FrogWild!

Fast PageRank Approximations on Graph Engines

Ioannis Mitliagkas
Michael Borokhovich
Alex Dimakis
Constantine Caramanis
Web Ranking

Given web graph
Find “important” pages
Web Ranking

Given web graph
Find “important” pages
Rank Based on In-degree
Classic Approach
Web Ranking

Given web graph
Find “important” pages

Rank Based on In-degree
Classic Approach

Susceptible
to manipulation by spammer networks
PageRank [Page et al., 1999]

Page Importance
Described by distribution $\pi$
PageRank \cite{Page et al., 1999}

Page Importance
Described by distribution $\pi$

Recursive Definition
Important pages are pointed to by
\begin{itemize}
  \item important pages are pointed to by
  \item important pages are pointed to by…
\end{itemize}
PageRank \cite{Page et al., 1999}

Page Importance
Described by distribution $\pi$

Recursive Definition
Important pages are pointed to by
\begin{itemize}
  \item important pages are pointed to by
  \item important pages are pointed to by…
\end{itemize}

Robust
to manipulation by spammer networks
PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly
PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly

Every Iteration
Each vertex spreads water evenly to successors
PageRank - Continuous Interpretation

**Start:** Gallon of *water* distributed evenly

**Every Iteration**
Each vertex spreads water evenly to successors
PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly

Every Iteration
Each vertex spreads water evenly to successors

Redistribute evenly a fraction, $p_T = 0.15$, of all water
PageRank - Continuous Interpretation

Start: Gallon of water distributed evenly

Every Iteration
Each vertex spreads water evenly to successors

Redistribute evenly
a fraction, \( p_T = 0.15 \), of all water

Repeat until convergence

Power Iteration employed usually
Discrete Interpretation

Frog walks randomly on graph
Next vertex chosen uniformly at random
Discrete Interpretation

Frog walks randomly on graph
Next vertex chosen uniformly at random
Frog walks randomly on graph
Next vertex chosen uniformly at random
Discrete Interpretation

Frog walks randomly on graph
Next vertex chosen uniformly at random

Teleportation
Every step: teleport w.p. $p_T$
Discrete Interpretation

Frog walks randomly on graph
Next vertex chosen uniformly at random

Teleportation
Every step: teleport w.p. $p_T$
Discrete Interpretation

Frog walks randomly on graph
Next vertex chosen uniformly at random

Teleportation
Every step: teleport w.p. $p_T$

Sampling after $t$ steps
Frog location gives sample from $\pi$
Discrete Interpretation

Frog walks randomly on graph
Next vertex chosen uniformly at random

Teleportation
Every step: teleport w.p. $p_T$

Sampling after $t$ steps
Frog location gives sample from $\pi$

PageRank Vector
Many frogs, estimate vector $\pi$
PageRank Approximation

Looking for $k$ “heavy nodes”
Do not need full PageRank vector

Random Walk Sampling
Favors heavy nodes

Captured Mass Metric
For node set $S$: $\pi(S)$
PageRank Approximation

Looking for k “heavy nodes”
Do not need full PageRank vector

Random Walk Sampling
Favors heavy nodes

Captured Mass Metric
For node set S: $\pi(S)$

k=2
Return set \{E,D\}
Captured mass = $\pi(\{E,D\})$
Graph Engines

- Engine splits graph across cluster
- Vertex program describes logic

GAS abstraction

Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]
Graph Engines

- Engine splits graph across cluster
- Vertex program describes logic

**GAS abstraction**

1. Gather

Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]
Graph Engines

- Engine splits graph across cluster
- **Vertex program** describes logic

**GAS abstraction**

1. Gather
2. Apply

Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]
Graph Engines

- Engine splits graph across cluster
- **Vertex program** describes logic

**GAS abstraction**

1. Gather
2. Apply
3. Scatter

Other approaches:
Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]
Edge Cuts

- Assign vertices to machines
- **Cross-machine** edges require network communication
- Pregel, GraphLab 1.0
- High-degree nodes generate large volume of traffic
- Computational load imbalance
Edge Cuts

- Assign vertices to machines
- Cross-machine edges require network communication
- Pregel, GraphLab 1.0
- High-degree nodes generate large volume of traffic
- Computational load imbalance
Vertex Cuts

- Assign edges to machines
- High-degree nodes replicated
- One replica designated master
- Need for synchronization
  1. Gather
  2. Apply [on master]
  3. Synchronize mirrors
  4. Scatter
- GraphLab 2.0 - PowerGraph
- Balanced - Network still bottleneck
Vertex Cuts

- Assign edges to machines
- High-degree nodes replicated
- One replica designated master
- Need for synchronization
  1. Gather
  2. Apply [on master]
  3. Synchronize mirrors
  4. Scatter
- GraphLab 2.0 - PowerGraph
- Balanced - Network still bottleneck
Vertex Cuts

- Assign edges to machines
- High-degree nodes replicated
- One replica designated master
- Need for synchronization
  1. Gather
  2. Apply [on master]
  3. Synchronize mirrors
  4. Scatter
- GraphLab 2.0 - PowerGraph
- Balanced - Network still bottleneck
Random Walks on GraphLab

Master node decides step

Decision synced to all mirrors

Only machine M needs it

Unnecessary network traffic

Average replication factor ~8
Random Walks on GraphLab

**Master** node decides step

Decision **synced** to all mirrors

Only machine M needs it

Unnecessary network traffic

Average replication factor ~8
Random Walks on GraphLab

**Master** node decides step

Decision **synced** to all mirrors

Only machine M needs it

**Unnecessary** network traffic

Average replication factor ~8
Objective

Faster PageRank approximation on GraphLab

Idea

Only synchronize the mirror that will receive the frog
Doable, but requires

1. Serious engine hacking
2. Exposing an ugly/complicated API to programmer

Simpler

Pick mirrors to synchronize at random!
Synchronize independently with probability $p_s$
FrogWild!

