# Distributed Estimation of Graph 4-Profiles

**Ethan R. Elenberg**, Karthikeyan Shanmugam, Michael Borokhovich, Alexandros G. Dimakis

University of Texas, Austin, USA

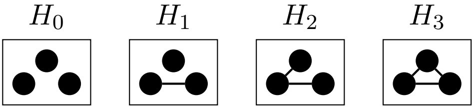
April 14, 2016

# Introduction

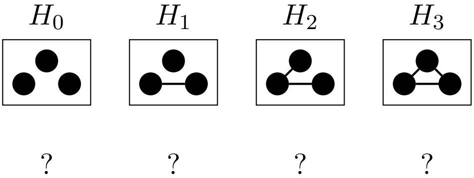
- Perform analytics on large graphs
  - How does information spread across the Web?
  - Is this social network user a spam bot?
  - Classify a protein as helpful or harmful
- Generalize existing subgraph analysis
  - Triangle counts, clustering coefficient, graphlet frequencies
- Scalable, distributed algorithms

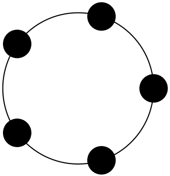
# Graph 3-profile

- Count the induced subgraphs formed by selecting all triples of vertices



- Count the induced subgraphs formed by selecting all triples of vertices

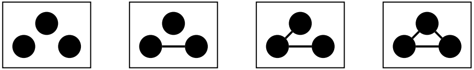


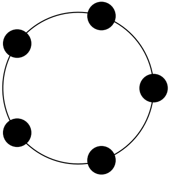
# Graph 3-profile

- Count the induced subgraphs formed by selecting all triples of vertices

| $H_0$ | $H_1$ | $H_2$ | $H_3$ |
|---|---|---|---|



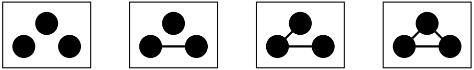
| 0 | ? | ? | ? |
|---|---|---|---|



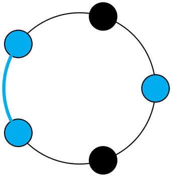
# Graph 3-profile

- Count the induced subgraphs formed by selecting all triples of vertices

| $H_0$ | $H_1$ | $H_2$ | $H_3$ |
|---|---|---|---|

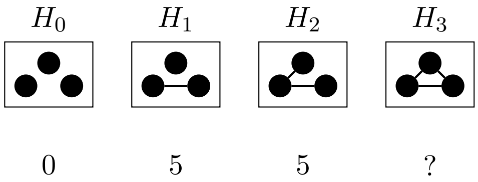


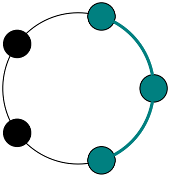
| 0 | 5 | ? | ? |
|---|---|---|---|



# Graph 3-profile

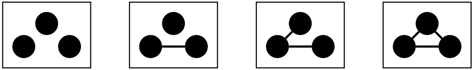
- Count the induced subgraphs formed by selecting all triples of vertices



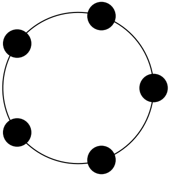



- Count the induced subgraphs formed by selecting all triples of vertices

| $H_0$ | $H_1$ | $H_2$ | $H_3$ |
| --- | --- | --- | --- |

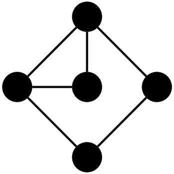


| 0 | 5 | 5 | 0 |
| --- | --- | --- | --- |

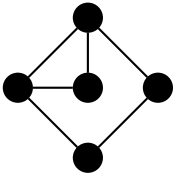


# Graph 4-profile

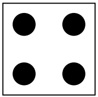
- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?]$



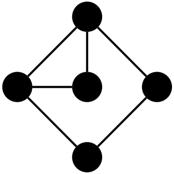
- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?]$

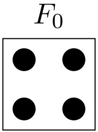


$F_0$

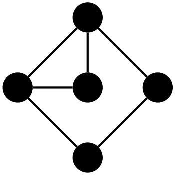


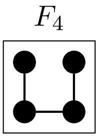
- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?]$





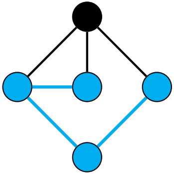
- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, 0, 0, 0, ?, ?, ?, ?, ?, ?, ?]$

