KDD'14, New York
August 25th, 2014

Graph Sample and Hold: A Framework for Big Graph Analytics

Nesreen K. Ahmed

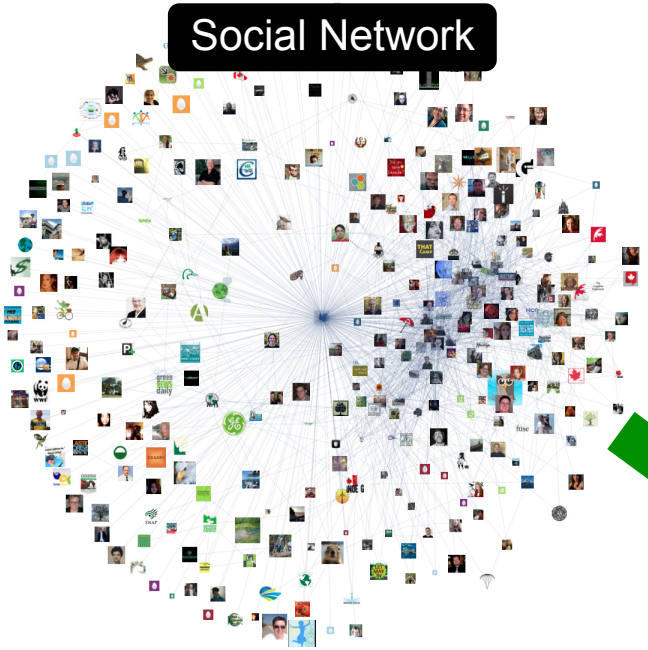
Department of Computer Science, Purdue University

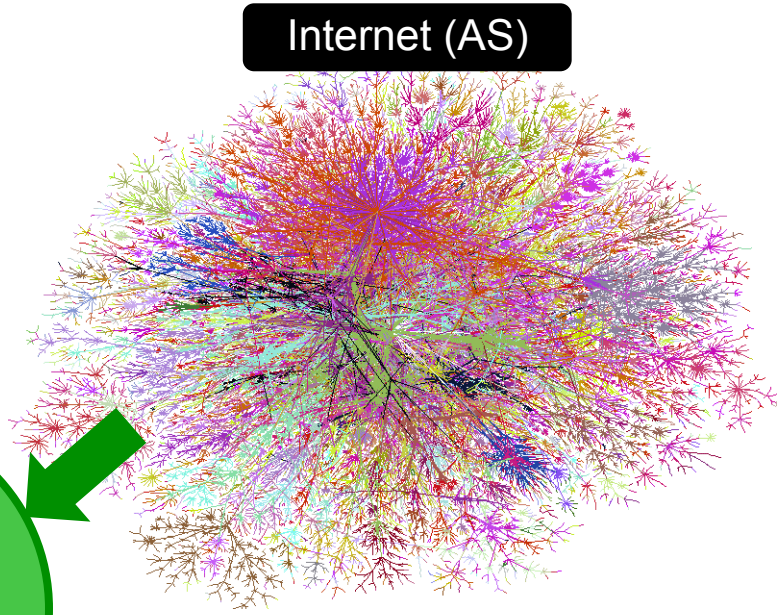
Joint work with

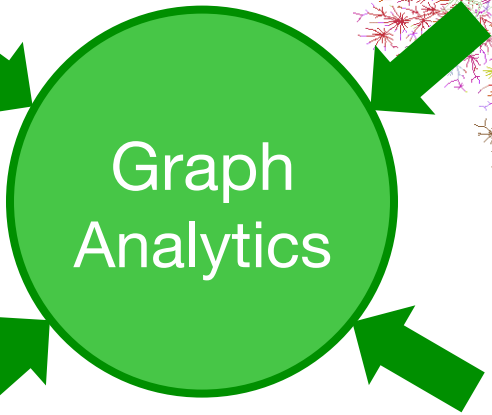
Nick Duffield[1], Jennifer Neville[2], Ramana Kompella[2]

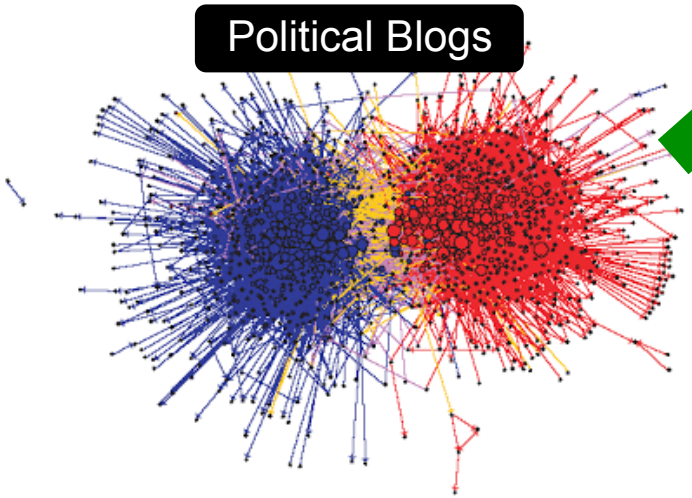
[1]Texas A&M University, [2]Purdue University

