FAST-PPR: Personalized PageRank Estimation for Large Graphs

Peter Lofgren (Stanford)
Joint work with Siddhartha Banerjee (Stanford),
Ashish Goel (Stanford), and C. Seshadhri (Sandia)
Motivation: Personalized Search
Motivation: Personalized Search

Re-ranked by PPR

<table>
<thead>
<tr>
<th>PPR</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.75236E-4</td>
<td>Adam Messinger</td>
<td>CTO @twitter</td>
</tr>
<tr>
<td>3.65838E-4</td>
<td>Adam D'Angelo</td>
<td>CEO of Quora</td>
</tr>
<tr>
<td>2.2774E-5</td>
<td>Adam Satariano</td>
<td>Technology Reporter, Bloomberg News</td>
</tr>
<tr>
<td>1.907E-5</td>
<td>Adam Steltzner</td>
<td>Rocket scientist, intermittent gardener,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>master of mars, and dangerous dinner guest.</td>
</tr>
<tr>
<td>8.789E-6</td>
<td>Adam Rugel</td>
<td>Hello</td>
</tr>
<tr>
<td>8.342E-6</td>
<td>AdamSerwer</td>
<td>Reporter. @msnbc. I like cats and nerd</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stuff. I also fight crime. Mostly loitering.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="mailto:adam.serwer@nbcuni.com">adam.serwer@nbcuni.com</a></td>
</tr>
</tbody>
</table>
Result Preview

Running Time on Twitter-2010

- Fast-PPR: 2 sec
- Monte-Carlo: 6 min
- Local-Update: 1.2 hour
Personalized PageRank

Given source $s$, target $t$, and ‘teleport probability’ $\alpha$
- Start a random walk from $s$.
- At each step, stop with probability $\alpha$, else continue.

Then Personalized PageRank from $s$ to $t$ is given by:

$$\pi_s(t) = \mathbb{P} \left[ \text{Walk from } s \text{ stops at } t \right]$$

- Equivalent to eigenvector/stationary distribution definitions
- FAST-PPR allows arbitrary starting set, e.g.
  random $s \in V \implies$ Global PageRank
  random $s \in S \implies$ personalize to $S$
Goal
Given $\alpha$, start node $s$, single target node $t$, threshold $\delta$
estimate

$$\pi_s(t)$$

when $\pi_s(t) > \delta$

• Natural primitive for personalized search
• Want only $\pi_s(t)$, not entire $\pi_s$ vector.
• Since average of $\pi_s$ is $\frac{1}{n}$, we want $\delta \sim \frac{1}{n}$. For realtime, running time must be be $\ll \frac{1}{\delta}$
Previous Algorithm: Monte-Carlo

[Avrachenkov, et al 2007]

Sample $O\left(\frac{1}{\delta}\right)$ random walks from $s$, and return estimate

$$\hat{\pi}(s, t) = \text{Fraction of walks ending at } t.$$ 

Running time:

$$\Theta\left(\frac{1}{\delta}\right)$$
Previous Algorithm: Local Update

- Works from target $t$ backwards along edges, updating Personalized PageRank estimates locally.

- Tight average running time $O\left(\frac{\bar{d}}{\delta}\right)$ where $\bar{d} = \frac{|E|}{|V|}$
Main Result

Theorem 1. Given $s$, $t$, and $\delta$, if $\pi_s(t) > \delta$ then estimate $\hat{\pi}_s(t)$ satisfies $|\pi_s(t) - \hat{\pi}_s(t)| \leq 0.1\pi_s(t)$ with probability 0.9. Furthermore, the average running time is

$$O \left( \frac{1}{\sqrt{\delta}} \sqrt{\bar{d}} \right)$$

where $\bar{d}$ is the average in-degree of the graph.

We also prove a lower bound of $\Omega \left( \frac{1}{\sqrt{\delta}} \right)$. 
Analogy: Bidirectional Search

\[ d^l \]

\[ 2d^{l/2} = 2\sqrt{d^l} \]
Bidirectional PageRank Algorithm

Forward Work (Random Walks)
Reverse Work (Frontier Discovery)
Main Idea

Decomposition

$$\pi(s, t) = \sum_{u \in F_t} \Pr \left[ \text{Walk from } s \text{ first hits } u \in F_t \right] \pi(u, t)$$
Experimental Setup

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th># Nodes</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP-2011</td>
<td>undirected</td>
<td>1.0M</td>
<td>6.7M</td>
</tr>
<tr>
<td>Pokec</td>
<td>directed</td>
<td>1.6M</td>
<td>30.6M</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>undirected</td>
<td>4.8M</td>
<td>69M</td>
</tr>
<tr>
<td>Orkut</td>
<td>undirected</td>
<td>3.1M</td>
<td>117M</td>
</tr>
<tr>
<td>Twitter-2010</td>
<td>directed</td>
<td>42M</td>
<td>1.5B</td>
</tr>
<tr>
<td>UK-2007-05</td>
<td>directed</td>
<td>106M</td>
<td>3.7B</td>
</tr>
</tbody>
</table>

- All three algorithms can trade-off accuracy and run-time. Use parameters with similar empirical relative error (10%), then compare running time.
Empirical Running Time

Running Time (Targets sampled by PageRank)

- FAST-PPR
- Local Update
- Monte-Carlo

Time per Query (ms)

Log Scale

Data for Dblp, Pokec, LJ, Orkut, Twitter, UK-Web
Summary

• Fast-PPR estimates $\pi_s(t)$ 10x faster.

