SINK-FREE ORIENTATIONS: A LOCAL SAMPLER WITH APPLICATIONS

KONRAD ANAND, GRAHAM FREIFELD, HENG GUO, CHUNYANG WANG, JIAHENG WANG

Abstract. For sink-free orientations in graphs of minimum degree at least 3, we show that there is a deterministic approximate counting algorithm that runs in time $O((n^{73}/\varepsilon^{72})\log(n/\varepsilon))$, a near-linear time sampling algorithm, and a randomised approximate counting algorithm that runs in time $O((n/\varepsilon)^2\log(n/\varepsilon))$, where n denotes the number of vertices of the input graph and $0 < \varepsilon < 1$ is the desired accuracy. All three algorithms are based on a local implementation of the sink popping method (Cohn, Pemantle, Propp, 2002) under the partial rejection sampling framework (Guo, Jerrum, Liu, 2019).

1. Introduction

The significance of counting has been recognised in the theory of computing since the pioneering work of Valiant [Val79b, Val79a]. In the late 80s, a number of landmark approximate counting algorithms [JS89, DFK91, JS93] were discovered. A common ingredient of these algorithms is the computational equivalence between approximate counting and sampling for self-reducible problems [JVV86]. The reduction from counting to sampling decomposes the task into a sequence of marginal probability estimations, each of which is tractable for sampling techniques such as Markov chains. However, while only the marginal probability of one variable is in question, simulating Markov chains requires keeping track of the whole state of the instance, and thus is obviously wasteful. It is more desirable to draw samples while accessing only some local structure of the target variable. We call such algorithms local samplers.

The first such local sampler was found by Anand and Jerrum [AJ22], who showed how to efficiently generate perfect local samples for spin systems even when the underlying graph is infinite. Using local information is essential here as it is not possible to perfectly simulate the whole state. Subsequently, Feng, Guo, Wang, Wang, and Yin [FGW+23] found an alternative local sampler, namely the so-called coupling towards the past (CTTP) method, which yields local implementations of rapid mixing Markov chains. It is also observed that sufficiently efficient local samplers lead to immediate derandomisation via brute-force enumeration. Moreover, local samplers are crucial to obtain sub-quadratic time approximate counting algorithms for spin systems [AFF+25]. Thus, local samplers are highly desirable algorithms as they can lead to fast sampling, fast approximate counting, and deterministic approximate counting algorithms.

Guo, Jerrum, and Liu [GJL19] introduced partial rejection sampling (PRS) as yet another efficient sampling technique. This method generalises the cycle-popping algorithm for sampling spanning trees [Wil96] and the sink-popping algorithm for sampling sink-free orientations [CPP02]. It also has close connections with the Lovász local lemma [EL75]. For extremal instances (in the sense of [KS11]), PRS is just the celebrated Moser-Tardos algorithm for the constructive local lemma [MT10]. The most notable application of PRS is the first fully polynomial-time randomised approximation scheme (FPRAS) for all-terminal network reliability [GJ19]. On the other hand, it is still open if all-terminal reliability and counting sink-free orientations admit deterministic fully polynomial-time approximation

(Konrad Anand, Graham Freifeld, and Heng Guo) School of Informatics, University of Edinburgh, Informatics Forum, Edinburgh, EH8 9AB, United Kingdom.

(Chunyang Wang) State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China.

(Jiaheng Wang) Faculty of Informatics and Data Science, University of Regensburg, Bajuwarenstrasse 4, 93053 Regensburg, Germany.

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). Jiaheng Wang also acknowledges support from the ERC (grant agreement No. 101077083).

schemes (FPTASes). Thus, in view of the aforementioned derandomisation technique [FGW+23], a local implementation of PRS is a promising way to resolve these open problems.

In this paper, we make some positive progress for sink-free orientations (SFOs). Given an undirected graph $G = (V, E)$, a sink-free orientation of G is an orientation of edges such that each vertex has at least one outgoing edge. SFOs were first studied by Bubley and Dyer [BD97a] as a restricted case of Sat.[1] They showed that exact counting of SFOs is #**P**-complete, and thus is unlikely to have a polynomial-time algorithm. For approximate counting and sampling, in [BD97b], they showed that a natural Markov chain has an $O(m^3)$ mixing time, where m is the number of edges. Later, Cohn, Pemantle, and Propp [CPP02] introduced an exact sampler, namely the aforementioned sink-popping algorithm that runs in $O(nm)$ time in expectation, where n is the number of vertices. Using the PRS framework, Guo and He [GH20] improved the running time of sink-popping to $O(n^2)$, and constructed instances where this running time is tight. It is open whether a faster sampling algorithm or an FPTAS exists.

Our main result is a local sampler based on PRS for SFOs. Using this local sampler, for graphs of minimum degree 3, we obtain a deterministic approximate counting algorithm that runs in time $O((n^{73}/\varepsilon^{72})\log(n/\varepsilon))$, a near-linear time sampling algorithm, and a randomised approximate counting algorithm that runs in time $O((n/\varepsilon)^2\log(n/\varepsilon))$, where ε is the given accuracy. All three algorithms appear difficult to obtain using previous techniques. We will describe the results in more detail in the next section.

1.1. **Our contribution and technique overview.** Our local sampler works for a slight generalisation of SFOs, which are intermediate problems required by the standard counting to sampling reduction [JVV86]. In these problems, a subset S of vertices is specified, which are required to be sink-free, and the task is to estimate the probability of a vertex v not in S not being a sink.

Before describing our technique, let us first review the sink-popping algorithm, (which is a special case of PRS and the same as the Moser-Tardos algorithm [MT10] as the instance is extremal). We orient each edge uniformly at random. As long as there is a sink in S, we select one such vertex, arbitrarily, and rerandomise all edges incident to it, until there is no sink belong to S.

Our key observation is that it is unnecessary to simulate all edges to decide if v is a sink. In particular, if, at any point of the execution of the algorithm, v is a sink, then no adjacent edges will ever be resampled and v stays a sink till the algorithm finishes. On the other hand, if at any point v belongs to a cycle, a path leading to a cycle, or a nonempty path leading to some vertex not in S, then the orientations of all edges involved will not be resampled, and v stays a non-sink until the algorithm terminates. Thus, this observation gives us an early termination criterion for determining whether v is a sink or not. Moreover, since in the sink-popping algorithm, the order of sinks popped can be arbitrary, we can reveal the random orientation of edges strategically, and pop sinks if necessary. To be more precise, we first reveal the edges adjacent to v one by one. Once there is an outgoing edge (v, u), we then move to u and repeat this process. If any sink is revealed, we erase the orientations of all its adjacent edges and backtrack. Eventually, one of the two early termination rules above will kick in, and this gives us a local sample.

Ideally, we want our local sampler to run in $O(\log n)$ time, where n is the number of vertices. Unfortunately, the one described above does not necessarily terminate this fast. To see this, consider a sequence of degree 2 vertices, where at each step there is equal probability to move forward or backtrack. Resolving such a path of length ℓ would require $\Theta(\ell^2)$ time. On the other hand, when the minimum degree of the input graph is at least 3, the length of the path followed by the sampler forms a submartingale. The vertex v can be a sink only if this path has length 0. Thus, once the length of the path is at least $C\log n$ for some constant C, the probability of v being a sink is very small. This allows us to truncate the local sampler with only a small error.

The FPTAS for #SFO follows from the derandomisation method of [FGW+23] to the truncated local sampler. By ε-approximation, we mean an estimate $\widetilde{Z}$ such that $1-\varepsilon \le \frac{\widetilde{Z}}{Z} \le 1+\varepsilon$, where Z is the target quantity. Also, all our algorithms work for not necessarily simple graphs.

