Approximating Single-Source Personalized PageRank with Absolute Error Guarantees

Zhewei Wei, Ji-Rong Wen, Mingji Yang

Renmin University of China, Beijing, China

ICDT 2024, 2024.03.27

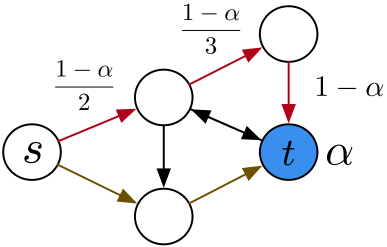
Introduction: Personalized PageRank (PPR)

- random-walk-based node proximity measure in graphs
- extension of PageRank, fundamental tool in graph mining
- many recent works in VLDB, SIGMOD, KDD...
- applications: local graph partitioning, graph sparsification, node embedding, graph neural networks...

- we consider efficient approximation of Single-Source PPR

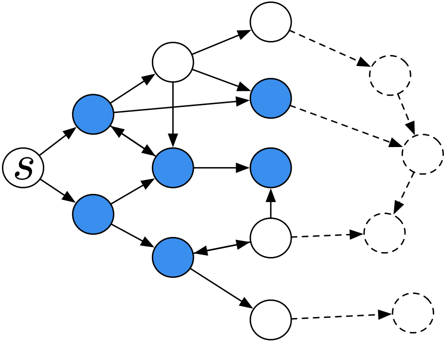
Problem Formulation: Random Walks and PPR

- graph $G = (V, E)$, $n = |V|$, $m = |E|$
- α-discounted random walk: random walk that terminates w.p. α at each step, *decay factor* $\alpha \in (0, 1)$ (i.e., length of the walk follows $\mathrm{Geom}(\alpha)$)
- for two nodes $s, t \in V$, PPR value $\pi(s, t)$ equals the probability that an α-discounted random walk from s terminates at t



Problem Formulation: Single-Source PPR (SSPPR)

- for a source node $s \in V$, estimate $\pi(s,t)$ for all $t \in V$
- can leave out some small $\pi(s,t)$
- sublinear-time complexity: $o(m)$ dependence on m, the algorithm may not inspect the whole graph



Problem Formulation: Error Guarantees

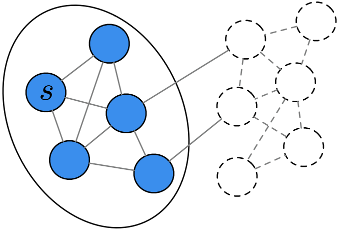
Absolute Error

given source node s and error parameter ε, return estimates $\hat{\pi}(s,t)$ such that $\left|\hat{\pi}(s,t) - \pi(s,t)\right| \leq \varepsilon$ for all $t \in V$

Degree-Normalized Absolute Error (for undirected graphs)

given source node s and error parameter ε_d, return estimates $\hat{\pi}(s,t)$ such that $\left|\hat{\pi}(s,t) - \pi(s,t)\right| \leq \varepsilon_d \cdot d(t)$ for all $t \in V$, where $d(t)$ is the degree of node t

Motivation: Local Graph Clustering



- detect a provably-good cluster near source node s
- PageRank-Nibble [Andersen, Chung, Lang, FOCS '06, Internet Math. '07] approximates SSPPR from s and returns nodes with large PPR values as a cluster
- their PPR estimates satisfy degree-normalized absolute error guarantees

Related Work

- sublinear-time algorithms for SSPPR with absolute error guarantees are rarely studied

- direct Monte Carlo sampling: $\widetilde{O}\big(1/\varepsilon^2\big)$
- Forward Push [ACL06]:
  - only on undirected graphs, $O(d_{\max}/\varepsilon)$, $d_{\max}$ denotes the maximum degree
  - $O(1/\varepsilon_d)$
- explained in detail later

- our "Efficient Algorithms for Personalized PageRank Computation: A Survey" [TKDE '24]

Our Results: Absolute Error

	Directed Graphs	Undirected Graphs	Power-Law Graphs
Monte Carlo	$\widetilde{O}\left(\frac{1}{\varepsilon^2}\right)$	$\widetilde{O}\left(\frac{1}{\varepsilon^2}\right)$	$\widetilde{O}\left(\frac{1}{\varepsilon^2}\right)$
Forward Push	$O\left(\frac{m}{\varepsilon}\right)$	$O\left(\frac{d_{\max}}{\varepsilon}\right)$	$\widetilde{O}\left(\frac{n}{\varepsilon}\right)$
Ours	$\widetilde{O}\left(\frac{\sqrt{m}}{\varepsilon}\right)$	$\widetilde{O}\left(\frac{\sqrt{d_{\max}}}{\varepsilon}\right)$	$\widetilde{O}\left(\frac{n^{\gamma-1/2}}{\varepsilon}\right)$

