Beyond Triangles: A Distributed Framework for Estimating 3-profiles of Large Graphs

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

University of Texas, Austin, USA

August 12, 2015

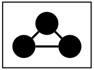
Introduction

- Perform analytics on large graphs
  - World Wide Web, social networks, bioinformatics
- More descriptive than triangle count, clustering coefficient
- Scalable, distributed algorithms

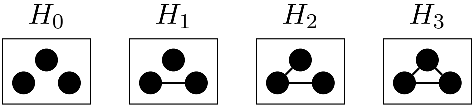
3-profile

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

H_3

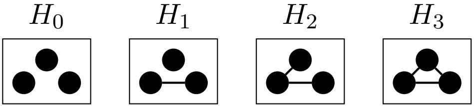


- Count the induced subgraphs formed by selecting all triples of vertices



3-profile

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



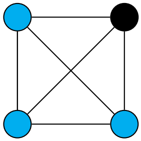
Definition

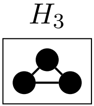
Let n_i be the number of H_i's in a graph G. The vector $\mathbf{n}(G) = [n_0, n_1, n_2, n_3]$ is called the 3-profile of G.

- Always sums to $\binom{|V|}{3}$, the total number of 3-subgraphs

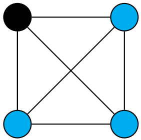
Examples

- 4-clique: $\mathbf{n}(K_4) = [0, 0, 0, 4]$

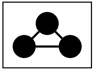




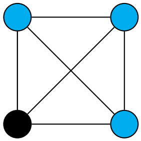
- 4-clique: $\mathbf{n}(K_4) = [0, 0, 0, 4]$



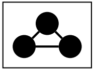
H_3



- 4-clique: $\mathbf{n}(K_4) = [0, 0, 0, 4]$

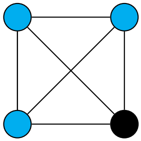


H_3

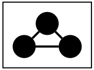


Examples

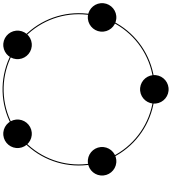
- 4-clique: $\mathbf{n}(K_4) = [0, 0, 0, 4]$



H_3

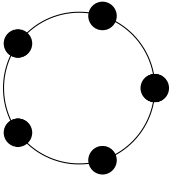


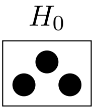
- 5-cycle: $\mathbf{n}(C_5) = [?, ?, ?, ?]$



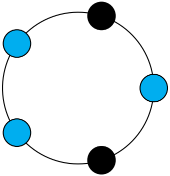
Examples

- 5-cycle: $\mathbf{n}(C_5) = [0, ?, ?, ?]$

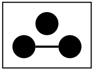




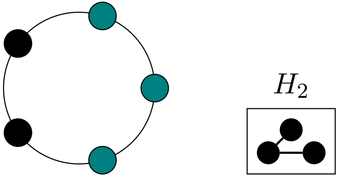
- 5-cycle: $\mathbf{n}(C_5) = [0, 5, ?, ?]$



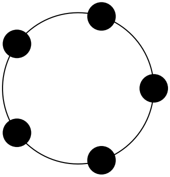
H_1

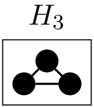


- 5-cycle: $\mathbf{n}(C_5) = [0, 5, 5, ?]$



- 5-cycle: $\mathbf{n}(C_5) = [0, 5, 5, 0]$





Related Terms

For each $v \in V$:

Definition

The local 3-profile counts how many times v participates in each H_i with 2 other vertices.

For each $v \in V$:

Definition

The local 3-profile counts how many times v participates in each H_i with 2 other vertices.

Definition

The ego 3-profile is the 3-profile of ego graph $N(v)$.

- Graph induced by set of neighbors $\Gamma(v)$

Motivation

- Global 3-profile concisely describes local connectivity
  - Molecule classification
- Local and ego 3-profiles are feature vectors for each vertex
  - Spam detection
  - Generative models

Introduction

- Problem: Compute (or approximate) 3-profile quantities for a large graph

- Problem: Compute (or approximate) 3-profile quantities for a large graph

- Approach: Edge sub-sampling and distributed implementation

Contributions

1. Derive a 3-profile *sparsifier* with provable guarantees
2. Design distributed, graph engine algorithms to calculate local and ego 3-profiles
3. Evaluate performance on real-world datasets

Related Work

Well studied across several communities:

- Graph sub-sampling
  [Kim, Vu '00] [Tsourakakis, et al. '08 -'11] [Ahmed, et al. '14]
- Large-scale triangle counting
  [Satish, et al. '14] [Shank '07] [Suri, Vassilvitskii '11]
- Subgraph counting
  [Alon, et al. '97] [Kloks, et al. '00] [Kowaluk, et al. '13]
- Graphlets
  [Pržulj '07] [Shervashidze, et al. '09]

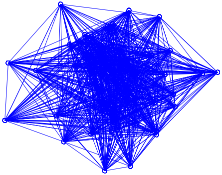
Outline

Edge Sub-sampling Process

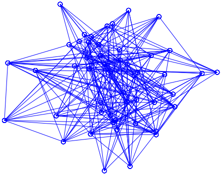
- Sub-sample each edge in the graph independently with probability p
- Relate the original and sub-sampled graphs via a 1-step Markov chain

Edge Sub-sampling Process

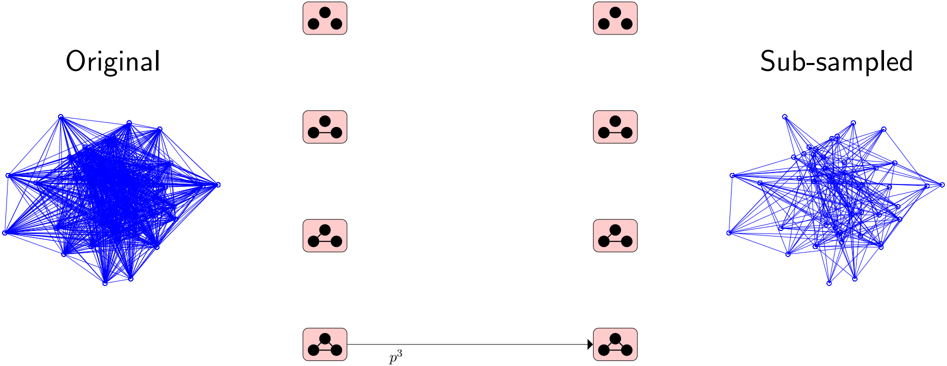
Original



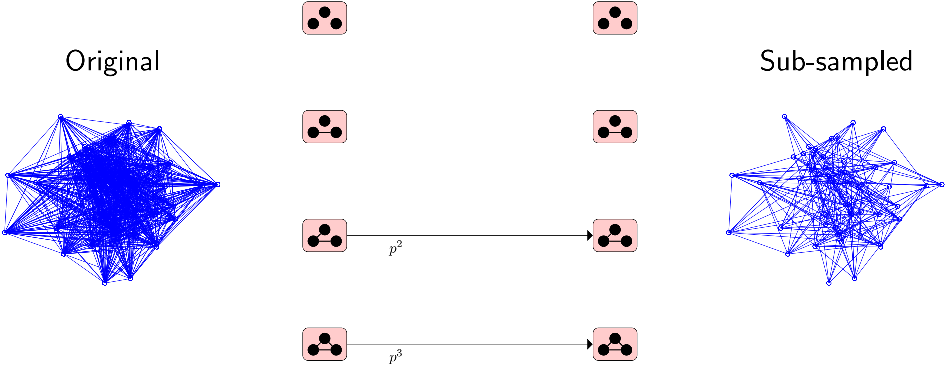
Sub-sampled

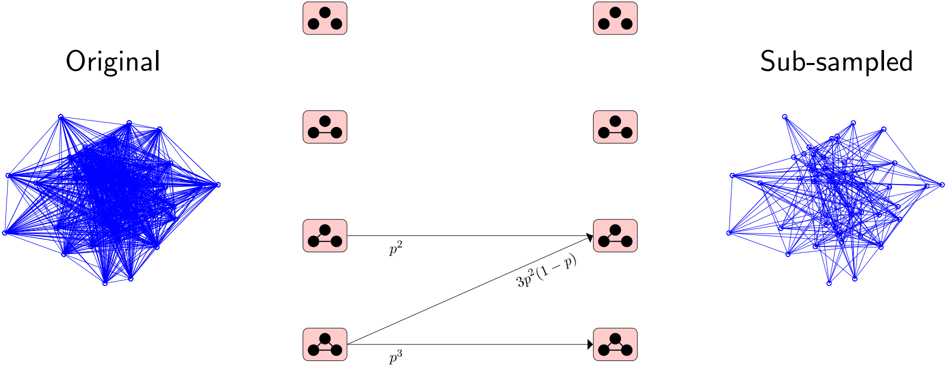


Edge Sub-sampling Process

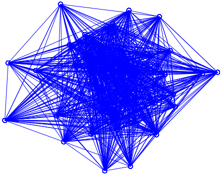


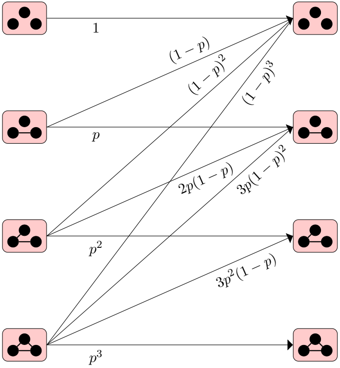
Edge Sub-sampling Process



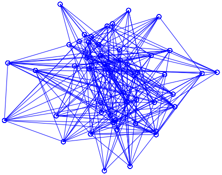

Original
Sub-sampled
p^2
$3p^2(1-p)$
p^3

Original



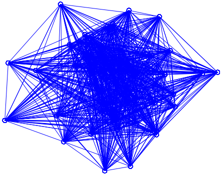


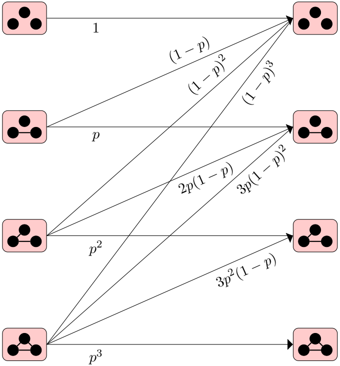
Sub-sampled



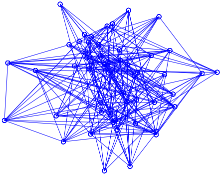
Edge Sub-sampling Process

Original





Sub-sampled



$$\begin{bmatrix} \text{Estimator} \end{bmatrix} = \begin{bmatrix} 1 & 1-p & (1-p)^2 & (1-p)^3 \\ 0 & p & 2p(1-p) & 3p(1-p)^2 \\ 0 & 0 & p^2 & 3p^2(1-p) \\ 0 & 0 & 0 & p^3 \end{bmatrix}^{-1} \begin{bmatrix} \text{Sub-sampled} \end{bmatrix}$$

Main Result

Theorem (*3-profile sparsifiers*)

For all (ϵ,p)-balanced graphs, the l_∞-norm of the 3-profile sparsifier error is bounded by $\epsilon\binom{|V|}{3}$ with high probability.*

