Sig-SR: SimRank Search over Singular Graphs

Weiren Yu[†], Julie A. McCann[†]
[†]Department of Computing, Imperial College London, UK
{weiren.yu, jamm}@imperial.ac.uk

ABSTRACT

SimRank is an attractive structural-context measure of similarity between two objects in a graph. It recursively follows the intuition that "two objects are similar if they are referenced by similar objects". The best known matrix-based method [1] for calculating SimRank, however, implies an assumption that the graph is *non-singular*, *i.e.,* its adjacency matrix is invertible. In reality, non-singular graphs are very rare; such an assumption in [1] is too restrictive in practice. In this paper, we provide a treatment of [1], by supporting similarity assessment on *non-invertible* adjacency matrices. Assume that a singular graph G has n nodes, with r ($< n$) being the rank of its adjacency matrix. (1) We show that SimRank matrix $\mathbf{S}$ on G has an elegant structure: $\mathbf{S}$ can be represented as a rank r matrix plus a scaled identity matrix. (2) By virtue of this, an efficient algorithm over singular graphs, Sig-SR, is proposed for calculating all-pairs SimRank in $O(r(n^2 + Kr^2))$ time for K iterations. In contrast, the only known matrix-based algorithm that supports singular graphs [2] needs $O(r^4n^2)$ time. The experimental results on real and synthetic datasets demonstrate the superiority of Sig-SR on singular graphs against its baselines.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Storage and Retrieval

1. INTRODUCTION

Modern means of similarity assessment between objects based on hyperlink structure have played a giant role in web information retrieval. One well-publicized link-based similarity measure is SimRank [3], invented by Jeh and Widom, due to its successful applications in *e.g.,* similarity join [4], k-nearest neighbor search [1], and citation analysis [2, 5]. Analogous to the famous PageRank algorithm [6] assigning a relevance score to each *object* (node), SimRank assigns a similarity score to each *pair* of nodes in a graph. The central idea behind SimRank is that "two nodes are similar if they are referenced by similar nodes", together with the base case that "nodes are maximally similar to themselves", which can be formulated in the matrix notation [1, 2] as follows:

$$\mathbf{S} = C \cdot \mathbf{W}^T \cdot \mathbf{S} \cdot \mathbf{W} + (1 - C) \cdot \mathbf{I}, \tag{1}$$

where $C \in (0, 1)$ is a damping factor, $\mathbf{W}$ is an $n \times n$ column normalized adjacency matrix, $(\star)^T$ is the matrix transpose, $\mathbf{S}$ is the similarity matrix, whose entry $[\mathbf{S}]_{i,j}$ is the SimRank score between nodes i and j, and $\mathbf{I}$ is the identity matrix. The recursive nature of SimRank not only allows its similarity scores to capture the global link structure of a graph, but also inspires research on efficient SimRank calculation since the naive way [3] of solving $\mathbf{S}$ needs $O(Kn^2d^2)$ time on a graph of n nodes and d average in-degree for K iterations. With the massive size of the Internet, recent years have witnessed an increasing attention in SimRank optimization (see [1, 2, 5, 7, 8] and the references therein).

Among those existing matrix-based SimRank algorithms, SimMat [1] is the state-of-the-art one, yielding $O(\alpha n^2)$ time for estimating SimRank, where α is the target rank of the adjacency matrix. However, the optimization method of SimMat is based on the following formula (*i.e.,* Eq.(3) in [1]):

$$\tfrac{1}{C}(\mathbf{W}^T)^{-1}\mathbf{S} - \mathbf{S}\mathbf{W} = \tfrac{1-C}{C}(\mathbf{W}^T)^{-1}, \tag{2}$$

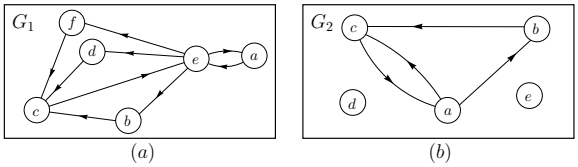
which is derived by left-multiplying both sides of Eq.(1) by $\frac{1}{C}(\mathbf{W}^T)^{-1}$. Unfortunately, this process rests on an underlying assumption that $(\mathbf{W}^T)^{-1}$ must exist (or equivalently, the adjacency matrix must be invertible), but this does not always hold in reality. Indeed, when $(\mathbf{W}^T)^{-1}$ does not exist, SimMat fails to work properly, as depicted in Example 1.

