Fundamentals of partial rejection sampling

Mark Jerrum[*]

School of Mathematical Sciences, Queen Mary, University of London, Mile End Road, London, E1 4NS, e-mail: m.jerrum@qmul.ac.uk

Abstract: Partial Rejection Sampling is an algorithmic approach to obtaining a perfect sample from a specified distribution. The objects to be sampled are assumed to be represented by a number of random variables. In contrast to classical rejection sampling, in which all variables are resampled until a feasible solution is found, partial rejection sampling aims at greater efficiency by resampling only a subset of variables that 'go wrong'. Partial rejection sampling is closely related to Moser and Tardos' algorithmic version of the Lovász Local Lemma, but with the additional requirement that a specified output distribution should be met. This article provides a largely self-contained account of the basic form of the algorithm and its analysis. Working within a unified framework allows a clean expression of the running time, and clarifies the scope for nondeterminism in its implementation.

Contents

1. The setting

The aim of this expository article is to provide a uniform treatment of a particular approach to sampling combinatorial structures. The method is a development of classical rejection sampling. Suppose $\Phi(\mathbf{X})$ is a predicate (Boolean function) depending on random variables $\mathbf{X} = (X_1, \ldots, X_n)$ coming from a product distribution. We would like to obtain a sample from the conditional distribution of $\mathbf{X}$ given that $\Phi(\mathbf{X})$ holds. Classical rejection sampling repeatedly generates realisations of $\mathbf{X}$ from the product distribution until one that satisfies Φ is found, and then outputs that. In many situations this approach is very inefficient, as satisfying assignments to Φ may occur with exponentially small probability (in n). The idea in Partial Rejection Sampling (PRS) is to identify small subsets of the variables that violate Φ (in some sense), and resample just those variables. Clearly, the choice of which variables to resample has to be done with care if the output distribution is to remain the correct one.

The phrase 'partial rejection sampling' appears to have been coined by Cohn, Pemantle and Propp [8] to describe their approach to sampling sink-free orientations. Noting the similarity to Wilson's approach to sampling spanning trees [34], they wondered whether one could develop a general theory. PRS as a general algorithmic technique was explored by Guo, Jerrum and Liu [20], and is our topic here.

We focus on examples of PRS inspired by algorithmic proofs of the Lovász Local Lemma (LLL). There is a substantial literature on this topic to assist us, but it is concerned only with problem of constructing *some* satisfying assignment to Φ. Partial rejection sampling adds the novel requirement that the output should be uniform (or, more generally, from the desired distribution) on satisfying assignments. This additional requirement adds a new challenge.

In the study of the LLL, the class of 'extremal' instances receives particular attention. The extremal instances are particularly suited to PRS and we treat them first. It is unlikely that anything in this section of the article is conceptually new. However, the treatment of PRS in the extremal regime involves some particularly beautiful combinatorial ideas, and it is seems a good time to bring this material together in one place, with a consistent approach and notation.

After that, we investigate to what extent the conditions defining extremal instances can be relaxed. The viewpoint taken in this section is novel in that correctness and efficiency are treated in a manner than is more consistent with that for extremal instances. This has the advantage of clarifying the conditions under which PRS continues to function in the non-extremal setting, in the hope that it will help in discovering new applications. Additionally, it allows greater attention to be paid to the flexibility in the order in which variables can be resampled.

The scope of this article is limited to versions of partial rejection sampling that stay close to the spirit of the algorithmic LLL pioneered by Moser and Tardos [32]. In particular, the number of random variables is finite and all constraints on them are 'hard'. Informally, we restrict attention to a 'combinatorial' setting, which excludes important applications to spin systems in statistical

physics. We finish with a few pointers to work that goes beyond the framework presented here.

It should be noted that PRS is not the only approach to perfect sampling. Perhaps the best known and most extensively studied is 'Coupling From The Past' (CFTP), which was pioneered by Propp and Wilson [34]. Other approaches include Fill and Huber's 'Randomness recycler' [13] and Anand and Jerrum's 'Lazy depth-first sampler' [1].

2. Partial rejection sampling in the extremal setting

Suppose $\Phi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_m$ is formula on variables $\mathbf{X} = (X_1, X_2, \ldots, X_n)$. Each clause φ_k depends on a certain tuple $(X_{i_1}, \ldots, X_{i_{a_k}})$ of variables, where a_k is the *arity* of φ_k. We refer to the tuple $\mathsf{Scp}(\varphi_k) = (i_1, \ldots, i_{a_k})$ of indices as the *scope* of the clause φ_k. By assuming $i_1 < i_2 < \cdots < i_{a_k}$ we can consider the scope $\mathsf{Scp}(\varphi_k)$ to be a a_k-tuple or a set of cardinality a_k, according to context. For a set $S \subseteq \{1, \ldots, n\}$ of indices we write $X_S = \{X_i : i \in S\}$. Then to emphasise the dependence on the variables we can write

$$\Phi(\mathbf{X}) = \varphi_1(X_{\mathsf{Scp}(\varphi_1)}) \wedge \cdots \wedge \varphi_m(X_{\mathsf{Scp}(\varphi_m)}).$$

Definition 2.1. *We say that the formula* $\Phi = \varphi_1 \wedge \cdots \wedge \varphi_m$ *is* extremal *if, for all* $1 \leq k < \ell \leq m$ *satisfying* $\mathsf{Scp}(\varphi_k) \cap \mathsf{Scp}(\varphi_\ell) \neq \emptyset$, *it is the case that* $\varphi_k(X_{\mathsf{Scp}(\varphi_k)}) \vee \varphi_\ell(X_{\mathsf{Scp}(\varphi_\ell)})$ *is a tautology. In other words, any two clauses that are probabilistically dependent cannot both be false.*

In this section we consider only extremal instances, as these can be dealt with using the basic form of partial rejection sampling.

Now suppose that variable X_i takes values from a set D_i. Equip D_i with a probability distribution and call the probability space $\mathcal{D}_i$. We are interested in sampling a realisation of the random variable $\mathbf{X}$ from the product distribution $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n$ conditioned on $\Phi(\mathbf{X})$ holding. Denote this desired distribution by $\mathcal{D}_\Phi$. Partial Rejection Sampling (PRS) is a simple algorithm for accomplishing this task in the context of extremal instances. It runs as follows.

Algorithm 1 Partial Rejection Sampling

$\mathrm{PRS}(\Phi, \mathcal{D})$ // Φ is a formula on variable set $\mathbf{X}$
Sample $\mathbf{X}$ from the product distribution $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$
while $\neg\Phi(\mathbf{X})$ **do**
    Choose any clause φ_k with $\neg\varphi_k(X_{\mathsf{Scp}(\varphi_k)})$
    Resample all variables in $\mathsf{Scp}(\varphi_k)$
end while

In the resampling step, the product distribution $\prod_{i \in \mathsf{Scp}(\varphi_k)} \mathcal{D}_i$ is naturally being used.

The algorithm PRS was first introduced by Moser and Tardos [32] in the context of an algorithmic proof of the Lovász Local Lemma. Its application to sampling from naturally specified distributions was studied by Guo, Jerrum and

$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\omega_{1,4}$	$\omega_{2,4}$	$\omega_{3,4}$	$\omega_{4,4}$	$\omega_{5,4}$
$\omega_{1,3}$	$\omega_{2,3}$	$\omega_{3,3}$	$\omega_{4,3}$	$\omega_{5,3}$
$\omega_{1,2}$	$\omega_{2,2}$	$\omega_{3,2}$	$\omega_{4,2}$	$\omega_{5,2}$
$\omega_{1,1}$	$\omega_{2,1}$	$\omega_{3,1}$	$\omega_{4,1}$	$\omega_{5,1}$
$\omega_{1,0}$	$\omega_{2,0}$	$\omega_{3,0}$	$\omega_{4,0}$	$\omega_{5,0}$
X_1	X_2	X_3	X_4	X_5

Fig 1. *A resampling table*

Liu [20], who analysed its correctness and efficiency. Although their investigation seems to be the first attempt to treat PRS as a general technique, several specific examples had previously appeared in the literature as we noted above.

Remark. *In the combinatorial community, the Moser and Tardos algorithm would be viewed as gradually eliminating the set of 'bad events' until none are left. In the area of constraint satisfaction, the goal is to simultaneously satisfy a collection of constraints. It is important to keep in mind that, of these diametrically opposing conventions, we use the latter here.*

In classical rejection sampling we would resample the whole of **X** on each iteration. In contrast, PRS resamples only a subset of offending variables. We cannot expect the correctness of the algorithm to survive such extreme corner cutting. Indeed, for general formulas Φ, the call $\mathrm{PRS}(\Phi, \mathcal{D})$ does not produce a sample from the distribution $\mathcal{D}_\Phi$. Surprisingly, PRS *does* achieve the desired distribution for extremal instances.

Theorem 2.2. *Suppose Φ is a satisfiable extremal instance. Then $\mathrm{PRS}(\Phi, \mathcal{D})$ terminates with probability 1. On termination, **X** is a realisation of a random variable from the distribution $\mathcal{D}_\Phi$.*

To analyse the algorithm, we need to introduce time explicitly. A *resampling table* is a semi-infinite matrix $(\omega_{i,j} : 1 \leq i \leq n$ and $j \in \mathbb{N})$. Each entry $\omega_{i,j}$ in the table is an independent sample from the distribution $\mathcal{D}_i$. Fixing i, the sequence $\omega_{i,0}, \omega_{i,1}, \omega_{i,2}, \ldots$ specifies the sequence of values taken by the random variable X_i during the execution of the algorithm. Introducing a superscript to indicate the time t (each iteration of the loop takes one time unit), we write $X_i^t = \omega_{i,j(i,t)}$. If X_i is resampled during iteration t then $j(i,t) = j(i,t-1)+1$, otherwise $j(i,t) = j(i,t-1)$. Initially, $j(i,0) = 0$ for all $1 \leq i \leq n$. (By convention, we start at time 0, and iteration t occupies the interval between time $t-1$ and time t.) At any time t, the *frontier* of the resampling table is $F(t) = \{(i, j(i,t)) : 1 \leq i \leq n\}$. See Figure 1.

