東華大學

Optimization Techniques for Structural Similarity Computation on Large Networks

Presented by

Weiren Yu

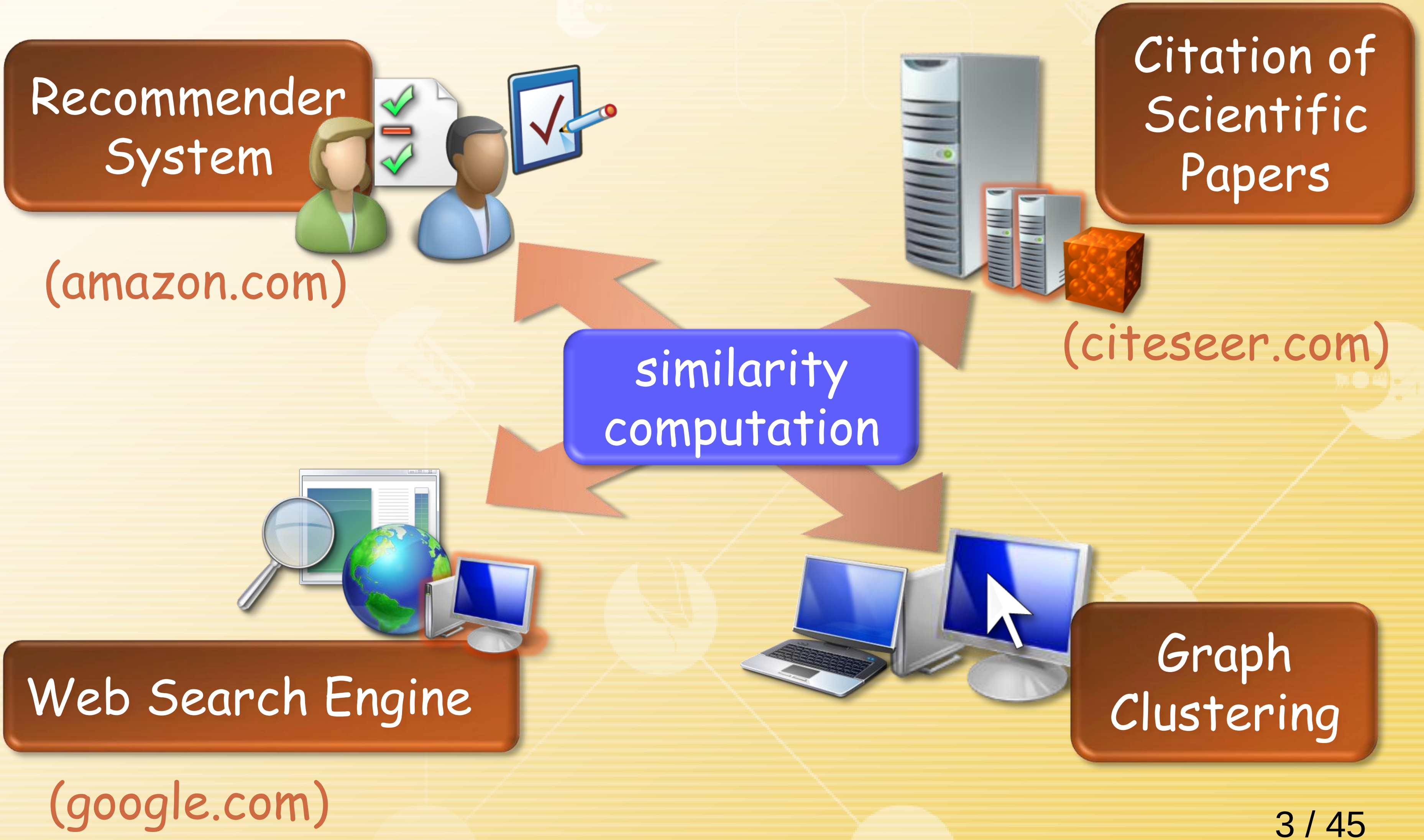
School of Computer Science & Engineering,

The University of New South Wales

Outline

1 Background

- Many applications require a measure of similarity between objects.



Recommender System

(amazon.com)

Citation of Scientific Papers

(citeseer.com)

similarity computation

Web Search Engine

(google.com)

Graph Clustering

1 Background

- Text-based Similarity (content)
  - Cosine similarity $\sigma_{\text{cosine}} = \frac{|\Gamma_i \cap \Gamma_j|}{\sqrt{|\Gamma_i||\Gamma_j|}}$ Jaccard index $\sigma_{\text{Jaccard}} = \frac{|\Gamma_i \cap \Gamma_j|}{|\Gamma_i \cup \Gamma_j|}$
- Link-based Similarity (structure)
  - PageRank [Larry Page, Google Tech. Rep.' 99]
    - *One page's authority is decided by its neighbors' authorities.*
  - SimRank [Jeh and Widom, SIGKDD'02]
  - Penetrating-Rank [Zhao et. al, CIKM'09]
    - *Two objects are similar if they are referenced by similar objects.*
  - SimFusion [Xi et. al, SIGIR'05]
    - *The similarity between two data objects is reinforced by the similarity of their related objects.*

1 Background

- The success of Google PageRank has demystified the importance of link-based similarity measure.

A space and time efficient algorithm for SimRank computation
dl.acm.org/citation.cfm?id=2158804 - 翻译此页
作者：W Yu - 2012 - 被引用次数：6 - 相关文章
A space and time efficient algorithm for SimRank computation, 2012 Article.
Bibliometrics Data Bibliometrics. · Downloads (6 Weeks): n/a · Downloads (12 ...

- Merits of link-based similarity measure:
  - Applicable to any domain with object-to-object relationships

    (It is a graph-theoretic model that reflects a better human intuition with a solid rationale.)
  - No requirement of extra human-built hierarchies

    (It purely hinges on the structure of linkage patterns.)
  - Possessing good expansibility

    (It can be combined with other domain-specific measures.)

2 Aims and Objectives

- Huge networks have been mounting up, calling for new techniques to efficiently handle similarity computations on large-scale graphs.
  - the increasing scale of the Web
  - the ubiquity of the Internet

High CPU time !!

High RAM space !!

- My research topic aims to develop, analyze, implement and evaluate novel approaches to optimize link-based similarity computation.
  - speed up the computations of the existing similarity models (i.e., SimRank, SimFusion, P-Rank)
  - improve existing models for effectively measuring similarity
  - develop a user-friendly system prototype for evaluation

3 Challenges

Focus on optimizing SimRank, SimFusion, P-Rank:

- To reduce the complexity of the best-known algorithms
  - computational time
  - memory space
  - convergence rate

efficiency

scalability

- To accurately compute the similarity scores
  - accuracy estimate
  - stability & sensitivity analysis

effectiveness

- To extend the existing models
  - static graphs → dynamic networks
  - single machine → parallel version

SimRank Measure

- Given a network G=(V,E), and a link-based scoring function s: V x V → [0,1], it is to efficiently compute similarity scores of all vertex-pairs in G.

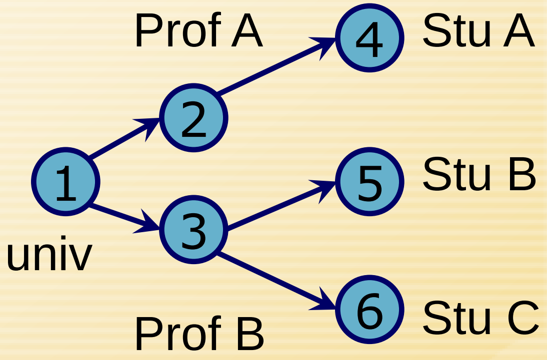
- SimRank Similarity [SIGKDD'02]
  - s (a, a) = 1,
  - s (a, b) = 0, if I(a) = Ø or I(b) = Ø,
  - otherwise:

$$s(a,b) = \frac{c}{|I(a)| \cdot |I(b)|} \cdot \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s\left(I_i(a), I_j(b)\right)$$

High complexity !!!