EXAMPLE 1. *Consider two web graphs G_1 and G_2 in Fig.1, where nodes are web pages, and edges are hyperlinks. We want to calculate SimRank scores of all-pairs on G_1 and G_2, respectively. However, we observe that SimMat [1] fails on both graphs as their adjacency matrices are not invertible:* [1] *$rank(\boldsymbol{Adj}(G_1)) = 3 < 6$, $rank(\boldsymbol{Adj}(G_2)) = 3 < 5$. Thus, there does not exist $(\mathbf{W}^T)^{-1}$ for G_1 and G_2 such that Eq.(2) holds, not to mention using the subsequent Eq.(2)-based optimization method in [1] for SimRank calculation.* □

Example 1 suggests that SimMat only works when $(\mathbf{W}^T)^{-1}$ exists. However, as demonstrated by our statistical experiments on SNAP[2], there are over 95% real graphs whose adjacency matrices are non-invertible, *i.e.,* $\text{rank}(\mathbf{W}) < n$ (known as *singular graphs*). Thus, it is imperative to provide

[1]According to linear algebra, for any $n \times n$ matrix $\mathbf{X}$, the following statements are equivalent: $\text{rank}(\mathbf{X}) = n \iff \mathbf{X}^{-1}$ exists.

[2]Stanford Large Network Platform (SNAP) collects many real datasets available at http://snap.stanford.edu/data/index.html.



(a) (b)

Figure 1: SimMat fails on Singular Graphs

a treatment for SimMat, by devising an efficient algorithm that supports SimRank computations on singular graphs. To this end, we study the following problem in this paper:

Problem (SimRank over Singular Graphs)
Given a singular graph G, and a damping factor $C \in (0,1)$.
Compute all-pairs of SimRank scores on G.

It is worth mentioning that there is a special subclass of singular graphs, in which there are many isolated nodes (*e.g.,* G_2 in Fig.1(b)). Singular graphs of this subclass are not the direct focus of our consideration, since they could be further simplified by removing trivial zero rows and columns (associated with isolated nodes) in the adjacency matrices. For example in G_2, once isolated nodes d and e are removed, its adjacency matrix $\mathbf{Adj}(G_2)$ can be reduced to $\mathbf{Adj}(G_2')$:

$$\mathbf{Adj}(G_2) = \begin{array}{c|ccc|cc} & a & b & c & d & e \\ \hline a & 0 & 1 & 1 & 0 & 0 \\ b & 0 & 0 & 1 & 0 & 0 \\ c & 1 & 0 & 0 & 0 & 0 \\ \hline d & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 0 \end{array} \implies \mathbf{Adj}(G_2') = \begin{array}{c|ccc} & a & b & c \\ \hline a & 0 & 1 & 1 \\ b & 0 & 0 & 1 \\ c & 1 & 0 & 0 \end{array}.$$

Since $\mathbf{Adj}(G_2')$ becomes invertible, SimMat is still applicable to the reduced G_2' (though not applicable to the original G_2). Hence, in this scenario, we call G_2 *a "false" singular graph*, a graph whose original non-invertible adjacency matrix can be converted to an invertible one if isolated nodes are removed. In this paper, our attention will be devoted to "true" singular graphs without isolated nodes (*e.g.,* G_1 in Fig.1(a)).

Contributions. Our main contributions are listed below.

- We prove that, on a singular graph with its adjacency matrix rank r, SimRank matrix can be represented as a rank r matrix plus an identity matrix. (Section 3.1)
- We devise an efficient algorithm on a singular graph, Sig-SR, entailing $O(r(n^2 + Kr^2))$ time to calculate all-pairs SimRank for K iterations. (Section 3.2)
- We empirically show that Sig-SR enables SimMat [1] to support SimRank assessment on singular graphs, and runs significantly faster than NI_Sim [2]. (Section 4)

2. RELATED WORK

SimRank is arguably one of the most successful link-based similarity measures in recent years. It was initially proposed by Jeh and Widom [3], who adopted an iterative paradigm to compute SimRank scores of all-pairs in $O(Kn^2d^2)$ time. Since then, there has been a surge of papers looking at various problems in efficient SimRank computing as the naive algorithm [3] has high time complexity. Recent results include matrix-based methods [1, 2], iterative optimization [7–9], random walk sampling [10, 11], incremental updating [12], parallel computing [13], and semantic improvement [14].

