Fast and Accurate Random Walk with Restart on Dynamic Graphs with Guarantees

Minji Yoon, Woojeong Jin, and U Kang
Data Mining Lab
Dept. of CSE
Seoul National University

Outline

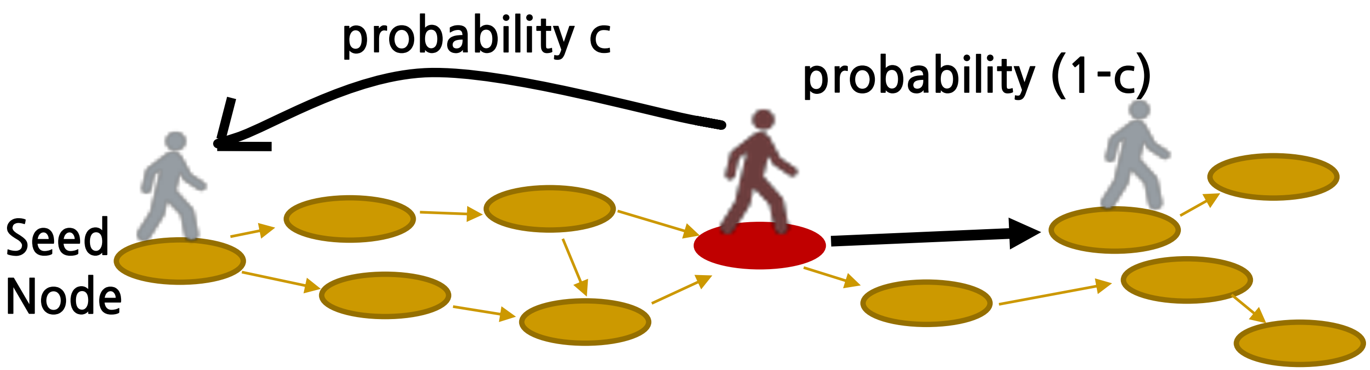
- **Problem Definition**
- Proposed Method
- Experiments
- Conclusion

Motivation

- Measuring similarity score between two nodes in a graph
  - Various applications across different domains
  - Ranking, Community detection, Link prediction, and Anomaly Detection.
- Random Walk with Restart (RWR)
  - Consider the global network from a particular user's point of view

Random Walk with Restart

- A random surfer
  - Start at seed node
  - Walk along edges with probability $(1 - c)$
  - Jump back to the seed node with probability c



Challenges

- Majority of RWR methods have focused on static graphs
- Many real-world graphs are dynamic
  - Facebook: +5 users/second
  - World Wide Web: $\pm$600,000 webpages/second
- RWR computation on dynamic graphs

Problem Definition
: Dynamic RWR

- **Given:** previous RWR vector $\mathbf{r}_{\boldsymbol{old}}$, row-normalized adjacency matrix $\tilde{\mathbf{A}}$, update in $\tilde{\mathbf{A}}$: $\Delta\mathbf{A}$, seed node s, restart probability c

$$\mathbf{r}_{\text{old}} = (1-c)\tilde{\mathbf{A}}^{\mathrm{T}}\mathbf{r}_{\text{old}} + c\boldsymbol{q}_{\text{s}}$$

- **Find:** updated RWR vector $\mathbf{r}_{\boldsymbol{new}}$ of updated graph $\tilde{\mathbf{A}} + \Delta\mathbf{A}$ which satisfice the following equation:

$$\mathbf{r}_{\text{new}} = (1-c)(\tilde{\mathbf{A}} + \Delta\mathbf{A})^{\mathrm{T}}\mathbf{r}_{\text{new}} + c\boldsymbol{q}_{\text{s}}$$

Problem Definition : Dynamic RWR

- **Input:**
  - $\mathbf{r}_{\text{old}} \in \mathbb{R}^{n\times 1}$: previous RWR score vector
  - $\tilde{\mathbf{A}} \in \mathbb{R}^{n\times n}$: row-normalized adjacency matrix of graph G
  - $\tilde{\mathbf{B}} \in \mathbb{R}^{n\times n}$: row-normalized adjacency matrix of updated graph $G + \Delta G$
    - $\Delta\mathbf{A} = \tilde{\mathbf{B}} - \tilde{\mathbf{A}}$, difference between $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$
  - $\mathbf{q}_s \in \mathbb{R}^{n\times 1}$: seed vector ($s$-th unit vector)
  - $c \in \mathbb{R}$: restart probability
- **Output:**
  - $\mathbf{r}_{\text{new}} \in \mathbb{R}^{n\times 1}$: updated RWR score vector

