Updating PageRank for Streaming Graphs

E. Jason Riedy
Graph Algorithms Building Blocks
23 May 2016

School of Computational Science and Engineering
Georgia Institute of Technology
Streaming Graph Analysis
- **Cyber-security** Identify anomalies, malicious actors
- **Health care** Finding outbreaks, population epidemiology
- **Social networks** Advertising, searching, grouping
- **Intelligence** Decisions at scale, regulating algorithms
- **Systems biology** Understanding interactions, drug design
- **Power grid** Disruptions, conservation
- **Simulation** Discrete events, cracking meshes

- Graphs are a motif / theme in data analysis.
- Changing and *dynamic* graphs are important!

Riedy, GABB 2016
1. Motivation and background

2. Incremental PageRank
   - Linear algebra & streaming graph data
   - Maintaining accuracy!

3. Performance and implementation aspects
   - Requests for GraphBLAS implementations
Potential 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 $< 5$ms...
  • Is the transfer by a virus / persistent threat?

• Bioinformatics, health
  • Construct gene sequences, analyze protein interactions, map brain interactions

• Credit fraud forensics $\Rightarrow$ detection $\Rightarrow$ monitoring

Riedy, GABB 2016
Streaming graph data

Networks data rates:

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

Person-level data rates:

• 500M posts per day on Twitter (6k / sec)¹
• 3M posts per minute on Facebook (50k / sec)²

We need to analyze only changes and not entire graph. Throughput & latency trade offs expose different levels of concurrency.

Streaming graph analysis

Terminology:

- **Streaming** changes into a massive, evolving graph
- Not CS streaming algorithm (tiny memory)
- Need to handle *deletions* as well as insertions

Previous *throughput* results (not comprehensive review):

**Data ingest**  \( >2 \text{M up/sec} \) [Ediger, McColl, Poovey, Campbell, & Bader 2014]

**Clustering coefficients**  \( >100 \text{K up/sec} \) [R, Meyerhenke, Bader, Ediger, & Mattson 2012]

**Connected comp.**  \( >1 \text{M up/sec} \) [McColl, Green, & Bader 2013]

**Community clustering**  \( >100 \text{K up/sec}^* \) [R & Bader 2013]
Incremental PageRank
Everyone’s “favorite” metric: PageRank.

- Stationary distribution of the random surfer model.
- Eigenvalue problem can be re-phrased as a linear system\(^3\)

\[
(I - \alpha A^T D^{-1}) x = kv,
\]

with

\[
\begin{align*}
\alpha & \text{ teleportation constant } < 1 \\
A & \text{ adjacency matrix} \\
D & \text{ diagonal matrix of out degrees, with} \\
x/0 &= x \text{ (self-loop)} \\
v & \text{ personalization vector, } \|v\|_1 = 1 \\
k & \text{ scaling constant}
\end{align*}
\]

\[^3\text{Gleich, Zhukov, & Berkhin 2004; Del Corso, Gull, & Romani 2005}\]

Riedy, ÑABB 2016
Incremental PageRank: Goals

- Efficiently update for streaming data; update PageRank without touching the entire graph.
- Keep the results accurate.
  - Updates can wander, and ranks deceive...
- Existing methods:
  - Compute “summaries” of non-changed portions: Walk whole graph per change. [Langville & Meyer, 2006]
  - Maintain databases of walks for dynamic resampling [Bahmani, Chowdhury, & Goel 2010]
  - Statistically based idea, very similar but... [Ohsaka, et al. 2015]
Incremental PageRank: First pass

• Let $A_\Delta = A + \Delta A$, $D_\Delta = D + \Delta D$ for the new graph, want to solve for $x + \Delta x$.
  • $A$: sparse and row-major, $A_{i,j} = 1$ if $i \rightarrow j \in$ edges
• Simple algebra:

\[
(I - \alpha A_\Delta^T D_\Delta^{-1}) \Delta x = \alpha (A_\Delta D_\Delta^{-1} - AD^{-1}) x
\]

