Semi-Streaming Approximations of Centrality Indices in Massive Graphs

A thesis submitted to the faculty in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Benjamin W. Priest

Thayer School of Engineering
Dartmouth College

benjamin.w.priest.th@dartmouth

February 5, 2019

Overview

1. Introduction & Background
2. Summary of Results
3. Pseudo-Asynchronous Communication for Distributed Algorithms
4. DegreeSketch and Generalizations of Degree Centrality
5. Semi-Streaming Random Walk Simulation and κ-Path Centrality

Overview

1. Introduction & Background
    1. Problem Overview
    2. Graph Primitives
    3. Streaming Data Model Background
    4. Sketching Definitions
2. Summary of Results
3. Pseudo-Asynchronous Communication for Distributed Algorithms
4. DegreeSketch and Generalizations of Degree Centrality
5. Semi-Streaming Random Walk Simulation and κ-Path Centrality

Motivation

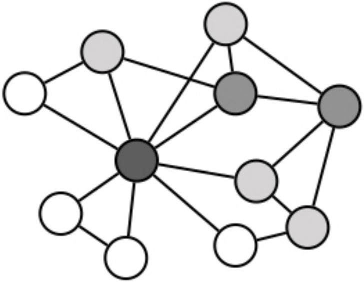
- Many modern computing problem focus on complex relational data
- Data are phrased as large graphs
  - e.g. the Internet, communication networks, transportation systems, protein networks, epidemiological models, social networks
- Often want to identify which vertices are "important"

Approach

Use data stream and distributed memory models

Centrality Indices

- Assign scores to vertices or edges
  - Higher score $\rightarrow$ more important
  - Depends on graph structure
  - Different indices in different domains
- Scores are not informative
  - Usually want top k elements
- Relative order-preserving approximation is acceptable



Exact algorithms do not scale to massive graphs

Massive-Scale Graph Centrality

The Problem

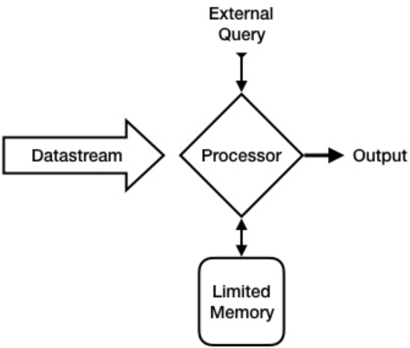
- High Memory, Computation, Comunication cost
- Wasted effort
  - Generally only need top elements vis-á-vis a centrality index

Our Solution

- Sketch data structures
  - Utilize composable streaming summaries of vertex-local information
- Distributed memory
  - Partition graph and distribute sketches
  - Polyloglinear computation, memory, and communication

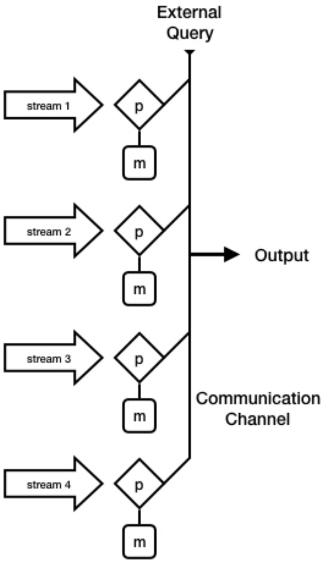
Overcoming Data Scale: Data Streaming

- Traditional RAM algorithms scale poorly
  - Awkward to store data in memory
  - Superlinear scaling unacceptable
- Data stream model to the rescue!
  - Sequential data access
  - Sublinear memory
  - Nearly linear amortized time
  - Constrained number of passes
  - Monte Carlo Approximations



Overcoming Data Scale: Distributed Data Streaming

- Distributed memory model a staple of HPC
  - Divide computation
  - Immense scaling
- Why not distributed data streams!?!
  - **Sketches** - composable summaries
  - vertex-centric algorithms
  - Even greater scaling
  - Linear communication



Streaming Background

Assume throughout that $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$, where $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$

- $\mathbf{w}_e$ is the weight of edge e if $e \in \mathcal{E}$ and zero otherwise
- $\mathcal{G}$ has adjacency matrix $A \in \mathbb{R}^{n \times n}$ so that $A_{x,y} = \mathbf{w}_{xy}$ for $xy \in \mathcal{E}$

$\mathcal{G}$ is given by a *stream* σ

- A list of edge insertions (*insert-only*)
- If deletions exist, say *turnstile stream*

An algorithm accumulating a data structure $\mathcal{S}$ and is called...

- *streaming* if $\mathcal{S}$ uses $\widetilde{O}(1) = O(\log n)$ [1] memory
- *semi-streaming* if $\mathcal{S}$ uses $\widetilde{O}(n) = O(n \operatorname{polylog} n)$ [2] memory

Want to minimize the number of passes over σ

- 1 pass ideal
- Constant or logarithmic passes sometimes acceptable

[1] $\widetilde{O}()$ notation suppresses polylogarithmic factors

[2] sometimes $\widetilde{O}\left(n^{1+\alpha}\right)$ for $\alpha \in (0, 1/2]$

Sketching

Definition (Sketch)

A *Sketch* is a streaming data structure $\mathcal{S}$ that admits a merge operator $\oplus$. If $\circ$ is the stream concatenation operator, then for any streams σ_1 and σ_2,

$$\mathcal{S}(\sigma_1) \oplus \mathcal{S}(\sigma_2) = \mathcal{S}(\sigma_1 \circ \sigma_2).$$

Definition (Linear Sketch)

A *Linear Sketch* $\mathcal{S}$ is a linear projection of $\mathbf{f}$ to a lower dimension. For any streaming frequency vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ and scalars a and b,

$$a\mathcal{S}(\mathbf{f}_1) + b\mathcal{S}(\mathbf{f}_2) = \mathcal{S}(a\mathbf{f}_1 + b\mathbf{f}_2).$$

Sketches are useful for stream summarization when *comparisons between streams* are important

Proposal Refresh

1. Semi-streaming approximation of betweenness centrality heavy hitters
   - Semi-streaming simulation of k random walks of length t
     - Lower and almost-tight upper bound algorithm
   - Semi-streaming κ-path centrality algorithm
2. Semi-streaming eigencentrality approximation
   - Explored at length by Eisha R. Nathan in [Nat18]
     - Mostly applied to Katz's Index
3. Sketch robustness analysis
   - Analysis of HyperLogLog cardinality sketches and their intersections
4. Implementation and empirical evaluation of semi-streaming graph algorithms
   - Implementation of YGM communication protocol
   - Implementation of DegreeSketch and related algorithms
     - Local neighborhood size and triangle count

Overview

1. Introduction & Background
2. Summary of Results
    1. Streaming Degree Centrality
    2. $O(1)$-Pass Semi-Streaming Closeness Centrality
    3. t-Pass Semi-Streaming Local tth Neighborhood Centrality
    4. 2-Pass Semi-Streaming Triangle Count Heavy Hitters
    5. Distributed Semi-Streaming Simulation of Random Walks
    6. Distributed Sublinear κ-Path Centrality
3. Pseudo-Asynchronous Communication for Distributed Algorithms
4. DegreeSketch and Generalizations of Degree Centrality
5. Semi-Streaming Random Walk Simulation and κ-Path Centrality

Summary of Results: Serial Algorithms

Degree Centrality

$$\mathcal{C}^{\mathrm{DEG}}(x) = |\{(u, v) \in E \mid x \in \{u, v\}\}| = \|A_{x,:}\|_1 = \|A_{:,x}\|_1$$

