

# Scaling High-Quality Pairwise Link-Based Similarity Retrieval on Billion-Edge Graphs (Full version)

WEIREN YU, University of Warwick, UK

JULIE MCCANN, Imperial College, UK

CHENGYUAN ZHANG, Hunan University, China

HAKAN FERHATOSMANOGLU, University of Warwick, UK

SimRank is an attractive link-based similarity measure, used in fertile fields of Web search and sociometry. However, the existing deterministic method by Kusumoto *et al.* [25] for retrieving SimRank does not always produce high-quality similarity results, as it fails to accurately obtain diagonal correction matrix  $D$ . Moreover, SimRank has a “connectivity trait” problem: increasing the number of paths between a pair of nodes would decrease its similarity score. The best-known remedy, SimRank++ [1], cannot completely fix this problem, since its score would still be zero if there are no common in-neighbors between two nodes.

In this article, we study fast high-quality link-based similarity search on billion-scale graphs. (1) We first devise a “varied- $D$ ” method to accurately compute SimRank in linear memory. We also aggregate duplicate computations, which reduces the time of [25] from quadratic to linear in the number of iterations. (2) We propose a novel “cosine-based” SimRank model to circumvent the “connectivity trait” problem. (3) To substantially speed up the partial-pairs “cosine-based” SimRank search on large graphs, we devise an efficient dimensionality reduction algorithm, PSR<sup>#</sup>, with guaranteed accuracy. (4) We give mathematical insights to the semantic difference between SimRank and its variant, and correct an argument in [25] that “if  $D$  is replaced by a scaled identity matrix  $(1 - \gamma)I$ , their top- $K$  rankings will not be affected much”. (5) We propose a novel method that can accurately convert from Li *et al.*'s SimRank  $\hat{S}$  to Jeh and Widom's SimRank  $S$ . (6) We propose GSR<sup>#</sup>, a generalisation of our “cosine-based” SimRank model, to quantify pairwise similarities across two distinct graphs, unlike SimRank that would assess nodes across two graphs as completely dissimilar. Extensive experiments on various datasets demonstrate the superiority of our proposed approaches in terms of high search quality, computational efficiency, accuracy, and scalability on billion-edge graphs.

CCS Concepts: • **Information systems** → *Web searching and information discovery; Retrieval models and ranking.*

## ACM Reference Format:

Weiren Yu, Julie McCann, Chengyuan Zhang, and Hakan Ferhatosmanoglu. 2021. Scaling High-Quality Pairwise Link-Based Similarity Retrieval on Billion-Edge Graphs (Full version). *ACM Transactions on Information Systems* 1, 1 (November 2021), 47 pages. <https://doi.org/XXXX/XXXXXXXX>

---

Authors' addresses: Weiren Yu, University of Warwick, Coventry, UK, [weiren.yu@warwick.ac.uk](mailto:weiren.yu@warwick.ac.uk); Julie McCann, Imperial College, Department of Computing, London, UK, [j.mccann@imperial.ac.uk](mailto:j.mccann@imperial.ac.uk); Chengyuan Zhang, Hunan University, School of Computer Science and Engineering, Changsha, China, [cyzhang@csu.edu.cn](mailto:cyzhang@csu.edu.cn); Hakan Ferhatosmanoglu, University of Warwick, Coventry, UK, [hakan.f@warwick.ac.uk](mailto:hakan.f@warwick.ac.uk).

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2021 Association for Computing Machinery.

1046-8188/2021/11-ART \$15.00

<https://doi.org/XXXX/XXXXXXXX>

## 1 INTRODUCTION

The Web today is a massive, self-organised, and hyperlinked network. These salient features have brought striking challenges to efficient web data management, thus calling for new search abilities to extract meaningful knowledge automatically from the gigantic Web. Link-based similarity search is a fundamental tool to quantify node-to-node relationships based on topologies [27], with a wide spectrum of successful real applications, e.g., link prediction [5, 31], collaborative filtering [10, 12], community detection [23, 45], co-citation analysis [13, 32], and automatic image annotation [18].

SimRank, conceived by Jeh and Widom [19], is one of the most influential similarity measures. The central theme underpinning SimRank is a simple recursion that “two distinct nodes are assessed as similar if they are in-linked from similar nodes”, together with the base case that “each node is maximally similar to itself”. Mathematically, given a digraph  $G = (V, E)$  with  $|V|$  nodes and  $|E|$  edges, let  $N_a^- = \{x \in V | (x, a) \in E\}$  be the in-neighbor set of node  $a$ , and  $|N_a^-|$  be the cardinality of the set  $N_a^-$ . Then, the SimRank similarity between nodes  $a$  and  $b$  is defined by<sup>1</sup>

$$s(a, b) = \begin{cases} 1 & (a = b) \\ \gamma \cdot \frac{\sum_{(i,j) \in N_a^- \times N_b^-} s(i,j)}{|N_a^-| |N_b^-|} & (a \neq b) \end{cases} \quad (1)$$

where  $\gamma \in (0, 1)$  is a decay factor. Generally,  $\gamma = 0.6$  [34] or  $0.8$  [19], which penalizes long paths relative to short ones.

In contrast to other link-based similarity models, SimRank has the following prominent features: (a) It takes a concise form that captures both direct and indirect neighbors recursively, unlike Bibliographic Coupling [17] and Co-citation [38] that focus only on direct neighbors. (b) It considers structural equivalence of two nodes, whereas Personalized PageRank [20, 24] focuses on reachability from every node to a query. Thus, SimRank has attracted increasing attention recently [6, 14, 34, 54].

### 1.1 Motivation: Undesirable Quality of SimRank Search

Despite much effort devoted to fast computation of SimRank similarities (e.g., [9, 25, 26, 34, 47]), the quality of SimRank search is still less desirable, due to the following two reasons:

**(1) Superfluous Diagonal Correction Error.** One of the most efficient deterministic methods, proposed by Kusumoto *et al.* [25], is based on the following “linearized SimRank formula”:

$$s(a, b) = e_a^\top D e_b + \gamma (P e_a)^\top D (P e_b) + \gamma^2 (P^2 e_a)^\top D (P^2 e_b) + \dots \quad (2)$$

where  $D$  is a precomputed diagonal correction matrix,  $e_a$  is a unit vector with a 1 in the  $a$ -th entry, and  $P$  is the column normalized adjacency matrix, i.e.,  $P_{a,b} = 1/|N_b^-|$  if  $(a, b) \in E$ ; and 0 otherwise.

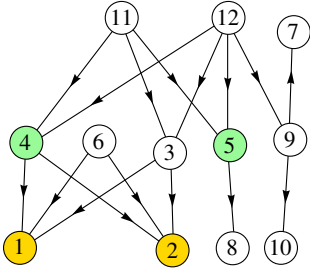
According to [25], before Eq.(2) is computed,  $D$  requires to be determined in advance. However, it is too difficult to compute the exact  $D$  (not to mention within linear memory) since SimRank results have a recursive impact on  $D$ . Note that even Kusumoto *et al.* [25] have not obtained the exact  $D$ , but simply approximated  $D$  by  $\tilde{D} := (1 - \gamma)I$ . Consequently, the computing method by Kusumoto *et al.* [25] implies *the diagonal correction error* as follows:

$$\epsilon_{\text{diag}} := |s(a, b) - s_{\tilde{D}}(a, b)|, \quad (3)$$

where  $s(a, b)$  is the exact solution, and  $s_{\tilde{D}}(a, b)$  is the estimated similarity when  $D$  is replaced by  $\tilde{D}$  in Eq.(2). After  $D$  is estimated, [25] uses an iterative method that sums up only the first  $k$  terms of series  $s_{\tilde{D}}(a, b)$ , denoted as  $s_{\tilde{D}}^{(k)}(a, b)$ . This yields another type of error, namely, *the iterative error*:

$$\epsilon_{\text{iter}} := |s_{\tilde{D}}(a, b) - s_{\tilde{D}}^{(k)}(a, b)| \leq \frac{\gamma^{k+1}}{1-\gamma}. \quad (4)$$

<sup>1</sup>To avoid division by 0 in Eq.(1),  $s(a, b) = 0$  if  $|N_a^-| |N_b^-| = 0$ .



|           |                     | $s(1, 2)$ | $s(4, 5)$ | $s(2, 8)$ | $s(8, 10)$ | $s(3, 9)$ |
|-----------|---------------------|-----------|-----------|-----------|------------|-----------|
| SimRank   | (SR)                | 0.24      | 0.30      | 0.12      | 0.18       | 0.30      |
| SimRank++ | (SR <sup>++</sup> ) | 0.20      | 0.23      | 0         | 0          | 0.15      |
| RoleSim   | (RS)                | 0.34      | 0.24      | 0.05      | 0.07       | 0.12      |
| SimRank#  | (SR <sup>#</sup> )  | 0.38      | 0.24      | 0.14      | 0.10       | 0.17      |

|                      | SimRank | SimRank++ | RoleSim | SimRank# |
|----------------------|---------|-----------|---------|----------|
| $s(1, 2) > s(4, 5)$  | ✗       | ✗         | ✓       | ✓        |
| $s(2, 8) > s(8, 10)$ | ✗       | ✗         | ✗       | ✓        |
| $s(4, 5) > s(3, 9)$  | ✗       | ✓         | ✓       | ✓        |

Fig. 1. SimRank++ (SR<sup>++</sup>) and RoleSim (RS) cannot fix the “connectivity trait” problem of SimRank (SR)

As a result, the total error  $\epsilon_{\text{total}}$  for approximating  $s(a, b)$  by  $s_{\tilde{D}}^{(k)}(a, b)$  consists of two parts:

$$\begin{aligned} \epsilon_{\text{total}} &:= |s(a, b) - s_{\tilde{D}}^{(k)}(a, b)| \leq \underbrace{|s(a, b) - s_{\tilde{D}}(a, b)|}_{\text{Part 1 in Eq.(3)}} + \underbrace{|s_{\tilde{D}}(a, b) - s_{\tilde{D}}^{(k)}(a, b)|}_{\text{Part 2 in Eq.(4)}} \\ &\leq \epsilon_{\text{diag}} + \epsilon_{\text{iter}}. \end{aligned}$$

We argue that  $\epsilon_{\text{diag}}$  is far more serious than  $\epsilon_{\text{iter}}$ . This is because  $\epsilon_{\text{iter}}$  is guaranteed to converge by [25], and will be minimized by increasing the number of iterations,  $k$ . The increase of  $k$ , however, cannot reduce the value of  $\epsilon_{\text{diag}}$ . Worse still, there is no bound on  $\epsilon_{\text{diag}}$  for Eq.(3). The only argument about  $\epsilon_{\text{diag}}$  in [25] is that “estimating  $D$  as  $\tilde{D} := (1 - \gamma)I$  does not much affect the top- $K$  rankings of  $s_{\tilde{D}}(*, *)$  and  $s(*, *)$ ”, but this argument bears a blemish, as will be shown in Section 5.1. Motivated by this, we aim to devise an accurate and fast similarity search method that does not produce the diagonal correction error  $\epsilon_{\text{diag}}$  while avoiding the computation of the exact value of  $D$ .

**(2) “Connectivity Trait” Problem.** Another factor that plagues the quality of SimRank search is the “connectivity trait” problem. That is, increasing the number of *paths* between nodes  $a$  and  $b$  often incurs a contrary decrease in  $s(a, b)$ . However, a paucity of existing works [1, 9, 33] only noticed a special case (1-hop neighbor) of the above phenomenon, *i.e.*, “increasing the number of *common in-neighbors* between nodes  $a$  and  $b$  will decrease  $s(a, b)$ .” The best-known treatment is due to Antonellis *et al.* who proposed SimRank++ [1] that replaces the decay factor  $\gamma$  in Eq.(1) with the following “evidence factor”:

$$\tilde{\gamma} := \gamma(1 - e^{-|N_a^- \cap N_b^-|}) \quad \text{or} \quad \tilde{\gamma} := \gamma \sum_{i=1}^{|N_a^- \cap N_b^-|} \frac{1}{2^i} \quad (5)$$

These revised “evidence factors” have a good property:  $\tilde{\gamma}$  is increasing with respect to  $|N_a^- \cap N_b^-|$ . Therefore, a larger  $\tilde{\gamma}$  implies that there are more common *direct* in-neighbors (*i.e.*, length-2 paths) between  $a$  and  $b$ .

However, we observe a weakness of SimRank++. That is, *SimRank++ score  $\tilde{s}(a, b)$  is always zero if there are no common (direct) in-neighbors between nodes  $a$  and  $b$ .* This is because, by the definition in Eq.(5), if  $N_a^- \cap N_b^- = \emptyset$ , then  $\tilde{\gamma} = 0$ . Thus,  $\tilde{s}(a, b) = 0$ , regardless of how many common  $l$ -hop in-neighbors ( $l > 2$ ) exist between  $a$  and  $b$ . Other pioneering works (*e.g.*, RoleSim [22], PSimRank [9], and MatchSim [33]) for evaluating  $s(a, b)$  also resort to common *direct* in-neighbors between  $a$  and  $b$ , all of which can resolve the special case (1-hop) of the SimRank “connectivity trait” problem (see Related Work in Section 8.2 for more details). However, increasing the number of paths with length  $> 2$  between  $a$  and  $b$  may still lead to a decrease in  $s(a, b)$ .

**EXAMPLE 1.** Consider a real Web graph  $G$  in Figure 1, where each node is a web page, and each edge is a hyperlink. We evaluate the similarity of each node-pair by four measures: (a) SR (Jeh and Widom’s SimRank [19]); (b) SR<sup>++</sup> (SimRank++ [1]); (c) RS (RoleSim [22]); (d) SR<sup>#</sup> (our method). The results are

partly depicted in the table. We notice that  $SR^{++}$  and  $RS$  do not well resolve the  $SR$  “connectivity trait” problem. For example, most people would agree  $s(1, 2) > s(4, 5)$  since node-pair  $(1, 2)$  has 3 common in-neighbors  $\{4, 6, 3\}$ , whereas  $(4, 5)$  has only 2 in-neighbors  $\{11, 12\}$  in common. However, although  $SR^{++}$  narrows the gap between  $s(1, 2)$  and  $s(4, 5)$  due to the use of an “evidence factor”, it gives the same counter-intuitive answer  $s(1, 2) < s(4, 5)$  as  $SR$ .