CPI: Cumulative Power Iteration

- Static RWR computation method
- Re-interpretation of RWR
- Propagation of scores across a graph
  1) Score c is generated from the seed node
  2) At each step, scores are divided evenly into out-edges with decaying coefficient $(1 - c)$
  3) Each node accumulates scores they have received
  4) Accumulated scores become RWR score of each node

CPI: Cumulative Power Iteration

$$\mathbf{x}^{(0)} = c\mathbf{q}$$

1) Initial score c at seed node

$$\mathbf{x}^{(i)} = (1-c)\tilde{\mathbf{A}}^\top \mathbf{x}^{(i-1)} = c\left((1-c)\tilde{\mathbf{A}}^\top\right)^i \mathbf{q}$$

2) scores are divided evenly into out-edges with (1-c)

$$\mathbf{r}_{\mathrm{CPI}} = \sum_{i=0}^{\infty} \mathbf{x}^{(i)} = c\sum_{i=0}^{\infty}\left((1-c)\tilde{\mathbf{A}}^\top\right)^i \mathbf{q}$$

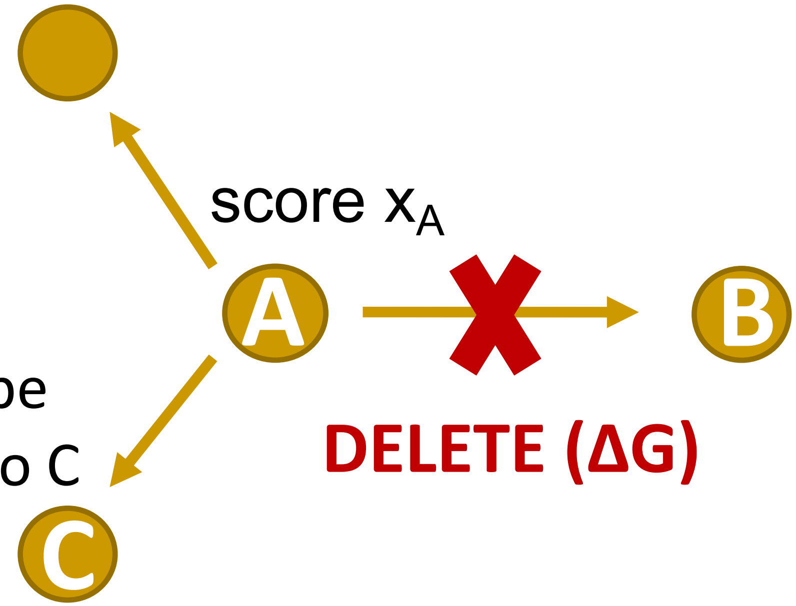
3) CPI accumulate interim scores of each node to get final results

- $\mathbf{x}(i) \in \mathbb{R}^{n\times 1}$
  - Interim score vector computed from ith iteration
  - Have scores propagated across nodes at ith iteration as entries

