Uniform Generation of w-free Strings and H-free Subgraphs with Partial Rejection Sampling

SPUR Final Paper, Summer 2018

Juan Gil and Joshua Amaniampong

Mentor: Jake Wellens

Project suggested by: Jake Wellens

August 1, 2018

Abstract

Recently, Guo, Jerrum, and Liu introduced a simple and powerful general purpose algorithm for exact uniform sampling, which they called *partial rejection sampling* (PRS). The technique has applications to a variety of sampling problems, including random spanning trees, sink-free orientations of a graph, satisfying assignments of certain CNFs, and point configurations from the so-called Hard Disks model. It was also used to give an FPRAS for the all-terminal network reliability problem. PRS always produces uniform samples by design, but in general, it can be quite inefficient – in many cases, it simply degenerates to *rejection sampling*. In this paper, we prove that PRS can be used to efficiently sample from two types of spaces. The first space is the set of strings $s \in \Sigma^n$ avoiding some contiguous substring w. The second is the set of (non-induced) subgraphs of various grids avoiding a certain subgraph H. In each case, we exploit the geometry of the underlying dependency graphs to prove the efficiency of PRS in parameter regimes that are much larger than those obtained by simply applying the original analysis of Guo et al. to these problems.

1 Introduction

Given a set of "bad" events $A_1, \ldots, A_m$ depending on variables $X_1, \ldots, X_n$, how can we sample a *uniformly random* assignment σ to the variables subject to the constraint that

$$\sigma \in \bigcap_{i=1}^{m} \overline{A_i} \ ?$$

The most naive approach would be to repeatedly sample an assignment σ uniformly at random until, by a stroke of luck, $\sigma \in \cap_{i=1}^{m} \overline{A_i}$. While this algorithm is attractively simple and always outputs a sample from the desired distribution upon termination, it takes an often intractably large $\frac{1}{\Pr[\cap_{i=1}^{m} \overline{A_i}]}$ rounds in expectation. Is there a general-purpose way to make rejection sampling more efficient? Enter *partial rejection sampling.*

Introduced by Guo, Jerrum and Liu in 2017, partial rejection sampling (PRS) is a general algorithm for uniform sampling which captures (at least partially) the simplicity of rejection sampling, without necessarily taking an enormous number of rounds to terminate when $\Pr[\cap_{i=1}^{m} \overline{A_i}]$ is small.

The basic idea behind partial rejection sampling is simple: instead of resampling all of the variables every time at least one bad event occurs, we only sample a subset of them. Suppose σ is some assignment to the variables $X_1, \ldots, X_n$. If $\mathsf{Bad}(\sigma)$ is the set of bad events which occur under this assignment, and each A_i depends on a subset $\mathsf{var}(A_i)$ of variables, then we will certainly want to resample all of the variables in $\cup_{i \in \mathsf{Bad}(\sigma)} \mathsf{var}(A_i)$, so that every one of these occurring events has a chance of being avoided in the next sample. (This is in fact what Moser and Tardos's celebrated constructive proof (7) of the Lovasz Local Lemma does).

There is an important class of instances for which this algorithm already outputs a uniform sample, i.e. resampling $\cup_{i \in \mathsf{Bad}} \mathsf{var}(A_i)$ is already sufficient to guarantee uniformity upon termination. These are called *extremal* instances, and are characterized by the property that *dependent bad events are disjoint.* In other words, the set of occurring bad events is always an independent set inside the dependency graph. The classic "sink-popping" algorithm of Cohn, Pemantle and Propp (1) – which generates a uniformly random sink-free orientation of an undirected graph by randomly reorienting all edges incident to a sink– fits into this framework[1], and is in fact where the term "partial rejection sampling" was first used. We'll discuss other examples of extremal partial rejection sampling in depth in Section 2 of this paper.

In general, however, to obtain a *uniform* sample avoiding the bad events, one has to resample a larger set of variables. Guo et al. provided one method of doing this, which is the one we discuss in Section 3. As shown in (3), this method of selecting the resampling set (Res) yields a sampling algorithm which is, generally speaking, efficient when any dependent pair of bad events *share many of their variables.* As an example, the authors of (3) use PRS to sample satisfying assignments of k-CNFs with the property that no variable occurs in more than $\approx 2^{k/2}$ clauses, and every pair of dependent clauses *share* at least $k/2$ variables. The problems we consider in this paper do *not* have this large-overlap feature, and so in proving

[1] No edge can point from one sink to another sink, and so the sinks form an independent set in the input graph, which is also the dependency graph for this problem.

efficiency of PRS we need to exploit different properties of the underlying dependency graphs, tailored to these specific examples.

The following theorem, proved in (3), gives a set of sufficient conditions under which PRS runs efficiently. To state their result we need to introduce a bit of notation. Let $p = \max_i \Pr[A_i]$ be the maximum probability of a bad event, and let R_{ij} be the event that a random assignment on $\mathsf{var}(A_i) \cap \mathsf{var}(A_j)$ can be extended to an assignment satisfying A_j. Set $r = \max_{i,j} \Pr[R_{ij}]$.

Theorem 1. *((3), Theorem 26): Let m be the number of bad events, n the number of variables, and Δ the maximum degree in the dependency graph. For any $\Delta \geq 2$, if $6ep\Delta^2 \leq 1$ and $3er\Delta \leq 1$, then the expected number of rounds used by PRS is $O(\log m)$ and the expected number of resampled events is at most $O(m)$.*

In this paper, we apply and analyze the performance of partial rejection sampling on two types of problems within the variable framework:

- **Sampling w-free strings:** Given input (Σ, w, n), where Σ is a finite alphabet and w is some string over Σ, generate a uniformly random element of Σ^n which does not contain w as a contiguous substring.
- **Sampling H-free subgraphs:** Given an input (G, H, λ), where G and H are graphs and $\lambda \in (0, 1)$, generate a sample from the following distribution, which is supported on H-free (non-induced) subgraphs of G:

$$\Pr(G') \propto \begin{cases} \lambda^{e(G')}(1-\lambda)^{e(G)-e(G')} \propto \left(\frac{\lambda}{1-\lambda}\right)^{e(G')} & \text{if } G' \subseteq G \text{ is } H\text{-free} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

  In this paper we only deal explicitly with the case that G is a subgraph of either the triangular or the square grid, and H is either a triangle or a square, respectively.

We quickly survey what is known about each of these problems, and finally state our results.

1.1 w-free string sampling

There is a somewhat obvious algorithm which solves this problem exactly: for each character $a \in \Sigma$, use generating functions and/or recursive formulas to compute the number of w-free strings whose first bit is a. Sample the first bit according to this distribution and proceed recursively. While this is technically an efficient algorithm, it is somewhat impractical and has a number of undesirable features. For one, this procedure becomes more complicated as $|\Sigma|$ and $|w|$ grow large – indeed, the most natural implementations involve computing $\Omega(|\Sigma|^{|w|})$ generating functions in general, and/or finding and the roots of various polynomial equations. Such algorithms are also not robust to slight changes in the problem structure – for example, if the string is wrapped back on itself into a circle or a figure 8, or more complicated shapes, the combinatorics of exact counting can quickly become impossible. Approximate sampling via Monte-Carlo-Markov-Chain methods is also possible, although we do not know if this can be turned into an exact sampler (e.g. via coupling from the past).

In the notation of Theorem 1, the general case of the w-free string sampling problem has $p = |\Sigma|^{|w|}$, $\Delta = 2(|w| - 1)$, and $r = \frac{1}{|\Sigma|}$. Appealing directly to Theorem 1 would require

$$|\Sigma| > 16|w|,$$

which is rather undesirable. We show that for $|\Sigma| \geq 3$ and *any* string w, this condition is not necessary.

