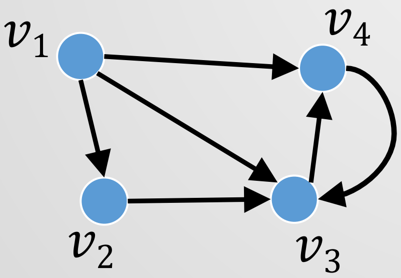
Fast Approximate All Pairwise CoSimRanks via Random Projection

Renchi Yang and Xiaokui Xiao

NUS National University of Singapore

Problem Definition

- Given a graph G with n nodes and m edges
  - A length-i random walk W_u starts from u
    - At each step, navigates to an out-neighbor of current node
    - At i-th step, stops at the current node x
  - $\mathbf{P}^i[u, x]$ is length-i random walk probability from u to x

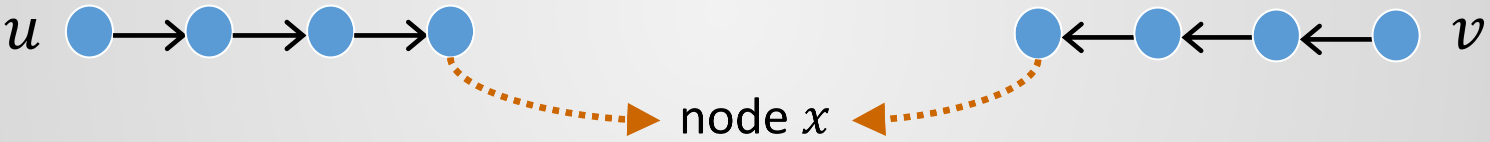


$$
\begin{array}{c}
\begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\
\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 1 \\ 1/3 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$

Random walk matrix $\mathbf{P}$

Problem Definition

- Given a graph G with n nodes and m edges
  - The probability of two length-i random walks W_u, W_v ending at the same node is $\sum_{x \in G} \mathbf{P}^i[u,x] \cdot \mathbf{P}^i[v,x]$



  - The CoSimRank is then
$$s(u,v) = \sum_{i=0}^{\infty} c^i \cdot \mathbf{P}^i[u] \cdot \mathbf{P}^i[v]$$
  - Approximate all pairwise CoSimRank query: for every node pair (u,v)
$$|s(u,v) - s'(u,v)| \leq \epsilon$$

National University of Singapore

Applications

- In natural language processing
  - synonym expansion
  - lexicon extraction
  - linguistically-informed statistical tool in Cistern project
- In knowledge graph mining
  - modelling entity relatedness
- In social network analysis
  - similarity measure of users

NUS National University of Singapore

Existing Solutions

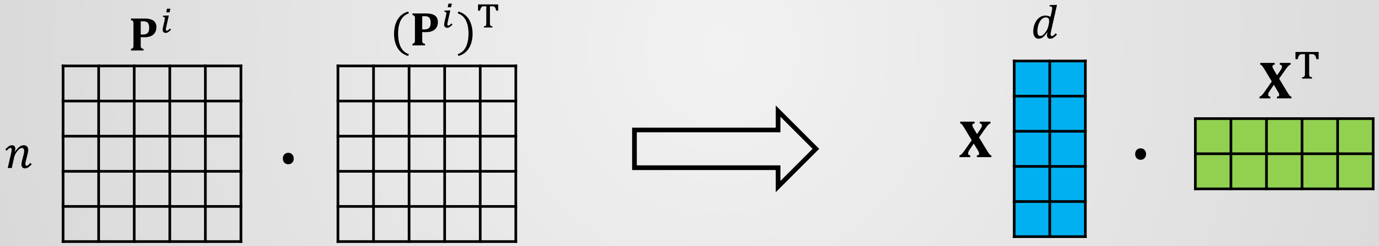
- Matrix form of CoSimRank: $\boldsymbol{S} = \sum_{i=0}^{\infty} c^i \cdot \mathbf{P}^i \cdot (\mathbf{P}^i)^{\mathrm{T}}$
- PowerMethod
  - solves the equation by iterative matrix multiplications
  - Time complexity: $O(n^3 \cdot \ln(1/\epsilon))$
- Co-Simmate
  - reuses the results from previous iterations to reduce repeated operations
  - Time complexity: $O(n^3 \cdot \log(\ln\left(\frac{1}{\epsilon}\right)))$
- F-CoSim is designed for dynamic graphs
  - Time complexity: $O(n^3 \cdot \ln(1/\epsilon))$