Outline

- Problem Definition
- **Proposed Method**
- Experiments
- Conclusion

Score Propagation on Dynamic Graph

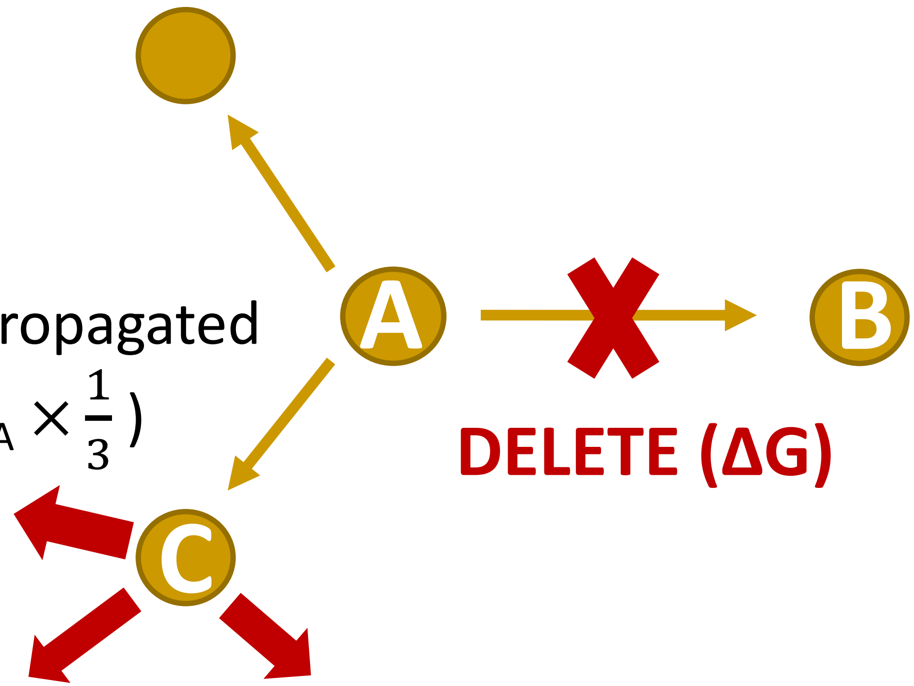


More scores would be propagated from A to C

: $\frac{1}{3}x_A \rightarrow \frac{1}{2}x_A$

- RWR scores of nodes are determined by arrangement of edges
  1. When the graph G is updated with ΔG
  2. Propagation of scores around ΔG is changed

Score Propagation on Dynamic Graph



The changes are propagated

: $(\frac{1}{3}X_A \times \frac{1}{3}) \rightarrow (\frac{1}{2}X_A \times \frac{1}{3})$

3. These small changes are propagated
4. Affect previous propagation pattern across whole graph
5. Finally lead to $\mathbf{r}_{\text{new}}$ different from $\mathbf{r}_{\text{old}}$

OSP: Offset Score Propagation

$$\mathbf{q}_{\text{offset}} \leftarrow (1-c)(\tilde{\mathbf{B}}^\top - \tilde{\mathbf{A}}^\top)\mathbf{r}_{\text{old}} = (1-c)(\Delta\mathbf{A})^\top \mathbf{r}_{\text{old}}$$

$$\mathbf{x}^{(i)}_{\text{offset}} \leftarrow ((1-c)\tilde{\mathbf{B}}^\top)^i \mathbf{q}_{\text{offset}}$$

$$\mathbf{r}_{\text{offset}} \leftarrow \sum_{i=0}^{\infty} \mathbf{x}^{(i)}_{\text{offset}} = \sum_{i=0}^{\infty}((1-c)\tilde{\mathbf{B}}^\top)^i \mathbf{q}_{\text{offset}}$$

$$\mathbf{r}_{\text{new}} \leftarrow \mathbf{r}_{\text{old}} + \mathbf{r}_{\text{offset}}$$

Convergence: Lemma3.1
Exactness: Theorem3.2

1. Calculate **an offset seed vector $\mathbf{q}_{offset}$**
2. Propagate the offset scores across G+ΔG to get **an offset score vector $\mathbf{r}_{offset}$**
3. Finally, OSP **adds up $\mathbf{r}_{old}$ and $\mathbf{r}_{offset}$** to get $\boldsymbol{r}_{new}$

OSP-T: OSP with Trade-off

Algorithm 1: OSP and OSP-T Algorithm

Require: previous RWR score vector: $\mathbf{r}_{\text{old}}$, row-normalized adjacency matrix: $\tilde{\mathbf{A}}$, update in $\tilde{\mathbf{A}}$: $\Delta\mathbf{A}$, restart probability: c, error tolerance: ϵ

Ensure: updated RWR score vector: $\mathbf{r}_{\text{new}}$

```
set seed offset vector q_offset = (1 − c)(ΔA)^T r_old
set r_offset = 0 and x^(0)_offset = q_offset
for iteration i = 1; ||x^(i)_offset||_1 > ϵ; i++ do
    compute x^(i)_offset ← (1 − c)(Ã + ΔA)^T x^(i−1)_offset
    compute r_offset ← r_offset + x^(i)_offset
end for
r_new ← r_old + r_offset
return r_new
```

- Approximate method for dynamic RWR
- Use the same algorithm with OSP
- Regulates accuracy and speed using higher error tolerance parameter ε

OSP-T: OSP with Trade-off

- Time complexity (Theorem 3.3)

