcal{B}}(\cdot)$ or simply $\pi(\cdot)$ the uniform distribution over $\mathcal{B}$. We refer the reader to [Mat77] for more details on bicircular matroids. Giménez and Noy [GN06] have shown that counting the number of bases for bicircular matroids is #**P**-complete. See also [GMN05] for extremal bounds on this number.

We now associate a random arc $a_v = (v, w)$ to each vertex $v \in V$, which is uniform over all neighbours w of v. Given an arbitrary assignment $\sigma = (a_v)_{v \in V}$, consider the directed graph (V, σ) with exactly those $|V|$ arcs. It is easy to see that each (weakly) connected component of this graph has the same number of arcs as vertices. Thus, there is exactly one (directed) cycle per connected component. Let $U(\sigma) \subseteq E$ be the subset of edges of G obtained by dropping the direction of arcs in σ. Consider the distribution $\tau(\cdot)$ on subsets of E induced by σ via the mapping $U(\sigma)$, There are two reasons why $\tau(\cdot)$ is not quite the same as $\pi(\cdot)$.

(1) It is possible to have 2-cycles in σ, in which case at least one connected component of $U(\sigma)$ will be a tree rather than a unicyclic graph.
(2) Every cycle in σ of length greater than 2 may be reversed without changing $U(\sigma)$. Thus, in $\tau(\cdot)$, each subgraph with k connected components arises in 2^k ways, skewing the distribution towards configurations with more connected components.

For each edge $e \in E$, let B_e denote the event that a 2-cycle is present at e, i.e., both orientations of e appear in σ. For each cycle C in G, we fix an arbitrary orientation and denote by B_C the event that C is oriented this way in σ. If we further condition on none of B_e or B_C happening, the resulting distribution τ induced by $U(\sigma)$ is exactly $\pi(\cdot)$.

Partial rejection sampling [GJL17] provides a useful framework to sample from a product distribution conditioned on a number of bad events not happening. In particular, we call an instance *extremal* if for any two bad events, they are either probabilistically independent or disjoint (i.e., cannot both occur). It is straightforward to verify that the collection of bad events $\{B_e \mid e \in E\} \cup \{B_C \mid C \text{ is a cycle in } G\}$ is extremal. (The reason is similar to the cycle-popping algorithm. See [GJL17, Section 4.2]. In fact, the bad events here are either identical or more restrictive than those for cycle-popping.) For an extremal instance, to draw from the desired distribution, we only need to randomly initialize all variables, and then repeatedly re-randomize variables responsible for occurring bad events. This is Algorithm 1, which we call "bicycle-popping".

We need to be a little bit careful about bad events (B_C), since there are potentially exponentially many cycles in G. We cannot afford to dictate the unfavourable orientation a priori, but rather need to figure it out as the algorithm executes. This is not difficult to get around,

Algorithm 1: Bicycle-popping

Let S be a subset of arcs obtained by assigning each arc a_v independently and uniformly among all neighbours of v;
while *a bad event* B_e *or* B_C *is present* **do**
  Let Bad be the set of vertices involved in any occurring bad event;
  Re-randomize $\{a_v \mid v \in \mathsf{Bad}\}$ to get a new S;
end
return *the undirected version of* S

since we only need an arbitrary (but deterministic) orientation of each cycle. For example, we may arbitrarily order all vertices, and give a sign $\pm$ to each direction of an edge according to the ordering. The sign of an odd-length cycle is the product over all its edges, and the sign of an even-length cycle is the product over all but the least indexed edge. Then, we can simply declare all orientations with a $+$ sign "bad".

Since the extremal condition is satisfied, applying [GJL17, Theorem 8] we get the correctness of Algorithm 1.

Proposition 1. *Conditioned on terminating, the output of Algorithm 1 is exactly* $\pi(\cdot)$.

