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

(insert prefix here)-scale data 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!**

Outline

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 $<$ 5ms...
  - 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

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.

1 www.internetlivestats.com/twitter-statistics/

2 www.jeffbullas.com/2015/04/17/21-awesome-facebook-facts-and-statistics-you-need-to-check-out/

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 $>$2M up/sec [Ediger, McColl, Poovey, Campbell, & Bader 2014]

Clustering coefficients $>$100K up/sec [R, Meyerhenke, Bader, Ediger, & Mattson 2012]

Connected comp. $>$1M up/sec [McColl, Green, & Bader 2013]

Community clustering $>$100K up/sec* [R & Bader 2013]

Incremental PageRank

PageRank

Everyone's "favorite" metric: PageRank.

- Stationary distribution of the *random surfer* model.
- Eigenvalue problem can be re-phrased as a linear system[3]

$$\left(I - \alpha A^T D^{-1}\right) x = kv,$$

with

α	*teleportation constant* < 1
A	adjacency matrix
D	diagonal matrix of out degrees, with $x/0 = x$ (self-loop)
v	*personalization* vector, $\|v\|_1 = 1$
k	scaling constant

[3] [Gleich, Zhukov, & Berkhin 2004; Del Corso, Gull, & Romani 2005]

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 $i \to j \in$ edges
- Simple algebra:

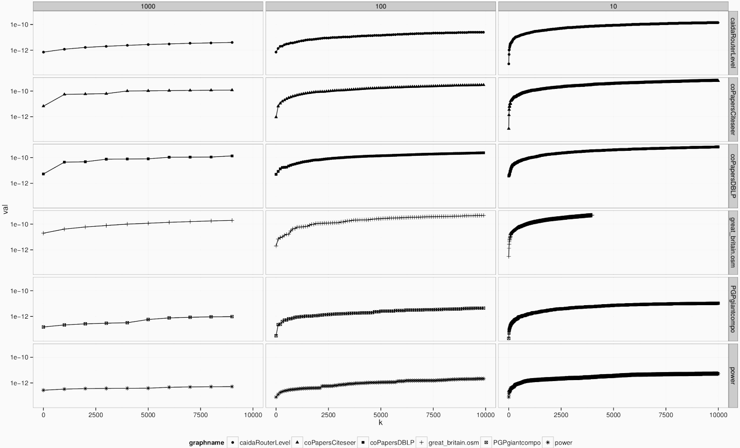
$$\left(I - \alpha A_\Delta^T D_\Delta^{-1}\right) \Delta x = \alpha \left(A_\Delta D_\Delta^{-1} - A D^{-1}\right) x$$

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

$$\Delta x^{(k+1)} = \alpha A_\Delta^T D_\Delta^{-1} \Delta x^{(k)} + \alpha \left(A_\Delta D_\Delta^{-1} - A D^{-1}\right) x,$$

  iterate, ...

Incremental PageRank: Accumulating error



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

Incremental PageRank: *Think* instead

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

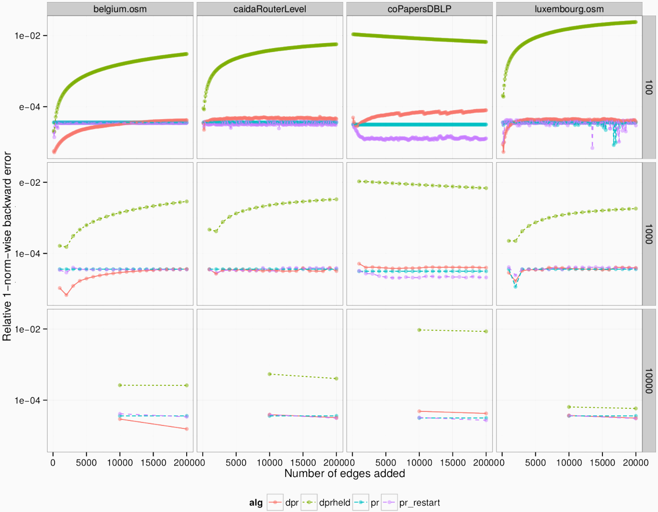
$$\begin{aligned} r' &= kv - x + \alpha A_\Delta D_\Delta^{-1} x \\ &= r + \alpha \left(A_\Delta D_\Delta^{-1} - A D^{-1} \right) x. \end{aligned}$$

- 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



Performance / Latency

Performance discussion

- Inherent trade-off between high throughput and low latency.

  Throughput Large batches imply great parallelism

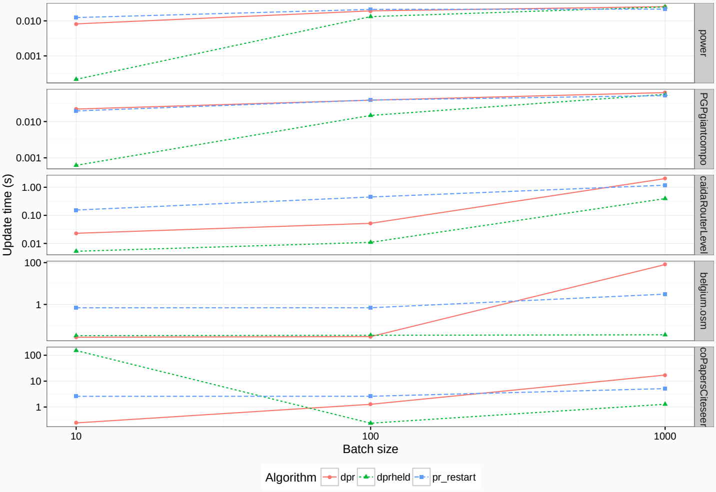
  Latency Small batches are highly independent
- Restarting PR iteration exposes massive parallelism
  - Sparse matrix - dense vector mult. (SpMV)
- Incremental is highly sparse, pays in overhead
  - Sparse matrix - sparse vector mult. (SpMSpV)
- Results are worst-case: changes are not related to conductance 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...

