SIGIR 2012

SimFusion+: Extending SimFusion Towards Efficient Estimation on Large and Dynamic Networks

Weiren Yu[1], Xuemin Lin[1], Wenjie Zhang[1], Ying Zhang[1] Jiajin Le[2],

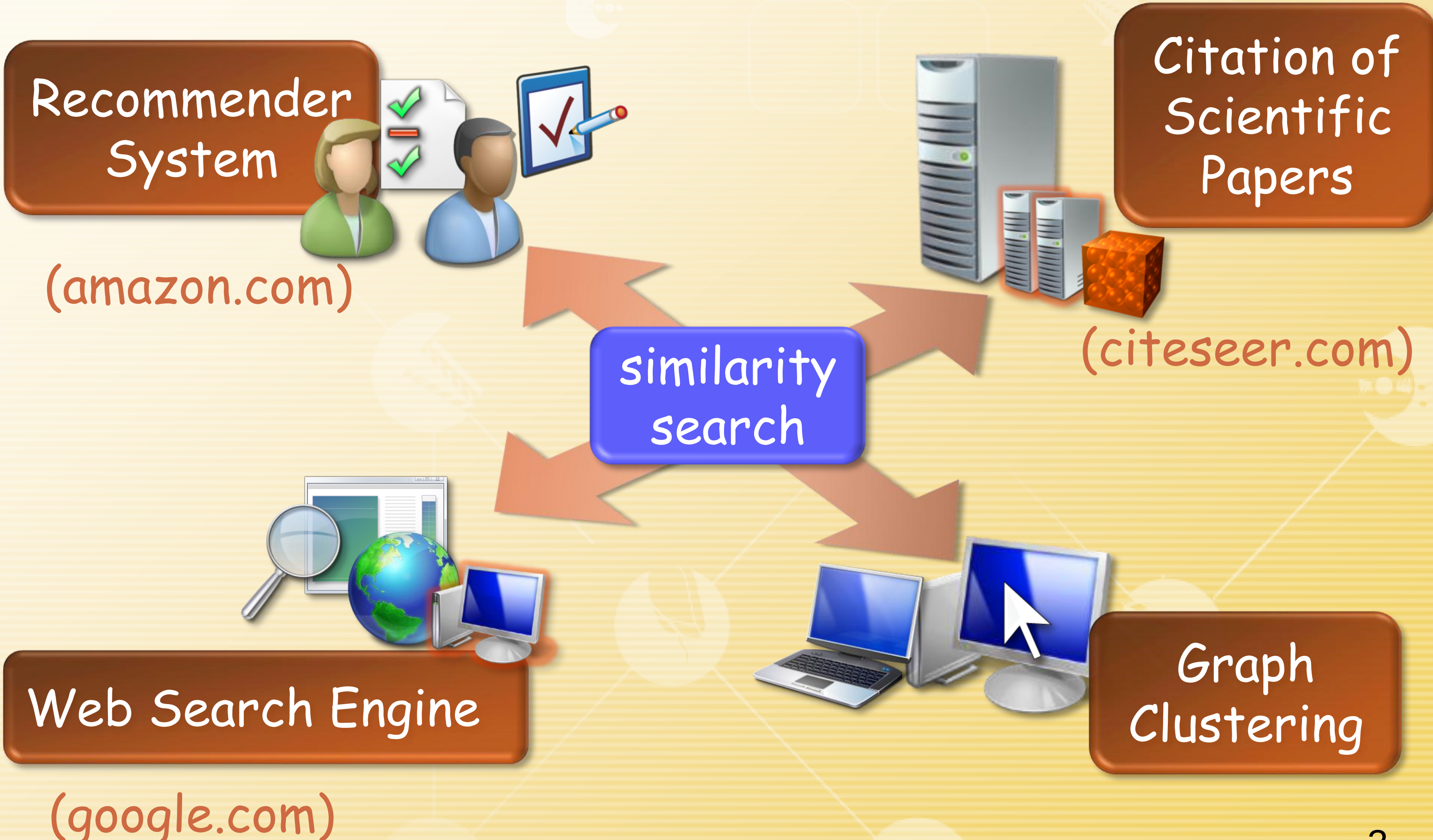
[1] University of New South Wales & NICTA, Australia

[2] Donghua University, China

Contents

1. Introduction

- Many applications require a measure of “similarity” between objects.



SimFusion: A New Link-based Similarity Measure

- Structural Similarity Measure
  - PageRank [Page et. al, 99]
  - SimRank [Jeh and Widom, KDD 02]
- SimFusion similarity
  - A new promising structural measure [Xi et. al, SIGIR 05]
  - Extension of Co-Citation and Coupling metrics
- Basic Philosophy
  - Following the Reinforcement Assumption:

    The similarity between objects is reinforced by the similarity of their related objects.

SimFusion Overview

- Features
  - Using a *Unified Relationship Matrix* (URM) to represent relationships among heterogeneous data
  - Defined recursively and is computed iteratively
  - Applicable to any domain with object-to-object relationships

- Challenges
  - URM may incur trivial solution or divergence issue of SimFusion.
  - Rather costly to compute SimFusion on large graphs
    - Naïve Iteration: matrix-matrix multiplication
    - Requiring $O(Kn^3)$ time, $O(n^2)$ space [Xi et. al. , SIGIR 05]
  - No incremental algorithms when edges update

Existing SimFusion: URM and USM

- *Data Space*: $\mathcal{D}=\{o_1,o_2,\cdots\}$ a finite set of data objects (vertices)
- *Data Relation* (edges) Given an entire space $\mathcal{D}=\bigcup_{i=1}^{N}\mathcal{D}_i$
  - Intra-type Relation $\mathcal{R}_{i,i}\subseteq\mathcal{D}_i\times\mathcal{D}_i$ carrying info. within one space
  - Inter-type Relation $\mathcal{R}_{i,j}\subseteq\mathcal{D}_i\times\mathcal{D}_j$ carrying info. between spaces
- *Unified Relationship Matrix* (URM):

$$\mathbf{L}_{\mathrm{URM}}=\begin{pmatrix}\lambda_{1,1}\mathbf{L}_{1,1} & \lambda_{1,2}\mathbf{L}_{1,2} & \cdots & \lambda_{1,N}\mathbf{L}_{1,N}\\ \lambda_{2,1}\mathbf{L}_{2,1} & \lambda_{2,2}\mathbf{L}_{2,2} & \cdots & \lambda_{2,N}\mathbf{L}_{2,N}\\ \vdots & \vdots & \ddots & \vdots\\ \lambda_{N,1}\mathbf{L}_{N,1} & \lambda_{N,2}\mathbf{L}_{N,2} & \cdots & \lambda_{N,N}\mathbf{L}_{N,N}\end{pmatrix}
\qquad
\mathbf{L}_{i,j}(x,y)=\begin{cases}\frac{1}{n_j}, & \text{if } \mathcal{N}_j(x)=\varnothing;\\ \frac{1}{|\mathcal{N}_j(x)|}, & \text{if } (x,y)\in\mathcal{R}_{i,j};\\ 0, & \text{otherwise.}\end{cases}$$