3. Run-time analysis

An advantage of adpoting the partial rejection sampling framework is that we have a closed form formula for the expected running time of these algorithms on extremal instances. Let Ω_0 be the set of assignments so that no bad event happens, and Ω_e (or Ω_C) be the set of assignments of $(a_v)_{v\in V}$ so that exactly B_e (or B_C) happens. Then $|\Omega_0| = |\mathcal{B}|$. For a bad event B, let $\mathsf{var}(B)$ be the set of variables defining B, namely, $\mathsf{var}(B_e) = \{a_u, a_v\}$ if $e = (u, v) \in E$, and $\mathsf{var}(B_C) = \{a_v \mid v \in C\}$ if C is a cycle in G. Define

$$\Omega_E^{\mathsf{var}} := \{(\sigma, a_v) \mid \exists e \in E,\ \sigma \in \Omega_e,\ a_v \in \mathsf{var}(B_e)\},$$

and

$$\Omega_{\mathrm{cycle}}^{\mathsf{var}} := \{(\sigma, a_v) \mid \exists \text{a cycle } C,\ \sigma \in \Omega_C,\ a_v \in \mathsf{var}(B_C)\}.$$

We have the following expression for the running time of Algorithm 1. See [GJL17, Lemma 12, Theorem 13] and [GH18, Eqn. (2)].

Proposition 2. *Let* T *be the number of resampled variables of Algorithm 1. Then*

$$\mathbb{E}\, T = \frac{\left|\Omega_E^{\mathsf{var}}\right|}{|\Omega_0|} + \frac{\left|\Omega_{\mathrm{cycle}}^{\mathsf{var}}\right|}{|\Omega_0|}.$$

A related upper bound of Proposition 2 was first shown by Kolipaka and Szegedy [KS11] in the general Lovász local lemma setting.

We bound these ratios using a combinatorial encoding idea. Namely, we want to design an injective mapping from Ω_E^{var} or $\Omega_{\mathrm{cycle}}^{\mathsf{var}}$ to Ω_0. To make the mapping injective, we in fact have to record some extra information. We first deal with $\Omega_{\mathrm{cycle}}^{\mathsf{var}}$.

Lemma 3. *For a connected graph* $G = (V, E)$ *with* $m \geq n$ *where* $m = |E|$ *and* $n = |V|$, $\left|\Omega_{\mathrm{cycle}}^{\mathsf{var}}\right| \leq n\,|\Omega_0|$.

Proof. We define a "repairing" mapping $\varphi : \Omega_{\mathrm{cycle}}^{\mathsf{var}} \to \Omega_0 \times V$, as follows. For $\sigma \in \Omega_C$, we define σ_{fix} to be the same as σ except that the orientation of C is reversed. Clearly $\sigma_{\mathrm{fix}} \in \Omega_0$. Let

$$\varphi(\sigma, a_v) = (\sigma_{\mathrm{fix}}, v) \quad \text{if } \sigma \in \Omega_C \text{ and } v \in C.$$

We claim that φ is injective. To see this, given σ_{fix} and v, we simply flip the orientations of the cycle containing v to recover σ. Since φ is injective, we have that $\left|\Omega_{\mathrm{cycle}}^{\mathsf{var}}\right| \leq n\,|\Omega_0|$. □

For Ω_E^{var}, the proof is slightly more involved. For $\sigma \in \Omega_e$, if we contract e, this component is a directed tree rooted at e.

Lemma 4. *For a connected graph $G = (V, E)$ with $m \geq n$ where $m = |E|$ and $n = |V|$, $\left|\Omega_E^{\mathsf{var}}\right| \leq 2mn\,|\Omega_0|$.*

Proof. Let $\Omega_E := \bigcup_{e \in E} \Omega_e$. Then $\left|\Omega_E^{\mathsf{var}}\right| = 2\,|\Omega_E|$.

Fix an arbitrary ordering of all vertices and edges. Our goal to define an injective "repairing" mapping $\varphi : \Omega_E \to \Omega_0 \times V \times E$. For $\sigma \in \Omega_e$, find the connected component of $U(\sigma)$ containing the edge $e = (v_1, v_2)$, and let its vertex set be S. Depending on whether $S = V$, there are two cases.

(1) If $S \neq V$, then, since the graph G is connected, there must be at least one edge joining the component to the rest of the graph. Pick the first such edge (u, u') where u is in S and u' is not.
(2) Otherwise $S = V$; then since the graph has at least n edges, there must be at least one edge not in $U(\sigma)$. Let $e' = (u, u')$ be the first such edge, and C be the cycle resulting from adding e' to $U(\sigma)$. Suppose the correct orientation on C induces the orientation $u \to u'$ on e'.