[1] As a side note, we remark that SFOs are also introduced in the context of distributed computing under the name of sinkless orientations, where they are used to give a lower bound for the distributed Lovász local lemma [BFH+16].

Theorem 1.1 (deterministic approximate counting). *For graphs with minimum degree at least* 3*, there exists a deterministic algorithm that, given* $0 < \varepsilon < 1$*, outputs an* ε*-approximation to the number of sink-free orientations with running time* $O((n^{73}/\varepsilon^{72})\log(n/\varepsilon))$*, where* n *is the number of vertices.*

Although high, the constant exponent in the running time of Theorem 1.1 is actually the most interesting feature of our algorithm. In contrast, the running time of most known FPTASes [Wei06, BG06, BGK+07, HSV18, Bar16, PR17, Moi19, FGW+23, CFG+24] has an exponent that depends on some parameter (such as the maximum degree) of the input graph. There are exceptions, for example, [LLY13, GL18], but the exponents of their running times still depend on the parameters of the problem (not of the instance).

For fast sampling, we need a slight modification of the idea above to sample orientations of edges one by one, resulting in the following approximate sampler.

Theorem 1.2 (fast sampling). *For graphs with minimum degree at least* 3*, there exists a sampling algorithm that, given* $0 < \varepsilon < 1$*, outputs a random orientation* σ *such that* σ *is* ε*-close to a uniform random sink-free orientation in total variation distance, with running time* $O\left(m \log\left(\frac{m}{\varepsilon}\right)\right)$*, where* m *is the number of edges.*

Our sampler runs in $\widetilde{O}(m)$ time[2] instead of the $O(n^2)$ time that sink-popping requires, at the cost of generating an approximate sample instead of a perfect sample. This improves over sink-popping when $m = o(n^2/\log n)$ and leads to a faster FPRAS using the counting-to-sampling reduction [JVV86]. In fact, the running time of the FPRAS can be improved further by directly invoking the truncated local sampler in the reduction.

Theorem 1.3 (fast approximate counting). *For graphs with minimum degree at least* 3*, there exists a (randomised) algorithm that, given* $0 < \varepsilon < 1$*, outputs a quantity that is an* ε*-approximation with probability at least* $3/4$ *to the number of sink-free orientations. The running time is* $O((n/\varepsilon)^2 \log(n/\varepsilon))$*, where* n *is the number of vertices.*

The success probability $3/4$ in Theorem 1.3 is standard in the definition of FPRAS, and can be easily amplified by taking the median of repeated trials and applying the Chernoff bound.

Note that directly combining Theorem 1.2 with the counting-to-sampling reduction results in an $\widetilde{O}(nm/\varepsilon^2)$ running time. Theorem 1.3 is faster when $m = \omega(n)$. Previously, the best running time for approximate counting is $\widetilde{O}(n^3/\varepsilon^2)$, via combining the $O(n^2)$ time sink-popping algorithm [GH20] with simulated annealing (see, for example, [GH20, Lemma 12]). Theorem 1.3 improves over this by roughly a factor of n. In very dense graphs (when $m = \Omega(n^2)$), Theorem 1.3 achieves near-linear time, which appears to be rare for approximate counting.

There are a plethora of fast sampling and deterministic approximate counting techniques by now. However, it appears difficult to achieve our results without the new local sampler. For example, the coupling of Bubley and Dyer [BD97b] does not seem to improve with the minimum degree requirement. On a similar note, the recent deterministic counting technique of [CFG+24] requires a distance-decreasing Markov chain coupling, whereas the Bubley-Dyer coupling is distance non-increasing. In any case, even if the technique of [CFG+24] applied, it would not imply a running time with a constant exponent. Other fast sampling and FPTAS techniques, such as spectral independence [ALO20, CLV21, CG24], correlation decay [Wei06, LLL14], and zero-freeness of polynomials [Bar16, PR17, GLLZ21], all seem difficult to apply. The main obstacle is that these techniques typically make use of properties that hold under arbitrary conditionings. However, for SFO, even if we start with a graph of minimum degree 3, conditioning the edges can result in a graph that is effectively a cycle, in which case no nice property holds. Our techniques, in contrast, require no hereditary properties and thus can benefit from the minimum degree requirement.

One much less obvious alternative approach to FPTAS is via the connection of the local lemma. In particular, because SFOs form extremal instances, their number can be computed via the independence polynomial evaluated at negative weights on the dependency graph. (We also see this fact in Section 4.1.) Normally this approach would not be efficient, because the dependency graph is usually exponentially large (for example for all-terminal reliability), but in the case of SFOs, the dependency graph is just the

[2]The $\widetilde{O}$ notation hides logarithmic factors.

input graph itself. There are more than one FPTASes [PR17, HSV18] for the independence polynomial at negative weights. However, neither appears able to recover Theorem 1.1. We provide a detailed discussion in Section 4.2. Here, let us briefly mention that the algorithm by Patel and Regts [PR17] requires a uniform bound on the weights, which cannot handle the mixture of high and low degree vertices. On the other hand, with the minimum degree ≥ 3 assumption, the probability vector for SFOs is within the so-called Shearer's region [She85]. This makes the algorithm by Harvey, Srivastava, and Vondrák [HSV18] more relevant. However, the running time of their algorithm has the form $(n/\varepsilon)^{O(\log d)}$,[3] where d is the maximum degree of the graph. Thus, in the setting of Theorem 1.1, their algorithm runs in quasi-polynomial time instead.

The rest of the paper is organised as follows. In Section 2, we introduce our local sampler. It is then analysed in Section 3. The main theorems are shown in Section 4. We conclude with a few open problems in Section 5.

2. A local sampler for sink-free orientations

Fix $G = (V, E)$ as an undirected graph. An orientation σ of G is an assignment of a direction to each edge, turning the initial graph into a directed graph. For any $S \subseteq V$, let Ω_S be the set of S-sink-free orientations of G, i.e., the set of orientations such that each vertex $v \in S$ is not a sink. Thus, Ω_V is the set of all (normal) sink-free orientations of G. When $|\Omega_S| \neq 0$, we use μ_S to denote the uniform distribution over Ω_S. For two adjacent vertices $u, v \in V$, we use $\{u, v\}$ to denote the undirected edge and (u, v) to denote the directed edge, from u to v.

We apply the following standard counting-to-sampling reduction [JVV86]. Let $V = \{v_1, v_2, \ldots, v_n\}$ be arbitrarily ordered and, for each $0 \leq i \leq n$, define $V_i = \{v_1, v_2, \ldots, v_i\}$. Then, $|\Omega_V|$ can be decomposed into a telescopic product of marginal probabilities:

$$|\Omega_V| = \left|\Omega_{V_0}\right| \cdot \prod_{i=1}^{n} \frac{\left|\Omega_{V_i}\right|}{\left|\Omega_{V_{i-1}}\right|} = 2^{|E|} \cdot \prod_{i=1}^{n} \mu_{V_{i-1}}(v_i \text{ is not a sink}). \tag{1}$$

Thus, our goal becomes to estimate $\mu_S(v \text{ is not a sink})$ for any $S \subseteq V$ and $v \notin S$.

We view S-sink-free orientations under the variable framework of the Lovász local lemma. Here, each edge corresponds to a variable that indicates its direction, and each vertex in S represents a bad event of being a sink. An instance is called *extremal* if any two bad events are independent (namely, they share no common variable) or disjoint. It is easy to see that all instances to the S-sink-free orientation problem are extremal: if a vertex is a sink then none of its neighbors can be a sink. For extremal instances like this, the celebrated Moser-Tardos algorithm [MT10] is guaranteed to output an assignment avoiding all bad events uniformly at random [GJL19]. This is summarised in Algorithm 1. Note that when $S = V$, Algorithm 1 is the sink-popping algorithm by Cohn, Pemantle, and Propp [CPP02].

