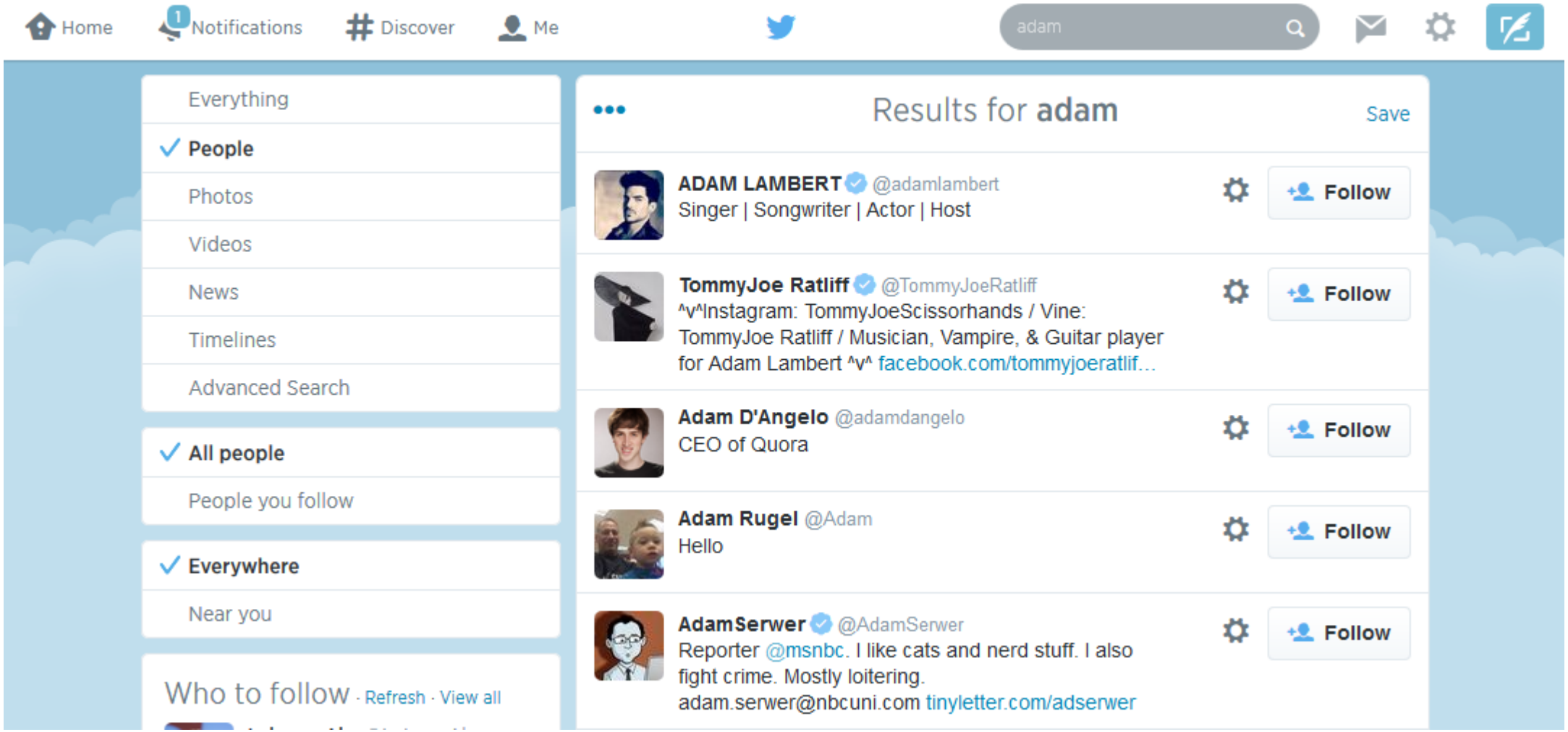
FAST-PPR: Personalized PageRank Estimation for Large Graphs

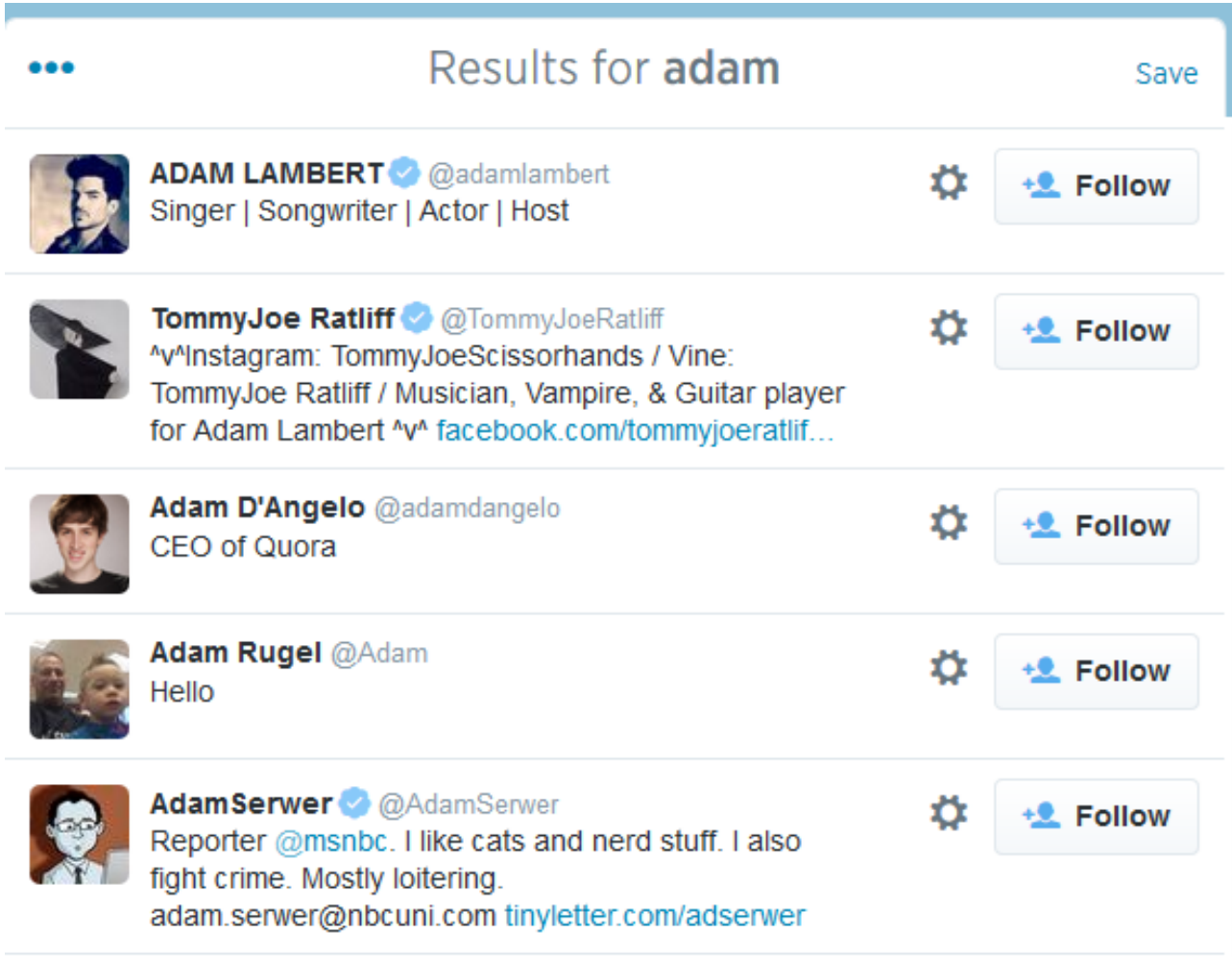
Peter Lofgren (Stanford)

