Co-Simmate: Quick Retrieving All Pairwise Co-Simrank Scores

Weiren Yu, **Julie A. McCann**
Department of Computing,
Imperial College London, UK
{weiren.yu, j.mccann}@imperial.ac.uk

Abstract

Co-Simrank is a useful Simrank-like measure of similarity based on graph structure. The existing method iteratively computes each pair of Co-Simrank score from a dot product of two Pagerank vectors, entailing $\mathcal{O}(\log(1/\epsilon)n^3)$ time to compute all pairs of Co-Simranks in a graph with n nodes, to attain a desired accuracy ϵ. In this study, we devise a model, Co-Simmate, to speed up the retrieval of all pairs of Co-Simranks to $\mathcal{O}(\log_2(\log(1/\epsilon))n^3)$ time. Moreover, we show the optimality of Co-Simmate among other hop-(u^k) variations, and integrate it with a matrix decomposition based method on singular graphs to attain higher efficiency. The viable experiments verify the superiority of Co-Simmate to others.

1 Introduction

Many NLP applications require a pairwise graph-based similarity measure. Examples are bilingual lexicon extraction (Laws et al., 2010), sentiment analysis (Scheible and Schütze, 2013), synonym extraction (Minkov and Cohen, 2014), named entity disambiguation (Alhelbawy and Gaizauskas, 2014), acronym expansion (Zhang et al., 2011). Recently, Co-Simrank (Rothe and Schütze, 2014) becomes an appealing graph-theoretical similarity measure that integrates both features of Simrank (Jeh and Widom, 2002) and Pagerank (Berkhin, 2005). Co-Simrank works by weighing all the number of connections between two nodes to evaluate how similar two nodes are. The intuition behind Co-Simrank is that "*more similar nodes are likely to be pointed to by other similar nodes*".

Co-Simrank is defined in a recursive style:

$$\mathbf{S} = c\mathbf{A}^T\mathbf{S}\mathbf{A} + \mathbf{I}, \quad (1)$$

where $\mathbf{S}$ is *the exact Co-Simrank matrix*, $\mathbf{A}$ is the column-normalised adjacency matrix of the graph, c is a decay factor, and $\mathbf{I}$ is an identity matrix.

The best-known method by (Rothe and Schütze, 2014) computes a single element of $\mathbf{S}$ iteratively from a dot product $\langle *, * \rangle$ of two Pagerank vectors:

$$\mathbf{S}_k(a,b) = c^k \langle \mathbf{p}_k(a), \mathbf{p}_k(b) \rangle + \mathbf{S}_{k-1}(a,b) \quad (2)$$

where $\mathbf{p}_k(a)$ is *a Pagerank vector*, defined as

$$\mathbf{p}_k(a) = \mathbf{A}^T\mathbf{p}_{k-1}(a) \text{ with } \mathbf{p}_0(a) = \mathbf{I}(*, a) \quad (3)$$

This method is highly efficient when only a small fraction of pairs of Co-Simranks need computing because there is no need to access the entire graph for computing only a single pair score. However, partial pairs retrieval is insufficient for many real-world applications (Zhou et al., 2009; Yu et al., 2012a; Zwick, 2002; Leicht et al., 2006) which require all-pairs scores. Let us look at two examples.

a) Co-Citation Analysis. In a co-citation network, one wants to retrieve the relevance between *any* two given documents *at any moment* based on their references. To answer such an *ad-hoc* query, quantifying scores of all document-pairs provides a comprehensive way to show where low and high relevance of pairwise documents may exist (Li et al., 2010; Yu et al., 2014; Haveliwala, 2002).

b) Water Burst Localization. In a water network, nodes denote deployed pressure sensor locations, and edges are pipe sections that connect the nodes. To determine the burst location, one needs to evaluate "proximities" of all pairs of sensor nodes first, and then compare all these "proximities" with the difference in the arrival times of the burst transient at sensor locations, to find the sensor node nearest to the burst event. (Srirangarajan and Pesch, 2013; Srirangarajan et al., 2013; Stoianov et al., 2007)

Hence, the retrieval of all pairwise Co-Simranks is very useful in many applications. Unfortunately, when it comes to *all pairs* computation of $\mathbf{S}(*,*)$, the way of (2) has no advantage over the naive way

$$\mathbf{S}_k = c\mathbf{A}^T\mathbf{S}_{k-1}\mathbf{A} + \mathbf{I} \text{ with } \mathbf{S}_0 = \mathbf{I} \quad (4)$$

as both entail $\mathcal{O}(\log(1/\epsilon)n^3)$ time to compute all pairs of Co-Simranks to attain desired accuracy ϵ.