Let $u = u_1, u_2, \ldots, u_\ell = v_1$ be the unique path between u and v_1 in $U(\sigma)$. (v_1 is chosen arbitrarily from the two endpoints of e.) Let σ_{fix} be the assignment so that a_{u_i} points to u_{i-1}, where $u_0 = u'$, and all other variables are unchanged from σ. It is easy to verify that $\sigma_{\text{fix}} \in \Omega_0$. Define $\varphi(\sigma) = (\sigma_{\text{fix}}, u, e)$.

We claim that φ is injective. We just need to recover σ given $(\sigma_{\text{fix}}, u, e)$. We first figure out whether $S = V$. Notice that u' can be recovered as $u \to u'$ is in σ_{fix}. If $S \neq V$, then the edge (u, u') is a bridge under σ_{fix}; whereas if $S = V$, (u, u') is not.

In the first case, simply find the path between u and v_1, and reverse the "repairing" to yield the original σ. In the second case, we remove (u, u') first, and then recover the unique path between u and v_1. The rest is the same as the first case.

Since φ is injective, we have that $\left|\Omega_E^{\mathsf{var}}\right| = 2\,|\Omega_E| \leq 2mn\,|\Omega_0|$. □

Combining Lemma 3, Lemma 4, and Proposition 2, we have the following theorem.

Theorem 5. *Let $G = (V, E)$ be a connected graph, $n = |V|$, $m = |E|$, and $m \geq n$. The expected number of random variables sampled in Algorithm 1 on G is at most $2n + 2mn$.*

4. An implementation based on a loop-erasing random walk

In the execution of Algorithm 1, during each iteration, one needs to find all bad events, and a naive implementation may take up to $O(n)$ time for this task, giving another factor on top of the bound in Theorem 5. Here we provide an implementation that has expected running time $O(mn)$, similar to the loop-erasing random walk of Wilson [Wil96]. A formal description is given in Algorithm 2.

Algorithm 2: A random walk implementation of bicycle-popping

```
V_u ← V;
S ← ∅;
while V_u ≠ ∅ do
    v ← an arbitrary vertex in V_u;
    Start a random walk from v, and erase any cycle C having length 2 or a wrong
      orientation, until some vertex in V \ V_u is reached, or a good cycle C is formed;
    Remove all vertices of the walk from V_u;
    Add all (undirected) edges along the walk to S;
end
return S
```

Observe that, in Algorithm 2, once a cycle is orientated correctly, none of its associated arcs will be resampled ever again, and the same holds for any arc attached to it. We will call such arcs "fixed". Starting from an arbitrary vertex v, we assign a random arc from v

to u, and continue this for u. So far this is just the normal random walk with memory. The difference is that whenever a cycle appears, we check whether it has length > 2 and the correct orientation. If not, then we erase it, and continue the random walk. Otherwise, we keep all random arcs leading towards this cycle, and mark them as fixed. Thus, Algorithm 2 amounts to a loop-erasing random walk with a special erasing rule.

Once the first random walk stops with a correctly oriented cycle, we do the same for the next vertex that has not been fixed yet. Now the new walk has two possible terminating conditions. Namely it is fixed if it has reached some fixed vertex, or a correctly oriented cycle of length > 2 is formed. This process is repeated until all vertices are fixed.

Algorithm 1 specifies a particular order of resampling bad events, modulo the ordering of bad events within each iteration of the while-loop. However, bad events can be sampled in any order, without affecting correctness or the expected number of resampled variables. Although the proof of this key fact has appeared in the context of specific instances of partial rejection sampling, such as cycle-popping [PW98] and sink-popping [CPP02], we are not aware that the argument has been presented in generality, so we do so presently. As a consequence of this key fact, Algorithm 2, which is sequential, has the same resulting distribution and expected number of resampled variables as Algorithm 1, which is parallel. In particular, the expected running-time of Algorithm 2 has the same order as the number of resampled variables, which is at most $O(mn)$ by Theorem 5.