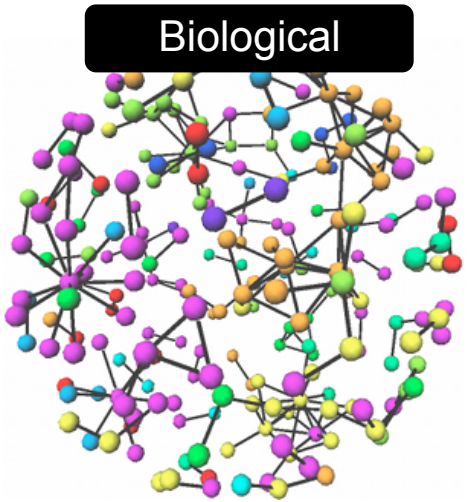
Graphs: Rich Data Representation











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

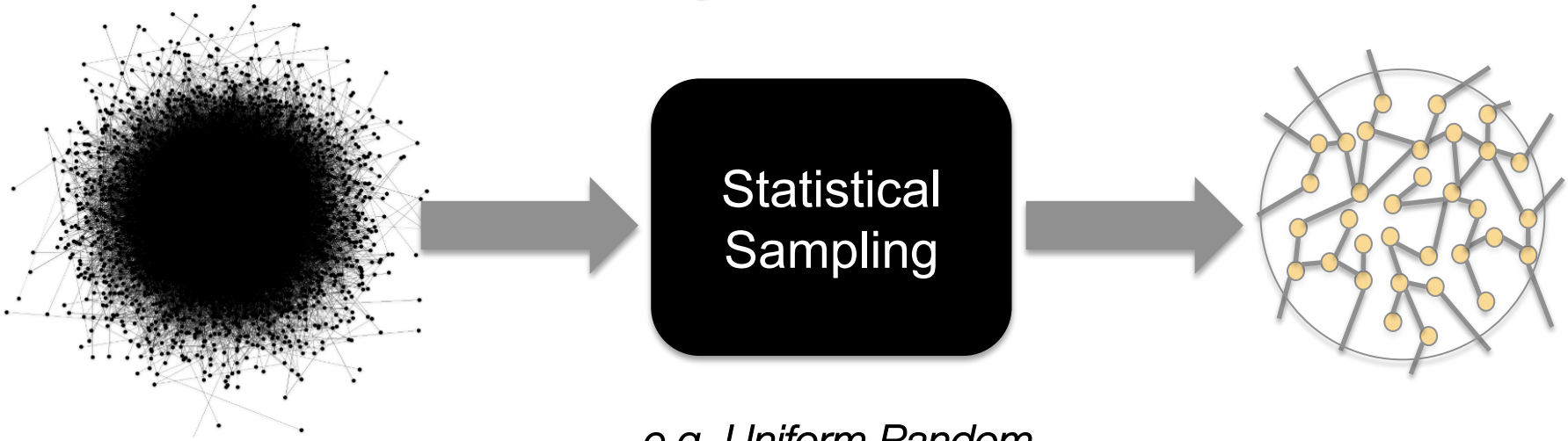
Studying and analyzing complex networks

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***



e.g. Uniform Random Sampling

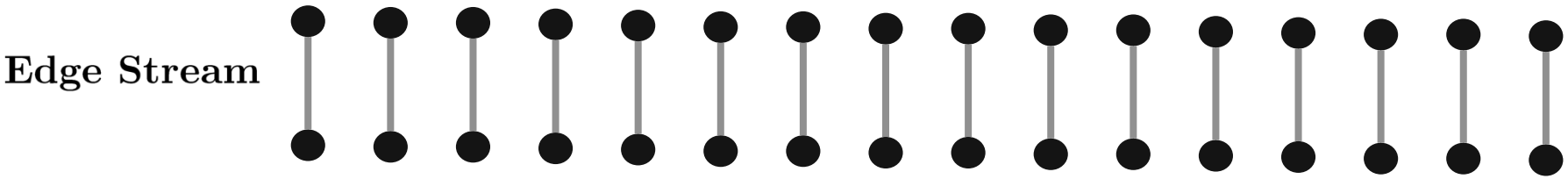
Graph **G**

Sample **S**

Motivation

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

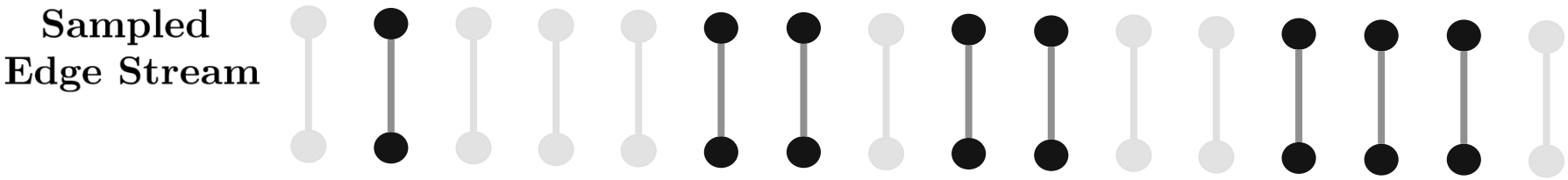
We show how to **efficiently sample** from G while limiting memory space to calculate **unbiased estimates** of various graph properties



Motivation

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

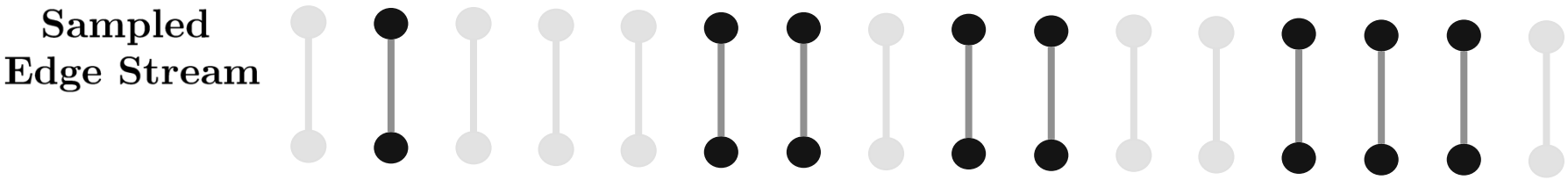
We show how to **efficiently sample** from G while limiting memory space to calculate **unbiased estimates** of various graph properties