The complexity $\mathcal{O}(\log(1/\epsilon)n^3)$ has two parts: The first part $\mathcal{O}(n^3)$ is for matrix multiplications $(\mathbf{A}^T\mathbf{S}_{k-1}\mathbf{A})$ at each step. A careful implementation, *e.g.,* partial sums memoisation (Lizorkin et al., 2010) or fast matrix multiplications (Yu et al., 2012b),[1] can optimise this part further to $\mathcal{O}(dn^2)$ or $\mathcal{O}(n^{\log_2 7})$, with d the average graph degree. The second part $\mathcal{O}(\log(1/\epsilon))$ is the total number of steps required to guarantee a given accuracy ϵ, because, as implied by (Rothe and Schütze, 2014),

$$|\mathbf{S}_k(a,b) - \mathbf{S}(a,b)| \leq c^{k+1}. \quad \forall a,b, \quad \forall k \tag{5}$$

To the best of our knowledge, there is a paucity of work on optimising the second part $\mathcal{O}(\log(1/\epsilon))$. Yu et al. (2012b) used a successive over-relaxation (SOR) method to reduce the number of steps for Simrank, which is also applicable to Co-Simrank. However, this method requires a judicious choice of an internal parameter (*i.e.*, relaxation factor ω), which is hard to determine a-priori. Most recently, Yu et al. (2015) propose an exponential model to speed up the convergence of Simrank:

$$\bar{\mathbf{S}}_0 = \exp(-c)\cdot\mathbf{I}, \qquad \mathrm{d}\bar{\mathbf{S}}_t/\mathrm{d}t = \mathbf{A}^T\cdot\mathbf{S}\cdot\mathbf{A}.$$

However, $\bar{\mathbf{S}}$ and $\mathbf{S}$ do not produce the same results. Thus, this exponential model, if used to compute Co-Simrank, will lose some ranking accuracy.

Contributions. In this paper, we propose an efficient method, Co-Simmate, that computes all pairs of Co-Simranks in just $\mathcal{O}(\log_2(\log(1/\epsilon))n^3)$ time, without any compromise in accuracy. In addition, Co-Simmate is parameter-free, and easy to implement. It can also integrate the best-of-breed matrix decomposition based method by Yu and McCann (2014) to achieve even higher efficiency.

2 Co-Simmate Model

First, we provide the main idea of Co-Simmate.

We notice that Co-Simrank solution $\mathbf{S}$ in (1) is expressible as a matrix series:

$$\begin{aligned}\mathbf{S} = \mathbf{I} &+ c\mathbf{A}^T\mathbf{A} + c^2(\mathbf{A}^T)^2\mathbf{A}^2 \\ &+ c^3(\mathbf{A}^T)^3\mathbf{A}^3 + c^4(\mathbf{A}^T)^4\mathbf{A}^4 + \cdots\end{aligned} \tag{6}$$

The existing iterative method (4) essentially uses the following association to compute (6):

$$\mathbf{S} = \Big(c\mathbf{A}^T\overbrace{\Big(c\mathbf{A}^T\underbrace{(c\mathbf{A}^T\mathbf{A}+\mathbf{I})}_{=\mathbf{S}_1}\mathbf{A}+\mathbf{I}\Big)}^{=\mathbf{S}_2}\mathbf{A}+\mathbf{I}\Big)+\cdots \tag{7}$$

The downside of this association is that the resulting $\mathbf{S}_{k-1}$ of the last step can be reused only *once* to compute $\mathbf{S}_k$. Thus, after k iterations, $\mathbf{S}_k$ in (4) grasps only the first k-th partial sums of $\mathbf{S}$ in (6).

To speed up the computation, we observe that (6) can be reorganised as follows:

$$\begin{aligned}\mathbf{S} =& \Big(\mathbf{I}+c\mathbf{A}^T\mathbf{A}\Big) + \Big(c^2(\mathbf{A}^T)^2\mathbf{A}^2 + c^3(\mathbf{A}^T)^3\mathbf{A}^3\Big) + \\ &+ \Big(c^4(\mathbf{A}^T)^4\mathbf{A}^4 + \cdots + c^7(\mathbf{A}^T)^7\mathbf{A}^7\Big) + \cdots \\ =& \Big(\mathbf{I}+c\mathbf{A}^T\mathbf{A}\Big) + \Big(c^2(\mathbf{A}^T)^2(\mathbf{I}+c\mathbf{A}^T\mathbf{A})\mathbf{A}^2\Big) + \\ &+ \Big(c^4(\mathbf{A}^T)^4\Big(\mathbf{I}+c\mathbf{A}^T\mathbf{A}+\cdots+c^3(\mathbf{A}^T)^3\mathbf{A}^3\Big)\mathbf{A}^4\Big) + \cdots\end{aligned}$$