The correctness of Algorithm 2 is due to the aforementioned fact that the ordering of resamplings does not matter for extremal instances. We now formalise and verify this fact. Consider a generic partial rejection sampling algorithm that repeatedly locates an occurring bad event and resamples the variables on which it depends. A specific implementation will choose a particular order for resampling the bad events. We can represent the choices made as a path in a countably infinite, directed "game graph" $\Gamma = (\Sigma, A)$. The vertex set Σ of Γ contains all multisets of bad events. We refer to these vertices as *states*. The arc set A is defined relative to a *resampling table*, as used in [GJL17], following Moser and Tardos. As the algorithm proceeds, the "frontier" in the table between used and fresh random variables advances; in the notation of [GJL17], the frontier at time t is specified by the indices $(j_{i,t} : 1 \leq i \leq n)$. At time t, the implementation will have sampled a certain multiset $M \in \Sigma$ of bad events: an event B_* that has been resampled k times will occur k times in M. Note that M determines the number of times each variable has been resampled, and hence the frontier of the table. So, even though we don't know the order in which those bad events were resampled, we do know the occurring bad events at time t. For each $M \in \Sigma$ and each possible occurring bad event B_*, we add an arc in Γ from M to $M' = M + B_*$. A state with outdegree 0 is a *terminating state*. Given a fixed resampling table, an implementation of partial rejection sampling will generate a directed path in Γ starting at the state $\emptyset$. With probability 1 (over the choice of resampling table), this path will be finite, i.e., end in a terminating state.

We now apply a Lemma of Eriksson [Eri96], which is similar in spirit to Newman's Lemma, but which is both more elementary and better suited to our needs. Observe that if two bad events occur at time t then they can be resampled in either order without altering the result; this is a consequence of the fact that the events are on disjoint sets of variables. In the terminology of [Eri96], the game graph Γ has the *polygon property*. It follows from his Theorem 2.1 that Γ has the *strong convergence property*: if there exists a path starting at $\emptyset$ and terminating at M, then every path starting at $\emptyset$ will terminate at M in the same number of steps. Since a terminating path exists with probability 1, we see that both the output and the number of resampled variables is independent of the order in which the implementation decides to resample bad events. In other words, the correctness of Algorithm 2 follows from that of Algorithm 1, and the distribution of the number of resampled variables is identical in the two algorithms.

5. Approximating the number of bases

For completeness, we include a standard self-reduction to count the number of bases of a bicircular matroid, utilising Algorithm 1.

Theorem 6. *There is an FPRAS for counting bases of a bicircular matroid, with time complexity $O(n^2m^3\varepsilon^{-2})$.*

Proof. Let $0 < \varepsilon < 1$ be a parameter expressing the desired accuracy. Also let $N(G)$ be the number of bases of $B(G)$, the bicircular matroid associated with G.

The technique for reducing approximate counting to sampling is entirely standard [Jer03, Chap. 3], but we include the details here for completeness. Fix any sequence of graphs $G = G_m, G_{m-1}, \ldots, G_{n+1}, G_n$, where each graph G_{i-1} is obtained from the previous one G_i by removing a single edge e_i, and G_n is a disjoint union of unicyclic components. (Thus the edge set of G_n is a basis of $B(G)$.) Then, noting $N(G_n) = 1$,

$$N(G)^{-1} = N(G_m)^{-1} = \frac{N(G_{m-1})}{N(G_m)} \times \frac{N(G_{m-2})}{N(G_{m-1})} \times \cdots \times \frac{N(G_{n+1})}{N(G_{n+2})} \times \frac{N(G_n)}{N(G_{n+1})}. \tag{1}$$

Let X_i be the random variable resulting from the following trial. Select, uniformly at random, a basis R from $B(G_i)$ and set

$$X_i = \begin{cases} 1, & \text{if } e_i \notin R; \\ 0, & \text{otherwise.} \end{cases}$$

Note that $\mu_i = \mathbb{E}\, X_i = N(G_{i-1})/N(G_i)$, so that

$$N(G)^{-1} = \mathbb{E}(X_m X_{m-1} \ldots X_{n+2} X_{n+1}) = \mu_m \mu_{m-1} \ldots \mu_{n+2} \mu_{n+1},$$

where the $m - n$ trials are assumed independent.