  - $\lambda_{i,j}$ is the *weighting factor* between D_i and D_j
- *Unified Similarity Matrix* (USM):

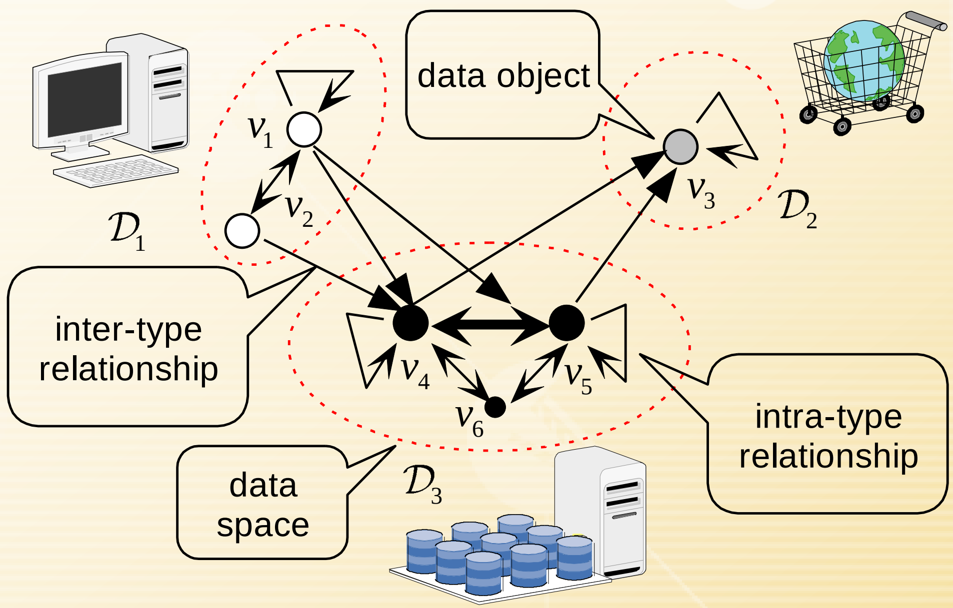
$$\exists\,\mathbf{S}=\begin{pmatrix}s_{1,1} & \cdots & s_{1,n}\\ \vdots & \ddots & \vdots\\ s_{n,1} & \cdots & s_{n,n}\end{pmatrix}\in\mathbb{R}^{n\times n}\quad s.t.\quad \mathbf{S}=\mathbf{L}\cdot\mathbf{S}\cdot\mathbf{L}^T.$$

Example.

Trivial Solution !!!

S=$[1]_{nxn}$

SimFusion Similarity on Heterogeneous Domain



$$\mathbf{L}_{\mathrm{URM}} = \left(\begin{array}{cc|c|ccc} \frac{1}{8} & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ \hline \frac{1}{16} & \frac{1}{16} & \frac{1}{4} & \frac{5}{24} & \frac{5}{24} & \frac{5}{24} \\ \hline \frac{1}{10} & \frac{1}{10} & \frac{3}{5} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} \\ \frac{1}{10} & \frac{1}{10} & \frac{3}{5} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} \\ \frac{1}{10} & \frac{1}{10} & \frac{3}{5} & \frac{1}{10} & \frac{1}{10} & 0 \end{array}\right)$$

$$\exists \mathbf{S} \in \mathbb{R}^{n\times n} \quad s.t. \quad \mathbf{S} = \mathbf{L}\cdot\mathbf{S}\cdot\mathbf{L}^T.$$

$$\mathbf{\Lambda} = \begin{array}{c|c} & \begin{array}{ccc} \mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 \end{array} \\ \begin{array}{c} \mathcal{D}_1 \\ \mathcal{D}_2 \\ \mathcal{D}_3 \end{array} & \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{8} & \frac{1}{4} & \frac{5}{8} \\ \frac{1}{5} & \frac{3}{5} & \frac{1}{5} \end{pmatrix} \end{array}$$

$$\mathbf{S}_{\mathrm{USM}} = \begin{pmatrix} 1 & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & 1 & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & 1 \end{pmatrix}$$

High complexity !!!

O(Kn^3) time

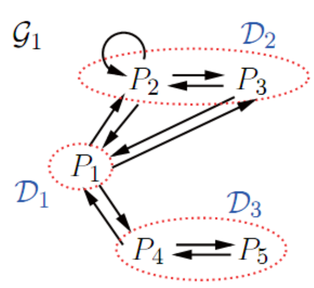
O(n^2) space

Contributions

- Revising the existing SimFusion model, avoiding
  - non-semantic convergence
  - divergence issue
- Optimizing the computation of SimFusion+
  - O(Km) pre-computation time, plus O(1) time and O(n) space
  - Better accuracy guarantee
- Incremental computation on edge updates
  - O(δn) time and O(n) space for handling δ edge updates