• Provable $O \left( \sqrt{\frac{\bar{d}}{\delta}} \right)$ average time; previous best $\Omega \left( \frac{1}{\delta} \right)$

• $O \left( \frac{\bar{d}}{\sqrt{\delta}} \right)$ storage per node $\Rightarrow O \left( \frac{1}{\sqrt{\delta}} \right)$ worst-case time

Future Work

• Close the gap between running time and lower bound.

• Build a scalable personalized search system (in progress).
Thank You

• Paper available on Arxiv
• Code available at cs.stanford.edu/~plofgren
Frontier is Important

Frontier Aided Significance Thresholding
Algorithm (Simple Version)

1. Use Local Update to compute estimates $\hat{\pi}(v, t)$ to accuracy $O(\sqrt{\delta})$.

2. Define

Target Set $\hat{T}_t = \{v \in V : \hat{\pi}(v, t) > \sqrt{\delta}\}$

Frontier $\hat{F}_t = \{u \in V \setminus \hat{T}_t : (u, v) \in E \text{ for some } v \in \hat{T}_t\}$
Algorithm (Simple Version)

3. Take $O\left(\frac{1}{\sqrt{\delta}}\right)$ Random Walks $\{W_i\}$, terminating each early if it hits $\widehat{F}_t$. Define

$$X_i = \begin{cases} \hat{\pi}(u, t), & W_i \text{ hits } u \in \widehat{F}_t \\ 0, & W_i \text{ does not hit } \widehat{F}_t \end{cases}$$

4. Return empirical mean$\{X_i\}$. 
Average Running Time
For a uniformly random target node $t$, the average per-query running time is

$$O \left( \frac{1}{\sqrt{\delta}} \left( \sqrt{\bar{d}} + 1 \right) \right).$$

Improved Implementation:

$$O \left( \frac{1}{\sqrt{\delta}} \sqrt{\bar{d}} \right).$$
Correctness

Now assume imperfect Local Update estimates: \( \forall u \in V, \)

\[ |\hat{\pi}(u, t) - \pi(u, t)| < \beta \sqrt{\delta} \]

where \( \beta \) is a constant like \( \frac{1}{6} \).

Problem: if \( \pi(u, t) < \beta \sqrt{\delta} \) for all \( u \in F_t \), we might have \( \hat{\pi}(u, t) = 0 \).

However, for \( u \in T_t, \pi(u, t) > \sqrt{\delta} \) so

\[ |\hat{\pi}(u, t) - \pi(u, t)| < \frac{\beta}{1 - \beta} \pi(u, t) \]

so we only want to use \( \hat{\pi}(u, t) \) for \( u \in T_t \).
Algorithm (Theoretical Version)

$$\Pr[\text{end at } t \text{ from } u] = \sum_{v \in T_t} \Pr[\text{transition } u \rightarrow v] \pi(v, t)$$

$$+ \Pr[\text{not enter } T_t \text{ from } u] \cdot \Pr[\text{end at } t \text{ from } u | \text{not immediately entering } T_t]$$

Use $\hat{\pi}(v, t)$

Use Monte-Carlo
Algorithm (Theoretical Version)

1. Run Local Update and compute Frontier as before.

2. Take $O\left(\frac{\log(n)}{\sqrt{\delta}}\right)$ walks $\{W_i\}$ from $u$.

3. If $W_i$ enters the frontier at $u \in F_t$,
   
   (a) Accumulate a score based on neighbors of $u$ in $T_t$.
   
   (b) Choose a random out-neighbor of $u$ not in $T_t$ continue the walk from there.
   
   (c) Multiply by the probability of not entering $T_t$ from $u$, to de-bias the estimate.

4. Return the empirical average of accumulated scores.
Local Update Algorithm

\[
\Pr[\text{end at } t \text{ from } u] = \sum_{v \in \text{Out}(u)} \Pr[\text{transition } u \rightarrow v] \Pr[\text{end at } t \text{ from } v]
\]

\[
\pi(u, v) = \sum_{v \in \text{Out}(u)} \frac{1 - \alpha}{d^{\text{out}}(u)} \pi(v, t)
\]
Local Update Algorithm

After 0 iterations

After 1 iteration

p=0.00  s=0.00
x

p=0.00  s=0.00
y

p=0.00  s=0.00
z

p=0.00  s=0.00
v

p=0.00  s=0.00
z

p=0.00  s=0.00
v

p=0.20  s=0.20

p=0.16  s=0.16

p=0.00  s=0.20
Local Update Algorithm

After 2 iterations:

- Node x: p=0.06, s=0.06
- Node y: p=0.13, s=0.13
- Node z: p=0.00, s=0.16
- Node v: p=0.13, s=0.33

After 3 iterations:

- Node x: p=0.06, s=0.06
- Node y: p=0.13, s=0.13
- Node z: p=0.10, s=0.26
- Node v: p=0.00, s=0.33