Release N frogs in parallel

**Vertex Program**

1. Each frog dies w.p. $p_T$ (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. $p_S$
3. Spread frogs evenly among synchronized mirrors.

$\text{Ber}(p_S)$
FrogWild!

Release N frogs in parallel

**Vertex Program**

1. Each frog dies w.p. $p_T$ (gives sample)  
   Assume K frogs survive
2. For every mirror, draw bridge w.p. $p_S$
3. Spread frogs evenly among synchronized mirrors.
FrogWild!

Release N frogs in parallel

**Vertex Program**

1. Each frog dies w.p. $p_T$ (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. $p_S$
3. Spread frogs evenly among synchronized mirrors.

Machine 1
- A → B
- B → C

Machine 2
- B → C

Machine 3
- B → D

Machine M
- B → Z

$\text{Ber}(p_S)$

$p_T$ (gives sample)
FrogWild!

Release N frogs in parallel

**Vertex Program**

1. Each frog dies w.p. $p_T$ (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. $p_S$
3. Spread frogs evenly among synchronized mirrors.
1. Each frog dies w.p. $p_T$ (gives sample) 
   Assume $K$ frogs survive 
2. For every mirror, draw bridge w.p. $p_S$ 
3. Spread frogs evenly among synchronized mirrors.

Release $N$ frogs in parallel

Vertex Program

FrogWild!
FrogWild!

Release N frogs in parallel

**Vertex Program**

1. Each frog dies w.p. $p_T$ (gives sample)
   Assume K frogs survive
2. For every mirror, draw bridge w.p. $p_S$
3. Spread frogs evenly among synchronized mirrors.

Machine 1

Machine 2

Machine 3

Machine M
FrogWild!

Release N frogs in parallel

**Vertex Program**

1. Each frog dies w.p. $p_T$ (gives sample)
   Assume K frogs survive

2. For every mirror, draw bridge w.p. $p_S$

3. Spread frogs evenly among synchronized mirrors.

**Bridges introduce dependencies!**

---

Machine 1

Machine 2

Machine 3

Machine M
Contributions

1. Algorithm for approximate PageRank
2. Modification of GraphLab
   Exposes very simple API extension (\(p_s\)). Allows for randomized synchronization.
3. Speedup of 7-10x
4. Theoretical guarantees for solution despite introduced dependencies
Theoretical Guarantee

Mass Captured by top-k set, S, of estimate from N frogs after t steps

$$\pi(S) \geq \text{OPT} - 2\epsilon \quad \text{w.p. } 1 - \delta$$

where

$$\epsilon < \sqrt{k} \lambda_2^t + \sqrt{\frac{k}{\delta}} \left[ \frac{1}{N} + (1 - p_S^2) p_n(t) \right]$$

probability two Frogs meet at first t steps

$$p_n(t) \leq \frac{1}{n} + \frac{t \|\pi\|_\infty}{p_T},$$
Experiments
Experimental Results

![Graph showing accuracy vs total time for different iterations and parameters.](image-url)
Experimental Results

![Graph showing time per iteration for different node counts under Twitter, AWS, 800K rw, 4 iters conditions. The graph compares GraphLab PR exact, FrogWild with different Ps values (Ps=1, Ps=0.7, Ps=0.4, Ps=0.1), and shows varying time per iteration for 12, 16, 20, and 24 nodes.]
Experimental Results

Twitter, AWS, 800K rw, 4 iters

GraphLab PR exact
GraphLab PR 2 ites
GraphLab PR 1 iters
FrogWild, Ps=1
FrogWild, Ps=0.1

Total time (s)

12 nodes 16 nodes 20 nodes 24 nodes
Thank you!


Backup Slides
PageRank [Page et al., 1999]
PageRank [Page et al., 1999]

Normalized Adjacency Matrix

\[ P_{i,j} = \frac{1}{d_{\text{out}}(j)}, \quad (j, i) \in G \]
PageRank [Page et al., 1999]

Normalized Adjacency Matrix

\[ P_{ij} = \frac{1}{d_{out}(j)}, \quad (j, i) \in G \]

Augmented Matrix \( p_T \in [0, 1] \)

\[ Q_{ij} = (1 - p_T) P_{ij} + \frac{p_T}{n} \]
PageRank \cite{Page et al., 1999}

Normalized Adjacency Matrix

\[ P_{ij} = \frac{1}{d_{out}(j)}, \quad (j, i) \in G \]

Augmented Matrix \( p_T \in [0, 1] \)

\[ Q_{ij} = (1 - p_T)P_{ij} + \frac{p_T}{n} \]

PageRank Vector \( \pi = Q\pi \)
PageRank [Page et al., 1999]

Normalized Adjacency Matrix

\[ P_{ij} = \frac{1}{d_{out}(j)}, \quad (j, i) \in G \]

Augmented Matrix \( p_T \in [0, 1] \)

\[ Q_{ij} = (1 - p_T)P_{ij} + \frac{p_T}{n} \]

PageRank Vector \( \pi = Q\pi \)

Power Method \( Q^t p^0 \rightarrow \pi \)
Here be dragons.
Backup