As time progresses, we record the actions of the algorithm in the form of a partition of the portion of the resampling table that lies behind the frontier, namely $\{(i,j) : 1 \leq i \leq n$ and $0 \leq j < j(i,t)\}$. The partition builds as the frontier advances. In iteration t, the variables in the scope of some clause φ_k are resampled. The locations $\{(i, j(i, t-1)) : i \in \mathsf{Scp}(\varphi_k))\}$ that were on the frontier now lie behind it; this set of locations now forms a new block of the

$$
\begin{array}{ccccc}
\vdots & & & \vdots & \\
0 & \vdots & \vdots & 1 & \vdots \\
1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
\hline
X_1 & X_2 & X_3 & X_4 & X_5
\end{array}
$$

$t = 7$: $\boxed{0 \quad 1 \quad 0 \quad 1 \quad 1}$

$t = 6$: (1) —— (0)(1) $\quad \varphi_2$

$t = 4, 5$: (0)(0) (0)(1) $\quad \varphi_1, \varphi_4$

$t = 3$: (1)(1) —— (0) $\quad \varphi_3$

$t = 2$: (0)(0) $\quad \varphi_1$

$t = 1$: (1) —— (0)(1) $\quad \varphi_2$

$t = 0$: (0)(1) $\quad \varphi_4$
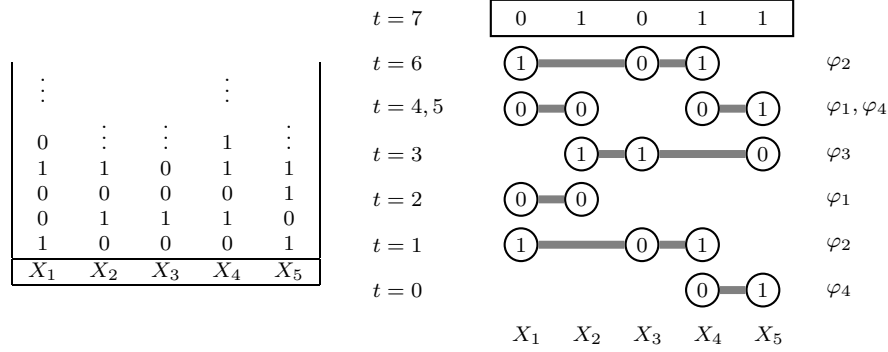
$$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5$$

FIG 2. *A realisation of a resampling table, and the corresponding transcript*

partition. We call these blocks the *resampling blocks*. At time t, the frontier together with the partition into resampling blocks forms a *transcript* of the run of the algorithm up to time t.

By way of example, consider the formula

$$\Phi(\mathbf{X}) = (X_1 \vee X_2) \wedge (\neg X_1 \vee X_3 \vee \neg X_4) \wedge (\neg X_2 \vee \neg X_3 \vee X_5) \wedge (X_4 \vee \neg X_5) \quad (2.1)$$

on variables $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5)$. (The formula Φ encodes sink-free orientations of a certain 4-vertex graph, a point we shall return to later.) Thus $\varphi_2 = \neg X_1 \vee X_3 \vee \neg X_4$ and $\mathsf{Scp}(\varphi_2) = \{1, 3, 4\}$, and similarly for the other clauses. A particular realisation of the resampling table that leads to termination of the algorithm PRS, together with its associated transcript, are depicted in Figure 2. In the pictorial representation of the transcript, the values in the resampling table are spread out along the columns so that each resampling block of the transcript occupies a single row. The rectangle at the top denotes the *final frontier*, i.e., the frontier at termination. Initially, $X_4 = 0$ and $X_5 = 1$, which violates clause φ_5. Accordingly, variables X_4 and X_5 are resampled, and $\{(4, 0), (5, 0)\}$ becomes the first resampling block of the transcript. The value of X_4 switches from 0 to 1, and this causes clause φ_2 to be violated, since now $X_1 = X_4 = 1$ and $X_3 = 0$. So X_1, X_3 and X_4 are resampled and $\{(1, 0), (3, 0), (4, 1)\}$ becomes the next resampling block of the transcript. Eventually, $\mathbf{X} = (0, 1, 0, 1, 1)$, which satisfies Φ, and the algorithm halts.

Suppose we run the algorithm PRS twice, using different non-deterministic choices (of which clauses to resample), until termination. A priori, it might be imagined that the two runs would in general have different transcripts, but this is not the case, as we shall see in Lemma 2.3. Some intuition can be gained from Figure 2. At time 4, $\mathbf{X} = (0, 0, 0, 0, 1)$, and hence clauses φ_1 and φ_4 are both violated. We can resample either $\{X_1, X_2\}$ first or $\{X_4, X_5\}$, but either way we end up with the *same* transcript. In this context, it is crucial that $\mathsf{Scp}(\varphi_1) \cap \mathsf{Scp}(\varphi_4) = \emptyset$, but in an extremal instance, this condition is guaranteed.

Lemma 2.3. *Let Φ be an extremal formula. Fix a resampling table. Suppose that for some sequence of non-deterministic choices,* $\mathrm{PRS}(\Phi, \mathcal{D})$ *terminates with*

a certain transcript. Then for any other sequence of choices, the algorithm will terminate with the same transcript.

To prove this lemma, we use a version of Newman's Lemma that is particularly convenient in this application. An (abstract) rewriting system is simply a set $\mathcal{T}$ of 'positions' together with a binary 'rewriting' relation $\rightarrow$ on T. For positions $t, s \in \mathcal{T}$, the relation $t \rightarrow s$ indicates that it is possible to go from t to s in one move. A position t from which no valid move $t \rightarrow s$ is possible is said to be *terminal*. A sequence of moves ending at a terminal state is said to be *terminating*. Following Eriksson [10], we say that the rewriting system $(\mathcal{T}, \rightarrow)$ has the *polygon property* if, given any position $t \in \mathcal{T}$ and two moves $t \rightarrow s$ and $t \rightarrow s'$, either (a) there are two sequences $s = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_\ell = t^*$ and $s' = s_0' \rightarrow s_1' \rightarrow \cdots \rightarrow s_\ell' = t^*$ of the same length ℓ that end at the same position t^*, or (b) there are two infinite sequences of moves starting from s and s'. A rewriting system is said to have the *strong convergence property* if, for any starting position t from which there exists a sequence of moves terminating at some position t^*, it is the case that every sequence of moves starting from t will lead to t^*, and in the same number of moves. Eriksson [10, Thm 2.1] showed the following.

Lemma 2.4. *A rewriting system has the strong convergence property iff it has the polygon property.*

Proof of Lemma 2.3. Fix a resampling table. View the collection of all possible transcripts as an abstract rewriting system by introducing a binary relation $\rightarrow$ on transcripts. The meaning of $t \rightarrow s$ is that s can follow t in one iteration of PRS. This rewriting system has the diamond property, namely if $t \rightarrow s$ and $t \rightarrow s'$ then there exists t^* such that $s \rightarrow t^*$ and $s' \rightarrow t^*$. (This diamond property is clearly stronger than the polygon property.) For suppose $t \rightarrow s$ is a result of resampling the variables in scope $\mathsf{Scp}(\varphi_k)$, and $t \rightarrow s'$ the result of resampling $\mathsf{Scp}(\varphi_\ell)$. Since the instance Φ is extremal we know that $\mathsf{Scp}(\varphi_k) \cap \mathsf{Scp}(\varphi_\ell) = \emptyset$. Thus, we can resample whichever scope was not resampled in the first step, to get to a common transcript t^*. The result now follows from Lemma 2.4. $\square$

Proof of Theorem 2.2. Fix a particular satisfying assignment $\mathbf{X} = \mathbf{b} = (b_1, \ldots, b_n)$ to Φ. At any point in the execution of the algorithm, the following fortuitous sequence of events may occur over the next n iterations: each time a variable X_i is resampled, it is assigned the value b_i. On each iteration, $\mathbf{X}$ approaches closer to $\mathbf{b}$ in Hamming distance. Thus, the algorithm will terminate in the next n iterations. Since this fortuitous sequence of events occurs with probability bounded away from 0, the running time of the algorithm PRS is stochastically dominated by an exponential random variable with finite mean. So the algorithm terminates with probability 1.

Fix a resampling table T, and run PRS on T to obtain a transcript. Since the algorithm has terminated, we know that the frontier contains a satisfying assignment. Create a new resampling table T' by replacing the values in the frontier by some other satisfying assignment. Now run the algorithm again on T' with the same nondeterministic choices of scopes to resample. Note that this
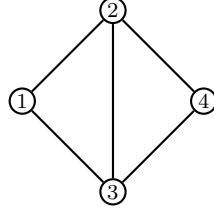
FIG 3. *The dependency graph Γ corresponding to formula Φ defined in (2.1).*

is always possible: whenever the algorithm running on T resamples $\mathsf{Scp}(\varphi_k)$ at time t it is because $\varphi_k(\omega_{i,j(i,t)} : i \in \mathsf{Scp}(\varphi_k))$ is false. None of the resampled variables are in the final frontier, since no variables beyond the final frontier are ever inspected. So the clause φ_k is also false when the algorithm is run on table T', and it is valid step to resample $\mathsf{Scp}(\varphi_k)$. Finally, on the same iteration that the algorithm terminates when run on table T, it will also terminate on T'. The same transcript (i.e., frontier $F(t)$ together with the partition of the table behind the frontier) arises from running the algorithm on T' as the one that arose from the run on T.

By Lemma 2.3 any sequence of non-deterministic choices made by the algorithm on table T' leads to the same transcript. Summarising, the final transcript does not depend on the nondeterministic choices made by the algorithm, and is also unchanged if one satisfying assignment is substituted for another in the final frontier. Thus, conditioned on the transcript, each satisfying assignment $\mathbf{X} = (b_1, \ldots, b_n)$ of Φ occurs with probability proportional to $\mathcal{D}_1(b_1)\mathcal{D}_2(b_2)\cdots\mathcal{D}_n(b_n)$. So, at termination, $\mathbf{X}$ is distributed as $\mathcal{D}_\Phi$. $\qquad\square$

There is a remarkably simple (though not simple to derive) formula for the expected number of iterations in a run of algorithm PRS, which we now present. Kolipaka and Szegedy [28] derived this formula as an upper bound, but it is in fact exact. Given Φ, define $\Gamma = \Gamma(\Phi)$ to be the *dependency graph* with vertex set $[m]$ (where vertex k corresponds to clause φ_k) and edge relation $\sim$ defined by $k \sim \ell$ iff $\mathsf{Scp}(\varphi_k) \cap \mathsf{Scp}(\varphi_\ell) \neq \emptyset$. (Refer to Figure 3 for an example.) Let $\Sigma = \{s_1, s_2, \ldots, s_m\}$ be an alphabet of m symbols. If $k \sim \ell$ then symbols s_k and s_ℓ do not commute; otherwise, s_k and s_ℓ do commute, i.e., $s_k s_\ell = s_\ell s_k$. Denote by $\mathcal{R}_\Gamma$ the set of commutation relations:

$$\mathcal{R}_\Gamma = \{s_k s_\ell = s_\ell s_k : k \not\sim \ell\}.$$

The set $\Sigma^*/\mathcal{R}_\Gamma$ of *traces* over Σ is the set of all words over the alphabet Σ quotiented by the commutation relations $\mathcal{R}_\Gamma$. So a trace can be thought of as word over Σ where we regard two words as indistinguishable if one can be obtained from the other by transposing adjacent commuting symbols.