Theorem 2. *For $|\Sigma| \geq 3$ and any w, the expected number of rounds used by PRS to sample a w-free string in Σ^n is $O(\log n)$ and the expected number of resampled events is $O(n)$.*

It will be clear from the proof that Theorem 2 also applies to circular strings (i.e. necklaces), and can be adapted to more complicated structures without much loss (one only needs to add the assumption $|w| \geq C$, for some C depending on the number of self intersections.) As we will show in Section 2, the condition $|\Sigma| \geq 3$ in Theorem 2 cannot be removed. However, as a corollary of our proof of this theorem, we are able to show that PRS works for w-free string sampling over binary alphabets for w satisfying certain conditions (see Corollary 17).

1.2 H-free subgraph sampling

In contrast to string sampling, the exact counting problem for subgraph sampling seems quite hard, even in subgraphs of a grid. However, when G and H have the property that no edge of G appears in more than 2 copies of H inside G, then there is an FPRAS for *approximately* sampling from this distribution when $\lambda \leq 1/2$, based on an algorithm of Lin, Liu and Lu (6). Both the square and triangular grid versions of the problem we consider in this paper satisfy this property. However, even on these grids the FPRAS can be quite inefficient – the runtime bound in (6) becomes $O(n^{11.16}(1/\epsilon)^{2.58})$ when applied to an $n \times n$ grid.

Again we can try appealing directly to Theorem 1 to see how PRS performs. In the case of sampling triangle free subgraphs of the triangular grid, the condition $3er\Delta \leq 1$ becomes

$$\lambda \leq \frac{1}{9e} \approx 0.04088$$

while for square-free subgraphs of the square grid, it becomes

$$\lambda \leq \frac{1}{12e} \approx 0.03065$$

We show that both of these bounds can be significantly improved.

Theorem 3. *If $\lambda \leq \lambda_\Delta \approx 0.3748$ and G is a subgraph of the $n \times n$ triangular grid, then running on the instance (G, K_3, λ), PRS takes $O(\log n)$ rounds in expectation.*

Theorem 4. *If $\lambda \leq \lambda_\square \approx 0.4063$ and G is a subgraph of the $n \times n$ square grid, then running on the instance (G, C_4, λ), PRS takes $O(\log n)$ rounds in expectation.*

Numerical evidence based on simulations of PRS leads us to conjecture that the optimal values of λ_Δ and $\lambda_\square$ are about 0.471 and 0.456 respectively. We suspect that the true values of

these constants may reflect important geometric properties of these lattices, from a statistical physics perspective.

Organization of the paper. In Section 2, we discuss *extremal* partial rejection sampling, and show how it can solve certain cases of the w-free string and the H-free subgraph problems. In Section 3, we discuss the general version of PRS introduced in (3). We give a brief overview of how Theorem 1 was proved, since we still use many of these ideas in proving the stronger versions tailored to our applications. In Section 4 we give a proof of Theorem 2, and in Section 5 we prove Theorems 3 and 4. Then in Section 6 we discuss some conjectures and possible directions for future work. Appendix A contains some numerical data from our simulations.

A word on exact versus approximate sampling. For many sampling problems, either an FPRAS for approximate sampling is known, or there is a hardness result making the existence of such an algorithm impossible under standard complexity assumptions. Comparatively little is known about the existence of *exact sampling* algorithms. Although a .01-approximate sampler almost always suffices in practice, we think it is a fundamentally interesting question whether exact sampling can be *harder* than approximate sampling. When exact counting isn't an option (as is almost always the case), there are (as far as the authors are aware) only two somewhat-general approaches to exact sampling: one of them is PRS, and the other is a technique called *coupling from the past* which can sometimes be used to turn MCMC-based approximate samplers into exact samplers (8).

2 Extremal partial rejection sampling

Let $X_1, ..., X_n$ be independent random variables and let $\{A_1 ..., A_m\}$ be a set of *bad events* dependent on some of the X_i. Let $\mathsf{var}(A_i)$ be the set of random variables that the bad event A_i depends on. We can define the *dependency graph* $G = (V, E)$, where $V = \{A_1, ..., A_n\}$, and $A_i \sim A_j$ (i.e. A_i and A_j are neighbors) if $\mathrm{var}(A_i) \cap \mathrm{var}(A_j) \neq \emptyset$. In other words, $A_i \sim A_j$ if they both depend on the same random variable X_k. Let S be a subset of events on the dependency graph G. Then we define $\Gamma(S) = \{A_i \,|\, A_i \sim A_j \text{ for some } A_j \in S, A_i \notin S\}$ and $\Gamma^+(S) = S \cup \Gamma(s)$.

We call an instance of this setup *extremal* if $A_i \sim A_j \implies \Pr[A_i \cap A_j] = 0$. As shown in (3), the following algorithm produces a uniform sample from the product distribution $(X_1, \ldots, X_n)$ conditioned on no A_i occurring.

Guo et al. give an exact formula for the expected runtime of Algorithm 1 on extremal instances:

Theorem 5. *Let q_S be the probability that the set of occurring bad events is exactly S, and suppose $q_\emptyset > 0$. Then the expected number of resampled events during Algorithm 1 is*

$$\sum_{i=1}^{n} \frac{q_{\{i\}}}{q_\emptyset}.$$

Algorithm 1: Partial Rejection Sampling

Result: Partial rejection sampling in the extremal case.
Draw independent samples of all random variables X_i;
while *at least one* A_i *holds* **do**
  find the independent set I of all occurring bad events A_i;
  independently resample all random variables $X_i \in \bigcup_{j \in I} \text{var}(A_j)$;
end
Output assignment of random variables X_i.

2.1 Sampling w-free strings: extremal case

Let Σ be a finite alphabet and let $w \in \Sigma^*$ be a string. We would like to uniformly sample from the distribution of w-free strings of length n; that is, strings of length n that do not contain w as a contiguous substring.

Viewing this problem in the variable framework, we let X_i be i.i.d. random variables over Σ for $i \in [1, n]$. The random variable X_i corresponds to the i-th index of the string. Then, the bad events A_i for $i \in [1, n - |w| - 1]$ correspond to an instance of the substring w whose first character is at index i.

Note that for general w this is not an extremal problem – that is, two bad events can depend on the same variable without being disjoint. For example, if $w = aba$ and the substring $ababa$ appears starting at index i, then A_i and A_{i+2} both occur despite both depending on X_{i+2}. As a result, in order to use Algorithm 1, we'll have to assume something about w. In Section 4 we'll be able remove this assumption.

Definition 6. *We say that w is non-translatable if no prefix of w is also a suffix of w.*

For example, the string $abcb$ is non-translatable, but the string $abcbabc$ is not because the prefix abc is also a suffix of that string. It is straight-forward to see that bad events defined by a non-translatable string are extremal, so we can apply Algorithm 1.

Theorem 7. *Let $\Sigma \geq 3$ be an alphabet and let w be a non-translatable substring in that alphabet. When running Algorithm 1 to uniformly sample a w-free string of length n, the expected number of resampled characters is at most n.*

To prove Theorem 7, we use the following notation. Let Σ^n be the set of strings of length n with characters in alphabet Σ, let Σ_w^n be the set of w-free strings of length n, and let $\Sigma_{w,1}^n$ be the set of strings with exactly one instance of substring w. Now we are ready to prove the theorem.

Proof. By applying Theorem 5, we obtain that the expected number of resampled events in the run of Algorithm 1 is at most $\frac{|\Sigma_{w,1}^n|}{|\Sigma_w^n|}$. To show that $\frac{|\Sigma_{w,1}^n|}{|\Sigma_w^n|} \leq n$, we exhibit an injection from $\Sigma_{w,1}^n$ to $\Sigma_w^n \times [n]$.

Let $|w| = k$, and let the first character of the substring w be at index $i \leq n - k + 1$. We claim that there is a way to change the k characters in indices i to $i + k - 1$ to create a w-free string.

There are $|\Sigma|^k$ possible reassignments of these k characters, and we will argue that at most

$$2\frac{|\Sigma|^k - 1}{|\Sigma| - 1} - 1$$

of these assignments will create an instance of w elsewhere in the string. Because this quantity is strictly less than $|\Sigma|^k$ for $|\Sigma| \geq 3$, this will imply the existence of such a reassignment of the k characters that eliminate all instances of w.

