Graph Sample and Hold: A Framework for Big Graph Analytics

Nesreen K. Ahmed
Department of Computer Science, Purdue University

Joint work with

Nick Duffield\textsuperscript{1}, Jennifer Neville\textsuperscript{2}, Ramana Kompella\textsuperscript{2}

\textsuperscript{1}Texas A&M University, \textsuperscript{2}Purdue University
Graphs: Rich Data Representation

- Social Network
- Internet (AS)
- Political Blogs
- Biological

Graph Analytics
Studying and analyzing complex networks is a **challenging** and **computational intensive** task

- Today’s networks are massive in size
  - e.g., online social networks have billions of users

- Today’s networks are dynamic/streaming over time
  - e.g., Twitter streams, email communications
Studying and analyzing complex networks is a **challenging and computational intensive** task.

- Today’s networks are massive in size
  - e.g., online social networks have billions of users

- Today’s networks are dynamic/streaming over time
  - e.g., Twitter streams, email communications

Due to these challenges, we usually need to **sample**

![Diagram](image.png)

**Graph G**

**Statistical Sampling**

**e.g. Uniform Random Sampling**

**Sample S**
Motivation

Given a large graph $G$ represented as a stream of edges $e_1, e_2, e_3\ldots$

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

Edge Stream
Motivation

Given a large graph $G$ represented as a stream of edges $e_1, e_2, e_3 \ldots$

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

Sampled Edge Stream

Sampled Edge Stream
Motivation

Given a large graph $G$ represented as a stream of edges $e_1, e_2, e_3 \ldots$

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

Sampled Edge Stream

No. Edges  No. Wedges  No. Triangles

Frequent connected subsets of edges
Given a large graph $G$ represented as a stream of edges $e_1, e_2, e_3\ldots$

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

<table>
<thead>
<tr>
<th>No. Edges</th>
<th>No. Wedges</th>
<th>No. Triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transitivity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frequent connected subsets of edges
Related Work – 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]
    - Sample vertices, then sample pairs of incident edges (wedges)
    - Output the estimate of the closed wedges (triangles)
Related Work – 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]
    - 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 – Stream Sampling

- 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 – Stream Sampling

- 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: \( g_{SH}(p, q) \)

Graph-Sample-and-Hold Framework: \( g_{SH}(p, q) \)

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

Input

Output

Sampled Edge stream \( S \)
Stored State = \( O(|S|) \)
**Graph-Sample-and-Hold**: $gSH(p, q)$

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

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

Flip a coin for each edge $e_i$

Output

Sampled Edge stream $S$
Stored State = $\mathcal{O}(|S|)$
Graph-Sample-and-Hold: $g_{\text{SH}}(p, q)$

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

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

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

Sampled Edge stream $S$
Stored State = $O(|S|)$
Graph-Sample-and-Hold: $g_{\text{SH}}(p, q)$

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

Input

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

Sample

Output

Sampled Edge stream $S$

$\mathcal{O}(|S|)$

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

If $e_i$ is independent $S$

Then $\mathcal{P}[e_i \text{ is selected}] = p$
Graph-Sample-and-Hold: $g_{\text{SH}}(p, q)$

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

Input

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

Sample

Hold

Output

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

if $e_i$ is independent $S$

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

if $e_i$ is dependent $S$

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

Sampled Edge stream $S$

Stored State $= O(|S|)$
Graph-Sample-and-Hold: $gSH(p, q)$

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

Output
Sample
Hold

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

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

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

If $e_i$ is dependent $S$
Then $P[e_i \text{ is selected}] = q$
Uniform Random Sampling

\[ p_i = p \]

Graph-Sample-and-Hold Framework

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

Sample

Hold

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

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

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

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

\( E = \{ e_1, e_2, ..., e_i, ... \} \)

1. **Sampling**
2. **Estimation**

- Edge stream
  \( e_1, e_2, ..., e_i, ... \)

- Sampled Edges \( S \)

- Estimated Counts
The Sampling

Start with an empty Sample $S$
The Sampling

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

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

Start with an empty Sample $S$

Is $e_i$ adjacent to Sample $S$?

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

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

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

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

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 $S$?