There is an elegant expression for the generating function for traces. Introduce

indeterminates $z_1, \ldots, z_m$ corresponding to the m clauses in Φ, and define

$$P_\Gamma(z_1, \ldots, z_m) = \sum_{I \in \mathcal{I}(\Gamma)} (-1)^{|I|} z_I,$$

where $z_I = \prod_{k \in I} z_k$ and $\mathcal{I}(\Gamma)$ is the set of all independent sets in Γ. Note that the polynomial P_Γ is the generating function of independent sets in Γ, with terms signed according to parity. The generating function for traces $\Sigma^*/\mathcal{R}$ is the multivariate polynomial $T_\Gamma(z_1, \ldots, z_m)$ in which the coefficient of $z_1^{e_1} z_2^{e_2} \cdots z_m^{e_m}$ is the number of traces in which symbol s_1 occurs e_1 times, s_2 occurs e_2 times, etc. The following expression for the trace generating function is due to Cartier and Foata [6]. The derivation can also be found, e.g., in Knuth [27, Thm F] and Viennot [38, Prop. 5.1].

Lemma 2.5. *With Γ, P_Γ as above, the generating function T_Γ for traces $\Sigma^*/\mathcal{R}$ is given by $T_\Gamma(z_1, \ldots, z_m) = P_\Gamma(z_1, \ldots, z_m)^{-1}$.*

Take, as an example, the dependency graph Γ from Figure 3. The generating function for signed independent sets in Γ is

$$P_\Gamma(z, z_2, z_3, z_4) = 1 - z_1 - z_2 - z_3 - z_4 + z_1 z_4,$$

encoding the empty independent set $\emptyset$, the four singleton independent sets $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, and the unique independent set $\{1, 4\}$ of size two. Then,

$$\begin{aligned}
P_\Gamma(z, z_2, z_3, z_4)^{-1} &= (1 - (1 - P_\Gamma))^{-1} \\
&= 1 + (1 - P_\Gamma) + (1 - P_\Gamma)^2 + (1 - P_\Gamma)^3 + \cdots \\
&= 1 + z_1 + z_2 + z_3 + z_4 + z_1^2 + z_2^2 + z_3^2 + z_4^2 \\
&\quad + 2z_1 z_2 + 2z_1 z_3 + z_1 z_4 + 2z_2 z_3 + 2z_2 z_4 + 2z_3 z_4 \\
&\quad + \text{terms of degree 3 and higher,}
\end{aligned}$$

(Note that $1 - P_\Gamma$ has no constant term, so the expansion makes sense.) Observe that the coefficient of $z_2 z_3$ is 2, reflecting the fact that $s_2 s_3$ and $s_3 s_2$ are distinct traces, while the coefficient of $z_1 z_4$ is 1, as $s_1 s_4$ and $s_4 s_1$ are equivalent as traces.

The motivation for introducing traces is that they are in perfect correspondence with transcripts, where the m symbols correspond to the m possible kinds of resampling blocks; specifically, symbol s_k corresponds to a block arising from resampling $\mathsf{Scp}(\varphi_k)$. Let $w = s_{i_1} s_{i_2} \ldots s_{i_t}$ be any word in Σ^*. Consider the transcript that results if the algorithm PRS performs block resamplings in the order $\mathsf{Scp}(\varphi_{i_1}), \mathsf{Scp}(\varphi_{i_2}), \ldots, \mathsf{Scp}(\varphi_{i_t})$. Now let $w' = s_{i'_1} s_{i'_2} \ldots s_{i'_t}$ be any word in Σ^* that is equivalent to w under the commutation relations $\mathcal{R}$. It is not difficult to see that the same transcript results from the sequence of block resamplings $\mathsf{Scp}(\varphi_{i'_1}), \mathsf{Scp}(\varphi_{i'_2}), \ldots, \mathsf{Scp}(\varphi_{i'_t})$. (Transposing the order of two adjacent commuting symbols transposes the order in which two blocks are resampled; however, those blocks have no variables in common, so there is no change in the transcript.) Conversely, if words w and w' lead to the same transcript then they must be equivalent under commutativity. (Suppose $s_{i_1} \neq s_{i'_1}$. Let $s_{i'_h}$ be the

first occurrence of the symbol s_{i_1} in w'. The first $h - 1$ resamplings prompted by w' did not disturb the variables in $\mathsf{Scp}(\Phi_{i_1})$. Therefore, $s_{i'_h}$ commutes with all earlier symbols in w' and can be 'bubbled' into first place. The remaining symbols can be brought into alignment inductively.) Transcripts are exactly the *empilements [des pièces]* or 'heaps of pieces' of Viennot [38], who gives a beautiful pictorial explanation of the correspondence between *empilements* (and hence transcripts) and traces. See also Knuth [27, §7.2.2.2].

The correspondence between traces and transcripts can be appreciated pictorially in Figure 2. The depiction of the transcript is based on Viennot's *empilements*. Knuth invites us to think of each symbol as a piece in Tetris that appears from above and descends until further progress is obstructed. The word $w = s_4 s_2 s_1 s_3 s_1 s_4 s_2$ specifies an order for the arriving pieces that leads to the transcript on the right of the figure. The word $w' = s_4 s_2 s_1 s_3 s_4 s_1 s_2$ leads to the same transcript, since s_1 and s_4 commute. In contrast, the word $s_4 s_2 s_3 s_1 s_1 s_4 s_2$ results in a different transcript, as s_1 and s_3 do not commute: the pieces corresponding to symbols s_1 and s_3 cannot pass each other. The equivalence class $\{w, w'\}$ is a trace, since the only adjacent commuting pair of symbols is $s_1 s_4$. Traces, *empilements* and transcripts are different views of the same concept.

Before analysing the runtime of algorithm PRS, let us observe that it is remarkably easy to compute the probability of observing a particular transcript such as the one in Figure 2. Recall that each value in the resampling table is the result of an independent toss of a fair coin. At time $t = 0$, we have that $\omega_{4,0} = 0$ and $\omega_{5,0} = 1$, an event that occurs with probability $\frac{1}{4}$. (The only way for $\varphi_4(X_4, X_5)$ to be false is for X_4 to be 0, and X_5 to be 1.) At time $t = 1$ we know that $\omega_{1,0} = 1$, $\omega_{3,0} = 0$ and $\omega_{4,1} = 1$, an event with probability $\frac{1}{8}$, and so on for times $t = 2, 3, 4, 5, 6$. All these events are independent, and the probability that they all occur is 2^{-17}. Finally, the frontier must contain a satisfying assignment; there are 10 satisfying assignments out of a total of 32, so the probability of observing the transcript depicted is 10×2^{-22}.

For $k \in [m]$, let $p_k = \mathrm{Pr}_{\mathcal{D}}(\neg \varphi_k)$ denote the probability that φ_k is false in the product distribution, and extend this notation to a set of clauses $S \subseteq [m]$ by letting $p_S = \prod_{k \in S} p_k$. Then define

$$q_S = \sum_{I \in \mathcal{I}(\Gamma) : I \supseteq S} (-1)^{|I \setminus S|} p_I.$$

Note that $q_S = 0$ if $S \notin \mathcal{I}(\Gamma)$. Note also that

$$P_\Gamma(p_1, \ldots, p_m) = \sum_{I \in \mathcal{I}(\Gamma)} (-1)^{|I|} p_I = q_\emptyset \tag{2.2}$$

and

$$p_k P_{\Gamma - N[k]}(p_1, \ldots, p_m) = p_k \sum_{I \in \mathcal{I}(\Gamma - N[k])} (-1)^{|I|} p_I$$

$$= \sum_{I \in \mathcal{I}(\Gamma) : I \ni k} (-1)^{|I| - 1} p_I$$

$$= q_{\{k\}}, \tag{2.3}$$

where $\Gamma - N[k]$ denotes the graph obtained from the dependency graph Γ by removing the closed neighbourhood of k and incident edges. (The *closed neighbourhood* $N[k]$ of k is the set containing vertex k and all its neighbours.) Although we have written $P_{\Gamma-N[k]}(p_1, \ldots, p_m)$ above, the polynomial $P_{\Gamma-N[k]}$ depends only on those variables from $p_1, \ldots, p_m$ that survive when vertex k is removed, and is independent of the others.

In the case of extremal instances, the quantity $q_\emptyset$ has a simple probabilistic interpretation. By the principle of inclusion-exclusion,

$$
\begin{aligned}
\Pr_{\mathcal{D}}(\Phi) &= \Pr_{\mathcal{D}} \left(\bigwedge_{k \in [m]} \varphi_k \right) \\
&= \sum_{S \subseteq [m]} (-1)^{|S|} \Pr_{\mathcal{D}} \left(\bigwedge_{k \in S} \neg\varphi_k \right) \\
&= \sum_{I \in \mathcal{I}(\Gamma)} (-1)^{|I|} \prod_{k \in I} \Pr_{\mathcal{D}}(\neg\varphi_k) \qquad \text{(see below)} \tag{2.4} \\
&= \sum_{I \in \mathcal{I}(\Gamma)} (-1)^{|I|} p_I \\
&= q_\emptyset. \tag{2.5}
\end{aligned}
$$

Equality (2.4) uses two facts: (a) when S is not an independent set the corresponding term is zero, by extremality, and (b) for any independent set I, the events $\{\neg\varphi_k : k \in I\}$ are probabilistically independent. Note, in particular, that $q_\emptyset > 0$ when Φ is satisfiable.

Theorem 2.6. *Suppose Φ is a satisfiable extremal instance. Then the expected number of resamplings of the scope of φ_k during a run of $\mathrm{PRS}(\Phi, \mathcal{D})$ is $q_{\{k\}}/q_\emptyset$.*

Proof. As noted earlier, the generating function for transcripts is $P_\Gamma(z_1, \ldots, z_m)^{-1}$. We claim that the generating function for transcripts weighted according to probability of occurrence is $q_\emptyset P_\Gamma(p_1 z_1, \ldots, p_m z_m)^{-1}$. In other words, the probability of observing a transcript with e_k resamplings of scope $\mathsf{Scp}(\varphi_k)$, for $1 \leq k \leq m$, is the coefficient of $z_1^{e_1} z_2^{e_2} \cdots z_m^{e_m}$ in $q_\emptyset P_\Gamma(p_1 z_1, \ldots, p_m z_m)^{-1}$. To see this, fix a transcript with e_k resamplings of scope $\mathsf{Scp}(\varphi_k)$, for $1 \leq k \leq m$, and consider the probability that a random resampling table will generate that transcript. The frontier must contain a satisfying assignment, which happens with probability $q_\emptyset$, by (2.5). Each block corresponding to a clause φ_k must contain an assignment making φ_k false, which happens with probability p_k. All these probabilities are independent, so the overall probability of observing the transcript is $q_\emptyset \prod_{1 \leq k \leq m} p_k^{e_k}$. The claim follows. Note that we have used that the fact that if it is *possible* for a certain transcript to arise from a given resampling table it *will* do so.