If a reassignment of the characters at indices i to $i + k - 1$ were to create an instance of w elsewhere in the string, the indices of the new string and the indices of the old string would necessarily intersect as no other characters in the string would have changed. As a result, the new instance of w would need to have its first character in one of the indices from $i - k + 1$ to $i + k - 1$. Suppose that the index of the first character of the new instance of w is j. Then, the maximum number of reassignments of characters in indices from i to $i + k - 1$ is $|\Sigma|^{|i-j|}$. Therefore, the maximum number of reassignments of the k characters that result in an instance of w is

$$|\Sigma|^{k-1} + |\Sigma|^{k-2} + ... + |\Sigma|^1 + |\Sigma|^0 + |\Sigma|^1 + ... + |\Sigma|^{k-2} + |\Sigma|^{k-1} = 2\frac{|\Sigma|^k - 1}{|\Sigma| - 1} - 1.$$

Now, we can define the injection. Given an element s from $\Sigma^n_{w,1}$ with the first element of substring w at index i, we can reassign the k characters of w to obtain an element s' of Σ^n_w. Then, we can construct the tuple (s', i), which is an element of $\Sigma^n_w \times [n]$, as desired. Note that given (s', i), the string s can be reconstructed by changing the k characters at indices i to $i + k - 1$ to the characters of the substring w. This proves the theorem.

□

Remark: The bound in the lemma is tight up to a factor which is constant in n. Indeed, if we send a pair $(s, i) \in \Sigma^n_w \times [n]$ to the string which has w spliced into it at location i, we obtain an element of $\Sigma^n_{w,1}$ (since w is non-translatable this only creates a single copy of it). The number of pre-images of any $s \in \Sigma^n_{w,1}$ under this mapping is at most $|\Sigma|^{|w|}$, which implies the bound

$$\frac{|\Sigma^n_{w,1}|}{|\Sigma^n_w|} \geq n \cdot |\Sigma|^{-|w|}.$$

Remark: The assumption $|\Sigma| \geq 3$ in the lemma is necessary. Indeed, consider $\Sigma = \{a, b\}$ and $w = ab$. Then $|\Sigma^n_w| = n + 1$, but $|\Sigma^n_w| \gtrsim n^3$, since it contains all strings of the form $b^i a^j b^k a^{n-(i+j+k)}$.

However, there is something we can say for sampling w-free strings from a binary alphabet when $|w| \geq 5$:

Proposition 8. *Let $\Sigma = \{0, 1\}$ and $|w| \geq 5$. Then the expected number of resampled events when using Algorithm 1 to uniformly sample over w-free strings of length n is logarithmic in n.*

We leave the proof of this proposition for Section 4.

2.2 Sampling H-free subgraphs: extremal case

Suppose we wish to sample a uniformly random (non-induced) subgraph of a graph G which does not contain a copy of some fixed graph H. In this context, the random variables $X_1, ..., X_n$ are i.i.d. Bernoulli random variables, indicating the presence/absence of each edge of G in our sampled subgraph. The corresponding dependency graph has a vertex for each copy of H in G, and two copies of H are connected by an edge in the dependency graph iff the corresponding copies of H share at least one edge in G. In general, the copies of H may overlap essentially arbitrarily inside of G, so this is certainly not an extremal problem. However, placing certain assumptions on G can create extremal instances.

For example, suppose G is C_4-free, and we wish to sample a triangle free subgraph of G. No two bad events (i.e. triangles) are dependent on one another (i.e. share an edge), so this is trivially extremal. (If two triangles shared an edge in G, then they would form a C_4.) It is easy to check that the bound from Theorem 5 is $O(|E(G)|)$ in this case.

This can be extended to give a sampling algorithm for instances in which the dependency graph has $O(1)$-sized connected components (e.g. sampling a triangle-free subgraph of a graph G which is C_{100} and $K_{2,100}$-free), but the running time is quite bad (although still polynomial in the size of G) and we omit the details.

3 General partial rejection sampling

We now discuss the general version of PRS (in (3)) which does not require any extremal conditions.

For any event E, let $\Gamma(E) = \{i : A_i \sim A_j \text{ and } A_i \neq E\}$, and for $S \subset [m]$, let $\Gamma^+(S) = S \cup_{i \in S} \Gamma(A_i)$. Let $p = \max_i \Pr[A_i]$ and $\Delta = \max_i |\Gamma(A_i)|$. Let R_{ij} be the event that the partial assignments on $\mathsf{var}(A_i) \cap \mathsf{var}(A_j)$ can be extended to an assignment satisfying A_j. Let $r_{ij} = \Pr[R_{ij}]$, and $r = \max_{i \sim j} r_{ij}$.

For a set of events F, let σ_F be the current assignment restricted to $\cup_{i \in F} \mathsf{var}(A_i)$. We say an assignment σ_F blocks an event i if i cannot occur given σ_F.

Let σ_t denote the assignment of the random variables after t rounds of 3 and Res_t denote the resampled events at this stage. Then the following theorem from (3) guarantees general partial rejection sampling the gives the desired output:

Theorem 9. *Given $\mathsf{Res}_0, \mathsf{Res}_1, \ldots, \mathsf{Res}_t$, for $t \geq 0$, σ_{t+1} has the product distribution conditioned on none of the events A_i occurring, where $i \in [n] \setminus \Gamma^+(\mathsf{Res}_t)$. In particular, the output upon termination is a sample from the product distribution conditioned on none of the bad events occurring.*

One of the main theorems in Guo, Jerrum and Liu is the following:

Theorem 10. *((3), Theorem 26): Let m be the number of events and n the number of variables. For any $\Delta \geq 2$, if $6ep\Delta^2 \leq 1$ and $3er\Delta \leq 1$, then the expected number of rounds used by Algorithm 3 is $O(\log m)$ and the expected number of resampled events is at most $O(m)$.*

Algorithm 2: Resample Set

Result: The set to be resampled at round t
let R = Bad(σ) #The events to be resampled;
let N = $\emptyset$ #The events that will not be resampled;
while $\partial R \setminus N = \emptyset$ **do**
  for $i \in \partial R \setminus N$ **do**
    if σ_R *blocks* i **then**
      Add i to N;
    else
      Add i to R;
    end
  end
end

Algorithm 3: General Partial Rejection Sampling

Result: A uniform sample from a distribution conditioned on no bad events occuring
Draw independent samples of every variable $X_1, .., X_n$ from their distributions;
while *At least one bad event occurs* **do**
  Res = Output from Algorithm 2 ;
  Resample the variables of every event in Res;
end

The proof of Theorem 10 works by showing that the expected size of the resampling set Res_t is exponentially decaying in t. The main idea is that when an event i is added during stage ℓ of Algorithm 5, there must be a chain of ℓ events ending at i and beginning at a bad event which *occurs*, such that the partial assignments never block the next event in the chain. The probability of such an occurrence can be bounded by $pr^{\ell-1}$, and the number of such potential paths can be bounded by something like $\Delta(\Delta-1)^{\ell}$. Hence the need for a condition like $r\Delta \lesssim 1$.

The following lemma is implicit in (3):

Lemma 11. *If i is added to R_ℓ during a run of Algorithm 5 on an assignment σ, then there exists a path $i_0, i_1, \ldots, i_\ell = i$ in the dependency graph such that, for each $0 \le k \le \ell$,*

(1) $i_k \in R_k$

(2) $i_{k'} \sim i_k \iff |k - k'| \le 1$

(3) the events $R_{i_{k-1} i_k}$ hold

Proof: If i is added in to R_ℓ, it must be because it was unblocked by some neighboring event $i_{\ell-1}$ which was added in the previous round. By induction, there is a path $i_0, i_1, \ldots, i_{\ell-1}$ as in the lemma. If $i \sim i_k$ for $k < \ell - 1$, then Algorithm 5 would've added i to either R_{k+1} or N_{k+1} during stage $k+1 < \ell$, which is impossible since i is added during stage ℓ. Therefore the path $i_0, \ldots, i_{\ell-1}, i_\ell$ satisfies the desired conditions. □

We shall henceforth refer to such paths as *bad paths* (as is done in (3)).

