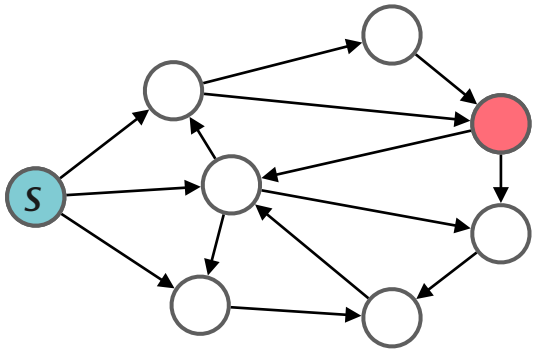
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

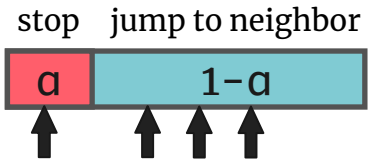
CIKM Research

Agenda: Robust Personalized PageRanks in Evolving Graphs

Dingheng Mo, Siqiang Luo

Personalized PageRank





The random walk interpretation of Personalized PageRank:

- $\pi(s, t)$ = probability that a random walk from s stops at t

A α-*decay* random walk from s

- Stop and restart from s with α probability at each step.
- If not stop, randomly jumps to one of its out-neighbors.

Approximate single-source PPR query

On graph G(V, E). With a source node s, for each node v , return an estimation value of PPR from s to v, denote as $\hat{\pi}(s, v)$.

Guarantee the estimation incurs at most ϵ relative error

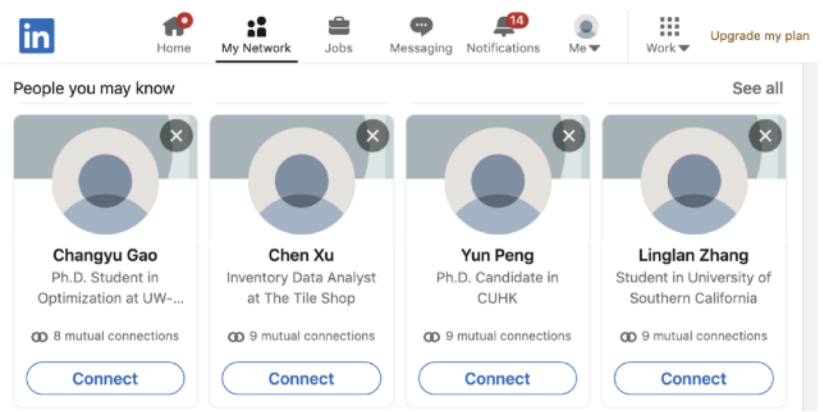
$$|\pi(s, v) - \hat{\pi}(s, v)| \leq \epsilon \cdot \pi(s, v)$$

for any $\pi(s,v) > \delta$.

Application

- Personalized search and recommendation

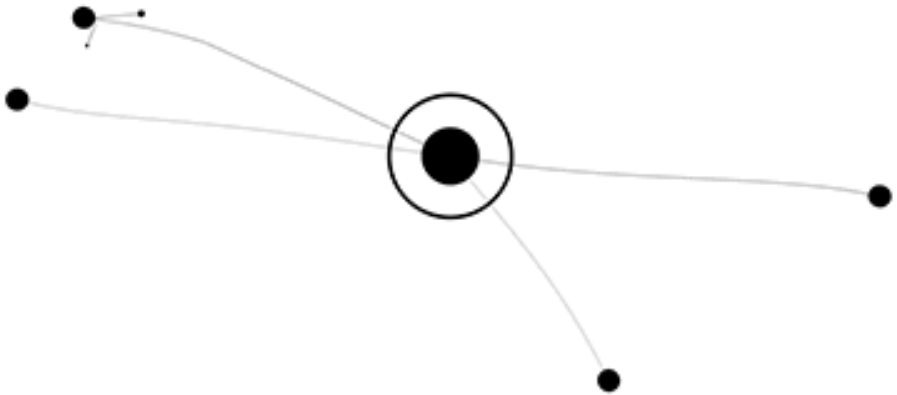
Twitter Who-to-Follow Service



Linkedin user recommendation

Motivation

Social networks in real applications are highly dynamic.



Problem definition

With current graph $G(V, E)$

Update(u, v):

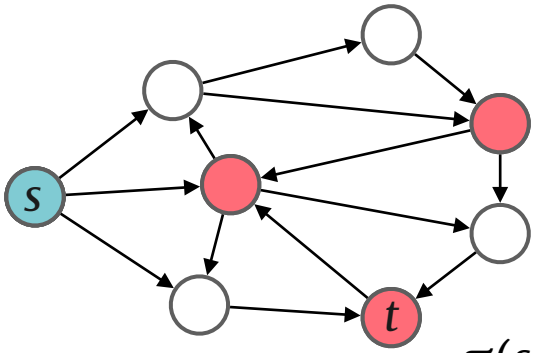
- If edge (u, v) is not in E, insert (u, v) into the graph.
- If edge (u, v) is already in E, delete (u, v) from the graph.

