Tutorial Outline

• Introduction (5 minutes)
  – Motivation, Challenges

• Sampling Background (10 minutes)
  – Assumption, Definition, Objectives

• Network Sampling methods (full access, restricted access, streaming access)
  – Estimating nodal or edge characteristics (30 minutes)
  – Sampling representative sub-networks (30 minutes)
  – Sampling and counting of sub-structure of networks (30 minutes)

• Conclusion and Q/A (15 minutes)
Introduction
motivation and challenges
Network analysis in different domains

Protein-Interaction network

Political Blog network

Social network

LinkedIn network

Food flavor network

Professional Network
Network characteristics

\[ \text{deg}(v_4) = 2 \quad \text{deg}(v_0) = 3 \quad \text{deg}(v_2) = 4 \quad \text{deg}(v_1) = 2 \quad \text{deg}(v_3) = 1 \]

\[ |V| = 5 \text{ vertices} \quad |E| = 6 \text{ edges} \]

\[ \text{Avg. degree} = 2.4 \]
Network characteristics

$|V| = 5$ vertices
$|E| = 6$ edges

Avg. Local Clustering Coeff = 0.6

Global Clustering Coeff:
$= (3 \times \text{# triangles}) / \text{# triplets}$
$= 0.55$

CCoeff($v_0$) = 2/3
CCoeff($v_1$) = 1
CCoeff($v_2$) = 2/6
CCoeff($v_3$) = 0
CCoeff($v_4$) = 1
Network characteristics

|V| = 5 vertices

|E| = 6 edges

Avg. Shortest Path Length = 1.4

Diameter = 2

Facebook New Orleans

Viswanath '09
Network analysis tasks

- Study the node/edge properties in networks
  - E.g., Investigate the correlation between attributes and local structure
  - E.g., Estimate node activity to model network evolution
  - E.g., Predict future links and identify hidden links

**Link Prediction:**
What is the probability that $u$ and $v$ will be connected in future?

Communities of physicists that work on network analysis
Network analysis tasks (cont.)

- Study the connectivity structure of networks and investigate the behavior of processes overlaid on the networks
  - E.g., Estimate centrality and distance measures in communication and citation networks
  - E.g., Identify communities in social networks
  - E.g., Study robustness of physical networks against attack

Source: Modularity in Protein Complex and Drug Interactions Reveals New Polypharmacological Properties, Jose C. Nacher mail, and Jean-Marc Schwartz, PloS One, 2012
Network analysis tasks

- Study local topologies and their distributions to understand local phenomenon
  - E.g., Discovering network motifs in biological networks
  - E.g., Counting graphlets to derive network “fingerprints”
  - E.g., Counting triangles to detect Web (i.e., hyperlink) spam

Graphlets

Graphlet Frequency Dist.
Computational complexity makes analysis difficult for very large graphs

- Best time complexities for various tasks: vertex count \( (n) \), edge count \( (m) \)
  - Computing centrality metrics \( O(mn) \)
  - Community Detection using Girvan-Newman Algorithm, \( O(m^2n) \)
  - Triangle counting \( O(m^{1.41}) \)
  - Graphlet counting for size \( k \) \( O(n^k) \)
  - Eigenvector computation \( O(n^3) \)
    - Pagerank computation uses eigenvectors
    - Spectral graph decomposition also requires eigenvalues

For graphs with billions of nodes, none of these tasks can be solved in a reasonable amount of time!
Other issues for network analysis

• Many networks are too **massive** in size to process offline
  – In October 2012, Facebook reported to have 1 billions users. Using 8 bytes for userID, 100 friends per user, storing the raw edges will take 1 Billions x 100 x 8 bytes = **800 GB**

• Some network structure may **hidden** or inaccessible due to privacy concerns
  – For example, some networks can only be crawled by accessing the one-hop neighbors of currently visiting node – it is not possible to query the full structure of the network

• Some networks are **dynamic** with structure changing over time
  – By the time a part of the network has been downloaded and processed, the structure may have changed
Network sampling (what and how?)

- We can sample a set of vertices (or edges) and estimate nodal or edge properties of the original network
  - E.g., Average degree and degree distribution

- Instead of analyzing the whole network, we can sample a small subnetwork similar to the original network
  - Goal is to maintain global structural characteristics as much as possible
    - E.g., degree distribution, clustering coefficient, community structure, pagerank

- We can also sample local substructures from the networks to estimate their relative frequencies or counts
  - E.g., sampling triangles, graphlets, or network motifs
Sampling objectives (Task 1)

- Estimate network characteristics by sampling vertices (or edges) from the original networks.
- Population is the entire vertex set (for vertex sampling) and the entire edge set (for edge sampling).
- Sampling is usually with replacement.

Estimating degree distribution of the original network.
Sampling objectives (Task 2)

- **Goal:** From $G$, sample a subgraph with $k$ nodes which preserves the value of key network characteristics of $G$, such as:
  - Clustering coefficient, degree distribution, diameter, centrality, and community structure
  - Note that, the sampled network is smaller, so there is a scaling effect on some of the statistics; for instance, average degree of the sampled network is smaller

- **Population:** All subgraph of size $k$
Sampling objective (Task 3)

- Sample sub-structure of interest
  - Find frequent induced subgraph (network motif)
  - Sample sub-structure for solving other tasks, such as counting, modeling, and making inferences

Sample \((S)\)

Build patterns for graph mining applications
Sampling scenarios

- **Full access assumption**
  - The entire network is visible
  - A random node or a random edge in the network can be selected

- **Restricted access assumption**
  - The network is hidden, however it supports crawling, i.e. it allows to explore the neighbors of a given node.
  - Access to one seed node or a collection of seed nodes are given.

- **Streaming access assumption (limited memory and fast moving data)**
  - In the data stream, edges arrive in an arbitrary order (arbitrary edge order)
  - In the data stream, edges incident to a vertex arrive together (incident edge order)
  - Stream assumption is particularly suitable for dynamic networks
Sampling methods
methodologies, comparison, analysis
TASK 1

Estimating node/edge properties of the original network
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restricted Access</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Stream Access</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Sampling Objective**
  - Estimate node/edge properties in original network
  - Sample representative subgraph/subnetwork
  - Estimate frequency of subgraph patterns

- **Sampling methodology**
  - Data Access Assumption
Sampling methods

Task 1, Full Access assumption

- Uniform node sampling
  - Random node selection (RN)
- Non-uniform node sampling
  - Random degree node sampling (RDN)
  - Random pagerank node sampling (RPN)
- Uniform edge sampling
  - Random Edge selection (RE)
- Non-uniform edge sampling
  - Random node-edge (RNE)
Random Node Selection (RN)

- In this strategy, a node is selected uniformly and independently from the set of all nodes
  - The sampling task is trivial if the network is fully accessible.
- It provides unbiased estimates for any nodal attributes:
  - Average degree and degree distribution
  - Average of any nodal attribute
  - $f(u)$ where $f$ is a function that is defined over the node attributes
RDN (degree prop), RPN (pagerank prop)

• In RDN sampling, selection of a node is proportional to its degree
  – If $\pi(u)$ is the probability of selecting a node, $\pi(u) = d(u)/2m$
  – High degree nodes have higher chances to be selected
  – Average degree estimation is higher than actual, and degree distribution is biased towards high-degree nodes.
  – Any nodal estimate is biased towards high-degree nodes.

• RPN samples in proportion to Pagerank, which is the stationary distribution vector of a specially constructed Markov process
  – Visit a neighbor following an outgoing link with probability $c$ (typically kept at 0.85), and jump to a random node (uniformly) with probability $1-c$
  – Node $u$ is sampled with a probability which is proportional to $c \cdot \frac{d_{in}(u)}{m} + \frac{1-c}{n}$
  – When $c=1$, the sampling is similar to RDN for the directed graph, when $c = 0$, sampling is similar to RN
  – Nodes with high in-degree have higher chances to be selected
Random Edge Selection (RE)

• In a random edge selection, we uniformly select a set of edges and the sampled network is assumed to comprise of those edges.