• The operator on the right-hand side, $A_\Delta D_\Delta^{-1} - AD^{-1}$, is sparse; non-zero only adjacent to changes in $\Delta$.
• Re-arrange for Jacobi (stationary iterative method),

\[
\Delta x^{(k+1)} = \alpha A_\Delta^T D_\Delta^{-1} \Delta x^{(k)} + \alpha (A_\Delta D_\Delta^{-1} - AD^{-1}) x,
\]
iterate, ...
Incremental PageRank: Accumulating error

- And **fail**. The updated solution wanders away from the true solution. Top *rankings* stay the same...

Riedy, GABB 2016
Incremental PageRank: *Think* instead

- Backward error view: The new problem is close to the old one, solution may be close.
- How close? Residual:

\[
\begin{align*}
 r' &= kv - x + \alpha A_\Delta D_\Delta^{-1} x \\
 &= r + \alpha \left( A_\Delta D_\Delta^{-1} - AD^{-1} \right) x.
\end{align*}
\]

- Solve \((I - \alpha A_\Delta D_\Delta^{-1}) \Delta x = r'\) (iterative refinement).
- Cheat by not refining *all* of \(r'\), only region growing around the changes:

\[
(I - \alpha A_\Delta D_\Delta^{-1}) \Delta x = r'|_\Delta
\]

- (Also cheat by updating \(r\) rather than recomputing at the changes.)
Incremental PageRank: Works

Riedy, GABB 2016
Performance / Latency
Performance discussion

• Inherent trade-off between high throughput and low latency.
  
  **Throughput**  Large batches imply great parallelism  
  **Latency**  Small batches are highly independent

• Restarting PR iteration exposes massive parallelism
  • Sparse matrix - dense vector mult. (SpMV)

• Incremental is highly sparse, pays in overhead
  • Sparse matrix - *sparse* vector mult. (SpMSpV)

• Results are worst-case: changes are not related to conductance communities.
  • (Also, very un-tuned SpMSpV...)
### Test cases

Using one CPU in an 8-core Intel Westmere-EX (E7-4820), 2.00GHz and 18MiB L3...

| Graph           | |V| | |E| | Avg. Degree | Size (MiB) |
|-----------------|---|---|---|---|---------|------------|
| power           | 4941| 6594| 1.33 | 0.07 |
| PGPgiantcompo   | 10680| 24316| 2.28 | 0.23 |
| caidaRouterLevel| 192244| 609066| 3.17 | 5.38 |
| belgium.osm     | 1441295| 1549970| 1.08 | 22.82 |
| coPapersCiteseer| 434102| 16036720| 36.94 | 124.01 |

Riedy, GABB 2016
Incremental PageRank: Worst latency

Incremental: Best at 4 threads. RPR: Best at 8.

Riedy, GABB 2016
Incremental PageRank: Worst throughput

Incremental: Best at 4 threads. RPR: Best at 8.

Riedy, GABB 2016
<table>
<thead>
<tr>
<th>Graph</th>
<th>Batch</th>
<th>dpr</th>
<th>dpr_held</th>
<th>pr_restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>10</td>
<td>.00304</td>
<td>1.8×</td>
<td>.00008</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.0109</td>
<td>.79×</td>
<td>.00707</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>.0126</td>
<td>.67×</td>
<td>.0124</td>
</tr>
<tr>
<td>PGPgiantcompo</td>
<td>10</td>
<td>.00211</td>
<td>4.3×</td>
<td>.00023</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.0257</td>
<td>.81×</td>
<td>.00874</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>.0372</td>
<td>.67×</td>
<td>.0341</td>
</tr>
<tr>
<td>caidaRouterLevel</td>
<td>10</td>
<td>.00710</td>
<td>16×</td>
<td>.00199</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.0314</td>
<td>7.1×</td>
<td>.00477</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1.30</td>
<td>.68×</td>
<td>.2290</td>
</tr>
<tr>
<td>belgium.osm</td>
<td>10</td>
<td>.0118</td>
<td>42×</td>
<td>.0128</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.0127</td>
<td>39×</td>
<td>.0131</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>.0461</td>
<td>52×</td>
<td>.0171</td>
</tr>
<tr>
<td>coPapersCiteseer</td>
<td>10</td>
<td>.0729</td>
<td>27×</td>
<td>.0128</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.8650</td>
<td>2.3×</td>
<td>.130</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>2.97</td>
<td>1.3×</td>
<td>1.13</td>
</tr>
</tbody>
</table>
## Fraction of traversed edges