Thereby, we can derive the following novel association, referred to as Co-Simmate, to compute (6):

$$\begin{aligned}\mathbf{S} =& \underbrace{\Big(\overbrace{(\mathbf{I}+c\mathbf{A}^T\mathbf{A})}^{=\mathbf{R}_1} + (c\mathbf{A}^T)^2\overbrace{(\mathbf{I}+c\mathbf{A}^T\mathbf{A})}^{=\mathbf{R}_1}\mathbf{A}^2\Big)}_{=\mathbf{R}_2} + \\ &(c\mathbf{A}^T)^4\underbrace{\Big((\mathbf{I}+c\mathbf{A}^T\mathbf{A}) + (c\mathbf{A}^T)^2(\mathbf{I}+c\mathbf{A}^T\mathbf{A})\mathbf{A}^2\Big)}_{=\mathbf{R}_2}\mathbf{A}^4 + \cdots\end{aligned} \tag{8}$$

There are two advantages of our association: one is that the resulting $\mathbf{R}_{k-1}$ from the last step can be reused *twice* to compute $\mathbf{R}_k$. Hence, $\mathbf{R}_k$ can grasp the first (2^k-1)-th partial sums[2] of $\mathbf{S}$ in (6). Another merit is that $\mathbf{A}^{2^k}$ can be obtained from the result of squaring $\mathbf{A}^{2^{k-1}}$, *e.g.,* $\mathbf{A}^4 = (\mathbf{A}^2)^2$. With these advantages, Co-Simmate can compute all pairs of scores much faster.

Next, let us formally introduce Co-Simmate:

Definition 1. *We call $\mathbf{R}_k$ a Co-Simmate matrix at k-th step if it is iterated as*

$$\begin{cases} \mathbf{R}_0 = \mathbf{I}, \qquad \mathbf{A}_0 = \mathbf{A} \\ \mathbf{R}_{k+1} = \mathbf{R}_k + c^{2^k}(\mathbf{A}_k{}^T\mathbf{R}_k\mathbf{A}_k) \\ \mathbf{A}_{k+1} = \mathbf{A}_k{}^2 \end{cases} \tag{9}$$

By successive substitution in (9), one can verify that $\lim_{k\to\infty}\mathbf{R}_k$ is the exact solution of $\mathbf{S}$ in (6). More precisely, the following theorem shows that, at step k, how many first terms of $\mathbf{S}$ in (6) can be grasped by $\mathbf{R}_k$, showing the fast speedup of (9).

Theorem 1. *Let $\mathbf{R}_k$ be the Co-Simmate matrix in* (9)*, and $\mathbf{S}_k$ the Co-Simrank matrix in* (4)*. Then,*

$$\mathbf{R}_k = \mathbf{S}_{2^k-1} \qquad \forall k = 0,1,2,\cdots \tag{10}$$

[1] These Simranks methods also suit Co-Simranks.

[2] This amount of the first partial sums will be proved later.

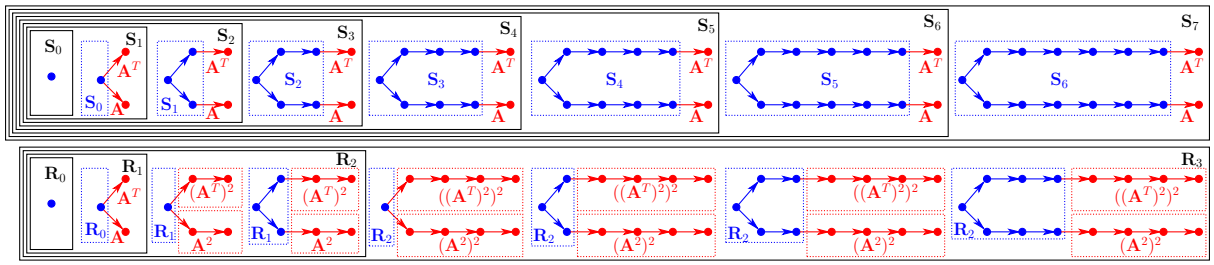


Figure 1: Co-Simmate speeds up Co-Simrank by aggregating more first terms of $\mathbf{S}$ in (6) at each step

Proof. Successive substitution in (4) produces

$$\mathbf{S}_k = \sum_{i=0}^{k} c^i (\mathbf{A}^i)^T \mathbf{A}^i \tag{11}$$

Thus, proving (10) is equivalent to showing that

$$\mathbf{R}_k = \sum_{i=0}^{2^k-1} c^i (\mathbf{A}^i)^T \mathbf{A}^i \tag{12}$$

To show (12), we will use induction on k.