$O(Kn^4)$ time

$O(n^2)$ space



Setting *c*=0.8, we compute s(2,3) and s(4,5).

$$\because\ I(2) = I(3) = \{1\}, \quad I(4) = \{2\}, \quad I(5) = \{3\}.$$

$$\therefore\ s(2,3) = \frac{0.8}{1 \times 1} \times s(1,1) = 0.8$$

$$s(4,5) = \frac{0.8}{1 \times 1} \times s(2,3) = 0.64$$

4 State of the Arts : Related Work

- **Deterministic Method** [SIGKDD'02, VLDBJ'10]

  (following the fixed-point iteration to compute similarity)

$$s^{(k+1)}(a,b)=\frac{c}{|I(a)|\cdot|I(b)|}\cdot\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}s^{(k)}\left(I_i(a),I_j(b)\right)$$

  - ✓ Advantage: accuracy guarantee
  - ✓ Disadvantage: high time and space (cubic time and quadratic space)

- **Probabilistic Method** [EDBT'05, TKDE'05]

  (utilizing the Monte-Carlo sampling approach to estimate similarity)

$$s(a,b)=E\left(c^{\tau(a,b)}\right)$$

  - ✓ Advantage: scalability on large graphs (linear time and space)
  - ✓ Disadvantage: low estimation quality

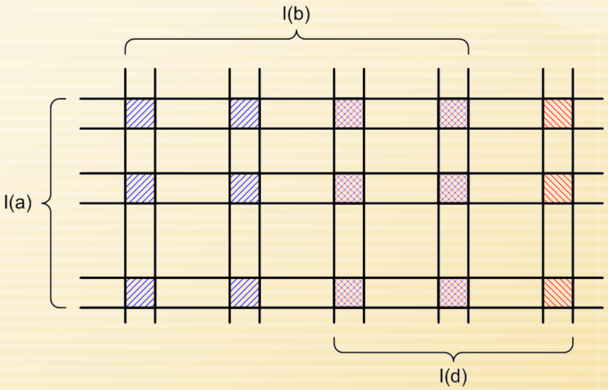
4 State of the Arts : Related Work

[Lizorkin et al. , VLDB J.'10]

- *Main Contributions.*
  - A precise accuracy estimate is presented for SimRank iteration.

$$\left|s^{(k)}(a,b)-s(a,b)\right|\leq c^{k+1}$$

  - A partial sum function is utilized to improve SimRank computational complexity from $O(kn^4)$ to $O(kn^3)$.

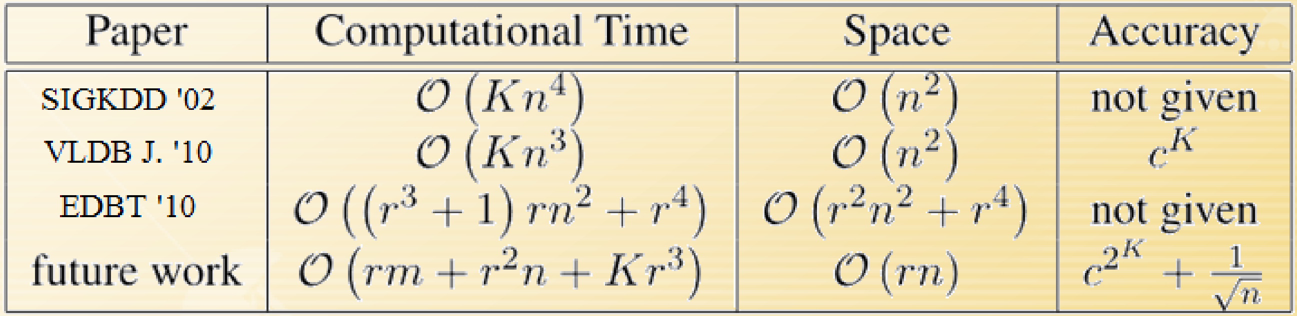


$$s^{(k+1)}(a,b)=\frac{c}{|I(a)|\cdot|I(b)|}\cdot\sum_{i=1}^{|I(a)|}\underbrace{\sum_{j=1}^{|I(b)|}s^{(k)}\left(I_i(a),I_j(b)\right)}_{=Partial_{I(a)}^{s^{(k)}}(I_j(b))}$$

  - A threshold sieving heuristic is introduced and its accuracy estimation is given that further improves the efficiency.

5.1 Contributions: SimRank

- Motivation:
  - The high complexity of time and space is still a mighty obstacle in using SimRank on large networks.



Paper	Computational Time	Space	Accuracy
SIGKDD '02	$\mathcal{O}\left(Kn^4\right)$	$\mathcal{O}\left(n^2\right)$	not given
VLDB J. '10	$\mathcal{O}\left(Kn^3\right)$	$\mathcal{O}\left(n^2\right)$	c^K
EDBT '10	$\mathcal{O}\left(\left(r^3+1\right)rn^2+r^4\right)$	$\mathcal{O}\left(r^2n^2+r^4\right)$	not given
future work	$\mathcal{O}\left(rm+r^2n+Kr^3\right)$	$\mathcal{O}\left(rn\right)$	$c^{2^K}+\frac{1}{\sqrt{n}}$

  - SimRank computation is iterative in nature, but no prior work has studied the stability of SimRank, which can

    (i) gauge the sensitivity of similarity to the perturbations in the link structure (e.g., by adding or removing edges)

    (ii) imply whether large amounts of accumulated round-off errors will run the risk of producing nonsensical similarity.

5.1 Contributions: SimRank

- Main Contributions:

convergence rate

- A "squaring memoization" technique is devised for SimRank computation, which cuts down the number of iterations exponentially for a given accuracy.

time / space complexity

- An order-r (≪ n) Krylov subspace is deployed for speeding up SimRank computation in $\mathcal{O}(rm+r^2n+K'r^3)$ time and $\mathcal{O}(rn)$ space up to an additive error of $\left(c^{2^{K'}}+\frac{1}{\sqrt{n}}\right)$ for any vertex-pair.

stability

- A notion of SimRank condition number is introduced, and a tight bound of this number is provided, aiming at analyzing similarity stability.

1) Speed up Convergence Rate

- Naïve SimRank Iterative Paradigm. [Lizorkin et al. , VLDB J.'10]

$$\begin{cases}\mathbf{S}^{(0)} = (1-c)\cdot\mathbf{I}_n, \\ \mathbf{S}^{(k+1)} = c\cdot\mathbf{Q}\cdot\mathbf{S}^{(k)}\cdot\mathbf{Q}^T + (1-c)\cdot\mathbf{I}_n.\end{cases}$$

$$\|\mathbf{S}^{(k)} - \mathbf{S}\|_{\max} \leq c^{k+1}$$

- "Squaring Memoization" Paradigm.

$$\begin{cases}\mathbf{S}_{\langle 2\rangle}^{(0)} = (1-c)\cdot\mathbf{I}_n, \\ \mathbf{S}_{\langle 2\rangle}^{(k+1)} = \mathbf{S}_{\langle 2\rangle}^{(k)} + c^{2^k}\cdot\mathbf{Q}^{2^k}\cdot\mathbf{S}_{\langle 2\rangle}^{(k)}\cdot(\mathbf{Q}^{2^k})^T\end{cases}$$

$$\|\mathbf{S}_{\langle 2\rangle}^{(k)} - \mathbf{S}\|_{\max} \leq c^{2^k}$$

- Main Idea:
  - Once squared, the matrix $\mathbf{Q}^{2^k}$ is memoized for the next iteration and thus will not be recomputed when subsequently needed.