<table>
<thead>
<tr>
<th>Graph</th>
<th>Batch</th>
<th>dpr</th>
<th>dpr_held</th>
<th>pr_restart</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>power</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>7.54</td>
<td>2.4 ×</td>
<td>0.0229</td>
<td>790 ×</td>
</tr>
<tr>
<td>100</td>
<td>29.2</td>
<td>1.2 ×</td>
<td>18.2</td>
<td>1.9 ×</td>
</tr>
<tr>
<td>1000</td>
<td>38.3</td>
<td>1.0 ×</td>
<td>37.1</td>
<td>1.0 ×</td>
</tr>
<tr>
<td><strong>PGP giantcompo</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.58</td>
<td>5.1 ×</td>
<td>0.0453</td>
<td>180 ×</td>
</tr>
<tr>
<td>100</td>
<td>21.3</td>
<td>1.1 ×</td>
<td>6.82</td>
<td>3.6 ×</td>
</tr>
<tr>
<td>1000</td>
<td>31.2</td>
<td>1.0 ×</td>
<td>28.4</td>
<td>1.1 ×</td>
</tr>
<tr>
<td><strong>caida Router Level</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.0625</td>
<td>32 ×</td>
<td>0.00252</td>
<td>790 ×</td>
</tr>
<tr>
<td>100</td>
<td>0.409</td>
<td>9.8 ×</td>
<td>0.0301</td>
<td>130 ×</td>
</tr>
<tr>
<td>1000</td>
<td>15.2</td>
<td>1.1 ×</td>
<td>3.07</td>
<td>5.2 ×</td>
</tr>
<tr>
<td><strong>belgium.osm</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.00020</td>
<td>9800 ×</td>
<td>0.00007</td>
<td>30000 ×</td>
</tr>
<tr>
<td>100</td>
<td>0.00203</td>
<td>990 ×</td>
<td>0.00066</td>
<td>3000 ×</td>
</tr>
<tr>
<td>1000</td>
<td>0.120</td>
<td>84 ×</td>
<td>0.00660</td>
<td>1500 ×</td>
</tr>
<tr>
<td><strong>coPapers Citeseer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.0563</td>
<td>36 ×</td>
<td>0.00646</td>
<td>310 ×</td>
</tr>
<tr>
<td>100</td>
<td>0.689</td>
<td>2.9 ×</td>
<td>0.0952</td>
<td>21 ×</td>
</tr>
<tr>
<td>1000</td>
<td>2.32</td>
<td>1.7 ×</td>
<td>0.864</td>
<td>4.6 ×</td>
</tr>
</tbody>
</table>
GraphBLAS Requests: Beat Down Overhead!
GraphBLAS background

- Provide a means to express linear-algebra-like graph algorithms
- Graphs can be modeled as (sparse) matrices
  - Adjacency (used here)
  - Vertex-edge
  - Bipartite...
- Some graph algorithms iterate as if applying a linear operator
  - BFS
  - Betweenness centrality
  - PageRank
- Ok, PageRank actually is linear algebra...
Lessons from sparse linear algebra

• Overhead is important.
  • Small average degree
    • $O(|E|)$ is $O(|V|)$...
    • If the average degree is 8, $8|V|$ storage is the matrix.
  • **Graphs**: Some very large degrees
  • **Graphs**: Low average diameter