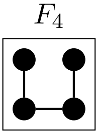




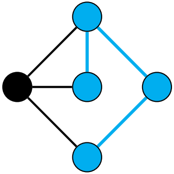
# Graph 4-profile

- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, 0, 0, 0, 1, ?, ?, ?, ?, ?, ?]$

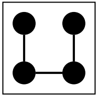




- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, 0, 0, 0, 2, ?, ?, ?, ?, ?, ?]$

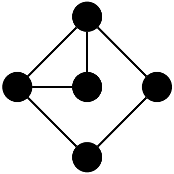


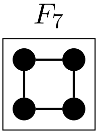
$F_4$



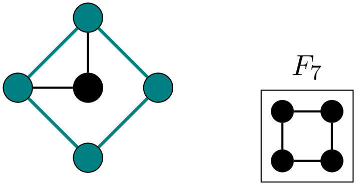
# Graph 4-profile

- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, 0, 0, 0, 2, 0, 0, ?, ?, ?, ?]$

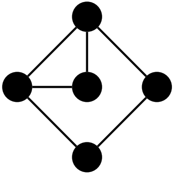




- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, 0, 0, 0, 2, 0, 0, 1, ?, ?, ?]$



- Similarly, count the induced subgraphs formed by selecting all sets of $4$ vertices
- $\mathbf{n}(G) = [0, 0, 0, 0, 2, 0, 0, 1, 2, 0, 0]$



# Graph 4-profile

**Definition**

Let $N_i$ be the number of $F_i$'s in a graph $G$. The vector $\mathbf{n}(G) = [N_0, N_1, \ldots, N_{10}]$ is called the global 4-profile of $G$.

- Always sums to $\binom{|V|}{4}$, the total number of 4-subgraphs

# Graph 4-profile

**Definition**

Let $N_i$ be the number of $F_i$'s in a graph $G$. The vector $\mathbf{n}(G) = [N_0, N_1, \ldots, N_{10}]$ is called the global 4-profile of $G$.

- Always sums to $\binom{|V|}{4}$, the total number of 4-subgraphs

**Definition**

For each $v \in V$, the local 4-profile counts how many times $v$ participates in each $F_i$ with 3 other vertices.

# Motivation

- Local 4-profiles embed each vertex into an $11$ dimensional feature space
  - Spam detection
  - Generative models
- Global 4-profile concisely describes local connectivity
  - Molecule classification

# Introduction

- Problem: Compute (or approximate) 4-profile quantities for a large graph

# Introduction

- Problem: Compute (or approximate) 4-profile quantities for a large graph
- Previous approaches have 2 drawbacks
  - Require global communication
  - Many vertices redundantly repeat the same calculations

# Contributions

1. Design novel, distributed algorithm to calculate local 4-profiles
2. Derive improved concentration bounds for 4-profile sparsifiers
3. Evaluate performance on real-world datasets

# Related Work

Well studied across several communities (graphlets, motifs, subgraph frequencies):

- Graph sub-sampling
  [Kim, Vu '00] [Tsourakakis, et al. '08 -'11] [Jha, et al. '15]
- Large-scale triangle counting
  [Shank '07] [Satish, et al. '14] [Eden, et al. '15]
- Subgraph/graphlet counting equations
  [Kloks, et al. '00] [Kowaluk, et al. '13]
  ORCA [Hočevar, Demšar '14] [E. '15] [Ahmed, et al. '15]

# Outline

# 4-Prof-Dist

- Message passing algorithm in the Gather-Apply-Scatter framework
  - GraphLab, Pregel, Spark GraphX, etc.
  - Communication only allowed between adjacent vertices
  - Intermediate results stored as edge data

# 4-Prof-Dist

1. Each vertex computes its local 3-profile and triangle list

# 4-Prof-Dist

1. Each vertex computes its local 3-profile and triangle list

2. Each vertex solves a $17 \times 17$ system of equations relating its local 4-profile to its neighbors' local 3-profiles

   - Edge Pivots [Kloks, et al. '00] [Kowaluk, et al. '13] [E. '15]
   - $4$-clique Counting
   - $2$-hop Histogram

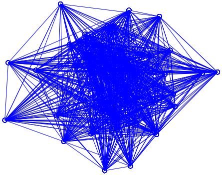
# Outline

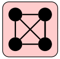
# Edge Sub-sampling Process

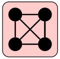
- Sub-sample each edge in the graph independently with probability $p$
- Relate the original and sub-sampled graphs via a 1-step Markov chain