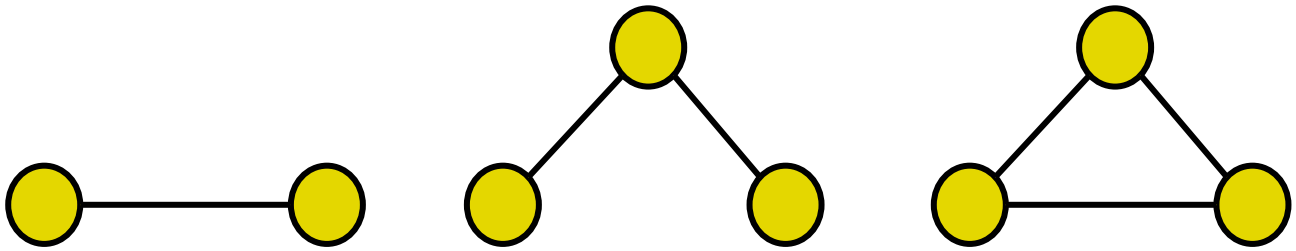


Motivation

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





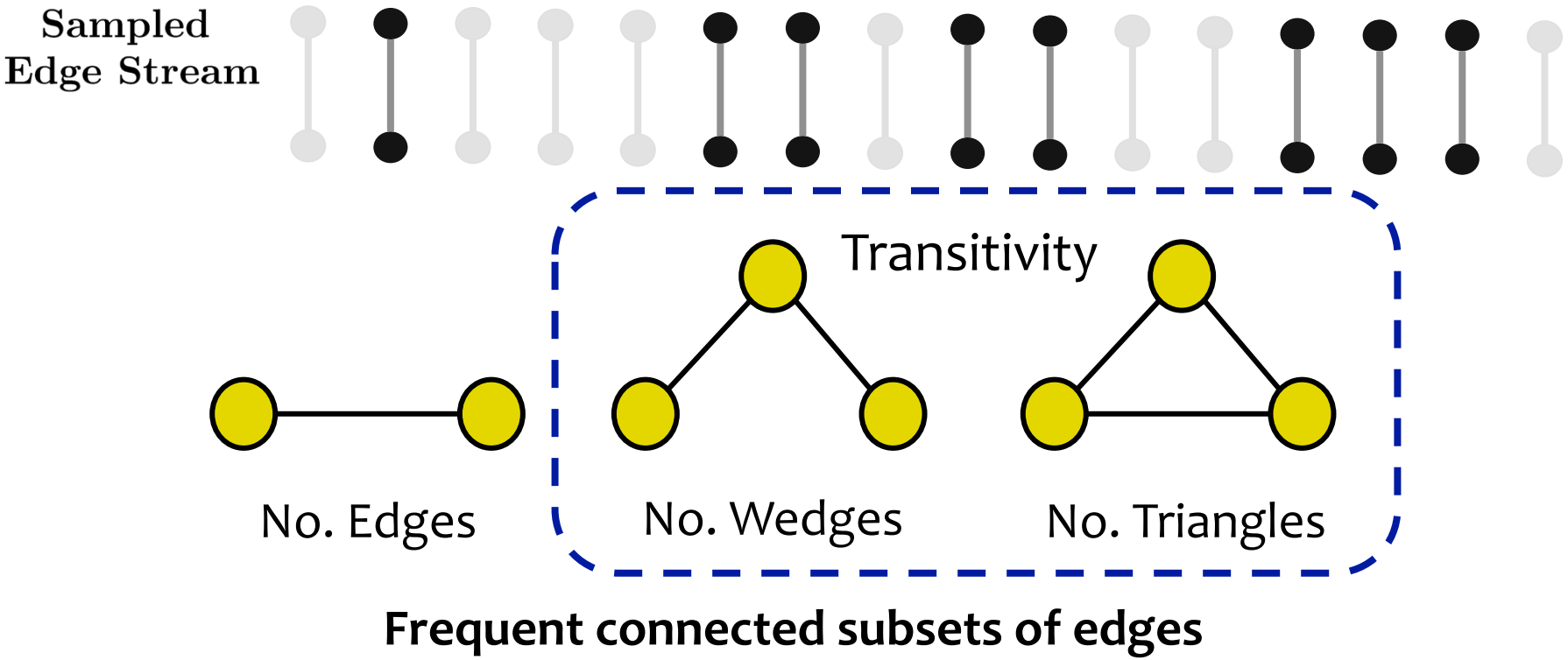
No. Edges No. Wedges No. Triangles

Frequent connected subsets of edges

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



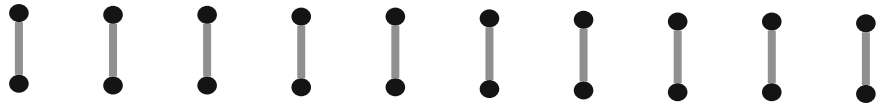
Related Work – Sampling

- 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

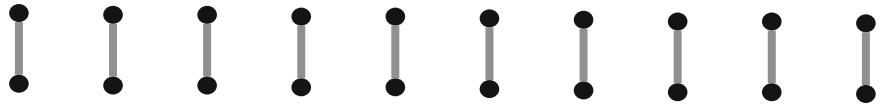
- Random Sampling
  - Uniform random sampling – [Tsourakakis et. al KDD'09]
    - Graph Sparsification with probability p
    - Chance of sampling a subgraph (e.g., triangle) is very low
    - Estimates suffer from high variance
  - Wedge Sampling – [Seshadhri et. al SDM'13]
    - Sample vertices, then 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)

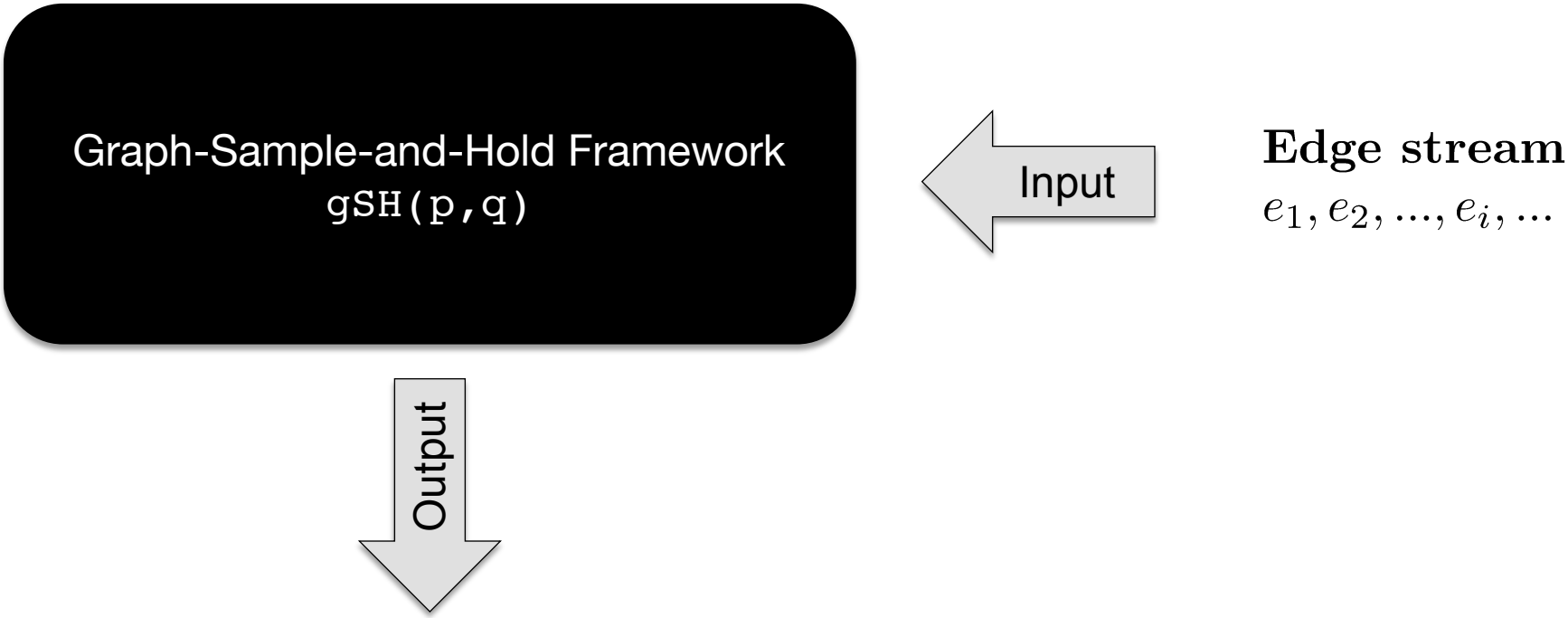


Related Work – Stream Sampling

- 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 : gSH(p,q)



Graph-Sample-and-Hold : gSH(p,q)

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

Input

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

Flip a coin for each edge e_i

Output

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

Graph-Sample-and-Hold : gSH(p,q)

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

Input

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

Output

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

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

Graph-Sample-and-Hold : gSH(p,q)

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

Input

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

Output

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

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

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

Graph-Sample-and-Hold : gSH(p,q)

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

Sample | Hold

Input

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

Output

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

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

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

If e_i is dependent S
Then $\mathcal{P}[e_i \textbf{ is selected}] = q$

Graph-Sample-and-Hold : gSH(p,q)