Another example is the comparison of  $s(2, 8)$  and  $s(8, 10)$ . For  $SR^{++}$ ,  $s(2, 8) = s(8, 10) = 0$ . This is because  $(2, 8)$  has no common direct in-neighbors,  $N_2 \cap N_8 = \emptyset$ ; neither has  $(8, 10)$ . Thereby, their “evidence factors”  $\tilde{\gamma} = 0$ . However, there are 4 indirect path-pairs in-linked from  $(2, 8)$ :

$$\begin{array}{l} 2 \leftarrow 4 \leftarrow \boxed{11} \rightarrow 5 \rightarrow 8, \quad 2 \leftarrow 3 \leftarrow \boxed{11} \rightarrow 5 \rightarrow 8 \\ 2 \leftarrow 4 \leftarrow \boxed{12} \rightarrow 5 \rightarrow 8, \quad 2 \leftarrow 3 \leftarrow \boxed{12} \rightarrow 5 \rightarrow 8 \end{array}$$

as opposed to only 1 from  $(8, 10)$ , i.e.,  $8 \leftarrow 5 \leftarrow \boxed{12} \rightarrow 9 \rightarrow 10$ . Thus, node-pair  $(2, 8)$  has a higher connectivity than  $(8, 10)$ , but this connectivity trait is ignored by  $SR^{++}$ . Regarding  $RS$ , since it is a “role” similarity measure, it emphasizes more on similar node degree than high connectivity. Thus,  $RS$  can only partially resolve the  $SR$  “connectivity trait” problem.  $\square$

Example 1 suggests that the state-of-the-art methods (e.g., SimRank++ [1] and RoleSim [22]) cannot solidly circumvent the “connectivity trait” problem of SimRank. Unfortunately, as illustrated by our statistical experiments in Section 7.5.4, there are many node-pairs suffering from this problem (e.g., 62.3% in social networks, 82.7% in Web graphs, and 56.4% in citation graphs), which has adversely affected the quality of similarity search. This highlights our need for a high-quality model for efficient similarity search to resolve the “connectivity trait” problem.

## 1.2 Main Contributions

Our main contributions are summarised as follows:

- We formulate the exact diagonal correction matrix  $D$ , and propose a “varied- $D$ ” method to accurately compute SimRank with no  $\epsilon_{\text{diag}}$  and in linear memory. Moreover, by grouping computation, we also optimize the algorithm [25] from quadratic to linear time *w.r.t.*  $k$ . (Section 2)
- We observe a “connectivity trait” problem for SimRank, which SimRank++ [1] cannot resolve in a recursive style. To circumvent this problem, we design a “cosine-based” SimRank model and improve the search quality. (Section 3)
- We propose an efficient dimensionality reduction method,  $PSR^\#$ , via block Arnoldi-Ruhe iterations, which will drastically accelerate partial-pairs “cosine-based” SimRank similarity join on billion-scale graphs, with provable guarantees on accuracy. (Section 4)
- We give mathematical insights to the semantic difference between Jeh and Widom’s model [19] and its variant [29], and correct an argument [25]: if  $D$  is replaced by  $(1 - \gamma)I$ , top- $K$  rankings will not be affected much. (Sections 5.1–5.2)
- We devise a novel algorithm that can make instant conversion from Li *et al.*’s SimRank  $\tilde{S}$  to Jeh and Widom’s SimRank  $S$  without any loss of accuracy, and provide key intuitions behind our conversion formulae. (Section 5.3)
- We also notice that, if two nodes are in two different graphs (or two disconnected components of the same graph), the existing SimRank model and the “cosine-based” model would assess these nodes as completely dissimilar. To alleviate this problem, we generalise our “cosine-based” SimRank model,  $GSR^\#$ , which effectively measures the similarity for the nodes across two distinct graphs (or two disconnected components). (Section 6)

Comprehensive experiments on various real datasets validate that (1)  $SR^\#$  improves an accuracy of average  $NDCG_{200}$  by  $\sim 30\%$  over SimRank on various real networks, and runs  $\sim 10x$  faster than

the state-of-the-art competitors on large datasets with 65.8M links for 1000 queries. (2)  $\text{PSR}^\#$  scales well on billion-edge graphs and runs  $\sim 8x$  faster than  $\text{SR}^\#$  with comparable memory consumption of  $\text{SR}^\#$ . (3) Our formulae that convert from Li *et al.*'s SimRank to Jeh and Widom's SimRank produce exact similarity scores without any sacrifice in accuracy, and converges faster than the traditional SimRank iterative approach. (4)  $\text{GSR}^\#$  achieves better semantics when assessing similarities for the nodes across two distinct graphs. Its Arnoldi-based variation,  $\text{AGSR}^\#$ , will enable one-order-of-magnitude speedup while achieving high accuracy to at least 7 decimal places. (Section 7)

Note that a preliminary version of this work has been published as a conference paper in SIGIR 2015 [52]. We summarise the main changes as follows:

- (1) Introduction (Section 1). We reorganise the abstract and introduction to enhance the motivation and contributions of this extended version.
- (2) Techniques and Methods (Section 4, Section 5.3, and Section 6). On top of the "cosine-based" SimRank model ( $\text{SR}^\#$ ) that we presented in the conference version [52], we add 3 new sections:
  - In Section 4, we propose a fast scalable algorithm,  $\text{PSR}^\#$ , for evaluating partial-pairs "cosine-based" SimRank similarities on large graphs, which enables a significant speedup in the retrieval of  $\text{SR}^\#$  similarities. We also provide theoretical guarantees on the accuracy of  $\text{PSR}^\#$ .
  - In Section 5.3, we devise a novel approach that accurately converts from Li *et al.*'s SimRank  $\tilde{S}$  to Jeh and Widom's SimRank  $S$ , and provide mathematical insights underpinning our conversion. In comparison, our conference paper [52] only illustrated the different pairs of paths tallied by  $\tilde{S}$  and  $S$ , with no investigations on how to convert from  $\tilde{S}$  to  $S$ .
  - In Section 6, we propose  $\text{GSR}^\#$ , a generalisation of  $\text{SR}^\#$  [52], to effectively quantify the similarities for the nodes across two distinct graphs. Moreover, we devise  $\text{AGSR}^\#$ , which incorporates  $\text{GSR}^\#$  with Arnoldi-based dimensionality reduction techniques, to substantially accelerate the retrieval of  $\text{GSR}^\#$  similarities.
- (3) Experiments (Sections 7.2–7.5). We conduct additional experiments to show (i) the retrieval quality of  $\text{PSR}^\#$  integrated to the overall performance comparison (§ 7.2.1), (ii) the time efficiency and high scalability of  $\text{PSR}^\#$  on more billion-scale real datasets (*e.g.*, UK02 and IT04) (§ 7.2.3), (iii) more qualitative case studies on retrieval quality over DBLP dataset (§ 7.2.2), (iv) sensitivity analysis of  $\text{PSR}^\#$  relevant to low dimensionality  $m$  and iteration number  $k$  (§ 7.2.3 and 7.2.6), (v) the memory efficiency of  $\text{PSR}^\#$  and  $\text{SR}^\#$ , and its impact relevant to  $m$  and  $k$  (§ 7.4.1 and 7.4.2), (vi) the exactness and faster convergence rate of our formulae that convert from Li *et al.*'s SimRank to Jeh and Widom's SimRank (§ 7.5.2), (vii) the better semantics of  $\text{GSR}^\#$  and  $\text{AGSR}^\#$  across two graphs, and high efficacy of  $\text{AGSR}^\#$  against  $\text{GSR}^\#$  in terms of time (§ 7.3.3), memory (§ 7.4.3), accuracy (§ 7.2.4, 7.2.5), and a case study of  $\text{GSR}^\#$  on two graphs with different scales (§ 7.2.7).
- (4) Related Work (Section 8). We also add new related work that has appeared recently to make the paper more complete.

## 2 ACCURATE AND FAST SIMRANK

We first show the sensitivity of diagonal correction matrix  $D$  to SimRank matrix  $S$ , and formulate the exact  $D$ . Then, we devise an accurate fast "varied- $D$ " model to compute  $S$ .

### 2.1 Sensitivity of Diagonal Correction Matrix

In matrix forms, SimRank in Eq.(1) can be rewritten as

$$S = \max\{\gamma P^\top S P, I\}, \quad (6)$$

where  $\max\{*\}$  denotes the matrix entry-wise maximum, *i.e.*,  $(\max\{A, B\})_{i,j} = \max\{A_{i,j}, B_{i,j}\}$ . The bottleneck for solving  $S$  in Eq.(6) is to deal with the *non-linear* matrix operator  $\max\{*\}$ . To tackle

this problem, based on the observation that  $S$  and  $\gamma P^\top SP$  differ only in their diagonal entries, Kusumoto *et al.* [25] have showed that there exists a unique diagonal matrix  $D$  such that Eq.(6) can be converted to

$$S = \gamma P^\top SP + D, \quad (7)$$

where  $D$  is called *the diagonal correction matrix*, which needs to be determined beforehand.

However, [25] did not mention how to accurately compute the exact  $D$ , but simply approximated  $D$  by  $\tilde{D} = (1 - \gamma)I$ . In fact,  $D$  is very sensitive to the resulting  $S$ . Even small errors in  $D$  may lead to large changes in SimRank scores  $S$  by a factor of up to  $\frac{1}{1-\gamma}$ , as shown in Lemma 1.

LEMMA 1. *Let  $S$  be the solution to Eq.(7), and  $S_{\tilde{D}}$  be the solution to the equation:*

$$S_{\tilde{D}} = \gamma P^\top S_{\tilde{D}} P + \tilde{D}, \quad (8)$$

and let  $\Delta D := D - \tilde{D}$  and  $\Delta S := S - S_{\tilde{D}}$ . Then,

$$\|\Delta S\|_{\max} \leq \frac{1}{1-\gamma} \|\Delta D\|_{\max}^2 \quad (9)$$

PROOF. The recursion of  $S_{\tilde{D}}$  in Eq.(8) naturally leads to the following series:

$$S_{\tilde{D}} = \tilde{D} + \gamma P^\top \tilde{D} P + \gamma^2 (P^\top)^2 \tilde{D} P^2 + \dots, \quad (10)$$

We subtract Eq.(10) from Eq.(2), and then take  $\|\cdot\|_{\max}$  norms on both sides:

$$\begin{aligned} \|\Delta S\|_{\max} &\leq \|\Delta D\|_{\max} + \sum_{i=1}^{\infty} \gamma^i \overbrace{\|(P^\top)^i \Delta D (P^i)\|_{\max}}^{\leq \|\Delta D\|_{\max}} \\ &\leq (1 + \gamma + \gamma^2 + \dots) \|\Delta D\|_{\max} \\ &= \frac{1}{1-\gamma} \|\Delta D\|_{\max}. \quad \square \end{aligned}$$

## 2.2 Formulating Diagonal Correction Matrix

We next derive an exact explicit formulation of  $D$  in Eq.(7). For ease of exposition, the following notations are adopted.

DEFINITION 1 (ENTRY-WISE PRODUCT). *For two  $n \times m$  matrices  $X$  and  $Y$ , their entry-wise product, denoted by  $X \circ Y$ , is an  $n \times m$  matrix, with each  $(i, j)$ -entry given by  $(X \circ Y)_{i,j} = X_{i,j} Y_{i,j}$ .*

Let  $\text{diag}(Z)$  be a diagonal matrix whose diagonal entries are those of  $Z$ , i.e.,  $(\text{diag}(Z))_{i,i} = Z_{i,i}$ . Using this notation, Eq.(6) can be represented as

$$S = \gamma P^\top SP + I - \gamma \text{diag}(P^\top SP). \quad (11)$$

Due to  $D$  uniqueness, Eqs.(7) and (11) imply that

$$D = I - \gamma \text{diag}(P^\top SP). \quad (12)$$

To formulate the exact  $D$  in Eq.(12) only in terms of  $P$ , we introduce the following lemma.

LEMMA 2. *Let  $\overrightarrow{\text{diag}}(Z)$  be a column vector of the diagonal entries of  $Z$ , i.e.,  $(\overrightarrow{\text{diag}}(Z))_i = Z_{i,i}$ . For two  $n \times n$  matrices  $X$  and  $Y$ , and an  $n \times n$  diagonal matrix  $Z$ , we have  $\overrightarrow{\text{diag}}(X^\top Z Y) = (X \circ Y)^\top \overrightarrow{\text{diag}}(Z)$ .*

Combining Lemma 2 with Eq.(12), we next formulate  $D$ .

<sup>2</sup>  $\|X\|_{\max}$  returns the maximum of the absolute values of all entries in matrix  $X$ .

**THEOREM 1.** *The diagonal correction matrix  $D$  in Eq.(7) can be explicitly formulated as*

$$\overrightarrow{\text{diag}}(D) = \left( \sum_{k=0}^{+\infty} \gamma^k (P^k \circ P^k) \right)^{-\top} \vec{1}, \quad (13)$$

where  $\vec{1}$  is a  $|V| \times 1$  vector of all 1s, and  $(*)^{-\top} := ((*)^{\top})^{-1}$ .

**PROOF.** The recursive SimRank equation defined by Eq.(7) implies the following unrolled form:

$$S = D + \gamma P^{\top} D P + \gamma^2 (P^{\top})^2 D P^2 + \gamma^3 (P^{\top})^3 D P^3 + \dots \quad (14)$$

Taking  $\overrightarrow{\text{diag}}(*)$  on both sides of Eq.(14) produces

$$\overrightarrow{\text{diag}}(S) = \overrightarrow{\text{diag}}(D) + \gamma \overrightarrow{\text{diag}}(P^{\top} D P) + \gamma^2 \overrightarrow{\text{diag}}((P^{\top})^2 D P^2) + \dots \quad (15)$$

According to SimRank definition,  $S_{i,i} = 1$  ( $\forall i \in V$ ), which implies  $\overrightarrow{\text{diag}}(S) = \vec{1}$ . Applying this and Lemma 2 to the right-hand side of Eq.(15) yields

$$\vec{1} = \left( I + \gamma(P \circ P) + \gamma^2(P^2 \circ P^2) + \dots \right)^{\top} \overrightarrow{\text{diag}}(D), \quad (16)$$

Since  $0 \leq (P \circ P)_{i,j} \leq P_{i,j} \leq 1$ , one can readily show that  $(I + \gamma(P \circ P) + \gamma^2(P^2 \circ P^2) + \dots)^{\top}$  is diagonally dominant. Multiplying both sides by its inverse produces Eq.(13).  $\square$

Theorem 1 characterizes the exact  $D$  as an *infinite* series. Hence, prior to computing  $S$ , it is too difficult to obtain the *exact*  $D$  in only a *finite* number of iterations. This tells us that using the method of [25] will innately produce  $\epsilon_{\text{diag}}$ .