# Edge Sub-sampling Process: 4-clique

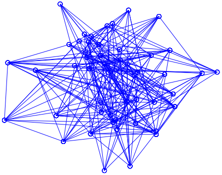
Original





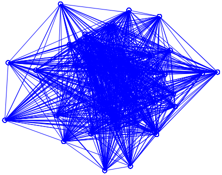


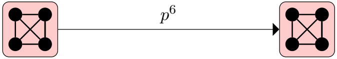
Sub-sampled



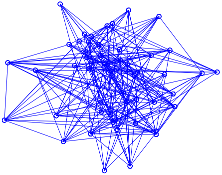
# Edge Sub-sampling Process: 4-clique

Original

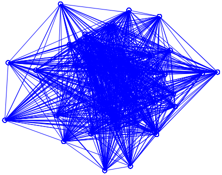


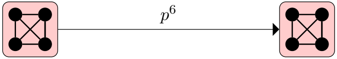



Sub-sampled

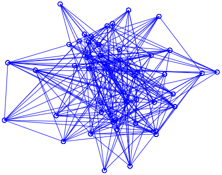


Original





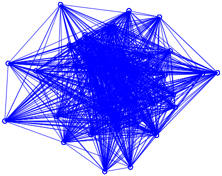

Sub-sampled

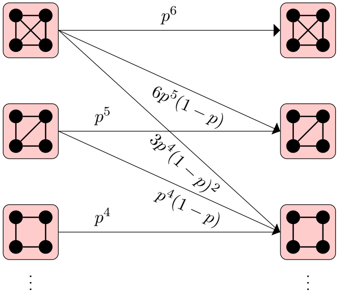


$$\left[\text{Estimator}\right] = \frac{1}{p^6}\left[\text{Sub-sampled}\right]$$

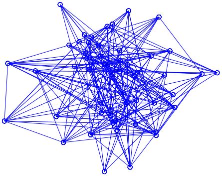
# Edge Sub-sampling Process: 4-clique

Original






Sub-sampled



$$\left[\text{Estimator}\right] = \frac{1}{p^6}\left[\text{Sub-sampled}\right]$$

# Edge Sub-sampling Process

In general, construct an invertible transition matrix $\mathbf{H}$

$$H_{ij} = \mathbb{P}(\text{sub-sampled} = F_i \mid \text{original} = F_j)$$

$$\begin{bmatrix} \text{Unbiased} \\ \text{Estimators} \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} \text{Sub-sampled} \\ \text{4-profile} \end{bmatrix}$$

- $N_{10}$ - # 4-cliques
- $k_{10}$ - Maximum # 4-cliques sharing a common edge

**Theorem** (*4-clique sparsifier*)

*If the sampling probability*

$$p \geq \left(\frac{\log(2/\delta)k_{10}}{2\epsilon^2 N_{10}}\right)^{1/12},$$

*then the relative error is bounded by $\epsilon$ with probability at least $1-\delta$.*

# Main Concentration Result

- $N_{10}$ - # 4-cliques
- $k_{10}$ - Maximum # 4-cliques sharing a common edge

**Theorem** (*4-clique sparsifier*)

*If the sampling probability*

$$p \geq \left( \frac{\log(2/\delta) k_{10}}{2\epsilon^2 N_{10}} \right)^{1/12},$$

*then the relative error is bounded by $\epsilon$ with probability at least $1 - \delta$.*

*Proof Sketch:*

- 4-clique estimator is associated with a read-$k_{10}$ function family

$$f(G, p) = e_1 e_2 e_4 e_5 e_7 e_8 + e_4 e_5 e_6 e_8 e_9 e_{10} + \ldots$$

# Outline

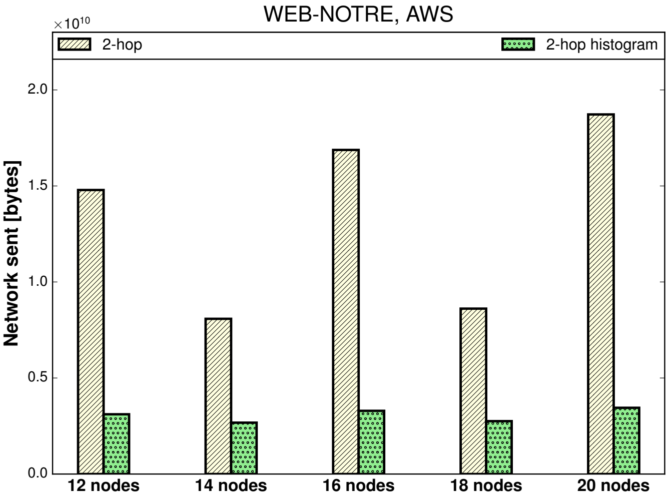
# Implementation

- GraphLab PowerGraph v2.2
- Multicore server
  - 256 GB RAM, 72 logical cores
- EC2 cluster (Amazon Web Services)
  - 20 c3.8xlarge, 60 GB RAM, 32 logical cores each