Let E_P denote the event that the path $P = i_0, i_1, \ldots, i_\ell$ is bad. Then condition (2) implies that the events $R_{i_{k-1}i_k}$ are independent for different values of k – indeed, $R_{i_{k-1}i_k}$ depends only on $\mathsf{var}(A_{i_{k-1}}) \cap \mathsf{var}(A_{i_k})$, and (2) implies that $\mathsf{var}(A_{i_k}) \cap \mathsf{var}(A_{i_{k'}}) = \emptyset$ unless $|k - k'| \leq 1$. Hence

$$\begin{aligned} \Pr[E_P] &= \Pr[(i_0 \in R_0) \wedge R_{i_0 i_1} \wedge (i_1 \in R_1) \wedge R_{i_1 i_2} \wedge \cdots \wedge (i_\ell \in R_\ell) \wedge R_{i_{\ell-1} i_\ell}] \\ &\leq \Pr[A_{i_0} \wedge R_{i_0 i_1} \wedge \cdots \wedge R_{i_{\ell-1} i_\ell}] \leq \Pr[A_{i_0} \wedge R_{i_0 i_1}] \prod_{k=2}^{\ell} r_{i_{k-1} i_k} \end{aligned} \tag{2}$$

To prove their Theorem 10, Guo et al. bound the probability of each $R_{i_{k-1}i_k}$ by r to obtain a uniform bound on the probability that a path of a length ℓ is bad, and then union bound over the possible paths of length ℓ. Improving upon their results will typically require more careful accounting of these paths.

The algorithm stops when $\mathsf{Res}_t = \emptyset$, and begins with $\mathsf{Res}_0 = \mathsf{Bad}_0 = [m]$. Summing the bound (2), the authors of (3) are able to obtain an upper bound of the form

$$\mathbb{E}[|\mathsf{Res}_{t+1}| \,|\, \mathsf{Res}_t] \leq C|\mathsf{Res}_t| \tag{3}$$

for some explicit $C < 1$, which as the following lemma shows, implies a logarithmic runtime bound.

Lemma 12. *If (3) holds with some $C < 1$, then the expected number of resampled events during Algorithm 6 is at most*

$$\frac{m}{1 - C}$$

and the expected number of rounds of resampling is at most

$$\log_{1/C} m + \frac{C}{1 - C}.$$

Proof: By Theorem 9, (3) is equivalent to

$$\mathbb{E}[|\mathsf{Res}_{t+1}| \,|\, \mathsf{Res}_0, \mathsf{Res}_1, \ldots, \mathsf{Res}_t] \leq C|\mathsf{Res}_t|$$

By the tower property of conditional expectations, $\mathbb{E}|\mathsf{Res}_t| \leq C^t \mathbb{E}|\mathsf{Res}_0| = C^t \cdot m$. If $C < 1$, then the expected total number of resampling events is

$$\sum_{t=0}^{\infty} \mathbb{E}|\mathsf{Res}_t| \leq m \cdot \sum_{t=0}^{\infty} C^t = \frac{m}{1 - C}$$

Let T be the number of rounds before Algorithm 6 terminates. Then

$$\mathbb{E}[T] \leq \log_{1/C} m + \sum_{t > \log_{1/C} m} \Pr[T > t] \leq \log_{1/C} m + \underbrace{\sum_{t > \log_{1/C} m} \mathbb{E}|\mathsf{Res}_t|}_{\leq \sum_{t=1}^{\infty} C^t}$$

and hence $\mathbb{E}[T] \leq \log_{1/C} m + \frac{C}{1-C}$. □

4 w-free strings

To sample w-free strings over Σ for *any* string w, we'll use Algorithm 3. Note that in this case $\Delta = 2|w| - 2$ and $r = \frac{1}{|\Sigma|}$. As mentioned in the introduction, to appeal directly to Theorem 10, we would need

$$|\Sigma| > 16|w|. \tag{4}$$

We now prove that for $|\Sigma| \geq 3$, this condition is unnecessary.

To prove Theorem 2, we improve upon the analysis in (3) used to prove Theorem 10. We'll make use of the following fact, which essentially says that if an event E does not share many dependent variables in common with a set S of bad events, then conditioning on $B(S)$ doesn't increase the likelihood of E by too much, where $B(S)$ denotes the event that none of the events in S occur. It can be proven inductively in a way similar to the Lovasz Local Lemma.

Lemma 13. *(Theorem 2.1 in (4), symmetric form): If $x \in \mathbb{R}^+$ is such that $x(1-x)^\Delta \geq p$, then*

$$\Pr[E \mid B(S)] \leq \Pr[E]\,(1-x)^{-|\Gamma(E)\cap S|} \tag{5}$$

for any event E and any set $S \subset [m]$.

Let $\Delta = 2(|w|-2)$ be the maximum degree in the dependency graph the w-free string problem, and $p = |\Sigma|^{-|w|}$ be the probability of any bad event. In everything that follows, we assume z is some positive number satisfying $z(1-z)^\Delta \geq p$, and $\alpha := (1-z)^{-1}$. Set $\delta \leq \Delta$ to be the number of *compatible shifts* of w in either direction, so that δ is, for a typical bad event, the number of neighboring bad events which are not blocked by it (and $\delta = 0 \iff w$ is non-translatable). Let C_t denote the number of connected components of Res_t.

Lemma 14. *At any stage t of Algorithm 3, we have*

$$\mathbb{E}[C_{t+1} \,|\mathsf{Res}_t] \leq \mathbb{E}[|\mathrm{Bad}_{t+1}| \,|\, \mathsf{Res}_t] \leq \frac{p\Delta}{2}|\mathsf{Res}_t| + \left(2p\alpha\left(\frac{\alpha^{\Delta/2}-1}{\alpha-1}\right) - \frac{p\Delta}{2} + p\right) C_t.$$

Proof: The first inequality is obvious from the nature of Algorithm 2 – it builds Res_{t+1} starting from Bad_{t+1} and attaching unblocked neighbors, so the number of connected components of Res_{t+1} is at most $|\mathsf{Bad}_{t+1}|$. To prove the second inequality, we need to look more closely at the structure of $\Gamma^+(\mathsf{Res}_t)$, which we know contains Bad_{t+1}.

Let S be some connected component of Res_t, and we will consider its contribution $\Gamma^+(S)$ to $\Gamma^+(\mathsf{Res}_t)$. Since S is connected, it looks like a line segment with gaps of at most $|w|-2$ between adjacent events, which will get included upon moving to $\Gamma^+(S)$. At each end of the segment, there are up to $|w|-1$ extra events. Thus,

$$|\Gamma^+(S)| \leq |S| + (|w|-2)|S| + 2(|w|-1) = \frac{\Delta}{2}|S| + \Delta \tag{6}$$

More precisely, $\Gamma^+(S)$ contains at most Δ events with any "unfresh" variables – just the ones on the fringes. There are at most

$$(|w|-1)(|S|-1)+1 = \frac{\Delta}{2}|S| - \frac{\Delta-2}{2},$$

interior (i.e. fresh) events, which therefore contribute at most an expected $\frac{p\Delta}{2}|S| - \frac{p(\Delta-2)}{2}$ events to Bad_{t+1}, while the unfresh events contribute at most

$$p \cdot 2 \cdot \sum_{k=1}^{\Delta/2} \alpha^k \leq 2p\alpha \cdot \left(\frac{\alpha^{\Delta/2}-1}{\alpha-1}\right)$$

in expectation by Lemma 13. Summing these contributions over all components S of Res_t, we obtain

$$\mathbb{E}[|\mathsf{Bad}_{t+1}| \,|\, \mathsf{Res}_t] \;\leq\; \frac{p\Delta}{2}|\mathsf{Res}_t| + \left(2p\alpha\left(\frac{\alpha^{\Delta/2}-1}{\alpha-1}\right) - \frac{p\Delta}{2} + p\right) C_t$$

as desired. □

Lemma 15. *If*

$$\frac{r\alpha^{|w|} - \alpha r^{|w|}}{\alpha - r} < 1,$$

then at any stage t of Algorithm 3, we have

$$\mathbb{E}[|\mathsf{Res}_{t+1}| \,|\, \mathsf{Res}_t] \leq B_{t+1} \cdot \left(1 + \frac{\alpha^{\Delta/2}\cdot\delta}{1 - \frac{r\alpha^{|w|}-\alpha r^{|w|}}{\alpha-r}}\right)$$

where B_{t+1} is the upper bound on $\mathbb{E}[|\mathsf{Bad}_{t+1}| \,|\, \mathsf{Res}_t]$ obtained in Lemma 14.