In addition, Theorem 1 implies that the estimation  $D \approx (1 - \gamma)I$  in [25] is not appropriate for accurately computing  $S$  in Eq.(7). This is because replacing  $(P^k \circ P^k)$  by  $P^k$  in Eq.(13) yields

$$\overrightarrow{\text{diag}}(D) \approx \left( \sum_{k=0}^{+\infty} \gamma^k P^k \right)^{-\top} \vec{1} = (I - \gamma P)^{\top} \vec{1} = (1 - \gamma) \vec{1},$$

which suggests that the approximation  $D \approx (1 - \gamma)I$  in [25] is equivalent to the approximation  $P^k \circ P^k \approx P^k$ . However, it is generally unreasonable to assume that  $((P^k)_{i,j})^2 \approx (P^k)_{i,j}$ . In Section 5.2, we will further discuss  $D \approx (1 - \gamma)I$  from the viewpoint of semantics.

### 2.3 A “Varied- $D$ ” Iterative Model

Another important consequence of Theorem 1 is to derive an accurate SimRank algorithm without  $\epsilon_{\text{diag}}$ . Instead of determining the exact  $D$  in advance, our method is to iteratively update  $D$  and  $S$  at the same time. Precisely, we leverage *the “varied- $D$ ” SimRank model* as follows:

$$S^{(k)} := D_k + \gamma P^{\top} D_{k-1} P + \dots + \gamma^k (P^{\top})^k D_0 P^k, \quad (17)$$

where  $\{D_k\}$  is a diagonal matrix sequence (convergent to  $D$ ), which can be iteratively obtained while  $S$  is being iterated.

Different from the model Eq.(2) by Kusumoto *et al.* [25], our “varied- $D$ ” model Eq.(17) replaces all  $D$ s by a convergent sequence  $\{D_k\}$ . The advantage of our replacement is that Eq.(17) can avoid determining the *exact*  $D$  beforehand, and thus, will not produce the superfluous error  $\epsilon_{\text{diag}}$ .

The correctness of our “varied- $D$ ” model can be verified by taking limits  $k \rightarrow \infty$  on both sides of Eq.(17). As  $k \rightarrow \infty$ ,  $D_k \rightarrow D$  and  $S^{(k)} \rightarrow S$ . Thus, Eq.(17) converges to Eq.(2).

2.3.1 *Finding  $D_k$  in “Varied- $D$ ” Model.* The challenging problem in our “varied- $D$ ” Eq.(17) is to determine the diagonal matrix  $D_k$ . Our main idea is based on two observations:

(a)  $S^{(k)}$  in Eq.(17) can be iterated as

$$S^{(l)} = \gamma P^\top S^{(l-1)} P + D_l \quad \text{with} \quad S^{(0)} = D_0. \quad (18)$$

(b) To ensure  $\text{diag}(S^{(l)}) = I$ ,  $D_l$  in Eq.(18) must satisfy

$$D_l = I - \gamma \text{diag}(P^\top S^{(l-1)} P). \quad (19)$$

Coupling these observations, we can compute  $D_k$  in Eq.(17).

**THEOREM 2.** *The diagonal correction matrices in Eq.(17) can be iteratively obtained as follows:*

$$(D_k)_{i,i} = 1 - \sum_{l=1}^k (h_l \circ h_l)^\top \overrightarrow{\text{diag}}(D_{k-l}) \quad \text{with} \quad D_0 = I, \quad (20)$$

where the auxiliary vectors  $h_1, \dots, h_k$  are derived from

$$\begin{cases} h_0 = e_i \\ h_l = \sqrt{\gamma} P h_{l-1} \quad (l = 1, 2, \dots, k) \end{cases} \quad (21)$$

**PROOF.** First, we derive a matrix formula of  $D_k$ . By Lemma 2, Eq.(20) can be converted to

$$(D_k)_{i,i} = 1 - \sum_{l=1}^k h_l^\top D_{k-l} h_l \quad (22)$$

Successive substitution applied to Eq.(21) yields  $h_l = \sqrt{\gamma^l} P^l e_i$ . Then, substituting this back into Eq.(22) produces

$$D_k = I - \sum_{l=1}^k \gamma^l \text{diag}((P^l)^\top D_{k-l} (P^l)) \quad (23)$$

Next, we show that  $D_k$  in Eq.(23) satisfies Eqs.(17)–(19). It follows from Eq.(17) that

$$\begin{aligned} \text{diag}(\gamma P^\top S^{(k-1)} P) &= \text{diag}(\sum_{l=0}^{k-1} \gamma^{l+1} (P^{l+1})^\top D_{k-1-l} P^{l+1}) \\ &= \text{diag}(\sum_{l=1}^k \gamma^l (P^l)^\top D_{k-l} P^l). \end{aligned}$$

Thus, the above equation implies that

$$I - \text{diag}(\gamma P^\top S^{(k-1)} P) = I - \sum_{l=1}^k \gamma^l \text{diag}((P^l)^\top D_{k-l} (P^l))$$

Applying Eq.(23) to the right-hand side yields Eq.(19).  $\square$

Theorem 2 provides a simple efficient method to compute  $D_k$ , as shown in Algorithm 1. It works as follows. The algorithm first initializes  $D_0 := I$ . To compute  $D_k$  ( $k \geq 1$ ), it then iteratively generates  $k$  auxiliary vectors  $h_1, \dots, h_k$  via Eq.(21). Finally, utilizing  $h_1, \dots, h_k$  and the diagonal matrices  $D_0, \dots, D_{k-1}$ , it obtains  $D_k$  from Eq.(20).

---

**Algorithm 1:** Compute Diagonal Matrix  $D_k$

---

```

1 initialize  $t := 0$ ,  $h_0 := e_i$ ,  $D_0 := I$ ;
2 for  $l := 1, 2, \dots, k$  do
3   compute  $h_l := \sqrt{\gamma} P h_{l-1}$ ;
4   update  $t := t + (h_l \circ h_l)^\top \overrightarrow{\text{diag}}(D_{k-l})$ ;
5 return  $(D_k)_{i,i} := 1 - t$ ;

```

---

The correctness of Algorithm 1 is verified by Theorem 2. Regarding computational complexity, we have the following result.

**THEOREM 3.** *Given the total number of iterations  $k = 1, 2, \dots$ , Algorithm 1 runs in  $O(k|V|)$  memory and  $O(k(|E| + |V|))$  time.*



PROOF. The complexity of Algorithm 1 is dominated by Lines 3 and 4. Specifically, for each iteration  $l$ , (a) in Line 3, it requires  $O(|E|)$  time and  $O(|V|)$  memory to compute  $h_l$ , involving matrix-vector multiplication  $Ph_{l-1}$ ; (b) in Line 4, updating  $t$  needs  $O(|V|)$  time and  $O(|V|)$  memory, including vector entry-wise product  $h_l \circ h_l$ , and a dot product between vectors  $(h_l \circ h_l)$  and  $\overrightarrow{\text{diag}}(D_{k-l})$ . Hence, for  $k$  iterations, it takes  $O(k(|E| + |V|))$  time and  $O(k|V|)$  memory in total.  $\square$

Theorem 3 implies that our “varied- $D$ ” method to compute  $D_k$  will not compromise the scalability of [25] for high quality search, since  $D_k$  can be computed in linear memory as well.

2.3.2 *Fast Convergence of “Varied- $D$ ” Model.* In contrast to the bound  $(\frac{\gamma^{k+1}}{1-\gamma})$  (see Eq.(10) of [25]), our “varied- $D$ ” model in Eq.(17) converges faster than [25], as shown below.

THEOREM 4. *The gap between the  $k$ -th iterative SimRank  $S^{(k)}$  (Eq.(17)) and the exact  $S$  (Eq.(7)) is*

$$\|S^{(k)} - S\|_{\max} \leq \gamma^{k+1}. \quad (24)$$

PROOF. We subtract Eq.(7) from Eq.(18) to obtain,  $\forall k$ ,

$$S^{(k)} - S = \gamma P^T (S^{(k-1)} - S) P + (D^{(k)} - D). \quad (25)$$

We notice from Eq.(19) that  $(S^{(k)})_{i,i} = S_{i,i} = 1, \forall i \in V$ . Thus, when  $i \neq j$ , it follows from Eq.(25) that,  $\forall i, j \in V$ ,

$$\begin{aligned} (S^{(k)} - S)_{i,j} &= \gamma (P^T)_{i,*} (S^{(k-1)} - S) P_{*,j} \\ &\leq \gamma \|S^{(k-1)} - S\|_{\max} \leq \dots \leq \gamma^k \|I - S\|_{\max} \end{aligned} \quad (26)$$

By Eq.(1),  $\|I - S\|_{\max} \leq \gamma$ . Thus, Eq.(24) holds.  $\square$

## 2.4 Efficiently Computing $S^{(k)}$

Having determined  $D_k$  in our “varied- $D$ ” model Eq.(17), we next propose our method to efficiently compute  $S^{(k)}$ . The existing method by Kusumoto *et al.* [25] requires  $O(k^2|E|)$  and  $O(k^2|V||E|)$  time, respectively, to compute single-source and all-pairs SimRank. We observe that there exist many duplicate products in [25]. Precisely, to obtain the result of the sums

$$(S^{(k)})_{i,*} = D_k x_0 + \gamma P^T D_{k-1} x_1 + \dots + \gamma^k (P^T)^k D_0 x_k, \quad (27)$$

the method [25] *separately* computes every  $(\gamma^l (P^T)^l (D_{k-l}) x_l)$  and then adds them together. Its main limitation is that, to compute any power of  $(P^T)$ , [25] has to go through all of the previous powers from scratch. As a result, there are  $l$  matrix-vector products to compute each  $h$ , leading to  $\sum_{l=1}^k l = O(k^2)$  products for  $k$  iterations in total.

We now propose an efficient method which reduces  $O(k^2|E|)$  to  $O(k|E|)$  time, with no loss of accuracy. Our key observation is that “doing each matrix-vector multiplication separately is equivalent to multiplying a matrix by a group of the resulting vectors added together”. Hence, we rearrange the computation of Eq.(27) as follows:

$$\begin{aligned} (S^{(k)})_{i,*} &= D_k x_0 + \gamma P^T (D_{k-1} x_1 + \gamma P^T (D_{k-2} x_2 + \dots \\ &\quad \dots + \gamma P^T (D_1 x_{k-1} + \gamma P^T (D_0 x_k))) \end{aligned} \quad (28)$$

and obtain the result by starting with the innermost brackets and working outwards. In contrast with the method [25], Eq.(28) has only  $O(k)$  matrix-vector products in  $k$  brackets.

It is noteworthy that our “varied- $D$ ” model and the efficient strategy to compute  $S$  from Eq.(28) in this section lay a solid foundation for optimising the computation of the enhanced “cosine-based” SimRank in the following sections. As will be seen shortly in Sections 3.2 and 6.3, these strategies will be applied in a similar manner to (i) the “cosine-based” SimRank model (leading to Eqs.(34)

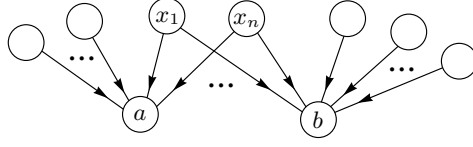


Fig. 2. SimRank “Connectivity Trait” Problem

and (35)) and (ii) our generalised “cosine-based” SimRank model (leading to Eqs.(64) and (65)), respectively. Moreover, the “varied- $D$ ” model makes it possible for us to accurately compute Jeh and Widom’s SimRank on graphs using only linear memory, thereby scaling the computation of Jeh and Widom’s SimRank on any sizable real-world graphs for future experiments on semantic comparisons in Section 7.2.1.

### 3 ENHANCING SIMRANK QUALITY

After the superfluous  $\epsilon_{\text{diag}}$  is avoided, we next focus on the “connectivity trait” problem of SimRank.

#### 3.1 The “Connectivity Trait” Problem

We observe that the root cause of the “connectivity trait” problem is that the order of the normalized factor  $\frac{1}{|N_a||N_b|}$  in the SimRank definition Eq.(1) is too high. To clarify this, let us consider the following situation in Figure 2.

Let  $\delta$  be the number of paths  $\{a \leftarrow x \rightarrow b\}$  to be inserted between nodes  $a$  and  $b$ . By SimRank definition Eq.(1), after insertions,  $s(a, b)$  will become a function of  $\delta$ :

$$s_\delta(a, b) = \gamma \cdot \frac{|N_a \cap N_b| + \delta}{(|N_a| + \delta)(|N_b| + \delta)} \sim \gamma \cdot \frac{\delta}{\delta^2} \rightarrow 0. \quad (\delta \rightarrow \infty) \quad (29)$$

This suggests that, for large  $\delta$ ,  $s_\delta(a, b)$  behaves like  $(\gamma \cdot \frac{1}{\delta})$ , which will eventually decrease *w.r.t.*  $\delta$ .

#### 3.2 Our Kernel-Based SimRank Model

To avoid the order inconsistency between denominator and numerator in Eq.(29), our goal is to judiciously adjust the order of  $\frac{1}{|N_a||N_b|}$  while normalizing  $s(a, b)$  correctly.

**DEFINITION 2.** Let  $A$  be an adjacency matrix. The “cosine-based” SimRank  $\hat{S}_{a,b}$  between  $a$  and  $b$  is defined by

$$\hat{S}_{a,b} = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \frac{e_a^\top (A^k)^\top A^k e_b}{\|A^k e_a\|_2 \|A^k e_b\|_2}, \quad (30)$$

where  $\|x\|_2 := \sqrt{\sum_i |x_i|^2}$  denotes the  $L_2$ -norm of vector  $x$ . To prevent division by zero in Eq.(30), we define the  $k$ -th term of the sums to be 0 if  $(A^k)_{*,a}$  or  $(A^k)_{*,b} = \vec{0}$ .

Our cosine-based SimRank  $\hat{S}_{a,b}$  integrates the weighted cosine similarities between  $a$ ’s and  $b$ ’s multi-hop in-neighbors. This can be seen more clearly when we rewrite Eq.(30) as

$$\hat{S}_{a,b} = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \phi(A^k e_a, A^k e_b) \quad \text{with } \phi(x, y) := \frac{x^\top y}{\|x\|_2 \|y\|_2}. \quad (31)$$

We call  $\phi(x, y)$  a *kernel similarity function*. In Definition 2, we take  $\phi(x, y)$  as the well-known cosine similarity function. The vector  $A^k e_a$  (resp.  $A^k e_b$ ) in Eq.(31) collects the information about  $k$ -hop in-neighbors of node  $a$  (resp.  $b$ ). Hence, the term  $\phi(A^k e_a, A^k e_b)$  in Eq.(31) evaluates how similar node  $a$ ’s and  $b$ ’s  $k$ -hop in-neighbor sets are likely to be in terms of the number of length- $k$  paths in-linked from both  $a$  and  $b$ . The factor  $\gamma^k$  penalizes connections made with distant  $k$ -hop in-neighbors, and  $(1 - \gamma)$  normalizes  $\hat{S}_{a,b}$  into  $[0, 1]$ . Thus,  $\hat{S}_{a,b}$  not only distills the self-referentiality of SimRank, but also extends a one-step cosine similarity to a multi-step one.

