Local access to Huge Random Objects

Amartya Shankha Biswas (MIT)
Ronitt Rubinfeld (MIT and TAU)
Anak Yodpinyanee (MIT)
Generating Huge Random Objects

Up front

D → Distribution

R → Sampled Random Object
Partial Sampling

“As needed” (local access queries)
Local Access to 1D Random Walk (on the line)

Query Height(t) returns position of walk at time t

Queries appear in arbitrary order

with probability $\frac{1}{2}$

with probability $\frac{1}{2}$
Random graph: Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generate “on the fly”
Amortize Sampling over Queries

Standard paradigm

“on-the-fly” sampler

Figure Adapted from [Even-Levi-Medina-Rosen 2017]
Query Requirements

- **Efficiency**: Use *polylogarithmic*
  - Time
  - Space
  - Random Bits

- All responses **consistent** with **single valid sample**

- **Output distribution** $\varepsilon$-close to the **true distribution** ($\ell_1$ distance)
Example Queries in Erdos-Renyi Graphs: $G(n, p)$

- **Every edge exists with probability $p$ (independently)**

- **Vertex-Pair($u, v$)**: Is edge $(u, v)$ present?
- **All-Neighbors($v$)**: Return full neighborhood of $v$
- **Degree($v$) → OPEN**
- **Next-neighbor($v$)**: **Lexicographically** next neighbor
- **Random-neighbor($v$)**: Return random neighbor of $v$
A harder setting
[B-Rubinfeld-Yodpinyanee]
Random (Valid) Coloring of a Graph

- **Input Graph:** $G$
  - Maximum Degree: $\Delta$
  - Number of colors: $q > \Delta$
- **Output:** Random Valid Coloring of $G$
  - Uniform over all valid colorings
- **Query:** Color of single vertex in sublinear time

- All responses must be consistent
- Overall coloring sampled from uniform distribution
Random Objects with **Huge Description Size**

- Can’t Read Full Description
- Model Parameters (in memory)
- Random bits
- Model Parameters
- Generation Algorithm
- Sublinear Probes per Query
- User
- Random Object (in memory)

- query
- response
Prior work
Local Access Model from [Goldreich-Goldwasser-Nussboim 03]

• Generators for huge random functions, codes, graphs, ...
• Important primitives
  • Sampling from binomial distribution
  • Interval-sum queries on random binary strings
    (see also [Gilbert-Guha-Indyk-Kotidis-Muthukrishnan-Strauss 02])
• Random graphs with specified property
  • e.g. Planted clique or Hamiltonian cycle
  • Focus on indistinguishable (under small number of queries and poly time)
Sparse $G(n, p)$ graph [Naor-Nussboim 2007]

• Degree at most polylog
• Queries:
  • Vertex-Pair
  • All-Neighbors

• Graphs generated:
  • Rooted tree/forest structure
  • Highly sequential random process
  • Sparse, but unbounded degree

• Queries (no bound on number):
  • Vertex-Pair
  • Next-Neighbor
    (Lexicographically in Adjacency List)
Summary of our Results
Erdos-Renyi $G(n, p)$

- Support all values of $p$
- Vertex-Pair
- Next-Neighbor
- Random-Neighbor

**Application:** Random walk in large degree graph!

Other Random Objects

- General graphs with **Independent edge probabilities** (under mild assumptions)
- 1D random walks
- Random Catalan objects
- (Simple) Domino Tilings

Today's Talk

- Unbounded Queries
- Polylog time space and random bits
- Generated objects are truly random (not just indistinguishable)
Random (Valid) Coloring of a Graph

- For $q > 9\Delta$
  - **Unbounded** Queries
  - Query color of specified vertex in **sublinear time**
    - Not polylog
  - **Memoryless**

*Local Computation Algorithms* [Rubinfeld, Tamir, Vardi, Xie] with **specific output distribution**

Sequential Markov Chain works for $q > 2\Delta$

For $q > 9\Delta$
probe complexity is $n^{6.12\Delta/q}$
$G(n, p)$ graphs
Vertex-pair query: Is there an edge from \( u \) to \( v \)?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generate “on the fly”

toss coins as needed

Undirected Symmetry

Query(3, 5)
Next-Neighbor and Random-Neighbor

• Dense case: $p > \frac{1}{\text{poly}(\log n)}$
  • Flip coins till you see 1
  • Time: $O(1/p)$