1. For $k = 0$, we have $\mathbf{R}_0 = \mathbf{I} = c^0(\mathbf{A}^0)^T\mathbf{A}^0$.
2. When $k > 0$, we assume that (12) holds for k, and want to prove that (12) holds for $k+1$.

From $\mathbf{A}_{k+1} = {\mathbf{A}_k}^2$ and $\mathbf{A}_0 = \mathbf{A}$ follows that

$$\mathbf{A}_k = {\mathbf{A}_{k-1}}^2 = {\mathbf{A}_{k-2}}^{2^2} = \cdots = \mathbf{A}^{2^k} \tag{13}$$

Plugging $\mathbf{R}_k$ (12) and $\mathbf{A}_k$ (13) into (9) yields

$$\begin{aligned}
&\mathbf{R}_{k+1} = \{\text{using (12) and (13)}\} \\
&= \mathbf{R}_k + c^{2^k} \left(\mathbf{A}^{2^k}\right)^T \left(\sum_{i=0}^{2^k-1} c^i (\mathbf{A}^i)^T \mathbf{A}^i \right) \mathbf{A}^{2^k} \\
&= \mathbf{R}_k + \sum_{i=0}^{2^k-1} c^{i+2^k} (\mathbf{A}^{i+2^k})^T \mathbf{A}^{i+2^k} \\
&= \mathbf{R}_k + \sum_{j=2^k}^{2^k-1+2^k} c^j (\mathbf{A}^j)^T \mathbf{A}^j \\
&= \sum_{j=0}^{2^{k+1}-1} c^j (\mathbf{A}^j)^T \mathbf{A}^j
\end{aligned}$$

Lastly, coupling (11) and (12) concludes (10). □

Theorem 1 implies that, at each step k, $\mathbf{R}_k$ in (9) can grasp the first (2^k-1)-th terms of $\mathbf{S}$, whereas $\mathbf{S}_k$ in (4) can grasp only the first k-th terms of $\mathbf{S}$. Thus, given the number of steps K, Co-Simmate is always more accurate than Co-Simrank because $\mathbf{R}_K$ is exponentially closer to $\mathbf{S}$ than $\mathbf{S}_K$ to $\mathbf{S}$.

Convergence Rate. We next provide a quantitative result on how closer $\mathbf{R}_k$ is to $\mathbf{S}$ than $\mathbf{S}_k$ to $\mathbf{S}$.

Theorem 2. *For any given step k, the difference between $\mathbf{R}_k$ and $\mathbf{S}$ can be bounded by*

$$|\mathbf{R}_k(a,b) - \mathbf{S}(a,b)| \leq c^{2^k}, \qquad \forall a, b \tag{14}$$

Proof. The Co-Simrank result in (5) implies that

$$|\mathbf{S}_{2^k-1}(a,b) - \mathbf{S}(a,b)| \leq c^{2^k}, \qquad \forall a, b$$

Plugging (10) into this inequality yields (14). □

Theorem 2 implies that, to attain a desired accuracy ϵ, Co-Simmate (9) takes exponentially fewer steps than Co-Simrank (4) since the total number of steps required for $\mathbf{R}_K$, as implied by (14), is

$$K = \max\{0, \lceil \log_2 \log_c \epsilon \rceil + 1\},$$

in contrast to the $\lceil \log_c \epsilon \rceil$ steps required for $\mathbf{S}_K$.

Total Computational Cost. Though Co-Simmate takes fewer steps than Co-Simrank for a desired ϵ, in each step Co-Simmate (9) performs one more matrix multiplication than Co-Simrank (4). Next, we compare their total computational time.

Theorem 3. *To guarantee a desired accuracy ϵ, the total time of Co-Simmate* (9) *is exponentially faster than that of Co-Simrank* (4).

Proof. For $k = 1$, both Co-Simmate (9) and Co-Simrank (4) take 2 matrix multiplications.

For $k > 1$, Co-Simmate (9) takes 3 matrix multiplications (2 for $\mathbf{A}_k^T\mathbf{R}_k\mathbf{A}_k$ and 1 for $\mathbf{A}_k^2$), whilst Co-Simrank (4) takes 2 (only for $\mathbf{A}_k^T\mathbf{S}_k\mathbf{A}_k$).