Revised SimFusion

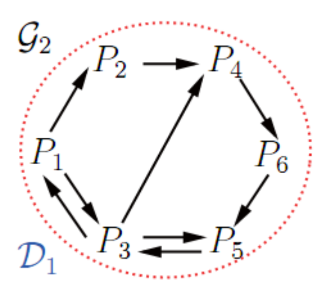
- Motivation: Two issues of the existing SimFusion model
  - Trivial Solution on Heterogeneous Domain



$$\lim_{k\to\infty}\mathbf{SimFusion}^{(k)} = \begin{array}{c c} \begin{array}{r} \mathcal{D}_1\{ \\ \mathcal{D}_2\{ \\ \\ \mathcal{D}_3\{ \\ \\ \end{array} & \begin{bmatrix} 1 & .219 & .219 & .219 & .219 \\ .219 & 1 & .219 & .219 & .219 \\ .219 & .219 & 1 & .219 & .219 \\ .219 & .219 & .219 & 1 & .219 \\ .219 & .219 & .219 & .219 & 1 \end{bmatrix} \\ & \underbrace{\quad}_{\mathcal{D}_1}\ \underbrace{\qquad\quad}_{\mathcal{D}_2}\ \underbrace{\qquad\quad}_{\mathcal{D}_3} \end{array}$$

$$\lim_{k\to\infty}\mathbf{SimFusion+}^{(k)} = \begin{array}{c} \begin{bmatrix} 1 & .290 & .194 & .139 & .100 \\ .290 & 1 & .304 & .217 & .156 \\ .194 & .304 & 1 & .145 & .105 \\ .139 & .217 & .145 & 1 & .075 \\ .100 & .156 & .105 & .075 & 1 \end{bmatrix} \\ \underbrace{\quad}_{\mathcal{D}_1}\ \underbrace{\qquad\quad}_{\mathcal{D}_2}\ \underbrace{\qquad\quad}_{\mathcal{D}_3} \end{array}$$

  - Divergent Solution on Homogeneous Domain



$$\lim_{k\to\infty}\mathbf{SimFusion}^{(2k)} = \begin{bmatrix} .46 & 0 & .46 & 0 & 0 & .46 \\ 0 & .38 & 0 & .38 & .38 & 0 \\ .46 & 0 & .46 & 0 & 0 & .46 \\ 0 & .38 & 0 & .38 & .38 & 0 \\ 0 & .38 & 0 & .38 & .38 & 0 \\ .46 & 0 & .46 & 0 & 0 & .46 \end{bmatrix} \neq \lim_{k\to\infty}\mathbf{SimFusion}^{(2k+1)} = \begin{bmatrix} .38 & 0 & .38 & 0 & 0 & .38 \\ 0 & .46 & 0 & .46 & .46 & 0 \\ .38 & 0 & .38 & 0 & 0 & .38 \\ 0 & .46 & 0 & .46 & .46 & 0 \\ 0 & .46 & 0 & .46 & .46 & 0 \\ .38 & 0 & .38 & 0 & 0 & .38 \end{bmatrix}$$

$$\lim_{k\to\infty}\mathbf{SimFusion+}^{(k)} = \begin{bmatrix} 1 & 0 & .25 & 0 & 0 & .25 \\ 0 & 1 & 0 & .25 & .18 & 0 \\ .25 & 0 & 1 & 0 & 0 & .18 \\ 0 & .25 & 0 & 1 & .25 & 0 \\ 0 & .18 & 0 & .25 & 1 & 0 \\ .25 & 0 & .18 & 0 & 0 & 1 \end{bmatrix}$$

Root cause: row normalization of URM !!!

From URM to UAM

- Unified Adjacency Matrix (UAM) $\mathbf{A} = \tilde{\mathbf{A}} + 1/n^2$

$$\tilde{\mathbf{A}} = \begin{pmatrix} \lambda_{1,1}\mathbf{A}_{1,1} & \lambda_{1,2}\mathbf{A}_{1,2} & \cdots & \lambda_{1,N}\mathbf{A}_{1,N} \\ \lambda_{2,1}\mathbf{A}_{2,1} & \lambda_{2,2}\mathbf{A}_{2,2} & \cdots & \lambda_{2,N}\mathbf{A}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N,1}\mathbf{A}_{N,1} & \lambda_{N,2}\mathbf{A}_{N,2} & \cdots & \lambda_{N,N}\mathbf{A}_{N,N} \end{pmatrix}, \quad \mathbf{A}_{i,j}(x,y) = \begin{cases} \frac{1}{n_j}, & \text{if } \mathcal{N}_j(x) = \varnothing; \\ {\color{red}1,} & \text{if } (x,y) \in \mathcal{R}_{i,j}; \\ 0, & \text{otherwise.} \end{cases}$$

- Example

$$\mathbf{\Lambda} = \begin{array}{c c} & \begin{array}{ccc} \mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 \end{array} \\ \begin{array}{c} \mathcal{D}_1 \\ \mathcal{D}_2 \\ \mathcal{D}_3 \end{array} & \begin{bmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{3} \\ \frac{1}{6} & \frac{7}{12} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{5}{12} \end{bmatrix} \end{array} \Rightarrow \tilde{\mathbf{A}} = \begin{bmatrix} \frac{1}{2}\begin{bmatrix}1\end{bmatrix} & \frac{1}{6}\begin{bmatrix}1 & 1\end{bmatrix} & \frac{1}{3}\begin{bmatrix}1 & 0\end{bmatrix} \\ \frac{1}{6}\begin{bmatrix}1 \\ 1\end{bmatrix} & \frac{7}{12}\begin{bmatrix}1 & 1 \\ 1 & 0\end{bmatrix} & \frac{1}{4}\begin{bmatrix}\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2}\end{bmatrix} \\ \frac{1}{3}\begin{bmatrix}1 \\ 0\end{bmatrix} & \frac{1}{4}\begin{bmatrix}\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2}\end{bmatrix} & \frac{5}{12}\begin{bmatrix}0 & 1 \\ 1 & 0\end{bmatrix} \end{bmatrix} = \left[\begin{array}{c|cc|cc} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & 0 \\ \hline \frac{1}{6} & \frac{7}{12} & \frac{7}{12} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{6} & \frac{7}{12} & 0 & \frac{1}{8} & \frac{1}{8} \\ \hline \frac{1}{3} & \frac{1}{8} & \frac{1}{8} & 0 & \frac{5}{12} \\ 0 & \frac{1}{8} & \frac{1}{8} & \frac{5}{12} & 0 \end{array}\right]$$

Revised SimFusion+

- Basic Intuition
  - replace URM with UAM to postpone “row normalization” in a delayed fashion while preserving the reinforcement assumption of the original SimFusion
- Revised SimFusion+ Model

Original SimFusion

$$\mathbf{S} = \frac{\mathbf{A} \cdot \mathbf{S} \cdot \mathbf{A}^T}{\boxed{\|\mathbf{A} \cdot \mathbf{S} \cdot \mathbf{A}^T\|_2}},$$

$$\mathbf{S} = \mathbf{L} \cdot \mathbf{S} \cdot \mathbf{L}^T$$

squeeze similarity scores in S into [0, 1].

