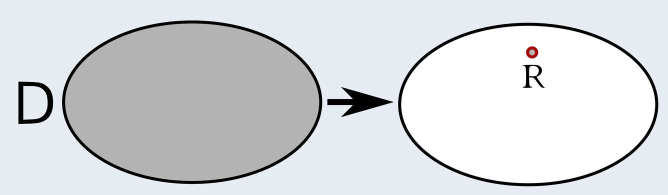
Local-Access Generators

Amartya Shankha Biswas, Ronitt Rubinfeld, Anak Yodpinyanee

CSAIL, MIT

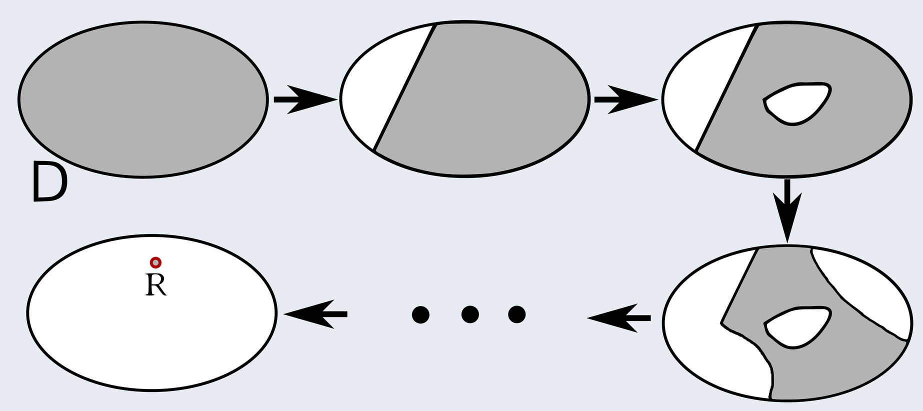
Partial Sampling from a Distribution

Full Sampling $R \sim \mathsf{D}$ in $\mathcal{O}(T)$ time



N steps of *Partial Sampling*

Each partial step should take $\tilde{\mathcal{O}}(T/N)$ time.



Problem Statement

A local-access generator of a random object $R \sim \mathsf{D}$, provides indirect access to R' with a *query oracle* s.t.

- All query responses (*partial samples*) are **consistent**
- The **distribution** of R' is ϵ-close to D in L_1 distance

Trivial Example - Sampling $G(n, p)$

Model: Undirected graph on N vertices (numbered $\langle 1, 2, 3, \cdots, N \rangle$), where each edge exists with prob. p.

Query Model: Given vertices u, v, is $(u, v) \in E$?

- Just a collection of $\frac{n(n-1)}{2}$ Bernoulli RVs with bias p.

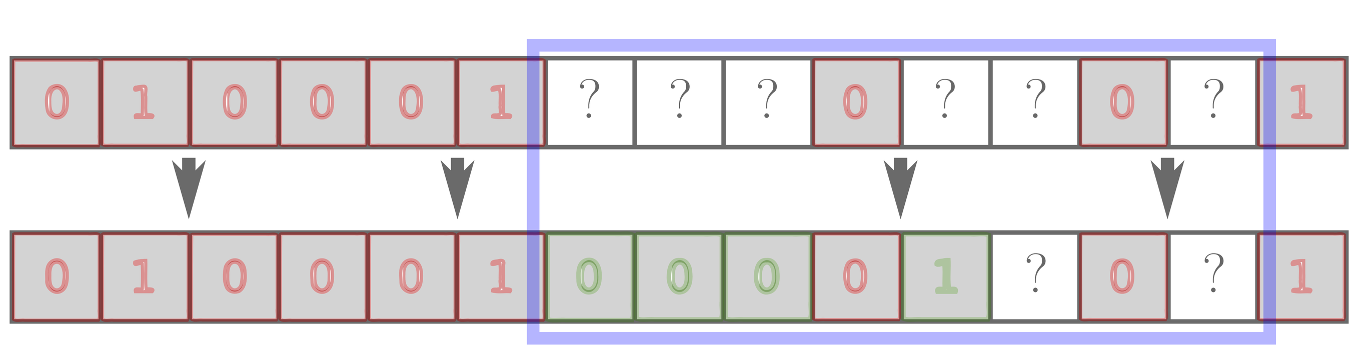
Find Next-Neighbor (*skip-sampling*)