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

Sample

Hold

Input

Edge stream

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

Output

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

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

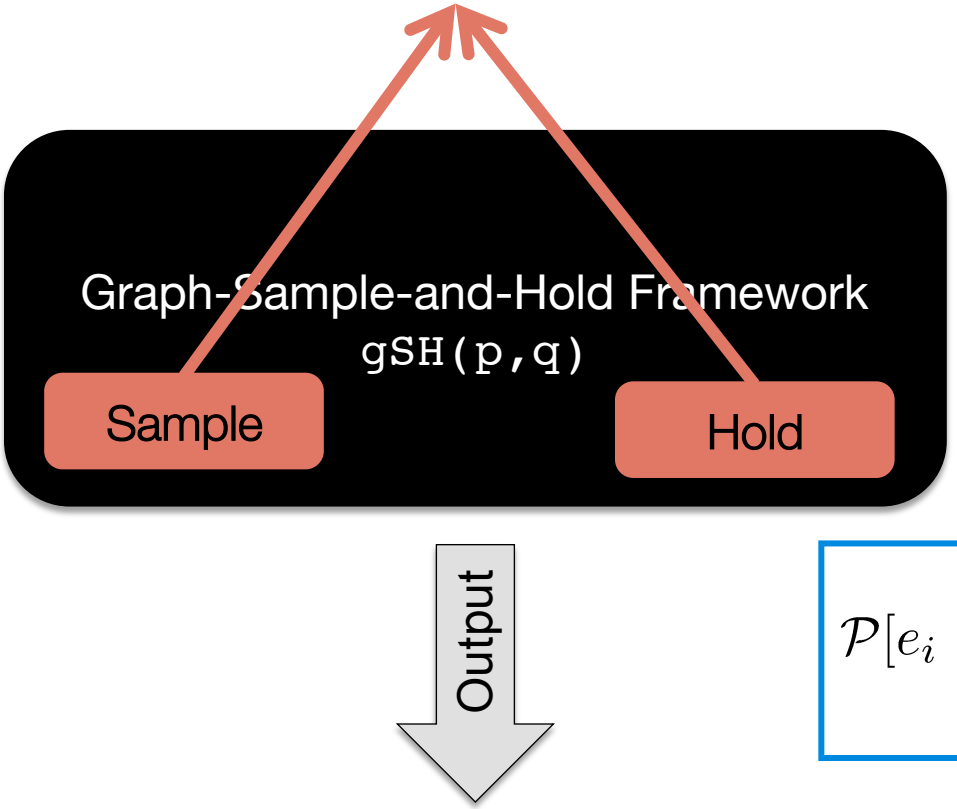
If e_i is dependent S
Then $\mathcal{P}[e_i \textbf{ is selected}] = q$

Sampled Edge stream S

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

Uniform Random Sampling

$$p_i = p$$



Input

Edge stream

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

Output

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

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

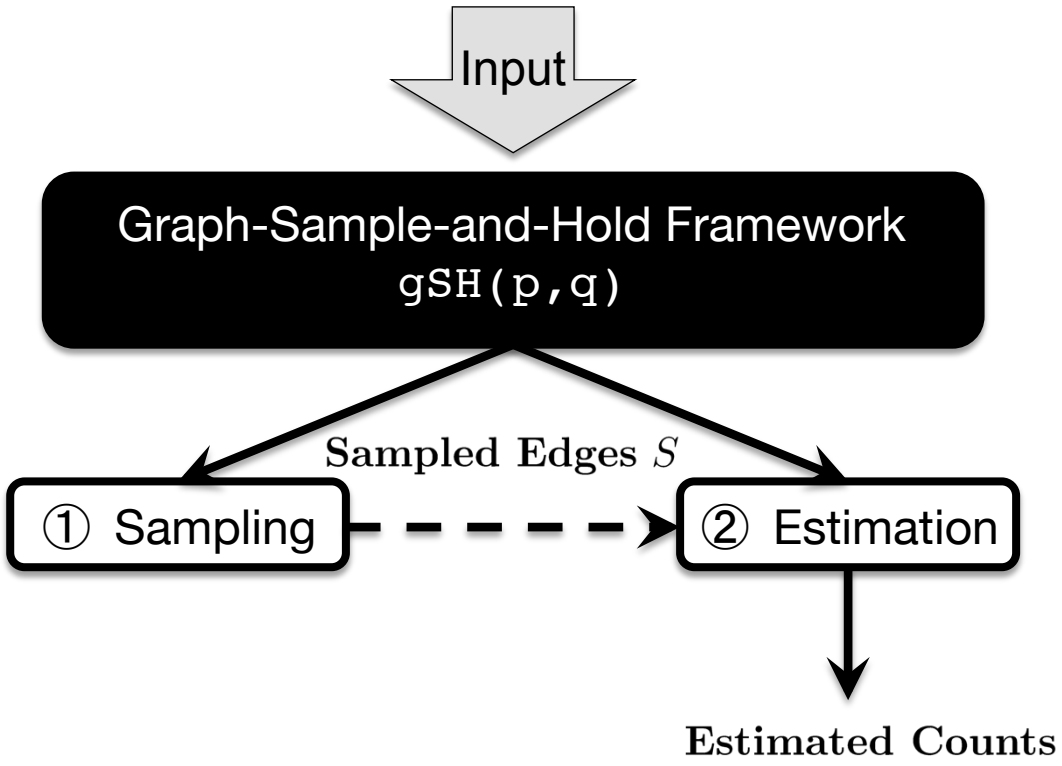
Estimates suffer from a high variance

Graph-Sample-and-Hold : gSH(p,q)

Given: Graph $G = (V, E)$
$E = \{e_1, e_2, ..., e_i, ...\}$

Edge stream

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



① The Sampling

Start with an empty Sample S

① The Sampling

Start with an empty Sample S

Is e_i adjacent to Sample S ?