• A vertex is selected in proportion to its degree

  – If $\rho = \frac{|E_s|}{|E|}$, the probability of a vertex $u$ to be selected is:
    $1 - (1 - \rho)^{d(u)}$

  – when $\rho \to 0$, the probability is: $\rho \cdot d(u)$

  – With more sample degree bias is reduced

  – The selection of vertices is not independent as both endpoints of an edge are selected

• Nodal statistics will be biased to high-degree vertices

• Edge statistics is unbiased due to the uniform edge selection.
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✅</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restricted Access</td>
<td></td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Data Stream Access</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Task 1**: Estimate node/edge properties in original network
- **Task 2**: Sample representative subgraph/subnetwork
- **Task 3**: Estimate frequency of subgraph patterns
Sampling under restricted access

• Assumptions
  – The network is connected, (if not) we can ignore the isolated nodes
  – The network is hidden, however it supports crawling, Access to one seed node or a collection of seed nodes is given.

• Methods
  – Graph traversal techniques (exploration without replacement)
    • Breadth-First Search (BFS), or Depth-First Search (DFS)
    • Forest Fire (FF)
    • Snowball Sampling (SBS)
  – Random walk techniques (exploration with replacement)
    • Classic Random Walk
    • Markov Chain Monte Carlo (MCMC) using Metropolis-Hastings algorithm
    • Random walk with restart (RWR), or with random jump (RWJ)

• During the traversal or walk, the visited nodes are collected in a sample set and those are used for estimating network parameters
Breadth-first Sampling

Discovers all vertices that are at distance $d$ before discovering any vertices that is at distance $d + 1$

Depth-first Sampling

Discovers farthest vertex along a chain, then backtracks
Forest Fire (FF) Sampling [Leskovec ’06]

- FF is a randomized version of BFS
- Every neighbor of current node is visited with a probability $p$. For $p=1$ FF becomes BFS.
- FF has a chance to die before it covers all nodes.
- It is inspired by a graph evolution model and is used as a graph sampling technique
- Its performance is similar as BFS sampling
Snowball Sampling

• Similar to BFS

• $n$-name snowball sampling is similar to BFS

• at every node $v$, not all of its neighbors, but exactly $n$ neighbors are chosen randomly to be scheduled

• A neighbor is chosen only if it has not been visited before.

• Performance of snowball sampling is also similar to BFS sampling

Snowball: $n=2$
Classic Random Walk Sampling (RWS)

• At each iteration, one of the neighbors of currently visiting node is selected to visit

• For a selected node, the node and all its neighbors are discovered (not yet explored though)

• The sampling follows depth-first search pattern

\[
p_{u,v} = \begin{cases} 
1/d(u) & \text{if } v \in \text{adj}(u) \\
0 & \text{otherwise}
\end{cases}
\]

• This sampling is biased as high-degree nodes have higher change to be sampled, probability that node \( u \) is sampled is

\[
\pi(u) = \frac{d(u)}{2m}
\]
Other variants of random walk

- Random walk with restart (RWR)
  - Behaves like RWS, but with some probability \((1-c)\) the walk restarts from a fixed node, \(w\)
  - The sampling distribution over the nodes models a non-trivial distance function from the fixed node, \(w\)

- Random walk with random jump (RWJ)
  - Access to arbitrary node is required
  - RWJ is motivated from the desire of simulating RWS on a directed network, as on directed network RWS can get stuck in a sink node, and thus no stationary distribution can be achieved
  - Behaves like RWS, but with some probability \((1-c)\), the walk jumps to an arbitrary node with uniform probability.
  - The stationary distribution is proportional to the pagerank score of a node, so the analysis is similar to RPN sampling
Exploration based sampling will be biased toward high degree nodes. How can we modify the algorithms to ensure nodes are sampled uniformly at random?
Uniform sampling by exploration

• Traversal/Walk based sampling are biased towards high-degree nodes

• Can we perform random walk over the nodes while ensuring that we sample each node uniformly?

• Challenges
  – We have no knowledge about the sample space
  – At any state, only the currently visiting nodes and its neighbors are accessible

• Solution
  – Use random walk with the Metropolis-Hastings correction to accept or reject a proposed move
  – This can guarantee uniform sampling (with replacement) over all the nodes
Uniform node sampling with Metropolis-Hastings method [Henzinger ’00]

- It works like random walk sampling (RWS), but it applies a correction so that high-degree bias of RWS is eliminated systematically.

- MH uses a proposal distribution, $Q$ to choose one of the neighbors (say, $j$) of the current node (say, $i$). If $Q$ is uniform, $q_{ij} = 1/d(i)$.

- Then, it accepts or rejects the sampled walk with probability $\alpha_{ij} = \min \left\{ \frac{b_j q_{ji}}{b_i q_{ij}}, 1 \right\}$, where $b_i$ is a target probability up to a constant factor.

- For uniform target distribution, $b_i = b_j$.

- Thus, $\alpha_{ij} = \min \left\{ \frac{d_i}{d_j}, 1 \right\}$ if $d(j) \leq d(i)$, the choice is accepted only with probability 1, otherwise with probability $\frac{d(i)}{d(j)}$.

- If a graph have $n$ vertices, using the above MH variant, every node is sampled with $\frac{1}{n}$ probability.
When sampling algorithm selects nodes non-uniformly...

Can we remove the sampling bias in nodal statistics by post-processing?
Correction for biased sampling (Salganik & Heckathorn ’04)

• Assume, \( \{x_s\}_{s=1}^{t} \) are the nodes sampled to compute the unbiased expectation of nodal attribute, \( f \) which is \( E_u[f] \)

• \( \pi \) is a distribution that we achieve by a biased sampling

• Let’s define a weight function \( w: V \to R \) such that \( w(x_i) = \frac{1}{\pi(x_i)} \)

• Then, \( E_u[f] = \frac{1}{t} \sum_{s=1}^{t} w(x_s) \cdot f(x_s) = \frac{\sum_{s=1}^{t} w(x_s) \cdot f(x_s)}{\sum_{s=1}^{t} w(x_s)} \)

• Estimating average degree by SRW, here \( \pi(x_s) = \frac{1}{d_{x_s}} \)

• Unbiased estimate of average degree is

\[
E_u(\bar{d}) = \frac{\sum_{s \in [1..t]} \left( \frac{1}{d_{x_s}} \right) d_{x_s}}{\sum_{s \in [1..t]} \frac{1}{d_{x_s}}} = \frac{t}{\sum_{s \in [1..t]} \frac{1}{d_{x_s}}}
\]
ANALYSIS
Comparison

- **Node property estimation**
  - Uniform node selection (RN) is the best as it selects each node uniformly
  - Average degree and degree distribution is unbiased

- **Edge property estimation**
  - Uniform edge selection (RE) is the best as it select each edge uniformly
  - For example, we can obtain an unbiased estimate of assortativity by RE method

<table>
<thead>
<tr>
<th>Method</th>
<th>Vertex Selection Probability, $\pi(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RN, MH- uniform target</td>
<td>$\frac{1}{n}$</td>
</tr>
<tr>
<td>RDN, RWS</td>
<td>$\frac{d(u)}{2m}$</td>
</tr>
<tr>
<td>RPN, RWJ</td>
<td>$c \cdot \frac{d_{in}(u)}{m} + (1 - c) \cdot \frac{1}{n}$ (undirected) [ c \cdot \frac{d(u)}{2m} + (1 - c) \cdot \frac{1}{n} ] (directed)</td>
</tr>
<tr>
<td>RE</td>
<td>$\sim \frac{d(u)}{2m}$</td>
</tr>
<tr>
<td>RNE</td>
<td>$\frac{1}{n} \left(1 + \sum_{x \in \text{adj}(u)} \frac{1}{\text{adj}(x)}\right)$</td>
</tr>
</tbody>
</table>
Expected average degree and degree distribution for walk based methods

- For a degree value $k$ assume $p_k$ is the fraction of vertices with that degree
  \[ \sum_{k>0} p_k = 1 \]

- **Uniform Sampling**, node sampling probability, $\pi(v) = \frac{1}{|V|} = \frac{1}{n}$
  - Expected value of $p_k$: $q_k = \sum_{v \in V} \pi(v) \cdot 1_{d(v)=k} = \frac{1}{|V|} \cdot p_k \cdot |V| = p_k$
  - Expected observed node degree: $\sum_{k>0} k \cdot q_k = \sum_{k>0} k \cdot p_k$