In comparison to the work on iterative optimization (*resp.* random walk sampling) that will produce deterministic (*resp.* probabilistic) errors, the work on matrix-based methods [1, 2] may accurately calculate SimRank without loss of exactness. The pioneering work of Li *et al.* [2] proposed a very elegant closed-form for SimRank:

$$vec(\mathbf{S}) = (1-C)(\mathbf{I} - C \cdot (\mathbf{W}^T \otimes \mathbf{W}^T))^{-1} vec(\mathbf{I}), \qquad (3)$$

where $\otimes$ is tensor product, and $vec(\star)$ matrix vectorization. Based on Eq.(3), [2] utilized rank r (*resp.* low-rank α ($< r$)) singular value decomposition to compute all-pairs SimRank in $O(r^4n^2)$ time (*resp.* $O(\alpha^4 n^2)$ with a little accuracy loss). In contrast, our work uses a trick to avoid directly applying matrix factorization to the high-dimensional tensor product, just entailing $O(rn^2)$ time for singular graphs.

Most recently, Fujiwara *et al.* [1] proposed SimMat to further speed up the computation of n-pairs of SimRank *w.r.t.* a given query node to $O(rn)$ time, thus needing $O(rn^2)$ time for all-pairs SimRank. However, their optimization method is based on Eq.(2), implying an underlying assumption that $(\mathbf{W}^T)^{-1}$ must exist, which does not always hold in general. In comparison, our work provides a treatment for SimMat, aiming to support SimRank computation on singular graphs, whose $(\mathbf{W}^T)^{-1}$ does not exist. Note that the matrix inversion in Eq.(3), different to $(\mathbf{W}^T)^{-1}$ in Eq.(2), always exists as $(\mathbf{I} - C \cdot (\mathbf{W}^T \otimes \mathbf{W}^T))$ is a diagonally dominant matrix.[3]

3. OUR SOLUTION

3.1 Characterizing SimRank on Singular Graphs

Unlike SimMat [1] that left-multiplies both sides of Eq.(1) by $\frac{1}{C}(\mathbf{W}^T)^{-1}$, we first provide the following lemma.

Lemma 1 ([12]). *Given any three $n \times n$ matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, if $\lambda_{\max}(\mathbf{A}) \cdot \lambda_{\max}(\mathbf{B}) < 1$,[4] then there exists a unique solution $\mathbf{X} = \sum_{i=0}^{\infty} \mathbf{A}^i \cdot \mathbf{C} \cdot \mathbf{B}^i$ to the following matrix equation*

$$\mathbf{X} = \mathbf{A} \cdot \mathbf{X} \cdot \mathbf{B} + \mathbf{C}. \qquad (4)$$

Main Idea. Based on Lemma 1, we show our main idea.

Let $\mathbf{A} := C \cdot \mathbf{W}^T$, $\mathbf{B} := \mathbf{W}$, $\mathbf{C} := (1-C) \cdot \mathbf{I}$ in Lemma 1. Then, the SimRank solution $\mathbf{S}$ to Eq.(1) takes the form: [5]

$$\mathbf{S} = (1-C) \cdot \sum_{i=0}^{\infty} C^i \cdot (\mathbf{W}^T)^i \cdot \mathbf{W}^i. \qquad (5)$$

As G is a rank r ($< n$) singular graph, we now apply *Gram–Schmidt method* [15, pp.316] to decompose $\mathbf{W} \in \mathbb{R}^{n \times n}$ as $\mathbf{W} = \mathbf{V}_r \cdot {\mathbf{H}_r}^T$, where $\mathbf{V}_r$ and $\mathbf{H}_r$ are both $n \times r$ matrices, and $\mathbf{V}_r$ is a column orthonormal matrix, *i.e.,* ${\mathbf{V}_r}^T \mathbf{V}_r = \mathbf{I}_r$.[6] Hence, $\mathbf{W}^i$ can be efficiently computed as

$$\mathbf{W}^i = (\mathbf{V}_r \cdot {\mathbf{H}_r}^T)^i = \mathbf{V}_r \cdot \bar{\mathbf{P}}_r \cdot {\mathbf{H}_r}^T \in \mathbb{R}^{n \times n}, \qquad (6)$$
$$\text{with } \bar{\mathbf{P}}_r := {\mathbf{P}_r}^{i-1} \text{ and } \mathbf{P}_r := {\mathbf{H}_r}^T \cdot \mathbf{V}_r \in \mathbb{R}^{r \times r}.$$