Joint work with Siddhartha Banerjee (Stanford), Ashish Goel (Stanford), and C. Seshadhri (Sandia)

Motivation: Personalized Search



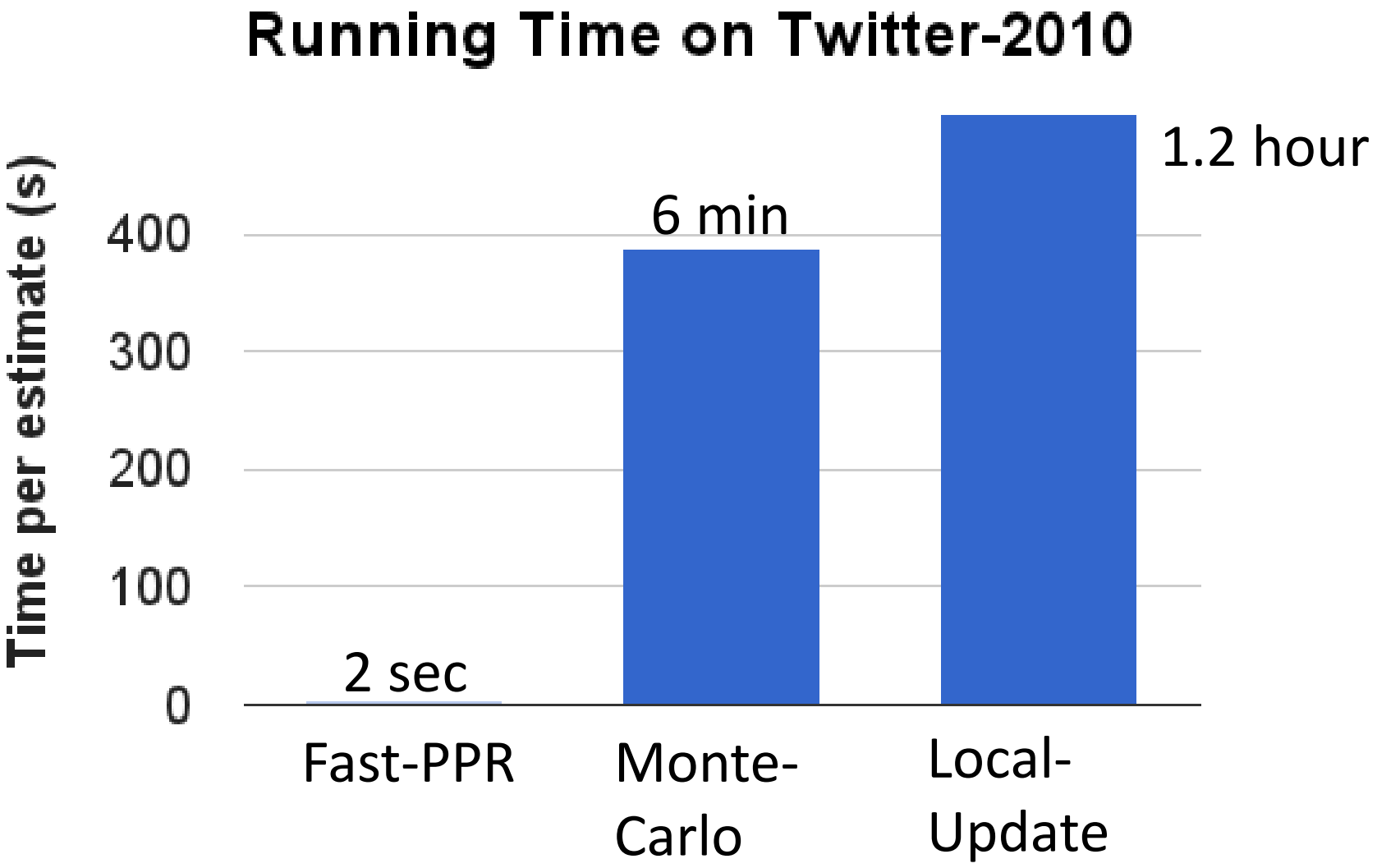
Motivation: Personalized Search



Re-ranked by PPR

PPR	Name	Description
6.75236E-4	Adam Messinger	CTO @twitter
3.65838E-4	Adam D'Angelo (webpage)	CEO of Quora
2.2774E-5	Adam Satariano (webpage)	Technology Reporter, Bloomberg News satariano.adam[at]http://t.co/g3neLm2J
1.907E-5	Adam Steltzner	Rocket scientist, intermittent gardener, master of mars, and dangerous dinner guest. Co-founder of Adam and Trisha's dog and baby farm.
8.789E-6	Adam Rugel (webpage)	Hello
8.342E-6	AdamSerwer (webpage)	Reporter @msnbc. I like cats and nerd stuff. I also fight crime. Mostly loitering. adam.serwer@nbcuni.com https://t.co /WkKdyjSHcP

Result Preview



Personalized PageRank

Given source s, target t, and 'teleport probability' α

- Start a random walk from s.
- At each step, stop with probability α, 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 α, start node s, single target node t, threshold δ estimate