THEOREM 3.3 (TIME COMPLEXITY OF OSP-T). *With error tolerance ϵ, OSP-T takes $O(m \log_{(1-c)}(\frac{\epsilon}{2}))$ where m is the number of nonzeros in $\tilde{\mathbf{A}} + \Delta\mathbf{A}$.*

- Error bound (Theorem 3.4)

THEOREM 3.4 (ERROR BOUND OF OSP-T). *When OSP-T converges under error tolerance ϵ, error bound of RWR score vector $\mathbf{r}_{new}$ computed by OSP-T is $O(\frac{\epsilon}{c})$.*

OSP-T: OSP with Trade-off

Method	Speed	Accuracy	Coverage	Accuracy Bound	Time complexity model
TrackingPPR	Fast	Low	Undirected graph	No	Only with insertion of edges
LazyForward	Fast	Low	Undirected graph	No	Only with undirected graph
OSP	**Medium**	**High**	**Directed/Undirected graph**	**Yes**	**General**
OSP-T	**Faster**	**Medium**	**Directed/Undirected graph**	**Yes**	**General**

- Previous Methods: TrackingPPR[1], LazyForward[2]
  - Fail to provide theoretical accuracy bound
  - Narrow down the scope of time complexity analysis
    - ΔG only with insertion of edges
    - ΔG on undirected graphs.

[1] Naoto Ohsaka, Takanori Maehara, and Kenichi Kawarabayashi, *Efficient PageRank tracking in evolving networks*, In Proceedings of the 21th ACM SIGKDD

[2] Hongyang Zhang, Peter Lofgren, and Ashish Goel, *Approximate Personalized PageRank on Dynamic Graphs,* In Proceedings of the 22th ACM SIGKDD

Discussion: Fast Convergence

- OSP,OSP-T, and CPI
  - Same upper bound O(m) for # visited edges / iteration
- In practice, OSP and OSP-T visit only small portion of edges:

Unit Vector

$$\mathbf{q}_{\text{offset}} = (1-c)(\Delta\mathbf{A})^{\top}\mathbf{r}_{\text{old}}$$

$$\|\mathbf{q}_{\text{offset}}\|_1 \leq (1-c)\|(\Delta\mathbf{A})^{\top}\|_1$$

Sparse Matrix with small update

$$\mathbf{x}_{\text{offset}}^{(i)} \leftarrow ((1-c)\tilde{\mathbf{B}}^{\top})^{i}\mathbf{q}_{\text{offset}}$$

When $\tilde{\mathrm{B}}$ is multiplied with $\mathbf{q}_{\text{offset}}$ in CPI,
only small number of edges in $\tilde{\mathrm{B}}$ would be visited

Small L1 length of q_{offset} leads to small computation!!

Discussion: Fast Convergence

#modified edges	CPI			OSP			OSP-T			
	$\|q_{CPI}\|_1$	# iter	#visited edges($\times 10^3$)	$\|q_{offset}\|_1$	# iter	#visited edges($\times 10^3$)	$\|q_{offset}\|_1$	# iter	#visited edges($\times 10^3$)	L1 norm error
1	1	116	3, 910, 864	2.60×10^{-9}	2	2, 145	2.60×10^{-9}	1	25	2.84×10^{-8}
10	1	116	3, 910, 863	1.51×10^{-7}	14	405, 717	1.51×10^{-7}	1	147	3.42×10^{-7}
10^2	1	116	3, 910, 858	2.19×10^{-6}	26	839, 137	2.19×10^{-6}	1	788	1.77×10^{-6}
10^3	1	116	3, 910, 808	2.31×10^{-5}	35	1, 169, 546	2.31×10^{-5}	1	4, 098	1.64×10^{-5}
10^4	1	116	3, 910, 300	2.30×10^{-4}	47	1, 604, 965	2.30×10^{-4}	2	44, 960	1.11×10^{-4}
10^5	1	116	3, 905, 224	2.05×10^{-3}	61	2, 104, 446	2.05×10^{-3}	4	130, 470	7.51×10^{-4}

- # edges of LiveJournal dataset: 34,681,189
- OSP and OSP-T visit only small portion of edges in the graph
  - Converge much faster than CPI does

Discussion: Effects of ΔG

- Two factors in ΔG: theoretical analysis in Section 3.3
  - How many nodes are modified?
  - Which nodes are modified?