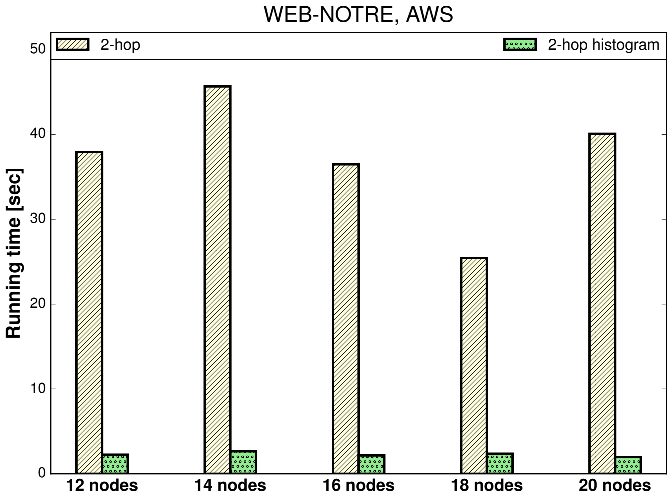
- GraphLab PowerGraph v2.2
- Multicore server
  - 256 GB RAM, 72 logical cores
- EC2 cluster (Amazon Web Services)
  - 20 c3.8xlarge, 60 GB RAM, 32 logical cores each

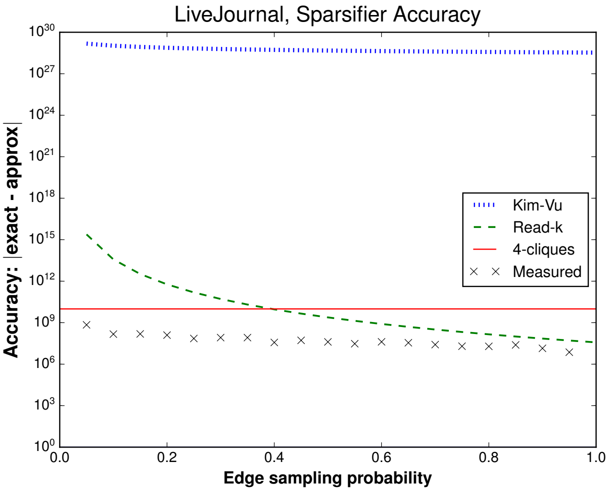
**Datasets**

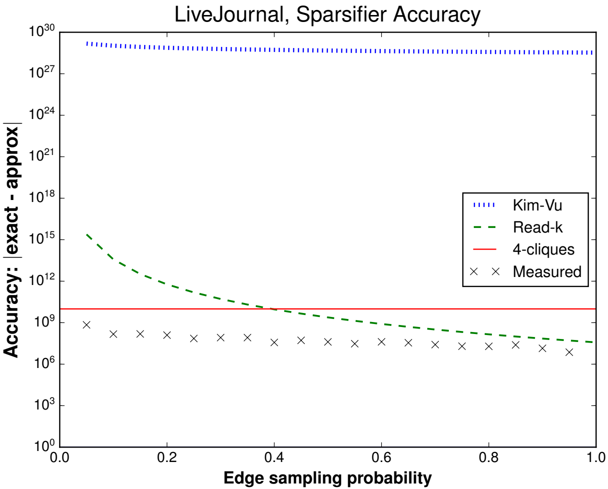
| Name | Vertices | Edges (undirected) |
|---|---:|---:|
| WEB-NOTRE | $325,729$ | $1,090,108$ |
| LiveJournal | $4,846,609$ | $42,851,237$ |


WEB-NOTRE, AWS
×10^10
2-hop
2-hop histogram
Network sent [bytes]
0.0
0.5
1.0
1.5
2.0
12 nodes
14 nodes
16 nodes
18 nodes
20 nodes

