Estimating Single-Node PageRank in $\tilde{O}(\min\{d_t, \sqrt{m}\})$ Time

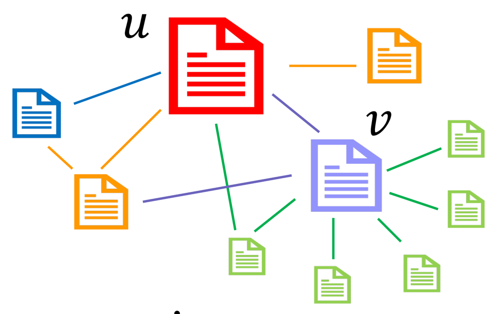
on undirected graphs!

Hanzhi Wang and Zhewei Wei ✉, Renmin University of China

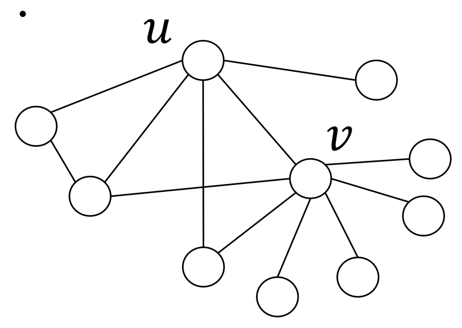
Problem Definition

- **Single-Node PageRank Query:** Given an undirected graph $G=(V,E)$, a target node $t \in V$ and a constant relative error $c \in (0,1)$, we aim to derive an estimated PageRank score $\hat{\boldsymbol{\pi}}(t)$ such that
$$|\boldsymbol{\pi}(t) - \hat{\boldsymbol{\pi}}(t)| < c \cdot \boldsymbol{\pi}(t)$$
holds with a constant probability.

- **PageRank:**
  - **History:** PageRank was first proposed by Google's cofounders to **evaluate the importance of web pages** in Google's search engine.
  - **Intuition:** a web page is important if
    - it is linked by many other web pages,
    - or by some important pages.



  - **Applications** has been far beyond web search, covering: information retrieval, recommender systems, social networks, biology, chemistry, neuroscience, …
  - **Definition Formula** of the PageRank vector $\boldsymbol{\pi}$:
$$\boldsymbol{\pi} = (1-\alpha)\mathbf{P}\boldsymbol{\pi} + \alpha \cdot \frac{\mathbf{1}}{n}.$$
    - $\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$: the probability transition matrix;
    - $\mathbf{A}$ and $\mathbf{D}$: the adjacency / diagonal degree matrix;
    - $\boldsymbol{\pi}(t)$: the PageRank score of node t.



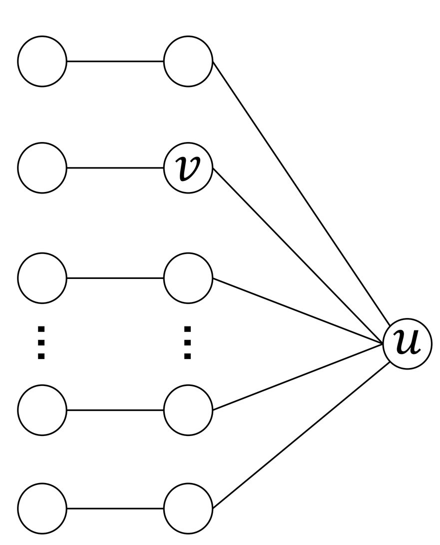
  - **Probabilistic Interpretation:**
    - $\boldsymbol{\pi}(t)$: an **α-random walk** generated from a random source node terminates at node t.

      At each step (e.g., at node u), the walk
      - **either** terminates at u w.p. α;
      - **or** moves to a random neighbor $v \in N(u)$ w.p. $1-\alpha$

Limitations of Existing Methods

- **Monte-Carlo method:** $\hat{\boldsymbol{\pi}}(t) = \dfrac{\#\text{ of } \alpha\text{-walks terminates at } t}{\text{total } \#\text{ of } \alpha\text{-walks generated in } G}$
  - According to the **Pigeonhole Principle**, to estimate $\pi(t) = O(1/n)$, we need to generate $\Omega(n)$ random walks in order to touch node t at least once.
- **LocalPush method:**
  - Deterministically touch every neighbor to push probability. Thus for large-degree nodes, the push method cost $O(n)$ right after the first push step.