$$\mathbf{Q}^{2^k} = \underbrace{\mathbf{Q}\cdot\mathbf{Q}\cdot\cdots\cdot\mathbf{Q}}_{2^k} \qquad \mathbf{Q}^{2^k} = \underbrace{\left(\left(\left(\mathbf{Q}^2\right)^2\right)^2\cdots\right)^2}_{k}$$

1) Speed up Convergence Rate

Naïve SimRank Iterative Paradigm	“Squaring Memoization” Paradigm
$\mathbf{S}^{(0)} = (1-c)\cdot\mathbf{I}_n$	$\mathbf{S}_{\langle 2\rangle}^{(0)} = (1-c)\cdot\mathbf{I}_n$
$\mathbf{S}^{(1)} = (1-c)\cdot\left[\mathbf{I}_n + c\mathbf{Q}\mathbf{Q}^T\right]$	$\mathbf{S}_{\langle 2\rangle}^{(1)} = (1-c)\cdot\left[\mathbf{I}_n + c\mathbf{Q}\mathbf{Q}^T\right]$
$\mathbf{S}^{(2)} = (1-c)\cdot\left[\mathbf{I}_n + c\mathbf{Q}\mathbf{Q}^T + c^2\mathbf{Q}^2(\mathbf{Q}^2)^T\right]$	$\mathbf{S}_{\langle 2\rangle}^{(2)} = (1-c)\cdot\left[\mathbf{I}_n + c\mathbf{Q}\mathbf{Q}^T + c^2\mathbf{Q}^2(\mathbf{Q}^2)^T + c^3\mathbf{Q}^3(\mathbf{Q}^3)^T\right]$
$\mathbf{S}^{(3)} = (1-c)\cdot\left[\mathbf{I}_n + c\mathbf{Q}\mathbf{Q}^T + c^2\mathbf{Q}^2(\mathbf{Q}^2)^T + c^3\mathbf{Q}^3(\mathbf{Q}^3)^T\right]$	$\boxed{\mathbf{S}_{\langle 2\rangle}^{(3)} = (1-c)\cdot\sum_{i=0}^{7} c^i\mathbf{Q}^i(\mathbf{Q}^i)^T}$
$\cdots \quad \cdots$	
$\boxed{\mathbf{S}^{(7)} = (1-c)\cdot\sum_{i=0}^{7} c^i\mathbf{Q}^i(\mathbf{Q}^i)^T}$	

$$\mathbf{S}_{\langle 2\rangle}^{(k)} = \mathbf{S}^{(2^k-1)}.$$

- In each step of “squaring memoization” iteration,
  one actually computes exponential steps (with base 2) of the conventional iteration.
  As a result, the convergence rate of “squaring memoization” iteration becomes exponentially faster than that of conventional iteration.

1) Speed up Convergence Rate

- “Squaring Memoization” Paradigm.

$$\begin{cases}\mathbf{S}_{\langle 2\rangle}^{(0)} = (1-c)\cdot\mathbf{I}_n,\\ \mathbf{S}_{\langle 2\rangle}^{(k+1)} = \mathbf{S}_{\langle 2\rangle}^{(k)} + c^{2^k}\cdot \mathbf{Q}^{2^k}\cdot \mathbf{S}_{\langle 2\rangle}^{(k)}\cdot \left(\mathbf{Q}^{2^k}\right)^T\end{cases}$$

$$\|\mathbf{S}_{\langle 2\rangle}^{(k)} - \mathbf{S}\|_{\max} \le c^{2^k}$$

- Extending to the “u-th Powering Memoization” Paradigm: (u=2, 3,…)

$$\begin{cases}\mathbf{S}_{\langle u\rangle}^{(0)} = (1-c)\cdot\mathbf{I}_n,\\ \mathbf{S}_{\langle u\rangle}^{(k+1)} = \sum_{i=0}^{u-1} c^{i\cdot u^k}\cdot \mathbf{Q}^{i\cdot u^k}\cdot \mathbf{S}_{\langle u\rangle}^{(k)}\cdot \left(\mathbf{Q}^{i\cdot u^k}\right)^T,\end{cases}$$

$$\|\mathbf{S}_{\langle u\rangle}^{(k)} - \mathbf{S}\|_{\max} \le c^{u^k}$$

- Complexity:

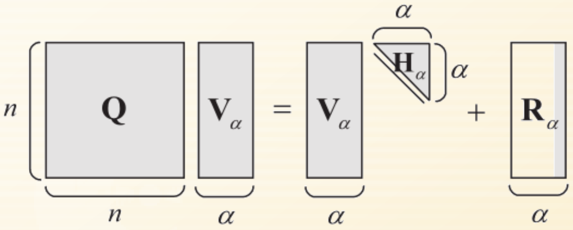
	FLOPs per iteration	#-iterations	total
naïve	$O(n^3)$	$\lceil \log_c \epsilon \rceil - 1$	$O((\lceil \log_c \epsilon \rceil - 1)\, n^3)$
u-th Powering	$O((u-1)\cdot n^3)$	$\lceil \log_u \log_c \epsilon \rceil$	$O(\lceil \log_u \log_c \epsilon \rceil\, (u-1)\, n^3)$

- “Squaring Memoization” achieves the best computational performance.

$$\mathcal{O}(f(u)) \text{ with } f(u) = \lceil \ln(\log_c \epsilon) \rceil \cdot \frac{u-1}{\ln u} \cdot n^3 \quad (u = 2, 3, \cdots)$$

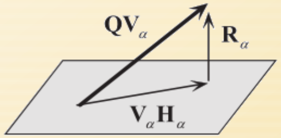
2) Improve Computational Efficiency

Krylov Subspace Projection

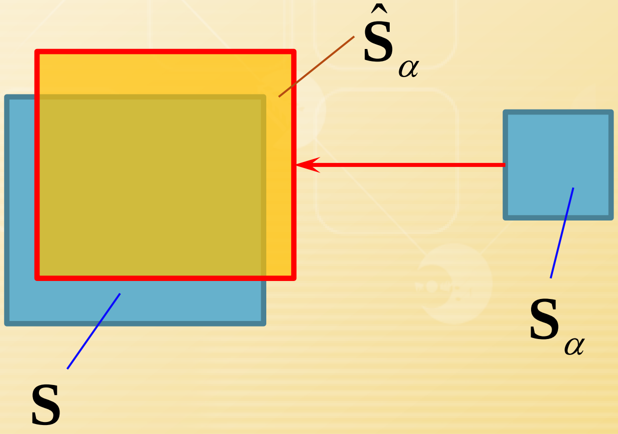


$$\Rightarrow \begin{aligned} \mathbf{V}_\alpha^T \cdot \mathbf{V}_\alpha &= \mathbf{I}_\alpha \\ \mathbf{V}_\alpha^T \cdot \mathbf{R}_\alpha &= 0 \\ \mathbf{V}_\alpha^T \cdot \mathbf{Q} \cdot \mathbf{V}_\alpha &= \mathbf{H}_\alpha \end{aligned}$$

$$\mathbf{Q}\cdot\overset{\substack{\text{orthogonal}\\ \downarrow}}{\mathbf{V}_\alpha} = \mathbf{V}_\alpha\cdot\overset{\substack{\text{upper Hessenburg}\\ \downarrow}}{\mathbf{H}_\alpha} + \overset{\substack{\text{residual}\\ \swarrow}}{\mathbf{R}_\alpha}$$



with $\mathbf{R}_\alpha = h_{\alpha+1,\alpha}\mathbf{v}_{\alpha+1}\mathbf{e}_\alpha^T$



Main Idea

- A projection of the matrix Q (n x n dimension) onto a Krylov subspace (α x α dimension with α ≪ n) is used for computing similarity.
- Due to its smaller dimension, the Krylov subspace based SimRank formula is relatively easier to solve with accuracy guarantees.

