FrogWild! — Fast PageRank Approximations on Graph Engines
Ioannis Mitliagkas, Michael Borokhovich, Alex Dimakis, Constantine Caramanis

Web Ranking
Given web graph
Find “important” pages
Rank Based on In-degree
Classic Approach
Susceptible to manipulation by spammer networks

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 $S$: $\pi(S)$

Graph Engines
Vertex program
1. Gather
2. Apply
3. Scatter
Used by Pregel/GraphLab
Other approaches: Giraph [Avery, 2011], Galois [Nguyen et al., 2013], GraphX [Xin et al., 2013]

Discrete Interpretation
Frog walks randomly on graph
Teleportation
Every step: teleport w.p. $p_T = 0.15$
Sampling after t steps
Frog location gives sample from $\pi$

Random Walks?
Master node decides step
Decision synced to all mirrors
Average replication factor ~8

Network Bottleneck
Unnecessary network traffic

Experiments
Time per iteration/step
Total time
Code Repository: git.io/frogwild
Project Page: mitliagkas.github.io/frogwild

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 dependencies

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

Experimental Results
- GraphLab PR
- FrogWild PR

Page Importance
Described by distribution $\pi$

Recursive Definition
important pages linked to by important pages

Discrete Interpretation
frog jumps randomly on graph
Random Walks?
Master node decides step
Decision synced to all mirrors
Average replication factor ~8
Unnecessary network traffic

Page Importance
Described by distribution $\pi$

Recursive Definition
important pages linked to by important pages

Iteration
1. Distribute mass to successors evenly
2. Distribute a small fraction, $p_T = 0.15$, globally

PageRank [Page et al., 1999]
Vertex Splitting
Assign edges to machines
High-degree nodes replicated
One replica designated master
Need for synchronization

Random Mirror Synchronization
Flip coins with bias $p_s$.

Bridges introduce dependencies!

Solution despite dependencies!

FrogWild
Random Mirror Synchronization
Flip coins with bias $p_s$.

Bridges introduce dependencies!

Random Mirror Synchronization
Flip coins with bias $p_s$.

Bridges introduce dependencies!

Solution despite dependencies!