- **Biased Sampling** (degree proportional), $\pi(v) = \frac{d(v)}{2 \cdot |E|} = \frac{d(v)}{2m}$
  - Expected value of $p_k$: $q_k = \sum_{v \in V} \pi(v) \cdot 1_{d(v)=k} = \frac{k}{2 \cdot |E|} \cdot p_k \cdot |V| = \frac{k \cdot p_k}{d}$
  - Expected observed node degree: $\sum_{k>0} k \cdot q_k = \sum_{k>0} \frac{k^2 \cdot p_k}{d} = \frac{d^2}{d}$

Overestimate high-degree vertices, underestimate low-degree vertices

Overestimate average degree
Example

- Actual degree distribution
  \[ q_k = \left\{ \frac{1}{12}, \frac{3}{12}, \frac{4}{12}, \frac{3}{12}, \frac{1}{12} \right\} \]

- Average degree = 3

- RWS degree distribution
  \[ q_k = \left\{ \frac{1}{36}, \frac{6}{36}, \frac{12}{36}, \frac{12}{36}, \frac{5}{36} \right\} \]

- RWS average degree
  - 3.39
Expected average degree for traversal based methods [Kurant ‘10]

- Note that, for walk based method, the sampling is with replacement, so their analysis does not change with the fraction of sampling.
- Traversal based method behaves like RWS when the sample size is small, but as the sample size increases its estimation quickly converges towards the true estimation.
- Behavior of all the traversal method is almost identical.
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restricted Access</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Data Stream Access</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Sampling Objective:
- Estimate node/edge properties in original network
- Sample representative subgraph/subnetwork
- Estimate frequency of subgraph patterns

Sampling methodology:
- Task 1: Full Access
- Task 2: Restricted Access
- Task 3: Data Stream Access
Sampling under data streaming access assumption

• Previous approaches assume:
  • **Full access** of the graph or **Restricted access** of the graph – access to only node’s neighbors

• Data streaming access assumption:
  • A graph is accessed only sequentially as a stream of edges
  • Massive stream of edges that cannot fit in main memory
  • Efficient/Real-time processing is important

Stream of edges over time

Graph with timestamps
Sampling under data streaming access assumption

• The complexity of sampling under streaming access assumption defined by:
  – Number of sequential passes over the stream
  – Space required to store the intermediate state of the sampling algorithm and the output
    • Usually in the order of the output sample

Stream of edges over time

No. Passes
Space
Sampling methods under data streaming access assumption

• Most stream sampling algorithms are based on random reservoir sampling and/or Min-wise Sampling
Min-Wise Sampling

• A random "tag" drawn independently from the Uniform(0, 1) distribution

• This "tag" is called the "hash value" associated with each arriving item

• The sample consists of the items with the $n$ smallest tags/hash values seen thus far

• Uniformity follows by symmetry: every size-$n$ subset of the stream has the same chance of having the smallest hash values

For any arriving item $i$

$h(i) \sim \text{Uniform}(0, 1)$
Reservoir Sampling

• Random Reservoir Sampling [Vitter’85]

  – A family of random algorithms for sampling from data streams
  – Choosing a set of $n$ records from a large stream of $N$ records
    • $n \ll N$
  – $N$ is too large to fit in main memory and usually unknown
  – One pass, $O(N)$ algorithm
Reservoir Sampling Algorithm

Step 1: Add first $n$ records to reservoir
Reservoir Sampling Algorithm

Step 2: Select the next record with probability \( P = \frac{n}{k} \)

- When a record is chosen for the reservoir, it becomes a candidate and replaces one of the former candidates
Reservoir Sampling Algorithm

Step 2: Select the next record with probability \( P = \frac{n}{k} \)

\[
\begin{align*}
\text{stream} & \quad \cdots \quad 13 \quad 18 \quad 19 \quad 14 \quad 17 \quad 15 \quad 23 \quad 22 \quad 26 \quad \cdots \\
\text{reservoir} & \quad 13 \quad 23 \quad 19 \quad 14 \quad 17 \quad 15
\end{align*}
\]
Reservoir Sampling Algorithm

After repeating Step 2 ...

- at the end of the sequential pass, the current set of \( n \) candidates is output as the final sample
Sampling methods under data streaming access assumption

• Streaming Uniform Edge Sampling
  – Extends traditional random edge sampling (RE) to streaming access

• Streaming Uniform Node Sampling
  – Extends traditional random node sampling (RN) to streaming access
Sampling methods under data streaming access assumption

• Streaming Uniform Edge Sampling
  – Extends traditional random edge sampling (RE) to streaming access

• Streaming Uniform Node Sampling
  – Extends traditional random node sampling (RN) to streaming access
Streaming Edge Sampling (RE)

Step 0: Start with an empty reservoir of size $n = 2$

Stream of edges

$V_1\rightarrow V_6\rightarrow V_2\rightarrow V_1\rightarrow V_2\rightarrow V_1\rightarrow V_2\rightarrow V_1$

reservoir

$n$
Streaming Edge Sampling (RE)

Step 1: Add first $n$ edges to reservoir

Stream of edges

V_1 V_6 V_2 V_1 V_2 V_1 V_2 V_1
V_5 V_2 V_1 V_3 V_4 V_6 V_1 V_2

reservoir

← $n$ →
Streaming Edge Sampling (RE)

Step 2: Sample next edge $V_2-V_1$ with prob. $P = n/k$

Stream of edges

reservoir

$n$
Streaming Edge Sampling (RE)

Edge $V_2 - V_1$ is selected

Stream of edges

$V_2 - V_1$ is selected replacing $V_6 - V_2$
Streaming Edge Sampling (RE)

At the end of the sequential pass . . .

Stream of edges

- at the end of the sequential pass, the current set of \( n \) edges is output as the final sample
Sampling methods under data streaming access assumption

- Streaming Uniform Edge Sampling
  - Extends traditional random edge sampling (RE) to streaming access

- Streaming Uniform Node Sampling
  - Extends traditional random node sampling (RN) to streaming access
Sampling from the stream directly selects nodes proportional to their degree.

Not suitable for sampling nodes uniformly.

Use Min-Wise Sampling.
Streaming Uniform Node Sampling (RN)

- **Assumption:** nodes arrive into the system only when an edge that contains the new node is streaming into the system

- It is difficult to identify which $n$ nodes to select a priori with uniform probability

- Use min-wise sampling

- Maintain a reservoir of nodes with top-n minimum hash values

- A node with smaller hash value may arrive late in the stream and replace a node that was sampled earlier
Streaming Uniform Node Sampling (RN) – Example

Step 0: Start with an empty reservoir

Stream of edges

V₁ V₂ V₁ V₂ V₁ V₂ V₁

V₅ V₂ V₁ V₃ V₄ V₆ V₁ V₂

reservoir

← n →
Streaming Uniform Node Sampling (RN) – Example

Step 1: Add first $n$ nodes to reservoir

Stream of edges

reservoir → $V_1$ | $V_5$ | $V_6$

- Apply a uniform random Hash Function to the node id $h(i)$
- $h$ defines a true random permutation on the node id’s
Streaming Uniform Node Sampling (RN) – Example

Step 1: Add first $n$ nodes to reservoir

- Apply a uniform random Hash Function to the node id $h(i)$
- $h$ defines a true random permutation on the node id’s

Stream of edges

$h(6) < h(1) < h(5)$ reservoir

$h(6), h(1), h(5)$
Streaming Uniform Node Sampling (RN) – Example

Step 2: Sample the next node . . .

Stream of edges

Assume $h(2) < h(5)$

$h(6) h(1) h(5)$

reservoir $\rightarrow$ $V_6$ $V_1$ $V_5$

$\leftarrow n \rightarrow$
Streaming Uniform Node Sampling (RN) – Example

Step 2: Sample the next node . . .

Stream of edges

$V_2$ replaced $V_5$

$h(2) < h(5)$

$h(6) < h(1) < h(2)$
At the end of the sequential pass . . .