• Sparse Case: $p < \frac{\text{poly}(\log n)}{n}$
  • Use All-Neighbors query from [Naor-Nussboim 07]

• Intermediate is harder: e.g. $p = \frac{1}{\sqrt{n}}$
  • Many neighbors
  • Large gaps between neighbors

Can we do $o(1/p)$?
Next-Neighbor and Random-Neighbor

- Dense case: $p > \frac{1}{\text{poly}(\log n)}$
  - Flip coins till you see 1
  - Time: $O(1/p)$

- Sparse Case: $p < \frac{\text{poly}(\log n)}{n}$
  - Use All-Neighbors query from [Naor-Nussboim 07]

- Intermediate is harder: e.g. $p = \frac{1}{\sqrt{n}}$
  - Many neighbors
  - Large gaps between neighbors

Can we do $o(1/p)$?
Skip-sampling for next-neighbor queries: The case of directed graphs

\[ P[k \text{ zeros followed by a } 1] = p(1 - p)^k \]

\[ \text{CDF} = \sum_{k' = 0}^{k} p(1 - p)^{k'} = 1 - (1 - p)^k \]

Binary search on CDF
Skip-sampling for next-neighbor queries:

Undirected graphs yields correct distribution?

Some are determined by other neighbor

Write down all 0s?
Random-Neighbor queries via Bucketing

- Equipartition each row into contiguous buckets such that:
  - Expected # of neighbors in bucket is $\Theta(1)$
  - w.h.p. $\frac{1}{3}$ of buckets are non-empty
  - w.h.p. no bucket has more than $\log n$ neighbors
- Always determine a bucket completely

- Could have $\sqrt{n}$ buckets, each of size $\sqrt{n}$
Random Neighbors with rejection sampling

**Bucketing:**
- expected #neighbors in a bucket = $\Theta(1)$ expected, $\leq O(\log n)$ w.h.p.
- $\Rightarrow$ #neighbors $\approx$ #buckets

$\mathbf{v} : \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 & 1 & \cdots \end{bmatrix}$

**Step 1** pick a uniform random bucket
- “fill” this bucket if needed

$\mathbf{v} : \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$

**Step 2** pick a uniform random neighbor $u$

**Step 3** return $u$ with probability
- $\frac{\#neighbors \text{ in the bucket}}{O(\log n)}$
- otherwise, try again

$\Pr[\text{return } u] = \frac{1}{\#\text{buckets}} \times \frac{1}{\#\text{neighbors in bucket}} \times \frac{\#\text{neighbors in bucket}}{O(\log n)} \approx \frac{\Omega(1/\log n)}{\#\text{neighbors of } u}$

$\Pr[\text{return any neighbor}] \approx \Omega(1/\log n) \Rightarrow O(\log n)$ iterations suffice
How to fill a bucket?

• Bucket may be **indirectly** filled in certain locations
  • "1" entries reported
  • "0" entries not reported but can be queried

• **Why fast?** . . . # of "1" entries is bounded by $\log n$
Bucketing provides Next-Neighbor queries too!

Just process the next bucket in order
General Graphs with Independent Edge Probabilities

- Need mild assumptions on computing sums/products of probabilities
- **Stochastic Block Model**
  - Community structure
  - Probability of edge depends on communities of endpoints
- **Kleinberg’s Small World Model**
Other Results

- Random walks on the line
- Random Catalan objects
  - Random Dyck paths
    (1D random walk always positive)
  - Well bracketed expressions
  - Random Ordered Trees
- **Height** queries
  - **Depth** queries
    (in brackets and trees)
- **First-Return** queries
  - **Matching-Bracket** queries
  - **Next-Neighbor** in trees
Open Problems: Random Graphs

- Degree queries
- $i^{th}$ neighbor queries
- More complex queries
  - Sample a random triangle/clique
  - Random triangle containing specified vertex/edge
Open Problems: Large Description size

- What about $2\Delta < q < 9\Delta$?
- Random walks on general graphs
- Random satisfying assignment
- Random Linear Extensions of posets
- Random domino tilings (perfect matching)
Thank you!