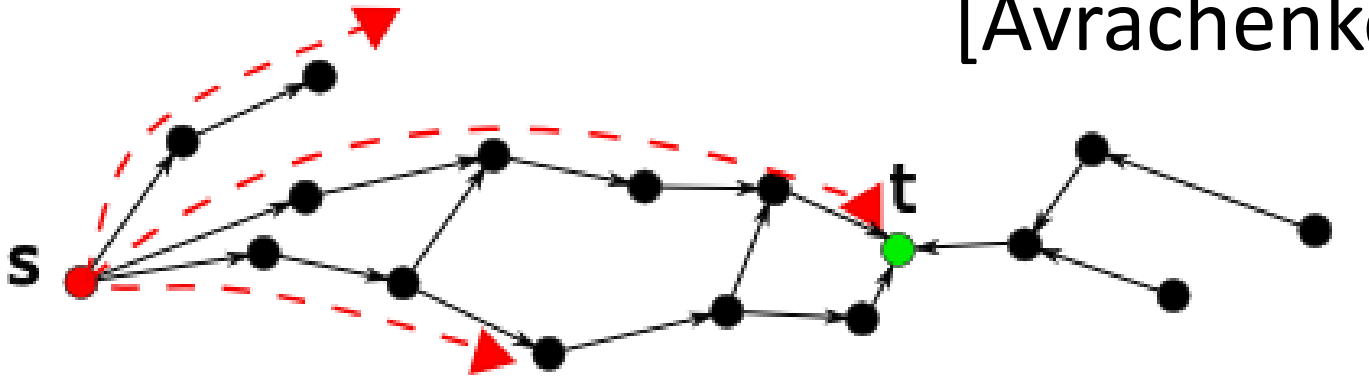
$$\pi_s(t)$$

when $\pi_s(t) > \delta$

- Natural primitive for personalized search
- Want only $\pi_s(t)$, not entire π_s vector.
- Since average of π_s is $\frac{1}{n}$, we want $\delta \sim \frac{1}{n}$. For real-time, 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

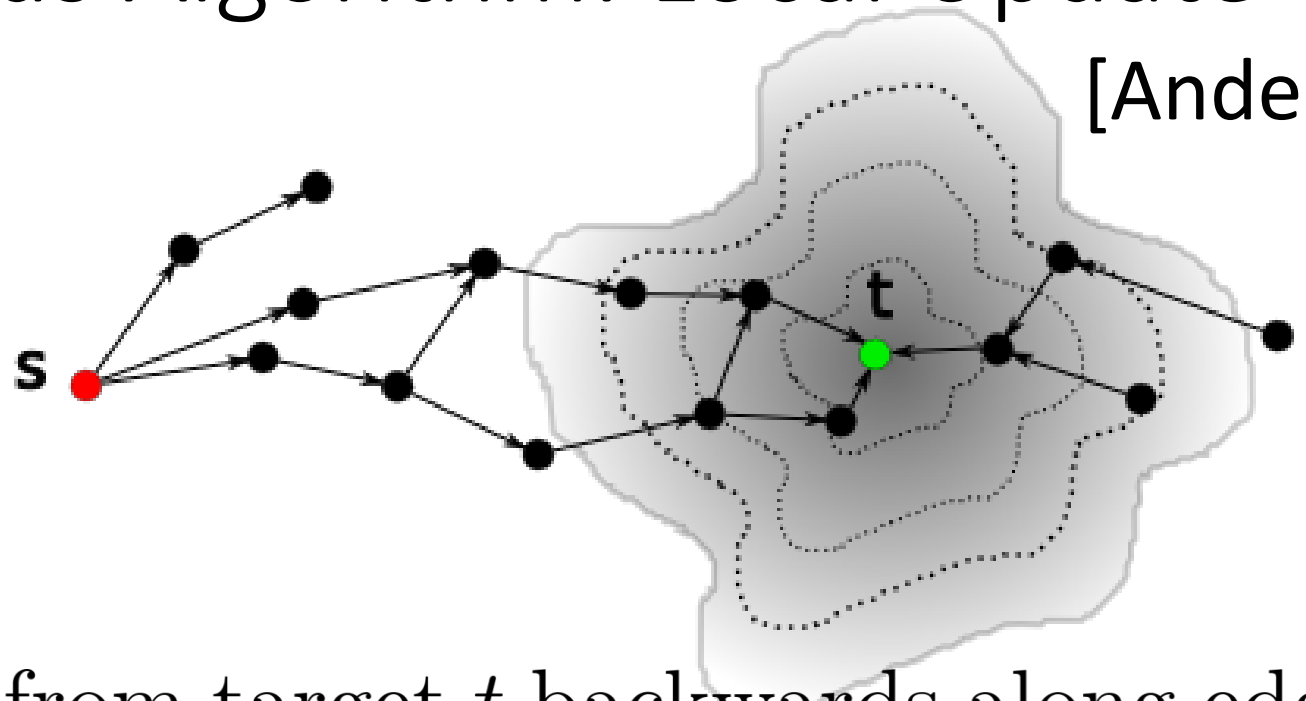
$$\hat{\pi}(s,t) = \text{Fraction of walks ending at } t.$$

Running time:

$$\Theta\left(\frac{1}{\delta}\right)$$

Previous Algorithm: Local Update

[Anderson, et al 2007]