Query(s):

- For each node v, return the approximate single source PPR value from s to v on current graph with accuracy guarantee.

Given a mixture of queries and updates, we aim to design query&update scheme to optimize the total running time of finishing all these queries and updates.

Previous solution: Monte-carlo



$\pi(s, t) = 1/3$

Basic idea: Simulate personalized pagerank by sampling random walks.

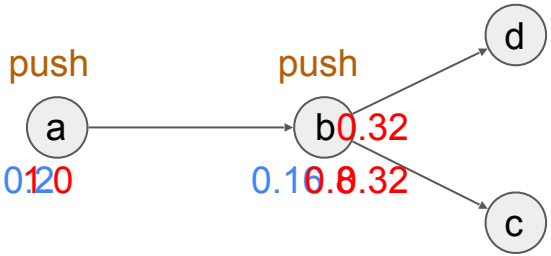
Compute $\pi(s, t)$: Sample random walks from s, return fraction of walks end at t.

[Bahmani VLDB'10] On dynamic graph, index and update random walk segments.

- Requires to store all the intermediate nodes of random walks to maintain the index.
- Stores $O(\frac{n^2 \log n}{\epsilon^2})$ random walks for queries on whole graph, which is too large.

Previous solution: Forward push

α=0.2



The forward push algorithm maintains a residue and a reserve for each node v

- Reserve : random walks that stops at node v.
- Residue : random walks that are currently at node v.

Forward push operation on node v

- Transfers α portion of its residue to its reserve
- Equally distributes the remaining $(1 - \alpha)$ portion to the out-neighbors of v.

State-of-the-art solution: FORA

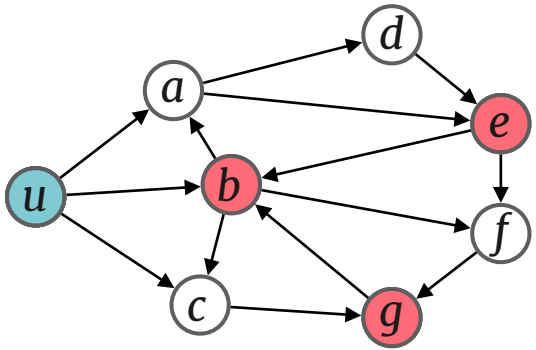
The PPR value can be estimated by

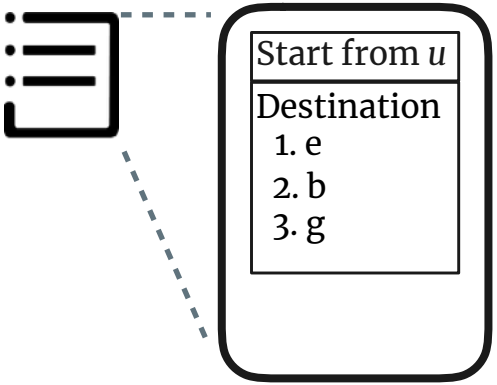
$$\pi(s,t) = \pi^{\circ}(s,t) + \sum_{v \in V} r(s,v) \cdot \pi(v,t)$$

Reserve *Residue*

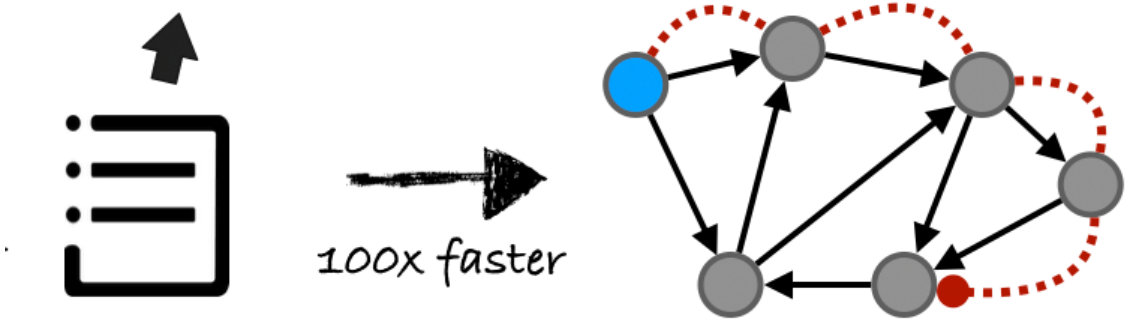
1. Process forward push from s to get reserve and residue.
2. Sample random walks from each node v to simulate $\sum_{v \in V} r(s,v) \cdot \pi(v,t)$

Index





Challenge



Index-based approaches works great for query heavy workload.
Index-free approaches works great for update heavy workload.