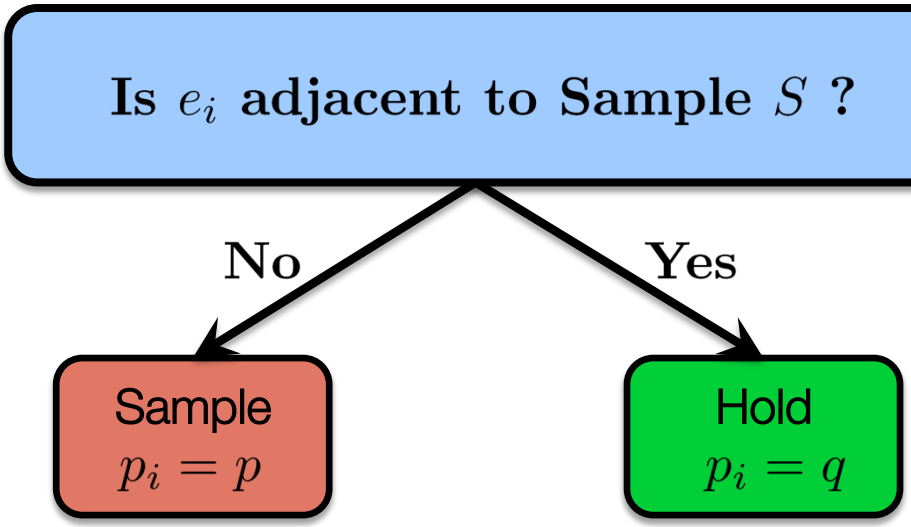
Input

Edge stream

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

① The Sampling

Start with an empty Sample S



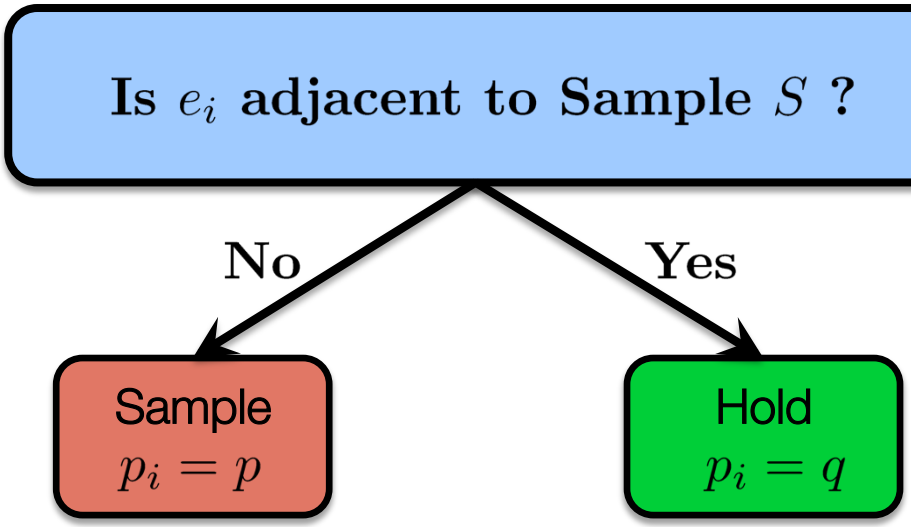
Edge stream

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

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

① The Sampling

Start with an empty Sample S



Edge stream

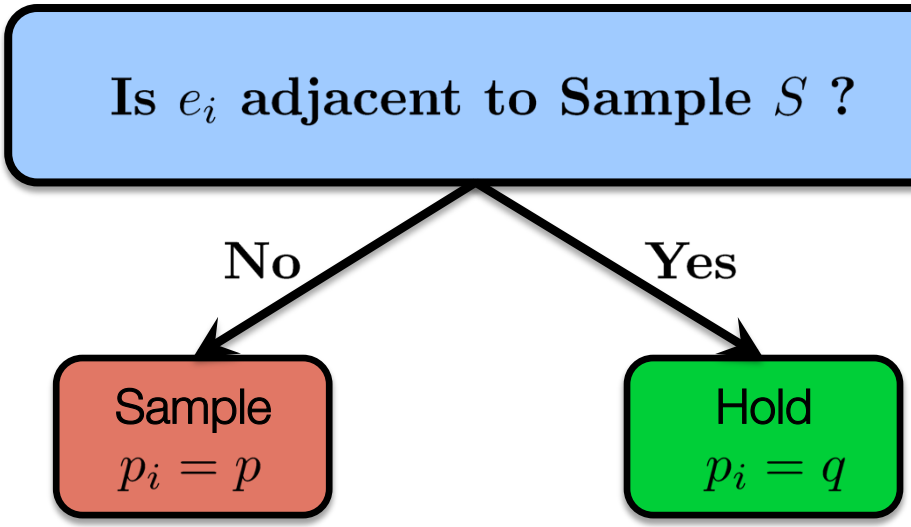
$e_1, e_2, ..., e_i, ...$

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

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

① The Sampling

Start with an empty Sample S



Edge stream

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

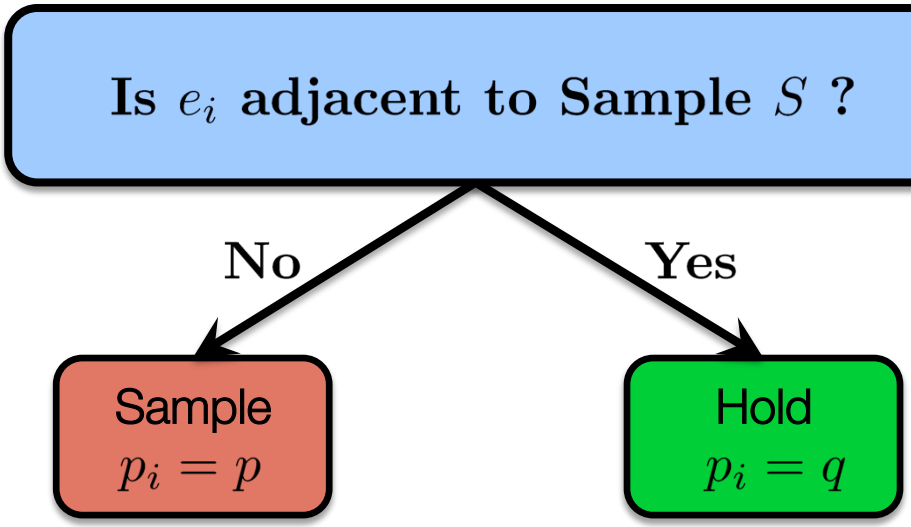
$\mathcal{P}[e_i$ **is selected** $|$ **Sample** $S] = p_i$

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



Edge stream

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

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

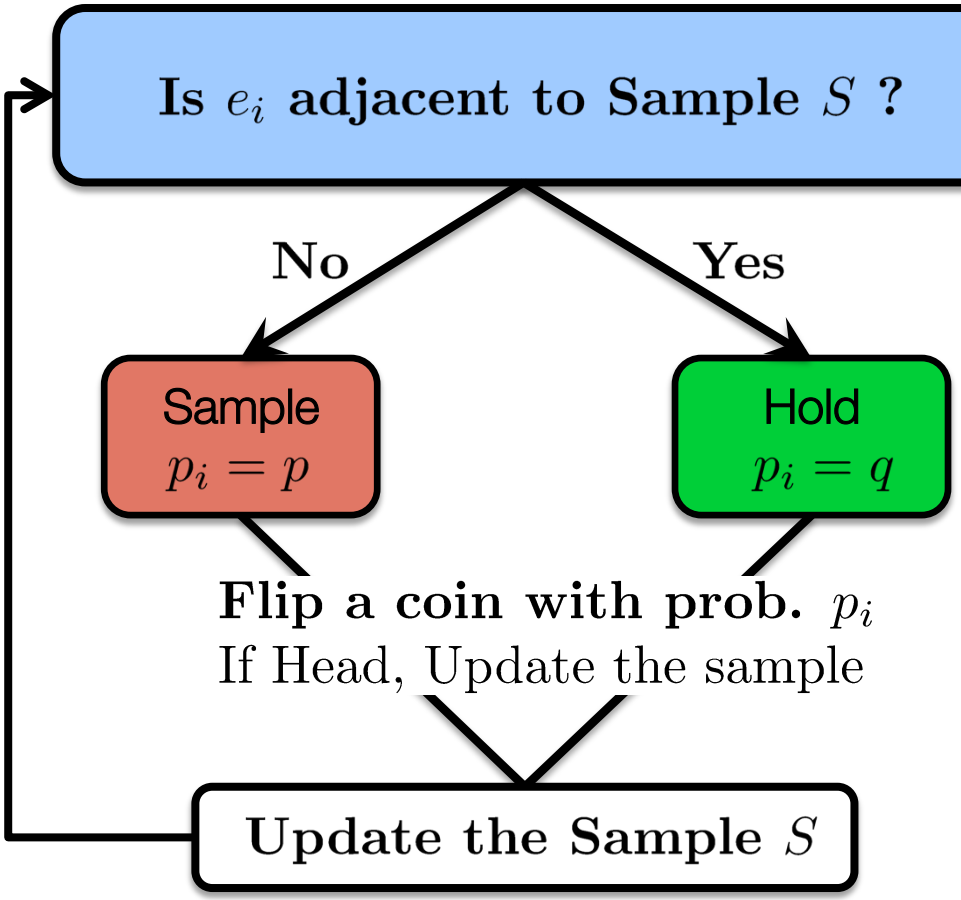
If e_i NOT adjacent to S
Then $p_i = p$

If e_i adjacent e_j, such that $e_j \in S$
Then $p_i = q$

Flip a coin with prob. p_i
If Head, Update the sample

① The Sampling – gSH(p,q)

Start with an empty Sample S



Input

Edge stream

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

$\mathcal{P}[e_i$ **is selected** $|$ **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 $p_i = q$

Add (e_i, p_i) to S

① The Sampling – gSH$_T$(p,q)

Start with an empty Sample S

Is e_i adjacent to Sample S ?