Optimizing SimFusion+ Computation

- Conventional Iterative Paradigm

$$\mathbf{S}^{(k+1)} = \frac{\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T}{\|\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T\|_2}.$$

  - Matrix-matrix multiplication, requiring $O(kn^3)$ time and $O(n^2)$ space

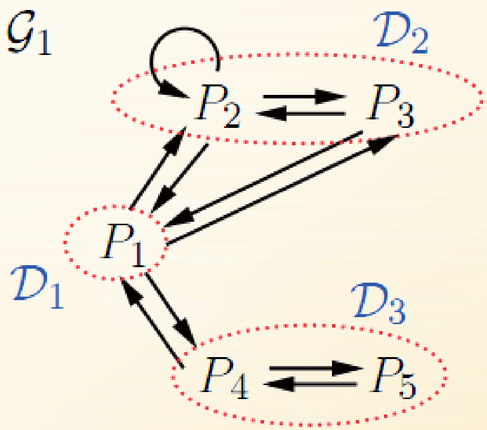
- Our approach: To convert SimFusion+ computation into finding the dominant eigenvector of the UAM A.

$$[\mathbf{S}]_{i,j} = [\sigma_{\max}(\mathbf{A})]_i \times \boxed{[\sigma_{\max}(\mathbf{A})]_j}$$

Pre-compute σ_{max}(A) only once, and cache it for later reuse

  - Matrix-vector multiplication, requiring $O(km)$ time and $O(n)$ space

Example



Assume $\mathbf{A} = \tilde{\mathbf{A}} + \mathbf{1}/5^2$ with

$$\tilde{\mathbf{A}} = \begin{bmatrix} \frac{1}{2}\begin{bmatrix}1\end{bmatrix} & \frac{1}{6}\begin{bmatrix}1 & 1\end{bmatrix} & \frac{1}{3}\begin{bmatrix}1 & 0\end{bmatrix} \\ \frac{1}{6}\begin{bmatrix}1\\1\end{bmatrix} & \frac{7}{12}\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix} & \frac{1}{4}\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix} \\ \frac{1}{3}\begin{bmatrix}1\\0\end{bmatrix} & \frac{1}{4}\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix} & \frac{5}{12}\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} \end{bmatrix} = \left[\begin{array}{c|cc|cc} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & 0 \\ \hline \frac{1}{6} & \frac{7}{12} & \frac{7}{12} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{6} & \frac{7}{12} & 0 & \frac{1}{8} & \frac{1}{8} \\ \hline \frac{1}{3} & \frac{1}{8} & \frac{1}{8} & 0 & \frac{5}{12} \\ 0 & \frac{1}{8} & \frac{1}{8} & \frac{5}{12} & 0 \end{array}\right]$$

- Conventional Iteration:

$$\mathbf{S}^{(0)} = \mathbf{1} \qquad \mathbf{S}^{(k+1)} = \frac{\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T}{\|\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T\|_2}.$$

$$\mathbf{S} = \begin{bmatrix} .186 & .290 & .194 & .139 & .100 \\ .290 & .453 & .304 & .217 & .156 \\ .194 & .304 & .203 & .145 & .105 \\ .139 & .217 & .145 & .104 & .075 \\ .100 & .156 & .105 & .075 & .054 \end{bmatrix}$$

- Our approach:

$$\sigma_{\max}(\mathbf{A}) = [.431 \quad .673 \quad .451 \quad .322 \quad .232]^T$$

$$[\mathbf{S}]_{1,2} = [\sigma_{\max}(\mathbf{A})]_1 \times [\sigma_{\max}(\mathbf{A})]_2 = .431 \times .673 = .290.$$

$$[\mathbf{S}]_{1,3} = [\sigma_{\max}(\mathbf{A})]_1 \times [\sigma_{\max}(\mathbf{A})]_3 = .431 \times .451 = .194.$$

Key Observation

- Kroneckor product "⊗":

$$\mathbf{X} \otimes \mathbf{Y} \stackrel{\text{def}}{=} \begin{bmatrix} x_{1,1}\mathbf{Y} & \cdots & x_{1,q}\mathbf{Y} \\ \vdots & \ddots & \vdots \\ x_{p,1}\mathbf{Y} & \cdots & x_{p,q}\mathbf{Y} \end{bmatrix}$$

e.g.

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}, \quad \mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} 1\times\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 2\times\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ 3\times\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 4\times\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{bmatrix} = \left[\begin{array}{cc|cc} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ \hline 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{array}\right]$$

- Vec operator:

$$vec(\mathbf{X}) \stackrel{\text{def}}{=} [x_{1,1}, \cdots, x_{p,1}, \cdots, x_{1,q}, \cdots, x_{p,q}]^T$$

e.g. $vec(\mathbf{X}) = [1 \quad 3 \quad 2 \quad 4]^T$

- Two important Properties:

$$vec(\mathbf{BCD}^T) = (\mathbf{D} \otimes \mathbf{B}) \cdot vec(\mathbf{C})$$

$$\sigma_{\max}(\mathbf{A} \otimes \mathbf{A}) = \sigma_{\max}(\mathbf{A}) \otimes \sigma_{\max}(\mathbf{A})$$

Key Observation

- Two important Properties:

P1. $vec(\mathbf{BCD}^T) = (\mathbf{D} \otimes \mathbf{B}) \cdot vec(\mathbf{C})$

P2. $\sigma_{\max}(\mathbf{A} \otimes \mathbf{A}) = \sigma_{\max}(\mathbf{A}) \otimes \sigma_{\max}(\mathbf{A})$

- Our main idea:

$$\mathbf{S}^{(k+1)} = \frac{\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T}{\|\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T\|_2}.$$

(1) $\Rightarrow$

$$vec(\mathbf{S}^{(k+1)}) = \frac{(\mathbf{A} \otimes \mathbf{A}) \cdot vec(\mathbf{S}^{(k)})}{\|(\mathbf{A} \otimes \mathbf{A}) \cdot vec(\mathbf{S}^{(k)})\|_2}$$

$\Downarrow$ Power Iteration

$$\lim_{k \to \infty} vec(\mathbf{S}^{(k)}) = \sigma_{\max}(\mathbf{A} \otimes \mathbf{A})$$

(2) $\Leftarrow$

$$vec(\mathbf{S}) = \sigma_{\max}(\mathbf{A}) \otimes \sigma_{\max}(\mathbf{A})$$