# Results: AWS Full Neighborhood vs. Histogram, 10 runs

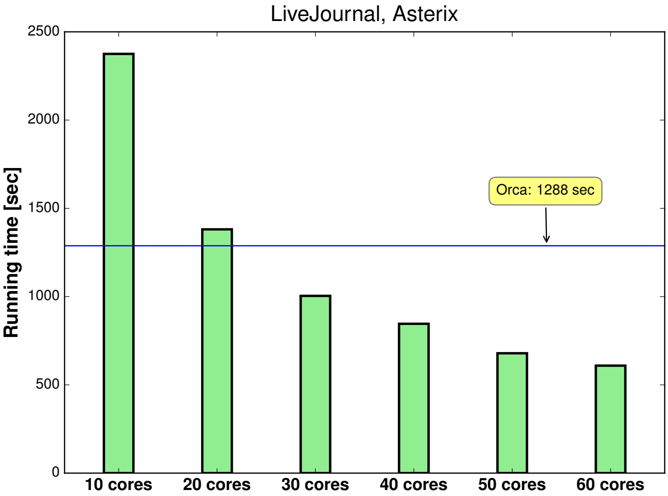



LiveJournal, Sparsifier Accuracy
Accuracy: |exact - approx|
Edge sampling probability
$10^{30}$
$10^{27}$
$10^{24}$
$10^{21}$
$10^{18}$
$10^{15}$
$10^{12}$
$10^{9}$
$10^{6}$
$10^{3}$
$10^{0}$
0.0
0.2
0.4
0.6
0.8
1.0
Kim-Vu
Read-k
4-cliques
Measured



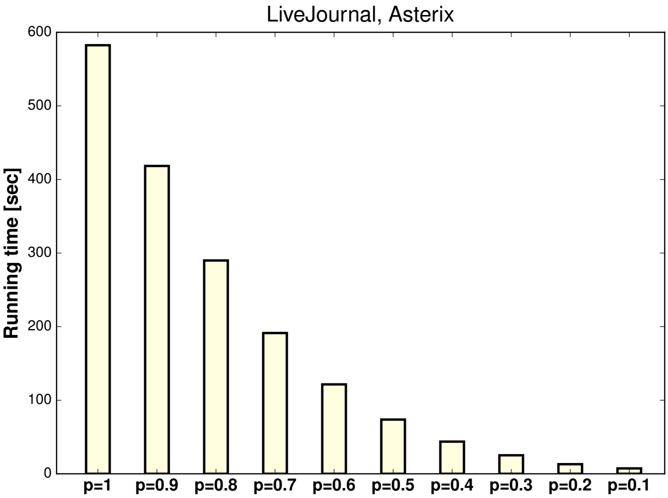

Improves on previous bounds if $p = \Omega(1/\log|E|)$

# Results: Multicore Running Time Comparison, 10 runs

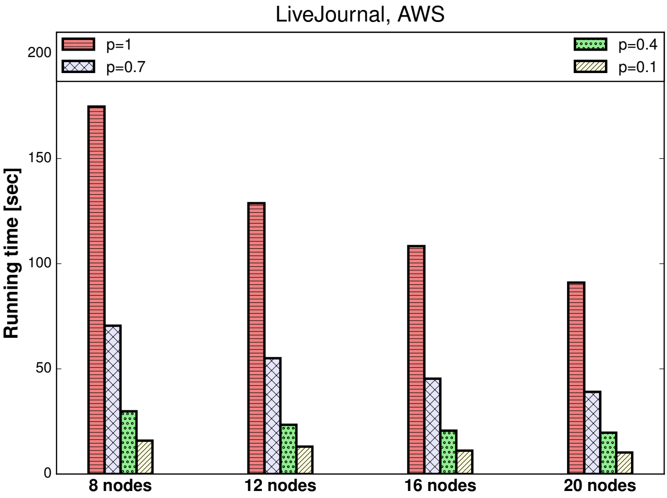



4-Prof-Dist, $p = 1$ vs. Orca [Hočevar, Demšar '14]

# Results: Multicore Running Time Comparison, 10 runs



# Results: AWS Running Time Comparison, 10 runs



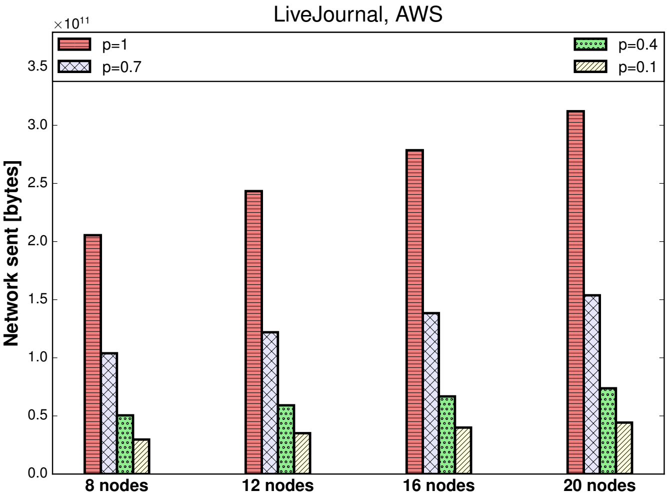
# Outline

# Summary

1. 2-hop histogram reduces network communication in a distributed setting
2. Edge sub-sampling produces fast, accurate 4-profile estimates
   - Bounds for other subgraphs in the full paper
3. Distributed/parallel implementation improves performance at scale