**THEOREM 5.** *The cosine-based SimRank model in Eq.(30) can circumvent the SimRank “connectivity trait” problem.*

**PROOF.** Let  $\text{hop}_k(x) = \{i \in V | (A^k e_x)_i > 0\}$  be the  $k$ -hop in-neighbor set of node  $x$ . Then, we have

$$\begin{aligned} e_a^\top (A^k)^\top A^k e_b &= |\text{hop}_k(a) \cap \text{hop}_k(b)|, \\ \|A^k e_a\|_2 &= \sqrt{e_a^\top (A^k)^\top A^k e_a} = \sqrt{|\text{hop}_k(a)|}. \end{aligned}$$

Plugging these into Eq.(30) produces

$$\hat{S}_{a,b} = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \frac{|\text{hop}_k(a) \cap \text{hop}_k(b)|}{\sqrt{|\text{hop}_k(a)| \cdot |\text{hop}_k(b)|}}. \quad (32)$$

When inserting the following  $\delta$  paths between  $a$  and  $b$ :

$$a \leftarrow \underbrace{\circ \leftarrow \cdots \leftarrow \circ}_{k_1 \text{ edges}} \left[ \square \right] \rightarrow \underbrace{\circ \rightarrow \cdots \rightarrow \circ}_{k_2 \text{ edges}} \rightarrow b \quad (33)$$

we notice that, only for  $k_1 = k_2$ , the  $k_1$ -th term of the series Eq.(32) is changed to a function of  $\delta$ :

$$f(\delta) = \gamma^{k_1} \frac{|\text{hop}_{k_1}(a) \cap \text{hop}_{k_1}(b)| + \delta}{\sqrt{(|\text{hop}_{k_1}(a)| + \delta) \cdot (|\text{hop}_{k_1}(b)| + \delta)}}. \quad (\delta > 0)$$

To show  $f(\delta)$  increases w.r.t.  $\delta$ , we take  $\log(*)$  on both sides, and then use implicit differentiation w.r.t.  $\delta$  on both sides:

$$f'(\delta) = f(\delta) \left( \frac{1}{|\text{hop}_{k_1}(a) \cap \text{hop}_{k_1}(b)| + \delta} - \frac{1}{2(|\text{hop}_{k_1}(a)| + \delta)} - \frac{1}{2(|\text{hop}_{k_1}(b)| + \delta)} \right).$$

Since  $f(\delta) > 0$  and  $|\text{hop}_{k_1}(a)| \geq |\text{hop}_{k_1}(a) \cap \text{hop}_{k_1}(b)|$  and  $|\text{hop}_{k_1}(b)| \geq |\text{hop}_{k_1}(a) \cap \text{hop}_{k_1}(b)|$ , we can obtain  $f'(\delta) > 0$ . Thus,  $f(\delta)$  increases w.r.t.  $\delta$ , which implies that paths (33) insertion will not decrease  $\hat{S}_{a,b}$ .  $\square$

Indeed, by using  $Pe_b = Ae_b / \|Ae_b\|_1$ <sup>3</sup> to the original SimRank Eq.(2), we notice that both Eqs.(2) and (30) tally the same paths in-linked from  $a$  and  $b$ . The difference is norms  $\|*\|_2$  and  $\|*\|_1$  used by Eq.(30) and Eq.(2)<sup>4</sup>, respectively. Since the SimRank “connectivity trait” problem is due to the high order of  $\frac{1}{|N_a||N_b|}$  in Eq.(1), it is reasonable for us to prevent its high order by replacing  $\|*\|_1$  with  $\|*\|_2$  since  $\|x\|_2 \leq \|x\|_1$ . Moreover, by using  $\|*\|_2$ ,  $\hat{S}_{a,b}$  can be correctly normalized into  $[0, 1]$ . This is because  $\phi(*, *) \in [0, 1]$ , which indicates that  $0 \leq \hat{S}_{a,b} \leq (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \leq 1$  in Eq.(31).

**EXAMPLE 2.** *Recall the  $\delta$  paths  $\{a \leftarrow x \rightarrow b\}$  to be added into  $G$  in Figure 2. After insertion,  $\hat{S}_{a,b}(\delta)$  in Eq.(30) can circumvent the “connectivity trait” problem. This is because*

$$Ae_a = \underbrace{(1, 1, \dots, 1, 0, 0, \dots, 0)}_{|N_a|} \underbrace{(0, 1, 1, \dots, 1)}_{|N_b - N_a|} \underbrace{(1, 1, \dots, 1)}_{\delta}^\top \quad Ae_b = \underbrace{(0, 0, \dots, 0, 1, 1, \dots, 1)}_{|N_a - N_b|} \underbrace{(1, 1, \dots, 1)}_{|N_b|} \underbrace{(1, 1, \dots, 1)}_{\delta}^\top$$

Then, we have  $(Ae_a)^\top Ae_b = |N_a \cap N_b| + \delta$  and

$$\|Ae_a\|_2 = \sqrt{\underbrace{1^2 + \dots + 1^2}_{|N_a|} + \underbrace{1^2 + \dots + 1^2}_{\delta}} = \sqrt{|N_a| + \delta}, \quad \|Ae_b\|_2 = \sqrt{|N_b| + \delta}$$

<sup>3</sup>  $\|x\|_1 := \sum_i |x_i|$  denotes the  $L_1$ -norm of vector  $x$ .

<sup>4</sup>  $P$  is associated with  $\frac{1}{|N_a||N_b|}$  ( $= \frac{1}{\|Pe_a\|_1 \|Pe_b\|_1}$ ) in Eq.(1).

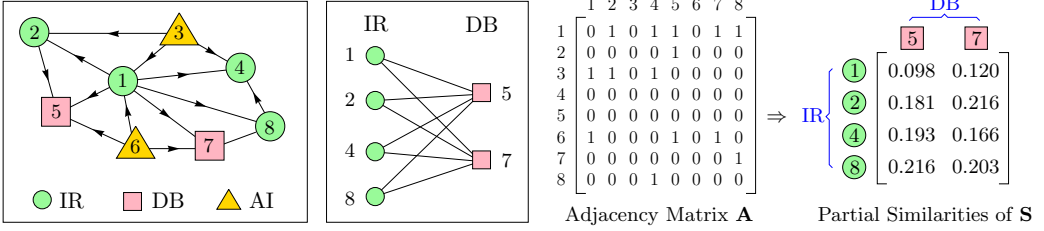


Fig. 3. Partial-pairs similarity search on citation graph  $G$ , where each node is a paper labelled with an area it belongs to (e.g., IR, DB, AI) - how to retrieve only the similarities of pairs of papers between IR and DB.

Therefore, it follows from Eq.(30) that  $\hat{S}_{a,b}(\delta) = (1 - \gamma)\gamma \cdot \frac{|N_a \cap N_b| + \delta}{\sqrt{|N_a| + \delta} \sqrt{|N_b| + \delta}} \rightarrow (1 - \gamma)\gamma$  ( $\delta \rightarrow \infty$ ).

Comparing this with Eq.(29),  $\hat{S}_{a,b}(\delta)$  is not decreasing w.r.t.  $\delta$ , due to norm  $\| * \|_2$  used in Eq.(30).  $\square$

In contrast to SimRank++ [1] and PSimRank [9] whose revised weight factors rely only on common  $1$ -hop neighbors  $N_a$  and  $N_b$ , our method by Eq.(30), even if  $N_a \cap N_b = \emptyset$ , can evaluate  $s(a, b)$  from common *multi-hops* neighbors  $\text{hop}_k(a)$  and  $\text{hop}_k(b)$ .

To compute the cosine-based SimRank score  $\hat{S}_{a,b}$ , if  $a = b$ , Eq.(30) implies  $\hat{S}_{a,b} = 1$ . If  $a \neq b$ , we compute  $\hat{S}_{a,b}$  as

$$\hat{S}_{a,b}^{(k)} = \hat{S}_{a,b}^{(k-1)} + (1 - \gamma)\gamma^k (u^{(k)})^\top v^{(k)} \quad \text{with } \hat{S}_{a,b}^{(0)} = 0 \quad (34)$$

where the auxiliary vectors  $u^{(k)}$  and  $v^{(k)}$  are obtained by

$$\begin{cases} u^{(0)} = e_a \\ u^{(k)} = \frac{A u^{(k-1)}}{\|A u^{(k-1)}\|_2} \end{cases} \quad \begin{cases} v^{(0)} = e_b \\ v^{(k)} = \frac{A v^{(k-1)}}{\|A v^{(k-1)}\|_2} \end{cases} \quad (35)$$

#### 4 ACCELERATIVE TECHNIQUES FOR COSINE-BASED SIMRANK SIMILARITY JOIN WITH GUARANTEED ACCURACY

The conventional method using Eqs.(34) and (35) for iteratively computing  $\hat{S}_{a,b}$  is rather cost-inhibitive. This is especially evident for the following partial-pairs search problem:

**PROBLEM (Partial-Pairs ‘‘Cosine-based’’ SimRank Similarity Search).**

**Given:** a graph  $G = (V, E)$ , and two subsets  $A$  and  $B$  of nodes  $V$ .

**Retrieve:** partial-pairs ‘‘cosine-based’’ SimRank scores  $\hat{S}_{A,B} := \{\hat{S}_{a,b} | \forall a \in A, \forall b \in B, a \neq b\}$  between  $A$  and  $B$  efficiently with guaranteed accuracy.

There are many real applications for the partial-pairs similarity search problem, e.g., entity resolution [3], data integration and cleaning [4], and duplicate detection [36]. One typical example is link-based similarity join, as illustrated below:

**EXAMPLE 3.** Consider citation digraph  $G$  in Figure 3, where each node represents a paper labelled with the research area it belongs to, e.g., IR (Information Retrieval), ML (Machine Learning), DB (Databases). Each edge is a citation from one paper to another. Partial-pairs similarity search is quite useful when one would like to efficiently retrieve only the similarities of pairs of papers between two given areas (e.g., IR and DB).  $\square$

If the single-pair algorithm in Eqs.(34) and (35) directly runs multiple times for partial-pairs similarity search, it would involve many duplicate computations, taking  $O(K|A||B|(|E| + |V|))$  time to evaluate  $A \times B$  pairs of similarities for  $K$  iterations, which is rather expensive.

#### 4.1 Dimensionality Reduction via Block Arnoldi–Ruhe Process

To efficiently accelerate the computation of our “cosine-based” SimRank model (Eqs.(34) and (35)) for partial-pairs similarity search, we first build a low-dimensional Krylov subspace using a variant of the standard Arnoldi iteration, namely, Block Arnoldi–Ruhe Process [7]. Specifically, given a low-order parameter  $m \ll |V|$ , a graph adjacency matrix  $A$ , and two distinct nodes  $a$  and  $b$  in  $V$ , we first construct a low  $2m$ -dimensional space, namely, block Krylov subspace, as follows:

$$K_m(A, [e_a, e_b]) := \underbrace{\text{span}\{e_a, e_b, Ae_a, Ae_b, A^2e_a, A^2e_b, \dots, A^{m-1}e_a, A^{m-1}e_b\}}_{2m\text{-dimension}}$$

where two unit vectors  $e_a$  and  $e_b$  are chosen as the starting vectors for generating the block Krylov subspace  $K_m(A, [e_a, e_b])$ , and  $\text{span}\{\cdot\}$  denotes the set of all linear combinations of vectors in  $\{\cdot\}$ .

Then the following procedure, also known as the Block Arnoldi–Ruhe Process [7], will produce an orthonormal basis  $Q_m = [q_1, q_2, \dots, q_{2m}]$  of the Krylov subspace  $K_m(A, [e_a, e_b])$ , which is a modified version of Gram-Schmidt process, as illustrated below:

---

**Procedure 2:** Block Arnoldi-Ruhe Process( $m, A, a, b$ )

---

**Input** : low-order parameter  $m$ , graph adjacency matrix  $A$ , two distinct nodes  $a$  and  $b$ .

**Output**: column orthonormal matrix  $Q_m := [q_1, q_2, \dots, q_{2m}]$ ,  
upper block Hessenberg matrix  $H_m := (h_{i,j})$ .

```

1 initialize  $q_1 := e_a$  and  $q_2 := e_b$ ;
2 for  $j := 2, 3, \dots, 2m + 1$  do
3   set  $t := j - 1$ ;
4   update  $w := Aq_t$ ;
5   for  $i := 1, 2, \dots, j$  do
6     compute  $h_{i,t} := w^\top q_i$ ;
7     update  $w := w - h_{i,t}q_i$ ;
8   compute  $h_{j+1,t} := \|w\|_2$ ;
9   if  $h_{j+1,t} \neq 0$  then compute  $q_{j+1} := w/h_{j+1,t}$ ;
10 return  $Q_m := [q_1, q_2, \dots, q_{2m}]$  and  $H_m := (h_{i,j})_{1 \leq i, j \leq 2m}$  ;
```

---

Procedure 2 provides an efficient Arnoldi-Ruhe decomposition of the graph adjacency matrix  $A$ . Given a user-specified low-order parameter  $m \ll |V|$ , and two initial distinct nodes  $a$  and  $b$ , the above Block Arnoldi-Ruhe Process, abbreviated as

$$[Q_m, H_m] \leftarrow \text{Arnoldi-Ruhe}(m, A, a, b)$$

can compress the large adjacency matrix  $A$  (of size  $|V| \times |V|$ ) into the small upper block Hessenberg matrix  $H_m$  (of size  $2m \times 2m$ ) via the column-orthonormal basis  $Q_m := [q_1, q_2, \dots, q_{2m}]$  (of size  $2m \times |V|$ ) of the Krylov subspace  $K_m(A, [e_a, e_b])$ , such that the following relations hold [7]:

$$AQ_m = Q_m H_m + \underbrace{[q_{2m+1} \mid q_{2m+2}] \begin{bmatrix} h_{2m+1,2m-1} & h_{2m+1,2m} \\ h_{2m+2,2m-1} & h_{2m+2,2m} \end{bmatrix} [e_{2m-1} \mid e_{2m}]^\top}_{\text{residual error after } m \text{ steps}} \quad (36a)$$

$$Q_m^\top A Q_m = H_m \quad (36b)$$

Figure 4 pictorially depicts the Block Arnoldi-Ruhe decomposition of Eqs.(36a) and (36b), which is useful to our following accelerative techniques for partial-pairs “cosine-based” SimRank search.