- Naïve online $O(n)$-space and -time algorithm exists

Summary of Results: Serial Algorithms

Degree Centrality

We show $\widetilde{O}(1)$-space distributable streaming algorithms

- Naïve online $O(n)$-space and -time algorithm exists

Summary of Results: Serial Algorithms

Degree Centrality

We show $\widetilde{O}(1)$-space distributable streaming algorithms

- Naïve online $O(n)$-space and -time algorithm exists

Closeness Centrality

$$\mathcal{C}^{\textsc{Close}}(x) = \frac{1}{\sum_{y \in V} d(x, y)}$$

- Online exact $O(n^2)$-space $O(nm)$-time algorithm [WC14]
- Batch Approximate $O(n^2)$-space and almost-linear time algorithm [CDPW14]

Summary of Results: Serial Algorithms

Degree Centrality

We show $\widetilde{O}(1)$-space distributable streaming algorithms

- Naïve online $O(n)$-space and -time algorithm exists

Closeness Centrality

$$\mathcal{C}^{\mathrm{CLOSE}}(x) = \frac{1}{\sum_{y} d(x,y)}$$

We show constant-pass semi-streaming algorithm

- Onl
- Batch Approximate $O(n^2)$-space and almost-linear time algorithm [CDPW14]

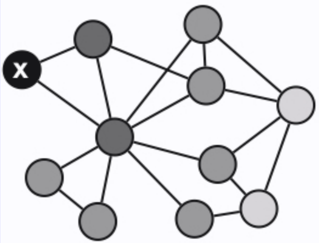
Summary of Results: Distributed Streaming Algorithms

Local tth Neighborhood Centrality

$$\mathcal{N}(t) = |\{(x,y) \in \mathcal{V} \times \mathcal{V} | d_{\mathcal{G}}(x,y) < t\}| \quad \text{(global) } t\text{th neighborhood}$$

$$\mathcal{C}_t^{\mathrm{NBHD}}(x) = |\{y \in \mathcal{V} | d_{\mathcal{G}}(x,y) < t\}| \quad \text{(local) } t\text{th neighborhood}$$

- Serial estimation algorithms ANF [PGF02] and HyperANF [BRV11]



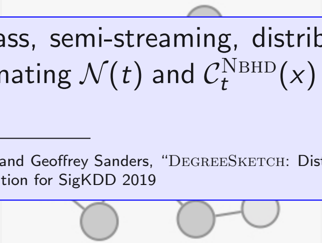
Summary of Results: Distributed Streaming Algorithms

Local tth Neighborhood Centrality

$$\mathcal{N}(t) = |\{(x, y) \in \mathcal{V} \times \mathcal{V} | d_{\mathcal{G}}(x, y) < t\}| \quad \text{(global) } t\text{th neighborhood}$$

$$\mathcal{C}_t^{\mathrm{NBHD}}(x) = |\{y \in \mathcal{V} | d_{\mathcal{G}}(x, y) < t\}| \quad \text{(local) } t\text{th neighborhood}$$

- Serial estimation algorithms ANF [PGF02] and HyperANF [BRV11]



We demonstrate a t-pass, semi-streaming, distributed DEGREESKETCH-based algorithm for estimating $\mathcal{N}(t)$ and $\mathcal{C}_t^{\mathrm{NBHD}}(x)$ for each $x \in \mathcal{V}$ [PPS19]. [a]

[a] Benjamin W. Priest, Roger Pearce and Geoffrey Sanders, "DEGREESKETCH: Distributed Cardinality Sketches on Graphs, with Applications," In preparation for SigKDD 2019

Summary of Results: Distributed Streaming Algorithms

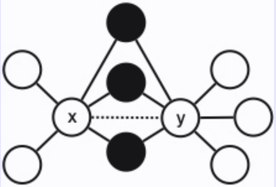
Triangle Count Centrality

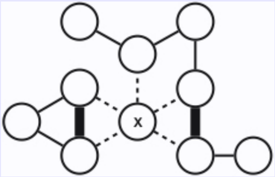
$$\mathcal{C}^{\mathrm{TRI}}(x) = |\{yz \in \mathcal{E} \mid xy, yz, xz \in \mathcal{E}\}| \qquad \text{(vertex-local)}$$

$$\mathcal{C}^{\mathrm{TRI}}(xy) = |\{z \in \mathcal{E} \mid xy, yz, xz \in \mathcal{E}\}| \qquad \text{(edge-local)}$$

- Exact $O(m)$-space, $O\left(m^{\frac{3}{2}}\right)$-time serial and distributed algorithms [AKM13, Pea17] [a]
- Semi-streaming sampling algorithms [LK15, SERU17] and distributed generalizations [SHL$^+$18, SLO$^+$18]

[a] Roger Pearce, "Triangle counting for scale-free graphs at scale in distributed memory," HPEC 2017





Summary of Results: Distributed Streaming Algorithms

Triangle Count Centrality

$$\mathcal{C}^{\mathrm{TRI}}(x) = |\{yz \in \mathcal{E} \mid xy, yz, xz \in \mathcal{E}\}| \qquad \text{(vertex-local)}$$

$$\mathcal{C}^{\mathrm{TRI}}(xy) = |\{z \in \mathcal{E} \mid xy, yz, xz \in \mathcal{E}\}| \qquad \text{(edge-local)}$$

- Exact $O(m)$-space, $O\left(m^{\frac{3}{2}}\right)$-time serial and distributed algorithms [AKM13, Pea17] [a]
- Semi-streaming sampling algorithms [LK15, SERU17] and distributed generalizations [SHL$^+$18, SLO$^+$18]

[a] Roger Pearce, "Triangle counting for scale-free graphs at scale in distributed memory," HPEC 2017

We show 2-pass, semi-streaming, distributed DegreeSketch-based algorithms for estimating heavy hitters [PPS18, PPS19] [a] [b]

[a] Benjamin W. Priest, Roger Pearce and Geoffrey Sanders, "Estimating Edge-Local Triangle Count Heavy Hitters in Edge-Linear Time and Almost-Vertex-Linear Space," HPEC 2018

[b] Benjamin W. Priest, Roger Pearce and Geoffrey Sanders, "DegreeSketch: Distributed Cardinality Sketches on Graphs, with Applications," In preparation for SigKDD 2019

Summary of Results: Distributed Streaming Algorithms

Semi-streaming simulation of k random walks of length t

- $O(nkt)$-space trivial algorithm
- $\Omega\left(n\sqrt{t}\right)$ lower bound and almost-tight upper bound for simulating a single random walk of length t [Jin18].

Summary of Results: Distributed Streaming Algorithms

Semi-streaming simulation of k random walks of length t

- $O(nkt)$-space trivial algorithm
- $\Omega\left(n\sqrt{t}\right)$ lower bound and almost-tight upper bound for simulating a single random walk of length t [Jin18].

- We show $\Omega\left(n\sqrt{kt}\right)$ space lower bound
- We demonstrate an $O\left(n\sqrt{kt}\frac{\log q}{q}\right)$ algorithm with failure probability ε, where $q = 2 + \frac{\log(1/\varepsilon)}{\sqrt{kt}}$ on insert-only streams.
- We demonstrate a distributed version of this algorithm, and describe how to generalize it to a faultless system utilizing the *playback* of adjacency substreams.