It is important to note that the second equality in Eq.(6) plays a critical role in speeding up the computation of $\mathbf{W}^i$. The original method for computing $\mathbf{W}^i$ requires $O(in^3)$ time, including multiplying i times two $n \times n$ matrices. In contrast, the right-hand side of Eq.(6) tells that, once $\mathbf{W}$ is decomposed into two $n \times r$ matrices, the computation of $\mathbf{W}^i$ can be improved to $O(r^2n + (i-1)r^3 + rn^2)$ time with $r < n$, involving (i) $O(r^2n)$ time to compute ${\mathbf{H}_r}^T \cdot \mathbf{V}_r$, whose result is an $r \times r$ matrix, memoized as $\mathbf{P}_r \in \mathbb{R}^{r \times r}$, (ii) $O((i-1)r^3)$

[3] $\mathbf{X}$ is *diagonally dominant* if $|x_{i,j}| \geq \sum_{j \neq i} |x_{i,j}|$, $\forall i$.

[4] $\lambda_{\max}(\mathbf{X})$ is the eigenvalue of the matrix $\mathbf{X}$ with the largest absolute value, *i.e.,* the spectral radius of $\mathbf{X}$.

[5] One can readily verify that $\lambda_{\max}(C \cdot \mathbf{W}^T) \cdot \lambda_{\max}(\mathbf{W}) < 1$, by using the matrix norm property $\lambda_{\max}(\star) \leq \| \star \|$ and the fact that $\mathbf{W}$ is column-normalized ($\|\mathbf{W}\|_1 = \|\mathbf{W}^T\|_\infty \leq 1$).

[6] $\mathbf{I}_r$ is an $r \times r$ identity matrix, as opposed to $\mathbf{I} \in \mathbb{R}^{n \times n}$.

time to compute the power $(i-1)$ of $\mathbf{P}_r$, memoized as $\bar{\mathbf{P}}_r$, and (iii) $O(rn^2)$ time to compute $\mathbf{V}_r \cdot \bar{\mathbf{P}}_r \cdot {\mathbf{H}_r}^T$.

Then, to efficiently compute $\mathbf{S}$, we plug Eq.(6) into Eq.(5):

$$\begin{aligned}\tfrac{1}{1-C}\mathbf{S} - \mathbf{I} &= \sum_{i=1}^{\infty} C^i \cdot \left(\mathbf{H}_r({\mathbf{P}_r}^T)^{i-1}{\mathbf{V}_r}^T\right) \cdot \left(\mathbf{V}_r{\mathbf{P}_r}^{i-1}{\mathbf{H}_r}^T\right) \\ &= \mathbf{H}_r\left(\sum_{i=1}^{\infty} C^i \cdot ({\mathbf{P}_r}^T)^{i-1} \cdot ({\mathbf{V}_r}^T\mathbf{V}_r) \cdot {\mathbf{P}_r}^{i-1}\right){\mathbf{H}_r}^T \\ &= C \cdot \mathbf{H}_r \cdot \mathbf{S}_r \cdot {\mathbf{H}_r}^T, \qquad (7)\end{aligned}$$

$$\text{with } \mathbf{S}_r := \sum_{i=0}^{\infty} C^i \cdot ({\mathbf{P}_r}^T)^i \cdot {\mathbf{P}_r}^i. \qquad (8)$$

Eq.(7) depicts an important feature of matrix $(\frac{1}{1-C}\mathbf{S}-\mathbf{I})$, based on the following theorem.

THEOREM 1. *For a singular graph G with the adjacency rank r ($< n$), its SimRank matrix $\mathbf{S}$ can be characterized as a rank r matrix plus a scaled identity matrix.*

PROOF. As $\mathbf{S}_r$ in Eq.(8) is an $r \times r$ matrix, Eq.(7) says that the matrix $(\frac{1}{1-C}\mathbf{S}-\mathbf{I})$ can be decomposed as the product of $(\mathbf{H}_r \cdot \mathbf{S}_r) \in \mathbb{R}^{n\times r}$ and $(\mathbf{H}_r^T) \in \mathbb{R}^{r\times n}$, meaning that $(\frac{1}{1-C}\mathbf{S}-\mathbf{I})$ is a rank r matrix. Thus, $\mathbf{S}$ on rank r singular graph can be expressed as the rank r matrix $(\mathbf{H}_r \cdot \mathbf{S}_r \cdot \mathbf{H}_r^T)$ plus the scaled identity matrix $(1-C)\cdot\mathbf{I}$. □

3.2 An Efficient Algorithm on Singular Graphs

Characterizing the structure of $\mathbf{S}$ in Theorem 1 is intended to speed up the computation of SimRank on singular graphs.