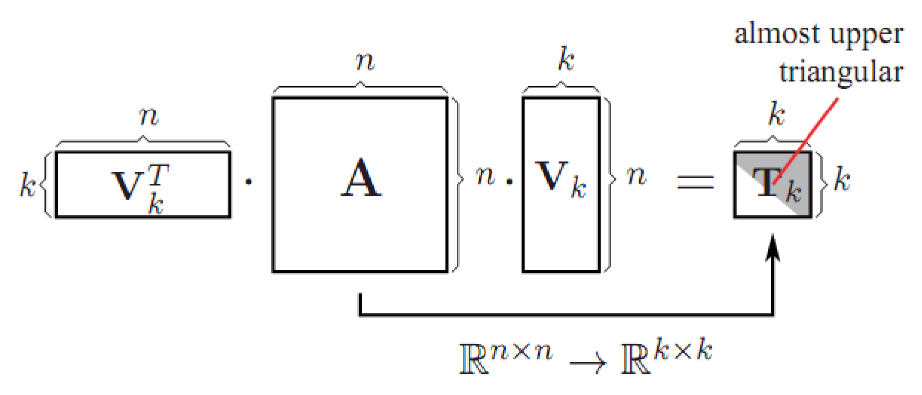
Accuracy Guarantee

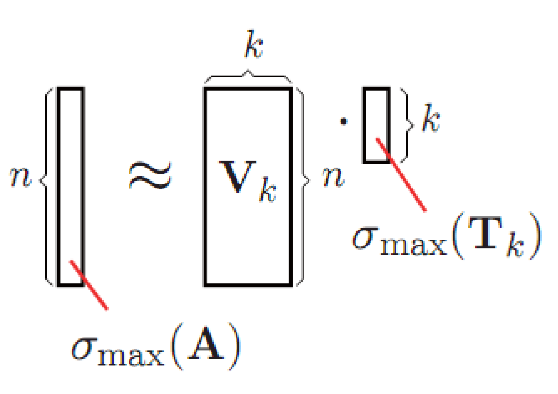
- Conventional Iterations: No accuracy guarantee !!!

$$\mathbf{S}^{(k+1)} = \frac{\mathbf{A}\cdot\mathbf{S}^{(k)}\cdot\mathbf{A}^T}{\|\mathbf{A}\cdot\mathbf{S}^{(k)}\cdot\mathbf{A}^T\|_2}. \qquad \mathbf{S} = \frac{\mathbf{A}\cdot\mathbf{S}\cdot\mathbf{A}^T}{\|\mathbf{A}\cdot\mathbf{S}\cdot\mathbf{A}^T\|_2}$$

Question: $\| S^{(k+1)} - S \| \le ?$

- Our Method: Utilize Arnoldi decomposition to build an order-k orthogonal subspace for the UAM A.





Due to T_k small size and almost "upper-triangularity",
Computing $\sigma_{max}(T_k)$ is less costly than $\sigma_{max}(A)$.

Accuracy Guarantee

- Arnoldi Decomposition:

$$\mathbf{V}_k^T \mathbf{A} \mathbf{V}_k = \mathbf{T}_k,$$

$$\mathbf{V}_k = [\mathbf{v}_1 \,|\, \mathbf{v}_2 \,|\, \cdots \,|\, \mathbf{v}_k]$$

$$\mathbf{A}\mathbf{V}_k - \mathbf{V}_k \mathbf{T}_k = \delta_k \mathbf{v}_{k+1} \mathbf{e}_k^T,$$

$$\mathbf{T}_{k+1} = \begin{bmatrix} \mathbf{T}_k & \star \\ \star & \star \end{bmatrix}$$

- k-th iterative similarity

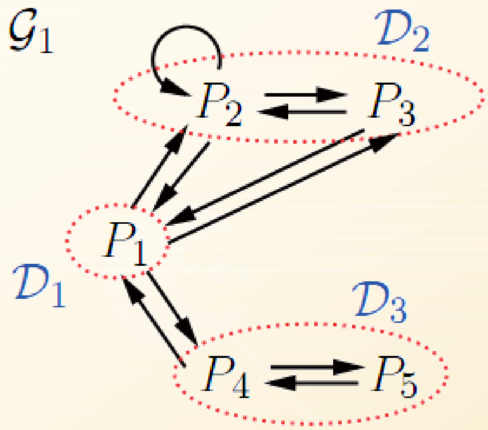
$$[\hat{\mathbf{S}}_k]_{i,j} = [\mathbf{V}_k \cdot \sigma_{\max}(\mathbf{T}_k)]_i \times [\mathbf{V}_k \cdot \sigma_{\max}(\mathbf{T}_k)]_j$$

- Estimate Error:

$$\|\hat{\mathbf{S}}_k - \mathbf{S}\|_2 \le \epsilon_k$$

$$\epsilon_k = 2 \times \left|\delta_k \times [\sigma_{\max}(\mathbf{T}_k)]_k\right|$$

Example



Assume $\mathbf{A} = \tilde{\mathbf{A}} + 1/5^2$ with

$$\tilde{\mathbf{A}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & 0 \\ \frac{1}{6} & \frac{7}{12} & \frac{7}{12} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{6} & \frac{7}{12} & 0 & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{3} & \frac{1}{8} & \frac{1}{8} & 0 & \frac{5}{12} \\ 0 & \frac{1}{8} & \frac{1}{8} & \frac{5}{12} & 0 \end{bmatrix}$$

Given $\epsilon = 0.05$

$$\epsilon_k = 2 \times |\delta_k \times [\sigma_{\max}(\mathbf{T}_k)]_k|$$

(2)

- Arnoldi Decomposition:

k	$\mathbf{T}_k$	$\mathbf{v}_{k+1}$	δ_k	$\sigma_{\max}(\mathbf{T}_k)$	ϵ_k
0	–	$[.447\ .447\ .447\ .447\ .447]^T$	–	–	–
1	$[1.08]$	$[.125\ .750\ -.125\ -.125\ -.625]^T$	.298	$[1]$	.596
2	$\begin{bmatrix} 1.08 & .298 \\ .298 & .190 \end{bmatrix}$	$[-.089\ .044\ .710\ -.697\ .032]^T$	.359	$\begin{bmatrix} .957 \\ .290 \end{bmatrix}$	.208
3	$\begin{bmatrix} 1.08 & .298 & 0 \\ .298 & .190 & .359 \\ 0 & .359 & -.083 \end{bmatrix}$	$[-.881\ .329\ .137\ .280\ .135\ .231]^T$	.231	$\begin{bmatrix} .945 \\ .316 \\ .090 \end{bmatrix}$	.041

$$\hat{\mathbf{S}}_3 = \begin{bmatrix} .206 & .301 & .203 & .146 & .103 \\ .301 & .440 & .296 & .213 & .151 \\ .203 & .296 & .199 & .143 & .102 \\ .146 & .213 & .143 & .102 & .073 \\ .103 & .151 & .102 & .073 & .051 \end{bmatrix}$$