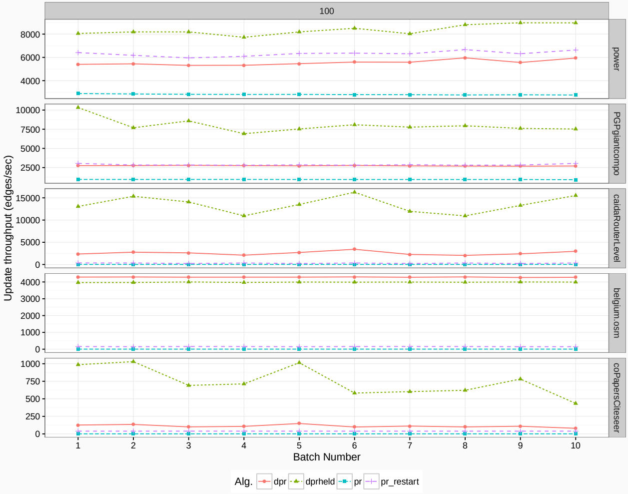
Graph	$|V|$	$|E|$	Avg. Degree	Size (MiB)
power *power grid*	4941	6594	1.33	0.07
PGPgiantcompo *social network*	10680	24316	2.28	0.23
caidaRouterLevel *networking*	192244	609066	3.17	5.38
belgium.osm *road map*	1441295	1549970	1.08	22.82
coPapersCiteseer *citation*	434102	16036720	36.94	124.01

Incremental PageRank: Worst latency



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

Incremental PageRank: Worst throughput



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

Median update time (s)

Graph	Batch	dpr		dpr_held		pr_restart
power	10	.00304	1.8×	.00008	68×	.00535
	100	.0109	.79×	.00707	1.2×	.00860
	1000	.0126	.67×	.0124	.68×	.00843
PGPgiantcompo	10	.00211	4.3×	.00023	39×	.00916
	100	.0257	.81×	.00874	2.4×	.0207
	1000	.0372	.67×	.0341	.73×	.0249
caidaRouterLevel	10	.00710	16×	.00199	57×	.112
	100	.0314	7.1×	.00477	47×	.224
	1000	1.30	.68×	.2290	3.9×	.889
belgium.osm	10	.0118	42×	.0128	39×	.498
	100	.0127	39×	.0131	38×	.499
	1000	.0461	52×	.0171	140×	2.41
coPapersCiteseer	10	.0729	27×	.0128	155×	1.98
	100	.8650	2.3×	.130	15×	1.98
	1000	2.97	1.3×	1.13	3.5×	3.95

Fraction of traversed edges

Graph	Batch	dpr		dpr_held		pr_restart
power	10	7.54	*2.4*×	0.0229	*790*×	18
	100	29.2	*1.2*×	18.2	*1.9*×	34
	1000	38.3	*1.0*×	37.1	*1.0*×	39
PGPgiantcompo	10	1.58	*5.1*×	0.0453	*180*×	8
	100	21.3	*1.1*×	6.82	*3.6*×	24
	1000	31.2	*1.0*×	28.4	*1.1*×	32
caidaRouterLevel	10	0.0625	*32*×	0.00252	*790*×	2
	100	0.409	*9.8*×	0.0301	*130*×	4
	1000	15.2	*1.1*×	3.07	*5.2*×	16
belgium.osm	10	0.00020	*9800*×	0.00007	*30000*×	2
	100	0.00203	*990*×	0.00066	*3000*×	2
	1000	0.120	*84*×	0.00660	*1500*×	10
coPapersCiteseer	10	0.0563	*36*×	0.00646	*310*×	2
	100	0.689	*2.9*×	0.0952	*21*×	2
	1000	2.32	*1.7*×	0.864	*4.6*×	4

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, Δ)

Let $D =$ diagonal matrix of vertex out-degrees

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

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

For $k = 1 \ldots$ `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 Δr

Fuse common sparse operation sequences to reduce overhead.

GraphBLAS in updating PR

`dpr_core` (A, z, r, Δ)

Let $D =$ diagonal matrix of vertex out-degrees

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

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

For $k = 1 \ldots$ `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 Δr

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

GraphBLAS in updating PR

`dpr_core` (A, z, r, Δ)

Let $D =$ diagonal matrix of vertex out-degrees

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

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

For $k = 1 \ldots$ `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 Δr

Fuse tests into loops when feasible.

GraphBLAS in updating PR

`dpr_core` (A, z, r, Δ)

Let $D =$ diagonal matrix of vertex out-degrees

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

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

For $k = 1 \ldots$ `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 Δr

Looks like the optimization point? Few vertices in Δ means *all* steps matter.

GraphBLAS in updating PR

`dpr_core` (A, z, r, Δ)

Let D = diagonal matrix of vertex out-degrees

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

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

For $k = 1 \ldots$ `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 Δr

Support fast restrictions to a known pattern.

GraphBLAS in updating PR

`dpr_core` (A, z, r, Δ)

Let $D =$ diagonal matrix of vertex out-degrees

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

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

For $k = 1 \ldots$ `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 Δr

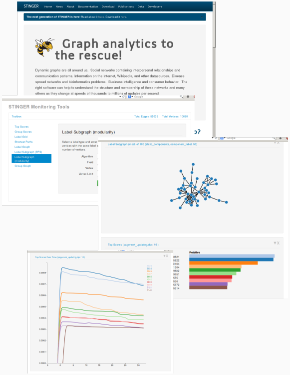
Keep growth in order, no need to subtract on new entries.

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