Input

Edge stream

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

No

Yes

Multiple dependencies

Sample

$p_i = p$

Hold

$p_i = q$

$p_i = 1$ if e_i closing triangle in S

$p_i = q$ otherwise

Flip a coin with prob. p_i

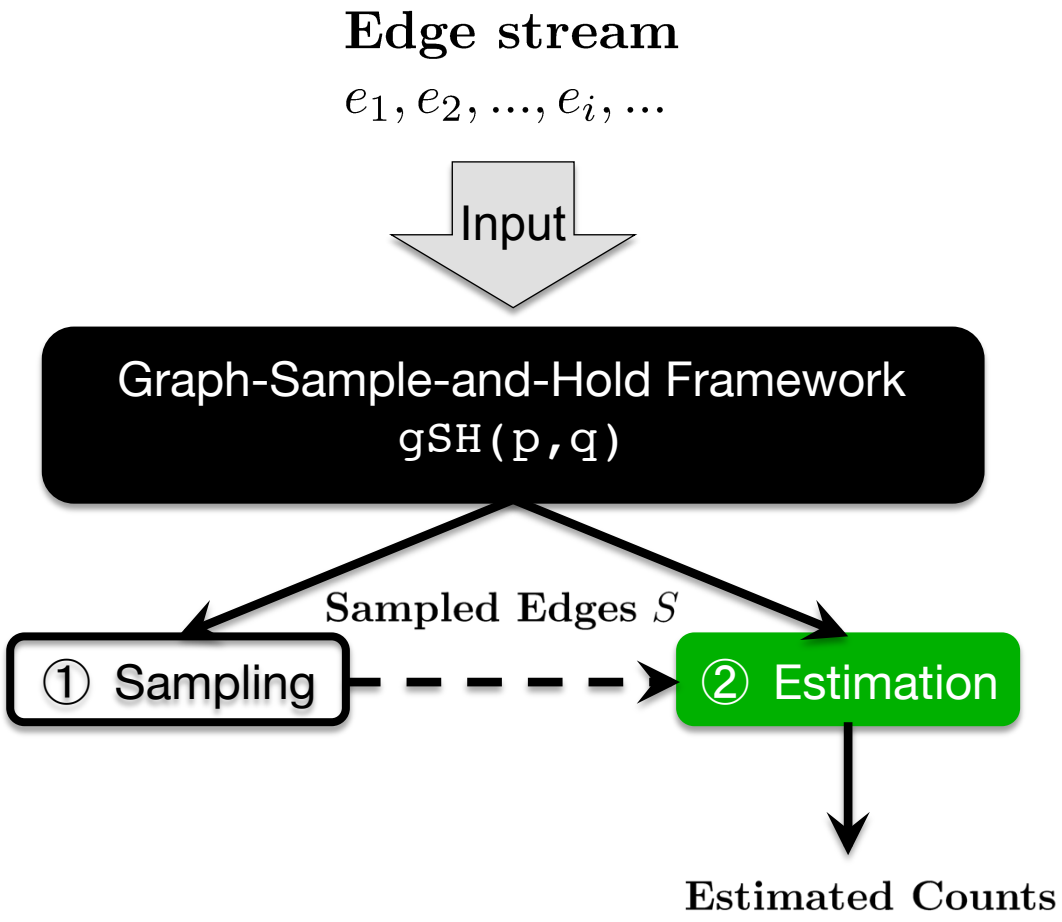
If Head, Update the sample

Update the Sample S

Add (e_i, p_i) to S

Graph-Sample-and-Hold : gSH(p,q)

Given: Graph $G = (V, E)$
$E = \{e_1, e_2, ..., e_i, ...\}$



② The Estimation

We use Horvitz-Thompson statistical estimation framework

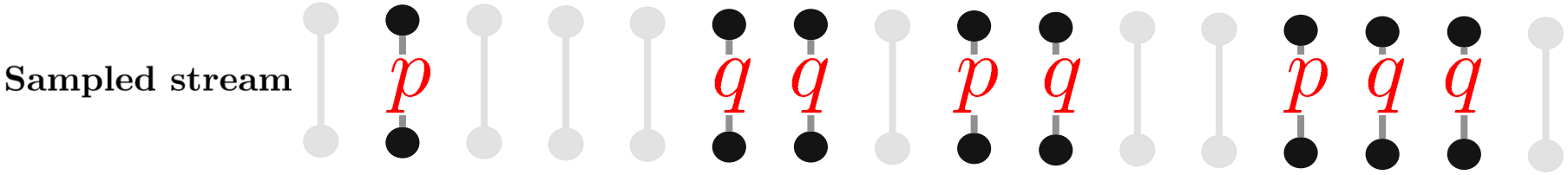
- Used for unequal probability sampling

[Horvitz and Thompson-1952]
Journal of American Stat. Assoc.

② The Estimation

- We define the selection estimator for an edge e_i

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



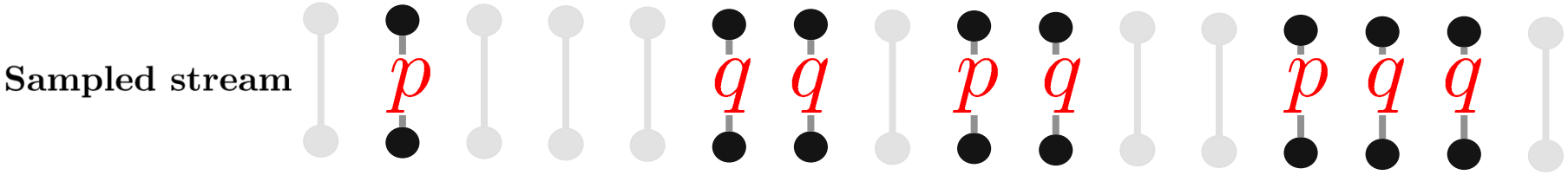
p_i is the sampling probability of edge e_i

② The Estimation

- We define the selection estimator for an edge e_i

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

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



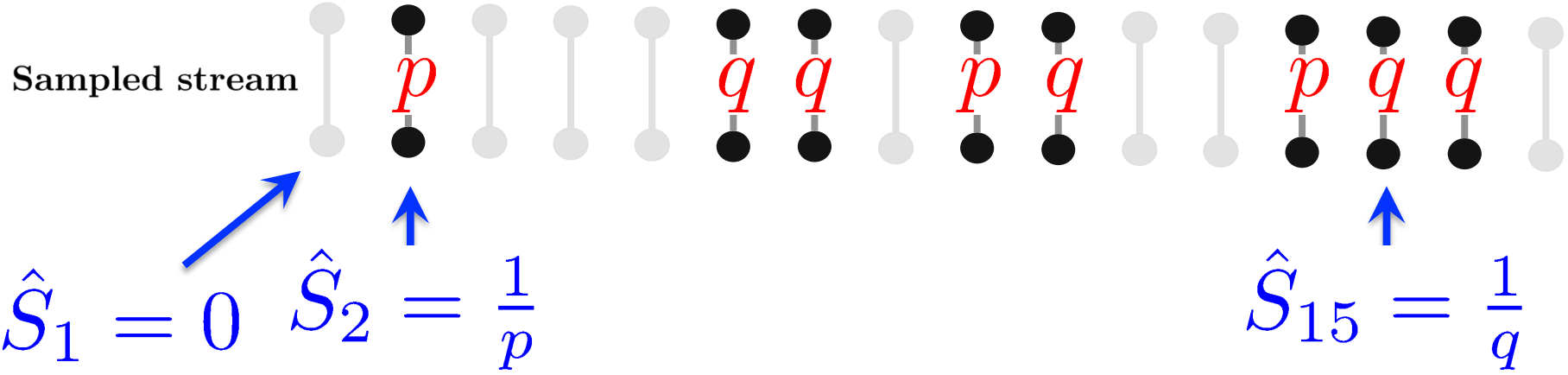
p_i is the sampling probability of edge e_i