Note that $1 - P_\Gamma(p_1, \ldots, p_m) = 1 - q_\emptyset \in [0, 1)$, and so the power series

expansion

$$P_\Gamma(p_1 z_1, \ldots, p_m z_m)^{-1} = \sum_{i=0}^{\infty} \left(1 - P_\Gamma(p_1 z_1, \ldots, p_m z_m)\right)^i$$

converges in an open neighbourhood of the point $z_1 = \cdots = z_m = 1$. The expected number of resamplings of the scope of φ_k is given by

$$q_\emptyset \frac{\partial}{\partial z_k} P_\Gamma(p_1 z_1, \ldots, p_m z_m)^{-1} \bigg|_{z_1 = \cdots = z_m = 1}$$

$$= -q_\emptyset P_\Gamma(p_1 z_1, \ldots, p_m z_m)^{-2} \frac{\partial}{\partial z_k} P_\Gamma(p_1 z_1, \ldots, p_m z_m) \bigg|_{z_1 = \cdots = z_m = 1}$$

$$= q_\emptyset P_\Gamma(p_1, \ldots, p_m)^{-2} p_k P_{\Gamma - N[k]}(p_1, \ldots, p_m).$$

We use here the fact that P_Γ is multilinear, so differentiating with respect to z_k eliminates terms corresponding to independent sets that do not include k. Using identities (2.2) and (2.3), we see that the expected number of times $\mathsf{Scp}(\varphi_k)$ is resampled is $q_{\{k\}}/q_\emptyset$. □

We can recast the above theorem in a simple, easy to use form.

Corollary 2.7. *The expected number of iterations of Algorithm PRS on input* $(\Phi, \mathcal{D})$ *is*

$$\mathbb{E}(\#\text{iterations}) = \frac{\Pr_{\mathcal{D}}(\text{Exactly one clause in } \Phi \text{ is false})}{\Pr_{\mathcal{D}}(\Phi \text{ is true})}.$$

Proof. Generalising the inclusion-exclusion argument used earlier, and assuming $S \in \mathcal{I}(\Gamma)$, we have

$$\Pr_{\mathcal{D}}\left(\bigwedge_{k \in S} \neg\varphi_k \wedge \bigwedge_{k \in [m] \setminus S} \varphi_k\right) = \sum_{S' \supseteq S} (-1)^{|S' \setminus S|} \Pr_{\mathcal{D}}\left(\bigwedge_{k \in S'} \neg\varphi_k\right)$$

$$= \sum_{I \in \mathcal{I}(\Gamma): I \supseteq S} (-1)^{|I \setminus S|} \prod_{k \in I} \Pr_{\mathcal{D}}(\neg\varphi_k)$$

$$= \sum_{I \in \mathcal{I}(\Gamma): I \supseteq S} (-1)^{|I \setminus S|} p_I = q_S.$$

When $S \notin \mathcal{I}(\Gamma)$, the above equality continues to hold, as both side are zero. In particular, the probability that clause φ_k is false, and all others true, is precisely $q_{\{k\}}$. The result now follows from Theorem 2.6. □

Thanks to Lemma 2.3, the above results are completely robust against changes in the implementation of algorithm PRS. Thus, the next scope to be resampled can be selected by arbitrary means: the choice can be made on the current values of variables, the past execution of the algorithm, or even externalities such as random bits or the system clock. It is also valid to resample several blocks
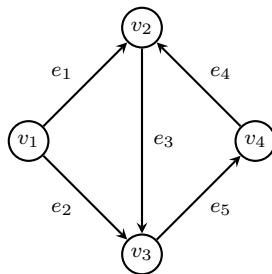
FIG 4. *A sample graph $\vec{G}$ incorporating a reference orientation*

simultaneously, in case several clauses are violated. If one is interested in the expected number of individual variables resampled, this can also be accessed though

$$\mathbb{E}(\#\text{variables resampled}) = \sum_{k=1}^{m} \frac{q_{\{k\}} a_k}{q_\emptyset}, \tag{2.6}$$

where a_k is the arity of φ_k, for $1 \leq k \leq m$.

3. Example applications

One application, to sink-free orientations, will be done in detail to illustrate the methods, and the other applications merely sketched. For ease of presentation, all examples will be unweighted, i.e, the probability distributions $\mathcal{D}_i$ are all uniform, as is the output distribution. Incorporating weights does not require any conceptual changes.

3.1. Sink-free orientations of a graph

This approach to sampling sink-free orientations of a graph was introduced by Cohn, Pemantle and Propp [8], and placed within the general framework of PRS by Guo, Jerrum and Liu [20, §4.1].

Suppose $G = (V, E)$ is a graph with vertex set $\{v_1, v_2, \ldots, v_m\}$ and edge set $E = \{e_1, e_2, \ldots, e_n\}$.[1] We wish to sample, uniformly at random, an orientation of the edges of G that has no sinks, where a *sink* is a vertex v_i at which all incident edges are oriented towards v_i. We assume that G has at least one such sink-free orientation. It is convenient to choose a reference orientation for the edges of G that is sink-free; denote by $\vec{G}$ the directed graph obtained from G by giving the edges of G this reference orientation.

To fit the pattern of PRS, we introduce Boolean variables $X_1, X_2, \ldots, X_n$ and associate variable X_i to edge e_i, for $1 \leq i \leq n$. These variables will be used to encode orientations of the edges of G. The variable X_i is to be interpreted as

---

[1]The roles of n and m are reversed relative to the usual convention in graph theory, but this is necessary to preserve consistency with the previous section.

follows: if $X_i = 0$ then the edge e_i is oriented against the reference orientation (of e_i in $\vec{G}$) and if $X_i = 1$ then e_i is oriented with the reference orientation. Next, introduce clauses $\{\varphi_k\}$ to encode the event that vertex v_k is not a sink. So the scope of Φ_k is the set $\mathsf{Scp}(\Phi_k) = \{i : e_i \text{ is incident at } v_k\}$, and the clause φ_k asserts that at least one edge incident at vertex v_k is oriented away from v_k. By way of example, consider the graph G in Figure 4, which has been assigned a reference orientation to give a sink-free directed graph $\vec{G}$. The condition that vertex v_2, for example, is not a sink is asserted by the clause $\varphi_2(X_1, X_3, X_4) = \neg X_1 \vee X_3 \vee \neg X_4$. Then Φ is the formula

$$\Phi(\mathbf{X}) = (X_1 \vee X_2) \wedge (\neg X_1 \vee X_3 \vee \neg X_4) \wedge (\neg X_2 \vee \neg X_3 \vee X_5) \wedge (X_4 \vee \neg X_5)$$

that we encountered already in the previous section.

We observed earlier that Φ is an extremal instance. This is true in general for sink-free orientations. If we have indices $1 \le k < \ell \le m$ such that $\mathsf{Scp}(\varphi_k) \cap \mathsf{Scp}(\varphi_\ell) \ne \emptyset$ then necessarily vertices v_k and v_ℓ are adjacent. But then it is impossible for v_k and v_ℓ to both be sinks, and hence $\varphi_k \vee \varphi_\ell$ must hold. So Theorem 2.2 immediately assures us that PRS will produce a uniform random sink-free orientation with probability 1. But is the expected running time polynomial in n and m? In order to apply Corollary 2.7 we need to bound the ratio $q_{\{k\}}/q_\emptyset$. Although we don't have a handle on $q_{\{k\}}$ and $q_\emptyset$ — and, in a sense, $q_\emptyset$ is a quantity we would like to compute — we can bound the ratio by defining an appropriate mapping from orientations with exactly one sink to those with none.

Introduce a function f from $\{1, \ldots, m\}$ to itself that is consistent with the reference orientation, that is to say, $(v_k, v_{f(k)})$ is a (directed) edge in $\vec{G}$ for all $1 \le k \le m$. This is possible because the reference orientation is sink-free. To each orientation of G that has a single sink at v_k we associate a sink-free orientation as follows. Let e_i be the edge $(v_k, v_{f(k)})$. Reverse the orientation of e_i, i.e., set X_i, which was previously 0, to 1. Vertex v_k is no longer a sink, but $v_{f(k)}$ may have become one. If $v_{f(k)}$ is not a sink than halt. Otherwise reverse the orientation of the edge $(v_{f(k)}, v_{f^2(k)})$, and continue. This process must terminate, as the Hamming distance to the reference orientation is decreasing. In fact, the process traces out a simple path. For suppose not. Let t be the first instant at which we revisit a vertex, i.e., such that $f^t(k) = f^s(k)$ for some $0 \le s < t$. The edge $(v_{f^s(k)}, v_{f^{s+1}(k)})$ is directed away from $v_{f^s(k)}$, and hence vertex $v_{f^s(k)}$ is not a sink, a contradiction. (It is important to note that we leave and revisit vertex v_s via different edges.)

The edges that were flipped in the above construction form a path $v_k = v_{f^0(k)}$, $v_{f^1(k)}, \ldots, v_{f^\ell(k)}$. We may undo the construction provided we know $f^0(k) = k$ and $f^\ell(k)$. It follows that the number of orientations with a single sink exceeds the number of sink free orientations by a factor at most m^2. So by Corollary 2.7 the expected number of iterations in a run of PRS — in this case the number of sinks that are 'popped' — is bounded above by $m^2 = |V(G)|^2$. We may also bound the number times the orientations of individual edges are flipped. Fix a vertex v_k. We saw above how to repair an orientation with a single sink

at v_k. To undo this repair, we just need to specify the index $f^\ell(k)$. Thus the number of orientations with a single sink at v_k exceeds the number of sink-free orientations by a factor m. Referring to (2.6), we have $q_{\{k\}}/q_\emptyset \leq m$ and $a_k = \deg(v_k)$, the degree of vertex v_k. Thus the expected number of edge orientation reversals is $\sum_{k=1}^m q_{\{k\}} a_k/q_\emptyset \leq \sum_{k=1}^m m \deg(v_k) = 2mn$. So the expected number of orientation reversals is a most $2|V(G)|\,|E(G)|$. All this is in agreement with [8].

Surprisingly, the upper bound on edge-reversals can be tightened further to $|E(G)|+|V(G)|^2$: see Guo and He [16]. Note that, the runtime analysis critically used the assumption that coin tosses are unbiased, so that either orientation of an edge is equally likely. (A simple counterexample shows that this assumption is necessary.) In contrast, correctness of the algorithm extends to asymmetric orientation probabilities.

3.2. *Spanning trees of a graph*

The Cycle-popping algorithm is an approach to uniformly sampling spanning trees in a graph, introduced by Propp and Wilson [34]. Suppose G is a connected graph with vertex set $V = \{v_0, v_1, \ldots, v_n\}$ and edge set E. Instead of sampling spanning trees in G we will instead sample spanning (in-)arborescences[2] rooted at v_0, which is of course equivalent.

For each $1 \leq i \leq n$, define $D_i = \{j : \{v_i, v_j\} \in E\}$, and make D_i into a probability space $\mathcal{D}_i$ by equipping it with the uniform distribution. Introduce random variables $X_1, \ldots, X_n$ distributed as $\mathcal{D}_1, \ldots, \mathcal{D}_n$. These variables indicate, for each $1 \leq i \leq n$, a possible exit from vertex v_i. For each simple (oriented) cycle $C = (v_{i_0}, v_{i_1}, \ldots, v_{i_{\ell-1}}, v_{i_\ell} = v_{i_0})$ define the predicate φ_C by

$$\varphi_C = \neg(X_{i_0} = i_1 \wedge X_{i_1} = i_2 \wedge \cdots \wedge X_{i_{\ell-1}} = i_\ell),$$

and the formula Φ by $\Phi = \bigwedge_C \varphi_C$, where the conjunction is over all oriented cycles in G. (In this context, 'simple' is taken to mean 'containing no repeated vertices'; thus we regard the 2-cycle $(v_{i_0}, v_{i_1}, v_{i_0})$ as simple.) The intended interpretation of the event $X_i = j$ is that vertex v_j is the ancestor of vertex v_i in the arborescence. The formula $\Phi(\mathbf{X})$ asserts that the ancestor relation is consistent (has no cycles) and hence that $\mathbf{X}$ encodes a spanning arborescence rooted at v_0.