- 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

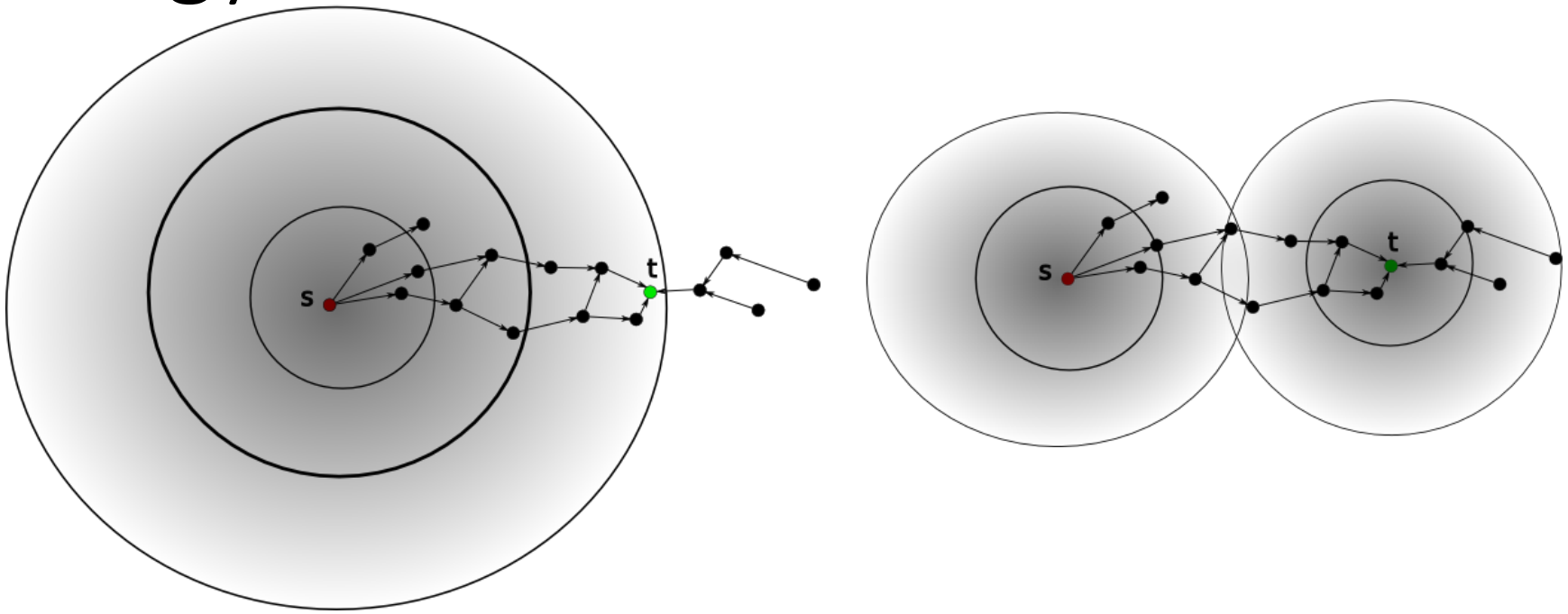
Theorem 1. *Given s, t, and δ, if $\pi_s(t) > \delta$ then estimate $\widehat{\pi}_s(t)$ satisfies $|\pi_s(t) - \widehat{\pi}_s(t)| \leq 0.1\pi_s(t)$ with probability 0.9. Furthermore, the average running time is*

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

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

We also prove a lowerbound of $\Omega\left(\frac{1}{\sqrt{\delta}}\right)$.