Summary of Results: Distributed Streaming Algorithms

κ-Path Centrality

$$\mathcal{C}_{\kappa}^{\mathrm{PATH}}(x) = \Pr_{p:|p|\leq\kappa}[x \in p \wedge p \text{ a simple path }]$$

- $O(m)$-space $O(n^{1+\alpha}\log^2 n)$-time approximation algorithm [KAS$^+$13]
- Empirical proxy for betweenness centrality heavy hitters
  - Online exact and approximate $O(n^2)$- and $O(m)$-space algorithms exist [GMB12, WC14, KMB15, BMS14]

Summary of Results: Distributed Streaming Algorithms

κ-Path Centrality

$$\mathcal{C}_{\kappa}^{\mathrm{PATH}}(x) = \Pr_{p:|p|\leq\kappa}[x \in p \wedge p \text{ a simple path }]$$

- $O(m)$-space $O(n^{1+\alpha}\log^2 n)$-time approximation algorithm [KAS+13]
- Empirical proxy for betweenness centrality heavy hitters
  - Online exact and approximate $O(n^2)$- and $O(m)$-space algorithms exist [GMB12, WC14, KMB15, BMS14]

- We show how to use the distributed random walk simulation framework with playback to estimate κ-path centrality at scale
- Yields sublinear distributed κ-path centrality approximation algorithm

Summary of Results

Table: Summary of Algorithmic Results

Problem	passes	distributed?	dynamic?	analytic bound?	lower bound?	experiments?
Degree Centrality	1		✓	✓		
Closeness Centrality	O(1)		✓	✓		
tth Neighborhood	t	✓		✓		
Local Triangle Counts	2	✓				✓
k t-step Random Walks	1[3]	✓	✓	✓	✓	
κ-Path Centrality	$*$	✓	✓	✓		

[3] Changes when subject to playback

Overview

1. Introduction & Background
2. Summary of Results
3. Pseudo-Asynchronous Communication for Distributed Algorithms
    1. Vertex-Centric Algorithm Challenges
    2. Pseudo-Asyncronous Communication Protocols
    3. Experiments & Implementation Details
4. DegreeSketch and Generalizations of Degree Centrality
5. Semi-Streaming Random Walk Simulation and κ-Path Centrality

Motivation: Vertex-Centric Algorithms

The Problem:

- Most distributed graph algorithms are vertex-centric
  - Partition local vertex information across processors
  - Processors communicate as in rounds [MAB$^+$10]
- Scale-free graphs common in applications
  - High degree vertices cause computation "hotspots"
  - Moves at the speed of the slowest processor

Solutions:

- Asynchronous Communication
  - Processors communicate point-to-point as needed
  - Increased implementation complexity
- Vertex delegation [PGA14] [4]
  - Cut hubs between processors

[4] Roger Pearce, Maya Gokhale and Nancy M. Amato, "Faster parallel traversal of scale free graphs at extreme scale with vertex delegates," SC 2014

Approach: Pseudo-Asynchronous Communication Protocol

The Idea

Aggregate and route messages "asynchronously", allowing processors to drop out of communication exchanges when finished

- Partion processor set $\mathcal{P}$ into *local* and *remote* exchanges
  - Takes advantage of hybrid distributed memory
- Mailbox abstraction
  - Aggregate messages at intermediaries
  - Route destination node traffic traffic through same remote channel
- Three protocols:
  - Node Local
  - Node Remote
  - Node Local Node Remote (NLNR)

Node Local and Node Remote

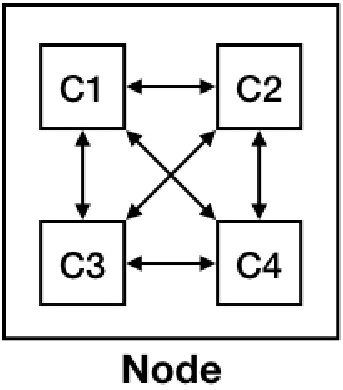
Node Local

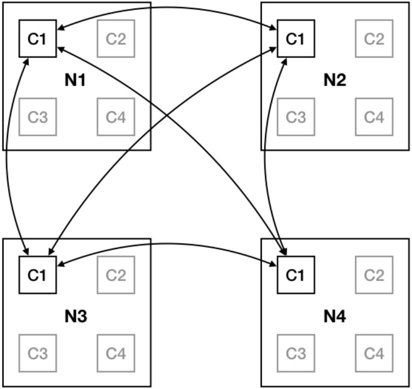
1. Exchange locally
2. Forward remotely

Node Remote ✓

1. Exchange remotely
2. Forward locally

Local Route	Remote Route





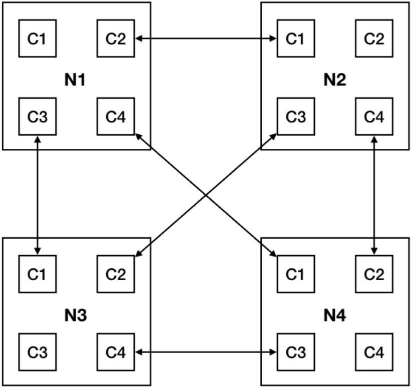
Node Local Node Remote

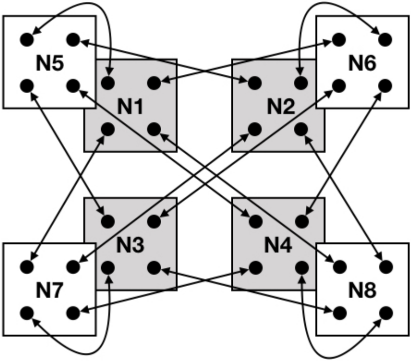
Node Local Node Remote

- Further partition $\mathcal{P}$ by *layers*
  - Set of ($\#$ cores) nodes

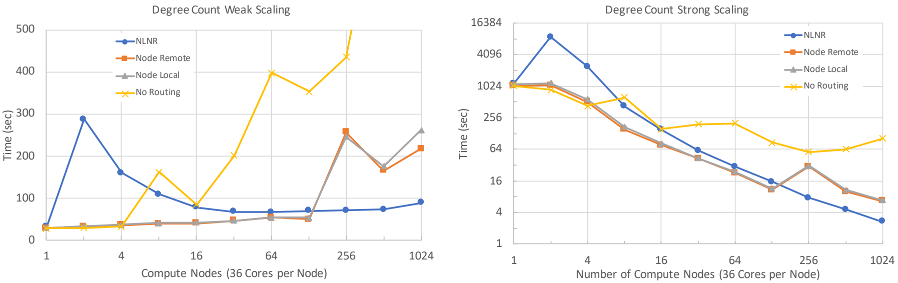
1. Exchange locally
2. Forward remotely
3. Forward locally

Intra-Layer Remote Route	Inter-Layer Remote Route





Validation of Claims



(a) Weak scaling. (b) Strong scaling

Figure: Weak and strong scaling wall time experiments for degree counting, as the number of nodes N varies from 1 up to 1024. Weak scaling experiments (a) assumed a universe of $N2^{28}$ vertices and sampled a total of $N2^{32}$ edges. Strong scaling experiments (b) assumed a universe of 2^{32} vertices and sampled a total of 2^{37} edges. Edge samplings uniformly sample two vertices without replacement from the universe of vertices. In all experiments mailbox sizes are fixed at 2^{18} messages.

Implementation Details

YGM **C++/MPI Library**

- Authored by myself, Trevor Steil (UMN), and Roger Pearce (LLNL)
- Simple API for handling pseudo-asynchronous communication
  - Clients need only specify receive behavior
- Supports LLNL Projects
  - HavoqGT (graph challenge & pattern matching)
  - Sierra 42 - largest scale to date graph 500 ($\sim$ 70T edges)
  - Possibly more in the future
    - e.g. Eccentricity
- Improvements over legacy HavoqGT routing
  - Flow control via pseudo-asynchronicity
  - NLNR more scalable
  - Variable length messages

YGM to be open sourced, submitted to IPDPS 2019 workshop

Overview

Motivation: Local tth Neighborhood Size

The Problem:

- tth neighborhood important for applications, e.g. effective diameter, load balancing, edge prediction, and probabilistic distance estimation
  - Exact computation expensive!
- Neighborhood Functions:

$$
\begin{aligned}
\mathcal{C}_t^{\mathrm{NBHD}}(x) &= |\{y \in \mathcal{V} | d_{\mathcal{G}}(x,y) < t\}| && \text{local } t\text{th neighborhood} \\
\mathcal{N}(t) &= |\{(x,y) \in \mathcal{V} \times \mathcal{V} | d_{\mathcal{G}}(x,y) < t\}| && \text{(global) } t\text{th neighborhood} \\
&= \sum_{x \in \mathcal{V}} \mathcal{C}_t^{\mathrm{NBHD}}(x)
\end{aligned}
$$