• Details matter
  • Entries that disappear are painful
  • Take care with definitions on $|V|$...
GraphBLAS in updating PR

dpr_core \((A, z, r, \Delta)\)

Let \(D = \text{diagonal matrix of vertex out-degrees}\)

\(z = \alpha (A^T D^{-1} x_{|\Delta} - z)\)  \hspace{1cm} \text{Note: } z = A^T D^{-1} x \text{ before update}

\(\Delta x^{(1)} = z + r \big|_{z}\)

For \(k = 1 \ldots \text{itmax}\)

\(\Delta x^{(k+1)} = \alpha A^T D^{-1} \Delta x^{(k \geq \gamma)} + \alpha \Delta x^{(k < \gamma)} + z\)

\(\Delta x^{(k+1)} = \Delta x^{(k+1)} + r \big|_{\Delta x^{(k+1)}}\)

Stop if \(\|x^{(k+1)} - x^{(k)}\|_1 < \tau\)

\(\Delta r = z + \Delta x^{(k+1)} - \alpha A^T D^{-1} \Delta x^{(k+1)}\)

Return \(\Delta x^{(k+1)} \) and \(\Delta r\)

Fuse common sparse operation sequences to reduce overhead.

Riedy, GABB 2016
**GraphBLAS in updating PR**

\[ \text{dpr\_core} (A, z, r, \Delta) \]

Let \( D = \text{diagonal matrix of vertex out-degrees} \)

\[ z = \alpha (A^T D^{-1} x |_{\Delta} - z) \quad \text{Note: } z = A^T D^{-1} x \text{ before update} \]

\[ \Delta x^{(1)} = z + r |_z \]

For \( k = 1 \ldots \text{itmax} \)

\[ \Delta x^{(k+1)} = \alpha A^T D^{-1} \Delta x^{(k)}_{\geq \gamma} + \alpha \Delta x^{(k)}_{< \gamma} + z \]

\[ \Delta x^{(k+1)} = \Delta x^{(k+1)} + r |_{\Delta x^{(k+1)}} \]

Stop if \( \| x^{(k+1)} - x^{(k)} \|_1 < \tau \)

\[ \Delta r = z + \Delta x^{(k+1)} - \alpha A^T D^{-1} \Delta x^{(k+1)} \]

Return \( \Delta x^{(k+1)} \) and \( \Delta r \)

Region-growing: Don’t duplicate / realloc only-extended patterns.

Riedy, GABB 2016
GraphBLAS in updating PR

dpr_core \( (A, z, r, \Delta) \)

Let \( D = \text{diagonal matrix of vertex out-degrees} \)

\[
z = \alpha (A^T D^{-1} x_{|\Delta} - z) \quad \text{Note: } z = A^T D^{-1} x \text{ before update}
\]

\[
\Delta x^{(1)} = z + r_{|z}
\]

For \( k = 1 \ldots \text{itmax} \)

\[
\Delta x^{(k+1)} = \alpha A^T D^{-1} \Delta x_{|\geq \gamma}^{(k)} + \alpha \Delta x_{|<\gamma}^{(k)} + z
\]

\[
\Delta x^{(k+1)} = \Delta x^{(k+1)} + r_{|\Delta x^{(k+1)}}
\]

Stop if \( \| x^{(k+1)} - x^{(k)} \|_1 < \tau \)

\[
\Delta r = z + \Delta x^{(k+1)} - \alpha A^T D^{-1} \Delta x^{(k+1)}
\]

Return \( \Delta x^{(k+1)} \) and \( \Delta r \)

Fuse tests into loops when feasible.
GraphBLAS in updating PR

dpr_core (A, z, r, Δ)

Let $D = \text{diagonal matrix of vertex out-degrees}$

$z = \alpha (A^TD^{-1}x_{|\Delta} - z)$ \quad \text{Note: } z = A^TD^{-1}x \text{ before update}$