(1)

$$\mathbf{A}\mathbf{V}_k - \mathbf{V}_k\mathbf{T}_k = \delta_k \mathbf{v}_{k+1}\mathbf{e}_k^T,$$

(3)

$$[\hat{\mathbf{S}}_k]_{i,j} = [\mathbf{V}_k \cdot \sigma_{\max}(\mathbf{T}_k)]_i \times [\mathbf{V}_k \cdot \sigma_{\max}(\mathbf{T}_k)]_j$$

Edge Update on Dynamic Graphs

- Incremental UAM

  Given old G =(D,R) and a new G'=(D,R'), the incremental UAM is a list of edge updates, i.e., $\bar{\mathbf{A}} = \mathbf{A}' - \mathbf{A}$

- Main idea

  To reuse $\bar{\mathbf{A}}$ and the eigen-pair (α_p, ξ_p) of the old A to compute $\mathbf{S}'$

  $\bar{\mathbf{A}}$ is a sparse matrix when the number δ of edge updates is small.

- Incrementally computing SimFusion+

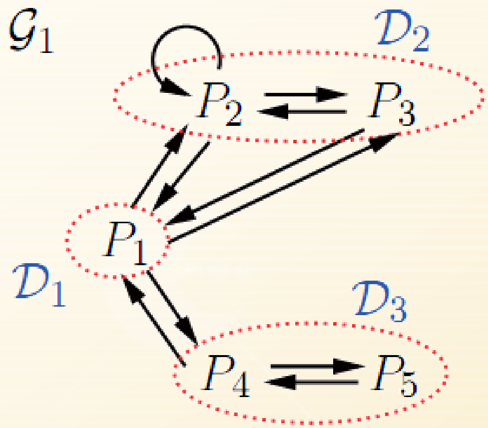
$$[\mathbf{S}']_{i,j} = [\boldsymbol{\xi}']_i \cdot [\boldsymbol{\xi}']_j \ \textit{with} \ [\boldsymbol{\xi}']_i = [\boldsymbol{\xi}_1]_i + \sum_{p=2}^{n} c_p \times [\boldsymbol{\xi}_p]_i$$

$$c_p = \frac{\boldsymbol{\xi}_p^T \cdot \boldsymbol{\eta}}{\alpha_p - \alpha_1} \ \ \textit{and} \ \ \boldsymbol{\eta} = \bar{\mathbf{A}} \cdot \boldsymbol{\xi}_1$$

O(δn) time

O(n) space

Example



Suppose edges (P1,P2) and (P2,P1) are removed.

$$\bar{\mathbf{A}} = \begin{bmatrix} 0 & -\frac{1}{6} & 0 & 0 & 0 \\ -\frac{1}{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

p	α_p	$\boldsymbol{\xi}_p$	c_p
1	1.184	$[.431\ .673\ .451\ .322\ .232]^T$	–
2	.503	$[.708\ -.522\ -.242\ .388\ .132]^T$	.062
3	-.480	$[-.256\ -.020\ .095\ .716\ -.641]^T$	-.018
4	-.366	$[-.021\ -.507\ .853\ -.119\ .017]^T$	-.025
5	.242	$[.497\ .127\ .037\ -.467\ -.719]^T$	.069

$$\boldsymbol{\eta} = \bar{\mathbf{A}} \cdot \boldsymbol{\xi}_1 = [-.112 \quad -.072 \quad 0 \quad 0 \quad 0]^T.$$

$$\begin{aligned} c_2 &= \boldsymbol{\xi}_2^T \cdot \boldsymbol{\eta}/(\alpha_2 - \alpha_1) = -.0419/(.503 - 1.184) = .062, \\ c_3 &= \boldsymbol{\xi}_3^T \cdot \boldsymbol{\eta}/(\alpha_3 - \alpha_1) = .030/(-.480 - 1.184) = -.018. \end{aligned}$$

$$\boldsymbol{\xi}' = \boldsymbol{\xi}_1 + \sum_{p=2}^{5} c_p \times \boldsymbol{\xi}_p = [.327\ .703\ .485\ .326\ .266]^T$$

$$\mathbf{S}' = \begin{bmatrix} .107 & .230 & .159 & .107 & .087 \\ .230 & .494 & .341 & .230 & .187 \\ .159 & .341 & .235 & .158 & .129 \\ .107 & .230 & .158 & .107 & .087 \\ .087 & .187 & .129 & .087 & .071 \end{bmatrix}$$

Experimental Setting

- Datasets
  - Synthetic data (RAND 0.5M-3.5M)
  - Real data (DBLP, WEBKB)

DBLP

	$\mathcal{G}_1$: 01-02	$\mathcal{G}_2$: 01-04	$\mathcal{G}_3$: 01-06	$\mathcal{G}_4$: 01-08	$\mathcal{G}_5$: 01-10
$\lvert\mathcal{D}\rvert$	1,838	3,723	5,772	9,567	12,276
$\lvert\mathcal{R}\rvert$	7,103	14,419	29,054	45,310	64,208