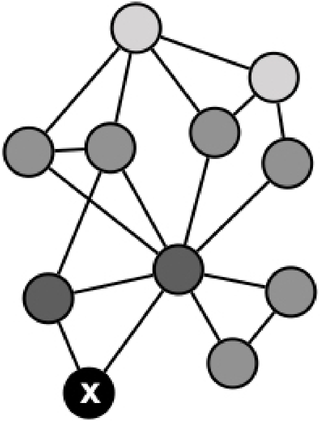
Existing Approximate Solutions:

- ANF algorithm [PGF02]
- HyperANF algorithm [BRV11]

Approach: DegreeSketch via Cardinality Sketches

Idea: Sketch and iteratively merge adjacency sets

- Cardinality sketches summarize set size
- Supports approximate union operation $\widetilde{\cup}$
- Assume some partitioning $f : \mathcal{V} \to \mathcal{P}$
- Distributed data structure $\mathcal{D}$
  - $\mathcal{D}[x]$ holds cardinality sketch for $x \in \mathcal{V}$
  - $|\mathcal{D}[x]|$ estimates $\mathbf{d}_x$
- Let $\mathcal{D}^1[x] = \mathcal{D}[x]$
- Set $\mathcal{D}^k[x] = \widetilde{\bigcup}_{y:xy\in\mathcal{E}} \mathcal{D}^{k-1}[y]$
- Then $\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x) = |\mathcal{D}^t[x]|$ and $\widetilde{\mathcal{N}}(t) = \sum_{x\in\mathcal{V}} \widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)$



HyperLogLog Cardinality Sketches

HLL cardinality sketches

Maintain $r = 2^p$ 6-bit registers M and a 64-bit hash function h

- Insert x: let $i = \langle x_1, \ldots, x_p \rangle$ and $w = \langle x_{p+1}, \ldots, x_{64} \rangle$
- $\rho(w) =$ initial zero bits of w plus 1
- $M_i = \max\{M_i, \rho(w)\}$
- Estimator derives from harmonic mean of M

HyperLogLog Cardinality Sketches

HLL cardinality sketches

Maintain $r = 2^p$ 6-bit registers M and a 64-bit hash function h

- Insert x:
- $\rho(w) =$ i
- $M_i = \max\{M_i, \rho(w)\}$
- Estimator derives from harmonic mean of M

Outputs $\widetilde{C}$ such that for cardinality C, w.h.p. $|C - \widetilde{C}| \leq \frac{1.04}{\sqrt{r}} C$ [FFGM07]

HyperLogLog Cardinality Sketches

HLL cardinality sketches

Maintain $r = 2^p$ 6-bit registers M and a 64-bit hash function h

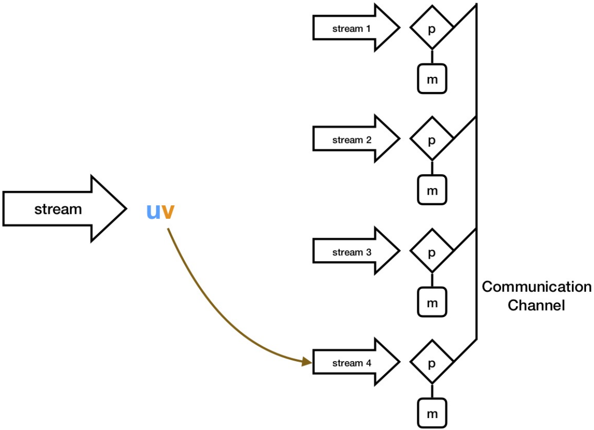
- Insert x:
- $\rho(w) = $ i
- $M_i = \max\{M_i, \rho(w)\}$
- Estimator derives from harmonic mean of M

Outputs $\widetilde{C}$ such that for cardinality C, w.h.p. $|C - \widetilde{C}| \leq \frac{1.04}{\sqrt{r}} C$ [FFGM07]

Useful improvements

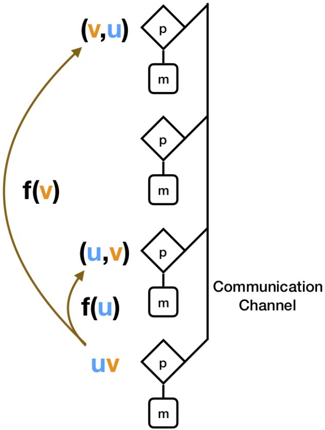
- Native union operator (elementwise maximum)
- Various improved harmonic [HNH13, QKT16] and maximum likelihood estimators [XZC17, Lan17, Ert17]
- Sparsification for low cardinality sets [HNH13]
- Compression to 4 and 3 bit registers [XZC17]
- Intersection estimators [Tin16, CKY17, Ert17]

DegreeSketch Accumulation



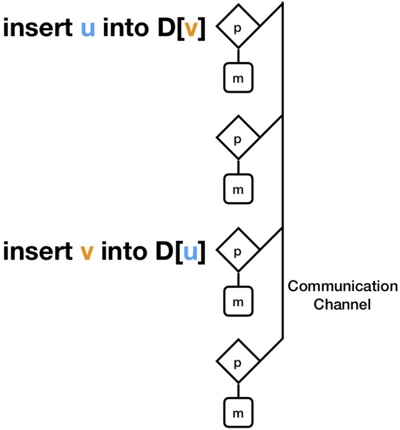
Partition stream across $\mathcal{P}$

DegreeSketch Accumulation



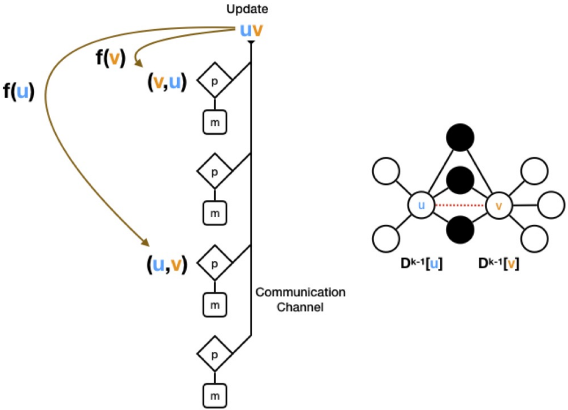
Distribute edges to endpoint owners

DegreeSketch Accumulation



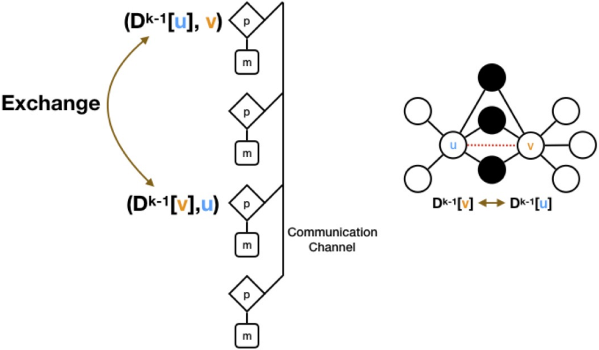
Insert into $\mathcal{D}$ for each vertex