Consider two clauses φ_C and $\varphi_{C'}$ corresponding to distinct cycles C and C'. If $\mathsf{Scp}(\varphi_C) \cap \mathsf{Scp}(\varphi_{C'}) \neq \emptyset$ then C and C' must have a vertex in common. Select a vertex v_i that is common to C and C' with the additional property that the successor to v_i in cycle C is not equal to the successor to v_i in cycle C'. Let v_j be the successor to v_i in C and $v_{j'}$ be the successor in C'. It is clear that $X_i = j$ and $X_i = j'$ cannot both be true, and hence φ_C and $\varphi_{C'}$ cannot both be false. Therefore Φ is extremal.

As in the previous example, we need to estimate the ratio between aborescences and 'near-arborescences' that contain a single cycle. In this context, a near-arborescence is a unicyclic subgraph of G in which vertex v_0 has outdegree

---

[2]That is, directed spanning trees with edges directed towards a root vertex.

0, and every other vertex has outdegree 1. Necessarily a near-arborescence has two (weakly) connected components: the one containing v_0 is a arborescence rooted at v_0, and the other is unicyclic. As before, by considering a suitable mapping from near-arborescences to arborescences, it can be shown that the number of the former is at most $|V(G)||E(G)|$ times the number of the latter. Thus, by Corollary 2.7, the number of iterations made by PRS is at most $|V(G)||E(G)|$. A more refined analysis, due to Guo and He [16, Thm 15], shows that the total number of *variable* updates is bounded by essentially the same expression.

3.3. *Root-connected subgraphs*

This 'cluster-popping' algorithm was proposed by Gorodezky and Pak [15], who conjectured it to be efficient on a certain class of directed graphs. The conjecture was resolved affirmatively by Guo and Jerrum [17].

Suppose $G = (V, A)$ is a directed graph with a distinguished root vertex $r \in V$. A spanning subgraph (V, S) of G is said to be *root-connected* if, for every vertex $v \in V$, there is a directed path in (V, S) from v to r. Our task is to sample, uniformly at random, a root-connected subgraph of G. As usual, we restrict our attention to the unweighted version. However, as we shall note later, the weighted version is of interest, owing to its connection to a network reliability problem.

A subgraph (V, S) may be encoded by variables $\mathbf{X} = (X_e : e \in A)$ taking values in $\{0, 1\}$. The interpretation of $X_e = 1$ is that $e \in S$. For an arc $e \in A$, denote by e^- and e^+ the start and end vertex of e. A *cluster* in (V, S) is a set $\emptyset \subset C \subseteq V \setminus \{r\}$ of vertices with the property that no edge $e \in S$ exists with $e^- \in C$ and $e^+ \in V \setminus C$. The property 'C is a cluster' can be expressed formally by the predicate $\psi_C = \bigwedge_{e \in \mathrm{cut}(C)} \neg X_e$, where $\mathrm{cut}(C) = \{e \in A : e^- \in C \text{ and } e^+ \in V \setminus C\}$. If the subgraph (V, S) has a cluster C then it is clear that no vertex in C can reach r, via a directed path in (V, S), and hence (V, S) is not root-connected. The converse is also true: Suppose (V, S) is not root-connected, and let v be some vertex from which the root r is not reachable. Let C be the set of all vertices reachable from v. Then C is a cluster in (V, S).

This observation suggests that we should define

$$\Phi = \bigwedge_{\emptyset \subset C \subseteq V \setminus \{r\}} \varphi_C, \tag{3.1}$$

where $\varphi_C = \neg \psi_C$. The formula Φ denies the existence of a cluster in the subgraph encoded by $\mathbf{X}$, and hence correctly expresses the property of being root-connected. The catch is that Φ is not in general extremal. It is perfectly conceivable that two clusters C, C' exist that have nonempty intersection $C \cap C' \neq \emptyset$. In that case, we might have $\mathsf{Scp}(\varphi_C) \cap \mathsf{Scp}(\varphi_{C'}) \neq \emptyset$ and yet φ_C and $\varphi_{C'}$ are both false. The solution is to make the predicates φ_C less demanding, while preserving the semantics of Φ. We say that the cluster C is *minimal* if it contains

no cluster C' with $C' \subset C$. Then we define φ_C to be true if C is not a minimal cluster. Formally,

$$\varphi_C = \neg\Big[\psi_C \wedge \bigwedge_{\emptyset \subset C' \subset C} \neg\psi_{C'}\Big] = \neg\psi_C \vee \bigvee_{\emptyset \subset C' \subset C} \psi_{C'}.$$

Then define Φ as in (3.1). We claim that Φ still expresses the condition that **X** encodes a root-connected subgraph (V, S). If (V, S) is root-connected, then no cluster exists and hence φ_C is satisfied for all $\emptyset \subset C \subseteq V \setminus \{r\}$. Conversely, suppose that (V, S) is not root-connected. Then there is at least one cluster, and hence at least one minimal cluster C. For this cluster, φ_C is contradicted, and hence Φ is false.

Although the meaning of Φ is unchanged, the formula is now extremal. First note that, for all subsets C,

$$\mathsf{Scp}(\varphi_C) = \{X_e : e^- \in C\}.$$

So if φ_C and $\varphi_{C'}$ are any two distinct clauses with $\mathsf{Scp}(\varphi_C) \cap \mathsf{Scp}(\varphi_{C'}) \neq \emptyset$, we must have $C \cap C' \neq \emptyset$. If C and C' are both clusters then $C \cap C'$ must also be a cluster. Therefore, C and C' cannot both be minimal clusters. It follows that at least one of φ_C or $\varphi_{C'}$ must hold. This deals with correctness of PRS in this context.

Unfortunately, PRS does not have expected polynomial runtime on general instances G, as can be appreciated by considering a counterexample presented by Gorodezky and Pak [15]. However, those same authors conjectured that the runtime is polynomial when the graph G is 'bidirected', i.e., an edge exists from vertex u to v in G if and only if an edge exists from v to u. This special case is of interest, since root-connected subgraphs of a bidirected graph G correspond (via a constantly many-one relation) to spanning connected subgraphs of the undirected version of G. (The details of this correspondence, which is a little involved, may be found in [17, §5].) Thus, cluster popping provides a efficient approach to sampling connected spanning subgraphs of a graph.

The conjecture of Gorodezky and Pak may be verified using Corollary 2.7. Again the argument involves a mapping from subgraphs with exactly one minimal cluster to root-connected subgraphs. The combinatorial details of this mapping and its analysis, which are more involved in this case that the previous ones, are given by Guo and Jerrum [17]. The resulting upper bound on the expected number of variable resamplings is $|V(G)|\,|E(G)|^2$, which can be improved to $|V(G)|\,|E(G)|$ by a more refined analysis [16]. For a short while, PRS provided the only known attack on sampling connected spanning subgraphs of a general undirected graph, and its weighted version, undirected all-terminal reliability. However the same problem (in a more general setting) has since been solved by Markov chain simulation by Anari, Liu, Oveis Gharan and Vinzant [2].

3.4. Bases of bicircular matroids

Another application of PRS is to sampling bases of a bicircular matroid. The algorithm was first presented in a slightly different guise by Kassel and Kenyon [26].

Suppose $G = (V, E)$ is an undirected graph with no tree components. The *bicircular matroid* associated with G has E as its ground set. The bases of the matroid are all spanning subgraphs of G in which every connected component is unicyclic; equivalently, every connected component has the same number of edges as it has vertices. The sampling algorithm may be derived methodically using PRS. The application has similarities with the cycle-popping algorithm described above in the context of sampling spanning trees.

As with cycle popping, variables are introduced that encode a function g from V to itself that respects the edges of G. (This is a slight deviation from the spanning trees case, where the function was from $V \setminus \{r\}$ to V.) The spanning subgraph (V, S) defined by $S = \{\{v, g(v)\} : v \in V\}$ is very like a basis of the bicircular matroid, with two caveats. First, we want to rule out cycles of length 2 — that is, situations in which $g(g(v)) = v$ for some $v \in V$ — as such functions g do not correspond to valid bases. Second, each basis with c connected components corresponds to 2^c distinct functions, as each cycle may be traced in either orientation.

To deal with these two objections, we specify a preferred orientation for every cycle in G. Our formula Φ includes a clause φ_C, for each potential cycle C that either (a) has length two, or (b) is oriented in the wrong sense. In each case, φ_C asserts that C does not occur. It is easy to check that Φ is extremal. The expected number of resamplings (either of clauses or individual variables) is $O(|V(G)|^2)$. Details are given by Guo and Jerrum [18].

3.5. Notes

The examples listed above are not the only known applications of PRS, but they are the only non-trivial ones for which polynomial-time running time bounds are known. At least, they are the one ones I am aware of.

One tempting potential application is to sampling strong orientations of an undirected graph. An orientation of the edges of an undirected graph G is *strong* if there is a directed path from every vertex of G to every other. If G is connected, strong orientations coincide with 'totally cyclic orientations'. The number of total cyclic orientations of a graph G is an evaluation of the Tutte polynomial (at the point $(0, 2)$). It is known that counting totally cyclic (and hence strong orientations) is #**P**-complete [25]. However, the computational complexity of approximately counting or uniformly sampling totally cyclic orientations is unknown.

The cluster-popping algorithm for root-connected subgraphs is easily adapted to strong orientations. For a set $\emptyset \subset S \subset V = V(G)$ of vertices of G, we say that S is *cluster* if all edges between S and $V \setminus S$ are directed into S. (The crucial difference with the root-connected case is that there is no distinguished root vertex r that is excluded from all clusters.) We say that a cluster is *minimal* if it is minimal with respect to inclusion. As usual, define $\Phi = \bigwedge_{\emptyset \subset S \subset V} \varphi_S$, where the formula φ_S expresses the condition that S is not a minimal cluster. It may be verified that Φ is extremal, and hence that PRS produces a uniform

random strong orientation (assuming that the G has one, which happens exactly when the graph G is bridgeless). Unfortunately, the expected runtime may be exponential, as can be appreciated by considering the ladder graph L_n on $2n$ vertices. (The ladder graph can be viewed as a $n \times 2$ rectangular piece of the square lattice, or as the cartesian product $P_n \times P_2$ of a path on n vertices and a path on 2 vertices.) If v_k is one of the degree-2 corner vertices then the ratio $q_{\{k\}}/q_\emptyset$ from Theorem 2.6 is exponential in n. (By induction on n, the number of strong orientations is $2 \times 3^{n-2}$, whereas the number of orientations with a unique minimal cluster $\{v_k\}$ is at least $3 \times 4^{n-2}$.) The fact that Theorem 2.6 gives an exact result and not just an upper bound comes in useful here, as it enables us to deduce a *lower* bound on the running time of PRS.