- n, m: number of nodes/edges
- ε: error bound parameter
- $d_{\max}$: maximum degree
- γ: exponent of the power law, $\gamma \in (1/2, 1)$

Our Results: Degree-Normalized Absolute Error

	Complexity for a given s	Average complexity when each $s \in V$ is chosen w.p. $d(s)/(2m)$
Forward Push	$O\left(\frac{1}{\varepsilon_d}\right)$	$O\left(\frac{1}{\varepsilon_d}\right)$
Ours	$\widetilde{O}\left(\frac{1}{\varepsilon_d}\sqrt{\sum_{t\in V}\frac{\pi(s,t)}{d(t)}}\right)$	$\widetilde{O}\left(\frac{1}{\varepsilon_d}\sqrt{\frac{n}{m}}\right)$

- ε_d: error bound parameter

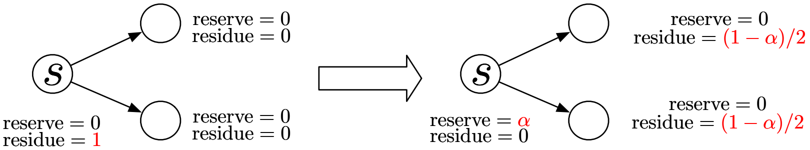
Basic Techniques: Monte Carlo Sampling

- simulate α-discounted random walks from s to estimate the probability that it terminates at each node
- by Chernoff bound, $\widetilde{O}\big(1/\varepsilon^2\big)$ complexity w.h.p.

Forward Push [ACL06]

- simulates α-discounted random walk from s in a deterministic way
- perform push operations to propagate and transfer probability mass
- each push operation corresponds to a step in α-discounted random walk

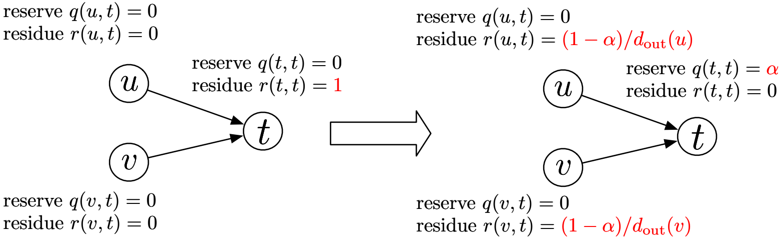
Forward Push (Cont'd)



- residue: probability mass to be propagated and transfered to reserve
- reserve: probability mass corresponding to stopping at each node, underestimate of PPR value
- repeat this push operations

Backward Push [WAW '07, Internet Math. '08]

- estimates PPR to a target node t (Single-Target PPR)
- a reversed counterpart of Forward Push
- each pushback operation corresponds to a "reversed" step in α-discounted random walk



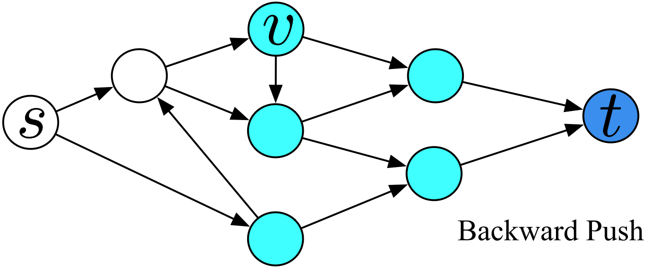
Backward Push (Cont'd)

- set a threshold $r_{\max}$ and repeatedly perform pushback operations until all residues $r(v, t) \leq r_{\max}$
- upon completion, the reserves satisfy $\pi(v, t) - r_{\max} \leq q(v, t) \leq \pi(v, t)$ (absolute error guarantees!)

Our Algorithms: High-Level Ideas

- mainly consider absolute error. our algorithm for degree-normalized error shares the same framework
- Monte Carlo and Forward Push inherently incur larger errors for nodes with larger PPR values or degrees
- we can use Backward Push to reduce errors for these hard-case nodes
- Backward Push can be combined with Monte Carlo to reduce cost: the goal of random walk sampling shifts from hitting t to hitting nodes explored by Backward Push

Our Algorithms: High-Level Ideas (Cont'd)