`github.com/eelenberg/4-profiles`

# (Backup) Results: AWS Network Communication, 10 runs



# (Backup) 2-hop Histogram

- At each vertex $a$, store a $(p, c_a[p])$ pair for each neighbor $p$
- For any $a \in \Gamma(v)$ and $p \notin \Gamma(v)$,

$$c_a[p] = 1 \quad \Leftrightarrow \quad vap \text{ forms a 2-path}$$

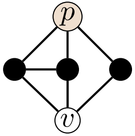
- Gather across neighbors to count the number of distinct $2$-paths from $v$ to $p$

# (Backup) 2-hop Histogram

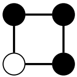
- At each vertex $a$, store a $(p, c_a[p])$ pair for each neighbor $p$
- For any $a \in \Gamma(v)$ and $p \notin \Gamma(v)$,

$$c_a[p] = 1 \quad \Leftrightarrow \quad vap \text{ forms a 2-path}$$

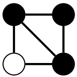
- Gather across neighbors to count the number of distinct $2$-paths from $v$ to $p$




$=$



$+$

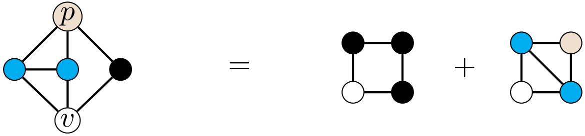


$\sum_{p \notin \Gamma(v)} \binom{\oplus_{a \in \Gamma(v)} c_a[p]}{2}$ $\qquad F_7(v) \qquad F_9(v)$

- At each vertex $a$, store a $(p, c_a[p])$ pair for each neighbor $p$
- For any $a \in \Gamma(v)$ and $p \notin \Gamma(v)$,

$$c_a[p] = 1 \quad \Leftrightarrow \quad vap \text{ forms a 2-path}$$

- Gather across neighbors to count the number of distinct $2$-paths from $v$ to $p$



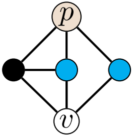

$\sum_{p \notin \Gamma(v)} \binom{\oplus_{a \in \Gamma(v)} c_a[p]}{2}$ $\qquad F_7(v) \qquad F_9(v)$

# (Backup) 2-hop Histogram

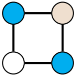
- At each vertex $a$, store a $(p, c_a[p])$ pair for each neighbor $p$
- For any $a \in \Gamma(v)$ and $p \notin \Gamma(v)$,

$$c_a[p] = 1 \quad \Leftrightarrow \quad vap \text{ forms a 2-path}$$

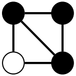
- Gather across neighbors to count the number of distinct $2$-paths from $v$ to $p$




$=$



$+$



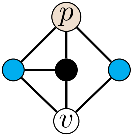
$$\sum_{p \notin \Gamma(v)} \binom{\bigoplus_{a \in \Gamma(v)} c_a[p]}{2} \qquad F_7(v) \qquad F_9(v)$$

# (Backup) 2-hop Histogram

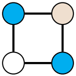
- At each vertex $a$, store a $(p, c_a[p])$ pair for each neighbor $p$
- For any $a \in \Gamma(v)$ and $p \notin \Gamma(v)$,

$$c_a[p] = 1 \quad \Leftrightarrow \quad vap \text{ forms a 2-path}$$

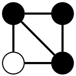
- Gather across neighbors to count the number of distinct $2$-paths from $v$ to $p$




$=$



$+$



$$\sum_{p \notin \Gamma(v)} \binom{\bigoplus_{a \in \Gamma(v)} c_a[p]}{2} \qquad F_7(v) \qquad F_9(v)$$

# (Backup) Local 3-profile

Vertex program in the Gather-Apply-Scatter framework [E. '15]

# (Backup) Local 3-profile

Vertex program in the Gather-Apply-Scatter framework [E. '15]

1. For each vertex $v$: Gather and Apply vertex IDs to store $\Gamma(v)$

# (Backup) Local 3-profile

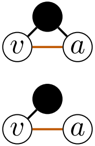
Vertex program in the Gather-Apply-Scatter framework [E. '15]