  For each $v \in N(u)$ **do**:
  $$\boldsymbol{r}_t^{(i+1)}(v) \leftarrow \boldsymbol{r}_t^{(i+1)}(v) + \frac{(1-\alpha)\cdot \boldsymbol{r}_t^{(i)}(u)}{\boldsymbol{d_v}};$$



- Some nice properties of pagerank held only on undirected graphs
  - Lower bound on directed graphs: $\tilde{O}\left(\min\{\sqrt{n\Delta}, n^{\frac{2}{3}} d^{\frac{1}{3}}\}\right)$

Our Contributions

- **A Humble Goal:** a local algorithm with $o(n)$ query time to derive $\hat{\boldsymbol{\pi}}(t)$

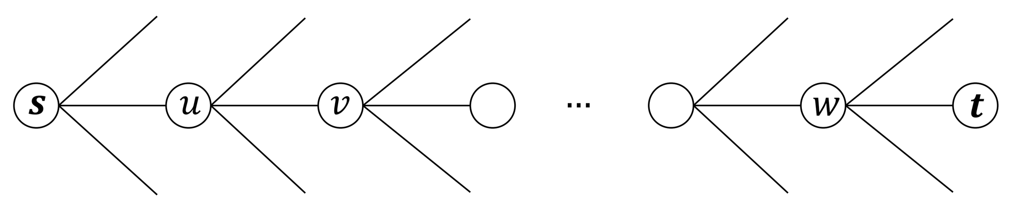
  only explores a small fraction of graph G

Algorithms	Worst-Case Query Time Complexity
Power Iteration [WWW'98]	$\tilde{O}(m)$
Monte-Carlo [Internet Mathematics'05]	$\tilde{O}(n)$
LocalPush [Lofgren et al.'13]	$\tilde{O}(\min\{n\cdot d_t, m\})$
RBS [KDD'20]	$\tilde{O}(n)$
FastPPR [KDD'14]	$\tilde{O}(\sqrt{n\cdot d_t})$ SOTA
BiPPR [WAW'15, WSDM'16]	$\tilde{O}(\sqrt{n\cdot d_t})$ SOTA
SubgraphPush [FOCS'18]	$\tilde{O}\left(\min\{n^{\frac{2}{3}}\Delta^{\frac{1}{3}}, n^{\frac{4}{5}} d^{\frac{1}{5}}\}\right)$
SetPush [Ours]	$\tilde{O}(\min\{d_t, \sqrt{m}\})$

- d_t: degree of node t; d: average node degree; Δ: maximum node degree;
- n: the number of nodes in G; m: the number of edges in G, $m = nd$;
- $\tilde{O}$: all poly-logarithmic factors are omitted.

- **High-Level Idea:** forward push probability from the target node t
  - utilize the **symmetry** of random walk probability on undirected graphs;
  $$\boldsymbol{\pi}_s(t)\cdot d_s = \boldsymbol{\pi}_t(s)\cdot d_t, \quad \text{for } \forall s,t \in V$$
  The probability that an α-walk generated **from s** terminates **at t**

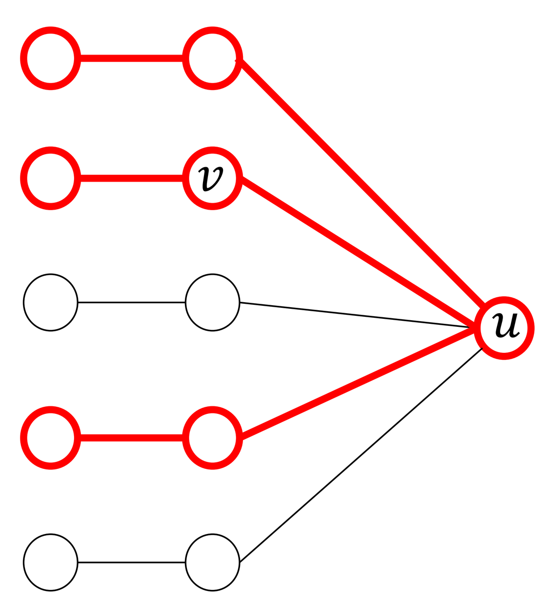
  The probability that an α-walk generated **from t** terminates **at s**