Algorithm 1: PRS algorithm for generating an S-sink-free orientation

Input : an undirected graph $G = (V, E)$ and a subset of vertices $S \subseteq V$
Output : an orientation σ of G

orient each edge $e \in E$ uniformly at random and independently to obtain an orientation σ;
while $\exists v \in S$ *s.t.* v *is a sink in* σ **do**
    choose such a v arbitrarily;
    resample the orientation of all edges incident to v in σ uniformly at random;
return σ;

The following lemma is a direct corollary from [GJL19, Theorem 8] and SFOs being extremal.

Lemma 2.1. *If* $|\Omega_S| \neq 0$, *Algorithm 1 terminates almost surely and returns an orientation distributed exactly as* μ_S.

[3] To be more precise, the hidden constant in the exponent is linear in the inverse of the multiplicative "slack" of how close the evaluated point is to the boundary of Shearer's region. For SFOs, when constant degree vertices are present, the slack is a constant.

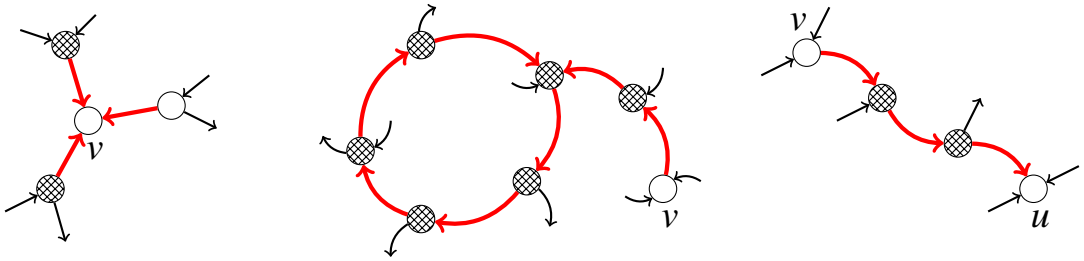

FIGURE 1. Illustration of Lemma 2.2. Shaded vertices are in the set S. Once these patterns are formed, thick red edges would never be resampled in Algorithm 1.

We remark that the only possible case for $\Omega_S = \emptyset$ is when S forms a tree and not connected to any vertex not in S.

Algorithm 1 requires one to generate a global sample when estimating $\mu_S(v \text{ is not a sink})$ for some $v \notin S$, which is wasteful. The following observation is crucial to turning it into a local sampler.

Lemma 2.2 (criteria for early termination)**.** *Suppose $|\Omega_S| \neq 0$. For any $v \notin S$, v is a sink upon the termination of Algorithm 1 if and only if*

(a) v becomes a sink at some iteration.

Conversely, v is not a sink upon the termination of Algorithm 1 if and only if one of the following holds:

(b1) a directed cycle C containing v, or a directed path P containing v which ends in a directed cycle C is formed in some iteration, or

(b2) a nonempty directed path P from v to some $u \notin S$ is formed in some iteration.

Proof. First, consider *(a)*. If v becomes a sink at any point, then for every $w \in S$ which is a neighbour of v, the edge (w, v) is oriented towards v. Since $v \notin S$, the edge (w, v) will not be resampled via v, and could only be resampled by w becoming a sink. Since w cannot become a sink without resampling (w, v), v will remain a sink. The other implication is obvious.

Now for *(b1)* and *(b2)*, we first consider the forward implications. For a cycle C, every vertex $u \in C$ has an edge pointing outwards towards some $w \in C$ which also has an edge pointing outwards. None of these edges can be resampled without another edge $e \in C$ being resampled first, so no vertex in the cycle will ever become a sink again.

Consider a path P that ends outside of S or in a cycle. Inductively, we see that no edge on this path will be resampled without the edge after it being resampled. The edge connected to the cycle or S^c will not be resampled, since that vertex may never be a sink, so no vertex in P can become a sink.

For the reverse implication, suppose v is eventually not a sink. In that case, there must be some edge (v, w) pointing towards a neighbour w. If w is a sink, then $w \notin S$ and we are in case *(b2)*. Otherwise, w is not a sink, and there is an adjacent edge pointing away from w, and we repeat this process. As the set of vertices is finite, if vertices considered in this process are all in S, then there must be repeated vertices eventually, in which case we are in *(b1)*. □

An illustration of Lemma 2.2 is given in Figure 1. Based on Lemma 2.2, we design a local sampling algorithm for determining whether some vertex $v \notin S$ is a sink or not, given in Algorithm 2. We assume that the undirected graph $G = (V, E)$ is stored as an adjacency list where the neighbors of each vertex are arbitrarily ordered. Algorithm 2 takes as input some $S \subseteq V$ and $v \notin S$, returns an indicator variable $x \in \{0, 1\}$ such that $\mathbf{Pr}[x = 1] = \mu_S(v \text{ is not a sink})$. We treat the path P as a subgraph, and $V(P)$ denotes the vertex set of P. When we remove the last vertex from P, we remove it and the adjacent edge as well. Informally, Algorithm 2 starts from the vertex v, and reveal adjacent edges one by one. If there is any edge pointing outward, say (v, u), we move to u and reveal edges adjacent to u. If a sink $w \in S$ is formed, then we mark all adjacent edges of w as unvisited and backtrack. This induces a directed path starting from v. We repeat this process until any of the early termination criteria of Lemma 2.2 is satisfied, in which case we halt and output accordingly.

Algorithm 2: Sample(G, S, v)

Input : an undirected graph $G = (V, E)$, a subset of vertices $S \subseteq V$, and a vertex $v \notin S$;
Output : a random value $x \in \{0, 1\}$;

```
1  Let P be a (directed) path initialised as P = (v);
2  Initialise a mapping M : E → {visited, unvisited} so that ∀e ∈ E, M(e) = unvisited;
3  while |V(P)| ≥ 1 do
4      Let u be the last vertex of P;
5      if |V(P)| ≥ 2 and u ∉ S then return 1;
6      if all edges incident to u are marked visited then
7          mark all edges incident to u as unvisited;
8          remove u from P;
9      else
10         let e = {u, w} be the first unvisited edge incident to u;
11         mark e as visited;
12         with probability 1/2 do
13             if w ∈ V(P), or there is a visited edge (w, w') for some w' ∈ V(P) then return 1;
14             append w to the end of P;
15 return 0;
```

3. Analysis of the local sampler

In this section, we analyse the correctness and efficiency of Algorithm 2.

Lemma 3.1 (correctness of Algorithm 2)**.** *Let $G = (V, E)$ be a graph. For any $S \subseteq V$ such that $|\Omega_S| \neq 0$ and any $v \notin S$,* Sample(G, S, v) *terminates almost surely and upon termination, returns an $x \in \{0, 1\}$ such that*

$$\mathbf{Pr}\left[x = 1\right] = \mu_S(v \text{ is not a sink}).$$

Proof. We claim that there exists a coupling between the execution of Algorithm 1 (with input G, S and output σ), and Sample(G, S, v) (with output x), such that

$$v \text{ is not a sink under } \sigma \iff x = 1. \tag{2}$$

The claim implies the lemma because of Lemma 2.1.