DegreeSketch Neighborhood Update



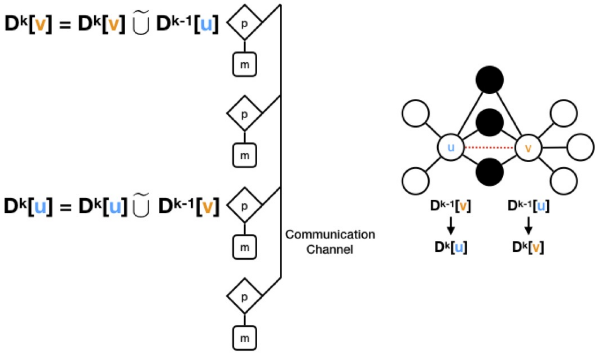
Updates route to each endpoint owner

DegreeSketch Neighborhood Update



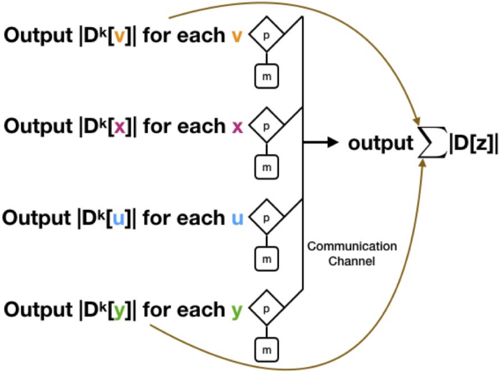
Owners exchange $(k-1)$th skeches

DegreeSketch Neighborhood Update



$(k-1)$th sketches are merged into kth sketches

DEGREESKETCH Neighborhood Update



After kth pass, output all kth local and global estimates

Neighborhood Estimation: Correctness

Theorem 6.3.1

Let $\mu_{r,n}$ and $\eta_{r,n}$ be the multiplicative bias and standard deviation for HLLs given in Theorem 1 of [FFGM07]. The output $\widetilde{\mathcal{N}}(t)$ and $\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)$ for $x \in \mathcal{V}$ at the t-th iteration satisfies

$$\frac{\mathbb{E}\left[\widetilde{\mathcal{N}}(t)\right]}{\mathcal{N}(t)} = \frac{\mathbb{E}\left[\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)\right]}{\mathcal{C}_t^{\mathrm{NBHD}}(x)} = \mu_{r,n} \text{ for } n \to \infty,$$

i.e. they are nearly unbiased.

Furthermore, both also have standard deviation bounded by $\eta_{r,n}$. That is,

$$\frac{\sqrt{\mathrm{Var}\left[\widetilde{\mathcal{N}}(t)\right]}}{\mathcal{N}(t)} \leq \eta_{r,n} \text{ and } \frac{\sqrt{\mathrm{Var}\left[\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)\right]}}{\mathcal{C}_t^{\mathrm{NBHD}}(x)} \leq \eta_{r,n}$$

Neighborhood Estimation: Correctness

Proof of Theorem 6.3.1

For each x, $\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x) = |\mathcal{D}^k[x]|$, where $\mathcal{D}^k[x]$ is a union of HLLs, into which every y such that $d(x, y) < t$ is inserted. Thus by Theorem 1 of [FFGM07],

$$\mathbb{E}\left[\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)\right] = \mu_{r,n}\mathcal{C}_t^{\mathrm{NBHD}}(x)$$

$$\sqrt{\mathrm{Var}\left[\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)\right]} = \eta_{r,n}\mathcal{C}_t^{\mathrm{NBHD}}(x).$$

Thus,

$$\mathbb{E}\left[\widetilde{N}(t)\right] = \sum_{x\in\mathcal{V}} \mathbb{E}\left[\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)\right] = \mu_{r,n} \sum_{x\in\mathcal{V}} \mathcal{C}_t^{\mathrm{NBHD}}(x) = \mu_{r,n}\mathcal{N}(t), \text{ and}$$

$$\sqrt{\mathrm{Var}\left[\widetilde{N}(t)\right]} \leq \sum_{x\in\mathcal{V}} \sqrt{\mathrm{Var}\left[\widetilde{\mathcal{C}}_t^{\mathrm{NBHD}}(x)\right]} \leq \eta_{r,n} \sum_{x\in\mathcal{V}} \mathcal{C}_t^{\mathrm{NBHD}}(x) = \eta_{r,n}\mathcal{N}(t).$$

Another Application: Local Triangle Counting

The Problem:

- Local triangle counting a common big data analytic
  - Exact computation expensive $O\left(m^{\frac{3}{2}}\right)$!
- Recall

$$\mathcal{C}^{\mathrm{TRI}}(x) = |\{yz \in \mathcal{E} \mid xy, yz, xz \in \mathcal{E}\}| \qquad \text{(vertex-local)}$$

$$\mathcal{C}^{\mathrm{TRI}}(xy) = |\{z \in \mathcal{E} \mid xy, yz, xz \in \mathcal{E}\}| \qquad \text{(edge-local)}$$

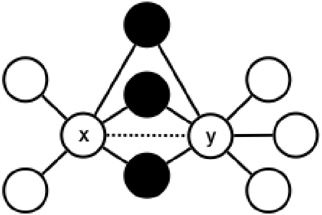
Existing Solutions:

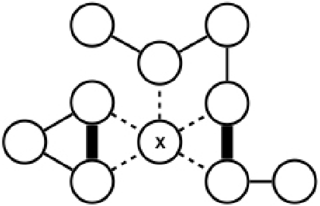
- Many exact distributed algorithms [AKM13, Pea17]
- Many approximate streaming algorithms via sampling [LK15, SERU17]
- ... and some utilizing both models [SHL+18, SLO+18]

Approach: Semi-Streaming Intersection Method

Idea: Intersection method, but using DegreeSketch

- Some cardinality sketches support a limited intersection operation $\widetilde{\cap}$
- Affords edge- and vertex-local triangle count estimation
- Approximate intersection implementation rules out analytic verification à la neighborhood estimation
  - Requires empirical evaluation

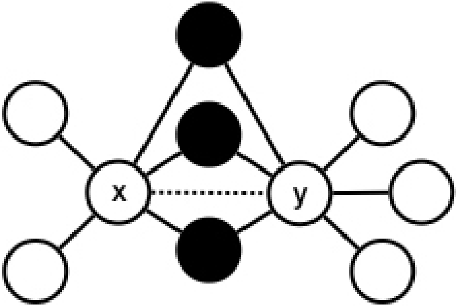




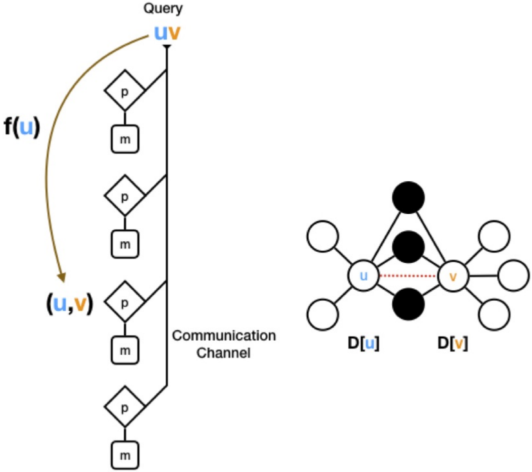
Set operation estimation with HLLs

Streaming sets X and Y with HLLs S_X and S_Y

- $S_X \approx |X|$
- $S_Y \approx |Y|$
- $S_X \widetilde{\cup} S_Y \approx |X \cup Y|$
  - Same error guarantees
- $S_X \widetilde{\cap} S_Y \approx |X \cap Y|$
  - High variance if $|X \cap Y|$ small
  - Optimization arbitrary if $S_X \leq S_Y$ element-wise

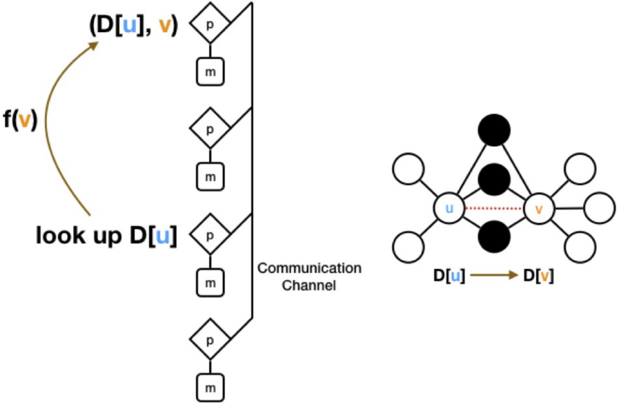


DegreeSketch Query



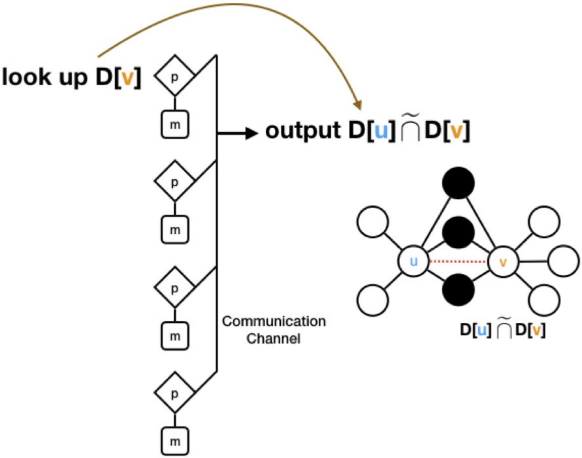
Query routes to one endpoint owner

DegreeSketch Query



Owner sends cardinality sketch to other endpoint owner

DegreeSketch Query



Final owner outputs intersection estimation

DegreeSketch and Triangle Counting

Assume a partition $f : \mathcal{V} \to \mathcal{P}$, and let $\mathcal{V}_P = \{v \in \mathcal{V} \mid f(v) = P\}$