Proof: By Lemma 11, we know that for any event i which is added to Res_{t+1}, there must be a bad path $i_0, i_1, \ldots, i_\ell = i$. Paths of length 0 correspond exactly to Bad_{t+1}, whose expected size we know how to bound from Lemma 14. For a given i_0, there are at most δ choices for the next event i_1, which also fixes the "direction" of the path (the path moves "right" iff $i_0 < i_1$). From here, the path is parametrized by a tuple

$$(d_1, d_2, \ldots, d_{\ell-1})$$

where each $d_i \in \{1, \ldots, \frac{\Delta}{2}\}$, and $i_{k+1} = i_k + d_k$ for rightward paths, while $i_{k+1} = i_k - d_k$ for leftward paths. For a tuple $(d_1, \ldots, d_{\ell-1})$, the probability (conditional on Res_t) that the corresponding path is bad is at most

$$\begin{aligned}
\Pr[A_{i_0} \wedge R_{i_0 i_1} \wedge \cdots \wedge R_{i_{\ell-1} i_\ell} \,|\, \mathsf{Res}_t] &= \Pr[A_{i_0} \wedge R_{i_1 i_2} \wedge \cdots \wedge R_{i_{\ell-1} i_\ell} \,|\, B([m] \setminus \Gamma^+(\mathsf{Res}_t)])\\
&\leq \Pr[A_{i_0} \wedge R_{i_1 i_2} \wedge \cdots \wedge R_{i_{\ell-1} i_\ell}] \cdot \alpha^{|E \setminus \Gamma^+(\mathsf{Res}_t)|}\\
&\leq p \cdot r^{(\ell-1)|w| - \sum_{k=1}^{\ell-1} d_k} \cdot \alpha^{|E \setminus \Gamma^+(\mathsf{Res}_t)|}
\end{aligned}$$

where $E = \Gamma^+(\{i_0, i_1, \ldots, i_\ell\})$. We can bound

$$\begin{aligned}
|E \setminus \Gamma^+(\mathsf{Res}_t)| &\leq |\Gamma^+(i_0) \setminus \Gamma^+(\mathsf{Res}_t)| + |\Gamma^+(i_0) \setminus \Gamma^+(i_1)| + \sum_{k \geq 1}^{\ell-1} |\Gamma^+(i_{k+1}) \setminus \Gamma^+(i_k)|\\
&\leq |\Gamma^+(i_0) \setminus \Gamma^+(\mathsf{Res}_t)| + \frac{\Delta}{2} + \sum_{k=1}^{\ell-1} d_k
\end{aligned}$$

and so

$$\Pr[A_{i_0} \wedge R_{i_0 i_1} \wedge \cdots \wedge R_{i_{\ell-1} i_\ell} \,|\, \mathsf{Res}_t] \leq p \cdot \alpha^{\Delta/2 + |\Gamma^+(i_0) \setminus \Gamma^+(\mathsf{Res}_t)|} \cdot r^{(\ell-1)|w|} \cdot \left(\frac{\alpha}{r}\right)^{\sum_{k=1}^{\ell-1} d_k}.$$

Summing this bound over all $i_0 \in \Gamma^+(\mathsf{Res}_t)$ and admissible choices of i_1, we obtain the bound

$$\mathbb{E}[|\mathsf{Res}_{t+1} \setminus \mathsf{Bad}_{t+1}| \,|\, \mathsf{Res}_t] \leq \alpha^{\Delta/2} \cdot B_{t+1} \cdot \delta \sum_{\ell \geq 1} r^{(\ell-1)|w|} \sum_{(d_1,\ldots,d_{\ell-1})} \left(\frac{\alpha}{r}\right)^{\sum_{k=1}^{\ell-1} d_k} \tag{7}$$

Observe that for any x, we have

$$\sum_{(d_1,\ldots,d_{\ell-1})} x^{\sum_{k=1}^{\ell-1} d_k} = (x + x^2 + \cdots + x^{|w|-1})^{\ell-1} = x^{\ell-1} \left(\frac{x^{|w|-1} - 1}{x - 1}\right)^{\ell-1} \tag{8}$$

Plugging in $x = \alpha/r$, we see that

$$\begin{aligned}
\sum_{\ell \geq 1} r^{(\ell-1)|w|} \sum_{(d_1,\ldots,d_{\ell-1})} \left(\frac{\alpha}{r}\right)^{\sum_{k=1}^{\ell-1} d_k} &= \sum_{\ell \geq 1} r^{(\ell-1)|w|} \cdot \left(\frac{\alpha}{r}\right)^{\ell-1} \cdot \left(\frac{\left(\frac{\alpha}{r}\right)^{|w|-1} - 1}{\frac{\alpha}{r} - 1}\right)^{\ell-1} && (9)\\
&= \sum_{\ell \geq 1} \left(\frac{r\alpha^{|w|} - \alpha r^{|w|}}{\alpha - r}\right)^{\ell-1} && (10)\\
&= \frac{1}{1 - \frac{r\alpha^{|w|} - \alpha r^{|w|}}{\alpha - r}} && (11)
\end{aligned}$$

whenever the series converges, i.e. when $\frac{r\alpha^{|w|} - \alpha r^{|w|}}{\alpha - r} < 1$.

So the upper bound in (7) becomes

$$\mathbb{E}[|\mathsf{Res}_{t+1} \setminus \mathsf{Bad}_{t+1}| \,|\, \mathsf{Res}_t] \leq \frac{\alpha^{\Delta/2} \cdot B_{t+1} \cdot \delta}{1 - \frac{r\alpha^{|w|} - \alpha r^{|w|}}{\alpha - r}} \tag{12}$$

Adding in the remaining $\mathbb{E}[|\mathsf{Bad}_{t+1} \,|\, \mathsf{Res}_t] \leq B_{t+1}$ paths of length 0 yields the lemma. □

Lemma 16. *Set*

$$X_t = \frac{p\Delta}{2}|\mathsf{Res}_t| + \left(2p\alpha\left(\frac{\alpha^{\Delta/2} - 1}{\alpha - 1}\right) - \frac{p\Delta}{2} + p\right) C_t.$$

Then at any stage t of Algorithm 3, we have

$$\mathbb{E}[X_{t+1} \,|\, \mathsf{Res}_t] \leq \left(\frac{p\Delta}{2}\left(\frac{\alpha^{\Delta/2}\delta}{1 - \frac{r\alpha^{|w|} - \alpha r^{|w|}}{\alpha - r}}\right) + 2p\alpha\left(\frac{\alpha^{\Delta/2} - 1}{\alpha - 1}\right) + p\right) X_t.$$

Proof: This follows immediately by taking the appropriate linear combination of the bounds in Lemmas 14 and 15. □

Proof of Theorem 2: We can assume $|w| \geq 2$, since $|w| = 1$ is trivial. Let $|\Sigma| \geq 3$. The upper bound in Lemma 16 is clearly increasing in δ, so we can assume $\delta = \Delta$. For $|\Sigma| \geq 4$, plugging this value of δ into Lemma 16, along with $\alpha = (1 + 1/\Delta)$ already yields the corollary (via Lemma 12). For $|\Sigma| = 3$, this works for $|w| \notin \{2, 3, 4\}$. For these remaining lengths, we can choose α more optimally: set z equal to the smallest value of x for which $x(1-x)^{\Delta} \geq 3^{-|w|}$, and pick $\alpha = (1-z)^{-1}$. This tightens the bounds enough to work in these cases, as can be easily verified. □