Specifically, by virtue of Eq.(7), solving $\mathbf{S} \in \mathbb{R}^{n\times n}$ boils down to computing $\mathbf{S}_r \in \mathbb{R}^{r\times r}$ from Eq.(8). Due to its small dimension, $\mathbf{S}_r$ is relatively easier to solve and can thus significantly improve the complexity for computing SimRank. Moreover, we observe that Eq.(8) takes an Eq.(5)-like form. Applying Lemma 1 to Eq.(8), we have the following theorem.

THEOREM 2. *SimRank matrix $\mathbf{S}$ over a rank r singular graph can be represented as*

$$\mathbf{S} = (1-C)\cdot(\mathbf{I} + C \cdot \mathbf{H}_r \cdot \mathbf{S}_r \cdot {\mathbf{H}_r}^T),$$
$$\textit{with} \quad \mathbf{S}_r = C \cdot {\mathbf{P}_r}^T \cdot \mathbf{S}_r \cdot \mathbf{P}_r + \mathbf{I}_r. \qquad (9)$$

PROOF. The correctness of Theorem 2 follows immediately from the combination of Eqs.(7) and (8) and Lemma 1.

One caveat is that the *existence* of $\mathbf{S}_r$ in Eq.(9) is guaranteed by the condition that $C \cdot \lambda_{\max}({\mathbf{P}_r}^T) \cdot \lambda_{\max}(\mathbf{P}_r) < 1$, or equivalently, $\lambda_{\max}(\mathbf{P}_r) \le 1$. This is always true because $\mathbf{W} = \mathbf{V}_r \cdot {\mathbf{H}_r}^T$ implies that

$$\begin{aligned}{\mathbf{V}_r}^T \cdot \mathbf{W} \cdot \mathbf{V}_r &= {\mathbf{V}_r}^T \cdot (\mathbf{V}_r \cdot {\mathbf{H}_r}^T) \cdot \mathbf{V}_r \\ &= ({\mathbf{V}_r}^T \cdot \mathbf{V}_r) \cdot ({\mathbf{H}_r}^T \cdot \mathbf{V}_r) = \mathbf{I}_r \cdot \mathbf{P}_r = \mathbf{P}_r.\end{aligned}$$

This suggests that $\mathbf{P}_r$ is a compression of $\mathbf{W}$ via orthogonal projection $\mathbf{V}_r$ onto a small $r \times r$ subspace. Thus, by Cauchy interlacing theorem, it is follows that $\lambda_{\max}(\mathbf{P}_r) \le \lambda_{\max}(\mathbf{W})$. Combining this with the spectral property that $\lambda_{\max}(\mathbf{W}) \le \|\mathbf{W}\|_1 \le 1$ yields $\lambda_{\max}(\mathbf{P}_r) \le 1$. □

Theorem 2 provides an efficient way of computing SimRank on rank r ($< n$) singular graphs. As depicted in Fig.2, the conventional method computes $\mathbf{S}$ on an $n\times n$ dimensional space via Eq.(1), whereas our approach solves $\mathbf{S}$ by resorting to the computation of $\mathbf{S}_r$ on a small $r \times r$ dimensional space via Eq.(9). Below is the complexity analysis of our solution.

THEOREM 3. *All-pairs SimRank on a rank r singular graph can be computed in $O(rn^2 + Kr^3)$ time for K iterations.*

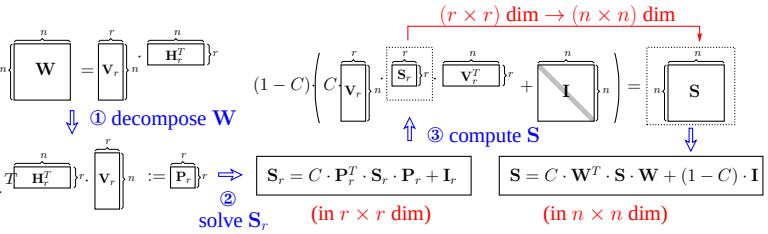


Figure 2: Compute S in $n \times n$ space by resorting to solving $\mathbf{S}_r$ in small $r \times r$ subspace

Algorithm 1: Sig-SR (G, C, K)

Input : singular graph G, damping factor C, iterations K.
Output: SimRank matrix $\mathbf{S}$.