NUS
National University of Singapore

Our Solution

- Matrix form of CoSimRank:

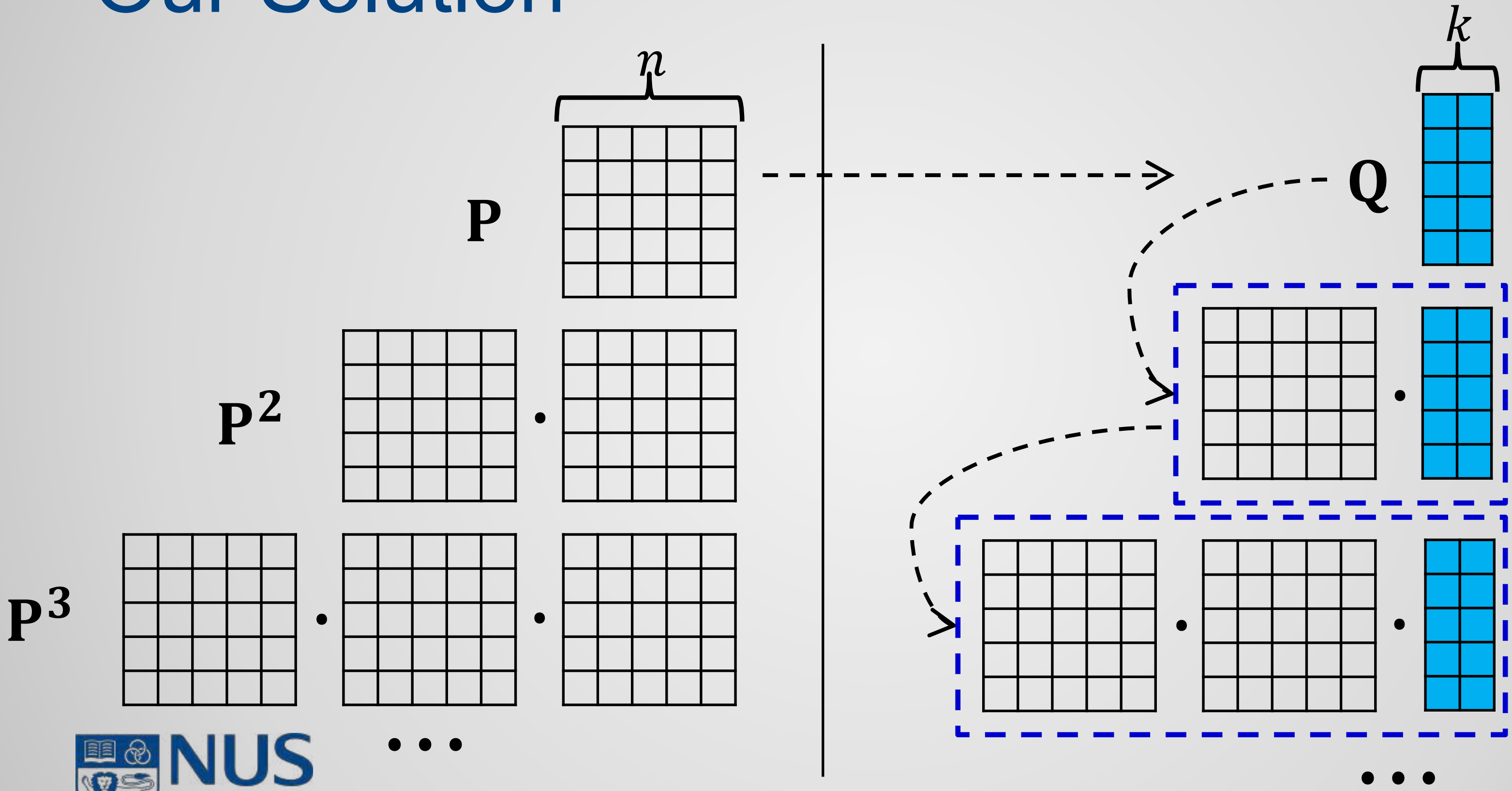
$$\mathbf{S} = \sum_{i=0}^{\infty} c^i \cdot \mathbf{P}^i \cdot (\mathbf{P}^i)^{\mathrm{T}}$$