Note that when $\delta = 0$, Algorithm 3 degenerates to Algorithm 1, and hence Lemma 16 can be used to obtain a runtime bound for Algorithm 1. In this case, Algorithm 1 takes $O(\log n)$ rounds when

$$\frac{2}{|\Sigma|^{|w|}} \cdot \left(\frac{1}{2} + \frac{\alpha^{|w|} - \alpha}{\alpha - 1} \right) < 1$$

for any $\alpha = (1-z)^{-1}$ with $z(1-z)^{2|w|-2} \geq |\Sigma|^{-|w|}$. When $|\Sigma| = 2$, this says nothing for $|w| = 2, 3, 4$, but for $|w| \geq 5$ we can take $z = 0.0453$ and $\alpha = 1.0475$, so that

$$\frac{2}{2^5} \cdot \left(\frac{1}{2} + \frac{1.0475^5 - 1.0475}{0.0475} \right) \approx 0.312 < 1.$$

Corollary 17. *For non-translatable $w \in \Sigma^n$, Algorithm 1 takes $O(\log n)$ rounds in expectation, except possibly for $|w| = 2, 3, 4$ over binary alphabets.*

5 Sampling H-free graphs

Recall that the H-free subgraph problem takes as input a graph G and a parameter λ and asks for a sample from the following distribution, supported on H-free (non-induced) subgraphs of G:

$$\Pr(G') \propto \begin{cases} \lambda^{e(G')}(1-\lambda)^{e(G)-e(G')} \propto \left(\frac{\lambda}{1-\lambda}\right)^{e(G')} & \text{if } G' \subseteq G \text{ is } H\text{-free} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

We can think of this problem as a special case of the hard core model in hypergraphs. When $\lambda = 1/2$, the problem is simply asking for a uniformly random H-free subgraph of an input graph. Viewed another way, each instance of the H-free subgraph problem (for $\lambda = \frac{1}{2}$) corresponds to a monotone $e(H)$-CNF

$$\bigwedge_{(e_1,\ldots,e_{e(H)}) \cong H} (\neg e_1 \vee \neg e_2 \vee \cdots \vee \neg e_{e(H)})$$

whose clauses correspond to copies of H and whose variables correspond to edges. Here we want to sample a uniformly random satisfying assignment. On instances G in which each edge

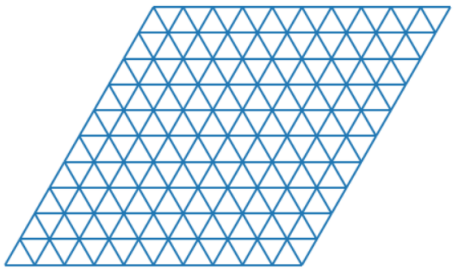

Figure 1: The 10×10 triangle grid.

appears in at most k different copies of H inside G, we say the corresponding CNF is a *read-k-monotone CNF*. When $k = 2$ (as it will be in the case of planar grid graphs), an algorithm of Lin, Liu and Lu (6) gives a FPTAS for approximately counting the set of satisfying assignments to a read-twice-monotone-CNF.

Theorem 18. *For $\lambda \leq 1/2$, there is an algorithm which ϵ-approximately solves the H-free subgraph problem on instances G in which each edge of G appears in at most 2 different copies of H. The running time is $O(n^{3.58}m^2 \cdot (1/\epsilon)^{2.58})$, where $n = |E(G)|$ and m is the number of copies of H in G.*

Proof: First suppose $\lambda = 1/2$. The approximate counting algorithm for read-twice-monotone-CNF given in (6) actually works by recursively estimating marginals – that is, for some edge e, it obtains an estimate for the ratio

$$\frac{Z(G \setminus e)}{Z(G)}$$

where $Z(G)$ denotes the number of H-free subgraphs of G. It then removes another edge and recurses down to the empty graph, whereby taking the product of all these estimates gives an estimate for $Z(G)$. When sampling instead of counting, rather than taking the product of the marginals one could simply sample from them recursively. This gives an approximate sampler for $\lambda = 1/2$.

For $\lambda < \frac{1}{2}$, one performs the above procedure but scales down each marginal probability of including an edge by a factor $\frac{\lambda}{1-\lambda}$. We refer the reader to (6) for further details. □

Remark: Liu and Lu (5) subsequently gave a FPTAS for counting solutions to read-k-monotone CNF, for $k \leq 5$. However, their algorithm is somewhat impractical, with a provable runtime bound on the order of $(n/\epsilon)^{144}$. For $k \geq 6$, there is no FPTAS (resp. FPRAS) for this problem unless NP = P (resp. NP = RP).

5.1 Triangle-free subgraphs of the triangular grid

We now analyze the performance of Algorithm 3 on instances G of the H-free subgraph problem, where G is a subgraph of the $m \times m$ triangular grid graph, and $H = K_3$ is the triangle.

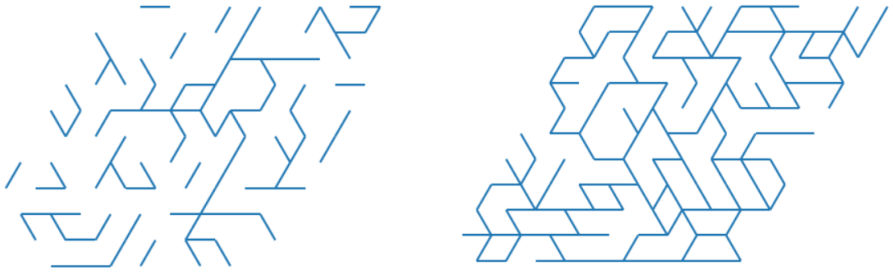

Figure 2: Randomly sampled K_3-free subgraphs with $\lambda = 0.25$ and $\lambda = 0.5$.

Here, the bad events correspond to the presence of each triangle, so $\Delta = 3$ and $p = \lambda^3$. Since any pair of dependent triangles share exactly one edge, $r_{ij} = \lambda$ for every $i \sim j$. Applying Theorem (10) directly, we see that for

$$\lambda \leq \frac{1}{9e} \approx 0.04088$$

Algorithm 3 solves this problem efficiently. Our present goal is to extend this range of λ.

Lemma 19. *At any stage t during Algorithm 3,*

$$|\Gamma^+(\mathsf{Res}_t)| \leq 2.5 \cdot |\mathsf{Res}_t|$$

Proof: Let $\Delta \in \mathsf{Bad}_t$ be an occurring triangle in round t. It suffices to prove the expansion inequality for the connected component S of Res_t containing Δ. There are a handful of cases to consider, depending on the number of neighbors of Δ in the dependency graph, which of course depends on the graph G. Since all edges of Δ are present by assumption, each of its neighbors will be added to S during stage 1 of Algorithm 2, so $S_1 := \Gamma^+(\{\Delta\}) \subseteq S$. In each case $|S_1| \in \{1, 2, 3, 4\}$, we can check that

$$\frac{|\Gamma^+(S_1)|}{|S_1|} \leq 2.5$$

holds. Now we observe that attaching a triangle to any non-empty set S' of triangles increases $|S'|$ by 1 and $|\Gamma^+(S')|$ by at most 2. Since $\frac{a}{b} \leq 2.5 \implies \frac{a+2k}{b+k} \leq 2.5$ for any $k \geq 0$, the inequality remains true for S, and hence for Res_t. □

Proposition 20. *If*

$$2.5\lambda^3 + \frac{7.5\lambda^3}{1 - \frac{2\lambda}{1+\lambda}} < 1$$

then Algorithm 3 takes an expected $O(\log m)$ rounds. In particular, we may take $\lambda \leq 0.3748$.

Proof: At a high level, instead of counting bad paths, we directly estimate the expected number of events added to Res_t during each phase of Algorithm 2, conditioned on the previous rounds.