Krylov subspace (α x α)

$$\mathbf{S}_\alpha = c \cdot \mathbf{H}_\alpha \cdot \mathbf{S}_\alpha \cdot \mathbf{H}_\alpha^T + (1-c) \cdot \mathbf{I}_\alpha$$

$$\downarrow$$

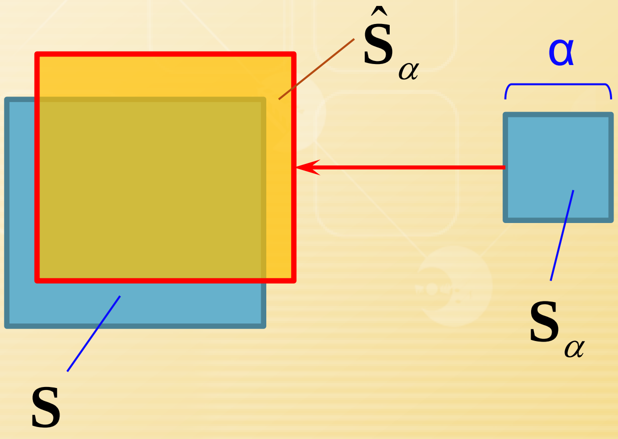
$$\hat{\mathbf{S}}_\alpha = \mathbf{V}_\alpha \cdot \mathbf{S}_\alpha \cdot \mathbf{V}_\alpha^T$$

original space (n x n)

$$\mathbf{S} = c \cdot (\mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T) + (1-c) \cdot \mathbf{I}_n$$

2) Improve Computational Efficiency

- Error Estimate



LEMMA. Let Err(⋆) be a matrix function defined by

$$Err\,(\mathbf{X}) \overset{def}{=} c\cdot\mathbf{Q}\cdot\mathbf{X}\cdot\mathbf{Q}^T-\mathbf{X}+(1-c)\cdot\mathbf{I}_n,\quad (\mathbf{X}\in\mathbb{R}^{n\times n})$$

Then for every α = 1,2,⋯ ,n, we have

$$\|Err(\hat{\mathbf{S}}_\alpha)\|_2\le\epsilon_\alpha,$$

where

$$\epsilon_\alpha=1-c\Big(1-h_{\alpha+1,\alpha}\sqrt{2\|\mathbf{H}_\alpha[\mathbf{S}_\alpha]_{\star,\alpha}\|_2^2+h_{\alpha+1,\alpha}^2[\mathbf{S}_\alpha]_{\alpha,\alpha}^2}\Big)$$

COROLLARY 1. $\|\mathrm{Err}(\hat{\mathbf{S}}_r)\|_2\le 1-c.$

THEOREM. For every α = 1,2,⋯ ,n, the following estimate holds:

$$\|\mathbf{S}-\hat{\mathbf{S}}_\alpha\|_2\le\frac{\sqrt{n}}{1-c}\cdot\|Err(\hat{\mathbf{S}}_\alpha)\|_2$$

COROLLARY 2. $\|\mathbf{S}-\hat{\mathbf{S}}_r\|_2\le\sqrt{n}.$

3) The Complete Framework

- Integrated with “Squaring Memoization”.

$$\begin{cases} \mathbf{S}_\alpha^{(0)} = (1-c)\cdot \mathbf{I}_\alpha, \\ \mathbf{S}_\alpha^{(k+1)} = \mathbf{S}_\alpha^{(k)} + c^{2^k}\cdot {\mathbf{H}_\alpha}^{2^k}\cdot \mathbf{S}_\alpha^{(k)} \cdot \left({\mathbf{H}_\alpha}^{2^k}\right)^T \end{cases}$$

$$\hat{\mathbf{S}}_\alpha^{(k)} = \mathbf{V}_\alpha \cdot \mathbf{S}_\alpha^{(k)}\cdot \mathbf{V}_\alpha^T$$

- Error Estimate.

$$\|\mathbf{S}-\hat{\mathbf{S}}_\alpha^{(k)}\|_2 \le \epsilon,\ \ \textit{with}\ \epsilon = n\cdot c^{2^k} + \frac{\sqrt{n}}{1-c}\cdot \epsilon_\alpha$$

- COROLLARY 3.

$$\tfrac{1}{n}\sqrt{\textstyle\sum_{i,j=1}^n \left([\mathbf{S}]_{i,j} - [\hat{\mathbf{S}}_r^{(k)}]_{i,j}\right)^2} \le c^{2^k}+\tfrac{1}{\sqrt{n}}$$

- Complexity Analysis.

Operation	**Time**	**Space**	**Error**
building Krylov subspace	O (rm)	O (rn)	
computing $s_r^{(K)}$ in the subspace	O(Kr^3)	O(r^2)	c^{2^K}
solving $\hat{s}_r^{(K)}$ in the whole space	O(r^{2}n + r^2)	O (rn)	$1/\sqrt{n}$
Total	O(rm+Kr^3+nr^2)	O(rn)	$c^{2^K}+1/\sqrt{n}$

4) SimRank Stability Analysis

- DEFINITION 1 (SimRank Condition Number).

For a graph G = (V,E) with Q being its backward transition matrix, let

$$\mathbf{L} \overset{def}{=} \mathbf{I}_{n^2} - c \cdot (\mathbf{Q} \otimes \mathbf{Q}).$$

The SimRank condition number of G, denoted by $\kappa_\infty(G)$, is defined as

$$\kappa_\infty(\mathcal{G}) \overset{def}{=} \|\mathbf{L}\|_\infty \cdot \|\mathbf{L}^{-1}\|_\infty$$

Here, $\|\mathbf{L}\|_\infty$ is the maximum absolute row sum of the matrix.

- Underlying Rationale.

$$s(a,b) = \frac{c}{|\mathcal{I}(a)|\,|\mathcal{I}(b)|} \sum_{i=1}^{|\mathcal{I}(a)|} \sum_{j=1}^{|\mathcal{I}(b)|} s\left(\mathcal{I}_i(a), \mathcal{I}_j(b)\right)$$

$$\mathbf{S} = c \cdot (\mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T) + (1-c) \cdot \mathbf{I}_n$$

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

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

$$vec\,(AXB) = (B^T \otimes A)\; vec(X)$$

$$vec(\mathbf{S}) = (1-c) \cdot \mathbf{L}^{-1} \cdot vec(\mathbf{I}_n) \text{ with } \mathbf{L} = \mathbf{I}_{n^2} - c \cdot (\mathbf{Q} \otimes \mathbf{Q}).$$

4) SimRank Stability Analysis

- THEOREM 1. Given a graph G = (V,E), for any damping factor c ∈ (0,1), the SimRank condition number has the following tight bound

$$\kappa_{\infty}(\mathcal{G}) \leq \frac{1+c}{1-c}$$

- Implications
  - evaluate how stable the similarity is to the perturbations in graphs
  - estimate the accuracy of the ranking results invoked by the iteration error

- Application

Actual version: $\mathbf{L} \cdot vec(\mathbf{S}) = (1-c) \cdot vec(\mathbf{I}_n)$

Perturbed version: $\tilde{\mathbf{L}} \cdot vec(\tilde{\mathbf{S}}) = (1-c) \cdot vec(\mathbf{I}_n)$

$$\frac{\|\mathbf{S}-\tilde{\mathbf{S}}\|_{\max}}{\|\tilde{\mathbf{S}}\|_{\max}} \leq \frac{1+c}{1-c} \cdot \frac{\|\tilde{\mathbf{L}}-\mathbf{L}\|_{\infty}}{\|\mathbf{L}\|_{\infty}}.$$

Setting c=0.95 holds the possibility that the relative error in similarity may be (1 + 0.95)/(1 − 0.95) = 40 times larger than the relative error in the link structure.