WEBKB

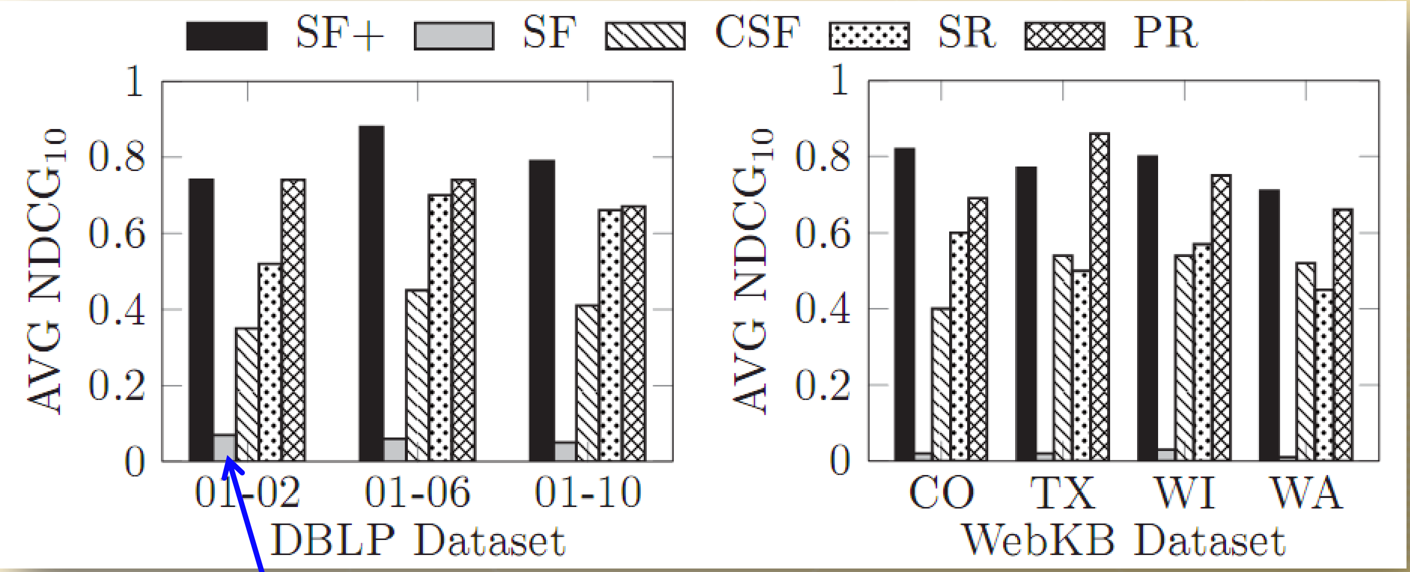
	U_1: CO	U_2: TE	U_3: WA	U_4: WI
$\lvert\mathcal{D}\rvert$	867	827	1,263	1,205
$\lvert\mathcal{R}\rvert$	1,496	1,428	2,969	1,805

- Compared Algorithms
  - SimFusion+ and IncSimFusion+ ;
  - SF, a SimFusion algorithm via matrix iteration [Xi et. al, SIGIR 05];
  - CSF, a variant SF, using PageRank distribution [Cai et. al, SIGIR 10];
  - SR, a SimRank algorithm via partial sums [Lizorkin et. al, VLDBJ 10];
  - PR, a P-Rank encoding both in- and out-links [Zhao et. al, CIKM 09];

Experiment (1): Accuracy

$$\mathrm{NDCG}_p = \frac{1}{\mathrm{IDCG}_p} \sum_{i=1}^{p} (2^{\mathrm{rank}_i} - 1)/(\log_2(1+i))$$

On DBLP and WEBKB

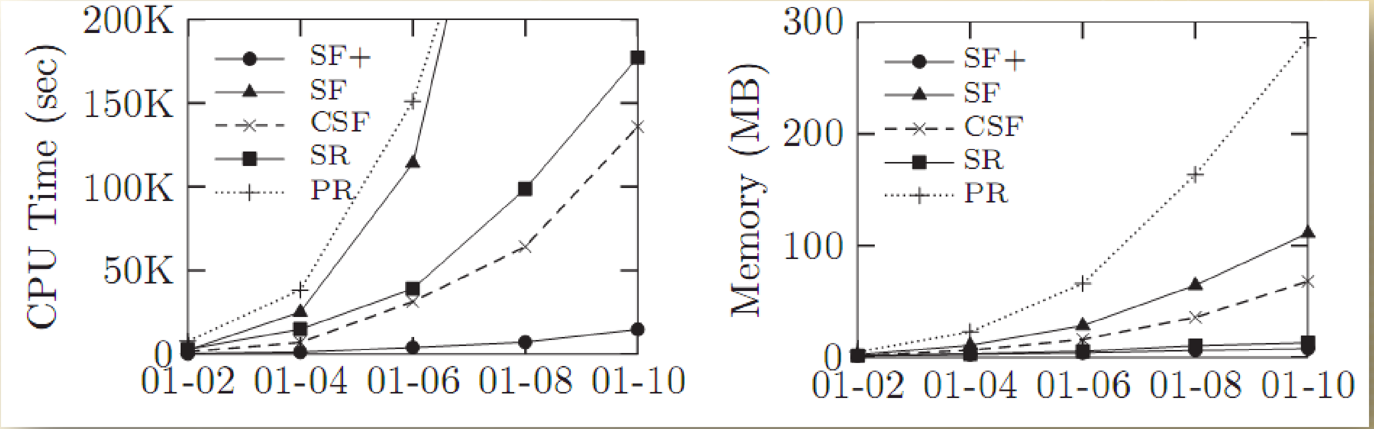


SF+ accuracy is consistently stable on different datasets.

SF seems hardly to get sensible similarities as all its similarities asymptotically approach the same value as K grows.