- Distribute DegreeSketch $\mathcal{D}$ across $\mathcal{P}$
  - $\mathcal{D}[v]$ holds a HLL for adjacency set of $v \in \mathcal{V}$
  - P holds $\mathcal{D}[v]$ for $v \in \mathcal{V}_P$
- Accumulate $\mathcal{D}$ in one pass over σ
  - Assume $P \in \mathcal{P}$ gets substream σ_P
  - P sends $xy \in \sigma_P$ to $f(x)$ and $f(y)$
  - When P gets $xy : x \in \mathcal{V}_P$, insert y into $\mathcal{D}[x]$
    - $\mathcal{D}[x]$ starts sparse and eventually saturates
- $\mathcal{D}$ can be queried after estimation, e.g.
  - Estimate $\widetilde{\mathcal{C}}^{\mathrm{DEG}}(v) = \mathrm{ESTIMATE}(\mathcal{D}[v])$
  - Estimate $\widetilde{\mathcal{C}}^{\mathrm{TRI}}(uv) = \mathcal{D}[u] \widetilde{\cap} \mathcal{D}[v]$
    - Involves communication if $f(u) \neq f(v)$
  - Estimate $\widetilde{\mathcal{C}}^{\mathrm{TRI}}(v) = \frac{\sum_{uv \in \mathcal{E}} \widetilde{\mathcal{C}}^{\mathrm{TRI}}(uv)}{2}$
    - Requires second pass in general

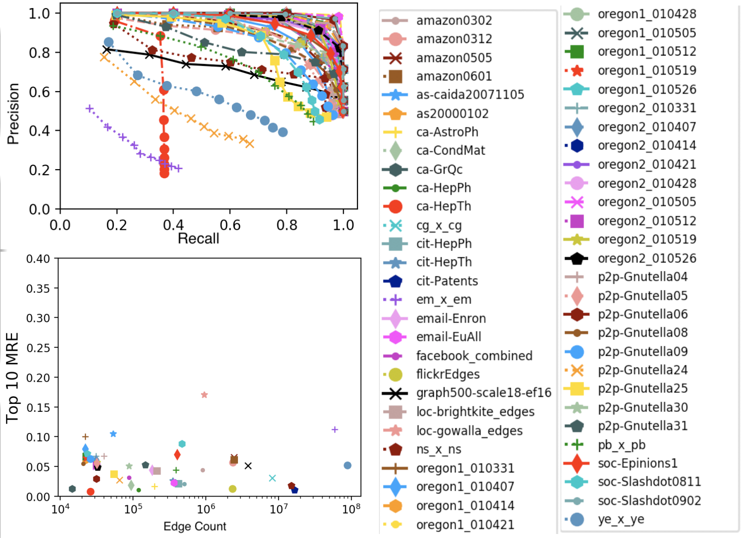
$\widetilde{O}(m)$ time and communication and $\widetilde{O}(\varepsilon^{-2} n)$ space!

Validation of Claims

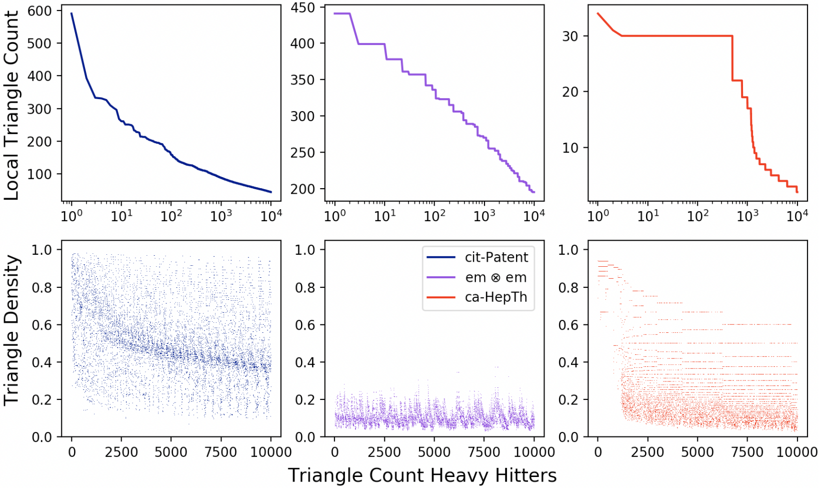
HPEC graph challenge results

Good performance on heavy hitters of *most* graphs

Why do some graphs perform worse?



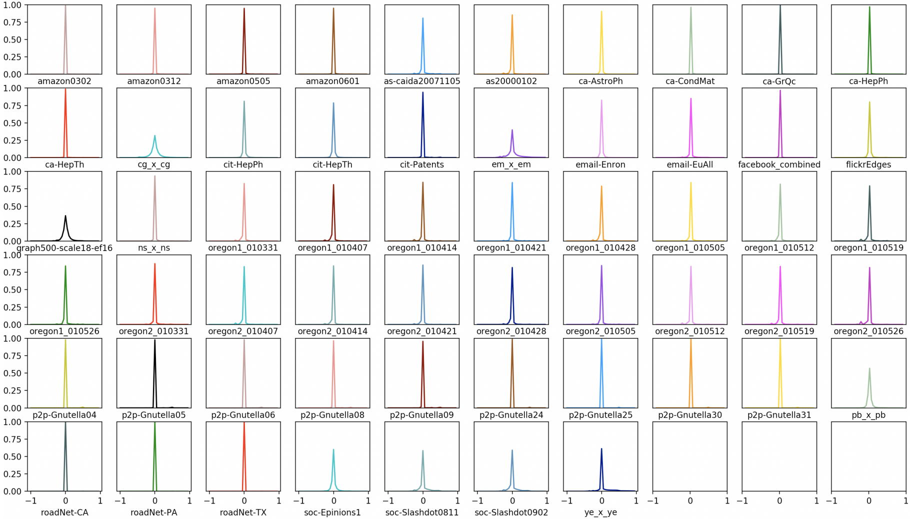
Heavy Hitter Distributions



Low triangle density → high variance

Many ties → poor recovery

Edge Local Relative Error - Experimental Distributions



Good relative error on reasonably sized graphs

Implementation Details

DegreeSketch C++/MPI Library

- Authored by myself
- Utilizes YGM for communication
- Accumulation and query API for DegreeSketch
- Supports sparse and compressed registers
- Implementations for edge- and vertex-local triangle count heavy hitter estimation
- Supports more exotic queries
    - e.g. Intersection of unions

DegreeSketch to be open sourced

Overview

Motivation: Betweenness Centrality Heavy Hitters

The Problem:

- Computing Betweenness centrality exactly amounts to computing AllSourcesAllShortestPaths
  - Expensive $O(mn)$!

Existing Solutions:

- Approximate via a logarithmic number of SingleSourceAllShortestPaths [GMB12, BMS14, Yos14, KMB15, RK16]
  - Difficult to distribute
  - Unclear if possible in $o(m)$ memory

Approach: Semi-Streaming κ-Path Centrality

Idea: "Come at the problem sideways"

- High κ-path centrality empirically correlates with high betweenness centrality [KAS$^+$13]
- Algorithm amounts to sampling random simple paths
  - Sublinearize by accumulating a fixed number of sketches ahead of time
- Sublinear approximation of κ-path centrality $\rightarrow$ emprical recovery of high betweenness centrality vertices?