Stream of edges

at the end of the sequential pass, the current set of \( n \) nodes is output as the final sample
TASK 2

Representative Sub-network Sampling
Representative Sub-network Sampling

- Select a representative sampled sub-network

- The sample is representative if its structural properties are similar to the full network
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Restricted Access</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Stream Access</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sampling Objective:
- Estimate node/edge properties in original network
- Sample representative subgraph/subnetwork
- Estimate frequency of subgraph patterns
Sampling methods under full access assumptions

• Node sampling
  – Starts by sampling nodes
  – Add all edges between sampled nodes (Induced subgraph)

• Edge Sampling
  – Uniform edge sampling
  – Uniform Edge Sampling with graph induction

• Exploration Sampling
  – Graph traversal techniques
  – Random walk techniques

• Sampling the network community structure
  – Graph traversal to maximize the sample expansion

• Metropolis-Hastings sampling
  – Search the space for the best representative sub-network
• Assume we have the following network

• **Goal:** sample a representative sub-network of size = 4 nodes
Sampling methods under full access assumptions

- Node sampling
  - Starts by sampling nodes
  - Add all edges between sampled nodes (Induced subgraph)

- Edge Sampling
  - Uniform edge sampling
  - Uniform Edge Sampling with graph induction

- Exploration Sampling
  - Graph traversal techniques
  - Random walk techniques

- Sampling the network community structure
  - Graph traversal to maximize the sample expansion

- Metropolis-Hastings sampling
  - Search the space for the best representative sub-network
Node Sampling

Original Network

Sampled Nodes

Sampled Sub-network

Edges added by graph induction
Node Sampling (NS)

- Node sampling samples nodes uniformly

- Generally biased to low degree nodes

- Sampled sub-network may include isolated nodes (zero degree)

- The sampled sub-network typically fails to preserve many properties of the original network
  - diameter, hop-plot, clustering co-efficient, and centrality values of vertices.
Sampling methods under full access assumptions

- **Node sampling**
  - Starts by sampling nodes
  - Add all edges between sampled nodes (Induced subgraph)

- **Edge Sampling**
  - Uniform edge sampling
  - Uniform Edge Sampling with graph induction

- **Exploration Sampling**
  - Graph traversal techniques
  - Random walk techniques

- **Sampling the network community structure**
  - Graph traversal to maximize the sample expansion

- **Metropolis-Hastings sampling**
  - Search the space for the best representative sub-network
Uniform Edge Sampling (ES)

Original Network

Sampled Edges

Sampled Sub-Network
Edge Sampling (ES)

- Edge sampling samples nodes proportional to their degree

- Generally biased to high degree nodes (hub nodes)

- The sampled sub-network typically **fails** to preserve many properties of the original network
  - diameter, hop-plot, clustering co-efficient, and centrality values of vertices.
Uniform Edge Sampling with graph induction (ES-i) [Ahmed ’13]

Original Network

Sampled Edges

Sampled Sub-Network

---

Edges added by graph induction
Edge Sampling with graph induction (ES-i)

- Similar to the edge sampling (ES)
  - ES-i samples nodes proportional to their degree
  - Generally biased to high degree nodes (hub nodes)

- Due to the additional graph induction step
  - The sampled sub-network preserves many properties of the original network
    - degree, diameter, hop-plot, clustering co-efficient, and centrality values of vertices.
Sampling methods under full access assumptions

• Node sampling
  – Starts by sampling nodes
  – Add all edges between sampled nodes (Induced subgraph)

• Edge Sampling
  – Uniform edge sampling
  – Uniform Edge Sampling with graph induction

• Exploration Sampling
  – Graph traversal techniques
  – Random walk techniques

• Sampling the network community structure
  – Graph traversal to maximize the sample expansion

• Metropolis-Hastings sampling
  – Search the space for the best representative sub-network
Exploration Sampling – Breadth First

Sampled sub-network consists of all nodes/edges visited
Exploration Sampling

- Generally biased to high degree nodes (hub nodes)

- The sampled sub-network can preserve the diameter of the network if enough nodes are visited

- The sampled sub-network usually under-estimates clustering coefficient

- One solution: induce the sampled sub-network
Sampling methods under full access assumptions

• Node sampling
  – Starts by sampling nodes
  – Add all edges between sampled nodes (Induced subgraph)

• Edge Sampling
  – Uniform edge sampling
  – Uniform Edge Sampling with graph induction

• Exploration Sampling
  – Graph traversal techniques
  – Random walk techniques

• Sampling the network community structure
  – Graph traversal to maximize the sample expansion

• Metropolis-Hastings sampling
  – Search the space for the best representative sub-network
Sampling Community Structure
[Maiya’10]

• Sampling a sub-network representative of the original network community structure
  – Expansion Sampling (XS)

• Expansion Sampling Algorithm:
  – **Step1:** Starts with a random vertex $u$, add $u$ to the sample set $S$
    
    $$ S = S \cup \{u\} $$

  – **Step2:** Choose $v$ from $S$’s neighborhood, s.t. $v$ maximizes
    
    $$ |\text{Neigh}(v) - \{\text{Neigh}(S) \cup S\}| $$

  – **Step3:** Add $v$ to $S$
    
    $$ S = S \cup \{v\} $$

  – Repeat Step 2 and 3 until $|S| = k$
Three Candidates for next step

Start with random node

Three Candidates for next step
Select the one with maximum expansion

Four Candidates for next step
Select the one with maximum expansion

Four Candidates for next step
Select the one with maximum expansion
Sampling methods under full access assumptions

• Node sampling
  – Starts by sampling nodes
  – Add all edges between sampled nodes (Induced subgraph)
• Edge Sampling
  – Uniform edge sampling
  – Uniform Edge Sampling with graph induction
• Exploration Sampling
  – Graph traversal techniques
  – Random walk techniques
• Sampling the network community structure
  – Graph traversal to maximize the sample expansion
• Metropolis-Hastings sampling
  – Search the space for the best representative sub-network (Hubler et al.)
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Restricted Access</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Data Stream Access</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sampling methods under restricted access assumptions

• When access is restricted, it is necessary to use exploration sampling to sample subnetwork
  – Graph traversal techniques
  – Random walk techniques
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Restricted Access</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Data Stream Access</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

**Sampling Objective**
- Estimate node/edge properties in original network
- Sample representative subgraph/subnetwork
- Estimate frequency of subgraph patterns
Sampling under data streaming access assumption

- Sampling a subgraph (sub-network) while the graph is streaming as a sequence of edges of any arbitrary order.

Stream of edges over time

Graph with timestamps
Sampling methods under data streaming access assumption

- Streaming Uniform Node Sampling
- Streaming Uniform Edge Sampling
- Partially Induced Edge sampling (PIES)
Sampling methods under data streaming access assumption

• Streaming Uniform Node Sampling
  – Similar to Task 1 but adding the edges from the graph induction

• Streaming Uniform Edge Sampling

• Partially Induced Edge sampling (PIES)
Sampling methods under data streaming access assumption

- Streaming Uniform Node Sampling
  - Similar to Task 1 but adding the induced edges

- Streaming Uniform Edge Sampling
  - Similar to Task 1

- Partially Induced Edge sampling (PIES)
Sampling methods under data streaming access assumption

• Streaming Uniform Node Sampling
  – Similar to Task 1 but adding the induced edges

• Streaming Uniform Edge Sampling
  – Similar to Task 1

• Partially Induced Edge sampling (PIES)
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 0: Start with an empty reservoir

Stream of edges

Node List

reservoir

Edge List

$n$
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 1: Add the subgraph with first $n$ nodes to reservoir

Stream of edges

Node List

Edge List

reservoir

$V_1$, $V_5$, $V_6$, $V_2$

$n$

$V_1$, $V_5$, $V_6$, $V_2$
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 3: Add the induced edge . . . $V_2 - V_1$

Stream of edges

$V_1$ and $V_2$ are sampled
Induced edge $V_2 - V_1$ added deterministically
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 3: Sample the next edge . . . $V_1 - V_3$ With prob. $P = n/k$

Stream of edges

Node List

reservoir

Edge List
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 3: Sample the next edge . . . \( V_1 - V_3 \) With prob. \( P = n/k \)

Stream of edges

Flip a coin with prob. \( P = n/k \)

reservoir

Node List \( \left\{ V_1, V_5, V_6, V_2 \right\} \)

Edge List
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 3: Sample the next edge . . . $V_1 - V_3$ With prob. $P = n/k$

Stream of edges

Edge is selected

reservoir

Node List

$V_1, V_5, V_6, V_2$

$n$

Edge List
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Step 3: Sample the next edge . . . $V_1 - V_3$ With prob. $P = \frac{n}{k}$

Stream of edges

$V_1$ is already sampled
We need to insert $V_3$
$V_3$ replaces $V_5$ randomly

reservoir

Node List

Edge List
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

At the end of the sequential pass . . .