There can be thorny workloads for both index-based and index-free approaches.

Our solution: Agenda

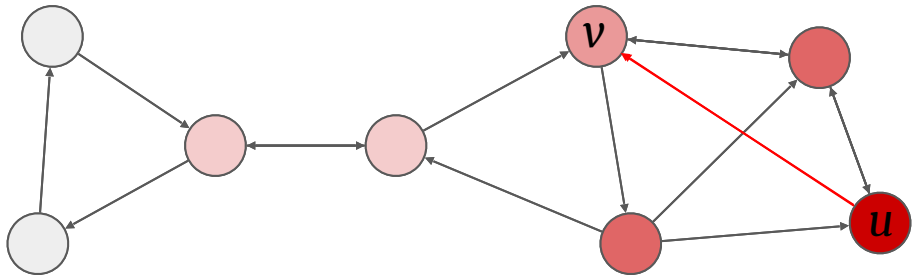
Agenda

- ➔ Is efficient on both query and update.
- ➔ Maintains index structure incrementally on dynamic graphs.
- ➔ Gives low query and update complexity.
- ➔ Faster than existing index-based and index-free methods on more than 95% of workloads.
- ➔ Up to 100x improvement than state-of-the-arts on dynamic graphs.

Intuition

An edge update (u, v) on graph increase inaccuracy of the random walk index.

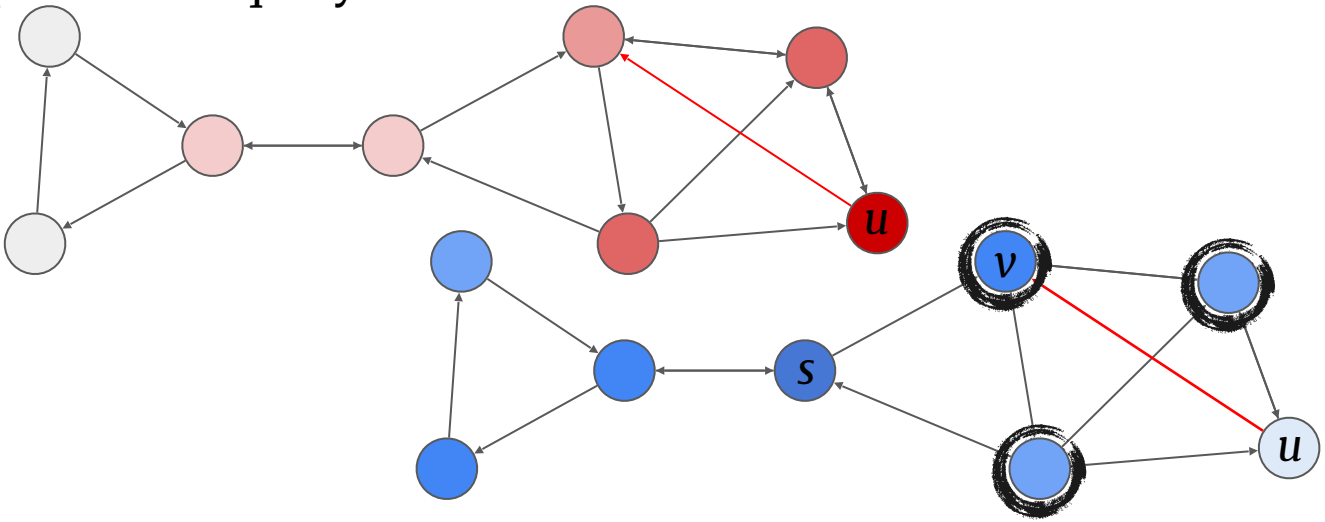
However, the increment of inaccuracy is very different on each node.



The index of nodes closer to u have larger inaccuracy.

Intuition

Even for the nodes with high index inaccuracy, we still need to consider their importance in query.