1. How many nodes are modified?
   - Size of ΔG

     Larger size of ΔG

     => Denser $\Delta\mathbf{A}$

     => Larger L1 length of $\mathbf{q}_{\text{offset}}$

     => Longer computation time for $\mathbf{x}_{\text{offset}}(i)$

$$\mathbf{q}_{\text{offset}} = (1-c)(\Delta\mathbf{A})^\top \mathbf{r}_{\text{old}}$$

$$\mathbf{x}_{\text{offset}}^{(i)} \leftarrow ((1-c)\tilde{\mathbf{B}}^\top)^i \mathbf{q}_{\text{offset}}$$

Discussion: Effects of ΔG

2. Which nodes are modified?

- Location of ΔG => Location of nonzeros in ΔA(= $\tilde{\mathbf{B}}$ - $\tilde{\mathbf{A}}$)

$$\mathbf{q}_{\text{offset}} = (1 - c)(\Delta\mathbf{A})^{\top}\mathbf{r}_{\text{old}}$$

- Nonzeros in ΔA with high RWR nodes in $\mathbf{r}_{\text{old}}$
  - Large $\mathbf{q}_{\text{offset}}$ => Running time skyrockets
- Nonzeros in ΔA with low RWR nodes in $\mathbf{r}_{\text{old}}$
  - Small $\mathbf{q}_{\text{offset}}$ => OSP-T converges quickly
- Real-world graphs follow power-law degree distribution
  - Few nodes having high RWR scores
  - Majority of nodes having low scores

Outline

- Problem Definition
- Proposed Method
- **Experiments**
- Conclusion

Experimental Questions

- Q1. Performance of OSP
- Q2. Performance of OSP-T
- Q3. Effects of ΔG : size and location

Experimental Settings

- Machine: single workstation with 512GB memory
- Datasets: large-scale real-world graph data

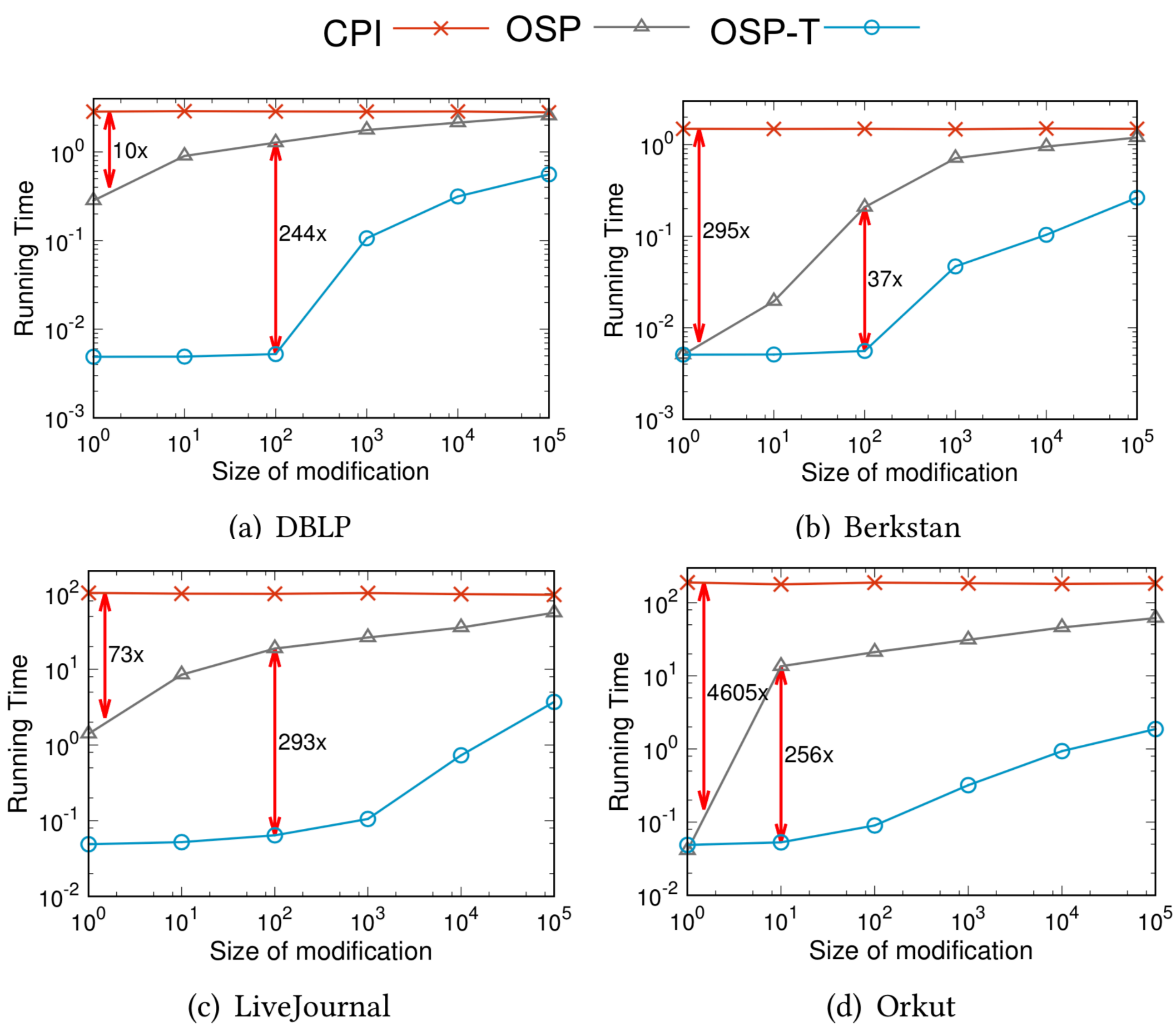
Dataset	Nodes	Edges	Direction	Error tolerance (OSP-T)
WikiLink[1]	12,150,976	378,142,420	Directed	10^{-2}
Orkut[2]	3,072,441	117,185,083	Undirected	5×10^{-3}
LiveJournal[2]	3,997,962	34,681,189	Undirected	5×10^{-3}
Berkstan[2]	685,230	7,600,595	Directed	10^{-4}
DBLP[2]	317,080	1,049,866	Undirected	10^{-4}
Slashdot[2]	82,144	549,202	Directed	10^{-4}