4. Non-extremal instances

In an extremal instance, no two clauses that share variables can be simultaneously false. We have seen that this leads to uniform outputs from PRS. It transpires that we can get away with a little less than this.

Definition 4.1. *We say that the formula* $\Phi = \varphi_1 \wedge \cdots \wedge \varphi_m$ *is* quasi-extremal *if the following holds, for all* $k, \ell \in [m]$ *and assignments* $\mathbf{X}$ *and* $\mathbf{X}'$: *if* $\neg\varphi_k(\mathbf{X}) \wedge \neg\varphi_\ell(\mathbf{X})$ *and it is possible to get from* $\mathbf{X}$ *to* $\mathbf{X}'$ *by resampling variables in the scope of* φ_ℓ, *then* $\neg\varphi_{k'}(\mathbf{X}')$ *for some* k' *with* $\mathsf{Scp}(\varphi_{k'}) \supseteq \mathsf{Scp}(\varphi_k)$.

Note that an extremal instance satisfies the above definition with $k' = k$, so the qualifier 'quasi-extremal' is a weakening of 'extremal'. The additional flexibility allows PRS to be applied to a significantly wider class of examples.

The algorithm is exactly as before except for one change. Although we have considerable flexibility in the order in which to resample (the scopes of) clauses, we no longer have complete freedom.

Algorithm 2 Partial Rejection Sampling with limited nondeterminism

$\mathrm{PRS}(\Phi, \mathcal{D})$ // Φ is a formula on variable set $\mathbf{X}$
Sample $\mathbf{X}$ from the product distribution $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$
while $\neg\Phi(\mathbf{X})$ **do**
    $N := \{\ell : \neg\varphi_\ell(X_{\mathsf{Scp}(\varphi_\ell)})\}$
    Choose $k \in N$ deterministically, *based only on the set N itself*
    Resample all variables in $\mathsf{Scp}(\varphi_k)$
end while

A referee has pointed out that Definition 4.1 has the following consequence for the algorithm. If $\varphi_k(\mathbf{X})$ is false at some point in the execution then (assuming the algorithm terminates) at some point in the future a resampling step must occur at which all variables in the scope of φ_k are simultaneously resampled (possibly together with some others). A weaker version of Definition 4.1 could be formulated along these lines, which would be adequate for correctness of the algorithm. It is not clear if any additional interesting examples would be captured by the weaker notion of quasi-extremality, and we do not pursue this

line here. However, the observation is worth bearing in mind when reading the following correctness proof.

Theorem 4.2. *Suppose Φ is a quasi-extremal satisfiable instance. Then* $\mathrm{PRS}(\Phi, \mathcal{D})$ *terminates with probability 1. On termination,* $\mathbf{X}$ *is a realisation of a random variable from the distribution* $\mathcal{D}_\Phi$.

Proof. Termination with probability 1 can be argued exactly as in the proof of Theorem 2.2.

For correctness, we set up the resampling table as in the proof of Theorem 2.2. As before, fix a resampling table T, and run PRS on T to obtain a transcript. Since the algorithm has terminated, we know that the frontier contains a satisfying assignment. Create a new resampling table T' by replacing the values in the frontier by some other satisfying assignment. Now run the algorithm on the new resampling table T'. We claim that this second run correctly outputs the planted satisfying assignment.

If, in both runs of the algorithm, the same clause φ_k is selected in every iteration then the output indeed will be correct. So assume that in some iteration different clauses are selected in the two runs. For this to occur, the set N must differ between the two runs. Consider the first iteration on which this occurs, and suppose φ_k is true in one run and false in the other. As before, let $S = \mathsf{Scp}(\varphi_k)$ and partition S as $S = S_I \cup S_F$, where variables S_I (respectively, S_F) take values from the interior (respectively, final frontier) of the resampling table. Note that $S_I \neq \emptyset$ (otherwise φ_k would be true in both runs) and $S_F \neq \emptyset$ (otherwise φ_k would have the same truth value in both runs).

There are two cases, both of which lead to a contradiction. Suppose first that φ_k is false in the T-run (and incidentally true in the T'-run, thought this is not relevant to the argument). Allow the T-run to continue. The algorithm does not resample $\mathsf{Scp}(\varphi_k)$ itself, since that action would take it past the final frontier of the table. If it resamples $\mathsf{Scp}(\varphi_\ell)$ for some $\ell \neq k$ then, since Φ is quasi-extremal, this action would leave behind a clause $\varphi_{k'}$ with $\varphi_{k'}$ false and $\mathsf{Scp}(\varphi_{k'}) \supseteq \mathsf{Scp}(\varphi_k)$. Arguing as before, the algorithm does not resample $\mathsf{Scp}(\varphi_{k'})$ so, by induction, at least one clause will always be false for the remainder of the run. Thus, we can never make all clauses of Φ true, which contradicts the fact that the transcript is finite.

The second and final case has φ_k false in the T'-run (and incidentally true in the T-run). Up to this point, the two runs have made exactly the same choices of scopes to resample. Now imagine that all the resampling steps in the T-run are faithfully mirrored in the T' run. We have deviated from the deterministic choice rule of the algorithm, but all resampling steps are legal, in the sense that we always resample scopes $\mathsf{Scp}(\varphi_\ell)$ for which φ_ℓ is currently false. The reason for this is exactly as in the proof of Theorem 2.2: briefly, that we never resample variables in the frontier and the variables sampled from the interior have the same values in both runs. The T'-run finishes with the same transcript as the T-run, but with a different satisfying assignment in the frontier. In particular all clauses of Φ are satisfied. On the other hand, we may argue as follows. Since $\mathsf{Scp}(\varphi_k)$ is never resampled again in the T-run, it is also never resampled in the

T'-run. Also, as in the first case, by resampling $\mathsf{Scp}(\varphi_\ell)$ for ℓ with $\ell \neq k$, we must leave behind a clause $\varphi_{k'}$ with $\varphi_{k'}$ false and $\mathsf{Scp}(\varphi_{k'}) \supseteq \mathsf{Scp}(\varphi_k)$. Arguing again by induction, we can never make all clauses of Φ true, which is a contradiction.

Summarising, the final transcript remains unchanged if one satisfying assignment is substituted for another in the final frontier. Thus, conditioned on the transcript, each satisfying assignment $\mathbf{X} = (b_1, \ldots, b_n)$ of Φ occurs with probability proportional to $\mathcal{D}_1(b_1)\mathcal{D}_2(b_2)\cdots\mathcal{D}_n(b_n)$. So, at termination, $\mathbf{X}$ is distributed as $\mathcal{D}_\Phi$. □

It will be seen from the proof that a somewhat more liberal non-determinism could be allowed in Algorithm 2,. Basing the choice on the set N strikes a balance between simplicity and generality.

4.1. Example: independent sets (the hard-core gas model)

Suppose wish to sample independent sets in a graph G. Introduce variables $\mathbf{X} = (X_v : v \in V(G))$ taking values in $\{0, 1\}$ to encode potential independent sets in G. The interpretation of $X_v = 1$ (respectively $X_v = 0$) is that vertex v is in (respectively not in) the independent set. In our product distribution we assume $\Pr(X_v = 1) = \lambda/(1 + \lambda)$ for all vertices of G, for some positive 'activity' λ. (It is not essential that the activity is constant over vertices, but it slightly simplifies the exposition.) We wish to sample from the conditional (Gibbs) distribution given that $\mathbf{X}$ encodes an independent set.

The natural formula expressing that $\mathbf{X}$ encodes an independent set is $\Phi'(\mathbf{X}) = \bigwedge_{\{u,v\} \in E(G)} \varphi_{\{u,v\}}$ where $\varphi_{\{u,v\}} = \neg(X_u \wedge X_v)$. However Φ' is not extremal. Following the example provided by cluster popping for root connected subgraphs, we try to re-express Φ' as a semantically equivalent extremal formula.

Suppose $S \subseteq V$ is a maximal set of vertices such that (i) the induced subgraph $G[S]$ is connected and (ii) $X_v = 1$ for all $v \in S$. If S is not a singleton, we say that S is a *cluster* (relative to the assignment $\mathbf{X}$). Denote by

$$\partial S = \{v : v \notin S \text{ and } \exists u \in S \text{ such that } \{u, v\} \in E(G)\}$$

the boundary of S, containing all vertices outside of S that are adjacent to some vertex in S. Naturally, $X_v = 0$ for all $v \in \partial S$. Refer to Figure 5, where solid (respectively, open) vertices v are ones where $X_v = 1$ (respectively, $X_v = 0$). For each S of the above form, we introduce a clause φ_S that asserts that S is not a cluster. In Figure 5, there are two clusters, and in each case the extent of $S \cup \partial S$ (which is also the scope of φ_S) is indicated. Let $\Phi = \bigwedge_S \varphi_S$, where S ranges over all vertex subsets of size at least two that induce a connected subgraph. It is clear that $\Phi(\mathbf{X})$ asserts that $\mathbf{X}$ encodes an independent set.

Unfortunately, a moment's reflection reveals that Φ is also not extremal. Denote by $\overline{S}$ the set $\overline{S} = S \cup \partial S$ and note that $\mathsf{Scp}(\varphi_S) = \overline{S}$. It is possible to have clusters S and S' with $\overline{S} \cap \overline{S'} \neq \emptyset$, in which case $\mathsf{Scp}(\varphi_S) \cap \mathsf{Scp}(\varphi_{S'}) \neq \emptyset$, and yet $\varphi_S \vee \varphi_{S'}$ is false. Indeed, the two clusters in Figure 5 are of this kind.
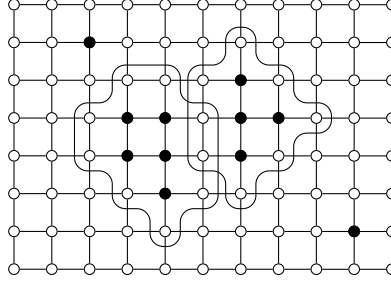
FIG 5. *A pair of clusters in $\mathbb{Z}^2$ with overlapping boundaries*

However, it is straightforward to verify that Φ is quasi-extremal. Suppose φ_S and $\varphi_{S'}$ are simultaneously false. It is easy to see that

$$\overline{S} \cap S' = (S \cup \partial S) \cap S' = S \cap S' = \emptyset,$$

and similarly that $S \cap \overline{S'} = \emptyset$. If $\overline{S} \cap \overline{S'} = \emptyset$ then $\mathsf{Scp}(\varphi_S) \cap \mathsf{Scp}(\varphi_{S'}) = \emptyset$, and Definition 4.1 is satisfied with $k' = k$. Otherwise, we are in the case

$$\overline{S} \cap \overline{S'} = (S \cup \partial S) \cap (S' \cup \partial S') = \partial S \cap \partial S' \neq \emptyset.$$

Resampling $\mathsf{Scp}(\varphi_{S'})$ can make φ_S true, but only at the expense of making some $\varphi_{S''}$ with $S'' \supset S$ false, since no variable in S is resampled. Again Definition 4.1 is satisfied, but now with $k' \neq k$. In Figure 5, resampling the left cluster (with boundary) may increase the right cluster but cannot decrease it.

