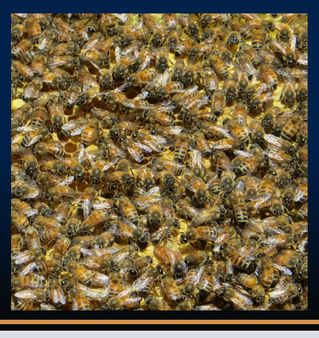


Concurrent Katz Centrality for Streaming Graphs

Chunxing Yin and Jason Riedy

Georgia Tech School of Computational Science and Engineering

Georgia Tech

1. Streaming Graph Analysis

Streaming updates to a single massive graph.

Applications:

- Social Networks
  - Identify *communities*, influences, bridges, trends, anomalies (trends *before* they happen)...
  - Potential to help social sciences, city planning, and others with large-scale data.
- Cybersecurity
  - Determine if new connections can access a device or represent new threat in < 5ms...
  - Is the transfer by a virus / persistent threat?
- Bioinformatics, health
  - Construct gene sequences, analyze protein interactions, map brain interactions
- Credit fraud forensics ⇒ detection ⇒ monitoring
  - Real-time integration of all the customer's data

2. Data Rates

Computer network analysis:

- Gigabit ethernet: 81k – 1.5M packets per second
- Over 130 000 flows per second on 10 GigE (< 7.7 μs)

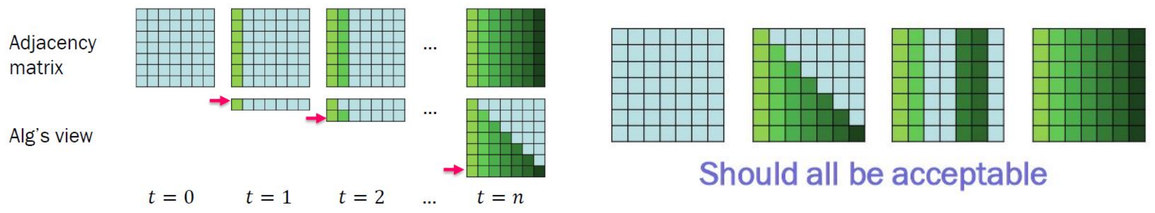
Social network analysis:

- 500M posts per day on Twitter (6k / sec)
- 3M posts per minute on Facebook (50k / sec)

3. Concurrent Execution Model

What can we say about algorithms executing concurrently with graph changes?

An algorithm is **valid** in our model if the result is correct for the initial graph plus *some* unspecified subset of concurrent changes.

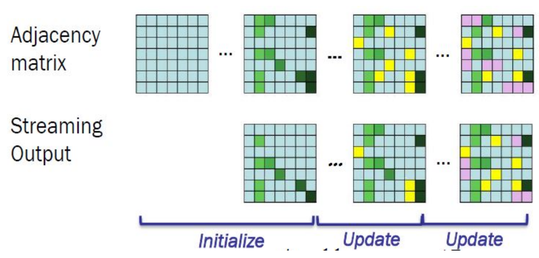


Can we **update** graph metrics as new data arrives?

- Track what changed during the initialization.
- Update locally around those changes, while changes occur concurrently.
- If the update algorithm is valid, can repeat to follow a streaming graph.

Examples:

- connected components, triangle counting
- PageRank, Katz: Jacobi for refinement