[1] http://konect.uni-koblenz.de/networks/
[2] http://snap.stanford.edu/data/

Q1. Performance of OSP

- How much does OSP improve performance for dynamic RWR computation from baseline static method CPI?
- Running time for tracking RWR exactly on a dynamic graph G varying the size of ΔG
  - Initial graph G with all its edges
  - Modify G by deleting edges.
    - 1 edges to 10^5 edges

Q1. Performance of OSP



(a) DBLP (b) Berkstan

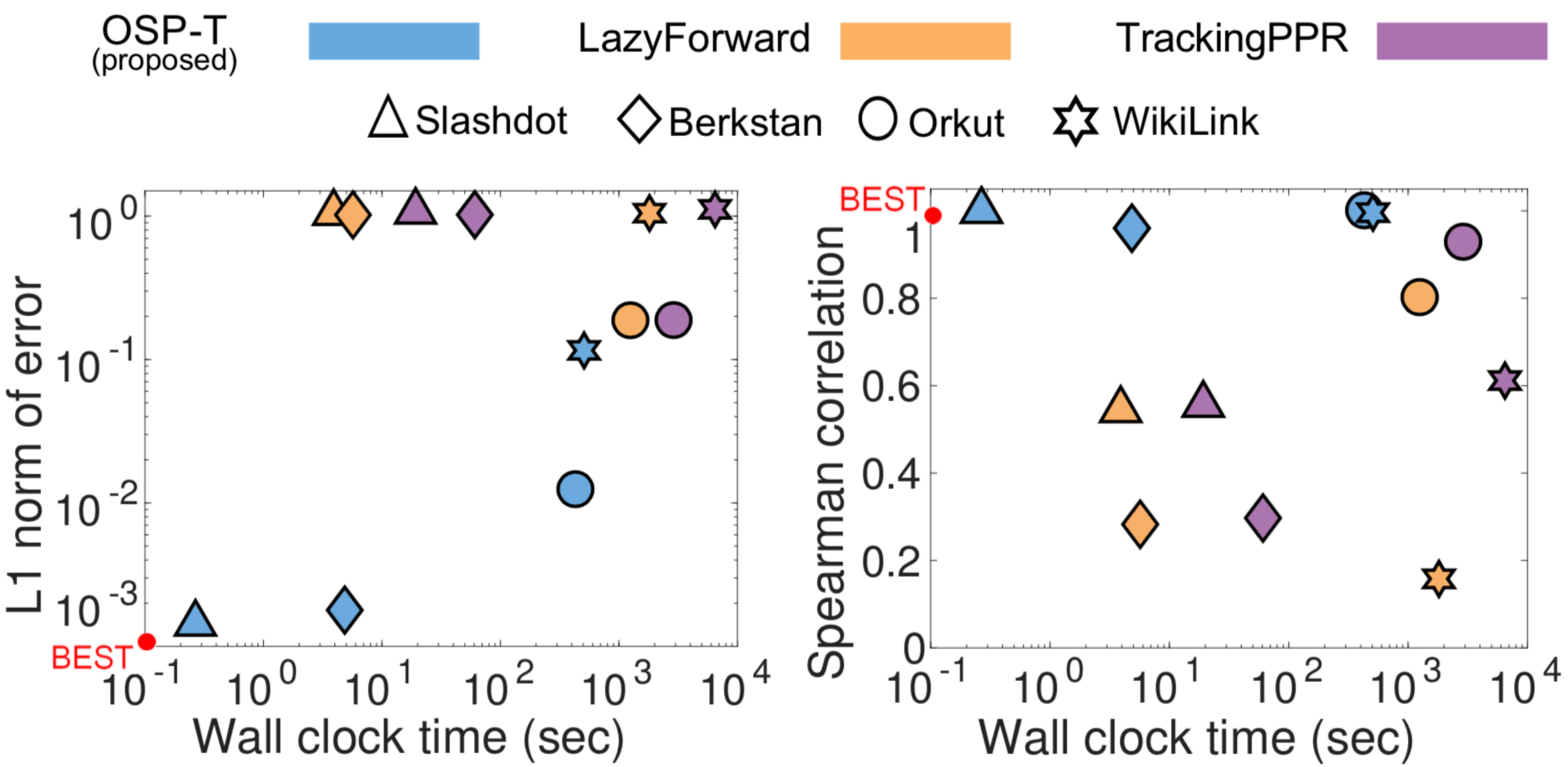
(c) LiveJournal (d) Orkut

Q2. Performance of OSP-T

