# Improving I/O Complexity of Triangle Enumeration

**Yi Cui, Di Xiao, Daren B.H. Cline, and Dmitri Loguinov**

Internet Research Lab (IRL)
Department of Computer Science and Engineering
Texas A&M University, College Station, TX, USA 77843
November 20, 2017

# Agenda

- Introduction
- Background
- Analysis of Previous Work
  - Pagh and Pruned Companion Files (PCF)
  - Comparison
- Trigon
- Experiments

# Introduction

- Problem definition: Given a simple undirected graph $G = (V, E)$ with $m$ edges and $n$ nodes, find all three-node tuples $(u, v, w)$, such that there exists an edge between any two of them
- Triangles are important in data mining
  - Clustering coefficient, graphics, databases
  - Spam/community detection, theory of complexity
- Challenges: With the explosion of big data, modern graphs normally do not fit in memory
  - Google web graphs consist of trillions of edges
  - Facebook maintains social networks of billions of users

# Agenda

- Introduction
- Background
- Analysis of Previous Work
  - Pagh and Pruned Companion Files (PCF)
  - Comparison
- Trigon
- Experiments

# Background

- There are 3! = 6 ways to list each triangle according to different orders of its three nodes

- To avoid duplicates and improve efficiency, preprocessing is required to convert the input graph into a directed version:
  - Relabeling: Shuffle nodes with some permutation, then sequentially label nodes from $1$ to $n$
  - Acyclic orientation: Direct edges from nodes with larger labels to those with smaller
  - Neighbors of each node are split into out-neighbors with smaller labels than source and in-neighbors with larger labels, and the graph is split into out-graph and in-graph

# Background

- Given $n$ nodes, there exist $n!$ different permutations, which can split neighbor lists in different ways
  - Which ones achieve optimal triangle-listing cost?
- Our previous studies [Cui16], [Xiao17] reveal 18 triangle-enumeration methods and model their in-memory cost under optimal permutations
  - Descending-degree permutation with edge-iterator $E_1$ is identified as the best in-memory solution
  - See paper for details
- This work assumes usage of $E_1$ and focuses on I/O performance

# Agenda

- Introduction
- Background
- Analysis of previous work
  - Pagh and Pruned Companion Files (PCF)
  - Comparison
- Trigon
- Experiments

# Analysis of Previous Work

- A majority of previous work, e.g., MGT [Hu13] and its successors, assumes a simple I/O model:
  - Given memory size $M$, in each iteration, load a size $M$ chunk of the graph into memory, scan the rest from disk
  - Requires quadratic I/O complexity $m^2/M$
  - Does not scale well for large graphs
- More recent work proposes two methods that achieve much better I/O than quadratic
  - Pagh (PODS 2014)
  - Pruned Companion Files (PCF, ICDM 2016)

# Pagh

- Pagh *randomly* colors nodes with $c$ colors
  - Creates $c$ partitions of nodes and $c^2$ partitions of edges
- To detect all triangles, the method must consider all $c^3$ different combination of colors
- Since Pagh does not have a reference implementation, we develop our version that works with $E_1$ and oriented graphs
  - We call this method Pagh+ since it achieves the best I/O constants in the literature, i.e., $2m^{1.5}/\sqrt{M}$
- Always better than MGT, but some drawbacks exist
  - Requires special handling and complex algorithms for large-degree nodes (e.g., in star graphs)

# Pruned Companion Files (PCF)

- PCF splits nodes *sequentially* into $p$ mutually exclusive and jointly exhaustive subsets $V_1, \ldots, V_p$
  - Edges are then partitioned by either destination (PCF-A) or source (PCF-B) nodes
  - PCF achieves deterministic load-balancing and requires $p = m/M$ partitions
- A special companion file is created for each subgraph, which is scanned sequentially from disk
  - The size of all companion files determines the amount of I/O
  - The paper goes into extensive modeling of PCF I/O under its optimal permutation and different scaling rates of RAM size, average degree, and variance of out-degree as $n \to \infty$
  - See the paper as the model is quite complex

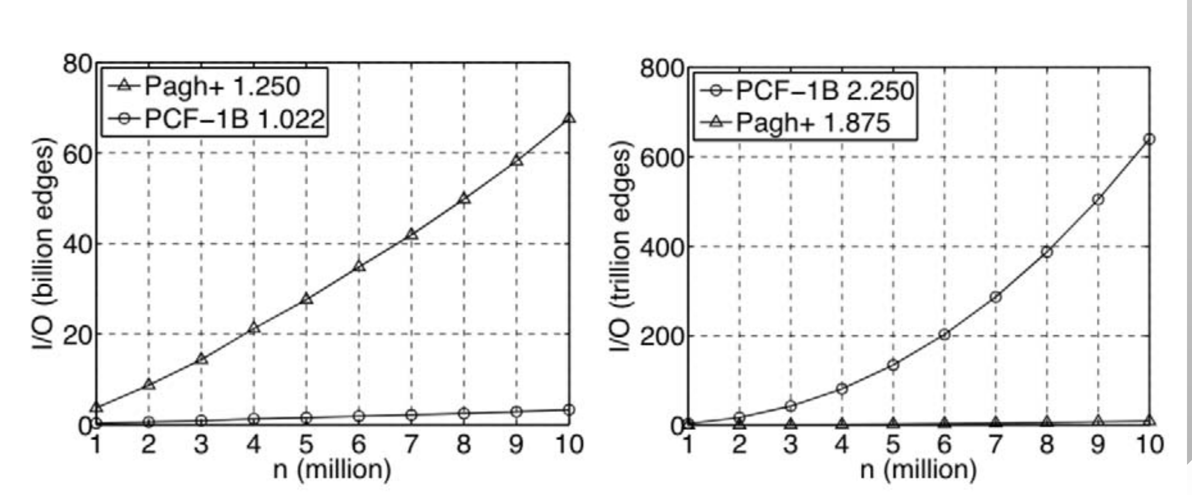
# Agenda

- Introduction
- Background
- Analysis of previous work
  - Pagh and Pruned Companion Files (PCF)
  - Comparison
- Trigon
- Experiments