4) SimRank Stability Analysis

- EXAMPLE 1. The bound of SimRank condition number is tight.

Setting c = 0.7, on one hand,

$$\begin{aligned}\mathbf{L} &= \mathbf{I}_{n^2} - c\cdot(\mathbf{Q}\otimes\mathbf{Q}) \\ &= \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} - 0.7\cdot\left(\begin{bmatrix}0&1\\1&0\end{bmatrix}\otimes\begin{bmatrix}0&1\\1&0\end{bmatrix}\right) = \begin{bmatrix}1&0&0&-.7\\0&1&-.7&0\\0&-.7&1&0\\-.7&0&0&1\end{bmatrix},\end{aligned}$$

$$\mathbf{L}^{-1} = \begin{bmatrix}1&0&0&-.7\\0&1&-.7&0\\0&-.7&1&0\\-.7&0&0&1\end{bmatrix}^{-1} = \begin{bmatrix}1.960&0&0&1.373\\0&1.960&1.373&0\\0&1.373&1.960&0\\1.373&0&0&1.960\end{bmatrix},$$

$$\kappa_\infty(\mathcal{G}_1) = \|\mathbf{L}\|_\infty\cdot\|\mathbf{L}^{-1}\|_\infty = 1.7\times 3.333 = 5.667.$$

On the other hand, $\frac{1+c}{1-c} = \frac{1+0.7}{1-0.7} = 5.667.$

5.2 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):

$$\mathbf{L}_{\text{URM}}=\begin{pmatrix}\lambda_{1,1}\mathbf{L}_{1,1} & \lambda_{1,2}\mathbf{L}_{1,2} & \cdots & \lambda_{1,N}\mathbf{L}_{1,N}\\ \lambda_{2,1}\mathbf{L}_{2,1} & \lambda_{2,2}\mathbf{L}_{2,2} & \cdots & \lambda_{2,N}\mathbf{L}_{2,N}\\ \vdots & \vdots & \ddots & \vdots\\ \lambda_{N,1}\mathbf{L}_{N,1} & \lambda_{N,2}\mathbf{L}_{N,2} & \cdots & \lambda_{N,N}\mathbf{L}_{N,N}\end{pmatrix}$$

$$\mathbf{L}_{i,j}(x,y)=\begin{cases}\frac{1}{n_j}, & \text{if } \mathcal{N}_j(x)=\varnothing;\\ \frac{1}{|\mathcal{N}_j(x)|}, & \text{if } (x,y)\in\mathcal{R}_{i,j};\\ 0, & \text{otherwise.}\end{cases}$$

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

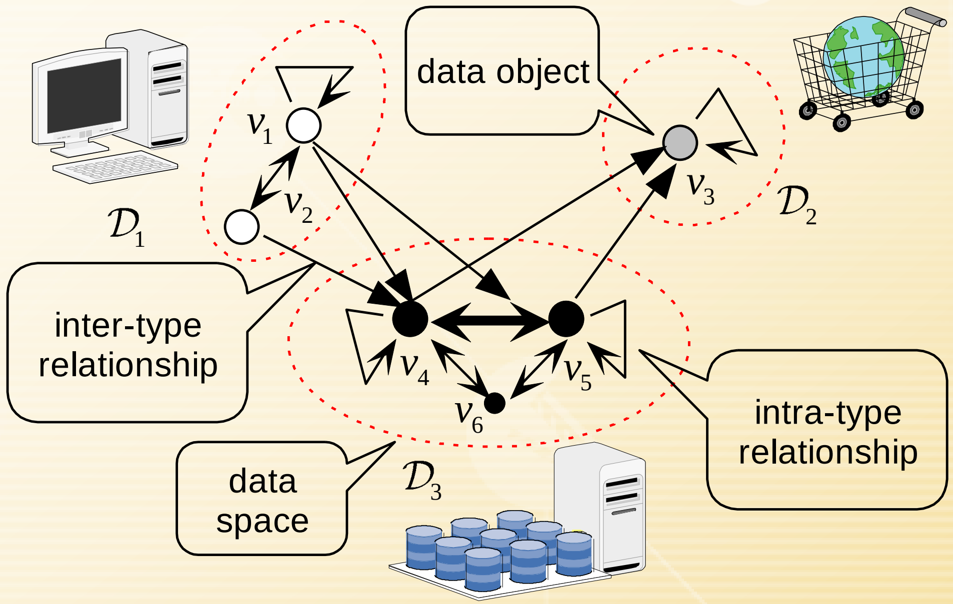
$$\exists\,\mathbf{S}=\begin{pmatrix}s_{1,1} & \cdots & s_{1,n}\\ \vdots & \ddots & \vdots\\ s_{n,1} & \cdots & s_{n,n}\end{pmatrix}\in\mathbb{R}^{n\times n}\quad s.t.\quad \mathbf{S}=\mathbf{L}\cdot\mathbf{S}\cdot\mathbf{L}^T.$$

Example.

SimFusion Similarity on Heterogeneous Domain

Trivial Solution !!!

$S=[1]_{nxn}$



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

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

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

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

High complexity !!!

$O(Kn^3)$ time

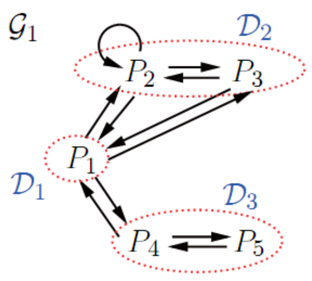
$O(n^2)$ space

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



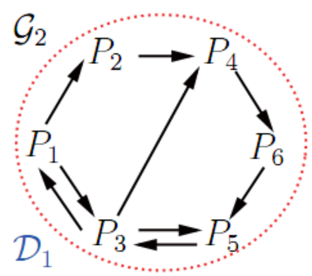
$$\lim_{k\to\infty} \textbf{SimFusion}^{(k)} = \begin{array}{c} \mathcal{D}_1\{ \\ \mathcal{D}_2\left\{\begin{array}{c} \\ \\ \end{array}\right. \\ \mathcal{D}_3\left\{\begin{array}{c} \\ \\ \end{array}\right. \end{array} \begin{bmatrix} 1 & .219 & .219 & .219 & .219 \\ .219 & 1 & .219 & .219 & .219 \\ .219 & .219 & 1 & .219 & .219 \\ .219 & .219 & .219 & 1 & .219 \\ .219 & .219 & .219 & .219 & 1 \end{bmatrix}$$

$$\underbrace{}_{\mathcal{D}_1} \underbrace{}_{\mathcal{D}_2} \underbrace{}_{\mathcal{D}_3}$$

$$\lim_{k\to\infty} \textbf{SimFusion+}^{(k)} = \begin{bmatrix} 1 & .290 & .194 & .139 & .100 \\ .290 & 1 & .304 & .217 & .156 \\ .194 & .304 & 1 & .145 & .105 \\ .139 & .217 & .145 & 1 & .075 \\ .100 & .156 & .105 & .075 & 1 \end{bmatrix}$$

$$\underbrace{}_{\mathcal{D}_1} \underbrace{}_{\mathcal{D}_2} \underbrace{}_{\mathcal{D}_3}$$

  - Divergent Solution on Homogeneous Domain



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

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

Root cause: row normalization of URM !!!

From URM to UAM

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

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

- Example

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

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

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

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

Original SimFusion

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

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

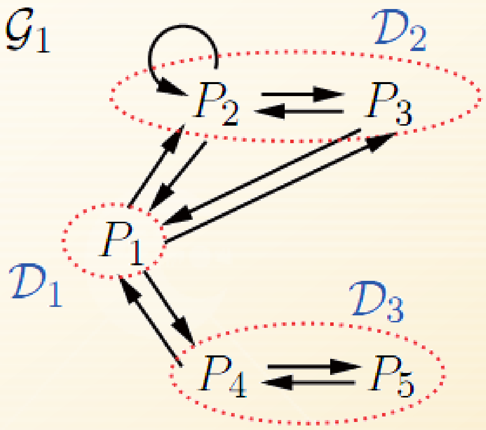
- Our approach: To convert SimFusion+ computation into finding the dominant eigenvector of the UAM A.