1 Initialize the column normalized adjacency matrix $\mathbf{W}$.
2 Decompose $\mathbf{W}$: $[\mathbf{V}_r, \mathbf{H}_r^T] \leftarrow$ Gram-Schmidt$(\mathbf{W})$.
3 Compute $\mathbf{P}_r \leftarrow {\mathbf{H}_r}^T \cdot \mathbf{V}_r$.
4 Initialize ${\mathbf{S}_r}^{(0)} \leftarrow \mathbf{I}_r$.
5 **for** $k \leftarrow 0, 1, \cdots, K-1$ **do**
6 $\quad {\mathbf{S}_r}^{(k+1)} \leftarrow C \cdot {\mathbf{P}_r}^T \cdot {\mathbf{S}_r}^{(k)} \cdot \mathbf{P}_r + \mathbf{I}_r$.
7 **return** $\mathbf{S} \leftarrow (1-C)\cdot(C \cdot \mathbf{H}_r \cdot {\mathbf{S}_r}^{(K)} \cdot {\mathbf{H}_r}^T + \mathbf{I})$.

To prove Theorem 3, we next provide an algorithm, denoted as Sig-SR, for SimRank assessment on a singular graph.

Algorithm. The algorithm Sig-SR works in three phases:

1) For the preprocessing (lines 1–3), it first applies Gram–Schmidt method [15, pp.316] to factorize $\mathbf{W}$ as $\mathbf{V}_r$ and ${\mathbf{H}_r}^T$, such that the column vectors in $\mathbf{V}_r$ form an orthonormal basis of $\mathbf{W}$ (line 2). Then, utilizing $\mathbf{V}_r$ and ${\mathbf{H}_r}^T$, it calculates auxiliary $\mathbf{P}_r \in \mathbb{R}^{r\times r}$ (line 3) for subsequent iterations.

2) In the computation of $\mathbf{S}_r \in \mathbb{R}^{r\times r}$ (lines 4–6), it adopts a fixed point iteration in $r \times r$ dimension to get $\mathbf{S}_r$ in Eq.(9). It is worth noting that, as $\mathbf{P}_r$ is full-rank (invertible) matrix, other prior optimization methods for computing SimRank (*e.g.*, [1, 7, 13]) can be applied to computing $\mathbf{S}_r$ as well.

3) For calculating $\mathbf{S} \in \mathbb{R}^{n\times n}$ (line 7), it uses $\mathbf{V}_r$ and the resulting $\mathbf{S}_r^{(K)}$, and returns $\mathbf{S}$ as the final result.

Correctness & Complexity. The correctness of $\mathbf{S}$ returned from Sig-SR can be readily verified by Theorem 2.

The total time of Sig-SR is bounded by $O(rn^2 + Kr^3)$, which consists of (i) $O(rn^2 + r^2n)$ time for the preprocessing (lines 1–3), including $O(rn^2)$ time to decompose $\mathbf{W}$ and $O(r^2n)$ time to compute $\mathbf{P}_r$, (ii) $O(Kr^3)$ time for computing $\mathbf{S}_r^{(K)}$ for K iterations (lines 4–6), and (iii) $O(rn^2)$ time for computing $\mathbf{S}$ (line 7). □

EXAMPLE 2. *Recall the singular graph G_1 in Fig.1(a). We illustrate how Sig-SR computes all-pairs SimRank in G_1. As previously used in [2, 8], we set $C = 0.8$ and $K = 10$.*

Firstly, by applying the Gram–Schmidt process, $\mathbf{W} \in \mathbb{R}^{6\times 6}$ can be decomposed as $\mathbf{W} = \mathbf{V}_3 \cdot {\mathbf{H}_3}^T$ (line 2):

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} 0 & 0.707 & 0 \\ 0 & 0 & -0.577 \\ 0 & 0.707 & 0 \\ 0 & 0 & -0.577 \\ -1 & 0 & 0 \\ 0 & 0 & -0.577 \end{bmatrix}}_{\mathbf{V}_3} \cdot \underbrace{\begin{bmatrix} 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0.707 & 0 \\ 0 & 0 & -0.577 & 0 & 0 & 0 \end{bmatrix}}_{{\mathbf{H}_3}^T}.$$

Then, using $\mathbf{V}_3$ and ${\mathbf{H}_3}^T$, we obtain $\mathbf{P}_3 \in \mathbb{R}^{3\times 3}$ (line 3) and ${\mathbf{S}_3}^{(10)} \in \mathbb{R}^{3\times 3}$ (lines 4–6) as follows:

$$\mathbf{P}_3 = \begin{bmatrix} 0 & 0 & 1.732 \\ -0.707 & 0 & 0 \\ 0 & -0.408 & 0 \end{bmatrix}, \quad {\mathbf{S}_3}^{(10)} = \begin{bmatrix} 1.666 & 0 & 0 \\ 0 & 1.666 & 0 \\ 0 & 0 & 5 \end{bmatrix}.$$