1. For each vertex $v$: Gather and Apply vertex IDs to store $\Gamma(v)$

2. For each edge $va$: Scatter

$$n_{3,va} = |\Gamma(v) \cap \Gamma(a)|,$$

$$n^c_{2,va} = |\Gamma(v)| - |\Gamma(v) \cap \Gamma(a)| - 1, \ \ldots$$



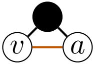
# (Backup) Local 3-profile

Vertex program in the Gather-Apply-Scatter framework [E. '15]

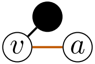
1. For each vertex $v$: Gather and Apply vertex IDs to store $\Gamma(v)$

2. For each edge $va$: Scatter

$$n_{3,va} = |\Gamma(v) \cap \Gamma(a)|,$$

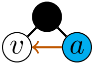



$$n^c_{2,va} = |\Gamma(v)| - |\Gamma(v) \cap \Gamma(a)| - 1, \ \ldots$$

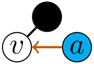


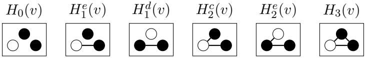

3. For each vertex $v$: Gather and Apply

$$n_{3,v} = \tfrac{1}{2} \sum_{a \in \Gamma(v)} n_{3,va}$$




$$n^c_{2,v} = \tfrac{1}{2} \sum_{a \in \Gamma(v)} n^c_{2,va}, \ \ldots$$






$$\sum_{a\in\Gamma(v)} \binom{n^e_{1,va}}{2} = F_1(v) + F_2(v)$$

$$\sum_{a\in\Gamma(v)} \binom{n^c_{2,va}}{2} = 3F_6^{'}(v) + F_8^{'}(v)$$

$$\sum_{a\in\Gamma(v)} \binom{n_{3,va}}{2} = F_9^{'}(v) + 3F_{10}(v)$$

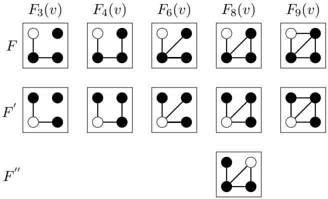
$$\sum_{a\in\Gamma(v)} n^e_{1,va} n^c_{2,va} = 2F_3^{'}(v) + F_4^{'}(v)$$

$$\sum_{a\in\Gamma(v)} n^e_{1,va} n_{3,va} = 2F_5(v) + F_8^{''}(v)$$

$$\sum_{a\in\Gamma(v)} n^c_{2,va} n^e_{2,va} = F_4^{'}(v) + 2F_7(v)$$

$$\sum_{a\in\Gamma(v)} n^c_{2,va} n_{3,va} = 2F_8^{'}(v) + 2F_9^{'}(v)$$

$$n^d_{1,v}|\Gamma(v)| = F_2(v) + F_4(v) + F_8(v)$$

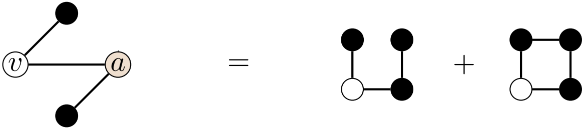


# (Backup) Edge Pivot Equations

- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors

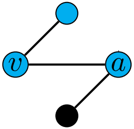
# (Backup) Edge Pivot Equations

- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors

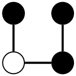



$\sum_{a \in \Gamma(v)} n^c_{2,va} n^e_{2,va}$ $\quad$ $F'_4(v)$ $\quad$ $2F_7(v)$

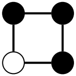
- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors




$=$

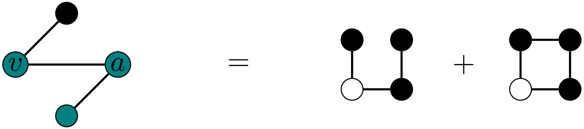


$+$



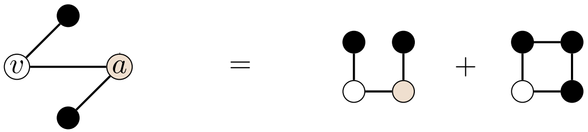
$\sum_{a\in\Gamma(v)} n^c_{2,va} n^e_{2,va}$ $\quad = \quad$ $F'_4(v)$ $\quad + \quad$ $2F_7(v)$

- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors




$$\sum_{a \in \Gamma(v)} n^{c}_{2,va} n^{e}_{2,va} \qquad = \qquad F'_4(v) \qquad + \qquad 2F_7(v)$$