Stream of edges

Node List

reservoir

Edge List
Example Partially Induced Edge Sampling (PIES) [Ahmed’13]

Stream of edges

Sampled Sub-network
Effective degree distribution of the induced graph

ArXiv Hep-PH
34K nodes
421K Edges

Avg. Degree = 24.4

Uniform Node Sampling (unbiased selection gets more low degree)

Edge/Exploration Sampling (biased selection gets more high degree)
Edge sampling with graph induction (Es-i) performs better than other methods

Ahmed’13
Partially Induced Edge Sampling (PIES) performs better than other methods.

Ahmed'13
More conclusions ...

• Sampling from the stream in one pass is a hard task
  – Edge-based methods generally perform well
    • But sometimes under-estimates clustering coefficient

• Stream sampling perform better for sparse graphs
TASK 3

Local Sub-structure Sampling
Task 3 (Sampling sub-structures from networks)

• In this case, our objective is to sample small substructure of interest
  – Sampling graphlets (Bhuiyan et al., 2012)
    • Is used for build graphlet degree distribution which characterize biological networks
  – Sampling triangles and triples (Rahman et al., 2014, Buriol et al., 2006)
    • Is used for estimating triangle count
  – Sampling network motifs (Kashtan et al., 2004, Wernicke 2006, Saha et al. 2014)
  – Sampling frequent patterns (Hasan et al. 2009, Saha et al. 2014)
Differentiating species via network analysis of protein-protein interactions

http://bioinformatics.oxfordjournals.org/content/27/2/245/F1.expansion.html
Q: Why counting triangles?

- Indicate the rate of transitivity in social networks
  - e.g., friends of friends tend to become friends themselves!

- Used for web spam detection
  - e.g., distribution of triangles among spam hosts deviates from non-spam hosts

- A core feature for graph generation models
  - e.g., Exponential Random Graph Models (ERGM)
  - e.g., Two-level Erdos-Renyi Block Model

- Used to improve community detection algorithms

Triangle is a Cycle of 3 nodes

[Burt'07] [Becchetti et.al’08] [Robins et.al’07] [Seshadhri et.al’12] [Berry et.al’11]
Triple sampling from a network: Motivation

- A triple, \((u, v, w)\) at a vertex \(v\), is a path of length two for which \(v\) is the center vertex.
- A Triple can be closed (triangle), or open (wedge)
- It is useful for approximating triangle counting and transitivity
- \(T\) is a set of uniformly sampled triples, and \(C \subseteq T\), is the set of closed triples
- Approximate transitivity is \(\sigma = \frac{|C|}{|T|}\)
- Approximate triangle count in a network is: \(\frac{\sigma}{3} \cdot \text{triple-count}\)
Induced $k$-Subgraph Sampling: motivation

- Network motifs are induced subgraphs that occur in the network far more often than in randomized networks with identical degree distribution.

- To identify network motif, we first construct a set of random networks (typically few hundreds) with identical degree distributions.

- If the concentration of some subgraph in the given network is significantly higher than the concentration of that subgraph in the random networks, those subgraphs are called network motif.

- Concentration of a size-$k$ subgraph $S_i^{(k)}$ is defined as: $C_i^{(k)}(G) = \frac{f_i^{(k)}(G)}{\sum_{j: \text{size}(j)=k} f_j^{(k)}(G)}$

- Exact counting of $k$-subgraphs are costly, so sampling methods are used to approximate $k$-subgraph concentrations.
<table>
<thead>
<tr>
<th>Sampling Objective</th>
<th>Data Access Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling methodology</td>
<td>Data Stream Access</td>
</tr>
<tr>
<td></td>
<td>Restricted Access</td>
</tr>
<tr>
<td></td>
<td>Full Access</td>
</tr>
</tbody>
</table>

- **Task 1**: Estimate node/edge properties in original network.
- **Task 2**: Sample representative subgraph/subnetwork.
- **Task 3**: Estimate frequency of subgraph patterns.
Triple sampling with full access

• Take a vertex \( V \) uniformly, choose two of its neighbors \( U \), and \( W \); return the triple \((u, v, w)\).

• This **does not** yield uniform sampling, as the number of triples centered to a vertex is non-uniform,

\[
P\{\text{triple } (u, v, w) \text{ is sampled}\} = \frac{1}{n \cdot \binom{d(v)}{2}}
\]

• For triangle counting from a set of sampled triples (say, \( T' \)), we can apply the same un-biasing trick that we applied for nodal characteristics estimation,

\[
TC = \frac{\text{triple-cnt}}{3 \cdot W} \times \sum_{t \in T} w_t \cdot 1_{\{t \text{ is closed}\}}
\]

here \( w_t \) is the probability of selecting a triple, and \( W = \sum_{t \in T} w_t \).
Uniform Triple sampling with full access [Schank ‘05]

- **Method**
  - Select a node \( v \) in proportion to the number of triples that are centered at \( v \), which is equal to \( \frac{d(u)(d(u) - 1)}{2} \)
  - Return one of the triple centered at \( v \) uniformly
  - For the first step, we need to know the degree of all the vertices

- **Triangle count estimate is easy, we just need to use the formula**

\[
TC = \frac{\text{triple-cnt}}{3 \cdot |T|} \times \sum_{t \in T} 1_{\{t \text{ is closed}\}}
\]

Node A has degree 3, it has 3 triples centered around it
Sampling of subgraphs of size $k$ [Kashtan ’04]

• Method
  1. Select an edge uniformly
  2. Populate Neighborhood
  3. Choose one neighbor uniformly
  4. If size $\neq k$, goto 2 else return the pattern

Probability of generating this pattern (in this edge order):

$$P = \frac{1}{m} \times \frac{1}{7} \times \frac{1}{7} = \frac{1}{49m}$$
Sampling of subgraphs of size $k$ (Kashtan ‘04)

Probability of generating this pattern (in this edge order):

$$= \frac{1}{m} \times$$

$$= \frac{1}{m} \times \frac{1}{3}$$

$$= \frac{1}{m} \times \frac{1}{3} \times \frac{1}{3}$$

$$= \frac{1}{9m}$$

- Sampling probability is not uniform, but we can compute the probability.
- We can apply the same unbiasing trick that we have used before to obtain uniform estimate of the concentration of some subgraph of size $k$.
- Assume, we take $B$ samples of size-$k$ subgraph, Then the concentration of some subgraph type $s_i$ is:

$$\frac{\sum_{j=1}^{B} w_j \cdot 1_{\{j\text{-type}=s_i\}}}{W} \quad \text{where} \quad w_j = \frac{1}{p_j}, \text{ and } W = \sum_{j=1}^{B} w_j$$
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Restricted Access</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Data Stream Access</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

- **Estimate node/edge properties in original network**
- **Sample representative subgraph/subnetwork**
- **Estimate frequency of subgraph patterns**

**Sampling methodology**

**Sampling Objective**
Triple sampling by exploration

- Perform a random walk over the triple space
  - Given a seed node, form a triple from its neighborhood information
  - Define a neighborhood in the triple space that can be instantiated online using local information
  - Choose the next triple from the neighborhood

- The stationary distribution of the walk is proportional to the number of neighbors of a triple in the sampling space
Neighborhood

Two neighboring triples share two vertices among them, entire neighborhood graph is not shown.
Triple sampling by exploration based solution for restricted access

- If we want to obtain unbiased estimate of the concentration of triangle (transitivity), we can use the same un-biasing strategy as before:

  - The concentration of triangle is:

    \[
    \sum_{j=1}^{B} \frac{w_j \cdot 1_{\{j\text{-type}=\text{closed}\}}}{W}, \quad \text{here, } W = \sum_{j=1}^{B} w_j \text{ and } w_j = 1/d(j)
    \]
Degree of the triple $(1, 2, 3)$ is 6 as can be seen in the above graph, so while unbiasing,

\[ w_{(1,2,3)} = \frac{1}{6} \]
Another exploration-based triple sampling that uses MH algorithm

- MH algorithm uses simple random walk as proposal distribution

- Then is uses the MH correction (accept/reject) so that the target distribution is uniform over all the triples

- Like before, if triple $T_i$ is currently visiting state, and the triple $T_j$ is chosen by the proposal move, then this move is accepted with probability $\min\left\{1, \frac{d(i)}{d(j)}\right\}$

- MH algorithm thus confirm uniform sampling, so the transitivity can be easily computed as $\frac{1}{|T|} \sum_{k=1}^{T} 1_{\{T_k \text{ is closed}\}}$
For uniform sampling, every move from triple $T_i$ to triple $T_j$ is accepted with probability $d(T_i)/d(T_j)$.