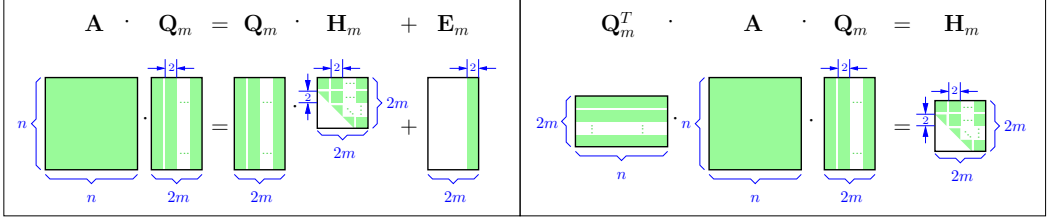


Fig. 4. Visualisation of Block Arnoldi-Ruhe Decomposition of Eqs.(36a) and (36b)

#### 4.2 Accelerating “Cosine-based” SimRank Search

By virtue of the Block Arnoldi-Ruhe decomposition, we are now ready to provide fast efficient techniques for partial-pairs “cosine-based” SimRank search. We first propose the following lemma.

LEMMA 3. *Given a low-order parameter  $m \ll |V|$ , an adjacency matrix  $A \in \mathbb{R}^{|V| \times |V|}$ , and two distinct query nodes  $a$  and  $b$ , let  $[Q_m, H_m] \leftarrow \text{Arnoldi-Ruhe}(m, A, a, b)$  be the  $m$  steps of the Block Arnoldi-Ruhe process. Then, the following equation holds:*

$$\sum_{k=0}^{m-1} \gamma^k \frac{e_a^\top (A^\top)^k A^k e_b}{\|A^k e_a\|_2 \|A^k e_b\|_2} = \sum_{k=0}^{m-1} \gamma^k \frac{e_1^\top (H_m^k)^\top H_m^k e_2}{\|H_m^k e_1\|_2 \|H_m^k e_2\|_2} \quad (37)$$

where  $e_1$  (resp.  $e_2$ ) is a  $2m \times 1$  unit vector, whose 1st (resp. 2nd) entry is 1, and 0s elsewhere.

Lemma 3 implies an efficient dimensionality reduction technique that can substantially speed up partial-pairs “cosine-based” SimRank search. Precisely, let  $\hat{S}_{a,b}^{(K)}$  be the  $K$ -th partial sum of the exact “cosine-based” SimRank score defined by Eq.(30), i.e.,

$$\hat{S}_{a,b}^{(K)} := (1 - \gamma) \sum_{k=0}^{K-1} \gamma^k \frac{e_a^\top (A^k)^\top A^k e_b}{\|A^k e_a\|_2 \|A^k e_b\|_2}. \quad (38)$$

For any user-specified low-order parameter  $m \ll |V|$ , Lemma 3 indicates that the  $m$ -th partial sum of the “cosine-based” SimRank score  $\hat{S}_{a,b}^{(m)}$  can be computed, quickly and accurately, via the  $m$  steps of the Block Arnoldi-Ruhe process as

$$\hat{S}_{a,b}^{(m)} = (1 - \gamma) \sum_{k=0}^{m-1} \gamma^k \frac{e_1^\top (H_m^k)^\top H_m^k e_2}{\|H_m^k e_1\|_2 \|H_m^k e_2\|_2}. \quad (39)$$

Due to the small dimensionality of  $H_m$ , the computation of Eq.(39) only runs iteratively over the low-order subspace ( $2m \times 2m$ ) yielding  $O(m^2)$  worst-case time for each iteration ( $m \ll |V|$ ), which is significantly faster than the conventional Eq.(35) that iterates over the original large space ( $|V| \times |V|$ ) entailing  $O(|V|^2)$  time in the worst case. Moreover, Lemma 3 implies that, when  $K \leq m$ , the speedup of Eq.(39) to evaluate  $\hat{S}_{a,b}^{(K)}$  does not sacrifice any accuracy. This is because, for any  $k = 0, 1, \dots, m-1$ , the projection of  $\{A^k e_a\}$  (resp.  $\{A^k e_b\}$ ) onto the Krylov subspace  $K_m(A, [e_a, e_b])$  with respect to the orthonormal basis  $Q_m$  is itself.

However, when  $K > m$ , for each  $k = m, m+1, \dots, K-1$ , projecting  $\{A^k e_a\}$  (resp.  $\{A^k e_b\}$ ) onto the Krylov subspace  $K_m(A, [e_a, e_b])$  will sacrifice a little accuracy due to the residual errors in the Arnoldi-Ruhe decomposition of Eq.(36a). The following theorem provides a light-weight upper bound of the approximation error for iteratively evaluating  $\hat{S}_{a,b}^{(K)}$  in the case of  $K > m$ .

THEOREM 6. *For any number of iterations  $K = 1, 2, \dots$ , let  $\hat{S}_{a,b}^{(K)}$  be the  $K$ -th partial sum of the exact “cosine-based” SimRank score defined by Eq.(38). Given a low-order parameter  $m \ll |V|$ , let  $\hat{R}_{a,b}^{(K)}[m]$*

be the  $K$ -th partial sums of the approximation of  $\hat{S}_{a,b}^{(K)}$  using the  $m$  steps of the Block Arnoldi–Ruhe process  $[Q_m, H_m] \leftarrow \text{Arnoldi-Ruhe}(m, A, a, b)$ , that is,

$$\hat{R}_{a,b}^{(K)}[m] = (1 - \gamma) \sum_{k=0}^{K-1} \gamma^k \frac{e_1^\top (H_m^k)^\top H_m^k e_2}{\|H_m^k e_1\|_2 \|H_m^k e_2\|_2} \quad (40)$$

Then, for any  $K > m$ , the difference between the values  $\hat{S}_{a,b}^{(K)}$  and  $\hat{R}_{a,b}^{(K)}[m]$  is bounded by

$$\left| \hat{S}_{a,b}^{(K)} - \hat{R}_{a,b}^{(K)}[m] \right| \leq 2(\gamma^m - \gamma^K) \quad (\forall a, b)$$

PROOF. Let

$$\delta(k) = \frac{e_a^\top (A^\top)^k A^k e_b}{\|A^k e_a\|_2 \|A^k e_b\|_2} - \frac{e_1^\top (H_m^k)^\top H_m^k e_2}{\|H_m^k e_1\|_2 \|H_m^k e_2\|_2}$$

By Cauchy–Schwarz inequality, it follows that

$$\begin{aligned} |e_a^\top (A^\top)^k A^k e_b| &= |(A^k e_a)^\top (A^k e_b)| \leq \|A^k e_a\|_2 \|A^k e_b\|_2 \\ |e_1^\top (H_m^k)^\top H_m^k e_2| &\leq \|H_m^k e_1\|_2 \|H_m^k e_2\|_2 \end{aligned}$$

Thus, substituting these results back into  $|\delta(k)|$ , we have

$$|\delta(k)| = \frac{|e_a^\top (A^\top)^k A^k e_b|}{\|A^k e_a\|_2 \|A^k e_b\|_2} + \frac{|e_1^\top (H_m^k)^\top H_m^k e_2|}{\|H_m^k e_1\|_2 \|H_m^k e_2\|_2} \leq 2 \quad (41)$$

By the definition of  $\hat{S}_{a,b}^{(K)}$  and  $\hat{R}_{a,b}^{(K)}[m]$  in Eqs.(38) and (40), since  $K \geq m$ , we obtain

$$\begin{aligned} |\hat{S}_{a,b}^{(K)} - \hat{R}_{a,b}^{(K)}[m]| &= (1 - \gamma) \left| \sum_{k=0}^{K-1} \gamma^k \delta(k) \right| = (1 - \gamma) \left| \overbrace{\sum_{k=0}^{m-1} \gamma^k \delta(k)}^{=0 \text{ by Lemma 3}} + \sum_{k=m}^{K-1} \gamma^k \delta(k) \right| \\ &= (1 - \gamma) \left| \sum_{k=m}^{K-1} \gamma^k \delta(k) \right| \leq (1 - \gamma) \sum_{k=m}^{K-1} \gamma^k |\delta(k)| \leq 2(1 - \gamma) \sum_{k=m}^{K-1} \gamma^k = 2(\gamma^m - \gamma^K) \quad \square \\ &\leq 2 \text{ by Eq.(41)} \end{aligned}$$

Theorem 6 provides a guaranteed a-priori error bound for the  $m$ -step Arnoldi-Ruhe approximation of the  $K$ -th partial sum  $\hat{S}_{a,b}^{(K)}$  of the exact “cosine-based” SimRank similarity in Eq.(38). Precisely, for a given low-order parameter  $m$  ( $\ll |V|$ ), Theorem 6 indicates that, when  $K > m$ , the first  $K$ -th partial sum of the exact similarity  $\hat{S}_{a,b}$  can be approximated by  $\hat{R}_{a,b}^{(K)}[m]$  with the approximation error bounded by  $\epsilon := 2(\gamma^m - \gamma^K)$ . While the low-order parameter  $m$  increases to  $K$ , we notice that  $\gamma^m$  approaches to  $\gamma^K$ , leading to  $\epsilon \rightarrow 0$ . When  $m \geq K$ , as indicated by Lemma 3, the error  $\epsilon$  becomes 0, i.e.,  $\hat{R}_{a,b}^{(K)}[m] = \hat{S}_{a,b}^{(K)}$  for  $\forall m \geq K$ .

Capitalizing on Theorem 6, we are now ready to estimate the difference between  $\hat{R}_{a,b}^{(K)}[m]$  and the exact “cosine-based” SimRank similarity  $\hat{S}_{a,b}^{(K)}$ , as shown in Theorem 7.

**THEOREM 7.** *Let  $\hat{S}$  be the exact “cosine-based” SimRank matrix, with each entry  $\hat{S}_{a,b}$  defined by Eq.(30), and let  $\hat{R}^{(K)}[m]$  be the approximation of  $\hat{S}$  via the  $m$  steps of the Arnoldi-Ruhe process, whose each entry  $\hat{R}_{a,b}^{(K)}[m]$  is defined by Eq.(40). Then, the difference between  $\hat{R}^{(K)}[m]$  and  $\hat{S}$  is bounded by*

$$\|\hat{S} - \hat{R}^{(K)}[m]\|_{\max} \leq \begin{cases} \gamma^K & (\text{if } K \leq m) \\ 2\gamma^m - \gamma^K & (\text{if } K > m) \end{cases} \quad (42a)$$

$$(42b)$$

PROOF. For each node pair  $(a, b) \in V^2$ , since  $\hat{S}_{a,b}^{(K)}$  be the  $K$ -th partial sum of the exact “cosine-based” SimRank score  $\hat{S}_{a,b}$ , we subtract Eq.(38) from Eq.(30), which produces

$$\hat{S}_{a,b} - \hat{S}_{a,b}^{(K)} = (1 - \gamma) \sum_{k=K}^{\infty} \gamma^k \frac{e_a^\top (A^\top)^k A^k e_b}{\|A^k e_a\|_2 \|A^k e_b\|_2}$$

From Cauchy-Schwarz inequality, it follows that

$$|e_a^\top (A^\top)^k A^k e_b| = |(A^k e_a)^\top (A^k e_b)| \leq \|A^k e_a\|_2 \|A^k e_b\|_2$$

which implies that

$$|\hat{S}_{a,b} - \hat{S}_{a,b}^{(K)}| = (1 - \gamma) \sum_{k=K}^{\infty} \gamma^k \frac{|e_a^\top (A^\top)^k A^k e_b|}{\|A^k e_a\|_2 \|A^k e_b\|_2} \leq (1 - \gamma) \sum_{k=K}^{\infty} \gamma^k = \gamma^K \quad (\forall a, b) \quad (43)$$

When  $K \leq m$ , it follows from Lemma 3 that  $\hat{S}_{a,b}^{(K)} = \hat{R}_{a,b}^{(K)}[m]$   $(\forall a, b)$ . When  $K > m$ , it follows from Theorem 6 that  $|\hat{S}_{a,b}^{(K)} - \hat{R}_{a,b}^{(K)}[m]| \leq 2(\gamma^m - \gamma^K)$   $(\forall a, b)$ . Thus,

$$\|\hat{S}^{(K)} - \hat{R}^{(K)}[m]\|_{\max} \leq \begin{cases} 0 & (\text{if } K \leq m) \\ 2(\gamma^m - \gamma^K) & (\text{if } K > m) \end{cases} \quad (44)$$

Therefore,

$$\begin{aligned} \|\hat{S} - \hat{R}^{(K)}[m]\|_{\max} &= \|(\hat{S} - \hat{S}^{(K)}) + (\hat{S}^{(K)} - \hat{R}^{(K)}[m])\|_{\max} \\ &\leq \underbrace{\|\hat{S} - \hat{S}^{(K)}\|_{\max}}_{\leq \gamma^K \text{ by Eq.(43)}} + \underbrace{\|\hat{S}^{(K)} - \hat{R}^{(K)}[m]\|_{\max}}_{\leq 0 \text{ or } 2(\gamma^m - \gamma^K) \text{ by Eq.(44)}} \leq \begin{cases} \gamma^K & (\text{if } K \leq m) \\ 2\gamma^m - \gamma^K & (\text{if } K > m) \end{cases} \end{aligned}$$

□

Theorem 7 derives the first concise a-priori error bound for the maximum approximation error between the exact “cosine-based” SimRank similarity  $\hat{S}$  and the approximate solution  $\hat{R}^{(K)}[m]$  obtained by the  $K$ -th partial sums of the  $m$ -step Arnoldi-Ruhe projection method. For any fixed low-order  $m$ , when  $K \leq m$ , the error between  $\hat{S}$  and  $\hat{R}^{(K)}[m]$  only comes from the truncation error  $\gamma^K$  for the  $K$ -th partial sums of  $\hat{S}$ . This is because, according to Lemma 3,  $\hat{S}^{(K)} = \hat{R}^{(K)}[m]$  for  $K \leq m$ , implying that the low-order Arnoldi-Ruhe projection would not incur any additional errors for this case. When  $K > m$ , the error bound between  $\hat{S}$  and  $\hat{R}^{(K)}[m]$  consists of two parts: The first part  $\gamma^K$  bounding  $\|\hat{S} - \hat{S}^{(K)}\|_{\max}$  is due to the truncation error for the  $K$ -th partial sums of  $\hat{S}$ . The second part  $2(\gamma^m - \gamma^K)$  bounding  $\|\hat{S}^{(K)} - \hat{R}^{(K)}[m]\|_{\max}$  is produced by the low-dimensionality reduction technique using the  $m$ -step Arnoldi-Ruhe process.