Now let $\overline{X}_i$ be obtained by taking the mean of t independent copies of the random variable X_i. Since $\mathbb{E}\,\overline{X}_i = \mu_i$, we have $N(G)^{-1} = \mathbb{E}\, Z$, where $Z = \overline{X}_m \overline{X}_{m-2} \cdots \overline{X}_{n+1}$. Also, $\operatorname{Var} \overline{X}_i = t^{-1} \operatorname{Var} X_i$, so if t is large enough the variance of Z will be small, and Z^{-1} will be a good estimate for $N(G)$. For this approach to yield a polynomial-time algorithm, we need that all the fractions appearing in the product (1) are not too small. In fact we will show that they are all bounded below by $1/2n$, which is sufficient.

For the moment, assume this claim, i.e., that $1/2n \le \mu_i \le 1$, for all $n < i \le m$. Note that $\mathbb{E}\, X_i^2 = \mathbb{E}\, X_i = \mu_i$ since X_i is a 0,1-variable. Standard manipulations give

$$\mathbb{E}\,\overline{X}_i^2 = \operatorname{Var} \overline{X}_i + (\mathbb{E}\,\overline{X}_i)^2 = t^{-1} \operatorname{Var} X_i + \mu_i^2 = t^{-1}(\mathbb{E}\, X_i^2 - \mu_i^2) + \mu_i^2 \le \mu_i^2 \left(1 + \frac{1}{t\mu_i}\right),$$

whence

$$\mathbb{E}\, Z^2 = \mathbb{E}\,\overline{X}_m^2 \ldots \mathbb{E}\,\overline{X}_{n+1}^2 \le \mu_m^2 \ldots \mu_{n+1}^2 \Big(1 + \frac{2n}{t}\Big)^m = (\mathbb{E}\, Z)^2 \Big(1 + \frac{2n}{t}\Big)^m$$

Setting $t = 40nm\varepsilon^{-2}$, we obtain $\mathbb{E}\, Z^2 = \exp(\varepsilon^2/20)(\mathbb{E}\, Z)^2 \le (1 + \varepsilon^2/16)(\mathbb{E}\, Z)^2$, which implies $\operatorname{Var} Z \le (\varepsilon^2/16)(\mathbb{E}\, Z)^2$. Thus, by Chebyshev's inequality,

$$\Pr\left[|Z - N(G)^{-1}| \le \tfrac{1}{2}\varepsilon N(G)^{-1}\right] = \Pr\left[|Z - \mathbb{E}\, Z| \le \tfrac{1}{2}\varepsilon\, \mathbb{E}\, Z\right] \ge \tfrac{3}{4}.$$

It follows that $\Pr\left[|Z^{-1} - N(G)| \le \varepsilon N(G)\right] \ge \frac{3}{4}$. In other words, the algorithm that returns the estimate Z^{-1} satisfies the conditions for an FPRAS.

To complete the proof we just need to bound the ratio $\mu_i = N(G_{i-1})/N(G_i)$. Let R be a basis of $B(G_i)$ that contains the edge e_i, i.e., that is not a basis of $B(G_{i-1})$. Let R_0 be the unique basis in $B(G_n)$ and note that $e_i \notin R_0$. Since R_0 is also a basis of $B(G_i)$, the exchange axiom for matroids asserts that there is an edge $f \in R_0 \setminus R$ such that $R + f - e_i$ is a basis of $B(G_i)$ and hence of $B(G_{i-1})$. This exchange operation associates a basis in $B(G_{i-1})$ with each basis in $B(G_i)$ that is not a basis in $B(G_{i-1})$; furthermore, every basis in $B(G_{i-1})$ arises at most $|R_0| = n$ times in this way. It follows that $N(G_i) \le (n+1)N(G_{i-1}) \le 2nN(G_{i-1})$, as required.

Overall we need $O(nm\varepsilon^{-2})$ samples for $m - n$ estimators each. For each sample, we use Algorithm 2 which has expected running time $O(nm)$ by Theorem 5, yielding the claimed time complexity. □

Acknowledgements

Part of the work was done while HG was visiting the Institute of Theoretical Computer Science, Shanghai University of Finance and Economics, and he would like to thank their hospitality.