To prove the claim, we first construct our coupling. We use the resampling table idea of Moser and Tardos [MT10]. For each edge, we associate it with an infinite sequence of independent random variables, each of which is a uniform orientation. This forms the "resampling table". Our coupling uses the same resampling table for both Algorithm 1 and Algorithm 2. As showed in [MT10], the execution of Algorithm 1 can be viewed as first constructing this (imaginary) table, and whenever the orientation of an edge is randomised, we just reveal and use the next random variable in the sequence. For Algorithm 2, we reveal the orientation of an edge when its status changes from unvisited to visited in Line 11. We execute Line 14 if the random orientation from the resampling table is (u, w), and otherwise do nothing. Namely, in the latter case, the revealed orientation is (w, u), and we just move forward the "frontier" of that edge by one step in the resampling table. We claim that (2) holds with this coupling.

Essentially, the claim holds since for extremal instances, given a resampling table, the order of the bad events resampled in Algorithm 1 does not affect the output. This fact is shown in [GJ21, Section 4]. (See also [CPP02, Lemma 2.2] for the case of $S = V$.) We can "complete" Algorithm 2 after it finishes. Namely, once Algorithm 2 terminates, we randomise all edges that are not yet oriented, and resample edges adjacent to sinks until there are none, using the same resampling table. Note that at the end of Algorithm 2, some edges may be marked unvisited but are still oriented. Suppose that the output of the completed algorithm is σ', an orientation of all edges. This completed algorithm is just another implementation of Algorithm 1 with a specific order of resampling bad events. Thus the fact mentioned earlier implies that $\sigma' = \sigma$.

On the other hand, the termination conditions of Algorithm 2 correspond to the cases of Lemma 2.2. One can show, via a simple induction over the while loop, that at the beginning of the loop, the path P is always a directed path starting from v, and all other visited edges point towards the path P. This implies that Line 13 corresponds to case *(b1)* in Lemma 2.2, Line 5 corresponds to case *(b2)*, and exiting the while loop in Line 3 corresponds to case *(a)*. When Algorithm 2 terminates, we have decided whether or not v is a sink. By Lemma 2.2, this decision stays the same under σ'. As $\sigma' = \sigma$, (2) holds. □

We then analyse the efficiency of Algorithm 2. The main bottleneck is when there are degree 2 vertices. It would take $\Omega(\ell^2)$ time to resolve an induced path of length ℓ. We then focus on the case where the minimum vertex degree is at least 3. Note that in this case we have $\Omega_S \neq \emptyset$ for any $S \subseteq V$. For two distributions P and Q over the state space Ω, their total variation distance is defined by $d_{\mathrm{TV}}(P, Q) \coloneqq \sum_{x \in \Omega} |P(x) - Q(x)|/2$. For two random variables $x \sim P$ and $y \sim Q$, we also write $d_{\mathrm{TV}}(x, y)$ to denote $d_{\mathrm{TV}}(P, Q)$. Then we have the following lemma.

Lemma 3.2 (efficient truncation of Algorithm 2)**.** *Let $G = (V, E)$ be a graph with minimum degree at least 3. Let $S \subseteq V$, $v \notin S$ and $0 < \varepsilon < 1$. Let x be the output of* Sample(G, S, v)*, and x' constructed as*

- *if* Sample(G, S, v) *terminates within $72 \ln(73/\varepsilon)$ executions of Line 12, let $x' = x$;*
- *otherwise, let $x' = 1$.*

Then, it holds that

$$d_{\mathrm{TV}}(x, x') \leq \varepsilon.$$

Proof. We track the length of the path P during the execution of Algorithm 2. When an edge is chosen in Line 4 and sampled in Line 12 of Algorithm 2, the following happens:

- with probability $1/2$, w is appended to P and the length of P increases by one;
- with probability $1/2$, $\{u, w\}$ is marked as visited, and the length of P decreases by one in the next iteration if and only if $\{u, w\}$ was the last unvisited edge incident to u.

Let X_i be the random variable denoting the length of P after the i-th execution of Line 12 in Algorithm 2. Then the observation above implies that $\{X_i\}_{i \geq 0}$ forms a submartingale. We construct another sequence of random variables $\{Y_i\}_{i \geq 0}$ modified from $\{X_i\}_{i \geq 0}$ as follows:

- $Y_0 = X_0 = 1$.
- At the i-th execution of Line 12 in Algorithm 2:
  - if $\{u, w\}$ is the only unvisited edge incident to u, set $Y_{i+1} = X_{i+1} - X_i + Y_i$,
  - otherwise, set $Y_{i+1} = X_{i+1} - X_i + Y_i - 1/2$.

It can be verified that the sequence $\{Y_i\}_{i \geq 0}$ is a martingale.

Claim 3.3. *For any $i \geq 0$, $X_i - Y_i \geq i/4$.*

Proof. Note that $X_i - Y_i = X_{i-1} - Y_{i-1} + c_i$ where $c_i = 0$ if $\{u, w\}$ is the only unvisited edge incident to u at the i-th execution of Line 12 in Algorithm 2, and $c_i = 1/2$ otherwise. Then we can write $X_i - Y_i$ as

$$X_i - Y_i = X_0 - Y_0 + \sum_{j=1}^{i} c_j.$$

For any i such that $c_i = 0$, let i' be the last index such that $c_{i'} = 0$, or $i' = 0$ if no such i' exists. Since the minimum degree of G is at least 3, when we append any vertex u to P, there are at least two unvisited edges incident to u. It implies that there must be some j such that $i' < j < i$ and $c_j = 1/2$. Thus $X_i - Y_i = \sum_{k=1}^{i} c_k \geq i/4$. □

Next we show that if Algorithm 2 doesn't terminate after $72 \ln(73/\varepsilon)$ steps, with high probability the length of the path will not return to 0. As $\{Y_i\}_{i \geq 0}$ is a martingale and $|Y_{i+1} - Y_i| \leq 3/2$ for all $i \geq 0$, the Azuma–Hoeffding inequality implies that, for any $T > 0$ and $C > 0$,

$$\mathbf{Pr}\left[Y_T - Y_0 \leq -C\right] \leq \exp\left(\frac{C^2}{9T/2}\right). \tag{3}$$

Thus,

$$\mathbf{Pr}\left[X_T = 0\right] \leq \mathbf{Pr}\left[Y_T \leq -T/4\right] \leq \mathbf{Pr}\left[Y_T - Y_0 \leq -T/4\right] \leq \exp\left(-T/72\right),$$

where the first inequality is by Claim 3.3, and the last inequality is by plugging $C = T/4$ into (3). Then, we have

(4)
$$\sum_{T=\lceil 72\ln\frac{73}{\varepsilon}\rceil}^{\infty} \mathbf{Pr}\left[X_T = 0\right] \le \sum_{T=\lceil 72\ln\frac{73}{\varepsilon}\rceil}^{\infty} \exp\left(-T/72\right) \le \sum_{T=72\ln\frac{73}{\varepsilon}}^{\infty} \exp\left(-T/72\right) = \frac{\varepsilon}{73(1-\mathrm{e}^{-1/72})} < \varepsilon.$$

To finish the proof, we couple x and x' by the same execution of Algorithm 2. Thus, if it terminates within $72\ln(73/\varepsilon)$ executions of Line 12, then $x = x'$ with probability 1. If not, (4) implies that $x = 0$ with probability at most ε. As we always output $x = 1$ in this case, $x' \neq x$ with probability at most ε, which finishes the proof. □

Note that Lemma 3.2 does not require $\Omega_S \neq \emptyset$. This is because it is implied by the minimum degree requirement. This implication is an easy consequence of the symmetric Shearer's bound. It is also directly implied by Lemma 4.1 which we show next.