Finally, $\mathbf{S} \in \mathbb{R}^{6\times 6}$ *is derived from* $\mathbf{S}_3{}^{(10)} \in \mathbb{R}^{3\times 3}$ *(line 7):*

$$\mathbf{S} = \begin{bmatrix} 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.467 & 0 & 0.267 & 0 & 0.267 \\ 0 & 0 & 0.467 & 0 & 0 & 0 \\ 0 & 0.267 & 0 & 0.467 & 0 & 0.267 \\ 0 & 0 & 0 & 0 & 0.333 & 0 \\ 0 & 0.267 & 0 & 0.267 & 0 & 0.467 \end{bmatrix}. \quad \square$$

4. EXPERIMENTAL EVALUATION

We present an empirical study on real and synthetic data, to show that our algorithm Sig-SR (1) provides an effective treatment for supporting SimMat [1] on singular graphs, and (2) is much faster than other competitors on singular graphs.

Experimental Setting. We used three real datasets.

(1) (CA)-HEPTH, a citation graph from the e-print arXiv, where nodes are papers, and edges are citations.

(2) (SOC)-SLASHDOT, a Slashdot Zoo social network, which contains friend links between the users of Slashdot.

(3) (EMAIL)-EUALL, an email dataset from a EU research institution, where an edge from an email address i to j is a message sent from i to j. We removed all the isolated nodes.

The sizes of these datasets are summarized below.

Dataset	$m = \|E\|$	$n = \|V\|$	r	$d = \|E\|/\|V\|$
CA	51,971	9,877	9,109	5.3
SOC	948,464	82,168	65,431	11.5
EMAIL	348,912	225,409	19,423	1.5

We designed a generator to produce synthetic data with 2 parameters: the number of nodes, and the number of edges. We generated graphs following the densification power law.

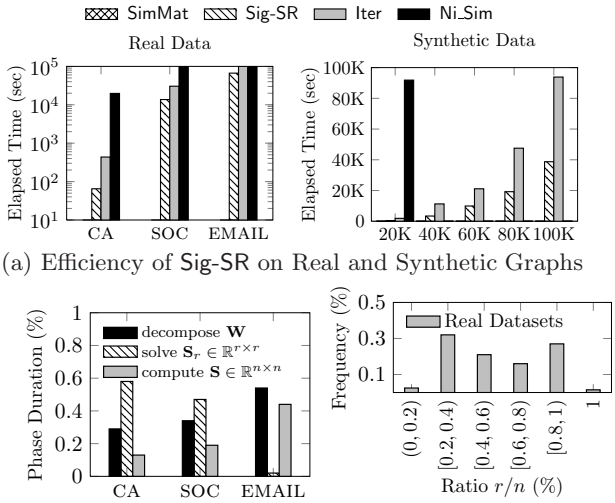
We implemented the following algorithms in Visual C++. 1) Sig-SR. 2) SimMat [1], the best known matrix-based SimRank algorithm. 3) Iter [7], the iterative algorithm computing SimRank via fine-grained memoization. 4) NI_Sim [2], the only known matrix-based SimRank that supports singular graphs via SVD method. All the algorithms are run on Windows 7 with an Intel Core 3.1G CPU and 8G RAM.

As previously used in [2,8], we set damping factor $C = 0.8$, and total iteration number $K = 10$ as default values.

Experimental Results. We next present our findings.

(1) Effectiveness. Fig.3(a) shows the high time efficiency of Sig-SR on real and synthetic datasets. Note that, on real data, y-axis is in log scale; on synthetic data, n is varied from 20K to 100K as shown in x-axis. We see that 1) on singular graphs, Sig-SR works fairly well, whereas SimMat always fails due to the non-invertible adjacency matrices. This tells that Sig-SR, as a treatment for SimMat, enables SimRank assessment on singular graphs. 2) On each dataset, Sig-SR consistently outperforms all other algorithms, *e.g.,* on real CA, Sig-SR (65s) runs 300x faster than NI_Sim (19786s), and 6.7x faster than Iter (436s). When $n > 40$K, NI_Sim explodes due to huge costs for tensor products; in contrast, Sig-SR uses Theorem 1 that avoids tensor products. These verify that our method of computing $\mathbf{S}$ (in $n\times n$) by resorting to the computation of $\mathbf{S}_r$ (in small $r \times r$) is effective.