Let $|\mathfrak{M}|$ be the number of operations for one matrix multiplication. Then, for Co-Simmate (9),

$$\text{(total \# of operations for } \mathbf{R}_k) = 3k|\mathfrak{M}|,$$

whereas for Co-Simrank (4), by Theorem 1,

$$\text{(total \# of operations for } \mathbf{S}_k) = 2(2^k-1)|\mathfrak{M}|.$$

Since $3k|\mathfrak{M}| \leq 2(2^k-1)|\mathfrak{M}|$, $\forall k = 2, 3, \cdots$, we can conclude that the total time of Co-Simmate is exponentially faster than that of Co-Simrank. □

Example. Figure 1 pictorially visualises how Co-Simmate accelerates Co-Simrank computation by aggregating more first terms of $\mathbf{S}$ in (6) each step.

Algorithm 1: Co-Simmate on Singular Graphs

Input : $\mathbf{A}$ – column-normalised adjacency matrix, c – decay factor, ϵ – desired accuracy.

Decompose $\mathbf{A}$ *s.t.* $[\mathbf{V}_r, \mathbf{H}_r^T] \leftarrow$ Gram-Schmidt($\mathbf{A}$).
Compute $\mathbf{P} \leftarrow \mathbf{H}_r^T \mathbf{V}_r$.
Initialise $K \leftarrow \max\{0, \lceil \log_2 \log_c \epsilon \rceil + 1\}$.
Initialise $\mathbf{S}_0 \leftarrow \mathbf{I}_r, \quad \mathbf{P}_0 \leftarrow \mathbf{P}$.
for $k \leftarrow 0, 1, \cdots, K-1$ **do**
  Compute $\mathbf{S}_{k+1} \leftarrow c^{2^k} (\mathbf{P}_k)^T \mathbf{S}_k (\mathbf{P}_k) + \mathbf{S}_k$.
  Compute $\mathbf{P}_{k+1} \leftarrow (\mathbf{P}_k)^2$.
return $\mathbf{S} \leftarrow c\mathbf{H}_r \mathbf{S}_K \mathbf{H}_r^T + \mathbf{I}$.

At k-th step, Co-Simrank $\mathbf{S}_k$ connects only two new *hop-1* paths with the old retrieved paths $\mathbf{S}_{k-1}$, whereas Co-Simmate $\mathbf{R}_k$ connects two new *hop-*(2^k) paths (by squaring the old hop-(2^{k-1}) paths) with the old retrieved paths $\mathbf{R}_{k-1}$. Consequently, in each step of Co-Simrank, Co-Simmate is exponential steps faster than Co-Simrank. Moreover, the speedup is more obvious as k grows. □

Optimality of Co-Simmate. To compute $\mathbf{S}$ in (6), besides the prior association methods (7) and (8), the following association can also be adopted:

$$\mathbf{S} = \overbrace{\left(\mathbf{I} + c\mathbf{A}^T\mathbf{A} + c^2(\mathbf{A}^T)^2\mathbf{A}^2\right)}^{=\mathbf{T}_1} + c^3(\mathbf{A}^T)^3 \underbrace{\left(\mathbf{I} + c\mathbf{A}^T\mathbf{A} + c^2(\mathbf{A}^T)^2\mathbf{A}^2\right)}_{=\mathbf{T}_1} \mathbf{A}^3 + \cdots \quad (15)$$

More generally, we can write the following model that covers (8) and (15) as special cases:

$$\begin{cases} \mathbf{R}_0^{(u)} = \mathbf{I}, \qquad \mathbf{A}_0 = \mathbf{A} \\ \mathbf{R}_{k+1}^{(u)} = \mathbf{R}_k^{(u)} + c^{u^k} \cdot \mathbf{A}_k^T \cdot \mathbf{R}_k^{(u)} \cdot \mathbf{A}_k \\ \qquad\qquad + c^{2\cdot u^k} \cdot (\mathbf{A}_k{}^2)^T \cdot \mathbf{R}_k^{(u)} \cdot \mathbf{A}_k{}^2 + \cdots + \\ \qquad\qquad + c^{(u-1)\cdot u^k} \cdot (\mathbf{A}_k{}^{u-1})^T \cdot \mathbf{R}_k^{(u)} \cdot \mathbf{A}_k{}^{u-1} \\ \mathbf{A}_{k+1} = \mathbf{A}_k{}^u \ \ (u = 2, 3, \cdots) \end{cases}$$

$\mathbf{R}_k^{(u)}$ is *a hop-*(u^k) *Co-Simmate matrix* at step k. $\mathbf{R}_k^{(u)}$ becomes Co-Simmate $\mathbf{R}_k$ in (8) when $u = 2$; and reduces to $\mathbf{T}_k$ in (15) when $u = 3$. For all u, it is easy to verify that $\lim_{k\to\infty} \mathbf{R}_k^{(u)} = \mathbf{S}$. Below, we show that Co-Simmate (8) $(u = 2)$ is optimal.