4. Applications of the local sampler

We show the main theorems in this section. Lemma 3.2 implies an additive error on the truncated estimator. As we are after relative errors in approximate counting, we need a lower bound of the marginal ratio.

Lemma 4.1. *Let $G = (V, E)$ be a graph with a minimum degree at least 3. For any $S \subseteq V$ and $v \notin S$, it holds that $|\Omega_S| \neq 0$ and*

$$\mu_S(v \text{ is not a sink}) > \frac{1}{2}.$$

The proof of Lemma 4.1 can be viewed as an application of the symmetric Shearer's Lemma [She85] on SFO, and is deferred to Section 4.1. Note that the minimum degree requirement is essential for such a marginal lower bound to hold, as the marginal ratio in Lemma 4.1 can be of order $O(1/n)$ when G is a cycle and $S = V \setminus \{v\}$.

We then show the two approximate counting algorithms first, namely Theorem 1.1 and Theorem 1.3.

Proof of Theorem 1.1. By (1), we just need to $\varepsilon/(2n)$-approximate $\mu_{V_{i-1}}(v_i \text{ is not a sink})$ for each i to ε-approximate $|\Omega_V|$, the number of sink-free orientations to G. The only random choice Algorithm 2 makes is Line 12. In view of Lemma 3.2, we enumerate the first $72\ln(292n/\varepsilon)$ random choices of choices Algorithm 2, and just output 1 if the algorithm does not terminate by then. Let the estimator be the average of all enumeration. Note that Lemma 4.1 implies that $\Omega_{V_i} \neq \emptyset$ for any i. Then, Lemmas 3.1 and 3.2 imply that the estimator is an $\varepsilon/(4n)$ additive approximation. By Lemma 4.1, it is also an $\varepsilon/(2n)$ relative approximation, which is what we need.

For the running time, there are n marginals, it takes $\exp(72\ln(292n/\varepsilon))$ enumerations for each marginal probability, and each enumeration takes time at most $O(\ln(292n/\varepsilon))$ time. Therefore, the overall running time is bounded by $O(n(n/\varepsilon)^{72}\log(n/\varepsilon))$. □

Proof of Theorem 1.3. We use (1) again. Denote $\nu_i = \mu_{V_{i-1}}(v_i \text{ is not a sink})$ and $\nu = \prod_{i=1}^{n}\nu_i$. Let $\widetilde{X}_i := \frac{1}{n}\sum_{i=1}^{n} x'_{i,t}$ be the average of n independent samples from Algorithm 2 truncated after $72\ln(73\times 12n/\varepsilon)$ executions of Line 12. Let $\widetilde{X} := \prod_{i=1}^{n}\widetilde{X}_i$ be an estimator for ν.

For any i and t, let $\widetilde{\nu}_i$ be the expectation of $x_{i,t}$ (note that it does not depend on t). By Lemmas 3.1 and 3.2, $|\widetilde{\nu}_i - \nu_i| \le \frac{\varepsilon}{12n}$. By Lemma 4.1, $1 - \frac{\varepsilon}{6n} \le \frac{\widetilde{\nu}_i}{\nu_i} \le 1 + \frac{\varepsilon}{6n}$. Let $\widetilde{\nu} = \prod_{i=1}^{n}\widetilde{\nu}_i$ so that $\mathbf{E}\left[\widetilde{X}\right] = \widetilde{\nu}$. Then, as $0 < \varepsilon < 1$,

(5)
$$1 - \frac{\varepsilon}{3} \le \frac{\widetilde{\nu}}{\nu} \le 1 + \frac{\varepsilon}{3}.$$

We bound $\mathbf{Var}\left[\widetilde{X}_i\right]$ and $\mathbf{Var}\left[\widetilde{X}\right]$ next. First,

$$\mathbf{Var}\left[\widetilde{X}_i\right] = \mathbf{Var}\left[\frac{1}{n}\sum_{t=1}^{n} x'_{i,t}\right] = \frac{1}{n^2}\sum_{t=1}^{n}\mathbf{Var}\left[x'_{i,t}\right] \le \frac{1}{n},$$

as each $x'_{i,t}$ is an indicator variable. Then,

$$\frac{\mathbf{Var}\left[\widetilde{X}\right]}{\left(\mathbf{E}\left[\widetilde{X}\right]\right)^2} = \frac{\mathbf{E}\left[\widetilde{X}^2\right]}{\left(\mathbf{E}\left[\widetilde{X}\right]\right)^2} - 1 = \frac{\prod_{i=1}^n \mathbf{E}\left[\widetilde{X}_i^2\right]}{\prod_{i=1}^n \left(\mathbf{E}\left[\widetilde{X}_i\right]\right)^2} - 1 = \prod_{i=1}^n \left(1 + \frac{\mathbf{Var}\left[\widetilde{X}_i\right]}{\left(\mathbf{E}\left[\widetilde{X}_i\right]\right)^2}\right) - 1$$

(by Lemma 4.1) $$\leq \left(1 + \frac{4}{n}\right)^n - 1 < \mathrm{e}^4 - 1 < 54.$$

To further reduce the variance, let $\widehat{X}$ be the average of N independent samples of $\widetilde{X}$, where $N := \lceil 36 \times 54/\varepsilon^2 \rceil$. Then, $\mathbf{Var}[\widehat{X}] \leq \frac{\mathbf{Var}[\widetilde{X}]}{N}$. By Chebyshev's bound, we have

$$\mathbf{Pr}\left[\left|\widehat{X} - \widetilde{\nu}\right| \geq \frac{\varepsilon}{3} \cdot \widetilde{\nu}\right] \leq \frac{9\mathbf{Var}[\widehat{X}]}{\varepsilon^2 \widetilde{\nu}^2} \leq \frac{9 \times 54 \varepsilon^2 \widetilde{\nu}^2}{36 \times 54} \cdot \frac{1}{\varepsilon^2 \widetilde{\nu}^2} \leq \frac{1}{4}.$$

Thus with probability at least $\frac{3}{4}$, we have that $(1 - \frac{\varepsilon}{3})\widetilde{\nu} \leq \widehat{X} \leq (1 + \frac{\varepsilon}{3})\widetilde{\nu}$. By (5), when this holds, $(1 - \varepsilon)\nu \leq \widehat{X} \leq (1 + \varepsilon)\nu$. It is then easy to have an ε-approximation of $|\Omega_V|$.

For the running time, each sample $x'_{i,t}$ takes $O(\log(n/\varepsilon))$ time. We draw n samples for each of the n vertices, and we repeat this process $N = O(\varepsilon^{-2})$ times. Thus, the total running time is bounded by $O((n/\varepsilon)^2 \log(n/\varepsilon))$. □

For Theorem 1.2, we will need a modified version of Algorithm 2 to sample from the marginal distributions of the orientation of edges. This is given in Algorithm 3. It takes as input a subset of vertices $S \subseteq V$ and an edge $e \in E$, then outputs a random orientation σ_e following the marginal distribution induced from μ_S on e. The differences between Algorithm 2 and Algorithm 3 are:

- In Algorithm 2, the number of vertices in P is initialised as $|V(P)| = 1$, while in Algorithm 3, it is initialised as $|V(P)| = 2$.
- When $|V(P)| = 1$ and all edges incident to the only vertex u in P are marked as visited:
  - In Algorithm 2, the algorithm terminates and returns 0;
  - In Algorithm 3, the algorithm terminates if and only if $u \notin S$, and would reinitialise the algorithm otherwise.