As an extreme case, when  $K$  and  $m$  are sufficiently large, the bound  $\|\hat{S} - \hat{R}^{(K)}[m]\|_{\max} \rightarrow 0$ , indicating that our low-dimensionality reduction method is convergent. Given a low-order parameter  $m$ , when  $K \rightarrow +\infty$ , we have the following corollary:

COROLLARY 1. Let  $\hat{S}$  be the exact “cosine-based” SimRank, and let  $\hat{R}[m]$  be the solution via the  $m$  steps of the Arnoldi-Ruhe process, whose each  $(a, b)$ -entry is the limitation of  $\hat{R}_{a,b}^{(K)}[m]$  as  $K \rightarrow \infty$ , i.e.,

$$\hat{R}_{a,b}[m] := \lim_{K \rightarrow +\infty} \hat{R}_{a,b}^{(K)}[m] = (1 - \gamma) \sum_{k=0}^{+\infty} \gamma^k \frac{e_1^\top (H_m^k)^\top H_m^k e_2}{\|H_m^k e_1\|_2 \|H_m^k e_2\|_2}$$

Then, the gap between  $\hat{R}[m]$  and  $\hat{S}$  is bounded by:

$$\|\hat{S} - \hat{R}[m]\|_{\max} \leq 2\gamma^m$$



### 4.3 An Efficient Algorithm for Partial-Pairs “Cosine-based” SimRank Similarity Join

Leveraging Theorems 6 and 7, we next present an efficient algorithm for partial-pairs “cosine-based” SimRank search, as illustrated in Algorithm 3.

---

**Algorithm 3:** Partial-Pairs Cosine-Based SimRank Evaluation ( $A, X, Y, m, K, \gamma$ )

---

**Input** : graph adjacency matrix  $A$ , two node subsets  $X$  and  $Y$ , low dimensionality  $m$ , number of iterations  $K$ , and decay factor  $\gamma$ .

**Output**: “cosine-based” SimRank scores  $\{\hat{R}_{x,y}^{(K)}[m] \mid \forall(x, y) \in X \times Y\}$  between  $X$  and  $Y$ .

```

1  foreach node  $x \in X$  do
2    foreach node  $y \in Y$  do
3       $[Q_m, H_m] \leftarrow$  Arnoldi-Ruhe ( $m, A, x, y$ );
4      initialize  $u^{(0)} := e_1$  and  $v^{(0)} := e_2$ ;
5      initialize  $s := 0$ ;
6      for  $k := 1, 2, \dots, K$  do
7        set  $u^{(k)} := H_m u^{(k-1)}$  and  $v^{(k)} := H_m v^{(k-1)}$ ;
8        set  $\alpha := \|u^{(k)}\|_2$  and  $\beta := \|v^{(k)}\|_2$ ;
9        if  $\alpha \neq 0$  and  $\beta \neq 0$  then
10         update  $u^{(k)} := u^{(k)}/\alpha$  and  $v^{(k)} := v^{(k)}/\beta$ ;
11         update  $s := s + \gamma^k (u^{(k)})^\top v^{(k)}$ ;
12       update  $\hat{R}_{x,y}^{(K)}[m] := (1 - \gamma)s$ ;
13 return  $\{\hat{R}_{x,y}^{(K)}[m] \mid \forall(x, y) \in X \times Y\}$ ;

```

---

Algorithm 3 works as follows. For every node-pair  $(x, y) \in X \times Y$ , it first invokes the “Block Arnoldi-Ruhe Process” in Procedure 2 that compresses the large adjacency matrix  $A$  into the small upper block Hessenberg matrix  $H_m$  (Line 3) using a low-dimensional Krylov subspace. Then, utilising  $H_m$ , the algorithm iteratively computes two auxiliary vectors  $u^{(k)}$  and  $v^{(k)}$  (Lines 4–7) and their Euclidean norms  $\alpha$  and  $\beta$  (Line 8) for  $K$  iterations. At each iteration  $k$ , the inner product of the normalized vectors  $u^{(k)}$  and  $v^{(k)}$ , once computed, is dampened by a factor of  $\gamma^k$  and aggregated together to produce  $s$  (Line 11). Finally, scaling  $s$  by a factor of  $(1 - \gamma)$  yields the scoring result  $\hat{R}_{x,y}^{(K)}[m]$  (Line 12).

**EXAMPLE 4.** Recall graph  $G$  in Figure 3, where each node is a paper (indexed by an integer). Given two node subsets (i.e., two areas of these papers)  $IR = \{1, 2, 4, 8\}$  and  $DB = \{5, 7\}$ , low-order parameter  $m = 3$ , number of iterations  $K = 4$ , and decay factor  $\gamma = 0.4$ , we assess the partial-pairs “cosine-based” SimRank scores between  $IR$  and  $DB$ ,  $\hat{R}_{IR \times DB}^{(4)}[3]$ , via Algorithm 3 as follows:

For each node-pair (e.g.,  $(8, 5)$ ), we first apply the Block Arnoldi-Ruhe technique (Line 3) to obtain upper block Hessenberg matrix  $H_3$  and column-orthonormal matrix  $Q_3$ :

$$H_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1.414 & 0.707 & 0.5 & 0.224 & -0.408 & 0.112 \\ 0 & 1.581 & 1.118 & 0.1 & -0.183 & 0.05 \\ 0 & 0 & 1.225 & 0.548 & -0.333 & 0.274 \\ 0 & 0 & 0 & 0.8 & -0.548 & 0.4 \end{bmatrix} \quad Q_3 = \begin{bmatrix} 0 & 0 & 0.707 & 0.316 & 0 & 0.158 \\ 0 & 0 & 0 & 0.633 & -0.577 & 0.316 \\ 0 & 0 & 0 & 0 & 0.577 & 0.791 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.633 & 0.577 & -0.474 \\ 0 & 0 & 0.707 & -0.316 & 0 & -0.158 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Next, using  $H_3$ , we iteratively compute two normalised vectors  $u^{(k)}$  and  $v^{(k)}$  (Line 10) and scalar  $s$  (Line 11) as follows:

| $k$ | $u^{(k)}$                             | $v^{(k)}$                                 | $(u^{(k)})^\top v^{(k)}$ | $s$ (Line 11) |
|-----|---------------------------------------|---|--------------------------|---------------|
| 0   | $[1, 0, 0, 0, 0]^\top$                | $[0, 0, 1, 0, 0]^\top$                    | 0                        | 0             |
| 1   | $[0, 0, 1, 0, 0]^\top$                | $[0, 0, 0.408, 0.913, 0, 0]^\top$         | 0.408                    | 0.163         |
| 2   | $[0, 0, 0.289, 0.646, 0.707, 0]^\top$ | $[0, 0, 0.289, 0.387, 0.707, 0.516]^\top$ | 0.833                    | 0.297         |
| 3   | $[0, 0, 0, 0.467, 0.853, 0.234]^\top$ | $[0, 0, 0, 0.467, 0.853, 0.234]^\top$     | 1                        | 0.361         |

Finally, normalising  $s$  at the last iteration by a factor of  $(1 - \gamma)$  (Line 12) produces

$$\hat{R}_{7,4}^{(4)}[3] = (1 - \gamma) \times s = (1 - 0.4) \times 0.361 = 0.216$$

Similarly, similarities  $\hat{R}_{IR \times DB}^{(4)}[3]$  for other node-pairs in  $IR \times DB = \{1, 2, 4, 8\} \times \{5, 7\}$  can be obtained. We compare the gap between  $\hat{R}_{IR \times DB}^{(4)}[3]$  and the original solution  $\hat{S}_{IR \times DB}^{(4)}$  via Eqs.(34) and (35) below:

$$\hat{R}_{IR \times DB}^{(4)}[3] = \begin{matrix} & \begin{matrix} 5 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 4 \\ 8 \end{matrix} & \begin{bmatrix} 0.098 & 0.120 \\ 0.181 & 0.216 \\ 0.193 & 0.165 \\ 0.216 & 0.203 \end{bmatrix} \end{matrix} \quad \hat{R}_{IR \times DB}^{(4)}[3] - \hat{S}_{IR \times DB}^{(4)} = \begin{matrix} & \begin{matrix} 5 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 4 \\ 8 \end{matrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.0016 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

It can be verified that this gap is bounded by our estimated error derived from Theorem 6:

$$\|\hat{R}_{IR \times DB}^{(4)}[3] - \hat{S}_{IR \times DB}^{(4)}\|_{\max} = 0.0016 \leq 2(\gamma^m - \gamma^K) = 2 \times (0.4^3 - 0.4^4) = 0.0768. \quad \square$$

**Time Complexity.** The total time complexity of Algorithm 3 is bounded by  $O(|X||Y|m(|E| + Km))$ . This is because, for each node-pair  $(x, y) \in X \times Y$ , it takes  $O(m|E|)$  time for the ‘‘Block Arnoldi-Ruhe Process’’ to generate the upper block Hessenberg matrix  $H_m$  (Line 3). Then, for each loop iteration (Lines 6–11), it requires  $O(m^2)$  time to compute  $u^{(k)}$  and  $v^{(k)}$  (Lines 6–7),  $O(m)$  time to evaluate the Euclidean norms  $\alpha$  and  $\beta$  (Line 8), and  $O(m)$  time to compute  $s$  that involves the inner product of  $u^{(k)}$  and  $v^{(k)}$  (Line 11), which is bounded by  $O(Km^2)$  for  $K$  iterations. Thus, putting these all together, the total time for computing each node-pair similarity  $\hat{R}_{x,y}^{(K)}[m]$  is  $O(m|E| + Km^2) = O(m(|E| + Km))$ , which is dominated by  $O(|X||Y|m(|E| + Km))$  time in total for  $|X| \times |Y|$  pairs of nodes.

**Space Complexity.** The space complexity of Algorithm 3 is dominated by  $O(m^2 + |E|)$ . We analyse it as follows. Firstly, the memory for the ‘‘Block Arnoldi-Ruhe Process’’ (Line 3) is bounded by  $O(m^2 + |E|)$ , including  $O(|E|)$  space for storing the adjacency matrix  $A$  and  $O(m^2)$  space for  $H_m$ . Then, in the iterative phase (Lines 6–11),  $O(m)$  memory is required for storing the vectors  $u^{(k)}$  and  $v^{(k)}$ . Hence, the total space is bounded by  $O(m^2 + |E| + m)$ , which is dominated by  $O(m^2 + |E|)$ .

**Choosing  $m$  and  $K$  with Guaranteed Error.** When one wants to achieve desired accuracy  $\epsilon$  such that  $\|\hat{S} - \hat{R}^{(K)}[m]\|_{\max} \leq \epsilon$ , there are two options to select appropriate values a-priori for low-dimensionality parameter  $m$  and number of iterations  $K$ :

1) One option is setting  $m = K = \lceil \log_\gamma \epsilon \rceil$ , which is in accordance with Eq.(42a) in Theorem 7. For example, to guarantee desired accuracy  $\epsilon = 0.001$ , given  $\gamma = 0.6$ , one can select

$$m = K = \lceil \log_{0.6} 0.001 \rceil = 14.$$

2) Another option is performing  $\delta$  more iterations relative to  $m$ , by setting  $m = \lceil \log_\gamma \frac{\epsilon}{2 - \gamma^\delta} \rceil$  and  $K = m + \delta$ , which is consistent with Eq.(42b) in Theorem 7. For example, to achieve desired accuracy  $\epsilon = 0.005$ , given  $\gamma = 0.6$ , we can perform  $\delta = 3$  more iterations relative to  $m$ , by choosing

$$m = \lceil \log_{0.6} \left( \frac{0.005}{2 - 0.6^3} \right) \rceil = 12 \quad \text{and} \quad K = m + \delta = 12 + 3 = 15.$$

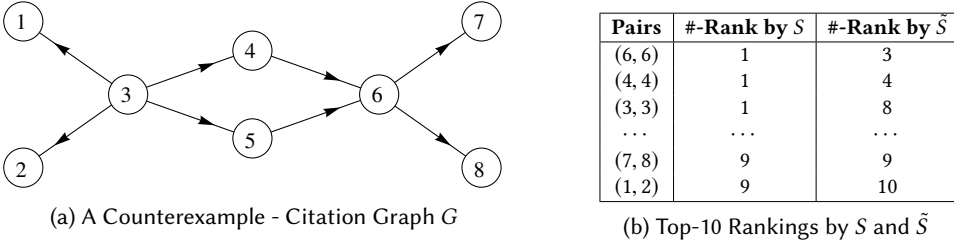


Fig. 5. A Citation Graph (a counterexample)

## 5 SEMANTIC DIFFERENCE

Apart from Jeh and Widom’s SimRank model [19]:

$$S = \max\{\gamma P^T S P, I\}, \quad (45)$$

recent years have witnessed many studies (e.g., [11, 15, 29, 48]) based on Li *et al.*’s model [29]:

$$\tilde{S} = \gamma P^T \tilde{S} P + (1 - \gamma)I. \quad (46)$$

It is worth mentioning that Li *et al.*’s SimRank Eq.(46) is essentially Kusumoto *et al.*’s model Eq.(7) that approximates the diagonal matrix  $D$  with  $(1 - \gamma)I$ . In Section 2, from the computational perspective, we have shown the difference between Li *et al.*’s SimRank and Jeh and Widom’s SimRank, implying that approximating  $D$  with  $(1 - \gamma)I$  is not accurate. In this section, we will explore further the relationship of the two models from the semantic perspective, and devise an instant formula that can accurately convert from Li *et al.*’s SimRank  $\tilde{S}$  to Jeh and Widom’s SimRank  $S$ . In particular, we will point out that the semantic meaning of Li *et al.*’s SimRank is more powerful than that of Jeh and Widom’s SimRank. As such, in Section 3.2, our “cosine-based” SimRank Eq.(30) employs  $(1 - \gamma)I$  (instead of  $D$ ) as the “seed” matrix.

### 5.1 A Fly in the Ointment of [25, 29]

There are two pieces of work [25, 29] that mentioned the relationship between  $\tilde{S}$  and  $S$ . 1) Li *et al.* [29] argued that “ $\tilde{S}$  affects only the absolute similarity value of  $S$ , but not the relative similarity ranking of  $S$ .” 2) The recent work by Kusumoto *et al.* [25] states that “ $\tilde{S}$  does not much affect the top- $K$  ranking of  $S$ .”<sup>5</sup> However, either of them implies a limitation, as disproved by the following counterexample.

**EXAMPLE 5.** Consider graph  $G$  in Figure 5a. For decay factor  $\gamma = 0.6$ , the node-pair similarity rankings by  $S$  (Jeh and Widom’s SimRank model [19]) and  $\tilde{S}$  (Li *et al.*’s model [29]) are partially shown in Figure 5b. From the results, we can discern the following.