Theorem 4. *To attain a desired accuracy ϵ, the total time of Co-Simmate* (8) *is minimum among all hop-(u^k) Co-Simmate models $\mathbf{R}_k^{(u)} (u = 2, 3, \cdots)$.*

Proof. Similar to Theorem 1, we can show that

$$|\mathbf{R}_k^{(u)}(a, b) - \mathbf{S}(a, b)| \le c^{u^k}, \ \ \forall a, b, \ \forall u \quad (16)$$

Thus, given ϵ, the total number of steps for $\mathbf{R}_K^{(u)}$ is

$$K = \max\{0, \lceil \log_u \log_c \epsilon \rceil + 1\}.$$

For each step k, for hop-(u^k) Co-Simmate $\mathbf{R}_k^{(u)}$,

(# of operations) $= ((u-1) + \sum_{i=0}^{u-2} i)|\mathfrak{M}| = \frac{(u-1)u}{2}|\mathfrak{M}|$.

Therefore, the total time of computing $\mathbf{R}_k^{(u)}$ is

$$\mathcal{O}(\max\{0, \lceil \log_u \log_c \epsilon \rceil + 1\}(u-1)u|\mathfrak{M}|).$$

This complexity is increasing with $u = 2, 3, \cdots$. Thus, Co-Simmate (8) $(u = 2)$ is minimum. □

Incorporate Co-Simmate into Singular Graphs. Co-Simmate (9) can also be combined with other factorisation methods, *e.g.*, Sig-SR, a Co-Simrank algorithm proposed by (Yu and McCann, 2014), to speed up all pairs of Co-Simrank computation from $\mathcal{O}(rn^2 + Kr^3)$ to $\mathcal{O}(rn^2 + (\log_2 K)r^3)$ time further on a singular graph with rank r for K steps. The enhanced Sig-SR is shown in Algorithm 1.

3 Experiments

3.1 Experimental Settings

Datasets. We use both real and synthetic datasets. Three real graphs (Twitter, Email, Facebook) are taken from SNAP (Leskovec and Sosič, 2014).

1) Twitter is a who-follows-whom social graph crawled from the entire Twitter site. Each node is a user, and each edge represents a social relation.

2) Email is an Email communication network from Enron. If an address i sent at least one email to address j, there is a link from i to j.

3) FB contains 'circles' (or 'friends lists') from Facebook. This dataset is collected from the survey participants using the Facebook app, including node features (profiles), circles, and ego networks.

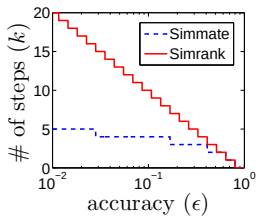
The statistics of these datasets are as follows:

Datasets	# edges	# nodes	ave degree
Twitter	1,768,149	81,306	21.70
Email	183,831	36,692	5.01
FB	88,234	4,039	21.84

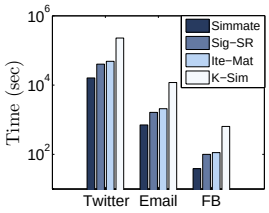
To build synthetic data, we use Boost toolkit (Lee et al., 2001).We control the number of nodes n and edges m to follow densification power laws (Leskovec et al., 2005; Faloutsos et al., 1999).

Baselines. We compare our Co-Simmate with 1) Ite-Mat (Rothe and Schütze, 2014), a Co-Simrank method using the dot product of Pagerank vectors. 2) K-Sim (Kusumoto et al., 2014), a linearized method modified to Co-Simrank. 3) Sig-SR (Yu and McCann, 2014), a SVD Co-Simrank method.

All experiments are on 64bit Ubuntu 14.04 with Intel Xeon E2650 2.0GHz CPU and 16GB RAM.