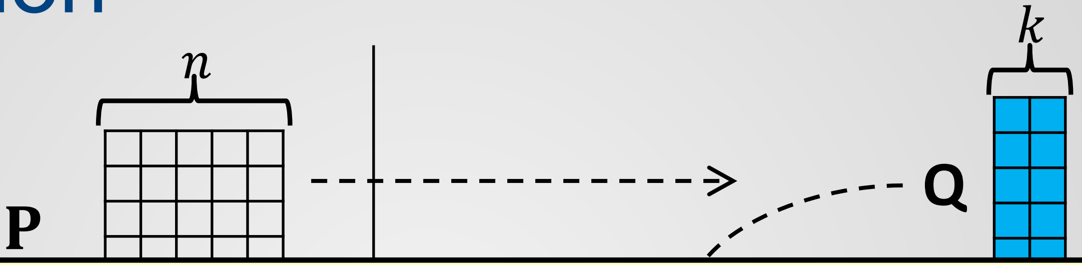
- Time complexity: from $O(n^3)$ to $O(n^2 d)$
- Dimensionality reduction over $\mathbf{P}^i$ is computationally costly

NUS National University of Singapore

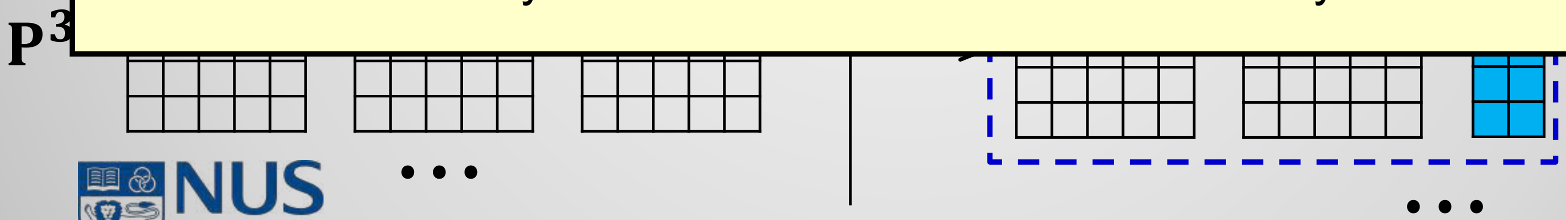
Our Solution



Our Solution



- Summary:
  - We do not need to compute $\mathbf{P}^i$ and its low-dimensional approximation with paying $O(n^3)$ cost
  - For all $\mathbf{P}^i$, we can just find the low-dimensional approximation $\mathbf{Q}$ of $\mathbf{P}$
  - Dimensionality reduction over $\mathbf{P}$ is fast as it has only m entries



NUS National University of Singapore

Our solution: Random Projection

- Johnson–Lindenstrauss lemma

Lemma 2 ((Preservation of inner products [10]). *Let $\delta, p_f \in (0,1)$ and $d \geq \frac{2\ln(1/p'_f)}{\delta - \ln(1+\delta)}$. Let $\mathbf{T}$ be an $n \times d$ matrix, where each entry is sampled i.i.d. from a Gaussian $\mathcal{N}(0,1)$. Given any two vectors $\mathbf{z}_i, \mathbf{z}_i \in \mathbb{R}^n$, we define $\mathbf{x}_i = \frac{1}{\sqrt{d}} \cdot \mathbf{z}_i\mathbf{T}$, $\mathbf{x}_j = \frac{1}{\sqrt{d}} \cdot \mathbf{z}_j\mathbf{T}$. Then, we have*

$$\mathbb{P}\left[\left|\mathbf{x}_i \cdot \mathbf{x}_j^\top - \mathbf{z}_i \cdot \mathbf{z}_j^\top\right| \leq \delta \cdot ||\mathbf{z}_i|| \cdot ||\mathbf{z}_j||\right] \geq 1 - p'_f. \quad (6)$$