1)  $\tilde{S}$  does not preserve the relative similarity rankings of  $S$ . In particular, for each singleton pair (e.g., (6, 6), (4, 4), (3, 3)), its similarity score by Jeh and Widom’s SimRank model is always 1, whereas Li *et al.*’s model can differentiate  $\{\tilde{S}_{x,x}\}_{x \in V}$  according to the complexity of the in-neighboring structure pointing to node  $x$ .

2) At least 5 out of top-10 rankings of  $S$  are affected by  $\tilde{S}$ . Even for non-singleton pairs (e.g., (7, 8) and (1, 2)), Jeh and Widom’s model is unable to differentiate them since two random surfers starting from two nodes 7 and 8 (resp. nodes 1 and 2) will not go any further after they meet at node 6 (resp. node 3). In contrast, for Li *et al.*’s model, after the random surfers starting from two nodes 7 and 8 meet at node 6, they continue moving against the in-coming edges of node 6 until they meet at node 3 again.

Thus, neither of the statements by [25, 29] is correct.  $\square$

<sup>5</sup>In essence,  $S \approx \tilde{S}$  is equivalent to  $D \approx (1 - \gamma)I$ .

## 5.2 Semantic Comparisons Between $S$ and $\tilde{S}$

To compare semantics between  $S$  and  $\tilde{S}$ , we first introduce the following notation:

**DEFINITION 3 (OFF-DIAGONAL OPERATOR).** For square matrix  $X$ , let  $(*)_{off}$  be a matrix operator defined by

$$(X)_{off} := X - \text{diag}(X).$$

This notation is introduced to bring new insights into  $S$ .

**THEOREM 8.** The similarity  $S$  in Jeh and Widom's model Eq.(45) can be characterized as follows:

$$\begin{aligned} S = & I + \gamma(P^\top P)_{off} + \gamma^2(P^\top(P^\top P)_{off}P)_{off} + \cdots + \\ & + \gamma^k \underbrace{(P^\top \cdots (P^\top(P^\top P)_{off}P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} + \cdots \end{aligned} \quad (47)$$

**PROOF.** Applying  $(*)_{off}$ , Eq.(45) can be iterated as

$$S_k = \gamma(P^\top S_{k-1}P)_{off} + I. \quad (48)$$

We now construct the iterations: starting with  $R_0 = I$ ,

$$R_k = \gamma^k \underbrace{(P^\top \cdots (P^\top(P^\top P)_{off}P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} + R_{k-1}. \quad (49)$$

Using induction on  $k$ , we next show that  $S_k = R_k$  ( $\forall k$ ). Clearly,  $S_0 = I = R_0$ . Assume  $S_k = R_k$  holds, we consider

$$\begin{aligned} S_{k+1} &= \gamma(P^\top S_k P)_{off} + I \quad (\text{using the hypothesis } S_k = R_k) \\ &= \gamma^{k+1} \underbrace{(P^\top(P^\top \cdots (P^\top(P^\top P)_{off}P)_{off} \cdots P)_{off})_{off}}_{k \text{ nested } (*)_{off}} + \underbrace{\gamma(P^\top R_{k-1}P)_{off} + I}_{=\{\text{using Eq.(48)}\}} \\ &= \gamma^{k+1} \underbrace{(P^\top(P^\top \cdots (P^\top(P^\top P)_{off}P)_{off} \cdots P)_{off})_{off}}_{(k+1) \text{ nested } (*)_{off}} + R_k = R_{k+1}. \end{aligned}$$

which completes the proof.  $\square$

By Theorem 8,  $S$  in Eq.(45) is the weighted sums of

$$\underbrace{(P^\top \cdots (P^\top(P^\top P)_{off}P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} \quad \forall k = 1, 2, \dots \quad (50)$$

In contrast,  $\tilde{S}$  in Eq.(46) is the weighted sums of the terms

$$\underbrace{(P^\top \cdots (P^\top(P^\top P)P) \cdots P)}_{k \text{ nested brackets}} \quad \forall k = 1, 2, \dots \quad (51)$$

To investigate the semantic relationship between  $S$  and  $\tilde{S}$ , we merely need to compare the paths tallied by (50) and (51), as shown below.

**THEOREM 9.** Given a graph  $G$ , the terms in Eq.(50) tally the following paths in  $G$ :

$$x_0 \leftarrow \underbrace{x_1 \leftarrow \cdots \leftarrow x_{k-1} \leftarrow \boxed{x_k}}_{k \text{ edges}} \rightarrow \underbrace{x_{k+1} \rightarrow \cdots \rightarrow x_{2k-1} \rightarrow x_{2k}}_{k \text{ edges}} \quad (52)$$









|  | $k = 0$ | $k = 1$  |  | $k = 2$  |  |  | ...  |     |
|--|---------|--|--|--|--|--|--|-----|
| Li <i>et al.</i> 's<br>SimRank<br>Variation<br>$\tilde{S}_k$ | $i(j)$  | ①<br> | ②<br> | ③<br> | ④<br> | ⑤<br> | ⑥<br> | ... |
| Jeh and<br>Widom's<br>SimRank<br>$S_k$                       | $i(j)$  |       |  |       |  |  | ...  |     |

 Fig. 6. Different Paths Tallied by  $S$  and  $\tilde{S}$ 

where  $x_0, \dots, x_{2k}$  can be any nodes, but with no repetition of nodes  $x_i$  and  $x_{2k-i}$  allowed,  $\forall i \in \{0, 1, \dots, 2k\} - \{k\}$ . In comparison, the terms in Eq.(51) tally the paths of (52) in  $G$  without having such a constraint on nodes  $x_i$  and  $x_{2k-i}$ .

PROOF. By the power property of the adjacency matrix,  $((P^k)^\top P^k)_{i,j}$  tallies the paths of (52) between  $i$  and  $j$ . To show the terms in Eq.(50) tally the paths of (52) with the additional constraint, we use induction on  $k$  as follows.

When  $k = 1$ ,  $((P^\top P)_{\text{off}})_{i,j} = (P^\top P)_{i,j}$  for  $i \neq j$ , and 0 for  $i = j$ . Thus,  $((P^\top P)_{\text{off}})_{i,j}$  tallies  $i \leftarrow x \rightarrow j$  with  $i \neq j$ .

Assume that, for the fixed  $k$ , the term

$$E_k := \underbrace{(P^\top \cdots (P^\top (P^\top P)_{\text{off}} P)_{\text{off}} \cdots P)_{\text{off}}}_{k \text{ nested } (*)_{\text{off}}}$$

tallies the length- $2k$  paths (52) with no repetition of nodes  $x_l$  and  $x_{2k-l}$  ( $\forall l$ ). We now consider the term  $E_{k+1}$  for  $k + 1$ . Due to  $(E_{k+1})_{i,j} = (P^\top)_{i,*} E_k (P)_{*,j}$  if  $i \neq j$ , and 0 if  $i = j$ ,  $(E_{k+1})_{i,j}$  tallies the length- $(2k + 2)$  paths concatenated by  $i \leftarrow x_0$ , paths (52), and  $x_{2k} \rightarrow j$ , which is

$$i \leftarrow x_0 \leftarrow \underbrace{x_1 \leftarrow \cdots \leftarrow x_{k-1} \leftarrow \boxed{x_k}}_{k \text{ edges}} \rightarrow \underbrace{x_{k+1} \rightarrow \cdots \rightarrow x_{2k-1} \rightarrow x_{2k}}_{k \text{ edges}} \rightarrow j$$

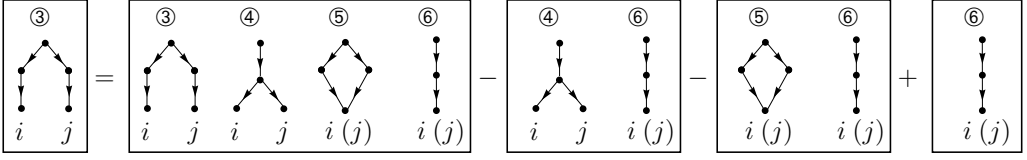
with no repetition of nodes  $x_i$  and  $x_{2k-i}$  and  $i \neq j$ . □

In light of Theorem 9, the semantic relationship between  $S$  and  $\tilde{S}$  is evident:  $\tilde{S}$  often aggregates more paths than  $S$ , and  $S$  excludes the paths with self-intersecting nodes that are considered by  $\tilde{S}$ . Figure 6 depicts an illustrative comparison of the paths tallied by  $(S_k)_{i,j}$  and  $(\tilde{S}_k)_{i,j}$  for  $k = 0, 1, 2$ .

For verification, let us apply  $(*)_{\text{off}}$  definition to expand, e.g., the term  $(P^\top (P^\top P)_{\text{off}} P)_{\text{off}}$  as follows:

$$\begin{aligned} \underbrace{((P^\top (P^\top P)_{\text{off}} P)_{\text{off}})_{i,j}}_{\textcircled{3}} &= \underbrace{((P^2)^\top P^2)_{i,j}}_{\textcircled{3} \textcircled{4} \textcircled{5} \textcircled{6}} - \underbrace{(P^\top \text{diag}(P^\top P) P)_{i,j}}_{\textcircled{4} \textcircled{6}} \\ &\quad - \underbrace{(\text{diag}((P^2)^\top P^2))_{i,j}}_{\textcircled{5} \textcircled{6}} + \underbrace{(\text{diag}(P^\top \text{diag}(P^\top P) P))_{i,j}}_{\textcircled{6}} \end{aligned}$$

where a circled number beneath each term is associated with a path numbered in the upper-left corner of Figure 6.



The following result shows the specific types of paths that are tallied by  $\tilde{S}$  but not by  $S$ .

**COROLLARY 2.** *Let  $\mathcal{P}(S)$  and  $\mathcal{P}(\tilde{S})$  be the sets of paths tallied by  $S$  and  $\tilde{S}$ , respectively. Then,  $\mathcal{P}(\tilde{S}) \supseteq \mathcal{P}(S)$ , and  $\mathcal{P}(\tilde{S}) - \mathcal{P}(S)$  is the set of “special” cycles of length  $2k$  ( $k = 1, 2, \dots$ ), with first  $k$  contiguous edges oriented in one direction, and next  $k$  contiguous edges in the opposite direction.*

### 5.3 Converting from $\tilde{S}$ to $S$

Having investigated the semantic relationship between  $\tilde{S}$  and  $S$ , we next show how to convert from Li *et al.*'s SimRank  $\tilde{S}$  to Jeh and Widom's SimRank  $S$  without any loss of accuracy.

It has been noticed from Corollary 2 that  $S_{a,b}$  counts only non-self-intersecting pairs of paths between nodes  $a$  and  $b$ , whereas  $\tilde{S}_{a,b}$  counts all the pairs of paths that include both non-self-intersecting and self-intersecting ones. The self-intersecting symmetric paths contain at least one “special” cycle at each intersecting node (e.g., the symmetric paths  $7 \leftarrow 6 \leftarrow 4 \leftarrow \boxed{3} \rightarrow 5 \rightarrow 6 \rightarrow 8$  in Figure 5a contains a “special” cycle  $6 \leftarrow 4 \leftarrow \boxed{3} \rightarrow 5 \rightarrow 6$  at the intersecting node 6). It is the contributions of these “special” cycles that make the similarity values of  $\tilde{S}$  and  $S$  different. Therefore, to convert from  $\tilde{S}$  to  $S$ , our main idea is removing all the similarity contributions of these “special” cycles from the pairs of paths counted by  $\tilde{S}$ .

There are two challenges for quantifying the similarities contributed by these “special” cycles: 1) Each self-intersecting pair of paths may have more than one intersecting node, implying that there may exist many “special” cycles in a single pair of paths. Since cycles may have variable length, and some edges in variable-length cycles may overlap each other, there is a pressing need for an effective method to avoid tallying duplicate edges in different cycles for each pair of paths. 2) Some “special” cycles counted by  $\tilde{S}$  may have repeated nodes between two branches of a path-pair, leading to such “special” cycles being degraded into a single “repeated” path (which we call “degenerated cycles”. For instance in Figure 5a, there is a degenerated cycle  $6 \leftarrow 4 \leftarrow \boxed{3} \rightarrow 4 \rightarrow 6$  at the intersecting node 6, which is degraded into a single “repeated” path  $3 \rightarrow 4 \rightarrow 6$ . Another degenerated cycle is  $6 \leftarrow \boxed{4} \rightarrow 6$ , which also reduces to a single “repeated” edge  $4 \rightarrow 6$ . When such degenerated cycles reduce to a single “repeated” path (or edge), it is imperative to find a way to differentiate these “repeated” paths from the original simple paths, which is a nontrivial task.

To address the above challenges, we propose the following theorem that effectively converts from  $\tilde{S}$  to  $S$  with no compromise in accuracy.