$\Delta x^{(1)} = z + r|_z$

For $k = 1 \ldots \text{itmax}$

$\Delta x^{(k+1)} = \alpha A^TD^{-1} \Delta x^{(k)}_{\geq \gamma} + \alpha \Delta x^{(k)}_{<\gamma} + z$

$\Delta x^{(k+1)} = \Delta x^{(k+1)} + r|_{\Delta x^{(k+1)}}$

Stop if $\|x^{(k+1)} - x^{(k)}\|_1 < \tau$

$\Delta r = z + \Delta x^{(k+1)} - \alpha A^TD^{-1} \Delta x^{(k+1)}$

Return $\Delta x^{(k+1)}$ and $\Delta r$

Looks like the optimization point? Few vertices in $\Delta$ means all steps matter.
GraphBLAS in updating PR

dpr_core (A, z, r, Δ)

Let $D = \text{diagonal matrix of vertex out-degrees}$

$z = \alpha(A^TD^{-1}x|_{\Delta} - z)$  \quad \text{Note: } z = A^TD^{-1}x \text{ before update}$

$\Delta x^{(1)} = z + r|_{z}$

For $k = 1 \ldots \text{itmax}$

$\Delta x^{(k+1)} = \alpha A^TD^{-1}\Delta x_{\geq \gamma}^{(k)} + \alpha \Delta x_{< \gamma}^{(k)} + z$

$\Delta x^{(k+1)} = \Delta x^{(k+1)} + r|_{\Delta x^{(k+1)}}$

Stop if $\|x^{(k+1)} - x^{(k)}\|_1 < \tau$

$\Delta r = z + \Delta x^{(k+1)} - \alpha A^TD^{-1}\Delta x^{(k+1)}$

Return $\Delta x^{(k+1)}$ and $\Delta r$

Support fast restrictions to a known pattern.
GraphBLAS in updating PR

dpr\_core (A, z, r, \Delta)

Let $D = \text{diagonal matrix of vertex out-degrees}$

$z = \alpha (A^T D^{-1} x |_{\Delta} - z)$ \quad \text{Note: } z = A^T D^{-1} x \text{ before update}$

$\Delta x^{(1)} = z + r|_z$

For $k = 1 \ldots \text{itmax}$

$\Delta x^{(k+1)} = \alpha A^T D^{-1} \Delta x^{(k)}_{|\geq \gamma} + \alpha \Delta x^{(k)}_{|< \gamma} + z$

$\Delta x^{(k+1)} = \Delta x^{(k+1)} + r|_{\Delta x^{(k+1)}}$

Stop if $\| x^{(k+1)} - x^{(k)} \|_1 < \tau$

$\Delta r = z + \Delta x^{(k+1)} - \alpha A^T D^{-1} \Delta x^{(k+1)}$

Return $\Delta x^{(k+1)}$ and $\Delta r$

Keep growth in order, no need to subtract on new entries.
New experience from streaming graphs

For GraphBLAS-ish low-latency streaming algorithms:

- Fuse common sparse operation sequences to reduce overhead.
- Region-growing: Don’t duplicate / realloc only-extended patterns.
- Fuse tests into loops when feasible.
- Remember the sparse vector set case!
- Support fast restrictions to a known pattern.
- Keep growth in order, no need to subtract on new entries.
STINGER: Where do you get it?

Home: www.cc.gatech.edu/stinger/
Code: git.cc.gatech.edu/git/u/eriedy3/stinger.git/

This code: src/clients/algorithms/pagerank_updating/.

Gateway to

- code,
- development,
- documentation,
- presentations...

Remember: Academic code, but maturing with contributions.

Users / contributors / questioners: Georgia Tech, PNNL, CMU, Berkeley, Intel, Cray, NVIDIA, IBM, Federal Government, Ionic Security, Citi...