- Apply random projection to $\mathbf{P}$
  - Find an approximate dimension d to ensure ϵ absolute error
  - Generate a random matrix $\mathbf{T} \in \mathbb{R}^{n \times d}$ and compute $\mathbf{Q} = \frac{1}{\sqrt{d}} \cdot \mathbf{PT}$

NUS National University of Singapore

Our Solution: RPCS

Algorithm 1: RPCS

Input: An input graph G, c, ϵ, p_f, δ.
Output: $\widehat{\mathbf{S}}$.

$t \leftarrow \left\lceil \frac{\ln\left(1-\frac{c-(1-c)\epsilon}{c(1-\delta)}\right)}{\ln(c)} \right\rceil$;
$d \leftarrow \left\lceil \frac{2\ln\left(\frac{n^2}{2p_f}\right)}{\delta-\ln(1+\delta)} \right\rceil$;
if $d \geq n$ **then**
  $\mathbf{Q} \leftarrow \mathbf{P}$
else
  Generate $\mathbf{T} \in \mathbb{R}^{n\times d} \sim \mathcal{N}(0,1)$; ▷ $O(nd)$ time
  $\mathbf{Q} \leftarrow \frac{1}{\sqrt{d}} \cdot \mathbf{PT}$; ▷ $O(md)$ time
$\mathbf{H}^{(1)} \leftarrow \sqrt{c}\cdot \mathbf{Q}$; $\widehat{\mathbf{S}} \leftarrow \mathbf{I} + \mathbf{H}^{(1)} \cdot \mathbf{H}^{(1)\top}$; ▷ $O(n^2 d)$ time
for $k \leftarrow 2$ *to* t **do**
  $\mathbf{H}^{(k)} \leftarrow \sqrt{c}\mathbf{P} \cdot \mathbf{H}^{(k-1)}$; ▷ $O(md)$ time
  $\widehat{\mathbf{S}} \leftarrow \widehat{\mathbf{S}} + \mathbf{H}^{(k)} \cdot \mathbf{H}^{(k)\top}$; ▷ $O(n^2 d)$ time
return $\widehat{\mathbf{S}}$;

- If the cost using random projection exceeds that of PowerMethod, switch to PowerMethod
- Running time $\propto \frac{\ln\left(\frac{c(1-\delta)}{(1-c)\epsilon - c\delta}\right)}{\delta - \ln(1+\delta)}$
- $0 < \delta < \frac{1-c}{c} \cdot \epsilon$

Our Solution: RPCS

Algorithm 2: TernarySearch

Input: c, ϵ.
Output: δ.

$\delta_l \leftarrow 0,\ \delta_u \leftarrow \frac{1-c}{c} \cdot \epsilon$;
while true **do**
    $\delta_l' \leftarrow \delta_l + \frac{\delta_u - \delta_l}{3}$;
    $\delta_u' \leftarrow \delta_u - \frac{\delta_u - \delta_l}{3}$;
    if $\delta_u' \leq \delta_l'$ *or* $\delta_u - \delta_l \leq \frac{1-c}{1000c} \cdot \epsilon$ **then break**;
    if $f(\delta_l') < f(\delta_u')$ **then**
        $\delta_l \leftarrow \delta_l'$;
    else
        $\delta_u \leftarrow \delta_u'$;
$\delta \leftarrow \frac{\delta_l + \delta_u}{2}$;
return δ;

- Optimize

$$\min_{<\delta<\frac{1-c}{c}\cdot\epsilon} \frac{\ln\left(\frac{c(1-\delta)}{(1-c)\epsilon - c\delta}\right)}{\delta - \ln(1+\delta)}$$