Adjacency List query: Return neighbors of v in order.

$$\mathbb{P}[k \text{ non-neighbors before next-neighbor}] = p(1-p)^k$$

- Can sample from this distribution in $\tilde{\mathcal{O}}(1)$ time [cite].
- Avoid sampling each 0 separately.

Issue: Adjacency matrix is symmetric. So, each zero must also appear in the corresponding column of v.



If the sampled neighbor is a 0, discard and resample.
Cannot afford too many re-samplings.

Bucketing-Generator & Random-Neighbor Queries

- Divide each row of the adjacency matrix into contiguous buckets
- Expected number of neighbors in a bucket is $\Theta(\log n)$
- Each vertex v is associated with buckets $\langle B_1^v, B_2^v, B_3^v, \cdots \rangle$
- An **unfilled** bucket may contain some indirectly exposed neighbors
- A **filled** bucket will contain every possible sampled neighbor

Filling the i^{th} bucket B_i^v of vertex v

- Use skip-sampling to produce a **potential** *next-neighbor* u of v in B_i^v
- Check if (u, v) was set to 0, by looking at bucket $\mathcal{B}$ of u containing v
- If so, re-sample. Otherwise, mark u as a neighbor of v, and update $\mathcal{B}$
- W.h.p, only $\mathcal{O}(\log^2 n)$ **potential** neighbors are generated in B_i^v

Degree Sampling: Sampling the degree of v seems to be much harder

- Sampling $deg(v)$ conditions the remaining RVs in very non-trivial ways
- However, we can stil sample a random neighbor (with prob. $1/deg(v)$)

Random-Neighbor(v)

- Choose a random bucket $\mathcal{B}$ of v. If the $\mathcal{B}$ is **unfilled**, fill it.
- If k neighbors found in $\mathcal{B}$, start over (reject) with probability $1 - k/M$.
- If accepted, return an uniformly random neighbor found in $\mathcal{B}$.

For $M = \mathcal{O}(\log^2 N)$, the max number of neighbors in any bucket is $< M$. So, the number of rejection sampling rounds is $\mathcal{O}(\log^2 N)$ in expectation.

Stochastic Block Model

Communities $\{C_i\}_{i \in [r]}$ partition V: If $u \in C_i, v \in C_j$, then $\mathbb{P}_{(u,v) \in E} = p_{ij}$.

Given sizes of each comunity C_i and a range of length ℓ

- Count number of occurrences of each community in any contiguous range
- Sample from *Multivariate Hypergeometric Distribution*

$$\Pr[\mathbf{S}_\ell^{\mathbf{C}} = \langle s_1, \ldots, s_r \rangle] = \frac{\binom{C_1}{s_1} \cdot \binom{C_2}{s_2} \cdots \binom{C_r}{s_r}}{\binom{B}{\ell}} \quad \text{where } \ell = \sum_{i=1}^{r} s_i \text{ and } B = \sum_{i=1}^{r} C_i$$

Multivariate Hypergeometric Distribution

[cite] solves the special case of $r = 2$ and $B = 2\ell$.

Counting-Generator

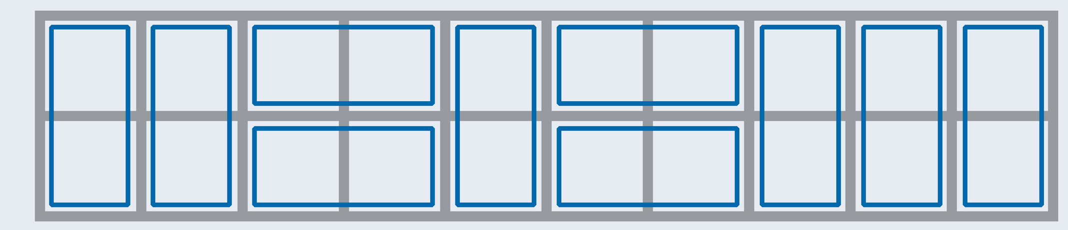
- **Extending to** $B \neq 2\ell$: Divide ℓ into $\mathcal{O}(\log n)$ dyadic segments.
- **Extending to** $r > 2$: Make a tree with r leaves (one for each C_i). Every branch down the tree is equivalent to a $r = 2$ splitting.