No

Sample

$p_i = p$

Yes

Hold

$p_i = q$

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

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

If $e_i$ NOT adjacent to $S$

Then $p_i = p$

If $e_i$ adjacent $e_j$, such that $e_j \in S$

Then $p_i = q$
1. **The Sampling**

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

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

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

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

- 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 \( S \)

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

If \( e_i \) 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 \( p_i = q \)
    - If \( e_i \) NOT adjacent to \( S \)
      - Then \( p_i = p \)
    - If \( e_i \) adjacent \( e_j \), such that \( e_j \in S \)
      - Then \( p_i = q \)
    - Add \( (e_i, p_i) \) to \( S \)
① 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

- **Yes**
  - Hold
  - $p_i = q$
  - $p_i = 1$ if $e_i$ closing triangle in $S$
  - $p_i = q$ otherwise

Add $(e_i, p_i)$ to $S$

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

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

Indicator variable:

$H_i = 1$, if $e_i \in S$

$H_i = 0$, o.w.

Sampled stream:

$\hat{S}_1 = 0$  $\hat{S}_2 = \frac{1}{p}$  $\hat{S}_{15} = \frac{1}{q}$

$p_i$ 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.

- Unbiasedness
  - [Theorem 1 (i)]

\[
E[\hat{S}_i] = \frac{E[H_i]}{p_i} = 1
\]
2 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.

Unbiasedness
[Theorem 1 (i)]

- We derive the estimate of edge count

$$\hat{N}_K = \sum_{e_i \in S} \frac{1}{p_i}$$
We generalize the equations to any subset of edges $J$. 

② The Estimation
② The Estimation

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

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

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

*Unbiasedness*
[Theorem 1 (ii)]
2. The Estimation

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

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

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

- Ex: The estimator of a sampled triangle is:

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

\[ \hat{S}(\tau_j) = \prod_{j_i \in \tau_j} \frac{1}{p_{j_i}} \]
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. \( = 3 \times \frac{\triangle}{\Lambda} \)
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 * \[ \text{Triangle Count} \] / \[ \text{Wedge Count} \]

We also compute HT unbiased variance estimators
## Experiments

6 graphs with 7K – 685K nodes, 250K– 7M edges

<table>
<thead>
<tr>
<th>Graphs</th>
<th>Edge Count</th>
<th>Triangle Count</th>
<th>Wedge Count</th>
<th>Global Clustering Coeff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social facebook Graphs</td>
<td>CMU, UCLA, and Wisconsin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Graphs</td>
<td>Google, Berkeley-Stanford, Stanford</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Social facebook Graphs: friends

Web Graphs: Links to
Relative Error = \frac{|\text{estimated} - \text{actual}|}{\text{actual}}

### Edge Count

| Dataset          | \(N_K\)   | \(\hat{N}_K\) | \(\frac{|\hat{N}_K - N_K|}{N_K}\) |
|------------------|-----------|---------------|-----------------------------------|
| 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

| Dataset          | \(N_T\)   | \(\hat{N}_T\) | \(\frac{|\hat{N}_T - N_T|}{N_T}\) |
|------------------|-----------|---------------|-----------------------------------|
| 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

| Dataset          | \(N_\Lambda\) | \(\hat{N}_\Lambda\) | \(\frac{|\hat{N}_\Lambda - N_\Lambda|}{N_\Lambda}\) |
|------------------|---------------|---------------------|-----------------------------------|
| 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

| Dataset          | \(\alpha\)  | \(\hat{\alpha}\) | \(\frac{|\hat{\alpha} - \alpha|}{\alpha}\) |
|------------------|--------------|------------------|-----------------------------------|
| 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                           |

Sample size <= 40K edges
Relative Error = <1%
Over all graphs, all properties

Table 1: Estimation on a path of length 3 using gSH

| Graph       | \(N_K\) | \(\hat{N}_K\) | \(|\hat{N}_K - N_K|\) / \(N_K\) |
|-------------|---------|---------------|-------------------------------|
| 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                        |