If $T_i = (1, 2, 3)$ and $T_j = (1, 3, 4)$, move from $T_i$ to $T_j$ is accepted with probability $\max\{1, \frac{6}{4}\} = 1$, but a move from $T_j$ to $T_i$ is accepted with probability $\max\{1, \frac{4}{6}\} = \frac{4}{6}$. 
Sampling subgraphs of size $k$ using exploration [Bhuiyan ’12]

- Both random walk and M-H algorithm can also be used for sampling subgraph of size $k$

- As we did in triangle sampling, we simply perform a random walk on the space of all subgraphs of size-$k$

- We also define an appropriate neighborhood
  - For example, we can assume that two $k$-subgraphs are neighbors, if they share $k - 1$ vertices

- This idea has recently been used in GUISE algorithm, which uniformly sampled all subgraphs of size 3, 4, and 5. It used M-H methods
  - Motivation of GUISE was to build a fingerprint for graphs that can be used for characterizing networks from different domains.
Graphlet sampling [Bhuiyan ’12]

- Given a large graph, we want to count the occurrences of each of the above graphlets
- Using the graphlet counts, we want to build a frequency histogram (GFD) to characterize the graph
Graphlet Frequency Distribution (GFD)

- Approximate the frequency of each graphlet using their relative frequency by random walk sampling
- Use logarithm of normalized frequency to plot GFD

Exact enumeration is too costly!
Exact Vs App GFD

Comparison with Actual and Approximate GFD for
(a) ca-GrQc (b) ca-Hepth (c) yeast (d) Jazz
Use of GFD [Rahman ’13]

- GFD can represent a group of graphs constructed using same mechanism.
- We used GFD in agglomerative hierarchical clustering which produced good quality (purity close to 90%) cluster.

Result of agglomerative hierarchical graph-clustering (first 18 dimensions of GFD used).

\[
Purity = \frac{9 + 5 + 3 + 3}{22} = 0.91
\]
<table>
<thead>
<tr>
<th>Data Access Assumption</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Access</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Restricted Access</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Data Stream Access</td>
<td>✔️</td>
<td>✔️</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Sampling Objective**
- **Estimate node/edge properties in original network**
- **Sample representative subgraph/subnetwork**
- **Estimate frequency of subgraph patterns**

**Sampling methodology**
Given a large graph $G$ represented as a stream of edges $e_1, e_2, e_3$…

We show how to **efficiently sample** from $G$ while limiting memory space to calculate **unbiased estimates** of various graph properties.

**Motivation**

Sampled
Edge Stream

No. Edges

No. Wedges

No. Triangles

Frequent connected subsets of edges
Triple Sampling from edge stream

• For graph steam, there are two scenarios:
  – Incident edge stream: Stream of the edges incident to some vertices come together, whereas the ordering of vertices is arbitrary
  – Arbitrary edge stream: Stream of the edges appear in arbitrary order

• Sampling subgraphs using streaming method is not easy, existing works only covered subgraphs with three and four vertices

• Large number of works exist that aim to sample triples for approximating triangle counting from stream data
Related Work – Triple Sampling

- **Random Sampling**
  - **Uniform random sampling** – [Tsourakakis et. al KDD’09]
    - Graph Sparsification with probability $p$
    - Chance of sampling a subgraph (e.g., triangle) is very low
    - Estimates suffer from high variance
  - **Wedge Sampling** – [Seshadhri et. al SDM’13, Rahman et. al CIKM’14]
    - Sample vertices, then sample pairs of incident edges (wedges)
    - Output the estimate of the closed wedges (triangles)

Assume graph fits into memory

or

Need to store a large fraction of data
Related Work – Triple Sampling from Stream

- Assume specific order of the stream – [Buriol et. al 2006]
  - Incidence stream model– neighbors of a vertex arrive together in the stream

- Use multiple passes over the stream – [Becchetti et. al KDD’08]

- Single-pass Algorithms
  - **Streaming-Triangles** – [Jha et. al KDD’13]
    - Sample edges using reservoir sampling, then, sample pairs of incident edges (wedges)
    - Scan for closed wedges (triangles)
Related Work – Triple Sampling from Stream

- Assume specific order of the stream – [Buriol et. al 2006]
  - Incidence stream model – neighbors of a vertex arrive together in the stream

- Use multiple passes over the stream – [Becchetti et. al KDD‘08]

- Single-pass Algorithms
  - **Streaming-Triangles** -- [Jha et. al KDD’13]
    - Sample edges using reservoir sampling, then, sample pairs of incident edges (wedges)
    - Scan for closed wedges (triangles)

*Sampling designs used for specific graph properties (triangles) and Not generally applicable to other properties*
Graph-Sample-and-Hold: $gSH(p, q)$

Graph-Sample-and-Hold Framework
$gSH(p, q)$

Input
$e_1, e_2, ..., e_i, ...$

Output

Sampled Edge stream $S$
Stored State $= \mathcal{O}(|S|)$

Graph Sample and Hold: A framework for Big-Graph Analytics-- [Ahmed et. al KDD’14]
Graph-Sample-and-Hold: $gSH(p, q)$

Graph-Sample-and-Hold Framework
$gSH(p, q)$

Input
Edge stream $e_1, e_2, \ldots, e_i, \ldots$

Flip a coin for each edge $e_i$

Output
Sampled Edge stream $S$
Stored State = $O(|S|)$

Graph Sample and Hold: A framework for Big-Graph Analytics-- [Ahmed et. al KDD’14]
Graph-Sample-and-Hold: \( \text{gSH}(p, q) \)

Graph-Sample-and-Hold Framework
\( \text{gSH}(p, q) \)

Input
Edge stream
\( e_1, e_2, \ldots, e_i, \ldots \)

Output
\( p_i \)
\( \mathbb{P}[e_i \text{ is selected} \mid \text{Stored State } S] = p_i \)

Sampled Edge stream \( S \)
Stored State = \( \mathcal{O}(|S|) \)

Graph Sample and Hold: A framework for Big-Graph Analytics-- [Ahmed et. al KDD’14]
Graph-Sample-and-Hold: $gSH(p, q)$

**Input**

Edge stream $e_1, e_2, ..., e_i, ...$

**Graph-Sample-and-Hold Framework**

$gSH(p, q)$

**Sample**

**Output**

Sampled Edge stream $S$

Store State $= \mathcal{O}(|S|)$

$\mathcal{P}[e_i \text{ is selected} | \text{Stored State } S] = p_i$

If $e_i$ is independent $S$

Then $\mathcal{P}[e_i \text{ is selected}] = p$

*Graph Sample and Hold: A framework for Big-Graph Analytics* -- [Ahmed et. al KDD’14]
Graph-Sample-and-Hold: $gSH(p, q)$

Graph-Sample-and-Hold Framework

- Sample
- Hold

Input

Edge stream $e_1, e_2, \ldots, e_i, \ldots$

Output

Sampled Edge stream $S$
Stored State $= \mathcal{O}(|S|)$

- If $e_i$ is independent $S$
  Then $\mathcal{P}[e_i \text{ is selected}] = p$
- If $e_i$ is dependent $S$
  Then $\mathcal{P}[e_i \text{ is selected}] = q$

$\mathcal{P}[e_i \text{ is selected} \mid \text{Stored State } S] = p_i$

Graph Sample and Hold: A framework for Big-Graph Analytics—[Ahmed et. al KDD’14]
Uniform Random Sampling

\[ p_i = p \]

Graph-Sample-and-Hold Framework

\[ g_{SH}(p, q) \]

Sample

Hold

Input

Edge stream

\[ e_1, e_2, \ldots, e_i, \ldots \]

Output

\[ \mathcal{P}[e_i \text{ is selected} \mid \text{Stored State } S] = p \]

Chance to sample a subgraph (e.g., triangle) is very low

Estimates suffer from a high variance
**Graph-Sample-and-Hold**: $g\text{SH}(p, q)$