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

Pre-compute $\sigma_{max}(A)$ 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:

$$\mathbf{S}^{(0)} = \mathbf{1}$$

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

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

- 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 "⊗":

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

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

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:

$$\begin{tikzcd}
\mathbf{S}^{(k+1)} = \dfrac{\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T}{\|\mathbf{A} \cdot \mathbf{S}^{(k)} \cdot \mathbf{A}^T\|_2}. \arrow[r, "(1)"] & vec(\mathbf{S}^{(k+1)}) = \dfrac{(\mathbf{A} \otimes \mathbf{A}) \cdot vec(\mathbf{S}^{(k)})}{\|(\mathbf{A} \otimes \mathbf{A}) \cdot vec(\mathbf{S}^{(k)})\|_2} \arrow[d, "\text{Power Iteration}"] \\
vec(\mathbf{S}) = \sigma_{\max}(\mathbf{A}) \otimes \sigma_{\max}(\mathbf{A}) & \lim\limits_{k \to \infty} vec(\mathbf{S}^{(k)}) = \sigma_{\max}(\mathbf{A} \otimes \mathbf{A}) \arrow[l, "(2)"']
\end{tikzcd}$$

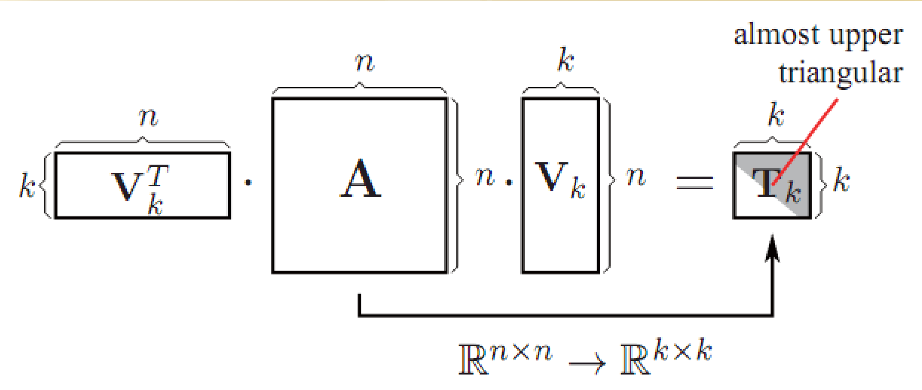
Accuracy Guarantee

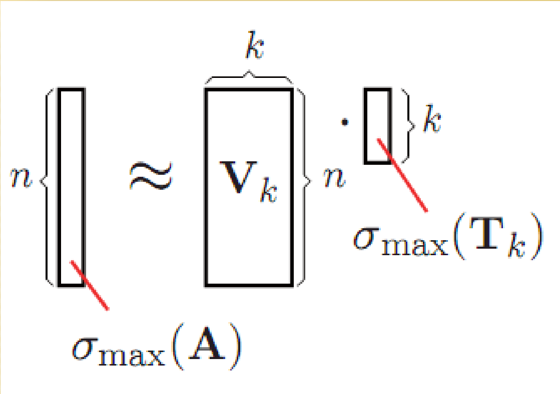
- Conventional Iterations: No accuracy guarantee !!!

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

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

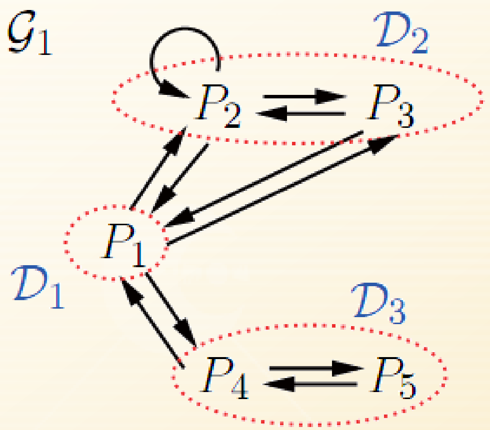
$$[\hat{\mathbf{S}}_k]_{i,j} = [\mathbf{V}_k \cdot \sigma_{\max}(\mathbf{T}_k)]_i \times [\mathbf{V}_k \cdot \sigma_{\max}(\mathbf{T}_k)]_j$$

- Estimate Error:

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

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

Example



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

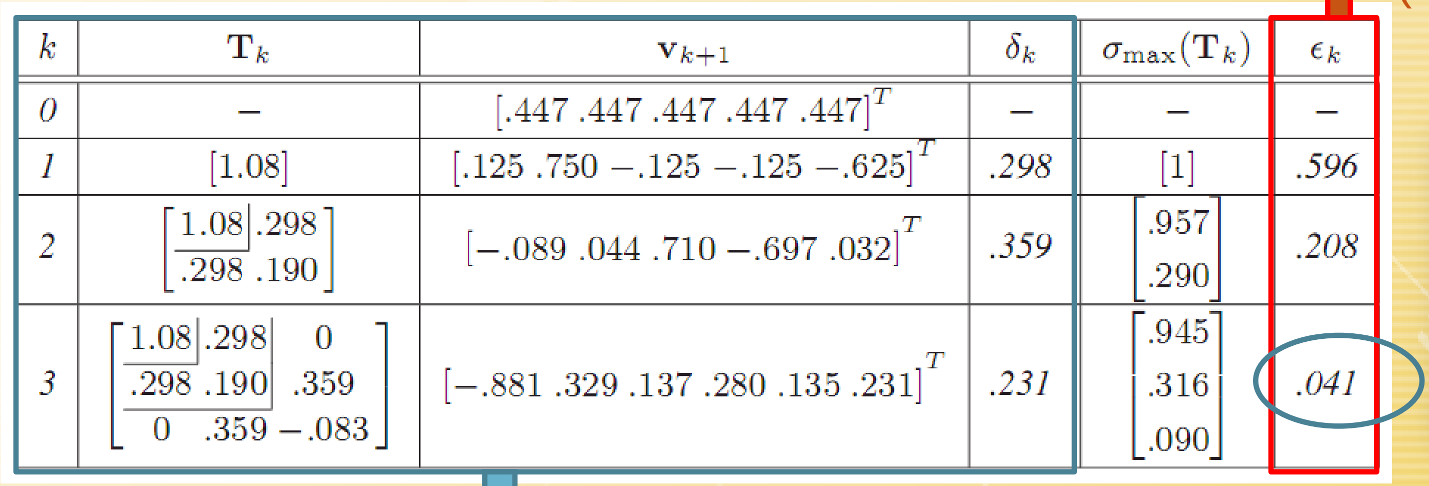
$$\tilde{\mathbf{A}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{3} & 0 \\ \frac{1}{6} & \frac{7}{12} & \frac{7}{12} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{6} & \frac{7}{12} & 0 & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{3} & \frac{1}{8} & \frac{1}{8} & 0 & \frac{5}{12} \\ 0 &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

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	$\left[\begin{array}{c\|c} 1.08 & .298 \\ .298 & .190 \end{array}\right]$	$[-.089\ .044\ .710\ -.697\ .032]^T$	.359	$\begin{bmatrix} .957 \\ .290 \end{bmatrix}$	.208
3	$\left[\begin{array}{c\|c\|c} 1.08 & .298 & 0 \\ .298 & .190 & .359 \\ 0 & .359 & -.083 \end{array}\right]$	$[-.881\ .329\ .137\ .280\ .135\ .231]^T$	.231	$\begin{bmatrix} .945 \\ .316 \\ .090 \end{bmatrix}$	.041