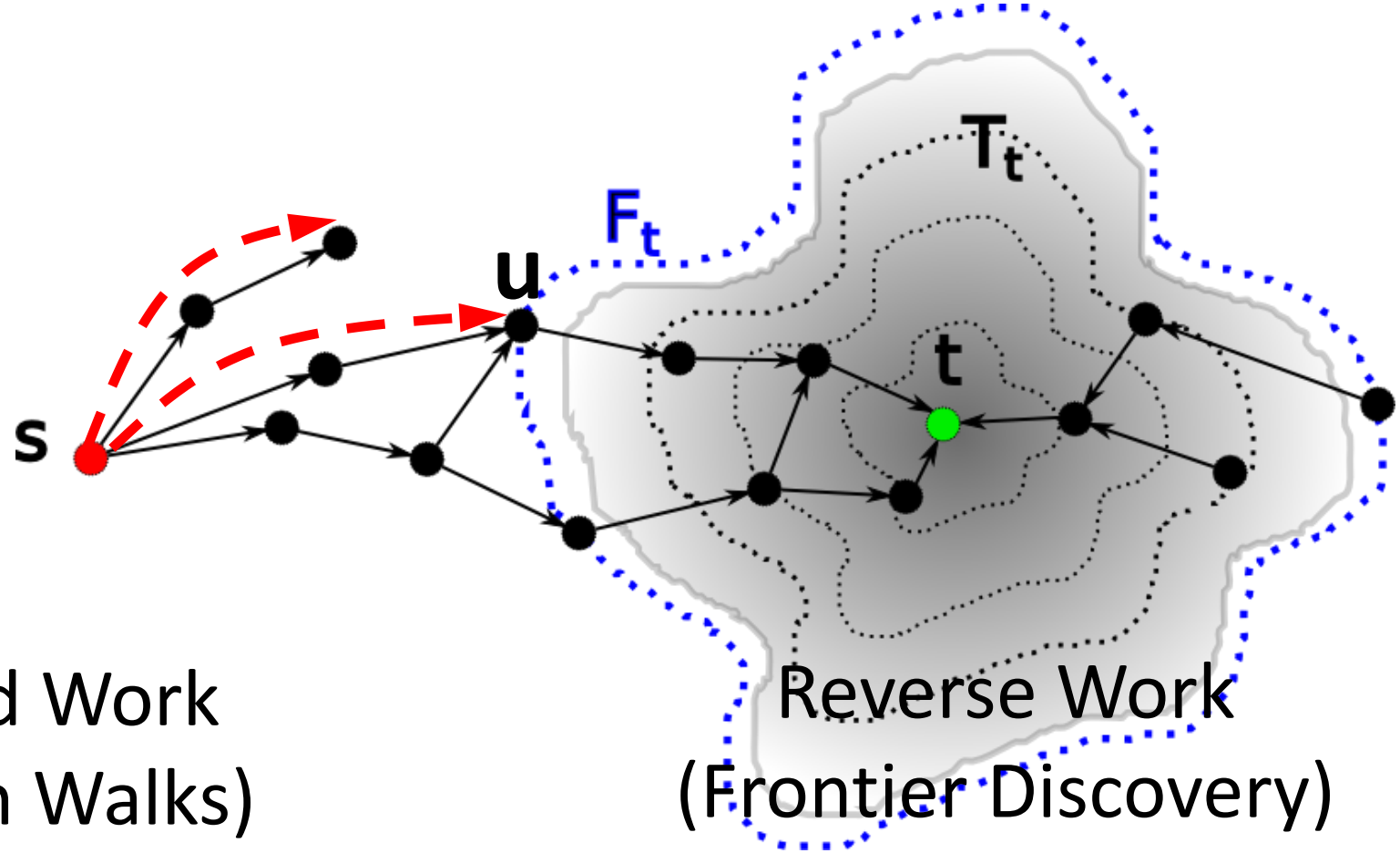
Analogy: Bidirectional Search



$$d^l$$

$$2d^{l/2} = 2\sqrt{d^l}$$

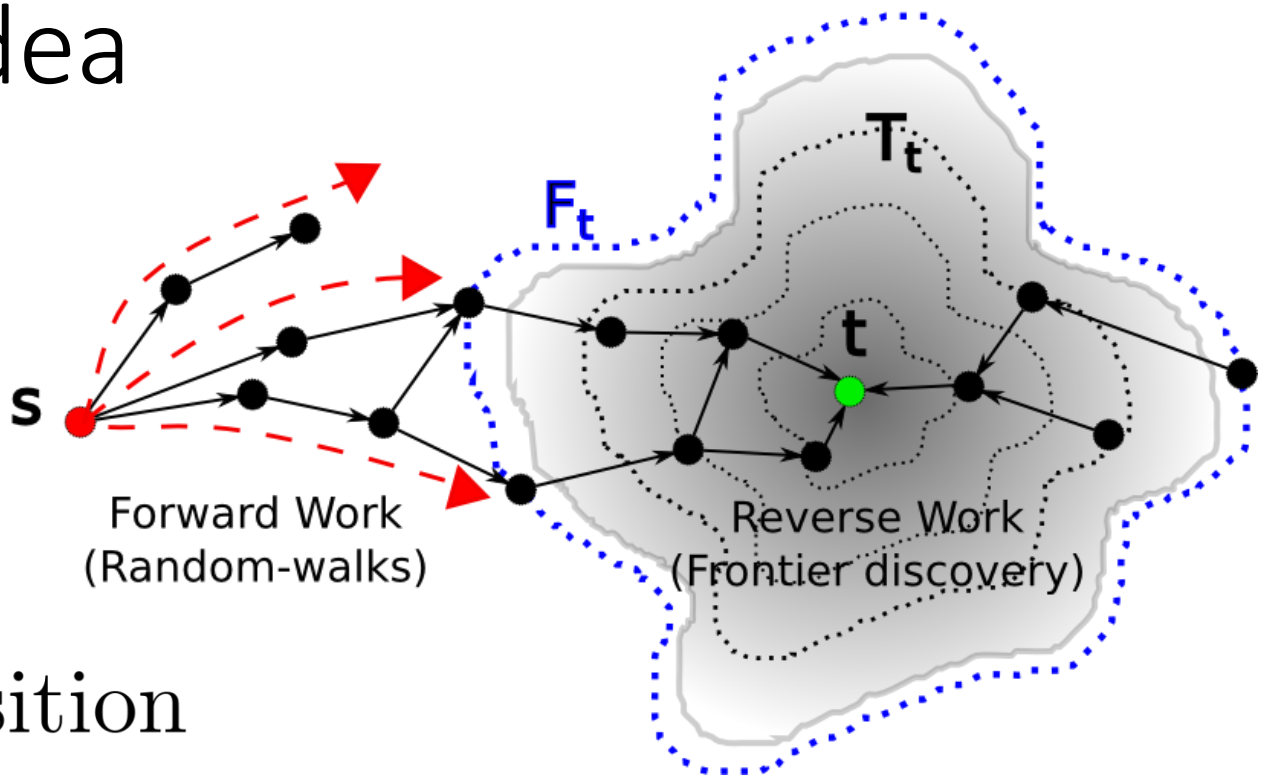
Bidirectional PageRank Algorithm



Forward Work
(Random Walks)

Reverse Work
(Frontier Discovery)

Main Idea



Decomposition

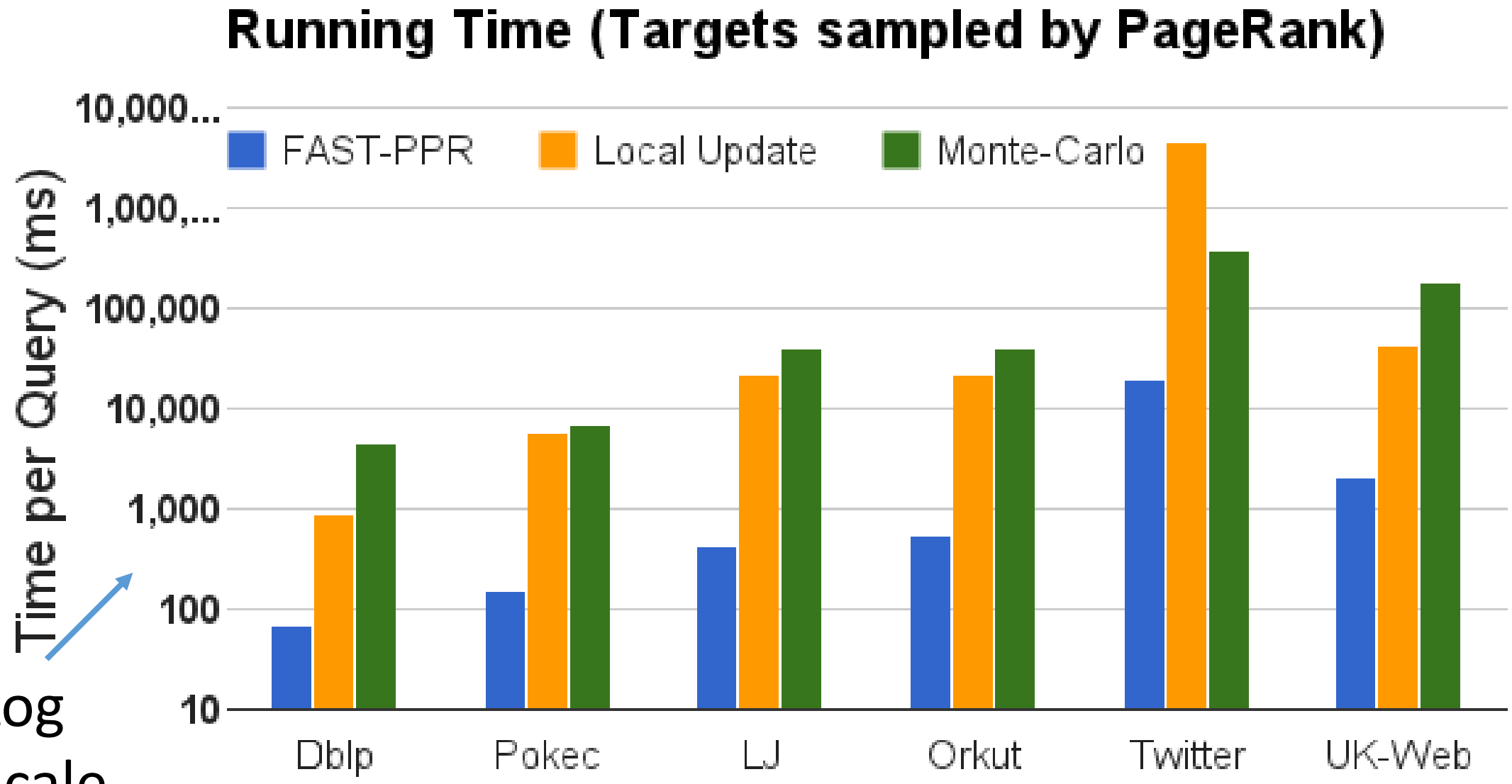
$$\pi(s,t) = \sum_{u \in F_t} \Pr\left[\text{Walk from } s \text{ first hits } u \in F_t\right] \pi(u,t)$$