The correctness of Algorithm 3 is due to a coupling argument similar to Lemma 3.1. We couple Algorithm 1 and Algorithm 3 by using the same resampling table. By the same argument as in Lemma 3.1, given the same resampling table, the orientation of e is the same in the outputs of both Algorithm 1 and Algorithm 3. Thus, σ_e follows the desired marginal distribution by Lemma 2.1. As for efficiency, we notice that the same martingale argument as in Lemma 3.2 applies to the length of P as well. Early truncation of the edge sampler only incurs a small error. However, we need some extra care for the self-reduction in the overall sampling algorithm.

Proof of Theorem 1.2. We sequentially sample the orientation of edges in G (approximately) from its conditional marginal distribution. Suppose we choose an edge $e = \{u, v\}$, and the sampled orientation is (u, v). Then, we can remove e from the graph, and let $S \leftarrow S \setminus \{u\}$. The conditional distribution is effectively the same as μ_S in the remaining graph.

One subtlety here is that doing so may create vertices of degree ≤ 2. To cope with this, we keep sampling edges adjacent to one vertex in S as much as possible before moving on to the next. Suppose the current focus is on v. We use SampleEdge to sample the orientation of edges adjacent to v one at a time until either v is removed from S or the degree of v becomes 1. In the latter case, the leftover edge must be oriented away from v, which also results in removing v from S. Note that, when either condition holds, the last edge sampled is oriented as (v, u) for some neighbour u of v. We then move our focus to u if $u \in S$, and move to an arbitrary vertex in S otherwise. The key property of choosing edges this way is that, whenever SampleEdge(G, S, e) is invoked, there can only be at most one vertex of degree 2 in S, and if it exists, it must be an endpoint of e. If all vertices are removed from S, we finish by simply outputing a uniformly at random orientation of the remaining edges.

To maintain efficiency, we truncate SampleEdge(G, S, e) in each step of the sampling process. More specifically, for some constant C, we output the first edge of P once the number of executions of

Algorithm 3: SampleEdge(G, S, e)

Input : an undirected graph $G = (V, E)$, a subset of vertices $S \subseteq V$, and an edge $e = \{x, y\} \in E$;
Output : a random orientation $\sigma_e \in \{(x, y), (y, x)\}$ of e;

```
1  Initialise a mapping M : E → {visited, unvisited} so that ∀e ∈ E, M(e) = unvisited;
2  Let P be a (directed) path initialised as P = (x, y) or P = (y, x) with equal probability, and mark
     e visited;
3  while True do
4      Let u be the last vertex of P;
5      if |V(P)| ≥ 2 and u ∉ S then return the first edge in P;
6      if all edges incident to u are marked visited then
7          mark all edges incident to u as unvisited;
8          if |V(P)| = 1 then
9              rerandomise P as P = (x, y) or P = (y, x) with equal probability;
10         else
11             remove u from P;
12     else
13         let e = {u, w} be the first unvisited edge incident to u;
14         mark e as visited;
15         with probability 1/2 do
16             if w ∈ V(P), or there is a visited edge (w, w') for some w' ∈ V(P) then
17                 return the first edge in P;
18             append w to the end of P;
```

Line 15 in Algorithm 3 exceeds $C \ln(m/\varepsilon)$. We claim that there is a constant C such that the truncation only incurs an ε/m error in total variation distance between the output and the marginal distribution. This is because the same martingale argument as in Lemma 3.2 still applies. Note that if P visits any vertex not in S, the algorithm immediately terminates. Thus degrees of vertices not in S do not affect the argument. Moreover, the only degree 2 vertex in S, say x, is adjacent to the first edge $e = \{x, y\}$ of P. If e is initialised as $\{x, y\}$, then when P returns to x, the algorithm immediately terminates. Otherwise e is initialised as $\{y, x\}$, in which case there is no drift in the first step of P. Thus, as long as we adjust the constant to compensate the potential lack of drift in the first step, the martingale argument in Lemma 3.2 still works and the claim holds. As the truncation error is ε/m, we may couple the untruncated algorithm with the truncated version, and a union bound implies that the overall error is at most ε.

As we process each edge in at most $O(\log(m/\varepsilon))$ time, the overall running time is then $O(m \log(m/\varepsilon))$. This finishes the proof of the fast sampling algorithm. □

4.1. **Proof of the marginal lower bound.** Now we prove the lower bound of marginal ratios for SFOs, namely, Lemma 4.1. Let us first recall the variable framework of the local lemma. Consider the probability space $\mathcal{P}$ of a uniformly random orientation of G (namely orienting each edge independently and uniformly at random), and each $u \in S$ corresponding to a bad event $\mathcal{E}_u$ of u being a sink. We then have

$$p_u := \Pr_{\mathcal{P}}[\mathcal{E}_u] = 2^{-d(u)}, \quad \forall u \in V,$$

where $d(u)$ denotes the degree of u. We also need some definitions, essentially from [HV17] and small variations from those in [She85].

Definition 4.2. We define the following notations.

- Let $\mathsf{Ind}(G)$ denote all independent sets of G, i.e.,
$$\mathsf{Ind}(G) \coloneqq \{I \subseteq V \mid I \text{ contains no edge of } G\}.$$
- For $J \subseteq V$, let
$$q_J \coloneqq \sum_{\substack{I \in \mathsf{Ind}(G) \\ I \subseteq J}} (-1)^{|I|} \prod_{u \in I} p_u, \tag{6}$$
and
$$P_J \coloneqq \Pr_{\mathcal{P}}\left[\bigwedge_{u \in J} \neg \mathcal{E}_u\right],$$
which is the probability under $\mathcal{P}$ that all vertices in J are sink-free.

We then proceed to the proof.

Proof of Lemma 4.1. For $u \in V$, let $\Gamma(u)$ denote the set of neighbours of u in G. We claim that for any $J \subseteq V$ and $u \in J$:

(1) $P_J = q_J > 0$;

(2) $\frac{q_J}{q_{J\setminus\{u\}}} > \begin{cases} \frac{1}{2} & \Gamma(u) \subseteq J; \\ \frac{2}{3} & \text{otherwise.} \end{cases}$

Lemma 4.1 immediately follows from the claim as $P_S = \frac{|\Omega_S|}{2^{|E|}} > 0$ and

$$\mu_S(v \text{ is not a sink}) = \Pr_{\mathcal{P}}\left[\neg\mathcal{E}_v \mid \bigwedge_{u \in S} \neg\mathcal{E}_u\right] = \frac{P_{S\cup\{v\}}}{P_S} = \frac{q_{S\cup\{v\}}}{q_S} > \frac{1}{2},$$

where the last equality is by Item 1 and the inequality is by Item 2.