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

(3)

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

(1)

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

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+

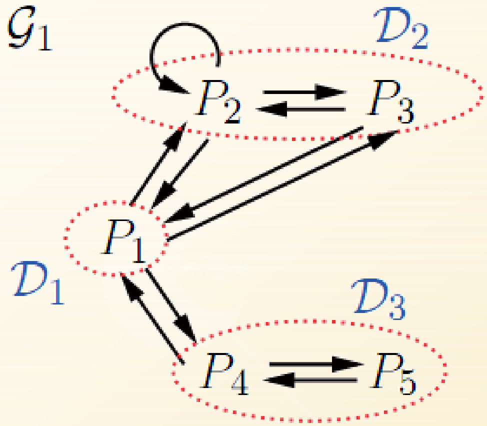
$$[\mathbf{S}']_{i,j} = [\boldsymbol{\xi}']_i \cdot [\boldsymbol{\xi}']_j \ \mathit{with} \ [\boldsymbol{\xi}']_i = [\boldsymbol{\xi}_1]_i + \textstyle\sum_{p=2}^{n} c_p \times [\boldsymbol{\xi}_p]_i$$

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

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	ξ_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

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

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

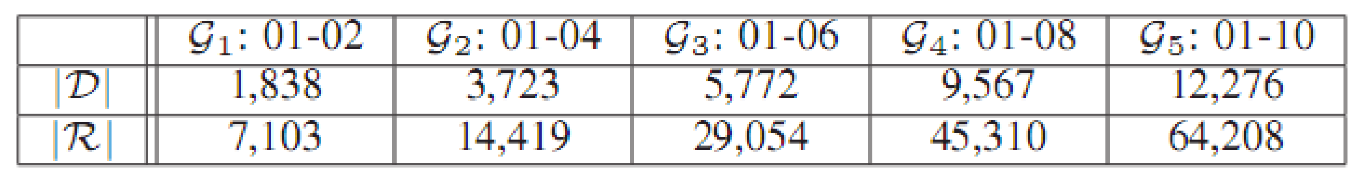
$$\boldsymbol{\xi}' = \boldsymbol{\xi}_1 + \sum\nolimits_{p=2}^{5} c_p \times \boldsymbol{\xi}_p = [.327\ .703\ .485\ .326\ .266]^T$$

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

Experimental Setting

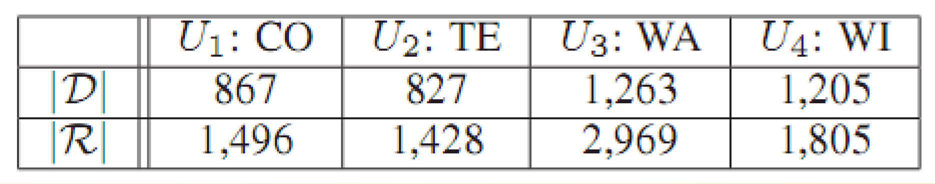
- Datasets
  - Synthetic data (RAND 0.5M-3.5M)
  - Real data (DBLP, WEBKB)

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



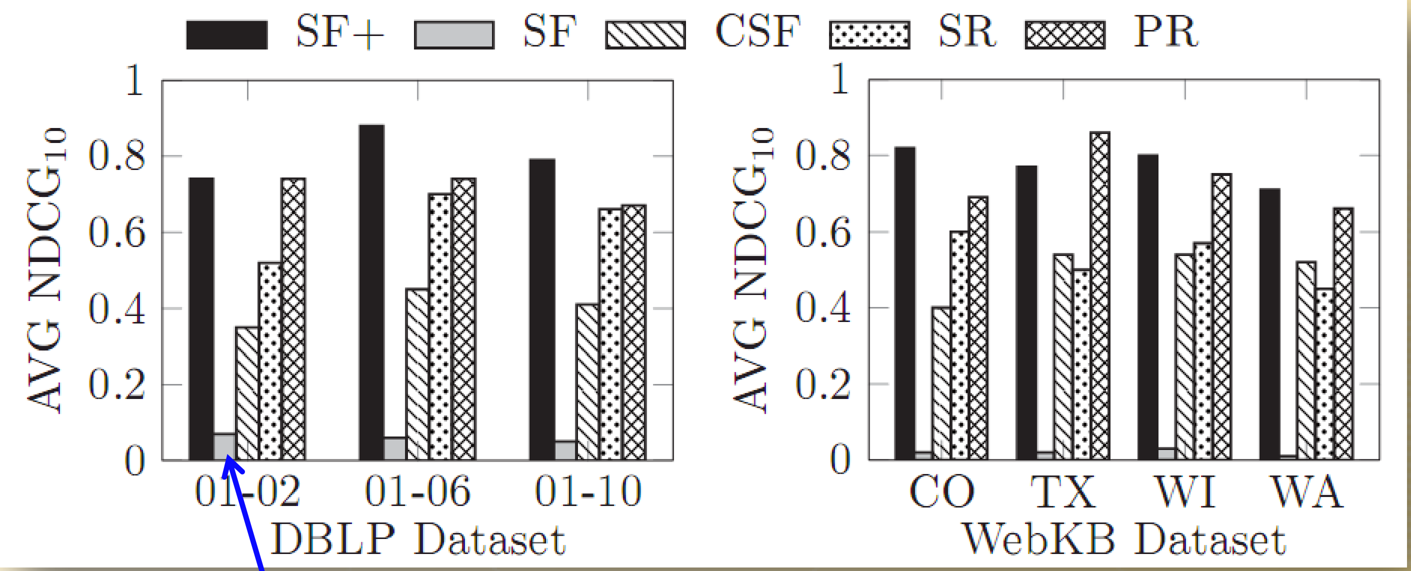
	U_1: CO	U_2: TE	U_3: WA	U_4: WI
$\lvert\mathcal{D}\rvert$	867	827	1,263	1,205
$\lvert\mathcal{R}\rvert$	1,496	1,428	2,969	1,805

- 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

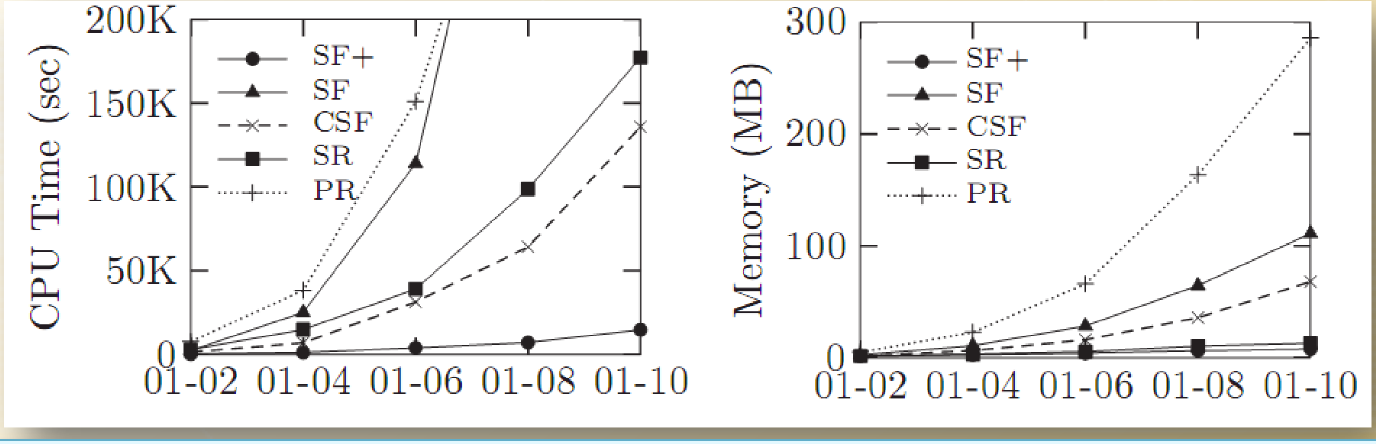


SF+ accuracy is consistently stable on different datasets.