- How much does OSP-T enhance computation efficiency, accuracy compared with its competitors?
- Experimental setting
  - Generate a uniformly random edge stream and divide the stream into two parts
  - Extract 10 snapshots from the second part
  - Initialize a graph with the first part of the stream
  - Update the graph for each new snapshot arrival
  - At the end of the updates, compare each algorithm.

Q2. Performance of OSP-T

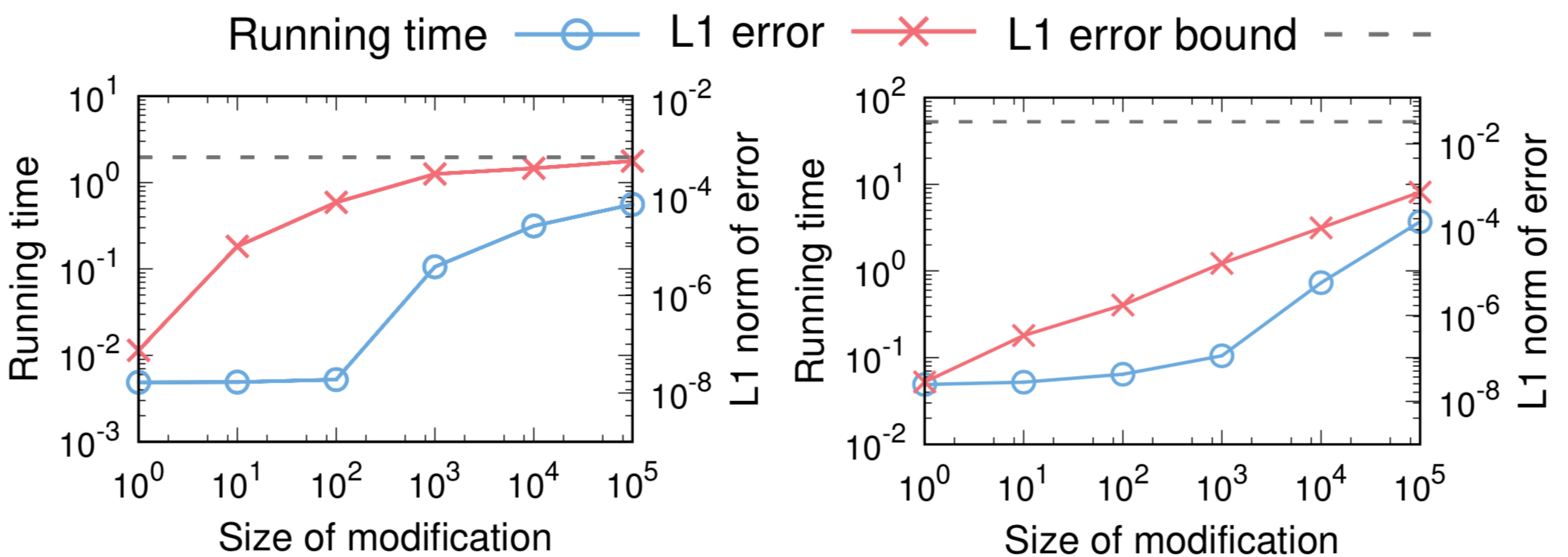
- Trade-off between accuracy and running time