More precisely, let R_ℓ^t be the intermediate set at stage ℓ in Algorithm 2, during round t of Algorithm 3 (so in particular $R_0^t = \mathsf{Bad}_t$ and $R_\infty^t = \mathsf{Res}_t$). We will show that for $\ell \geq 1$,

$$\mathbb{E}[|R_{\ell+1}^{t+1}| \,|\, R_\ell^{t+1}, \ldots, R_0^{t+1}, \mathsf{Res}_t] \leq C(\lambda)|R_\ell^{t+1}| \tag{14}$$

for some constant $C(\lambda)$. Since $\mathbb{E}[|R_0^{t+1}| \,|\, \mathsf{Res}_t] \leq \lambda^3|\Gamma^+(\mathsf{Res}_t)| \leq 2.5\lambda^3|\mathsf{Res}_t|$ (by Lemma 19) and $|R_1^{t+1}| \leq 3|R_0^{t+1}|$, we have

$$\mathbb{E}[[|R_1^{t+1}| \,|\, \mathsf{Res}_t] \leq 7.5\lambda^3|\mathsf{Res}_t|.$$

To upper bound $\mathbb{E}[|R_2^{t+1}| \,|\, R_1^{t+1}, R_0^{t+1}, \mathsf{Res}_t]$, we observe that each event (i.e. triangle) in R_2^{t+1} is one of the $\leq 2|R_1^{t+1}|$ triangles which share an edge with a triangle in R_1^{t+1} and do not belong to $R_0^{t+1} \cup R_1^{t+1}$. For each triangle Δ in R_1^{t+1}, we know that a certain edge (and possibly one more) is present (the common edge(s) between Δ and R_0^{t+1}), and that *not all three edges in* Δ *are present* – otherwise it would have been in R_0^{t+1}. If Δ shares two edges with R_0^{t+1}, we know for sure the third edge is not present, and so it contributes nothing to R_2^{t+1}. Otherwise, I claim that

$$\begin{aligned} \Pr(\Delta \text{ has another edge} \,|\, R_1^{t+1}, R_0^{t+1}, \mathsf{Res}_t) &\leq \Pr(\Delta \text{ has another edge} \,|\, \Delta \notin \mathsf{Bad}_{t+1}) \\ &= \frac{2\lambda}{1+\lambda} \end{aligned}$$

and hence $\mathbb{E}[|R_2^{t+1}| \,|\, R_1^{t+1}, R_0^{t+1}, \mathsf{Res}_t] \leq \frac{2\lambda}{1+\lambda} \cdot |R_1^{t+1}|$. The same argument works for 2 and 1 replaced by $\ell+1$ and ℓ, and so (14) holds with $C(\lambda) = \frac{2\lambda}{1+\lambda}$. By the tower property of conditional expectations, it follows that

$$\mathbb{E}[|R_{\ell+1}^{t+1}| \,|\, \mathsf{Res}_t] \leq C(\lambda)^\ell \cdot 7.5\lambda^3|\mathsf{Res}_t|. \tag{15}$$

Summing over $\ell \geq 0$, we see that

$$\mathbb{E}[|\mathsf{Res}_{t+1}| \,|\, \mathsf{Res}_t] \leq \left(2.5\lambda^3 + \frac{7.5\lambda^3}{1 - \frac{2\lambda}{1+\lambda}}\right)|\mathsf{Res}_t|$$

from which the proposition follows. □

5.2 Square-free subgraphs of the square grid

We now analyze the performance of Algorithm 3 on instances G of the H-free subgraph problem, where G is a subgraph of the $m \times m$ square grid graph, and $H = C_4$ is the square.

Here the bad events correspond to the presence of each square, so $\Delta = 4$ and $p = \lambda^4$. Since any pair of dependent squares share exactly one edge, $r_{ij} = \lambda$ for every $i \sim j$. Applying Theorem (10) directly, we see that for

$$\lambda \leq \frac{1}{12e} \approx 0.03065$$

Algorithm 3 solves this problem efficiently. Our present goal is to extend this range of λ.

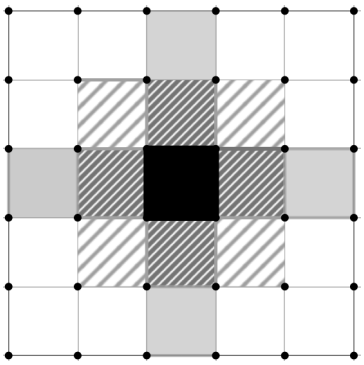

Figure 3: All squares which can be added to Res_{t+1} in stage 1 or 2 of Algorithm 2 due to $i_0 \in \mathsf{Bad}_{t+1}$.

Lemma 21. *At any stage t during Algorithm 3,*

$$|\Gamma^+(\mathsf{Res}_t)| \leq 3 \cdot |\mathsf{Res}_t|$$

Proof: The idea is the same as in the proof of Lemma 19. Let $\square \in \mathsf{Bad}_t$, and let S be the component of Res_t containing $\square$, which also must contain $S_1 := \Gamma^+(\{\square\})$. It is easy to check that for every possible S_1 we have

$$\frac{|\Gamma^+(S_1)|}{|S_1|} \leq 3,$$

and since attaching an additional square to any nonempty set S' of squares increases $|S'|$ by 1 and $|\Gamma^+(S')|$ by at most 3, the inequality persists all the way to S and hence to Res_t. $\square$

Proposition 22. *Let G be a subgraph of the $m \times m$ square grid. If*

$$3\lambda^4 \left(5 + \frac{4\lambda - 4\lambda^2 - 12\lambda^3}{1 - 2\lambda - \lambda^2}\right) < 1$$

then sampling a square-free subgraph with Algorithm 3 takes an expected $O(\log m)$ rounds. In particular we may take $\lambda \leq 0.4063$.

Proof: For each $i_0 \in \Gamma^+(\mathsf{Res}_t)$, we bound the expected number of squares which are added to Res_{t+1} as a result of a bad path rooted at i_0. All probabilities will be conditional on Res_t, that is, conditional on a certain set of squares *not* being present – however, by monotonicity, we can ignore this conditioning and exploit the resulting independence. The square i_0 (represented in black in Figure 3 above) is present with probability at most λ^4. If i_0 is present, it automatically brings the ($\leq$) four adjacent (heavy-hatched) squares into Res_{t+1}. Each of the four light-hatched squares can only be added to Res_{t+1} if the two edges they share with the heavy-hatched squares are present, which happens with probability at most λ^2. If those edges are not present, the squares will be *blocked* by the heavy-hatched squares and hence not added to Res_{t+1}. Each of the light-grey squares will be blocked unless their edge bordering a heavy-hatched square

is present, which happens with probability at most λ. Thus, the expected number of squares added due to i_0 during stages 1 and 2 of Algorithm 5 is at most

$$\lambda^4 \left(\underbrace{5}_{i_0 \text{ and its 4 neighbors}} + \underbrace{4\lambda^2}_{\text{light-hatched squares}} + \underbrace{4\lambda}_{\text{light grey squares}} \right).$$

Next we bound the expected number of squares added due to paths $i_0, i_1, \ldots, i_\ell$, with $\ell > 2$ and $i_k \sim i_{k'} \iff |k - k'| = 1$. This last condition is stronger than that of a self-avoiding walk – even the *neighbors* of previously visited sites must be avoided on the next step – so we'll call them *very self-avoiding walks*, or vSAWs. We can break up the set of such walks into two categories – S and D – based on their first two moves: either the first two moves are the same (e.g. left, left or up, up) or the two moves are different (e.g. up, right). I claim that for $\ell \geq 2$,

$$S_{\ell+1} = S_\ell + 2D_\ell \tag{16}$$
$$D_{\ell+1} = S_\ell + D_\ell \tag{17}$$

where S_ℓ and D_ℓ count the number of length-ℓ vSAWs of type S and D respectively starting from a fixed square. Indeed, if the first two moves of a length $\ell + 1$ vSAW are the same – say left, left – then up, down, left are the next available moves, which correspond to walks of length ℓ of type D, D, and S respectively, proving equation (16). To prove (17), suppose without loss that the first two moves are left, up. Then the next move is either right or up, since a left move would land on a neighbor of the original square. This yields (17).