**Given**: Graph $G = (V, E)$

$E = \{e_1, e_2, \ldots, e_i, \ldots\}$

Edge stream

$e_1, e_2, \ldots, e_i, \ldots$

---

**Input**

**Graph-Sample-and-Hold Framework**: $g\text{SH}(p, q)$

**Sampled Edges** $S$

---

① **Sampling**

② **Estimation**

**Estimated Counts**

[Ahmed et. al KDD’14]
The Sampling

Start with an empty Sample $S$
The Sampling

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

Input

Edge stream $e_1, e_2, \ldots, e_i, \ldots$
1. The Sampling

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

- No
  - Sample
    - $p_i = p$

- Yes
  - Hold
    - $p_i = q$

Edge stream $e_1, e_2, ..., e_i, ...$

$P[e_i \text{ is selected } | \text{ Sample } S] = p_i$
The Sampling

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

- No
  - Sample
    - $p_i = p$
- Yes
  - Hold
    - $p_i = q$

Input

Edge stream

$e_1, e_2, \ldots, e_i, \ldots$

$\mathcal{P}[e_i \text{ is selected } | \text{ Sample } S] = p_i$

If $e_i$ NOT adjacent to $S$
Then $p_i = p$
The Sampling

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

- Yes: $\mathbb{P}[e_i \text{ is selected } | \text{ Sample } S] = p_i$
  - If $e_i$ NOT adjacent to $S$
    - Then $p_i = p$
  - If $e_i$ adjacent $e_j$, such that $e_j \in S$
    - Then $p_i = q$

- No: $p_i = p$
**The Sampling**

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

- **No**
  - Sample
    - $p_i = p$
  - If Head, Update the sample

- **Yes**
  - Hold
    - $p_i = q$

**Edge stream**
$e_1, e_2, \ldots, e_i, \ldots$

$\mathbb{P}[e_i \text{ is selected } | \text{ Sample } S] = p_i$

If $e_i$ NOT adjacent to $S$
Then $p_i = p$

If $e_i$ adjacent $e_j$, such that $e_j \in S$
Then $p_i = q$
The Sampling – \( gSH(p, q) \)

Start with an empty Sample \( S \)

Is \( e_i \) adjacent to Sample \( S \)?

- Yes: \( \mathcal{P}[e_i \text{ is selected } | \text{ Sample } S] = p_i \)
  - If \( e_i \) NOT adjacent to \( S \)
    - If Head, Update the sample
    - Then \( p_i = p \)
  - If \( e_i \) adjacent \( e_j \), such that \( e_j \in S \)
    - Then \( p_i = q \)
  - Add \((e_i, p_i)\) to \( S \)

- No: Sample
  - \( p_i = p \)
  - Flip a coin with prob. \( p_i \)
  - If Head, Update the sample

Update the Sample \( S \)
The Sampling – $g_{SH_T}(p, q)$

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

- **No**
  - Sample
    - $p_i = p$
    - Flip a coin with prob. $p_i$
    - If Head, Update the sample
  - Update the Sample $S$

- **Yes**
  - Hold
    - $p_i = q$
    - Add $(e_i, p_i)$ to $S$

Edge stream $e_1, e_2, \ldots, e_i, \ldots$

Multiple dependencies

$p_i = 1$ if $e_i$ closing triangle in $S$

$p_i = q$ otherwise
Graph-Sample-and-Hold: $g_{SH}(p, q)$

**Given:** Graph $G = (V, E)$

$E = \{e_1, e_2, ..., e_i, ...\}$

**Edge stream**

$e_1, e_2, ..., e_i, ...$

**Input**

Graph-Sample-and-Hold Framework

$g_{SH}(p, q)$

Sampled Edges $S$

① Sampling → ② Estimation

Estimated Counts

[Ahmed et. al KDD’14]
2 The Estimation

We use Horvitz-Thompson statistical estimation framework

- Used for unequal probability sampling

[Horvitz and Thompson-1952]
The Estimation

- We define the selection estimator for an edge $e_i$

$$\hat{S}_i = \frac{1}{p_i}$$

$p_i$ is the sampling probability of edge $e_i$
The Estimation

- We define the selection estimator for an edge $e_i$

\[
\hat{S}_i = \frac{1}{p_i} \quad \rightarrow \quad \hat{S}_i = \frac{H_i}{p_i}
\]

Indicator variable
$H_i = 1$, if $e_i \in S$
$H_i = 0$, o.w.

$p_i$ is the sampling probability of edge $e_i$
The Estimation

- We define the selection estimator for an edge $e_i$

\[
\hat{S}_i = \frac{1}{p_i} \quad \rightarrow \quad \hat{S}_i = \frac{H_i}{p_i}
\]

- Sampled stream

\[
\hat{S}_1 = 0 \quad \hat{S}_2 = \frac{1}{p} \quad \hat{S}_{15} = \frac{1}{q}
\]

$p_i$ is the sampling probability of edge $e_i$
We define the selection estimator for an edge $E_i$ as:

$$\hat{S}_i = \frac{1}{p_i} \quad \Rightarrow \quad \hat{S}_i = \frac{H_i}{p_i}$$

Indicator variable:
- $H_i = 1$, if $e_i \in S$
- $H_i = 0$, o.w.

Unbiasedness:

$$E[\hat{S}_i] = \frac{E[H_i]}{p_i} = 1$$  

[Theorem 1 (i)]
We define the selection estimator for an edge $e_i$

$$\hat{S}_i = \frac{1}{p_i} \quad \rightarrow \quad \hat{S}_i = \frac{H_i}{p_i}$$

Unbiasedness

$E[\hat{S}_i] = \frac{E[H_i]}{p_i} = 1$

We derive the estimate of edge count

$$\hat{N}_K = \sum_{e_i \in S} \frac{1}{p_i}$$
The Estimation

- We generalize the equations to any subset of edges $J$. 
2. The Estimation

- We generalize the equations to any subset of edges $J$

\[ J = \{ e_{j_1}, e_{j_2}, \ldots e_{j_i}, \ldots \} \]

\[ \hat{S}(J) = \prod_{j_i \in J} \hat{S}_{j_i} \quad \rightarrow \quad \mathbb{E}[\hat{S}(J)] = 1 \]

**Unbiasedness**

[Theorem 1 (ii)]
② The Estimation

- We **generalize** the equations to any subset of edges $J$

$$J = \{ e_{j_1}, e_{j_2}, \ldots, e_{j_i}, \ldots \}$$

$$\hat{S}(J) = \prod_{j_i \in J} \hat{S}_{j_i} \quad \Rightarrow \quad \mathbb{E}[\hat{S}(J)] = 1$$

- **Ex:** The estimator of a sampled triangle is:

$$\hat{S}(\tau_j) = \hat{S}_{j_1} \cdot \hat{S}_{j_2} \cdot \hat{S}_{j_3}$$

$$\hat{S}(\tau_j) = \prod_{j_i \in \tau_j} \frac{1}{p_{j_i}}$$

Unbiasedness
[Theorem 1 (ii)]
The Estimation

- The estimate of triangle counts is
  \[ \hat{N}_T = \sum_{\tau_j \in \hat{T}} \hat{S}(\tau_j) \]

- The estimate of wedge counts is
  \[ \hat{N}_\Lambda = \sum_{L_j \in \hat{\Lambda}} \hat{S}(L_j) \]

- Using the gSH framework, we obtain estimators for:
  - Edge Count
  - Triangle Count
  - Wedge Count
  - Global clustering Coeff. = \( 3 * \begin{array}{c} \text{triangle} \\ \text{edge} \end{array} \) / \( \begin{array}{c} \text{wedge} \\ \text{triangle} \end{array} \)
2 The Estimation

- The estimate of triangle counts is

\[ \hat{N}_T = \sum_{\tau_j \in \hat{T}} \hat{S}(\tau_j) \]

- The estimate of wedge counts is

\[ \hat{N}_\Lambda = \sum_{L_j \in \hat{\Lambda}} \hat{S}(L_j) \]

- Using the gSH framework, we obtain estimators for:
  - Edge Count
  - Triangle Count
  - Wedge Count
  - Global clustering Coeff. = \( 3 \times \triangle / \triangle \)

We also compute HT unbiased variance estimators.
Relative Error = \frac{|estimated - actual|}{actual}