Specialising the generic PRS algorithm to this example, we obtain the following algorithm for sampling independent sets, which is a slight variant of one first described by Guo, Jerrum and Liu [20].

Algorithm 3 Partial Rejection Sampling for independent sets

PRSforIS(G, λ)   // G is an undirected graph, and λ a positive real number
Sample **X** from the product, over all variables, of the distribution Bernoulli($\lambda/(1 + \lambda)$).
while X does not encode an independent set **do**
   Choose a cluster S using a valid rule
   Resample all variables $\{X_v : v \in \overline{S}\}$
end while

Lemma 4.3. PRSforIS(G, λ) *terminates with probability 1. On termination,* **X** *is a realisation of a random variable from the Gibbs distribution for independent sets in G with activity λ.*

Proof. Follows immediately from Theorem 4.2. □

4.2. Runtime analysis

Sampling independent sets is in general an NP-hard problem [30, Thm 4], so we need to make some assumption about the graph G and activity λ. Our goal is to

find $\lambda_\Delta > 0$ such that PRS terminates rapidly, for all $\lambda < \lambda_\Delta$ and all graphs G of maximum degree Δ.

We take as our starting point the runtime analysis for extremal instances from Section 2. One problem extending this analysis to the non-extremal situation is that the proof of Theorem 2.6 fails. The reason for this is that the interpretation of $q_\emptyset$ as the probability $\Pr_\mathcal{D}(\Phi)$ that Φ is satisfied is no longer valid. It transpires that this problem can be avoided by using a different line of proof. Kolipaka and Szegedy [28, Thm 4] show that the number of resamplings of the scope of φ_k is bounded above by $q_{\{k\}}/q_\emptyset$, provided the point $(p_1, \ldots, p_m)$ lies within a certain region. (Refer to the preamble to Theorem 2.6 for notation.) This region was identified by Shearer [36] as the theoretical limit of validity of the Lovász Local Lemma, even in the non-algorithmic setting. Although elegant, it is difficult to use this result directly: testing membership in the Shearer region in specific examples is challenging, as is computing $q_{\{k\}}$ and $q_\emptyset$, which no longer have simple combinatorial interpretations. Fortunately, there are several weaker conditions that can be feasibly tested.

Just as we weakened the definition of extremal to quasi-extremal, we can weaken the concept to dependency graph or relation to a lopsided dependency ('lopsidependency') graph [32, §6].

Definition 4.4. *Given a satisfiable instance* $\Phi = \varphi_1 \wedge \cdots \wedge \varphi_m$, *let* $k, \ell \in [m]$ *with* $k \neq \ell$ *be arbitrary. Suppose there is a resampling table relative to which it is possible to resample* $\mathsf{Scp}(\varphi_k)$ *and then immediately resample* $\mathsf{Scp}(\varphi_\ell)$ *but it is not possible to perform these operations in the reverse order (either because* φ_ℓ *is true initially, or because* φ_k *is true after* $\mathsf{Scp}(\varphi_\ell)$ *has been resampled). Then we write* $k \sim \ell$ *and say that* k *and* ℓ *are* lopsidedly dependent. *The graph* $([m], \sim)$ *is the* lopsided dependency graph *of* Φ. *(We regard the pair* k, ℓ *as unordered, and the graph as undirected.)*

Note also that the lopsided dependency graph is a subgraph, in general strict, of the usual dependency graph.

In the independent set example it is easy to characterise the lopsided dependency graph: specifically, $S \nsim S'$ iff $S \cap \overline{S'} = \emptyset$ and $\overline{S} \cap S' = \emptyset$. (In fact, the two parts of thie latter condition are logically equivalent, so only one needs to be tested.) To see this, consider two clauses φ_S and $\varphi_{S'}$ with $S \cap \overline{S'} = \emptyset$. Suppose it is possible to resample $\mathsf{Scp}(\varphi_S)$ followed by $\mathsf{Scp}(\varphi_{S'})$. We claim that the order of the resamplings can be reversed (leading potentially to a locally different transcript). The case $\mathsf{Scp}(\varphi_S) \cap \mathsf{Scp}(\varphi_{S'}) = \overline{S} \cap \overline{S'} = \emptyset$ is uninteresting. So consider a variable X_i with $i \in \overline{S} \cap \overline{S'}$. Necessarily, $i \in \partial S \cap \partial S'$. It follows that X_i takes the value 0 before $\mathsf{Scp}(\varphi_S)$ is resampled (since $\neg \varphi_S$) and retains that value after (since $\neg \varphi_{S'}$). Therefore the two scopes could as well have been resampled in the opposite order, and $S \nsim S'$. Conversely, if $S \cap \overline{S'} \neq \emptyset$ then at most one of φ_S or $\varphi_{S'}$ can be false initially, so the ordering of resampling is forced, and $S \sim S'$. In Figure 5, the two clusters are related in the dependency graph but not in the lopsided dependency graph: only the boundaries intersect.

Definition 4.4 is sometimes portrayed as as a positive dependency condition, but in the resampling table view of the world it seems more natural to interpret

it as a commutativity condition. We say that a clause is *atomic* if it is falsified by exactly one assignment. Definition 4.4 takes a simpler form when all clauses are atomic; see Moser and Tardos [32, §6], where 'elementary' is used as a synonym for 'atomic'.

Observation 4.5. *Suppose φ_k is atomic, for every $k \in [m]$. Then $k \sim \ell$, i.e., k and ℓ are lopsidedly dependent, iff $\varphi_k \vee \varphi_\ell$ is a tautology.*

Moser and Tardos [32, Thm 6.1] prove the following runtime bound.

Theorem 4.6. *Suppose that Φ is a satisfiable instance (formula) with lopsided dependency graph $([m], \sim)$. Suppose also that there exists a sequence of reals $(x_k \in (0,1) : k \in [m])$ such that, for all $k \in [m]$,*

$$\Pr_{\mathcal{D}}(\neg\varphi_k) \leq x_k \prod_{\ell \in [m]:\ell \sim k} (1 - x_\ell).$$

Then, in expectation, Algorithm 2 resamples $\mathsf{Scp}(\varphi_k)$ at most $x_k/(1 - x_k)$ times before halting.

Lemma 4.7. *Suppose G is a graph with n vertices and maximum degree Δ. There exists $\lambda_\Delta > 0$ such that the expected number of variable resamplings made during the execution of $\mathrm{PRSforIS}(G, \lambda)$ is at most $2n$ whenever $\lambda \leq \lambda_\Delta$. Asymptotically, $\lambda_\Delta = \Theta(\Delta^{-1})$.*

Proof. Identifying vertices of G with $[n]$, let

$$\mathcal{C} = \big\{ S \subseteq [n] : |S| \geq 2 \text{ and } G[S] \text{ is connected} \big\}$$

be the set of all subsets of $V(G)$ that induce connected subgraphs of G with at least two vertices. We need to find quantities $\{x_S : S \in \mathcal{C}\}$ satisfying

$$\Pr_{\mathcal{D},\lambda}(\neg\varphi_S) \leq x_S \prod_{S' \in \mathcal{C}:S' \sim S} (1 - x_{S'}). \tag{4.1}$$

Note that we have included the activity λ explicitly in the notation here, as we are about to introduce a second artificial activity μ. We define the required quantities x_S by $x_S = \Pr_{\mathcal{D},\mu}(\neg\varphi_S)$ for some suitably chosen μ (one that will make the right hand side of inequality (4.1) large), and then choose $\lambda < \mu$ as large as possible while still satisfying the inequality. The thinking here is that as S varies, x_S should shadow $\Pr_{\mathcal{D},\lambda}(\neg\varphi_S)$, but with enough slack to allow inequality (4.1) to be satisfied. For convenience, let $q = \mu/(1+\mu)$. We start with a preliminary calculation. For any $i \in [n]$,

$$\begin{aligned}
\sum_{S' \in \mathcal{C}:i \in S'} x_{S'} &= \sum_{S' \in \mathcal{C}:i \in S'} \Pr_{\mathcal{D},\mu}(\neg\varphi_{S'}) \\
&= \Pr_{\mathcal{D},\mu}\left(\bigvee_{S' \in \mathcal{C}:i \in S'} \neg\varphi_{S'} \right) \tag{4.2} \\
&= \Pr_{\mathcal{D},\mu}(i \text{ is contained in some cluster})
\end{aligned}$$

$$\leq \Delta q^2. \tag{4.3}$$

Equality (4.2) follows from disjointness of the events $\neg\varphi_{S'}$ over all $S' \in \mathcal{C}$ with $i \in S'$ (by lopsided dependency). Inequality (4.3) is an upper bound on the probability that vertex i finds itself in a cluster, obtained as a union bound over the (at most) Δ events that i and some particular neighbour of i are selected.

Now suppose that $S \in \mathcal{C}$ and let $c = |S|$ and $b = |\partial S|$. Then

$$x_S \prod_{S' \sim S} (1 - x_{S'}) \geq x_S \prod_{i \in S \cup \partial S} \prod_{S' \in \mathcal{C}: i \in S'} (1 - x_{S'}) \qquad \text{(by over-counting)} \tag{4.4}$$

$$\geq x_S \prod_{i \in S \cup \partial S} \left(1 - \sum_{S' \in \mathcal{C}: i \in S'} x_{S'}\right)$$

$$\geq q^c(1-q)^b(1 - \Delta q^2)^{b+c} \qquad \text{(by (4.3))}. \tag{4.5}$$

This deals with the right hand side of (4.1). The left hand side is simply

$$\mathrm{Pr}_{\mathcal{D},\lambda}(\neg\varphi_S) = p^c(1-p)^b, \tag{4.6}$$

where p stands for $\lambda/(1+\lambda)$. Recall that we want to ensure that (4.6) is less than or equal to (4.5). Since $q > p$, this goal is hardest to achieve, for any given c, when b is as large as possible. Certainly $b \leq (\Delta-1)c$, so we assume $b = (\Delta-1)c$ from now on. With this simplification, the inequality we wish to satisfy is

$$p^c(1-p)^{(\Delta-1)c} \leq q^c(1-q)^{(\Delta-1)c}(1 - \Delta q^2)^{\Delta c},$$

or, equivalently,

$$p(1-p)^{(\Delta-1)} \leq q(1-q)^{(\Delta-1)}(1 - \Delta q^2)^{\Delta}. \tag{4.7}$$

We are free to choose q as we like. If we let $q = \Delta^{-1}$ then the right hand side is greater than $1/(8\Delta)$, enabling us to take p at most $1/(8\Delta)$ and λ at most $1/(8\Delta - 1)$.