Theorem (*3-profile sparsifiers*)

For all (ϵ,p)-balanced graphs, the l_∞-norm of the 3-profile sparsifier error is bounded by $\epsilon\binom{|V|}{3}$ with high probability.*

Definition

A graph is (ϵ,p)-balanced if the majority of "triangles," "wedges," or "single-edges" do not depend on one common edge.

Main Result

Theorem (*3-profile sparsifiers*)

For all (ϵ,p)-balanced graphs, the l_∞-norm of the 3-profile sparsifier error is bounded by $\epsilon\binom{|V|}{3}$ with high probability.*

Definition

A graph is (ϵ,p)-balanced if the majority of "triangles," "wedges," or "single-edges" do not depend on one common edge.

Proof Sketch:

- Apply multivariate polynomial concentration inequalities [Kim, Vu '00] to each estimator

$$f(G, p) = e_1e_2e_4 + e_4e_5e_6 + \dots$$

Outline

3-PROF

Vertex program in the Gather-Apply-Scatter framework

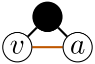
Vertex program in the Gather-Apply-Scatter framework

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

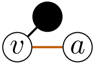
Vertex program in the Gather-Apply-Scatter framework

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

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



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



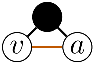
3-PROF

Vertex program in the Gather-Apply-Scatter framework

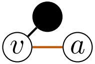
1. For each vertex v: Gather and Apply vertex IDs to store $\Gamma(v)$