4. Katz centrality

Measure the relative influence of vertices by counting the total number of walks between a pair of vertices

$$C = ((I - \alpha A)^{-1} - I)\vec{1}$$

Katz centrality vector C is the solution of linear system

$$(I - \alpha A)x = \vec{1}$$

A valid updating Katz centrality means, for each time t_i,

$$(I - \alpha(G_{t_i} + \delta_i))x_i = \vec{1}$$

5. Theoretical Results

1. Let S_v be the sub-stream of all visible changes in an input stream S, then an algorithm is valid if and only if for any input stream S, the algorithm is valid for the *visible* substream S_v.
2. If an algorithm is invalid, then there exists a single change input stream S that makes the algorithm invalid for S.
3. For any algorithm producing a subgraph and without a snapshot view as input, there exist a graph G and a finite input stream S such that $OUT(G, S) \neq OUT(G_i)$ for any $0 \leq i \leq k$, where G_i's are all possible snapshots.
   - Without being *given* a snapshot view, you **cannot** guarantee a snapshot result.
   - Validity is the most stringent possible definition without snapshot graph views

6. Katz Centrality Algorithm

Algorithm 1 Jacobi method for solving $Mx = b$

Input: Graph $H^{(k)}$
Output: Katz centrality vector $x^{(k)}$

```
k = 0
x^(0) = 0
r^(0) = +∞
D = diag(M)
R = M − D
while r^(k) ≥ ε do
    x^(k+1) = D^−1 (b − R x^(k))
    r^(k+1) = b − M x^(k+1)
    k += 1
end while
return x^(k+1), r^(k+1)
```

Verify solution $x^{(k+1)}$ against currently traversed graph

Algorithm 2 Update x and r with given Δ

Input: Graph H_i, previous solution x_i, batch Δ_i
Output: Updated Katz centrality vector x_{i+1}

```
r_i = 1 − (I − αH_i)x_i   ← saved from previous round
r̃_{i+1} = r_i + αΔ_i x_i
Δx = JACOBI(I − αH_{i+1}, r̃_{i+1}, tol)
x_{i+1} = x_i + Δx
Δr = αΔ_i x_i − (I − αH_{i+1})Δx
r_{i+1} = r_i + Δr
return x_{i+1}
```

Approximated residual

Save from Jacobi

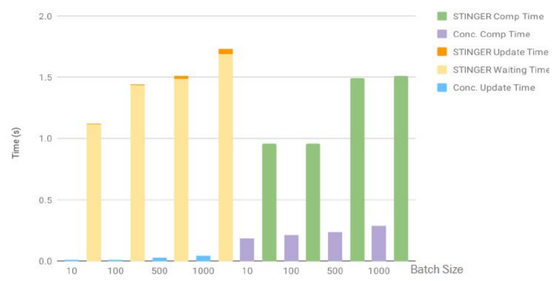
The absolute error from the approximations are bounded by

$$\frac{nnz(\delta_{i+1}) + nnz(\delta_i)}{\sqrt{|E|}}\|x_i\|,$$

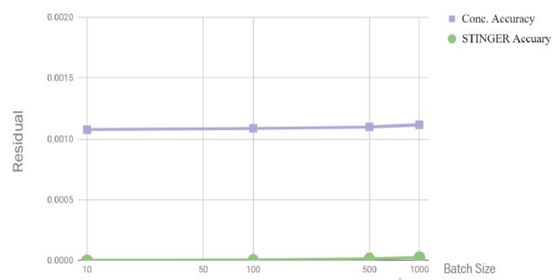
which is related the fraction of edges that are modified concurrently. Thus the error is constantly bounded and does not accumulate over time.

7. Performance

- Implementation is based on STINGER (Georgia Tech's streaming semantic graph analysis framework). Graph updates include insertions, removals, and weight changes.
- Overall, we reduced the delay to results from 11.3x to 179x while maintaining same accuracy on real world graphs



Time usage of two models



Accuracy of two models

8. Future Work

- Defining a *fair* comparison between the synchronous and non-stop models.
  - Not only aggregate updates per second, but also accuracy!
  - Theory work ongoing as well.
- Using the model to pass extracted graphs to arbitrary analysis code.
  - Many algorithms do not scale to billions of vertices.
  - Accelerators, remote systems, *etc*.
- Extending the model for memory coherence...
  - For how long can Pregel-like algorithms use old information?
  - Similar for PGAS / DSM systems.