The complete generator is implemented by using Counting-Generator to sample the number of community members in a range, and then running the Bucketing-Generator as before.

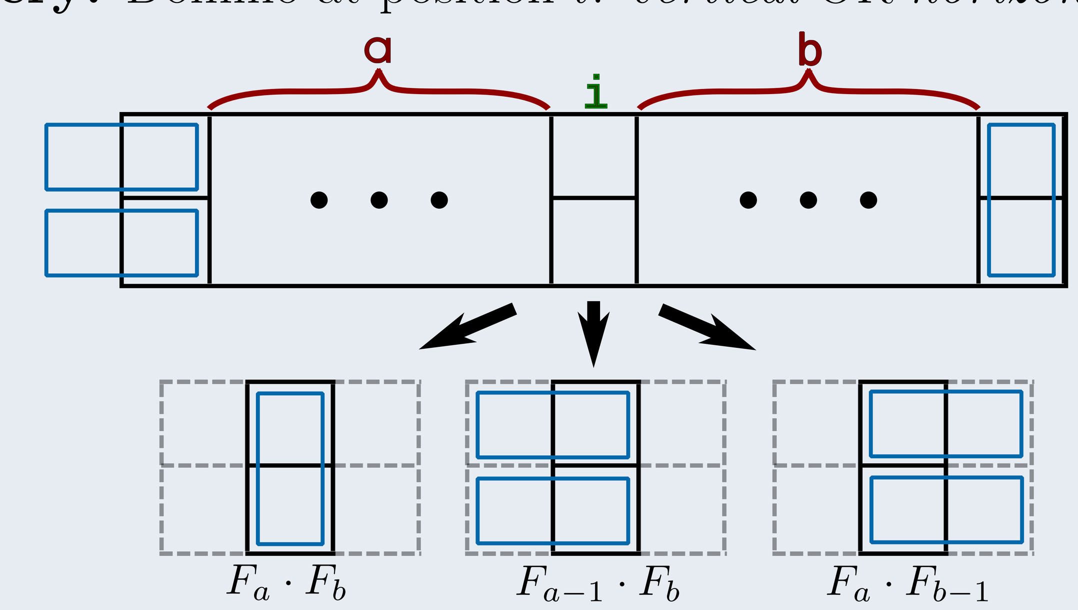
Work in Progress

Domino Tiling

A $2 \times n$ grid tiled with dominoes: F_n tilings possible.



Query: Domino at position i: *vertical* OR *horizontal*?



Sufficent to approximate F_c/F_{c-1}: Use ϕ if $c = \Omega(\log(n))$

Open: (Dimer model) $k \times n$ grid for $k = \Omega(1)$.

Graph Coloring: Glauber Dynamics

Find random k-coloring for graph with max degree Δ

Global Algorithm (for $k > 2\Delta$)

- Sample $\mathcal{O}(n \log n)$ (vertex, color) pairs: $\{(v_1, c_1), (v_2, c_2), (v_3, c_3), \cdots, (v_r, c_r)\}$
- For steps $i \in [1 \cdots r]$
  - If no neighbor of v_i has color c_i set v_i's color to c_i.
  - Else, do nothing

Local Algorithm (for $k = \Theta(\Delta \log n)$)

- Given v, what is $color(v)$ (in some random coloring)?
- Locally sample occurences of $(v, \star)$ using the *Count Splitting Generator*
- Sample $(w, \star)$ if necessary, where w is neighbor of v
- Query tree is bounded for $k = \Theta(\Delta \log n)$

Dyck Paths