(a) Accuracy on L1 norm of error (b) Accuracy on Rank

Q3. Effects of ΔG - Size

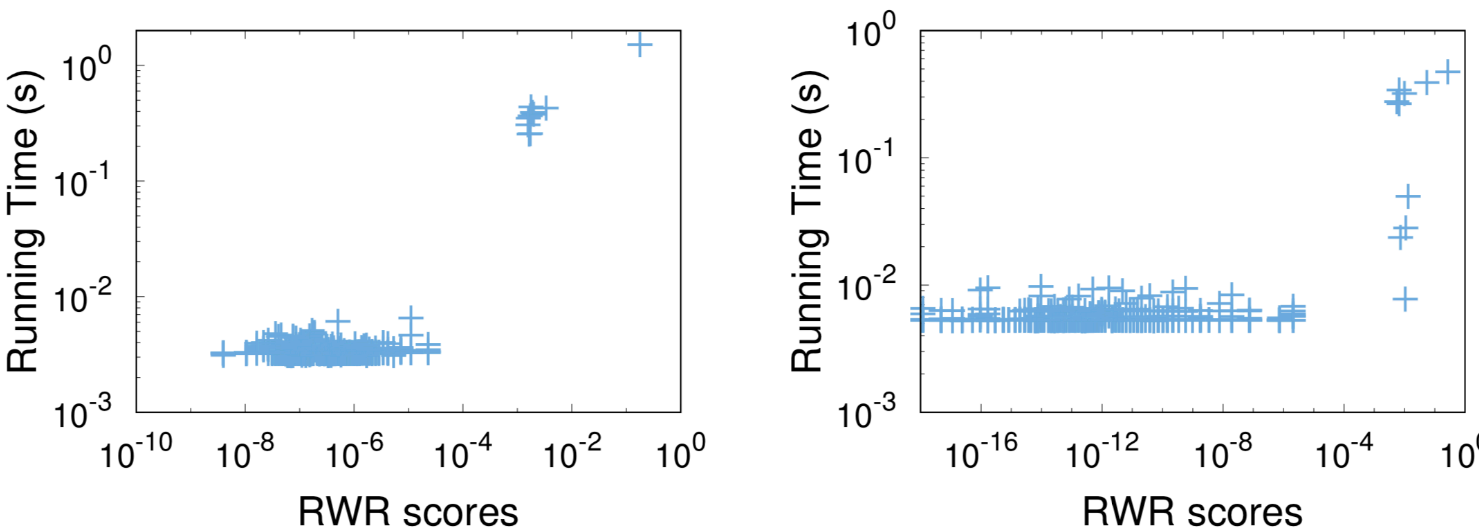
- How does size of ΔG affect the performance of OSP-T?



(a) DBLP (b) LiveJournal

Q3. Effects of ΔG - Location

- Experimental setting
  - Divide nodes evenly into 100 groups in the order of RWR scores.
  - Sample 10 nodes from each group.
  - For each sampled node u, delete an edge (u, v)



(a) DBLP (b) Berkstan

Outline

- Problem Definition
- Proposed Method
- Experiments
- **Conclusion**

Conclusion

- **OSP** (Offset Score Propagation)
  1. Calculate offset scores around the modified edges
  2. Propagate the offset scores across the updated graph
  3. Merge them with previous RWR scores to get updated RWR scores

- Main Results
  - Exactness of OSP
  - Error bound and time complexity of OSP-T
  - Faster and more accurate RWR computation than other methods on Dynamic graphs

Thank you !

Codes & datasets
http://datalab.snu.ac.kr/osp