We then prove claim by induction on the size of J. The base case is when $J = \emptyset$, and all items directly hold as $P_\emptyset = q_\emptyset = 1$. For the induction step, we first prove Item 1. For $J \subseteq V$ and $u \in J$, denote $\Gamma^+(u) = \Gamma(u) \cup \{u\}$. We have

$$\begin{aligned}\Pr_{\mathcal{P}}\left[\mathcal{E}_u \wedge \left(\bigwedge_{j \in J\setminus\{u\}} \neg\mathcal{E}_j\right)\right] &= \Pr_{\mathcal{P}}[\mathcal{E}_u] \Pr_{\mathcal{P}}\left[\left(\bigwedge_{j \in J\setminus\{u\}} \neg\mathcal{E}_j\right) \mid \mathcal{E}_u\right] = p_u \Pr_{\mathcal{P}}\left[\left(\bigwedge_{j \in J\setminus\Gamma^+(u)} \neg\mathcal{E}_j\right) \mid \mathcal{E}_u\right] \\ &= p_u \Pr_{\mathcal{P}}\left[\left(\bigwedge_{j \in J\setminus\Gamma^+(u)} \neg\mathcal{E}_j\right)\right] = p_u \cdot P_{J\setminus\Gamma^+(u)},\end{aligned}$$

where in the second equality we used the fact that S-SFO instances are extremal. Thus,

$$P_J = P_{J\setminus\{u\}} - \Pr_{\mathcal{P}}\left[\mathcal{E}_u \wedge \left(\bigwedge_{j \in J\setminus\{u\}} \neg\mathcal{E}_j\right)\right] = P_{J\setminus\{u\}} - p_u \cdot P_{J\setminus\Gamma^+(u)}. \tag{7}$$

Also, by separating independent sets according to whether they contain u or not, we have

$$\begin{aligned} q_J &= \sum_{\substack{I \in \mathsf{Ind}(G) \\ I \subseteq J}} (-1)^{|I|} \prod_{i \in I} p_i \\ &= \sum_{\substack{I \in \mathsf{Ind}(G) \\ I \subseteq J\setminus\{u\}}} (-1)^{|I|} \prod_{i \in I} p_i - p_u \cdot \sum_{\substack{I \in \mathsf{Ind}(G) \\ I \subseteq J\setminus\Gamma^+(u)}} (-1)^{|I|} \prod_{i \in I} p_i \\ &= q_{J\setminus\{u\}} - p_u \cdot q_{J\setminus\Gamma^+(u)}. \end{aligned} \tag{8}$$

Combining (7), (8), and the induction hypothesis, we have that $P_J = q_J$. For the positivity, let $J \cap \Gamma^+(u)$ be listed as $\{u, u_1, \ldots, u_k\}$ for some $k \le d(u)$. For $0 \le i \le k$, let $U_i = \{u, u_1, \ldots, u_i\}$. Then we have

$$\frac{q_{J\setminus\{u\}}}{q_{J\setminus\Gamma^+(u)}} = \prod_{i=1}^{k} \frac{q_{J\setminus U_{i-1}}}{q_{(J\setminus U_{i-1})\setminus\{u_i\}}} > 2^{-k} \ge 2^{-d(u)} = p_u,$$

where the first inequality is by Item 2 of the induction hypothesis. Thus, $q_J > 0$ by (8) and Item 1 holds.

It remains to show Item 2. Recall that $J \cap \Gamma^+(u)$ is listed as $\{u, u_1, \ldots, u_k\}$. For any $0 \le i \le k$, $\Gamma^+(u_i) \not\subseteq J \setminus U_i$ because $u \in \Gamma^+(u_i)$ and $u \notin (J \setminus U_i)$. Then by the induction hypothesis on the second case of Item 2,

$$\frac{q_{(J\setminus U_i)\setminus\{u_i\}}}{q_{J\setminus U_i}} < \frac{3}{2}. \tag{9}$$

By (8), we have

$$\frac{q_J}{q_{J\setminus\{u\}}} = 1 - p_u \cdot \frac{q_{J\setminus\Gamma^+(u)}}{q_{J\setminus\{u\}}} = 1 - 2^{-d(u)} \cdot \prod_{i=0}^{k-1} \frac{q_{(J\setminus U_i)\setminus\{u_i\}}}{q_{J\setminus U_i}} \overset{(9)}{>} 1 - 2^{-d(u)} \cdot \left(\frac{3}{2}\right)^k. \tag{10}$$

If $\Gamma(u) \subseteq J$, $k \le d(u)$ and as $d(u) \ge 3$,

$$1 - 2^{-d(u)} \cdot \left(\frac{3}{2}\right)^k \ge 1 - \left(\frac{3}{4}\right)^{d(u)} \ge \frac{37}{64} > \frac{1}{2}.$$

If $\Gamma(u) \not\subseteq J$, $k \le d(u) - 1$ and, again, as $d(u) \ge 3$,

$$1 - 2^{-d(u)} \cdot \left(\frac{3}{2}\right)^k \ge 1 - \frac{2}{3} \cdot \left(\frac{3}{4}\right)^{d(u)} \ge \frac{23}{32} > \frac{2}{3}.$$

Together with (10), this finishes the proof of Item 2 and the lemma. □

In the proof above, Item 1 holds mainly because the instance is extremal. For general non-extremal cases, we would have $\frac{P_J}{P_{J\setminus\{u\}}} \ge \frac{q_J}{q_{J\setminus\{u\}}}$ for $J \subseteq V$ and $u \in J$ instead.

4.2. **Independence polynomial at negative weights.** An interesting consequence of Item 1 in the proof of Lemma 4.1 is that the number of SFOs can be computed using the independence polynomial evaluated at negative activities. More specifically, similar to (6), let

$$q_G(\mathbf{x}) = \sum_{I \in \mathsf{Ind}(G)} \prod_{u \in I} x_u,$$

where $\mathbf{x}$ is a vector of weights for each vertex. Then, $q_G(-\mathbf{p}) = q_V$ where q_V is defined in (6), and thus $|\Omega_V| = 2^{|E|} q_G(-\mathbf{p})$, where $\mathbf{p}$ is the vector $(p_u)_{u\in V}$ of failure probabilities at the vertices. Namely $p_u = 2^{-d(u)}$ where $d(u)$ is the degree of u.

There are more than one FPTASes that can efficiently approximate the independence polynomial at negative weights. The algorithm by Patel and Regts [PR17] requires that $|x_u| < \frac{(d-1)^{d-1}}{d^d}$ for all $u \in V$, where d is the maximum degree. Thus, it does not apply to the case of SFOs when there are vertices of degree, say, both 3 and 4. As the threshold is decreasing in d, it actually does not apply to any graph with constant degree vertices together with vertices of unbounded degrees.

The other algorithm, by Harvey, Srivastava, and Vondrák [HSV18], on the contrary, can work within Shearer's region, and is thus more relevant. To explain Shearer's region, let us abuse the notation slightly and extend the definition in (6) to a function $q_J(\mathbf{x}) = \sum_{I \in \mathsf{Ind}(G), I \subseteq J} \prod_{u \in I} x_u$ to take an input weight vector $\mathbf{x}$. Then, a vector $\mathbf{p}$ is in Shearer's region if and only if $q_S(-\mathbf{p}) > 0$ for all $S \subseteq V$. Lemma 4.1 implies that the probability vector for SFOs is in Shearer's region. Moreover, we say a vector $\mathbf{p}$ has slack α if $(1+\alpha)\mathbf{p}$ is in Shearer's region. For a vector $\mathbf{x}$ with slack α, the algorithm in [HSV18] approximates $q_G(\mathbf{x})$ in time $(n/(\alpha\varepsilon))^{O(\log d/\sqrt{\alpha})}$. It also does not recover Theorem 1.1 as the slack is a constant when constant degree vertices exist. If, in the meantime, some other vertices have unbounded degrees, the algorithm in [HSV18] runs in quasi-polynomial time.

To see the last point, we construct a graph that contains vertices of unbounded degrees but with constant slack for SFOs. Consider the wheel graph G, which consists of a cycle C_n of length n, and a central vertex v that connects to all vertices of C_n. Thus, $p_v = 2^{-n}$ and $p_u = 1/8$ for any u in C_n. For the cycle, as there are two SFOs, we see that $q_{C_n}(-\mathbf{1}/4) = 2^{-n+1}$ (where we use Item 1 in the proof of Lemma 4.1). Thus, by (8),

$$q_G(-2\mathbf{p}) = q_{C_n}(-\mathbf{1}/4) - 2p_v = 2^{-n+1} - 2 \cdot 2^{-n} = 0.$$

Therefore, the slack here is at most 1, despite the existence of a high degree vertex.