Experimental Setup

Dataset	Type	# Nodes	# Edges
DBLP-2011	undirected	1.0M	6.7M
Pokec	directed	1.6M	30.6M
LiveJournal	undirected	4.8M	69M
Orkut	undirected	3.1M	117M
Twitter-2010	directed	42M	1.5B
UK-2007-05	directed	106M	3.7B

- 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



Log Scale

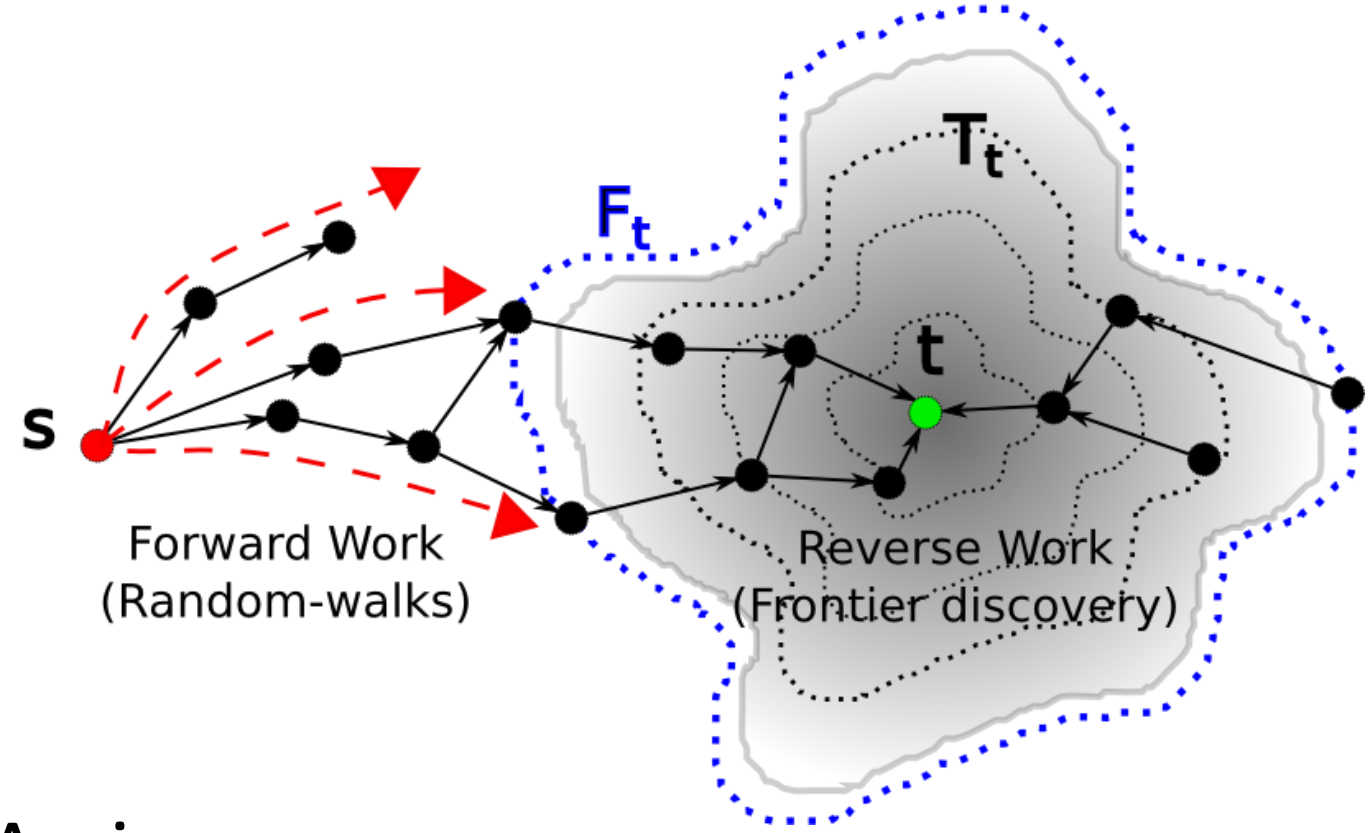
Summary

- 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

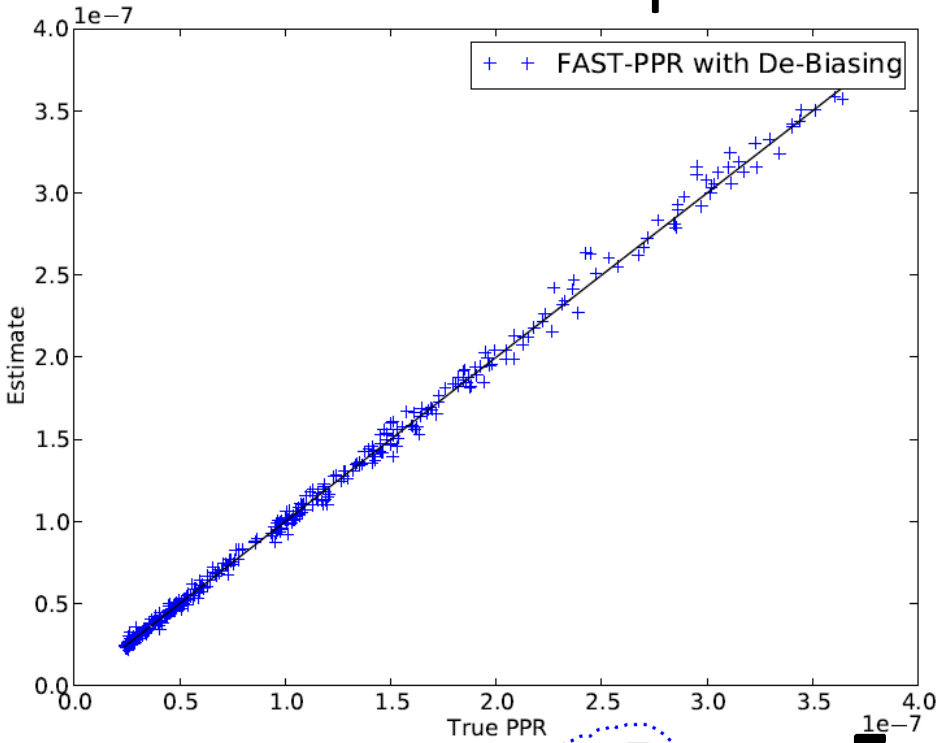
- Close the gap between running time and lower bound.
- Build a scalable personalized search system (in progress).