(2) Phase Duration. Fig.3(b) depicts the time percentage of Sig-SR for three phases (*i.e.,* (I) decompose $\mathbf{W}$, (II) solve $\mathbf{S}_r$, (III) compute $\mathbf{S}$) on real datasets. We can see that Phase (II) is most time-consuming on CA (58%) and SOC (47%), whereas on EMAIL, the time for Phase (II) is negligibly small (2%). The reason is that on EMAIL, r is 11.6x smaller than n, in contrast with CA and SOC (1.25x) where r is only 1.08x and 1.25x smaller, respectively. Thus, when $r \approx n$, the time for Phase (II) is comparable to that for (I) and (III). This confirms our complexity analysis of Sig-SR in Section 3.2.



(a) Efficiency of Sig-SR on Real and Synthetic Graphs

(b) % Time in Each Phase (c) Histogram on Rank r

Figure 3: Experimental Evaluations

(3) Singular Graph Statistics. To verify that singular graphs are commonly existent, Fig.3(c) statistically depicts the frequencies of $\frac{r}{n}$ occurring in certain ranges of ratios (as depicted in x-axis) over 100 real datasets. We see that 1) there are only 2% real graphs whose $\frac{r}{n} = 1$, *i.e.,* non-singular graphs are very rare in reality. Thus, Sig-SR is typically useful in many cases. 2) Almost 32% (*resp.*21%) real graphs have a small rank $r \in [0.2n, 0.4n)$ (*resp.*$r \in [0.4n, 0.6n)$). The smaller r is, the faster Sig-SR runs. This explains why, in general, the better speedup of Sig-SR is guaranteed.

5. CONCLUSIONS

We provide a treatment of SimMat [1], by supporting SimRank assessment on singular graphs. First, we show that SimRank matrix on rank r graph can be represented as a rank r matrix plus a scaled identity matrix. Then, in light of this, an efficient algorithm Sig-SR is proposed. Our empirical evaluations verify the effectiveness of Sig-SR against its competitors over real and synthetic datasets.

6. REFERENCES

[1] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, and M. Onizuka, "Efficient search algorithm for SimRank," in *ICDE*, pp. 589–600, 2013.

[2] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu, "Fast computation of SimRank for static and dynamic information networks," in *EDBT*, 2010.

[3] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity," in *KDD*, pp. 538–543, 2002.

[4] W. Zheng, L. Zou, Y. Feng, L. Chen, and D. Zhao, "Efficient SimRank-based similarity join over large graphs," *PVLDB*, vol. 6, no. 7, pp. 493–504, 2013.

[5] G. He, H. Feng, C. Li, and H. Chen, "Parallel SimRank computation on large graphs with iterative aggregation," in *KDD*, pp. 543–552, 2010.

[6] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," tech. rep., Stanford InfoLab, 1999.

[7] W. Yu, X. Lin, and W. Zhang, "Towards efficient SimRank computation on large networks," in *ICDE*, pp. 601–612, 2013.

[8] D. Lizorkin, P. Velikhov, M. N. Grinev, and D. Turdakov, "Accuracy estimate and optimization techniques for SimRank computation," *VLDB J.*, vol. 19, no. 1, pp. 45–66, 2010.

[9] W. Yu, W. Zhang, X. Lin, Q. Zhang, and J. Le, "A space and time efficient algorithm for SimRank computation," *World Wide Web*, vol. 15, 2012.

[10] D. Fogaras and B. Rácz, "Scaling link-based similarity search," in *WWW*, 2005.

[11] P. Lee, L. V. S. Lakshmanan, and J. X. Yu, "On top-k structural similarity search," in *ICDE*, 2012.

[12] W. Yu, X. Lin, and W. Zhang, "Fast incremental SimRank on link-evolving graphs," in *ICDE*, 2014.

[13] G. He, C. Li, H. Chen, X. Du, and H. Feng, "Using graphics processors for high performance SimRank computation," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 9, pp. 1711–1725, 2012.

[14] W. Yu, X. Lin, W. Zhang, L. Chang, and J. Pei, "More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks," *PVLDB*, vol. 7, no. 1, pp. 13–24, 2013.

[15] C. Meyer, *Matrix Analysis and Applied Linear Algebra*. SIAM, Feb. 2001.