Having shown that the premise of Theorem 4.6 holds, we can read off an upper bound on the expected number of resamplings. Repeating an earlier trick,

$$\sum_{S \in \mathcal{C}} x_S |S| = \sum_{i \in V(G)} \sum_{S \in \mathcal{C}: i \in S} x_S$$

$$\leq n\Delta q^2 \qquad \text{(by (4.3))}$$

$$= n/\Delta.$$

Noting that $|\partial S| \leq (\Delta - 1)|S|$ and $x_S \leq \frac{1}{2}$ we have that the expected total number of variable resamplings is

$$\sum_{S \in \mathcal{C}} \frac{x_S}{1 - x_S} |S \cup \partial S| \leq \sum_{S \in \mathcal{C}} 2\Delta x_S |S| \leq 2n,$$

by Theorem 4.6. □

Remark. *Of course, Δ^{-1} was merely a convenient choice for q and not an optimal one. When $\Delta = 3$, we find numerically that the right hand side of (4.7) attains a maximum of $0.0892275+$ at around $q = 0.172016$. Thus, we can satisfy inequality (4.7) by setting $p = 0.113551$, which is ensured by taking $\lambda_3 = 0.128$. When Δ is large, a similar line of argument gives $\lambda_\Delta \sim C/\Delta$ asymptotically, where $x = C = 0.327+$ is the smallest solution to $xe^{-x} = \frac{1}{2}e^{-3/4}$.*

Our calculation has some slack at a number of locations. One easy win is to replace Δq^2 in (4.3) by the tighter, in fact exact, $q(1 - (1 - q)^\Delta)$. Another is achieved by reducing the degree of overcounting in (4.4). Although we do need to let i range over the whole of ∂S, it is enough to let it range over a minimum vertex cover of (the subgraph induced by) S. However, these improvements, and others that may suggest themselves, seem to yield only marginal improvements in the achievable lambda, and do not justify the increase in combinatorial complexity. It seems that some significant new ingredient would be required to achieve an outcome competitive with other approaches to perfectly sampling independent sets.

To provide some context for the above working, we review the hard-core model on an infinite regular tree of degree Δ. It is known that this model exhibits a phase transition at $\lambda_c = (\Delta - 1)^{\Delta-1}/(\Delta - 2)^\Delta$ [5, §4]. For $\lambda < \lambda_c$ there is a unique Gibbs measure and for $\lambda > \lambda_c$ there are two. A remarkable discovery is that that λ_c also marks a computational threshold of the hard-core model. On the one hand, Sly and Sun [37] and Galanis, Štefankovič and Vigoda [14] showed that it is NP-hard to sample, even approximately, from the hard-core distribution in general graphs of maximum degree Δ, when $\lambda > \lambda_c$. One the other hand, for the same class of graphs, approximate sampling is possible in time $O(n \log n)$ when $\lambda < \lambda_c$. This was shown by Chen, Liu and Vigoda [7], building on the spectral independence approach of Anari, Liu and Oveis-Gharan [3].

Against this benchmark, the performance of PRS is unimpressive. The asymptotic upper bound on λ of $0.327/\Delta$ derived above is beaten by various other approaches to perfect sampling: $1/\Delta$ using the lazy depth-first sampler of Anand and Jerrum [1, §3.1], $\frac{4}{3}/\Delta$ using the randomness recycler of Fill and Huber [13], and $2/\Delta$ using coupling from the past allied to bounding chains by Huber [23]. The last of these is an $O(n \log n)$ time (as opposed to linear time) perfect sampler. All of these bounds are below the uniqueness threshold λ_c, and it would be interesting to know whether the limit for linear time perfect sampling is λ_c, or whether there is a barrier below this.

Extensive research on the algorithmic Lovász Local Lemma has brought to light a number of alternatives to Theorem 4.2. Examples which may be useful in analysing PRS algorithms have been given by Bissacot, Fernández, Procacci, and Scoppola [4], Harris [21], Harvey and Vondrak [22], Iliopoulos [24], Kolmogorov [29], and Pegden [33]. A comprehensive treatment of the circle of ideas surrounding the independent set polynomial and the Lovász Local Lemma has been given by Scott and Sokal [35].

5. Generalisations

In this article, we have restricted attention to the simplest version of PRS based directly on the Moser Tardos algorithmic LLL. Specifically, we resample the variables of just one clause at each step. This involved recasting the 'obvious' encoding of a problem as a CNF formula in a form suitable for application of the method. For example, in the case of independent sets, we replaced the natural two-variable clauses by larger clauses based on clusters. Alternatively, it is possible to stick with the 'natural' formula at the expense of complicating the resampling algorithm. This was the approach originally taken by Guo, Jerrum and Liu [20].

We dealt here exclusively with hard constraints which either permit or deny a particular assignment to the variables. Soft constraints can be incorporated by introducing an auxiliary variable taking values in the real interval $[0, 1]$. Applied to the Ising model, for example, one would end up with a representation akin to that of Edwards and Sokal [9]. Alternatively, Feng, Vishnoi and Yin [12] incorporated soft constraints directly, thereby allowing a wider range of spin systems to be addressed more naturally. Another possible extension is to continuous state spaces, with Guo and Jerrum [19] (see also Wellens [39]) treating the hard-disks model, and Moka and Kroese [31] more general point processes. Feng, Guo and Yin [11] show how to achieve perfect sampling when strong spatial mixing holds. This last work is quite far from PRS, but still relies on growing a sample by repeatedly 'repairing' parts of the current configuration.

Acknowledgments

References

[1] Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *SIAM J. Comput.*, 51(4):1280–1295, 2022. MR4456711

[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. *Ann. of Math. (2)*, 199(1):259–299, 2024. MR4681146

[3] Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science*, pages 1319–1330. IEEE Computer Soc., Los Alamitos, CA, [2020] ©2020. MR4232133

[4] Rodrigo Bissacot, Roberto Fernández, Aldo Procacci, and Benedetto Scoppola. An improvement of the Lovász local lemma via cluster expansion. *Combin. Probab. Comput.*, 20(5):709–719, 2011. MR2825585

[5] Graham R. Brightwell and Peter Winkler. Graph homomorphisms and phase transitions. *J. Combin. Theory Ser. B*, 77(2):221–262, 1999. MR1719348

[6] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements.* Lecture Notes in Mathematics, No. 85. Springer-Verlag, Berlin-New York, 1969. MR0239978

[7] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of Glauber dynamics: entropy factorization via high-dimensional expansion. In *STOC'21—Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1537–1550. ACM, New York, [2021] ©2021. MR4398939

[8] Henry Cohn, Robin Pemantle, and James G. Propp. Generating a random sink-free orientation in quadratic time. *Electr. J. Comb.*, 9(1), 2002. MR1946148

[9] Robert G. Edwards and Alan D. Sokal. Generalization of the Fortuin-Kasteleyn-Swendsen-Wang representation and Monte Carlo algorithm. *Phys. Rev. D (3)*, 38(6):2009–2012, 1988. MR0965465

[10] Kimmo Eriksson. Strong convergence and a game of numbers. *European J. Combin.*, 17(4):379–390, 1996. MR1387469

[11] Weiming Feng, Heng Guo, and Yitong Yin. Perfect sampling from spatial mixing. *Random Structures Algorithms*, 61(4):678–709, 2022. MR4504840

[12] Weiming Feng, Nisheeth K. Vishnoi, and Yitong Yin. Dynamic sampling from graphical models. *SIAM J. Comput.*, 50(2):350–381, 2021. MR4235179

[13] James Allen Fill and Mark Huber. The randomness recycler: a new technique for perfect sampling. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 503–511. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000. MR1931847

[14] Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *Combin. Probab. Comput.*, 25(4):500–559, 2016. MR3506425

[15] Igor Gorodezky and Igor Pak. Generalized loop-erased random walks and approximate reachability. *Random Structures Algorithms*, 44(2):201–223, 2014. MR3158629

[16] Heng Guo and Kun He. Tight bounds for popping algorithms. *Random Structures Algorithms*, 57(2):371–392, 2020. MR4129725

[17] Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019. MR3948248

[18] Heng Guo and Mark Jerrum. Approximately counting bases of bicircular matroids. *Combin. Probab. Comput.*, 30(1):124–135, 2021. MR4205662

[19] Heng Guo and Mark Jerrum. Perfect simulation of the hard disks model by partial rejection sampling. *Ann. Inst. Henri Poincaré D*, 8(2):159–177, 2021. MR4261668

[20] Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *J. ACM*, 66(3):Art. 18, 31, 2019. MR3941342

[21] David G. Harris. Lopsidependency in the Moser-Tardos framework: beyond the lopsided Lovász local lemma. *ACM Trans. Algorithms*, 13(1):Art. 17, 26, 2016. MR3598122

[22] Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. *SIAM J. Comput.*, 49(2):394–428, 2020. MR4082011

[23] Mark Huber. Perfect sampling using bounding chains. *Ann. Appl. Probab.*, 14(2):734–753, 2004. MR2052900

[24] Fotis Iliopoulos. Commutative algorithms approximate the LLL-distribution. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, volume 116 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 44, 20. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018. MR3857282

[25] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Cambridge Philos. Soc.*, 108(1):35–53, 1990. MR1049758

[26] Adrien Kassel and Richard Kenyon. Random curves on surfaces induced from the Laplacian determinant. *Ann. Probab.*, 45(2):932–964, 2017. MR3630290

[27] Donald E. Knuth. *The Art of Computer Programming*, volume 4, Fascicle 6. Addison-Wesley Professional, 2015. MR2251595

[28] Kashyap Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 235–243. ACM, New York, 2011. MR2931973

[29] Vladimir Kolmogorov. Commutativity in the algorithmic Lovász local lemma. *SIAM J. Comput.*, 47(6):2029–2056, 2018. MR3874020

[30] Michael Luby and Eric Vigoda. Fast convergence of the Glauber dynamics for sampling independent sets. *Random Structures Algorithms*, 15(3-4):229–241, 1999. MR1716763

[31] Sarat B. Moka and Dirk P. Kroese. Perfect sampling for Gibbs point processes using partial rejection sampling. *Bernoulli*, 26(3):2082–2104, 2020. MR4091102

[32] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász Local Lemma. *J. ACM*, 57(2), 2010. MR2606086

[33] Wesley Pegden. An extension of the Moser-Tardos algorithmic local lemma. *SIAM J. Discrete Math.*, 28(2):911–917, 2014. MR3215059

[34] 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. MR1622393

[35] Alexander D. Scott and Alan D. Sokal. The repulsive lattice gas, the independent-set polynomial, and the Lovász local lemma. *J. Stat. Phys.*, 118(5-6):1151–1261, 2005. MR2130890

[36] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985. MR0837067

[37] Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Ann. Probab.*, 42(6):2383–2416, 2014. MR3265170

[38] Gérard Xavier Viennot. Heaps of pieces. I. Basic definitions and combinatorial lemmas. In *Combinatoire énumérative (Montreal, Que., 1985/Quebec, Que., 1985)*, volume 1234 of *Lecture Notes in Math.*, pages 321–350. Springer, Berlin, 1986. MR0927773

[39] Jake Wellens. A note on partial rejection sampling for the hard disks model in the plane, 2018.