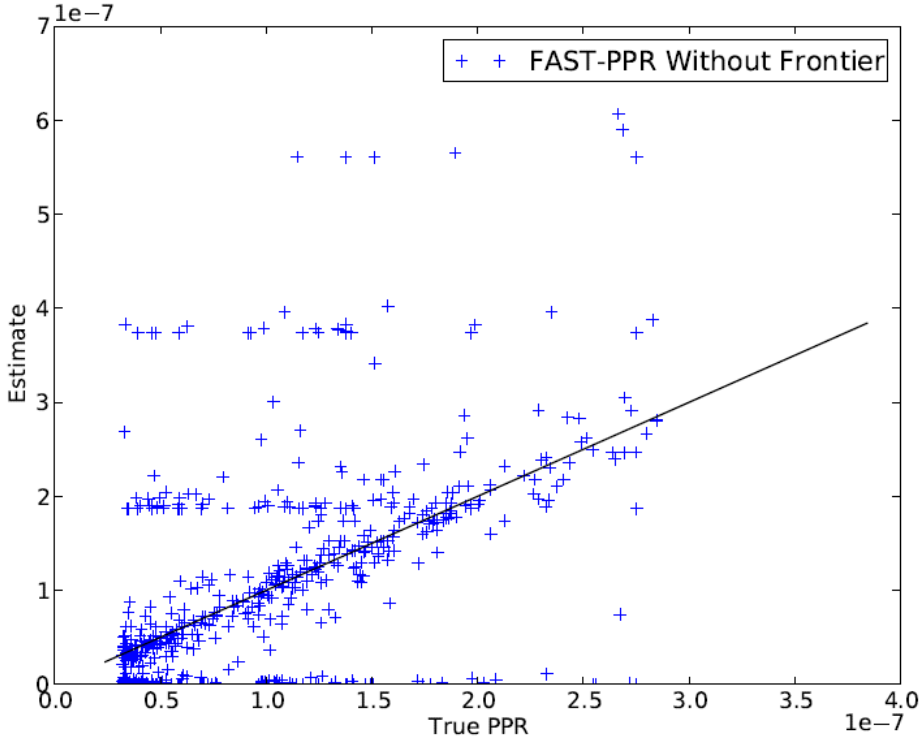
Thank You

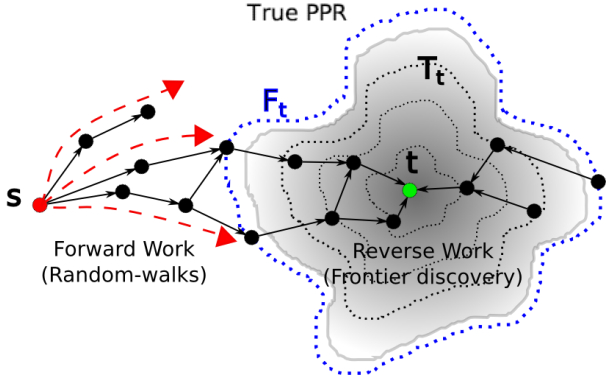


- Paper available on Arxiv
- Code available at cs.stanford.edu/~plofgren

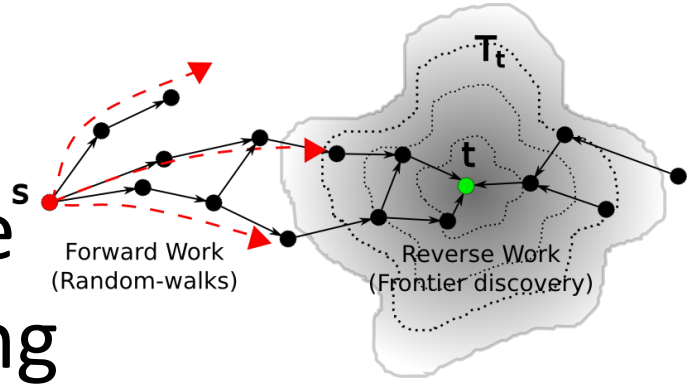
Frontier is Important







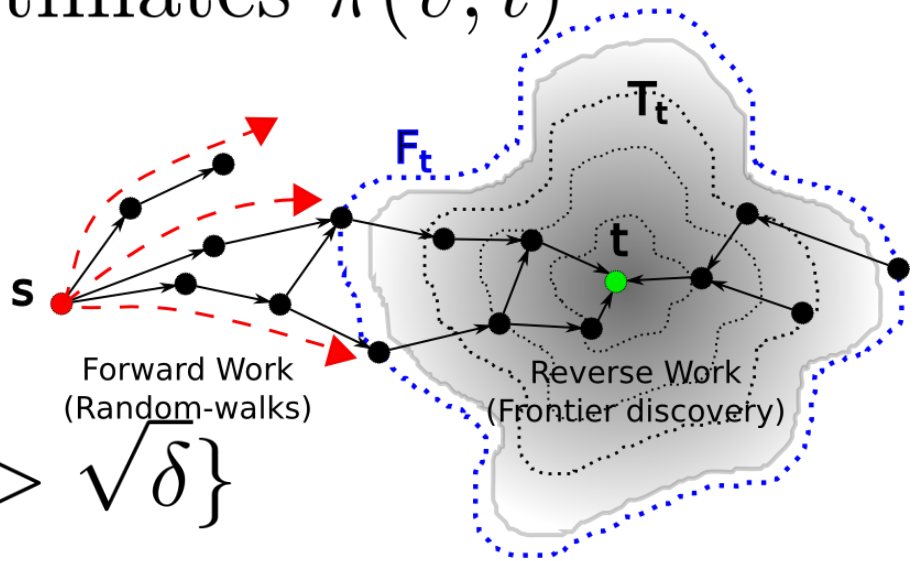
Frontier
Aided
Significance
Thresholding



Algorithm (Simple Version)