κ-path centrality

$$\mathcal{C}_\kappa^{\mathrm{PATH}}(x) = \Pr_{p:|p|\leq\kappa}[x \in p \wedge p \text{ a simple path}]$$

"simple path" $=$ non-self-intersecting path

Must simulate many history-avoiding random walks

Parallel Random Walk Simulation - Lower Bound

Lemma (Index Problem)

Alice gets $X \in \{0,1\}^n$ and Bob gets $i \in [n]$. Alice must send $\Omega(n)$ bits for Bob to guess X_i w. p. $> \frac{1}{2}$.

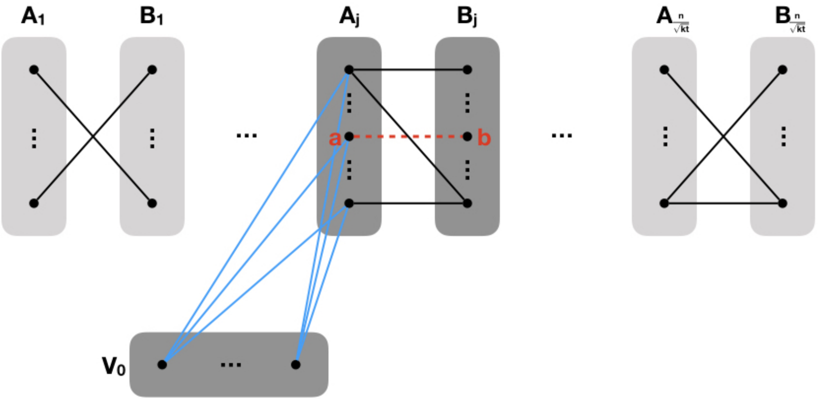
Theorem 7.2.2

For $t = O(n^2)$ and $k = O(n^2)$, simulating k t-step random walks on a simple undirected graph in the insertion-only model within error $\varepsilon = \frac{1}{3}$ requires $\Omega(n\sqrt{kt})$ space.

Proof Sketch

Alice and Bob agree upon an encoding of $X \in \{0,1\}^{n\sqrt{kt}}$ into a graph partitioned into $\frac{n}{\sqrt{kt}}$ bipartite subgraphs with $2\sqrt{kt}$ vertices. Bob's index $i \in \left[n\sqrt{kt}\right]$ identifies one such subgraph, and k simulated random walks of length t allow probabilistic recovery of X_i.

Parallel Random Walk Simulation - Lower Bound



i indicates A_j and B_j, and in particular the edge *ab*. Bob adds the blue edges. and simulates k random walks of length t starting in V_0. If *ab* exists, Bob returns X_i with with probability $> \frac{1}{2}$, $\rightarrow$ solving INDEX.

Parallel Random Walk Simulation - Algorithm Outline

Lemma 7.1.4 (Reservoir Sampling)

Given an insert-only stream σ consisting of n insertions, there is a procedure that uniformly samples $t \leq \frac{n}{2}$ items with replacement in a single pass using $O(t \log(n/t))$ bits of space.

- Split $\mathcal{E}$ based upon endpoint degree for a to-be-specified c
  - $\mathcal{E}_{\mathcal{S}} = \{(x, y) \in \mathcal{E} \mid \mathbf{d}_y \leq c\}$ (important)
  - $\mathcal{E}_{\mathcal{B}} = \{(x, y) \in \mathcal{E} \mid \mathbf{d}_y > c\}$ (unimportant)
- $|\mathcal{E}_{\mathcal{S}}| = O(nc)$, so can store in memory if c small enough
- In a pass over $\mathcal{G}$, sample $O(c)$ unimportant edges per vertex and build the distributed dictionaries:
  - $\mathcal{N}_{\mathcal{S}}[x] = \{(u, v) \in \mathcal{E}_{\mathcal{S}} \mid u = x\}$
  - $\mathcal{N}_{\mathcal{B}}[x] = \{(u, v) \in \mathcal{E}_{\mathcal{B}} \mid u = x \wedge (u, v) \text{ is sampled}\}$
- During each simulation, toss a coin whether to pull from $\mathcal{N}_{\mathcal{S}}$ or $\mathcal{N}_{\mathcal{B}}$ at each step
  - Simulation *fails* on a vertex if it runs out of unimportant samples

Parallel Random Walk Simulation - Correctness Outline

Lemma 7.2.3

Suppose for every $x \in \mathcal{V}$, $\Pr[x \text{ fails} \mid x \text{ a starting vertex}] \leq \delta$. Then $\Pr[\text{any vertex fails}] \leq tk\delta$.

Lemma 7.2.4

$\exists c = O\left(\sqrt{kt} \cdot \frac{q}{\log q}\right)$, where $q = 2 + \frac{\log(1/\delta)}{\sqrt{kt}}$ s. t. for all $x \in \mathcal{V}$

$$\Pr[x \text{ fails} \mid x \text{ a starting vertex, others drawn from } \mu] \leq \delta.^{a}$$

[a] μ is the steady state distribution of $\mathcal{G}$

Theorem 7.2.5

Can simulate k t-step random walks where sources are drawn with replacement from μ in a one pass within error ε using $O\left(n\sqrt{kt}\frac{q}{\log q}\right)$ words of memory, where $q = 2 + \frac{\log(1/\varepsilon)}{\sqrt{kt}}$.

Parallel Random Walk Simulation - Extensions in Brief

- Recording and playback of adjacency substreams
  - Each processor records $\mathcal{M}[x]$ in faster-than-disc external memory[5] while accumulating $\mathcal{N}_{\mathcal{B}}[x]$

$$\mathcal{M}[x] = \{(u, v) \in \mathcal{E}_{\mathcal{B}} \mid u = x\}$$

  - Instead of failing when $\mathcal{N}_{\mathcal{B}}[x]$ runs out of samples, simply take another pass over $M[x]$
    - Partially avoids the steady state distribution heavy hammer
  - I/O versus memory tradeoff
    - Sublinear storage of graph
    - Playbacks incur additional I/O on some processors
- History-Avoiding Walk Simulation
  - Sample via playback, ignoring previous vertices
  - Permits the sublinear space simulation of simple paths

[5] e.g. NVRAM

Semi-Streaming κ-Path Centrality

κ-Path Centrality Approximation Algorithm ([KAS+13]):

1. Simulate $T = 2\kappa^2 n^{1-2\alpha} \ln n$ $(\leq \kappa)$-length simple paths over $\mathcal{G}$
   - maintain $count[x]$ for each $x \in \mathcal{V}$
2. $\widetilde{\mathcal{C}}_\kappa^{\text{PATH}}(x) \leftarrow \frac{count[x]}{2\kappa n^{-2\alpha} \ln n}$

Theorem (κ-Path correctness)

The serial algorithm runs in $O(\kappa^3 n^{2-2\alpha} \log n)$ time and $\Theta(m)$ space, where accuracy parameter $\alpha \in \left[-\frac{1}{2}, \frac{1}{2}\right]$. For each $x \in \mathcal{V}$ it produces estimates $\widetilde{\mathcal{C}}_\kappa^{\text{PATH}}[x]$ such that $\left|\widetilde{\mathcal{C}}_\kappa^{\text{PATH}}[x] - \mathcal{C}_\kappa^{\text{PATH}}[x]\right| \leq n^{\frac{1}{2}+\alpha}$ with probability at least $1 - \frac{1}{n^2}$.