Table 2: Statistics of datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Edges</th>
<th>Nodes</th>
<th>Triangles</th>
<th>Density</th>
<th>SSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-Stanford</td>
<td>11.3M</td>
<td>3.7M</td>
<td>18.8M</td>
<td>0.0023</td>
<td>14.8K</td>
</tr>
<tr>
<td>socfb-CMU</td>
<td>107.8M</td>
<td>100.1M</td>
<td>115.42M</td>
<td>0.0060</td>
<td>5K</td>
</tr>
<tr>
<td>socfb-UCLA</td>
<td>11.3M</td>
<td>3.7M</td>
<td>18.8M</td>
<td>0.0023</td>
<td>5K</td>
</tr>
<tr>
<td>socfb-Wisconsin</td>
<td>4.8M</td>
<td>4.8M</td>
<td>5.7M</td>
<td>0.0058</td>
<td>5K</td>
</tr>
<tr>
<td>web-Google</td>
<td>13.4M</td>
<td>13.3M</td>
<td>859.1K</td>
<td>0.0029</td>
<td>812.2K</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>65M</td>
<td>64.6M</td>
<td>14.8K</td>
<td>0.0063</td>
<td>14.8K</td>
</tr>
</tbody>
</table>

Table 3: Estimates of expected value, relative error, sample size lower, upper bound.

| Graph       | \(N_T\) | \(\hat{N}_T\) | \(|\hat{N}_T - N_T|\) / \(N_T\) |
|-------------|---------|---------------|-------------------------------|
| 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 | 13.4M   | 13.4M         | 0.0029                        |
| web-BerkStan| 65M     | 65M           | 0.0063                        |

Global Clustering

| Graph       | \(\alpha\) | \(\hat{\alpha}\) | \(|\hat{\alpha} - \alpha|\) / \(\alpha\) |
|-------------|------------|-----------------|--------------------------------|
| 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                        |

Sample size <= 40K edges
for all possible samples with Google, web-Stanford, socfb-Wisconsin, and socfb-CMU graphs.

Estimated / Actual

Edge Count

\[ \hat{N}_K / N_K \]

Triangle Count

\[ \hat{N}_T / N_T \]

Wedge Count

\[ \hat{N}_\Lambda / N_\Lambda \]

Global Clustering

\[ \hat{\rho} / \alpha \]

Datasets:
- facebook friendship graph at UCLA

Actual
- Estimated/Actual
- Confidence Upper & Lower Bounds
- Sample Size = 40K edges
Comparison to previous work

- Comparison to Streaming-Triangles [Jha et. al-KDD’13]

<table>
<thead>
<tr>
<th>Graph</th>
<th>Jha et al.</th>
<th>gSH</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-Stanford</td>
<td>0.07</td>
<td>0.0023</td>
<td>14.8K 5% of edges</td>
</tr>
<tr>
<td>web-Google</td>
<td>0.04</td>
<td>0.0029</td>
<td>25.2K 0.6% of edges</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>0.12</td>
<td>0.0063</td>
<td>39.8K 0.57% of edges</td>
</tr>
</tbody>
</table>

Relative Error = |estimated−actual|actual
92% - 96% Improvement

- Comparison to Streaming-Triangles [Jha et. al-KDD’13]

<table>
<thead>
<tr>
<th>Graph</th>
<th>Jha et al.</th>
<th>gSH</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-Stanford</td>
<td>≈ 0.07</td>
<td>0.0023</td>
<td>40K</td>
</tr>
<tr>
<td>web-Google</td>
<td>≈ 0.04</td>
<td>0.0029</td>
<td>40K</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>≈ 0.12</td>
<td>0.0063</td>
<td>40K</td>
</tr>
</tbody>
</table>

Relative Error = \frac{|\text{estimated} - \text{actual}|}{\text{actual}}
Conclusion

- A sample is *representative* if graph properties of interest can be estimated with a known degree of accuracy.

- We proposed a generic framework **Graph Sample and Hold (gSH)**
  - gSH is an *efficient* single-pass streaming framework.
  - gSH selects a *representative* sample and computes *unbiased* estimates of counts of connected subsets of edges (e.g., triangles, wedges ...)
  - Theoretical properties of gSH are supported by empirical analysis.

- gSH admits generalizations by allowing the *dependence* of the sampling process to *vary* based on the *stored* state.

- gSH has a relative estimation error < 1% for a sample size <40K edges.
Future Work

- Extend gSH to adaptively change the parameters \((p,q)\), to maintain a fixed size stored state

- Extend gSH to estimate graphlets and induced subgraphs with more than three nodes
Thank you!

Questions?

Nesreen Ahmed, Nick Duffield, Jennifer Neville, Ramana Kompella

{nkahmed, neville, kompella} @ cs.purdue.edu, duffieldng@tamu.edu