② The Estimation

- We define the selection estimator for an edge e_i

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

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

② The Estimation

- We define the selection estimator for an edge e_i

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

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

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

Unbiasedness [Theorem 1 (i)]

② The Estimation

- We define the selection estimator for an edge e_i

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

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

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

Unbiasedness [Theorem 1 (i)]

- We derive the estimate of edge count

$$\hat{N_K} = \sum_{e_i \in S} \frac{1}{p_i}$$

② 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

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

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

Unbiasedness
[Theorem 1 (ii)]

② The Estimation

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

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

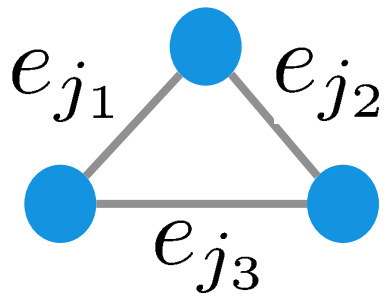
$$\hat{S}(J) = \prod_{j_i \in J} \hat{S}_{j_i} \longrightarrow \mathrm{E}[\hat{S}(J)] = 1$$

Unbiasedness [Theorem 1 (ii)]

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

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

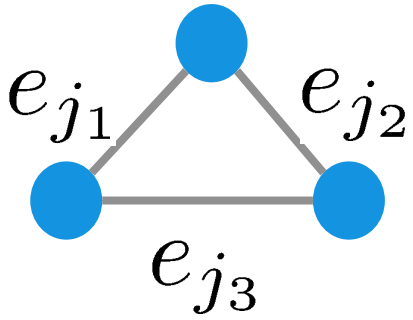
$$\hat{S}(\tau_j) = \prod_{j_i \in \tau_j} \frac{1}{p_{j_i}}$$



② The Estimation

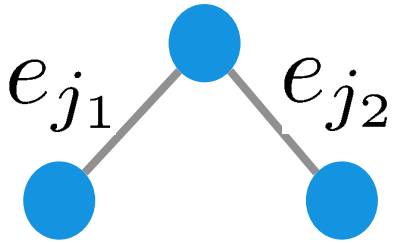
- The estimate of triangle counts is

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



- The estimate of wedge counts is

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

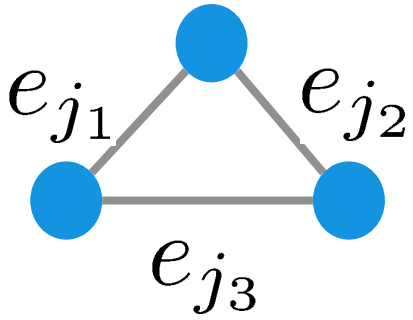


- Using the gSH framework, we obtain estimators for:
  - Edge Count 
  - Triangle Count 
  - Wedge Count 
  - Global clustering Coeff. = 3 *  / 

② The Estimation

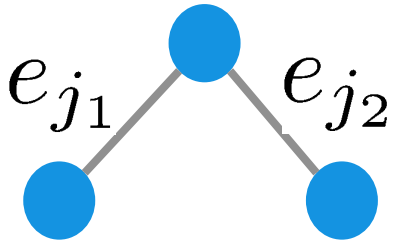
- The estimate of triangle counts is

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

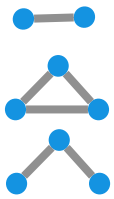


- The estimate of wedge counts is

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



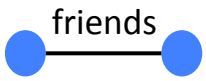
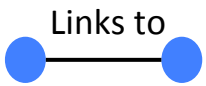
- Using the gSH framework, we obtain estimators for:
  - Edge Count
  - Triangle Count
  - Wedge Count
  - Global clustering Coeff. = 3 * /



We also compute HT unbiased variance estimators

Experiments

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

Graphs	Edge Count	Triangle Count	Wedge Count	Global Clustering Coeff.
Social facebook Graphs 	CMU, UCLA, and Wisconsin			
Web Graphs 	Google, Berkeley-Stanford, Stanford			

$$\textbf{Relative Error} = \frac{|\textbf{estimated} - \textbf{actual}|}{\textbf{actual}}$$

Edge Count

	N_K	$\widehat{N}_K$	$\frac{\lvert\widehat{N}_K - N_K\rvert}{N_K}$
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

	N_T	$\widehat{N}_T$	$\frac{\lvert\widehat{N}_T - N_T\rvert}{N_T}$
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

	N_Λ	$\widehat{N}_\Lambda$	$\frac{\lvert\widehat{N}_\Lambda - N_\Lambda\rvert}{N_\Lambda}$
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

	α	$\widehat{\alpha}$	$\frac{\lvert\widehat{\alpha} - \alpha\rvert}{\alpha}$
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

Edge Count

	N_K	$\widehat{N}_K$	$\frac{\lvert\widehat{N}_K - N_K\rvert}{N_K}$
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

Triang Count

	N_T	$\widehat{N}_T$	$\frac{\lvert\widehat{N}_T - N_T\rvert}{N_T}$
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

	N_Λ	$\widehat{N}_\Lambda$	$\frac{\lvert\widehat{N}_\Lambda - N_\Lambda\rvert}{N_\Lambda}$
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

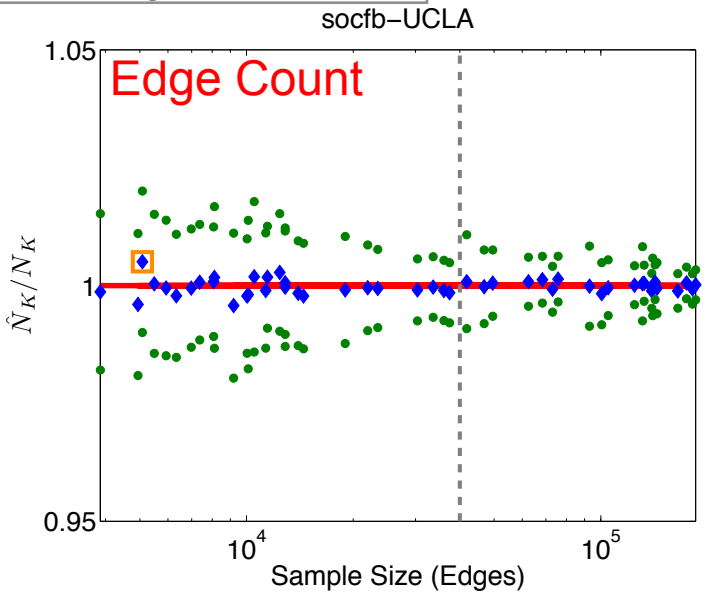
	α	$\widehat{\alpha}$	$\frac{\lvert\widehat{\alpha} - \alpha\rvert}{\alpha}$
socfb-CMU	0.18526	0.18574	0.00260
socfb-UCLA	0.14314	0.14363	0.00340
socfb-Wisconsin	0.12013	0.12101	0.00730
web-Stanford	0.00862	0.00862	0.00020
web-Google	0.05523	0.05565	0.00760
web-BerkStan	0.00694	0.00698	0.00680