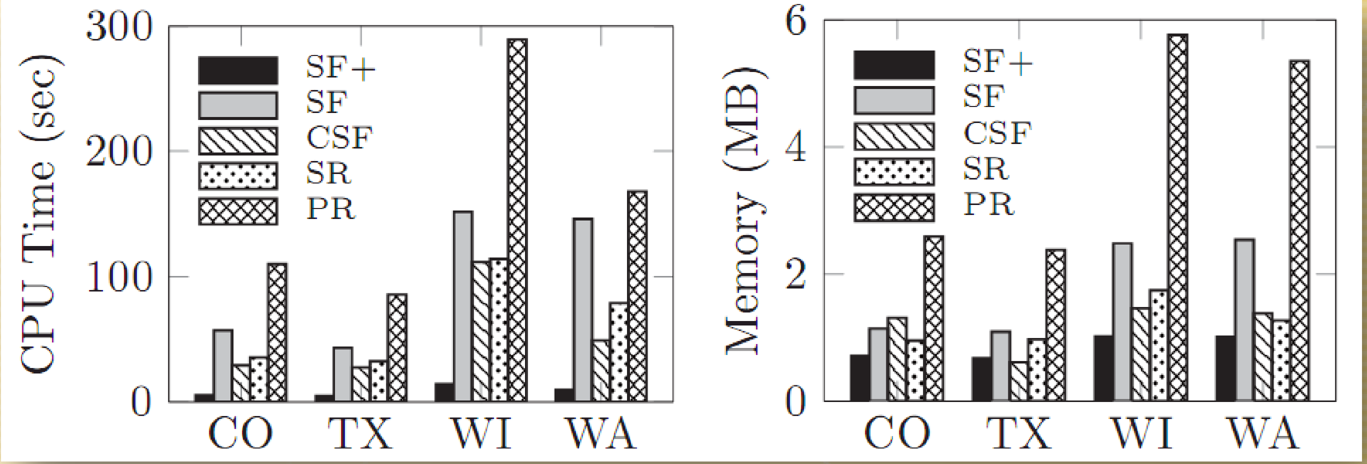
Experiment (2): CPU Time and Space

On DBLP



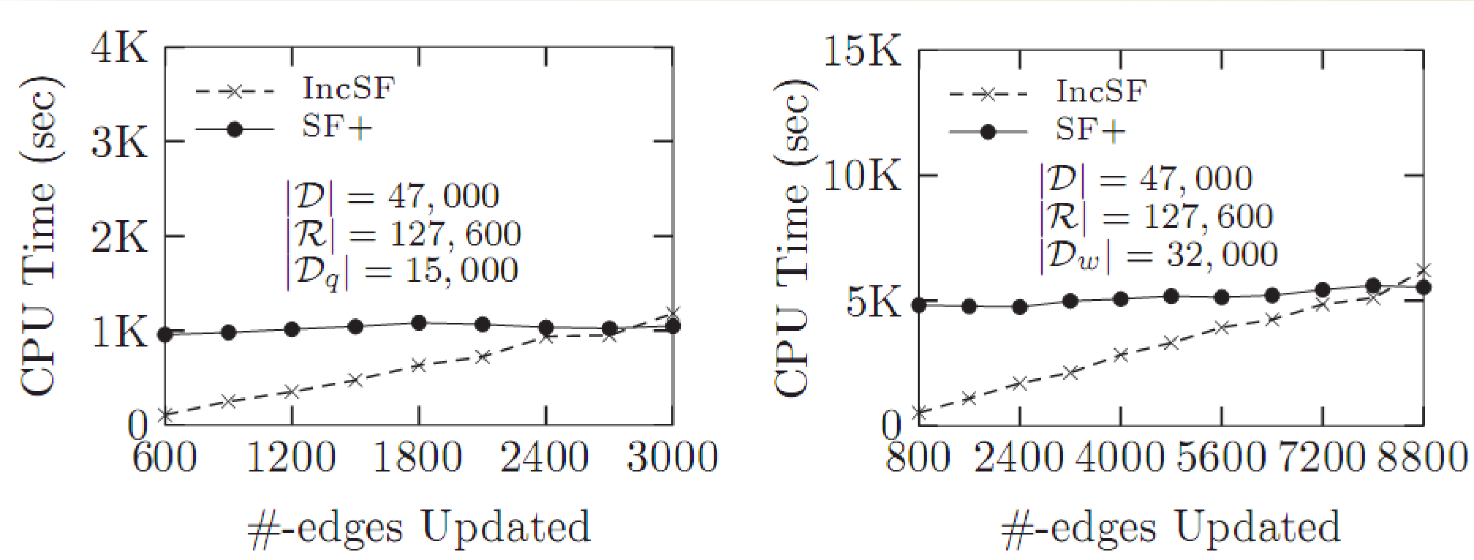
SF+ outperforms the other approaches, due to the use of $\sigma_{max}(T_k)$

On WEBKB



Experiment (3): Edge Updates

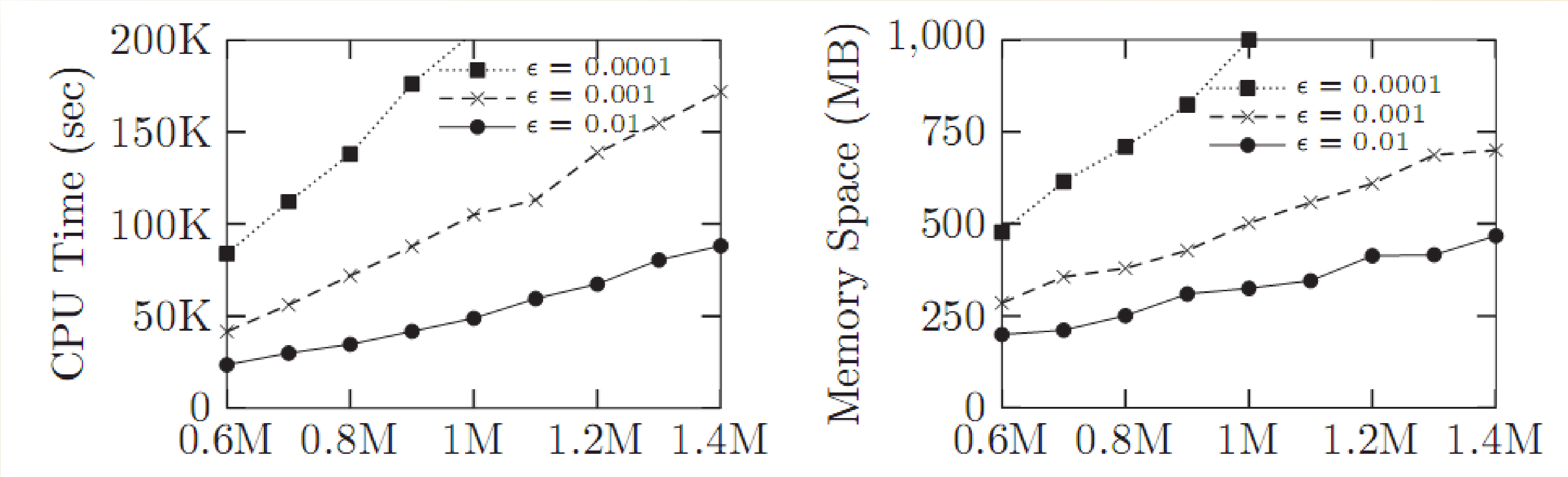
Varying δ



IncSF+ outperformed SF+ when δ is small.

For larger δ, IncSF+ is not that good because the small value of δ preserves the sparseness of the incremental UAM.

Experiment (4) : Effects of ϵ



The small choice of ϵ imposes more iterations on computing T_k and v_k, and hence increases the estimation costs.

Conclusions

- A revision of SimFusion+, for preventing the trivial solution and the divergence issue of the original model.
- Efficient techniques to improve the time and space of SimFusion+ with accuracy guarantees.
- An incremental algorithm to compute SimFusion+ on dynamic graphs when edges are updated.

Future Work

- Devise vertex-updating methods for incrementally computing SimFusion+.
- Extend to parallelize SimFusion+ computing on GPU.

Thank You !