- Convex function
- Search the minimizer by ternary search algorithm

National University of Singapore

Our Solution: RPCS

Name	Accuracy	Time Complexity
PowerMethod [19]	$\lvert s(v_i, v_j) - \hat{s}(v_i, v_j)\rvert \leq \epsilon,\ \forall v_i, v_j \in V$	$O\left(n^3 \ln(\frac{1}{\epsilon})\right)$
Co-Simmate [31]	$\lvert s(v_i, v_j) - \hat{s}(v_i, v_j)\rvert \leq \epsilon,\ \forall v_i, v_j \in V$	$O\left(n^3 \log_2(\ln(\frac{1}{\epsilon}))\right)$
F-CoSim [32]	$\lvert s(v_i, v_j) - \hat{s}(v_i, v_j)\rvert \leq \epsilon,\ \forall v_i, v_j \in V$	$O\left(n^3 \ln(\frac{1}{\epsilon})\right)$
RPCS	$\mathbb{P}\left[\lvert s(v_i, v_j) - \hat{s}(v_i, v_j)\rvert \leq \epsilon,\ \forall v_i, v_j \in V\right] \geq 1 - \frac{1}{n}$	$O\left(\min\left\{\frac{n^2 \ln(n)}{\epsilon^2} \cdot \ln(\frac{1}{\epsilon}), n^3 \ln(\frac{1}{\epsilon})\right\}\right)$

NUS National University of Singapore

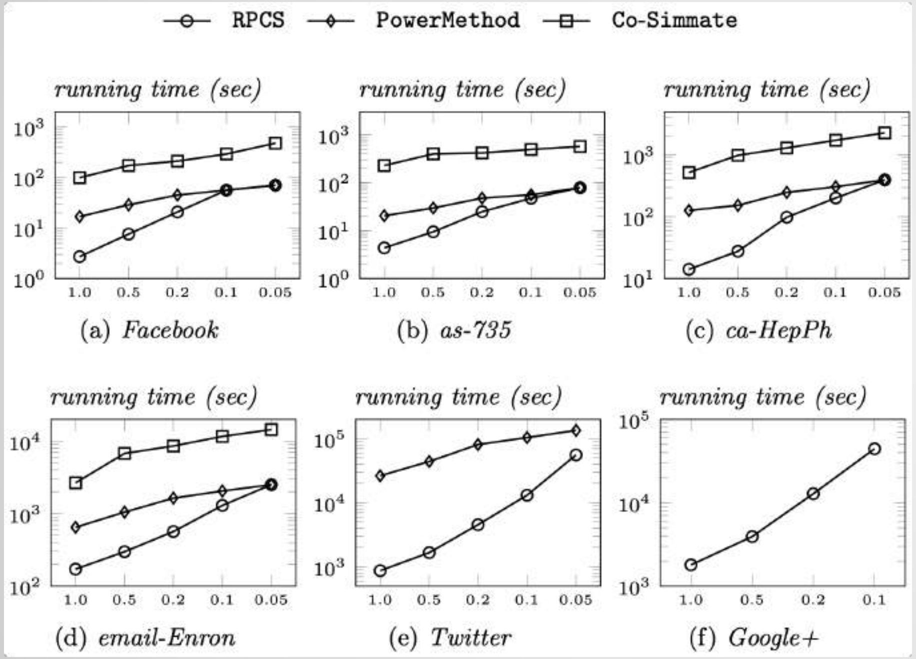
Experimental Settings

Name	#Nodes (n)	#Edges (m)	Type
Facebook	4,039	88,234	undirected
as-735	7,716	26,467	undirected
ca-HepPh	12,008	237,010	undirected
email-Enron	36,692	183,831	directed
Twitter	81,306	1,768,149	directed
Google+	107,614	13,673,453	directed

- Compepititors:
  - PowerMethod,
  - Co-Simmate
- Damping factor $c = 0.8$
- Varying error threshold ϵ in {1.0, 0.5, 0.2, 0.1, 0.05}
- Intel Xeon 2.60GHz CPU
- 377GB RAM

NUS National University of Singapore

Experimental Results



- On small graphs, RPCS is 2-9× faster when $\epsilon > 0.1$
- On Twitter, RPCS is by up to three orders of magnitude faster
- On Google+, RPCS is the only viable solution
- Omited if cannot terminate within 2 days or is OOM

THANK YOU

National University of Singapore