### Edge Count

| Sample          | \(N_K\)  | \(\hat{N}_K\) | \(|\hat{N}_K - N_K| / N_K\) |
|-----------------|----------|--------------|-------------------------------|
| socfb-CMU       | 249.9K   | 249.6K       | 0.0013                        |
| socfb-UCLA      | 747.6K   | 751.3K       | 0.0050                        |
| socfb-Wisconsin | 835.9K   | 835.7K       | 0.0003                        |
| web-Stanford    | 1.9M     | 1.9M         | 0.0004                        |
| web-Google      | 4.3M     | 4.3M         | 0.0007                        |
| web-BerkStan    | 6.6M     | 6.6M         | 0.0006                        |

### Triangle Count

| Sample          | \(N_T\)  | \(\hat{N}_T\) | \(|\hat{N}_T - N_T| / N_T\) |
|-----------------|----------|--------------|-------------------------------|
| socfb-CMU       | 2.3M     | 2.3M         | 0.0003                        |
| socfb-UCLA      | 5.1M     | 5.1M         | 0.0095                        |
| socfb-Wisconsin | 4.8M     | 4.8M         | 0.0058                        |
| web-Stanford    | 11.3M    | 11.3M        | 0.0023                        |
| web-Google      | 13.3M    | 13.4M        | 0.0029                        |
| web-BerkStan    | 64.6M    | 65M          | 0.0063                        |

### Wedge Count

| Sample          | \(N_A\)  | \(\hat{N}_A\) | \(|\hat{N}_A - N_A| / N_A\) |
|-----------------|----------|--------------|-------------------------------|
| socfb-CMU       | 37.4M    | 37.3M        | 0.0018                        |
| socfb-UCLA      | 107.1M   | 107.8M       | 0.0060                        |
| socfb-Wisconsin | 121.4M   | 121.2M       | 0.0018                        |
| web-Stanford    | 3.9T     | 3.9T         | 0.0004                        |
| web-Google      | 727.4M   | 724.3M       | 0.0042                        |
| web-BerkStan    | 27.9T    | 27.9T        | 0.0002                        |

### Global Clustering

| Sample          | \(\alpha\) | \(\tilde{\alpha}\) | \(|\tilde{\alpha} - \alpha| / \alpha\) |
|-----------------|------------|-------------------|-------------------|
| socfb-CMU       | 0.18526    | 0.18574           | 0.00260           |
| socfb-UCLA      | 0.14314    | 0.14363           | 0.00340           |
| socfb-Wisconsin | 0.12013    | 0.12101           | 0.00730           |
| web-Stanford    | 0.00862    | 0.00862           | 0.00020           |
| web-Google      | 0.05523    | 0.05565           | 0.00760           |
| web-BerkStan    | 0.00694    | 0.00698           | 0.00680           |

Sample size <= 40K edges
Relative Error = \textless 1\% 

Over all graphs, all properties

| Sample size \(\leq 40K\) edges |

### Edge Count

| Graph                  | \(N_K\)  | \(\hat{N}_K\) | \(\frac{|\hat{N}_K - N_K|}{N_K}\) |
|------------------------|----------|---------------|----------------------------------|
| socfb-CMU              | 249.9K   | 249.6K        | 0.0013                           |
| socfb-UCLA             | 747.6K   | 751.3K        | 0.0050                           |
| socfb-Wisconsin        | 835.9K   | 835.7K        | 0.0003                           |
| web-Stanford           | 1.9M     | 1.9M          | 0.0004                           |
| web-Google             | 4.3M     | 4.3M          | 0.0007                           |
| web-BerkStan           | 6.6M     | 6.6M          | 0.0006                           |

### Triangle Count

| Graph                  | \(N_T\)  | \(\hat{N}_T\) | \(\frac{|\hat{N}_T - N_T|}{N_T}\) |
|------------------------|----------|---------------|----------------------------------|
| socfb-CMU              | 2.3M     | 2.3M          | 0.0003                           |
| socfb-UCLA             | 5.1M     | 5.1M          | 0.0095                           |
| socfb-Wisconsin        | 4.8M     | 4.8M          | 0.0058                           |
| web-Stanford           | 11.3M    | 11.3M         | 0.0023                           |
| web-Google             | 13.3M    | 13.4M         | 0.0029                           |
| web-BerkStan           | 64.6M    | 65M           | 0.0063                           |

### Wedge Count

| Graph                  | \(N_\Lambda\) | \(\hat{N}_\Lambda\) | \(\frac{|\hat{N}_\Lambda - N_\Lambda|}{N_\Lambda}\) |
|------------------------|---------------|-----------------------|----------------------------------|
| socfb-CMU              | 37.4M         | 37.3M                 | 0.0018                           |
| socfb-UCLA             | 107.1M        | 107.8M                | 0.0060                           |
| socfb-Wisconsin        | 121.4M        | 121.2M                | 0.0018                           |
| web-Stanford           | 3.9T          | 3.9T                  | 0.0004                           |
| web-Google             | 727.4M        | 724.3M                | 0.0042                           |
| web-BerkStan           | 27.9T         | 27.9T                 | 0.0002                           |

### Global Clustering

| Graph                  | \(\alpha\)   | \(\hat{\alpha}\) | \(\frac{|\hat{\alpha} - \alpha|}{\alpha}\) |
|------------------------|---------------|-------------------|----------------------------------|
| socfb-CMU              | 0.18526       | 0.18574           | 0.00260                          |
| socfb-UCLA             | 0.14314       | 0.14363           | 0.00340                          |
| socfb-Wisconsin        | 0.12013       | 0.12101           | 0.00730                          |
| web-Stanford           | 0.00862       | 0.00862           | 0.00020                          |
| web-Google             | 0.05523       | 0.05565           | 0.00760                          |
| web-BerkStan           | 0.00694       | 0.00698           | 0.00680                          |
Dataset:
facebook friendship graph at UCLA

Estimated
Actual
Conclusion

- **Full Access**
- **Restricted Access**
- **Data Stream Access**

Sampling Objective:
- Estimate node/edge properties in original network
- Sample representative subgraph/subnetwork
- Estimate frequency of subgraph patterns

Data Access Assumption:
- **Full Access**
- **Restricted Access**

Algorithms covered in tutorial:

Sampling methodology:
- **Sampling methodology**
Concluding remarks

• Generally for many tasks, you will need uniform sampling of “objects” from the network (eg. Task 1 and Task 3 in our slides)
  
  – Uniform sampling with full access is generally easier for node or edge sampling,

  – sampling higher order structure is still hard because the sampling space is not instantly realizable without brute-force enumeration.

  – For uniform sampling under access restriction, both M-H, and SRW-weighted (with bias correction) works; both guarantee uniformity by theory, but SRW-weighted provides marginally better empirical sampling accuracy.
Concluding remarks

- For sampling a representative sub-networks (Task 2), unbiased sampling is typically NOT a good choice. From empirical observation, biased sampling helps obtain better representative structure.
  - It is difficult to select a subgraph that preserves ALL properties, so sampling mechanism should focus on properties of interest.
  - Theory has generally focused on kind of nodes that are sampled. More works are needed to show the connection between types of nodes sampled and resulting higher order topological structure.
Concluding remarks (cont.)

• In present days, streaming is becoming more natural data access mechanism for graphs. So, currently it is an active area of research

  – It presents unique challenges to sample topology in an unbiased way, because local views of stream do not necessarily reflect connectivity.

  – Existing works have solved some isolated problems exactly (eg. triangle sampling, pagerank computation), and some other empirically (eg. representative sub-network)
Given a new domain, how should you sample the network for analysis?
Define your task

Select sampling objective

Consider data access restrictions

Choose sampling method

Evaluate sampling quality
References (1)

- Jure Leskovec and Christos Faloutsos. 2006. “Sampling from Large Graphs”. In proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’06, pp. 631-636
- Bhuiyan, M.A.; Rahman, M.; Rahman, M.; Al Hasan, M., "GUISE: Uniform Sampling of Graphlets for Large Graph Analysis," Data Mining (ICDM), 2012 IEEE 12th International Conference on, pp.91-100
References (2)


• Arun Maiya and Tanya Berger-Wolf. 2010. “Sampling Community Structure”. In Proc. of the 19th International Conference on World wide web, WWW ’10, pp. 701-710


References (3)


• Jeffrey S. Vitter. 1985, “Random sampling with a reservoir”. ACM Trans. Math. Softw. 11, 1


Thank you!

Questions?