2. For each edge va: Scatter

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

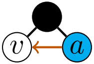


$$n^c_{2,va} = |\Gamma(v)| - |\Gamma(v) \cap \Gamma(a)| - 1, \ \ldots$$

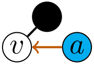


3. For each vertex v: Gather and Apply

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



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



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

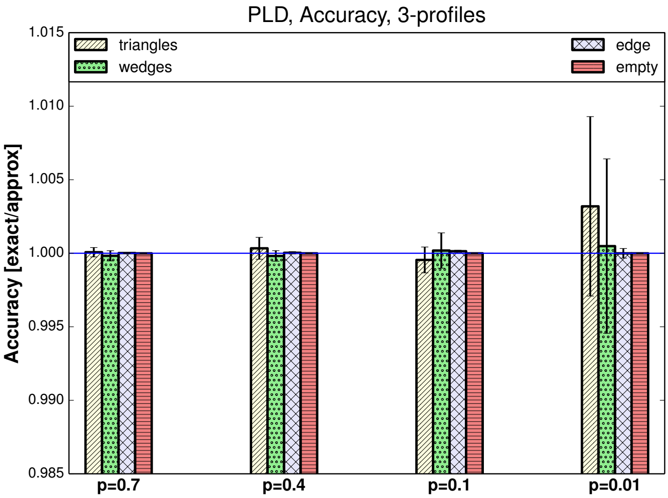
Implementation

- 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

Datasets

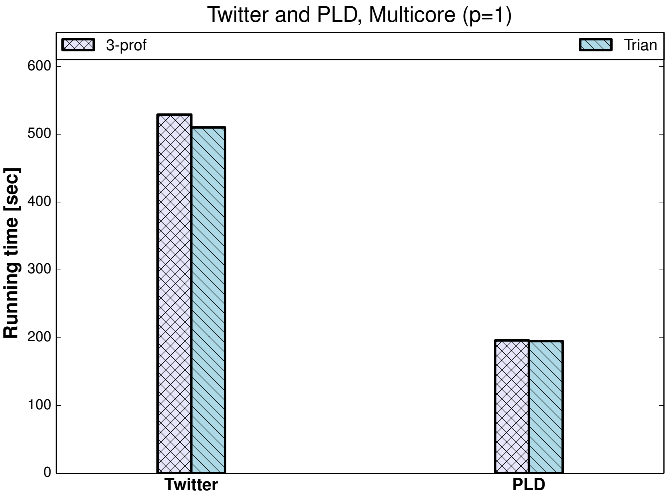
Name	Vertices	Edges (undirected)
Twitter	$41,652,230$	$1,202,513,046$
PLD	$39,497,204$	$582,567,291$
LiveJournal	$4,846,609$	$42,851,237$
Wikipedia	$3,515,067$	$42,375,912$
DBLP	$317,080$	$1,049,866$

Results: 3-profile Sparsifier Accuracy, 5 runs



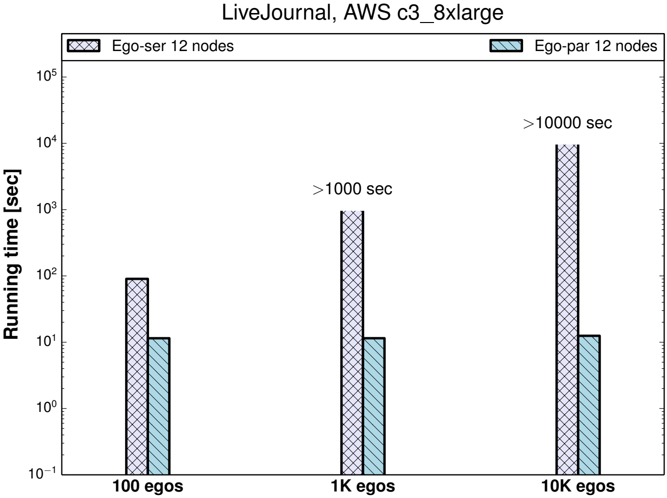
Results: Multicore, 3 runs

Compare *3-PROF* to GraphLab's default triangle count

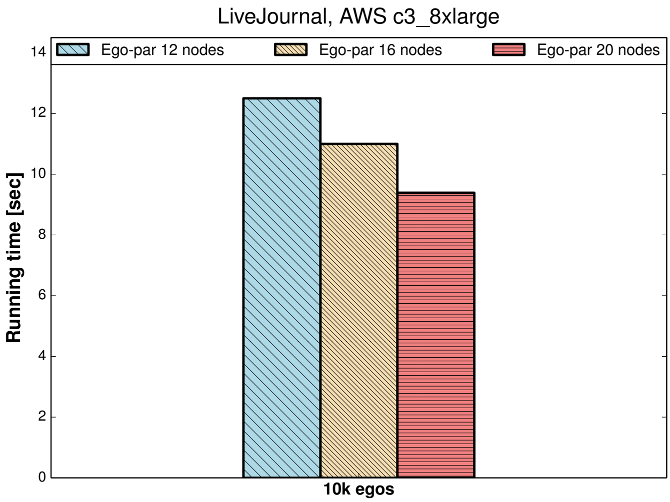


Results: AWS, 5 runs

Compare *EGO-PAR* to naive, serial algorithm (*EGO-SER*)