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

2. Define



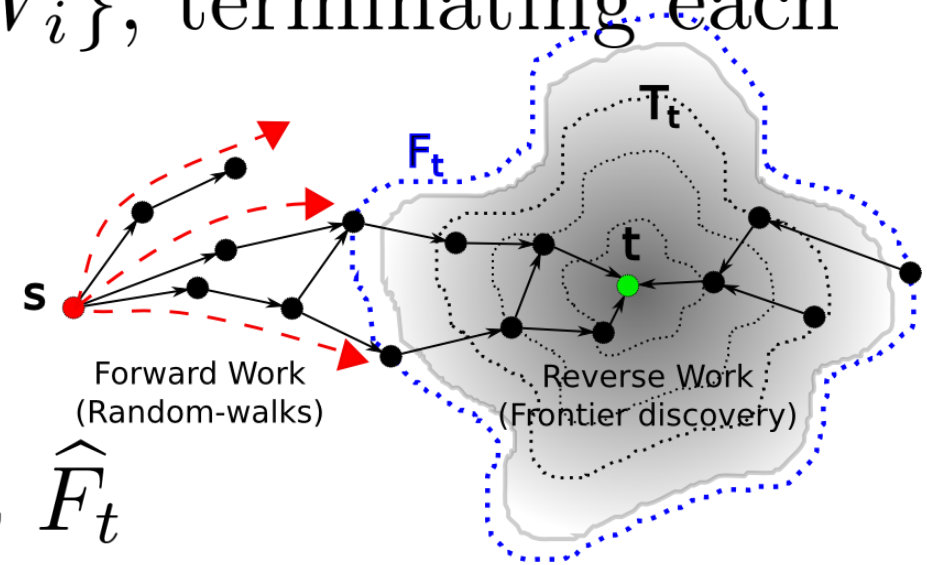
$$\text{Target Set } \widehat{T}_t = \{v \in V : \hat{\pi}(v,t) > \sqrt{\delta}\}$$

$$\text{Frontier } \widehat{F}_t = \{u \in V \backslash \widehat{T}_t : (u,v) \in E \text{ for some } v \in \widehat{T}_t\}$$

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

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

Reverse Work (Local Update)

Forward Work (Monte-Carlo)

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 β 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)

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

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

$$+ \Pr[\text{not enter } T_t \text{ from } u]$$

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

Use Monte-Carlo

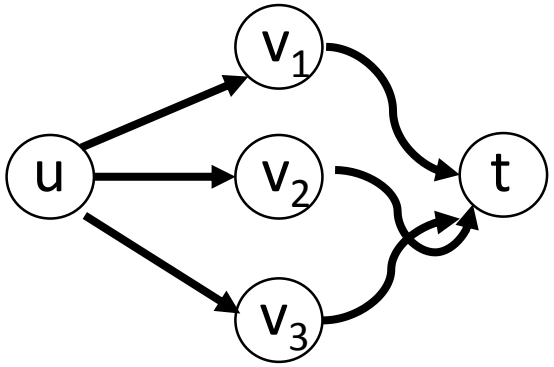
Algorithm (Theoretical Version)

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 nieghbors 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 emperical average of accumulated scores.

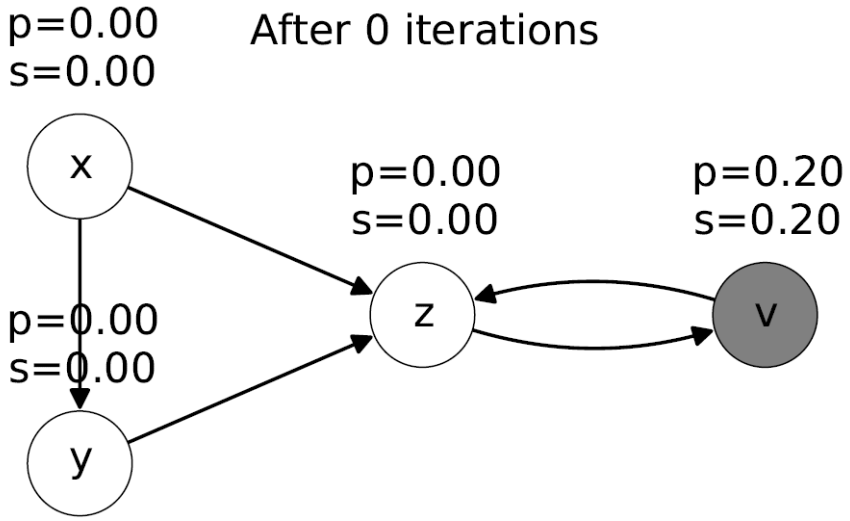
Local Update Algorithm

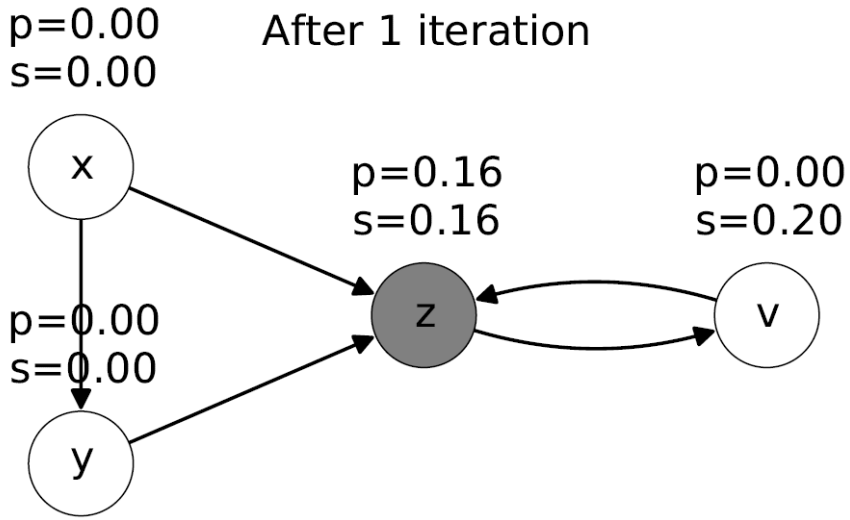
$$\Pr[\text{end at } t \text{ from } u\,] = \sum_{v \in \mathrm{Out}(u)} \Pr[\text{transition } u \to v] \Pr[\text{end at } t \text{ from } v]$$

$$\pi(u,v) = \sum_{v \in \mathrm{Out}(u)} \frac{1-\alpha}{d^{\mathrm{out}}(u)} \pi(v,t)$$



Local Update Algorithm





Local Update Algorithm