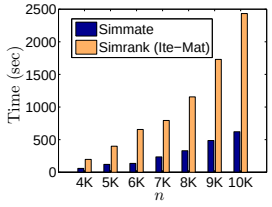
(a) Rate of Convergence (on FB dataset, $c = 0.8$)



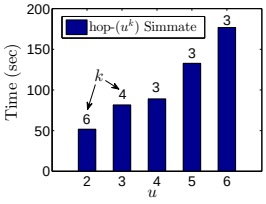
(b) Total Computational Time (on three real datasets, $c = 0.8$)

ϵ	$c = 0.6$		$c = 0.7$		$c = 0.8$	
	SM	SR	SM	SR	SM	SR
0.1	3	4	3	6	4	10
0.01	4	9	4	12	5	20
0.001	4	13	5	19	5	30
0.0001	5	18	5	25	6	41
0.00001	5	22	6	32	6	51

(c) Effect of Damping Factor c on Iterations k (on FB)



(d) Scalability *w.r.t.* # nodes (on 7 synthetic datasets)



(e) Effect of Hop-(u^k) (on FB dataset, $c = 0.8$)

Figure 2: Compare Co-Simmate with Baselines

3.2 Experimental Results

Exp-I. Convergence Rate. We compare the number of steps k needed for Co-Simmate and Co-Simrank (Ite-Mat) to attain a desired accuracy ϵ on Twitter, Email, FB. The results on all the datasets are similar. Due to space limits, Figure 2(a) only reports the result on FB. We can discern that, when ϵ varies from 0.01 to 1, k increases from 1 to 5 for Co-Simmate, but from 1 to 20 for Co-Simrank. The fast convergence rate of Co-Simmate is due to our model that *twice* reuses $\mathbf{R}_{k-1}$ of the last step.

Exp-II. Total Computational Time. Figure 2(b) compares the total computational time of Co-Simmate with 3 best-known methods on real data. The result shows Co-Simmate runs 10x, 5.6x, 4.3x faster than K-Sim, Ite-Mat, Sig-SR, respectively. This is because 1) K-Sim is efficient only when a fraction pair of scores are computed, whereas Co-Simmate can efficiently handle all pairs scores, by twice sharing $\mathbf{R}_{k-1}$ and repeated squaring $\mathbf{A}^{2^{k-1}}$. 2) Co-Simmate grasps exponential new terms of $\mathbf{S}$ per step, but Ite-Mat grasps just 1 new term of $\mathbf{S}$. 3) Sig-SR does not adopt association tricks in the subspace, unlike our methods that integrate (9).

Exp-III. Effect of Damping Factor c. Using real datasets (Twitter, Email, FB), we next evaluate the effect of damping factor c on the number of iterations k to guarantee a given accuracy ϵ. We vary ϵ from 0.1 to 0.00001 and c from 0.6 to 0.8, the results of k on all the datasets are similar. For the interests of space, Figure 2(c) tabularises only the results on FB, where 'SM' columns list the number of iterations required for Co-Simmate, and 'SR' columns lists that for Co-Simrank. From the results, we can see that, for any given ϵ and c, the number of iterations for Co-Simmate is consistently smaller than that for Co-Simrank. Their gap is more pronounced when ϵ becomes smaller or c is increased. This is because, at each iteration, Co-Simmate can grasp far more first terms of $\mathbf{S}$ than Co-Simrank. Thus, for a fixed accuracy, Co-Simmate requires less iterations than Co-Simrank. This is consistent with our analysis in Theorem 2.

Exp-IV. Scalability. By using synthetic datasets, we fix $\epsilon = 0.0001$ and vary n from 4,000 to 10,000. Figure 2(d) depicts the total time of Co-Simmate and Ite-Mat. We can notice that, as n grows, the time of Co-Simmate does not increase so fast as Co-Simrank. The reason is that the number of steps of Co-Simmate is greatly cut down by twice $\mathbf{R}_{k-1}$ sharing and $\mathbf{A}^{2^{k-1}}$ memoisation.

Exp-V. Effect of Hop-u^k. Finally, we test the impact of u on the total time of our hop-(u^k) Co-Simmate variations on real datasets. Due to similar results, Figure 2(e) merely reports the results on FB. It can be observed that, as u grows from 2 to 6, the total number of steps for hop-(u^k) Co-Simmate decreases, but their total time still grows. This is because, in each step, the cost of hop-(u^k) Co-Simmate is increasing with u. Thus, the lowest cost is Co-Simmate when $u = 2$.

4 Conclusions

We propose an efficient algorithm, Co-Simmate, to speed up all pairs Co-Simranks retrieval from $\mathcal{O}(\log(1/\epsilon)n^3)$ to $\mathcal{O}(\log_2(\log(1/\epsilon))n^3)$ time, to attain a desired accuracy ϵ. Besides, we integrate Co-Simmate with Sig-SR on singular graphs to attain higher efficacy. The experiments show that Co-Simmate can be 10.2x faster than the state-of-the-art competitors. As future work, we will incorporate our partial-pairs Simrank (Yu and McCann, 2015) into partial-pairs Co-Simmate search.