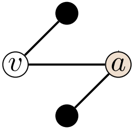
- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors



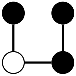

$\sum_{a\in\Gamma(v)} n^{c}_{2,va} n^{e}_{2,va}$ $F'_4(v)$ $2F_7(v)$

# (Backup) Edge Pivot Equations

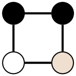
- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors




=



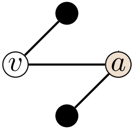
+



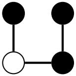
$\sum_{a\in\Gamma(v)} n^{c}_{2,va} n^{e}_{2,va}$ $\qquad$ $F'_4(v)$ $\qquad$ $2F_7(v)$

# (Backup) Edge Pivot Equations

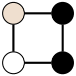
- Relate local 4-profile at $v$ to neighboring local 3-profiles
- Accumulate pairs of subgraph counts involving edge $va$, summed over all neighbors




$=$



$+$



$$\sum_{a \in \Gamma(v)} n^c_{2,va} n^e_{2,va} = F'_4(v) + 2F_7(v)$$

# (Backup) Clique Counting

- Directly count the number of 4-cliques, $F_{10}$

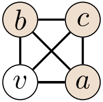
1. Initially, each vertex stores its triangle list $\Delta(v)$

- Directly count the number of 4-cliques, $F_{10}$

1. Initially, each vertex stores its triangle list $\Delta(v)$
2. For each edge $v$:
   Gather triangles of $a$ that contain $3$ neighbors of $v$

- Directly count the number of 4-cliques, $F_{10}$

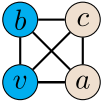
1. Initially, each vertex stores its triangle list $\Delta(v)$
2. For each edge $v$:
   Gather triangles of $a$ that contain $3$ neighbors of $v$




$$\sum_{a \in \Gamma(v)} |(b,c) \in \Delta(a) : b \in \Gamma(v), c \in \Gamma(v)|$$

- Directly count the number of 4-cliques, $F_{10}$

1. Initially, each vertex stores its triangle list $\Delta(v)$
2. For each edge $v$:
   Gather triangles of $a$ that contain $3$ neighbors of $v$

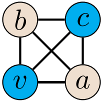



$$\sum_{a \in \Gamma(v)} |(b, c) \in \Delta(a) : b \in \Gamma(v), c \in \Gamma(v)|$$

# (Backup) Clique Counting

- Directly count the number of 4-cliques, $F_{10}$

1. Initially, each vertex stores its triangle list $\Delta(v)$
2. For each edge $v$:
   Gather triangles of $a$ that contain $3$ neighbors of $v$

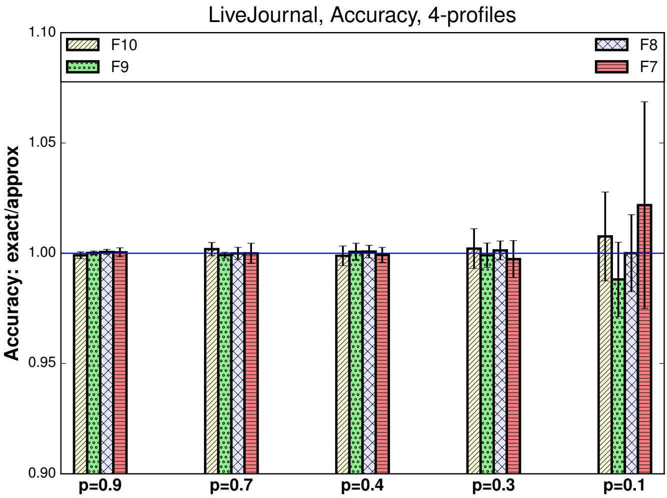



$$\sum_{a\in\Gamma(v)} |(b,c) \in \Delta(a) : b \in \Gamma(v), c \in \Gamma(v)|$$

- Because there are no large constants, our concentration bounds improve on earlier work in practical settings

**Corollary (*Comparison with [Kim,Vu]*)**

*Let $G$ be a graph with $m$ edges. If $p = \Omega(1/\log m)$ and $\delta = \Omega(1/m)$, then read-$k$ provides better triangle sparsifier accuracy than [Kim,Vu]. If additionally $k_{10} \leq N_{10}^{5/6}$, then read-$k$ provides better $4$-clique sparsifier accuracy than [Kim,Vu].*


LiveJournal, Accuracy, 4-profiles
Accuracy: exact/approx
1.10
1.05
1.00
0.95
0.90
F10
F9
F8
F7
p=0.9
p=0.7
p=0.4
p=0.3
p=0.1