References

[CPP02] Henry Cohn, Robin Pemantle, and James G. Propp. Generating a random sink-free orientation in quadratic time. *Electron. J. Combin.*, 9(1):10:1–10:13, 2002. 1, 5

[CW96] Laura Chávez Lomelí and Dominic Welsh. Randomised approximation of the number of bases. In *Matroid theory (Seattle, WA, 1995)*, volume 197 of *Contemp. Math.*, pages 371–376. Amer. Math. Soc., Providence, RI, 1996. 1

[Eri96] Kimmo Eriksson. Strong convergence and a game of numbers. *European J. Combin.*, 17(4):379–390, 1996. 5

[FM92] Tomás Feder and Milena Mihail. Balanced matroids. In *STOC*, pages 26–38, 1992. 1

[GH18] Heng Guo and Kun He. Tight bounds for popping algorithms. *CoRR*, abs/1807.01680, 2018. 3

[GJ18] Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. In *ICALP*, volume 107 of *LIPIcs*, pages 68:1–68:12, 2018. 1

[GJL17] Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovasz local lemma. In *STOC*, pages 342–355, 2017. 2, 3, 5

[GMN05] Omer Giménez, Anna de Mier, and Marc Noy. On the number of bases of bicircular matroids. *Ann. Combin.*, 9(1):35–45, 2005. 2

[GN06] Omer Giménez and Marc Noy. On the complexity of computing the Tutte polynomial of bicircular matroids. *Combin. Probab. Comput.*, 15(3):385–395, 2006. 1, 2

[GP14] Igor Gorodezky and Igor Pak. Generalized loop-erased random walks and approximate reachability. *Random Struct. Algorithms*, 44(2):201–223, 2014. 1

[Jer03] Mark Jerrum. *Counting, sampling and integrating: algorithms and complexity*. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, 2003. 1, 6

[Jer06] Mark Jerrum. Two remarks concerning balanced matroids. *Combinatorica*, 26(6):733–742, 2006. 2

[JSTV04] Mark Jerrum, Jung-Bae Son, Prasad Tetali, and Eric Vigoda. Elementary bounds on Poincaré and log-Sobolev constants for decomposable Markov chains. *Ann. Appl. Probab.*, 14(4):1741–1765, 2004. 1

[Kai04] Volker Kaibel. On the expansion of graphs of 0/1-polytopes. In *The sharpest cut*, MPS/SIAM Ser. Optim., pages 199–216. SIAM, 2004. 1

[KS11] Kashyap Babu Rao Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC*, pages 235–244, 2011. 3

[Mat77] Laurence R. Matthews. Bicircular matroids. *Quart. J. Math. Oxford Ser. (2)*, 28(110):213–227, 1977. 2

[Mau76] Stephen B. Maurer. Matrix generalizations of some theorems on trees, cycles and cocycles in graphs. *SIAM J. Appl. Math.*, 30(1):143–148, 1976. 1

[Mih92] Milena Mihail. On the expansion of combinatorial polytopes. In *MFCS*, volume 629 of *Lecture Notes in Computer Science*, pages 37–49, 1992. 1

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, 1995. 1

[PW71] M. J. Piff and D. J. A. Welsh. The number of combinatorial geometries. *Bull. London Math. Soc.*, 3:55–56, 1971. 2

[PW98] James G. Propp and David B. Wilson. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*, 27(2):170–217, 1998. 1, 5

[Sno12] Michael Snook. Counting bases of representable matroids. *Electron. J. Combin.*, 19(4):41:1–41:11, 2012. 1

[Wil96] David B. Wilson. Generating random spanning trees more quickly than the cover time. In *STOC*, pages 296–303, 1996. 1, 4

(Heng Guo) School of Informatics, University of Edinburgh, Informatics Forum, Edinburgh, EH8 9AB, United Kingdom.

Email address: hguo@inf.ed.ac.uk

(Mark Jerrum) School of Mathematical Sciences, Queen Mary, University of London, Mile End Road, London, E1 4NS, United Kingdom.

Email address: m.jerrum@qmul.ac.uk