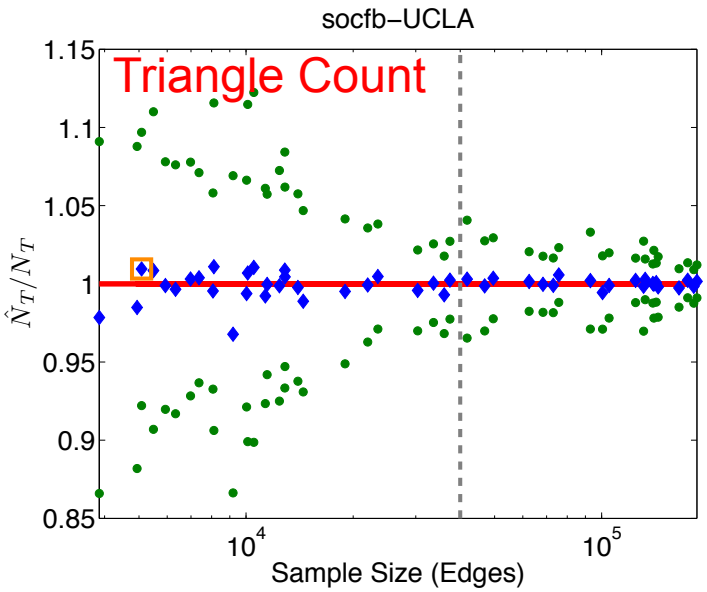
Sample size <= 40K edges

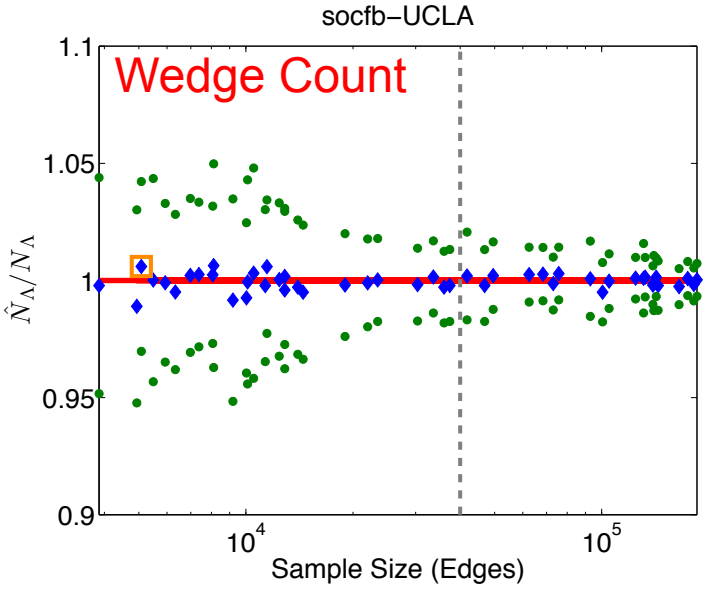
Actual
Estimated/Actual
Confidence Upper & Lower Bounds
Sample Size = 40K edges

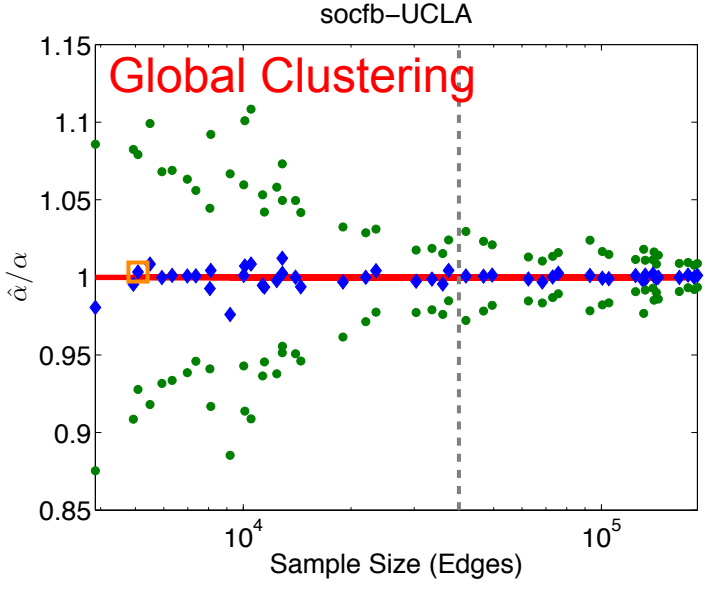
Dataset:
facebook friendship graph at UCLA

$$\frac{\text{Estimated}}{\text{Actual}}$$



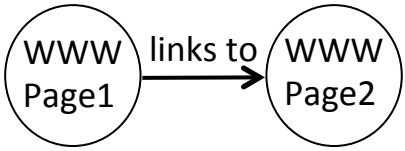






Comparison to previous work

Datasets:
Web graphs



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

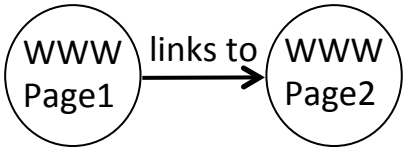
Our approach

graph	Jha *et al.* $\frac{\lvert\widehat{N}_T - N_T\rvert}{N_T}$	Jha *et al.* $SSize$	gSH $\frac{\lvert\widehat{N}_T - N_T\rvert}{N_T}$	gSH $SSize$	Sample size
web-Stanford	≈ 0.07	40K	0.0023	14.8K	5% of graph edges
web-Google	≈ 0.04	40K	0.0029	25.2K	0.6% of graph edges
web-BerkStan	≈ 0.12	40K	0.0063	39.8K	0.57% of graph edges

$$\textbf{Relative Error} = \frac{|\textbf{estimated} - \textbf{actual}|}{\textbf{actual}}$$

92% - 96% Improvement

Datasets:
Web graphs



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

Our approach

graph	Jha *et al.* $\frac{\lvert\widehat{N}_T - N_T\rvert}{N_T}$	Jha *et al.* $SSize$	gSH $\frac{\lvert\widehat{N}_T - N_T\rvert}{N_T}$	gSH $SSize$	Sample size
web-Stanford	≈ 0.07	40K	0.0023	14.8K	5% of graph edges
web-Google	≈ 0.04	40K	0.0029	25.2K	0.6% of graph edges
web-BerkStan	≈ 0.12	40K	0.0063	39.8K	0.57% of graph edges

$$\textbf{Relative Error} = \frac{|\textbf{estimated} - \textbf{actual}|}{\textbf{actual}}$$

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