$$\Pr\{s\to t\} = \frac{1}{d_s}\cdot\frac{1}{d_u}\cdot\frac{1}{d_v}\cdots\frac{1}{d_w}$$
$$\Pr\{t\to s\} = \frac{1}{d_u}\cdot\frac{1}{d_v}\cdots\frac{1}{d_w}\cdot\frac{1}{\boldsymbol{d_s}}$$

  - $\boldsymbol{\pi}(t) = \frac{1}{n}\cdot\sum_{u\in V}\boldsymbol{\pi}_u(t) = \frac{1}{n}\cdot\sum_{u\in V}\frac{d_t}{d_u}\cdot\boldsymbol{\pi}_t(u)$
  - For **small-degree nodes**: deterministically push probability mass to all neighbors;
  - For **large-degree nodes**: sample a small fraction of neighbors to push probability.

randomized forwad push



If $d_u < (1-\alpha)\cdot \boldsymbol{r}_t^{(i)}(u)/\theta$ **do**:

  For each $v \in N(u)$ **do**:

  $\boldsymbol{r}_t^{(i+1)}(v) \leftarrow \boldsymbol{r}_t^{(i+1)}(v) + \frac{(1-\alpha)\cdot\boldsymbol{r}_t^{(i)}(u)}{\boldsymbol{d_u}}$;

Else do:

  Independently sample each $v \in N(u)$ w.p. $\frac{(1-\alpha)\cdot\boldsymbol{r}_t^{(i)}(u)}{\theta\cdot d_u}$;

  For each sampled neighbor $v \in N(u)$ **do**:

  $\boldsymbol{r}_t^{(i+1)}(v) \leftarrow \boldsymbol{r}_t^{(i+1)}(v) + \theta$;

Experiments

Datasets	# of nodes n	# of edges m
Youtube (YT)	1,138,499	5,980,886
IndoChina (IC)	7,414,768	301,969,638
Orkut-Links (OL)	3,072,441	234,369,798
Friendster (fr)	68,349,466	3,623,698,684

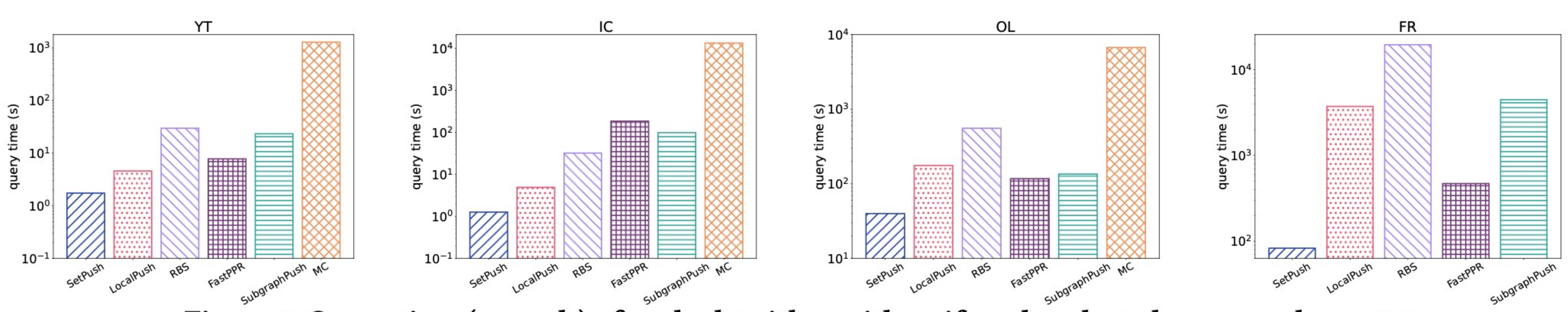


Figure 3: Query time (seconds) of each algorithm with uniformly selected query node, $c = 0.1$