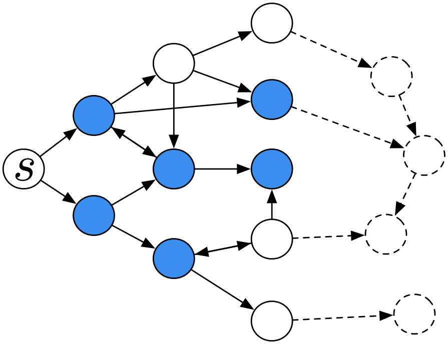
random walk sampling increases the estimate for $\pi(s, t)$ as long as it hits the intermediate nodes explored by Backward Push from t

Our Algorithms: High-Level Ideas (Cont'd)

- however, performing "deep" Backward Push for each node is prohibitively costly and unnecessary
- we need to:
  (1) only conduct Backward Push for a small number of nodes
  (2) only conduct "deep" Backward Push for hard-case nodes
- for (1), using Monte Carlo to detect the nodes t with $\pi(s,t) > \varepsilon$ only requires $\widetilde{O}(1/\varepsilon)$ time
- for (2), we wish to set $r_{\max}(t)$ smaller for nodes t with larger PPR values
  - a straightforward way is to set $r_{\max}(t)$ inversely proportional to $\pi(s,t)$
  - dilemma: $\pi(s,t)$ are what we want to estimate
  - workaround: rough Monte Carlo estimates suffice

Our Algorithms: Process

- Phase I: run Monte Carlo to detect candidate nodes and obtain rough PPR estimates, takes $\widetilde{O}(1/\varepsilon)$ time



Our Algorithms: Process (Cont'd)

- Phase II: run Backward Push for candidate nodes, where $r_{\max}(t)$ are set inversely propotional to Monte Carlo estimates for $\pi(s, t)$
- Phase III: perform Monte Carlo simulations again and combine the results with Backward Push, yielding final estimates

Our Algorithms: Analysis of Error Guarantees

- Backward Push for node t expresses $\pi(s,t)$ as $q(s,t) + \sum_{v \in V} \pi(s,v) r(v,t)$
- use Monte Carlo to estimate $\pi(s,v)$ therein
- as $r(v,t) \le r_{\max}(t)$, this leads to a low-variance estimator
- by Chebyshev's inequality, it suffices to set $r_{\max}(t) = \frac{\varepsilon^2 \cdot n_r}{\pi(s,t)}$, where n_r denotes the number of random walk samplings in Phase III

Our Algorithms: Analysis of Complexity

- complexity of Backward Push for node t is $O\left(\frac{\sum_{v\in V}\pi(v,t)d_{\text{in}}(v)}{r_{\max}(t)}\right)$
- plugging in $r_{\max}(t) = \frac{\varepsilon^2\cdot n_r}{\pi(s,t)}$ leads to a total complexity of

$$\widetilde{O}\left(\frac{1}{\varepsilon^2 n_r}\sum_{t\in V}\pi(s,t)\sum_{v\in V}\pi(v,t)d_{\text{in}}(v)+n_r\right)$$

- setting n_r to balance the two terms leads to

$$\widetilde{O}\left(\frac{1}{\varepsilon}\sqrt{\sum_{t\in V}\pi(s,t)\sum_{v\in V}\pi(v,t)d_{\text{in}}(v)}\right)$$

Our Algorithms: Analysis of Complexity (Cont'd)

- this complexity can be bounded by $\widetilde{O}\left(\sqrt{m}/\varepsilon\right)$
- on undirected graphs, it can be further bounded by $\widetilde{O}\left(\sqrt{d_{\max}}/\varepsilon\right)$, using a symmetry property of PPR
- if we assume PPR values follow a power law, it can be bounded by $\widetilde{O}\left(n^{\gamma-1/2}/\varepsilon\right)$, $\gamma \in (1/2, 1)$

- subtlety: we do not know the best setting of n_r beforehand, but can use a doubling technique to achieve these bounds
- for degree-normalized error, we set $r_{\max}(t) = \left(d(t)\right)^2 \cdot \frac{\varepsilon_d^2 \cdot n_r}{\pi(s,t)}$

Conclusions

- we combine Monte Carlo and Backward Push to improve the upper bounds of approximating Single-Source Personalized PageRank with (degree-normalized) absolute error guarantees
- e.g., on undirected graphs
  - $O(d_{\max}/\varepsilon) \Rightarrow \widetilde{O}\left(\sqrt{d_{\max}}/\varepsilon\right)$ for absolute error
  - $O(1/\varepsilon_d) \Rightarrow \widetilde{O}\left(1/\varepsilon_d \cdot \sqrt{n/m}\right)$ for degree-normalized error

Future Directions

- tighten the upper bounds and/or lower bounds
  - nontrivial lower bounds?
- apply our algorithm for degree-normalized error to local graph clustering

Thank you!