Easy application of distributed, semi-streaming history-avoiding random walk simulation

Summary of Results

- The Goal: distributed semi-streaming approximations of centrality indices
- Engineering Results
  - YGM: Pseudo-Asynchronous Communication Handler
- Algorithmic Results
  - A streaming degree centrality approximation and heavy hitter recovery algorithms
  - A O(1)-pass semi-streaming closeness centrality approximation algorithm
  - 2-pass distributed semi-streaming local neighborhood and triangle count estimation algorithms using DegreeSketch
  - Distributed semi-streaming random walk simulation algorithm
  - Distributed sublinear semi-streaming κ-path centrality estimation algorithm
- Future Work
  - Applications for DegreeSketch
  - Semi-streaming random walk with playback implementation

Dartmouth Publications

1. **Benjamin W. Priest** and George Cybenko. Efficient inference of hidden Markov models from large observation sequences. In *Proceedings of the 2016 SPIE Defense + Security Conference*, SPIE D+S. 2016.
2. **Benjamin W. Priest** and George Cybenko. Approximating centrality in evolving graphs: toward sublinearity. In *Proceedings of the 2017 SPIE Defense + Security Conference*, SPIE D+S. 2017.
3. Luan Hoy Pham, Massimiliano Albanese, and **Benjamin W. Priest**. A quantitative framework to model advanced persistent threats. In *Proceedings of the 15th International Conference on Security and Cryptography*, SECRYPT. 2018. **[Best Paper Award]**
4. **Benjamin W. Priest**, Roger Pearce, and Geoffrey Sanders. Estimating edge-local triangle count heavy hitters in edge-linear time and almost-vertex-linear space. In *Proceedings of the IEEE High Performance Extreme Computing Conference*, HPEC. 2018.
5. **Bejamin W. Priest**, George Cybenko, Satinder Singh, Massimiliano Albanese and Peng Liu. Online and Scalable Adaptive Cyber Defense. In: Michael Wellman (Ed.), *Adversarial and Uncertain Reasoning in Adaptive Cyber-Defense*, ch. 11, pp. xxx–xxx. 2019. **[In Press]**.
6. **Benjamin W. Priest**, Trevor Steil, Roger Pearce, and Geoffrey Sanders. You've Got Mail: Building missing asynchronous communication primitives. In: GraML Workshop at the 2019 IEEE International Symposium on Parallel and Distributed Processing, IPDPS. 2019. **[To Be Submitted]**.
7. Trevor Steil, **Benjamin W. Priest**, Roger Pearce, Keita Iwabuchi, Timothy LaFond, and Geoff Sanders. Massive-scale graph generation with ground truth: Triangles, centrality, and communities. GraPL Workshop at the 2019 IEEE International Symposium on Parallel and Distributed Processing, IPDPS. 2019. **[To Be Submitted]**.
8. **Benjamin W. Priest**, Roger Pearce, and Geoffrey Sanders. DegreeSketch: Distributed cardinality sketches on graphs with applications. **[In Preparation]**.
9. **Benjamin W. Priest**. Efficient distributed and semi-streaming simulation of random walks. **[In Preparation]**.

Questions?

Shaikh Arifuzzaman, Maleq Khan, and Madhav Marathe.
Patric: A parallel algorithm for counting triangles in massive networks.
In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 529–538. ACM, 2013.

Elisabetta Bergamini, Henning Meyerhenke, and Christian L Staudt.
Approximating betweenness centrality in large evolving networks.
In *2015 Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 133–146. SIAM, 2014.

Paolo Boldi, Marco Rosa, and Sebastiano Vigna.
Hyperanf: Approximating the neighbourhood function of very large graphs on a budget.
In *Proceedings of the 20th international conference on World wide web*, pages 625–634. ACM, 2011.

Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F Werneck.
Computing classic closeness centrality, at scale.
In *Proceedings of the second ACM conference on Online social networks*, pages 37–50. ACM, 2014.

Reuven Cohen, Liran Katzir, and Aviv Yehezkel.
A minimal variance estimator for the cardinality of big data set intersection.
In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 95–103. ACM, 2017.

Otmar Ertl.
New cardinality estimation algorithms for hyperloglog sketches.
arXiv preprint arXiv:1702.01284, 2017.

Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier.
Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm.
In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

Oded Green, Robert McColl, and David A Bader.
A fast algorithm for streaming betweenness centrality.

In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 11–20. IEEE, 2012.

Stefan Heule, Marc Nunkesser, and Alexander Hall.
Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm.
In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 683–692. ACM, 2013.

Ce Jin.
Simulating random walks on graphs in the streaming model.
arXiv preprint arXiv:1811.08205, 2018.

Nicolas Kourtellis, Tharaka Alahakoon, Ramanuja Simha, Adriana Iamnitchi, and Rahul Tripathi.
Identifying high betweenness centrality nodes in large social networks.
Social Network Analysis and Mining, 3(4):899–914, 2013.

Nicolas Kourtellis, Gianmarco De Francisci Morales, and Francesco Bonchi.

Scalable online betweenness centrality in evolving graphs.
IEEE Transactions on Knowledge and Data Engineering, 27(9):2494–2506, 2015.

Kevin J Lang.
Back to the future: an even more nearly optimal cardinality estimation algorithm.
arXiv preprint arXiv:1708.06839, 2017.

Yongsub Lim and U Kang.
Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams.
In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 685–694. ACM, 2015.

Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski.
Pregel: a system for large-scale graph processing.
In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

Eisha Nathan.
Numerical and Streaming Analyses of Centrality Measures on Graphs.
PhD thesis, Georgia Institute of Technology, 2018.

Roger Pearce.
Triangle counting for scale-free graphs at scale in distributed memory.
In *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*, pages 1–4. IEEE, 2017.

Roger Pearce, Maya Gokhale, and Nancy M Amato.
Faster parallel traversal of scale free graphs at extreme scale with vertex delegates.
In *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, pages 549–559. IEEE, 2014.

Christopher R Palmer, Phillip B Gibbons, and Christos Faloutsos.
Anf: A fast and scalable tool for data mining in massive graphs.
In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90. ACM, 2002.

Benjamin W. Priest, Roger Pearce, and Geoffrey Sanders.

Estimating edge-local triangle count heavy hitters in edge-linear time and almost-vertex-linear space.
In *High Performance Extreme Computing Conference (HPEC), 2018 IEEE*. IEEE, 2018.

Benjamin W. Priest, Roger Pearce, and Geoffrey Sanders.
DegreeSketch: Distributed cardinality sketches on graphs, with applications to counting triangles.
In *In Preparation for ACM SigKDD 2019*. ACM, 2019.

Jason Qin, Denys Kim, and Yumei Tung.
Loglog-beta and more: A new algorithm for cardinality estimation based on loglog counting.
arXiv preprint arXiv:1612.02284, 2016.

Matteo Riondato and Evgenios M Kornaropoulos.
Fast approximation of betweenness centrality through sampling.
Data Mining and Knowledge Discovery, 30(2):438–475, 2016.

Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal.

Trièst: Counting local and global triangles in fully dynamic streams with fixed memory size.
ACM Transactions on Knowledge Discovery from Data (TKDD), 11(4):43, 2017.

Kijung Shin, Mohammad Hammoud, Euiwoong Lee, Jinoh Oh, and Christos Faloutsos.
Tri-Fly: Distributed estimation of global and local triangle counts in graph streams.
In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 651–663. Springer, 2018.

Kijung Shin, Euiwoong Lee, Jinoh Oh, Mohammad Hammoud, and Christos Faloutsos.
Dislr: Distributed sampling with limited redundancy for triangle counting in graph streams.
arXiv preprint arXiv:1802.04249, 2018.

Daniel Ting.

Towards optimal cardinality estimation of unions and intersections with sketches.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1195–1204. ACM, 2016.

Wei Wei and Kathleen Carley.
Real time closeness and betweenness centrality calculations on streaming network data.
In *Proceedings of the 2014 ASE Big-Data/SocialCom/Cybersecurity Conference, Stanford University*, 2014.

Qingjun Xiao, You Zhou, and Shigang Chen.
Better with fewer bits: Improving the performance of cardinality estimation of large data streams.
In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

Yuichi Yoshida.
Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches.

In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1416–1425. ACM, 2014.