Combining all of the above, we know that each $i_0 \in \Gamma^+(\mathsf{Res}_t)$ contributes at most

$$5\lambda^4 + \lambda^4 \cdot \left\langle \sum_{\ell \geq 0} \begin{pmatrix} \lambda & 2\lambda \\ \lambda & \lambda \end{pmatrix}^\ell \begin{pmatrix} 4\lambda \\ 4\lambda^2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\rangle = \lambda^4 \left(5 + \frac{4\lambda - 4\lambda^2 - 12\lambda^3}{1 - 2\lambda - \lambda^2} \right) \tag{18}$$

to $\mathbb{E}[|\mathsf{Res}_{t+1}| \,|\, \mathsf{Res}_t]$ (where we have used the identity $\sum_{\ell \geq 0} A^\ell = (I - A)^{-1}$). Summing up these contributions over all $|\Gamma^+(\mathsf{Res}_t)| \leq 3|\mathsf{Res}_t|$ (Lemma 21), we have shown

$$\mathbb{E}[|\mathsf{Res}_{t+1}| \,|\, \mathsf{Res}_t] \leq 3\lambda^4 \left(5 + \frac{4\lambda - 4\lambda^2 - 12\lambda^3}{1 - 2\lambda - \lambda^2} \right).$$

By Lemma 12, this implies the proposition. □

6 Conjectures and Future Work

6.1 w-free strings

We have shown that partial rejection sampling is takes an expected $O(\log n)$ rounds of resampling for $|\Sigma| \geq 3$, as well as for non-translatable $|w| \geq 5$ over binary alphabets.

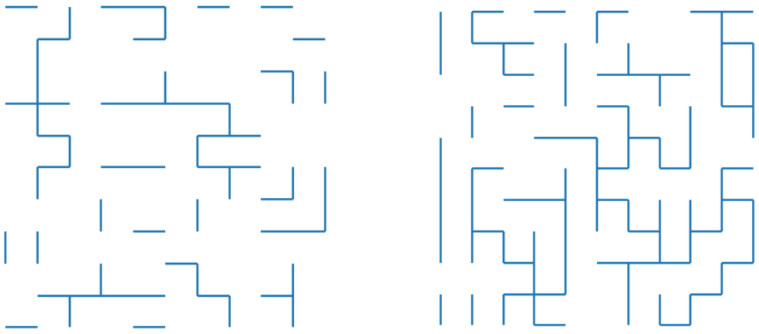

Figure 4: Randomly sampled C_4-free subgraphs with $\lambda = 0.25$ and $\lambda = 0.5$.

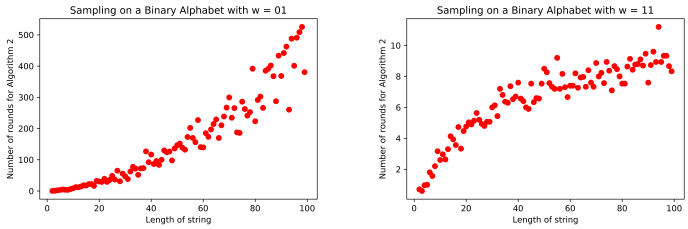

Figure 5: Average number of resampling rounds for w-free string sampling. Left: $w = 01$, Right: $w = 11$.

Based on our simulations (see figures), we make the following conjecture regarding the performance of PRS on w-free string sampling over binary alphabets:

Conjecture: Consider the w-free string problem over $\Sigma = \{0, 1\}$. If $w = 10$ or $w = 01$, then the expected number of rounds required by PRS is $\Theta(n^2)$, while if $|w| \geq 3$ or $w \in \{11, 00\}$, it is $O(\log n)$.

6.2 H-free subgraphs

We have proved (Theorem 3) that, for $\lambda \leq 0.3748$, sampling a triangle-free subgraph in the triangular grid graph takes an expected $O(\log(n))$ rounds. Similarly, we have shown (Theorem 4) that the same is true on for sampling square-free subgraphs in the square grid, for $\lambda \leq 0.4063$. However, based on our simulations, we make the following conjecture:

Conjecture: Theorem 3 holds for $\lambda \leq 0.471$, and Theorem 4 holds for $\lambda \leq 0.456$.

One source of slack in Theorems 4 and 4 comes from the boundary expansion lemmas (Lemmas 19 and 21). While these are tight in the worst case, the sets with $|\Gamma^+(S)|/|S|$ near these bounds are essentially one dimensional, which is *not* true of a typical component of Res_t. Replacing the number 2.5 in Lemma 19 with the number 1 for instance, already yields a bound on λ which nearly matches our conjecture.

Note that while the logarithmic behavior appears to transition beyond these values of λ, it is

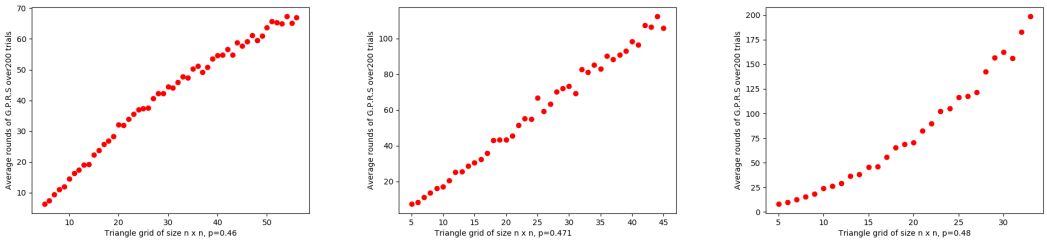

Figure 6: Average number of resampling rounds for K_3-free subgraph sampling on the triangular grid with $\lambda = 0.46, 0.471$ and 0.48 respectively.

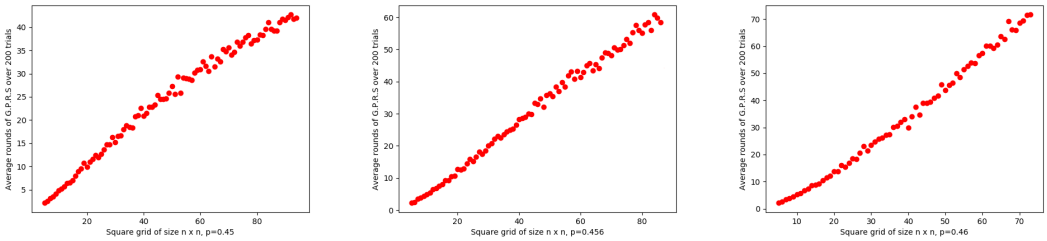

Figure 7: Average number of resampling rounds for C_4-free subgraph sampling on the square grid with $\lambda = 0.45, 0.456$, and 0.46 respectively.

not clear what type of behavior kicks in next and for what range of λ. Understanding this phase transition would be quite interesting, and will require techniques substantially different from ours, as ours inevitably prove either exponential decay or nothing at all.

It would also be interesting to know if, in situations where this version of general PRS is inefficient, Algorithm 2 could be modified to produce a smaller resampling set, while only breaking uniformity slightly. That is, when can PRS be turned into an FPRAS?

7 Acknowledgments

We would like to thank the MIT Department of Mathematics for providing so many research opportunities to students; SPUR+ has been an excellent and exciting program. Thank you to Professor Davesh Maulik and Professor Ankur Moitra for your helpful feedback and guidance throughout the course of this research. Last but not least, many thanks to our mentor Jake Wellens, without whom this project would not have been possible.

References

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

[2] Heng Guo, Mark Jerrum. Hard Disks

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

[4] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovasz Local Lemma. *J. ACM*, 58(6):28:1, 2011

[5] Jincheng Liu, Pinyan Lu. FPTAS for Counting Monotone CNF. In *SODA* 1531–1548. 2015.

[6] Chengyu Lin, Jingcheng Liu, Pinyan Lu. A Simple FPTAS for Counting Edge Covers. In *SODA*, 922–940, 2012.

[7] Robin A. Moser and Gabor Tardos. A constructive proof of the general Lovasz Local Lemma. In *J. ACM*, 57(2), 2010.

[8] James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Struct. Algorithms,* 9(1-2):223–252, 1996.