Results: AWS, 5 runs



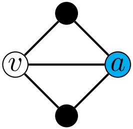
Outline

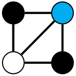
Summary

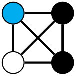
1. Edge sub-sampling produces fast, accurate 3-profile estimates
2. 3-profile counting consumes roughly the same resources as triangle counting
3. Distributed algorithms scale well over large data and large computing clusters

`github.com/eelenberg/3-profiles`

(Backup) Edge Pivot Equations

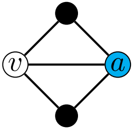






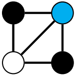
$$\sum_{a\in\Gamma(v)} \binom{n_{3,va}}{2} = F_2(v) + 3F_3(v)$$

(Backup) Edge Pivot Equations



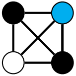
$\sum_{a \in \Gamma(v)} \binom{n_{3,va}}{2}$

$=$

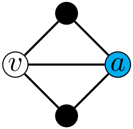


$F_2(v)$

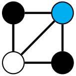
$+$

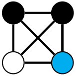


$3F_3(v)$

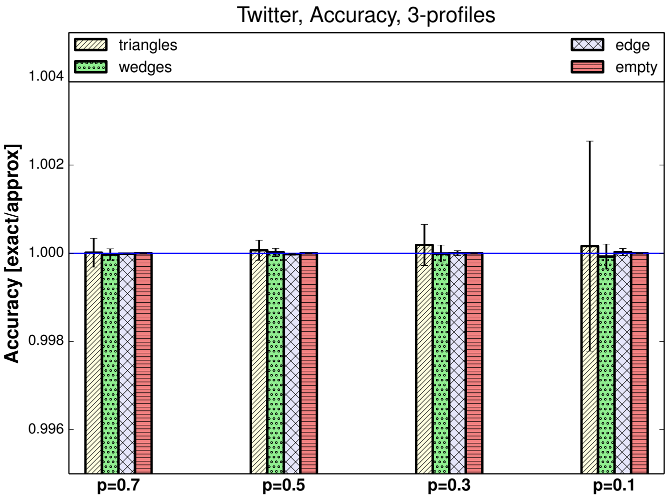


$$\sum_{a\in\Gamma(v)} \binom{n_{3,va}}{2} = F_2(v) + 3F_3(v)$$

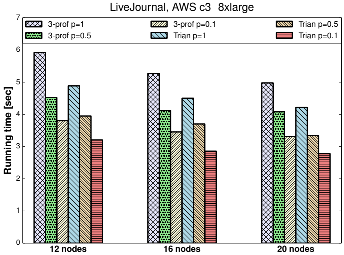




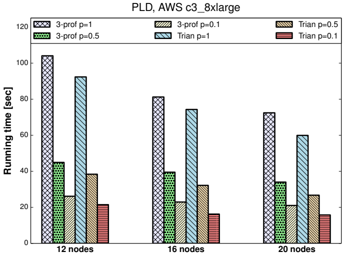
(Backup) Results: 3-profile Sparsifier Accuracy, 5 runs



(Backup) Results: *3-PROF* vs. *TRIAN*, AWS, 3 runs

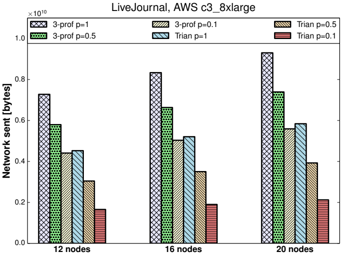


LiveJournal Running Time

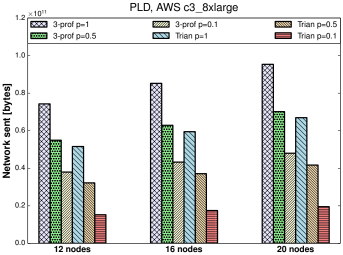


PLD Running Time

(Backup) Results: *3-PROF* vs. *TRIAN*, AWS, 3 runs

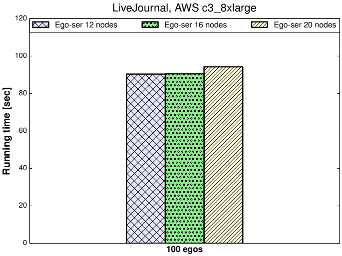


LiveJournal Network Usage

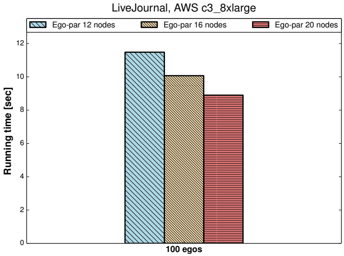


PLD Network Usage

(Backup) Results: AWS, 5 runs



EGO-SER



EGO-PAR