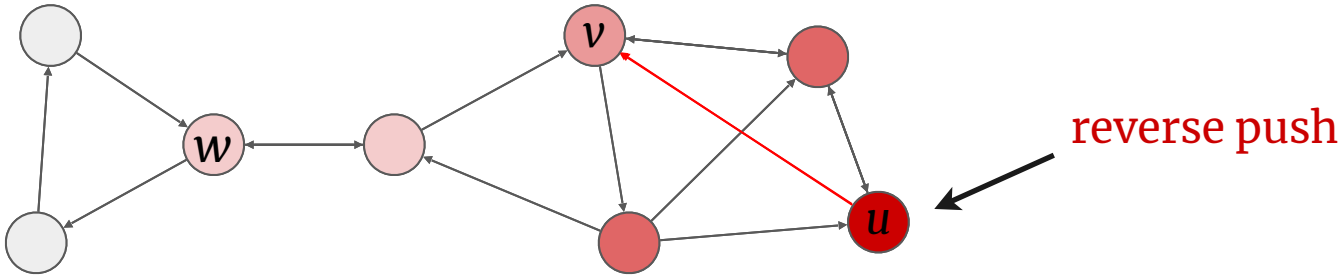


Framework

When it arrives an update(u, v):

- Perform a reverse push on u to compute the increment of the index inaccuracy on each node.
- Keep track of the index inaccuracy of each node.

ΔIndex inaccuracy(w) 🔗 $\pi(w, u)$



Framework

When it arrives a query(s):

- Perform forward push on s to get reserve and residue.
- Update indexes on nodes which are overly stale according to index inaccuracy and residue on each node.
- Process random walks to get result with guarantee.

Low update cost of Agenda

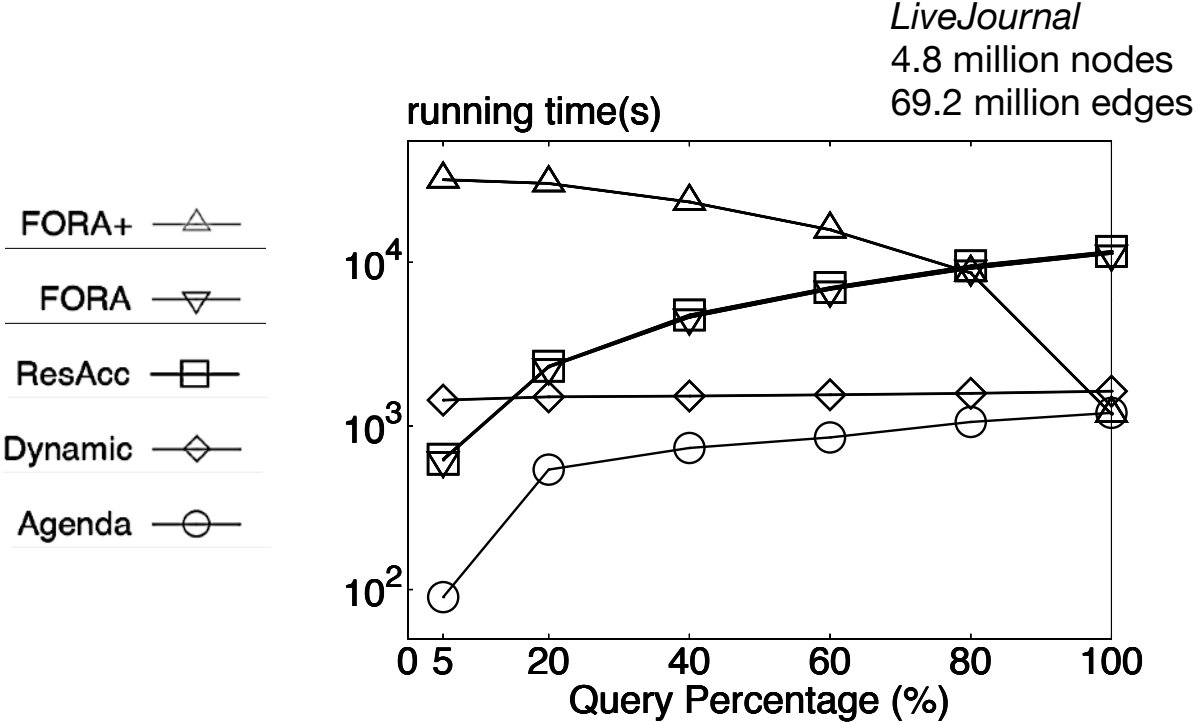
Agenda guarantees

- In an undirected graph, an edge update incurs at most $O(\epsilon/\log n)$ fraction of the index to be regenerated.

Evaluation Baseline

- FORA & FORA+ [Wang et al. KDD’17]
- ResAcc [Lin et al. ICDE’20]
- Dynamic [Bahmani VLDB’10]+[Wang et al. KDD’17]

Evaluation



Agenda shows better robustness compared with state-of-the-art approaches.

Conclusion

We present Agenda, an efficient & robust algorithm for single-source PPR queries in dynamic graphs.

Future work

- Extend our framework to different types of queries.
- Make better trade-off between update, query and space.

Thank you

Q&A