In summary, the existing FPTASes on the independence polynomial with negative weights do not handle the mixture of high and low degree vertices well enough for the case of SFOs. However, it might provide an alternative approach to derive FPTASes to count solutions to extremal instances of the local lemma, which is worthy of further study.

5. Concluding remarks

Originally, Bubley and Dyer [BD97a] introduced sink-free orientations as a special case of read-twice Sat. Here, "read-twice" means that each variable in a CNF formula appears exactly twice, and it corresponds to an edge of the graph. Vertices of the graph correspond to clauses of the formula. The assignment of the edge is an orientation. This represents that the variable appears with opposite signs in the formula. In fact, Bubley and Dyer showed an FPRAS for all read-twice #Sat. It is natural to ask if they admit FPTAS as well. This question was first raised by Lin, Liu, and Lu [LLL14], who also gave an FPTAS for monotone read-twice #Sat (which is equivalent to counting edge-covers in graphs). The monotone requirement means that the two appearances of any variable have the same sign. From this perspective, our FPTAS is complementary to that of [LLL14]. However, as our techniques are drastically different from [LLL14], to give an FPTAS for all read-twice #Sat, one may need to find a way to combine these two techniques to handle mixed appearances of variables.

Another immediate question is to generalise our local sampler under the partial rejection sampling framework. The first step would be to be able to handle degree 2 vertices for SFOs, which breaks our current submartingale argument. To go a bit further, a local sampler for all extremal instances would yield an FPTAS for all-terminal reliability, whose existence is a major open problem. Also, for all-terminal reliability, one may also attempt to localise the near-linear time sampler in [CGZZ24].

Lastly, in addition to the discussion of Section 4.2, let us discuss another polynomial associated with SFOs and its zero-freeness. Fix a SFO σ. Let $p(x) = \sum_{i=0}^{m} C_i x^i$, where m is the number of edges, and C_i indicates how many SFOs exist that agree with σ in exactly i edges. It is easy to evaluate $p(0) = 1$, and $p(1)$ is the total number of SFOs. However, for a cycle, this polynomial becomes $1 + x^m$, which can have a zero arbitrarily close to 1. This zero defeats, at least, the standard application of Barvinok's method [Bar16, PR17]. Although one could exclude cycles by requiring the minimum degree to be at least 3 (like we did in this paper), current techniques of proving zero-freeness seem to hinge on handling all subgraphs. For example, to use Ruelle's contraction like in [GLLZ21], one has to start from small fragments of the graph and gradually rebuild it. The obstacle then is to avoid starting from or encountering cycles in the rebuilding process. Other methods, such as the recursion-based one in [LSS19], require hereditary properties (similar to the so-called strong spatial mixing) that break in cycles as well. It would be interesting to see if any of our arguments can help in proving zero-freeness of the polynomial above.

References

[AFF+25] Konrad Anand, Weiming Feng, Graham Freifeld, Heng Guo, and Jiaheng Wang. Approximate counting for spin systems in sub-quadratic time. *TheoretiCS*, 4:3:1–3:27, 2025.

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

[ALO20] Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *FOCS*, pages 1319–1330. IEEE, 2020.

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

[BD97a] Russ Bubley and Martin E. Dyer. Graph orientations with no sink and an approximation for a hard case of #SAT. In *SODA*, pages 248–257. ACM/SIAM, 1997.

[BD97b] Russ Bubley and Martin E. Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *FOCS*, pages 223–231. IEEE Computer Society, 1997.

[BFH+16] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In *STOC*, pages 479–488. ACM, 2016.

[BG06] Antar Bandyopadhyay and David Gamarnik. Counting without sampling: new algorithms for enumeration problems using statistical physics. In *SODA*, pages 890–899. ACM Press, 2006.

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

[CFG+24] Xiaoyu Chen, Weiming Feng, Heng Guo, Xinyuan Zhang, and Zongrui Zou. Deterministic counting from coupling independence. *arXiv*, abs/2410.23225, 2024.

[CG24] Zongchen Chen and Yuzhou Gu. Fast sampling of b-matchings and b-edge covers. In *SODA*, pages 4972–4987. SIAM, 2024.

[CGZZ24] Xiaoyu Chen, Heng Guo, Xinyuan Zhang, and Zongrui Zou. Near-linear time samplers for matroid independent sets with applications. In *APPROX/RANDOM*, volume 317 of *LIPIcs*, pages 32:1–32:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[CLV21] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauber dynamics: Entropy factorization via high-dimensional expansion. In *STOC*, page 1537–1550. ACM, 2021.

[CPP02] Henry Cohn, Robin Pemantle, and James Gary Propp. Generating a random sink-free orientation in quadratic time. *Electron. J. Comb.*, 9(1), 2002.

[DFK91] Martin E. Dyer, Alan M. Frieze, and Ravi Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[EL75] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vols. I, II, III*, volume Vol. 10 of *Colloq. Math. Soc. János Bolyai*, pages 609–627. North-Holland, Amsterdam-London, 1975.

[FGW+23] Weiming Feng, Heng Guo, Chunyang Wang, Jiaheng Wang, and Yitong Yin. Towards derandomising Markov chain Monte Carlo. In *FOCS*, pages 1963–1990. IEEE, 2023.

[GH20] Heng Guo and Kun He. Tight bounds for popping algorithms. *Random Struct. Algorithms*, 57(2):371–392, 2020.

[GJ19] Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019.

[GJ21] Heng Guo and Mark Jerrum. Approximately counting bases of bicircular matroids. *Comb. Probab. Comput.*, 30(1):124–135, 2021.

[GJL19] Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *J. ACM*, 66(3):Art. 18, 31, 2019.

[GL18] Heng Guo and Pinyan Lu. Uniqueness, spatial mixing, and approximation for ferromagnetic 2-spin systems. *ACM Trans. Comput. Theory*, 10(4):17:1–17:25, 2018.

[GLLZ21] Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang. Zeros of holant problems: Locations and algorithms. *ACM Trans. Algorithms*, 17(1):4:1–4:25, 2021.

[HSV18] Nicholas J. A. Harvey, Piyush Srivastava, and Jan Vondrák. Computing the independence polynomial: from the tree threshold down to the roots. In *SODA*, pages 1557–1576. SIAM, 2018.

[HV17] Nicholas J. A. Harvey and Jan Vondrák. Short proofs for generalizations of the Lovász local lemma: Shearer's condition and cluster expansion. *arXiv*, abs/1711.06797, 2017.

[JS89] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.

[JS93] Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993.

[JVV86] 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.

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

[LLL14] Chengyu Lin, Jingcheng Liu, and Pinyan Lu. A simple FPTAS for counting edge covers. In *SODA*, pages 341–348. SIAM, 2014.

[LLY13] Liang Li, Pinyan Lu, and Yitong Yin. Correlation decay up to uniqueness in spin systems. In *SODA*, pages 67–84. SIAM, 2013.

[LSS19] Jingcheng Liu, Alistair Sinclair, and Piyush Srivastava. A deterministic algorithm for counting colorings with 2-Delta colors. In *FOCS*, pages 1380–1404. IEEE Computer Society, 2019.

[Moi19] Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *J. ACM*, 66(2):10:1–10:25, 2019.

[MT10] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11, 2010.

[PR17] Viresh Patel and Guus Regts. Deterministic polynomial-time approximation algorithms for partition functions and graph polynomials. *SIAM J. Comput.*, 46(6):1893–1919, 2017.

[She85] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.

[Val79a] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[Val79b] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[Wei06] Dror Weitz. Counting independent sets up to the tree threshold. In *STOC*, pages 140–149. ACM, 2006.

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