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

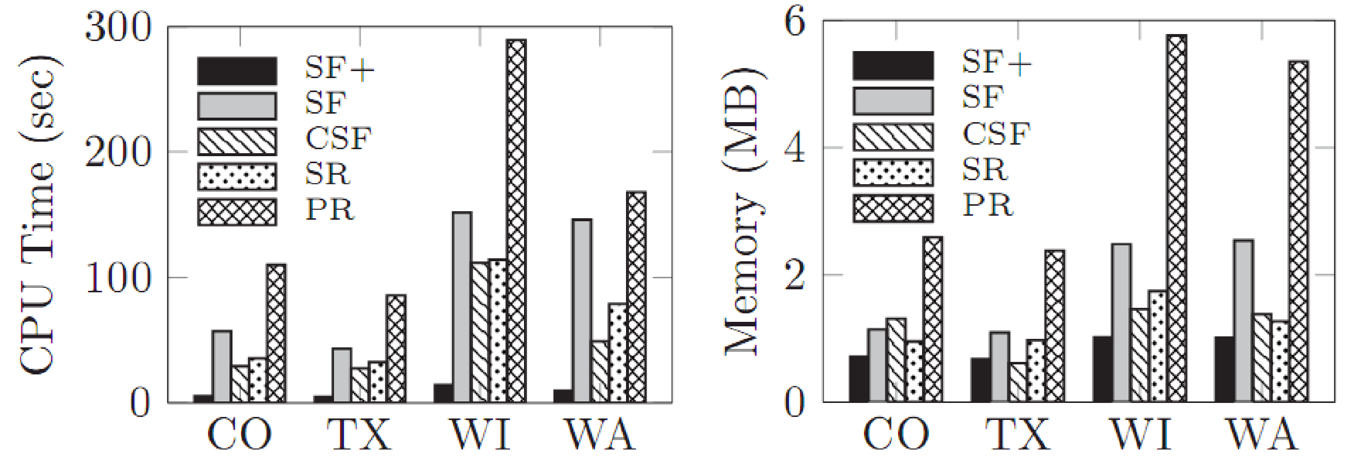
Experiment (2): CPU Time and Space

On DBLP



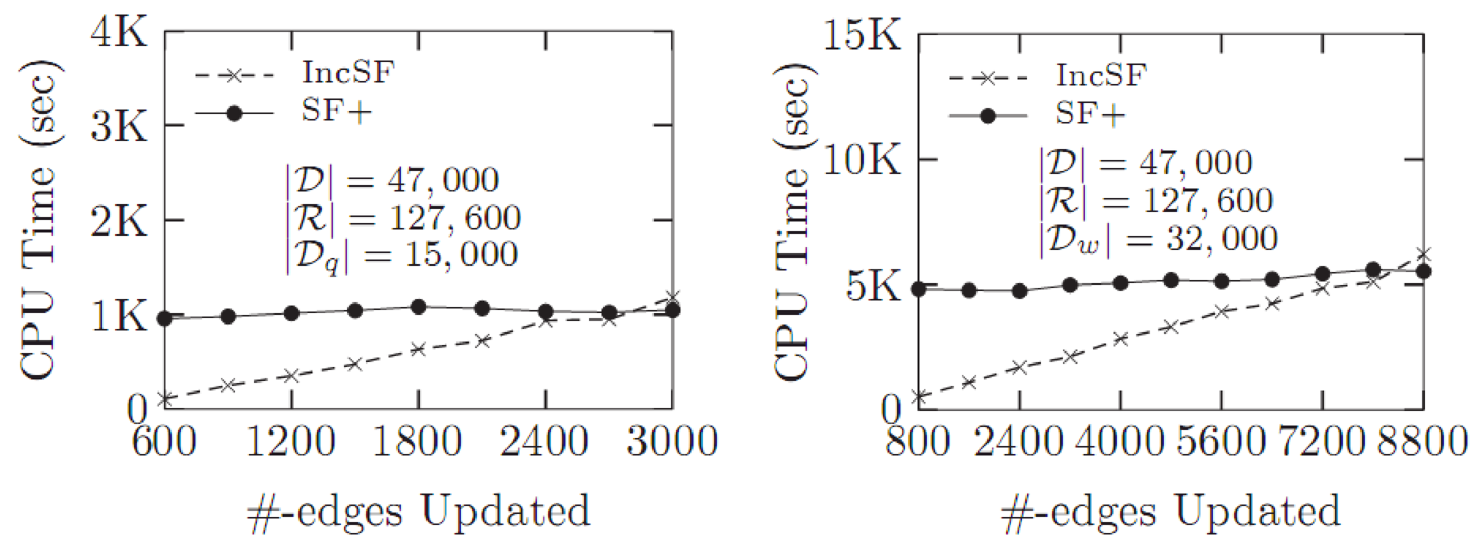
SF+ outperforms the other approaches, due to the use of $\sigma_{max}(T_k)$

On WEBKB



Experiment (3): Edge Updates

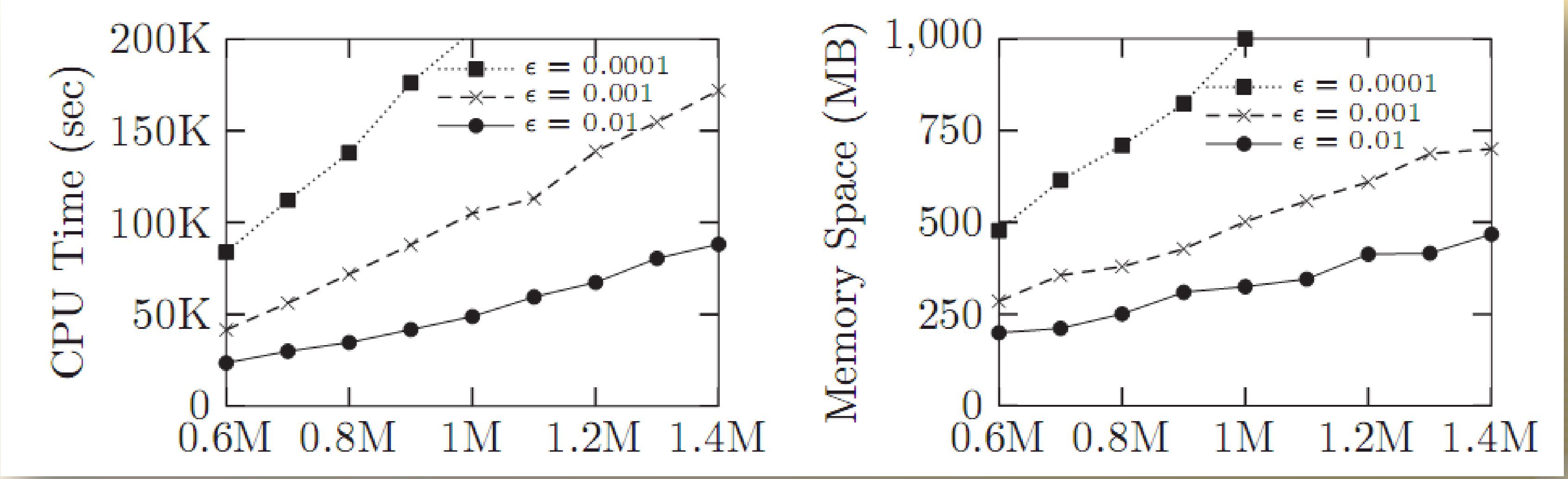
Varying δ



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 !