# Comparison

- Our comparison shows that neither Pagh+ nor PCF is asymptotically better than the other
  - PCF has less I/O if the graph is sparse, out-degree variance is small, or graph size is large compared to memory
  - Pagh is better when the conditions are reversed
  - Each method can beat the other by $\sqrt{n}$

model predicts Pagh+ is worse by $n^{0.25}$




sparse graphs with constant average degree

dense graphs with average degree $n^{0.5}$

model predicts PCF is worse by $n^{3/8}$



Computer Science, Texas A&M University

# Comparison

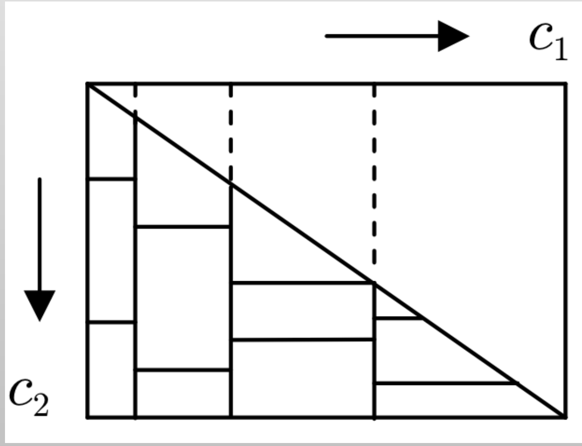
- An ideal method should combine the strengths of Pagh+ and PCF, i.e.,
  - Prevent redundant edges from being loaded into RAM
  - Split each neighbor list into at most $\sqrt{p}$ files
  - Use sequential ranges to decide partitioning
  - Deterministically load-balance subgraphs
  - Be able to operate with $O(1)$ memory
  - Handle special cases (e.g., star graphs) without additional workarounds
- By doing so, it should also beat both previous methods in terms of I/O
  - We next offer such an approach

# Agenda

- Introduction
- Background
- Analysis of Previous Work
  - Pagh and Pruned Companion Files (PCF)
  - Comparison
- Trigon
- Experiments

# Trigon

- Idea: apply 2D sequential partitioning with $c_1$ primary colors along destinations nodes and $c_2$ secondary colors along source nodes




- Because of orientation, only the bottom half of the matrix is split
  - Each partition can be a rectangle, triangle, or trapezoid in the picture
- This creates $c_1 c_2 = p$ subgraphs
  - The paper shows how to achieve deterministic load-balancing
  - Similar to PCF, a companion file is created for each subgraph
  - A model is derived for the size of companion files



Computer Science, Texas A&M University

# Trigon

- With $c_1 = 1$, Trigon becomes PCF-B and with $c_2 = 1$ it is exactly PCF-A (i.e., they are special 1D cases)
- We also show that Trigon beats Pagh+ when $c_1 = c_2 = \sqrt{p}$
  - Thus, with an optimal choice of $(c_1, c_2)$, Trigon's I/O is always no worse than either of its predecessors
- The paper also takes into account the number of hash-table lookups and intersection, where Trigon again beats the previous methods
- The derived models can be used to decide the best $c_1$ for each $G$, while $p = m/M$ and $c_2 = p/c_1$ are known



Computer Science, Texas A&M University

# Agenda

- Introduction
- Background
- Analysis of Previous Work
  - Pagh and Pruned Companion Files (PCF)
  - Comparison
- Trigon
- Experiments

Computer Science, Texas A&M University

# Experiments

- Experiment setup: single 3-TB magnetic hard drive that can read @ 160 MB/sec
- Datasets

| Graphs | Nodes | Edges | Size | Triangles |
|---|---|---|---|---|
| Twitter | 41M | 1.2B | 9.3 GB | 35B |
| Yahoo | 720M | 6.4B | 53.3 GB | 86B |
| IRL-domain | 86M | 1.7B | 13.3 GB | 133B |
| IRL-host | 642M | 6.4B | 52.7 GB | 437B |
| IRL-ip | 1.6M | 818M | 6.1 GB | 1040B |
| ClueWeb | 8.2B | 51B | 358 GB | 879B |
| Complete | 100K | 5.0B | 37.2 GB | 167T |
| Bipartite | 100K | 2.5B | 18.6 GB | 0 |

# Experiments

- Comparison of I/O (billion edges)

| Graphs | p | Pagh+ | PCF | Trigon |
|---|---|---|---|---|
| Twitter | 1,024 | 75.6 | 43.5 | 19.5 |
| Yahoo | 1,024 | 392.3 | 25.5 | 25.5 |
| IRL-domain | 1,024 | 104.8 | 98.4 | 33.8 |
| IRL-host | 1,024 | 386.5 | 137.9 | 59.7 |
| IRL-ip | 1,024 | 51.5 | 145.7 | 23.4 |
| ClueWeb | 1,024 | 2,869.9 | 457.1 | 326.2 |
| Complete | 10,000 | 995.0 | 15,742 | 493.0 |
| Bipartite | 10,000 | 497.0 | 2.5 | 2.5 |

  - On real graphs, Trigon beats Pagh+ by up to 15x and PCF by up to 6x; on the complete graph, it is better than PCF by 32x and on the bipartite graph it needs 200x less I/O than Pagh+
  - For the actual runtime and other metrics, see the paper

# Thank you!
# Any questions?