Acknowledgement. This research is supported by NEC Smart Water Network research project.

References

Ayman Alhelbawy and Robert J. Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 75–80.

Pavel Berkhin. 2005. Survey: A survey on PageRank computing. *Internet Mathematics*, 2(1):73–120.

Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 1999. On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 1999)*, pages 251–262.

Taher H Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web (WWW 2002)*, pages 517–526. ACM.

Glen Jeh and Jennifer Widom. 2002. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, pages 538–543.

Mitsuru Kusumoto, Takanori Maehara, and Ken-ichi Kawarabayashi. 2014. Scalable similarity search for SimRank. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD 2014)*, pages 325–336.

Florian Laws, Lukas Michelbacher, Beate Dorow, Christian Scheible, Ulrich Heid, and Hinrich Schütze. 2010. A linguistically grounded graph model for bilingual lexicon extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010, Poster)*, pages 614–622.

Lie-Quan Lee, Andrew Lumsdaine, and Jeremy G Siek. 2001. The boost graph library. `http://www.boost.org/`.

E. A. Leicht, Petter Holme, and M. E. J. Newman. 2006. Vertex similarity in networks. *Physical Review E*, 73(2):026120.

Jure Leskovec and Rok Sosič. 2014. SNAP: A general purpose network analysis and graph mining library in C++. `http://snap.stanford.edu/snap`, June.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (SIGKDD 2005)*, pages 177–187. ACM.

Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of SimRank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT 2010)*, pages 465–476.

Dmitry Lizorkin, Pavel Velikhov, Maxim N. Grinev, and Denis Turdakov. 2010. Accuracy estimate and optimization techniques for SimRank computation. *The VLDB Journal (The International Journal on Very Large Data Bases)*, 19(1):45–66.

Einat Minkov and William W. Cohen. 2014. Adaptive graph walk-based similarity measures for parsed text. *Natural Language Engineering*, 20(3):361–397.

Sascha Rothe and Hinrich Schütze. 2014. CoSimRank: A flexible & efficient graph-theoretic similarity measure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 1392–1402.

Christian Scheible and Hinrich Schütze. 2013. Sentiment relevance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 954–963.

Seshan Srirangarajan and Dirk Pesch. 2013. Source localization using graph-based optimization technique. In *IEEE Wireless Communications and Networking Conference (WCNC 2013)*, pages 1127–1132.

Seshan Srirangarajan, Michael Allen, Ami Preis, Mudasser Iqbal, HockBeng Lim, and AndrewJ. Whittle. 2013. Wavelet-based burst event detection and localization in water distribution systems. *Journal of Signal Processing Systems*, 72(1):1–16.

Ivan Stoianov, Lama Nachman, Steve Madden, Timur Tokmouline, and M Csail. 2007. PIPENET: A wireless sensor network for pipeline monitoring. In *The 6th International Symposium on Information Processing in Sensor Networks (IPSN 2007)*, pages 264–273.

Weiren Yu and Julie A. McCann. 2014. Sig-SR: SimRank search over singular graphs. In *Proceedings of the 37th ACM SIGIR International Conference on Research & Development in Information Retrieval (SIGIR 2014)*, pages 859–862.

Weiren Yu and Julie A McCann. 2015. Efficient partial-pairs SimRank search on large networks. *Proceedings of the VLDB Endowment (PVLDB 2015)*, 8(5):569–580.

Weiren Yu, Xuemin Lin, Wenjie Zhang, Ying Zhang, and Jiajin Le. 2012a. SimFusion+: Extending SimFusion towards efficient estimation on large and dynamic networks. In *Proceedings of the 35th ACM SIGIR International Conference on Research & Development in Information Retrieval (SIGIR 2012)*, pages 365–374.

Weiren Yu, Wenjie Zhang, Xuemin Lin, Qing Zhang, and Jiajin Le. 2012b. A space and time efficient algorithm for SimRank computation. *World Wide Web*, 15(3):327–353.

Weiren Yu, Xuemin Lin, and Wenjie Zhang. 2014. Fast incremental SimRank on link-evolving graphs. In *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE 2014)*, pages 304–315.

Weiren Yu, Xuemin Lin, Wenjie Zhang, and Julie A. McCann. 2015. Fast all-pairs SimRank assessment on large graphs and bipartite domains. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 27(7):1810–1823.

Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. 2011. Entity linking with effective acronym expansion, instance selection and topic modeling. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1909–1914.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural / attribute similarities. *Proceedings of the VLDB Endowment (PVLDB)*, 2(1):718–729.

Uri Zwick. 2002. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM (JACM)*, 49(3):289–317.