**THEOREM 10.** *Given Li *et al.*'s SimRank matrix  $\tilde{S}$  on graph  $G$ , we can obtain Jeh and Widom's SimRank matrix  $S$  from  $\tilde{S}$  as follows:*

*First, we iteratively construct a sequence of vectors  $\{v_l\}$  ( $l = 0, 1, 2, \dots$ ):*

$$v_0 = \frac{1}{1-\gamma} \overrightarrow{\text{diag}}(\tilde{S}) - \vec{1} \quad \text{and} \quad v_l = \sum_{k=1}^{\infty} \gamma^k (P^k \circ P^k)^\top v_{l-1} \quad (l = 1, 2, \dots) \quad (53)$$

*Next, let  $w$  be an auxiliary vector generated by  $\{v_l\}$  as*

$$w = \sum_{l=0}^{\infty} (-1)^l v_l \quad (54)$$

*Then,  $S$  can be derived from  $\tilde{S}$  as*

$$S = \frac{1}{1-\gamma} \tilde{S} - \Delta S \quad \text{with} \quad \Delta S = \sum_{k=0}^{\infty} \gamma^k (P^\top)^k \text{diag}(w) P^k \quad (55)$$

PROOF. By definition, Jeh and Widom's SimRank equation can be rewritten as

$$S = \gamma P^T S P \vee I \Leftrightarrow S = \gamma P^T S P + D \quad \text{with} \quad D = I - \gamma \text{diag}(P^T S P) \quad (56)$$

According to Theorem 1, the diagonal correction matrix  $D$  in Eq.(55) can be represented as

$$\left(\sum_{k=0}^{+\infty} \gamma^k (P^k \circ P^k)\right)^T \overrightarrow{\text{diag}}(D) = \vec{1} \quad (57)$$

In what follows, we shall show that the solution  $\overrightarrow{\text{diag}}(D)$  to Eq.(57) is

$$\overrightarrow{\text{diag}}(D) = \vec{1} - \sum_{l=0}^{+\infty} (-1)^l v_l \quad (58)$$

To this end, we plug Eq.(58) into the left-hand side of Eq.(57), which produces

$$\begin{aligned} & \left(\sum_{k=0}^{+\infty} \gamma^k (P^k \circ P^k)\right)^T \left(\vec{1} - \left(\sum_{l=0}^{+\infty} (-1)^l v_l\right)\right) \\ &= \underbrace{\sum_{k=0}^{+\infty} \gamma^k (P^k \circ P^k)^T \vec{1}}_{\text{using Lemma 2}} - \underbrace{\sum_{k=0}^{+\infty} \sum_{l=0}^{+\infty} (-1)^l \gamma^k (P^k \circ P^k)^T v_l}_{\text{interchange the order of summation}} \\ &= \text{diag} \left( \underbrace{\sum_{k=0}^{+\infty} \gamma^k (P^k)^T P^k}_{=1/(1-\gamma)\tilde{S}} - \sum_{l=0}^{+\infty} (-1)^l \left( v_l + \underbrace{\sum_{k=1}^{+\infty} \gamma^k (P^k \circ P^k)^T v_l}_{=v_{l+1}} \right) \right) \\ &= \frac{1}{1-\gamma} \text{diag}(\tilde{S}) - \underbrace{\left( \sum_{l=0}^{+\infty} (-1)^l v_l + \sum_{l=0}^{+\infty} (-1)^l v_{l+1} \right)}_{=\sum_{l=0}^{+\infty} (-1)^l v_l - \sum_{l=1}^{+\infty} (-1)^l v_l = v_0} = \frac{1}{1-\gamma} \text{diag}(\tilde{S}) - v_0 = \vec{1} \end{aligned}$$

Combining this with Eq.(56), we can rewrite Jeh and Widom's SimRank equation as

$$S = \sum_{l=0}^{+\infty} \gamma^k (P^T)^k D P^k \quad \text{with} \quad D = I - \text{diag}(w) \quad \text{and} \quad w = \sum_{l=0}^{+\infty} (-1)^l v_l$$

Therefore, it follows that

$$\begin{aligned} S &= \sum_{l=0}^{+\infty} \gamma^k (P^T)^k (I - \text{diag}(w)) P^k \\ &= \underbrace{\sum_{l=0}^{+\infty} \gamma^k (P^T)^k P^k}_{=1/(1-\gamma)\tilde{S}} - \underbrace{\sum_{l=0}^{+\infty} \gamma^k (P^T)^k \text{diag}(w) P^k}_{=\Delta S} = \frac{1}{1-\gamma} \tilde{S} - \Delta S \quad \square \end{aligned}$$

Theorem 10 provides an effective approach through which we can make instant conversion from Li *et al.*'s SimRank  $\tilde{S}$  to Jeh and Widom's SimRank  $S$  without any loss of accuracy. The key intuition underpinning our approach is pictorially depicted in Figure 7. It can be noticed that each  $v_i$  ( $i = 0, 1, 2, \dots$ ) in Eq.(53) tallies the "special" cycles in which two surfers, starting from the same node and moving against incoming edges, will meet again at least  $(i + 1)$  times. That is, the pair of paths tallied by  $v_i$  ( $i = 0, 1, 2, \dots$ ) has at least  $(i + 2)$  self-intersecting nodes. For instance in Figure 7, each pair of paths tallied by  $v_0$  has at least 2 self-intersecting nodes. To be specific, the path-pairs tagged by ③, ⑤, ⑧ (*resp.* ⑦, ⑨, ⑩) in  $v_0$  have 2 (*resp.* 3) self-intersecting nodes; and the path-pair tagged by ① in  $v_0$  has 4 self-intersecting nodes (*i.e.*, two branches are totally overlapped). Similarly, the pair of paths tallied by  $v_1$  (*resp.*  $v_2$ ) has at least 3 (*resp.* 4) self-intersecting nodes.

To guarantee the pairs of paths with exactly 2 self-intersecting nodes (*i.e.*, the start and end nodes) are the only pairs tallied by  $w$ , we use the principle of inclusion and exclusion by setting  $w = v_0 - v_1 + v_2 - \dots$ , which can avoid overcounting or undercounting the path-pairs. For instance, the pair of paths with 4 self-intersecting nodes, tagged by ①, should not be counted by  $w$ , but ① has appeared once in  $v_0$ , twice in  $v_1$ , and again once in  $v_2$ , respectively. Thus, to avoid ① appearing in  $w$ ,  $v_0 - v_1$  would undercount the path ① once, whereas ① in  $v_0 - v_1 + v_2$  can be cancelled out in

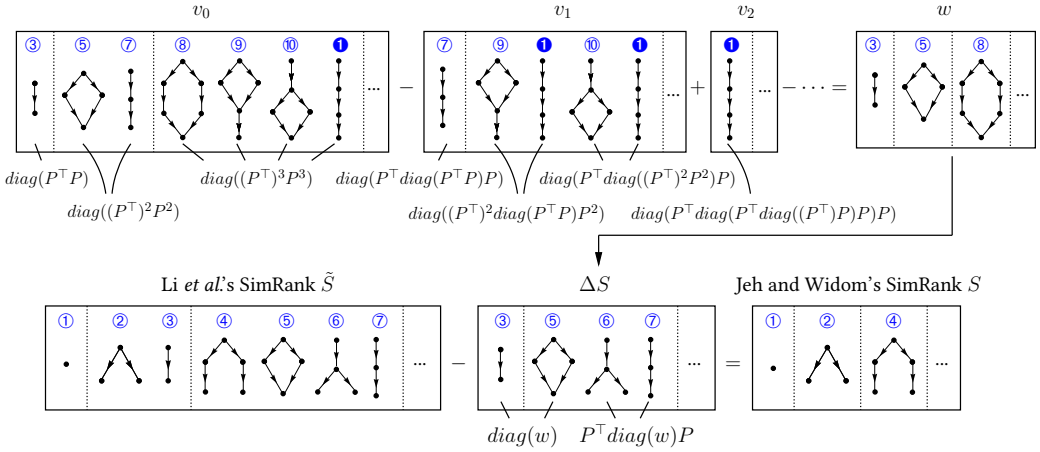


Fig. 7. Key Intuition Behind Conversion from Li *et al.*'s SimRank  $\tilde{S}$  to Jeh and Widom's SimRank  $S$

an appropriate way. As a result,  $w$  tallies all the non-degenerate “special” cycles in  $\mathcal{P}(\tilde{S}) - \mathcal{P}(S)$  as specified by Corollary 2, that is, each pair of paths in  $w$  has exactly 2 self-intersecting nodes (*i.e.*, only the start and end nodes in the pair of paths can repeat). Therefore,  $\Delta S$  in Eq.(55) tallies the pair of paths with at least 2 self-intersecting nodes, meaning that two surfers, starting from two distinct nodes (*resp.* the same node) and moving against incoming edges, will meet more than (*resp.* at least) once. Since Li *et al.*'s SimRank  $\tilde{S}$  counts all the meeting times of the two surfers, subtracting  $\Delta S$  from  $\tilde{S}$  will result in Jeh and Widom's SimRank  $S$ , in which only the first meeting time of the two surfers is counted.

## 6 “COSINE-BASED” SIMRANK BETWEEN TWO GRAPHS

In Section 3, our “cosine-based” SimRank in Eq.(30) is designed for assessing the similarity between two nodes in one graph. In many real applications, there is a pressing need to evaluate pairwise similarities for the nodes of two different graphs. Examples include pattern matching in protein-protein interaction networks (bioinformatics), chemical compounds identification (cheminformatics), and bilingual translation (NLP). These inspire us to study the problem of graph similarity search:

**PROBLEM (“Cosine-based” SimRank Similarity Search across Two Graphs).**

Given two graphs  $G_A$  and  $G_B$ , we want to assess the similarity  $s(a, b)$  with  $a \in G_A$  and  $b \in G_B$ , such that node  $a$  in graph  $G_A$  and node  $b$  in graph  $G_B$  are assessed to be similar if their respective multi-hop neighbouring structures within  $G_A$  and  $G_B$  are similar.

To address the problem, in this section, we generalise the idea of our “cosine-based” SimRank model (Eq.(30) in Section 3), and propose a new variation for our graph similarity model that can effectively measure the “cosine-based” similarity for the nodes of two distinct graphs.

### 6.1 Limitation of Existing “Cosine-Based” SimRank Model Applied to Two Graphs

It is important to note that, if our “cosine-based” SimRank model Eq.(30) is directly applied to measure the similarity across two graphs, it would produce meaningless scores, as explicated in the following example:

**EXAMPLE 6.** Consider two graphs  $G_A$  and  $G_B$  in Figure 8. We want to evaluate the similarities  $\{s(a, b)\}_{a \in G_A, b \in G_B}$  between each node  $a$  in  $G_A$  and each node  $b$  in  $G_B$ . If we regard  $G_A$  and  $G_B$  as two disconnected components of the whole graph  $G := G_A \cup G_B$  and directly apply our “cosine-based”



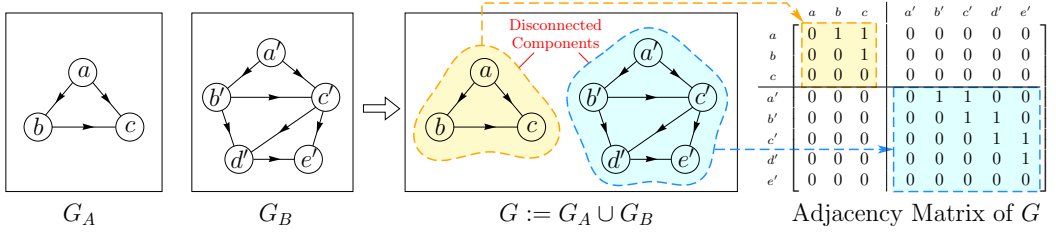


Fig. 8. Limitation of “cosine-based” SimRank model to assess similarity of nodes across graphs  $G_A$  and  $G_B$ . For any node  $x \in G_A$  and  $y \in G_B$ , since there are no connected paths between  $x$  and  $y$ , the “cosine-based” SimRank model applied to  $G := G_A \cup G_B$  would always produce  $s(x, y) = 0$ , which is rather counterintuitive.

SimRank model Eq.(30) (or Jeh and Widom’s SimRank model Eq.(45)) to the entire graph  $G$ , then these models would produce zero similarities, i.e.,  $s(a, b) = 0$  for all  $a \in G_A$  and  $b \in G_B$ . The reason is that there are no connected paths between nodes  $a$  and  $b$ . Mathematically, for example, since

$$e_a^\top \begin{bmatrix} (A^\top)^k & 0 \\ 0 & (B^\top)^k \end{bmatrix} \begin{bmatrix} A^k & 0 \\ 0 & B^k \end{bmatrix} e_b = e_a^\top \begin{bmatrix} (A^\top)^k A^k & 0 \\ 0 & (B^\top)^k B^k \end{bmatrix} e_b = 0 \quad (\forall k \geq 0, \forall a \in G_A, \forall b \in G_B)$$

our “cosine-based” SimRank model Eq.(30), when applied to  $G$ , becomes

$$s(a, b) = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \frac{e_a^\top \left( \begin{bmatrix} A & \\ & B \end{bmatrix}^\top \right)^k \begin{bmatrix} A & \\ & B \end{bmatrix}^k e_b}{\left\| \begin{bmatrix} A & \\ & B \end{bmatrix}^k e_a \right\|_2 \left\| \begin{bmatrix} A & \\ & B \end{bmatrix}^k e_b \right\|_2} = 0 \quad (\forall a \in G_A, \forall b \in G_B)$$

which leads to meaningless similarity results. Jeh and Widom’s SimRank implies a similar problem.  $\square$

Example 6 indicates that, when nodes  $a$  and  $b$  are in two disconnected components (i.e., there are no connected paths between  $a$  and  $b$ ), it is imperative to make modifications to our “cosine-based” SimRank model Eq.(30) to evaluate similarity for the nodes of two different graphs.

## 6.2 Generalised “Cosine-Based” SimRank Between Graphs

Before illustrating our generalised “cosine-based” SimRank model to two graphs, we first introduce the notion of *in- and out-linkage matrices* between two graphs.

**DEFINITION 4.** Given two digraphs  $G_A$  and  $G_B$ , let  $n_A$  (resp.  $n_B$ ) be the number of nodes in  $G_A$  (resp.  $G_B$ ). The in-linkage matrix  $E^-$  (resp. out-linkage matrix  $E^+$ ) between  $G_A$  and  $G_B$  is a matrix of the size  $n_A \times n_B$ , whose each  $(i, j)$ -entry ( $i \in G_A, j \in G_B$ ) is defined as follows:

$$(E^-)_{i,j} = f(|N_i^-|, |N_j^-|) \quad \text{and} \quad (E^+)_{i,j} = f(|N_i^+|, |N_j^+|),$$

$$\text{with } f(x, y) = \begin{cases} \frac{x+y}{2 \times \max(x, y)}, & \text{if } (x, y) \neq (0, 0); \\ 1, & \text{if } (x, y) = (0, 0). \end{cases} \quad (59)$$

where  $|N_i^-|$  and  $|N_i^+|$  denote the in-degree and out-degree of node  $i$ , respectively.

**EXAMPLE 7.** Recall the two graphs  $G_A$  and  $G_B$  in Figure 8. Since  $|N_c^-| = 2$  and  $|N_{b'}^-| = 1$ , it follows from Eq.(59) that

$$(E^-)_{c,b'} = \frac{|N_c^-| + |N_{b'}^-|}{2 \times \max(|N_c^-|, |N_{b'}^-|)} = \frac{2+1}{2 \times \max(2,1)} = \frac{3}{4}. \quad \square$$

It is worth noticing that the function  $f(x, y)$  defined by Eq.(59) has some key properties, as indicated by Lemma 4.

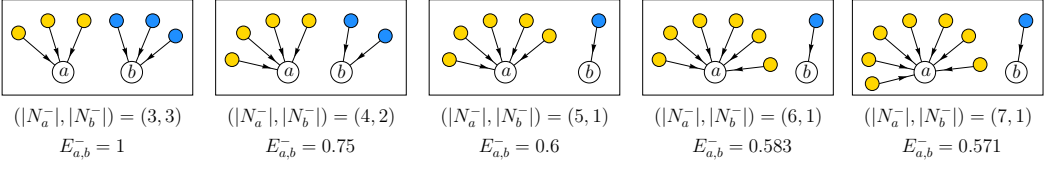


Fig. 9. Illustrative examples of Properties 3 and 4 in Lemma 4. The first (resp. last) three subfigures depict Property 3 for  $|N_a^-| + |N_b^-| \equiv 6$  (resp. Property 4 for fixing  $|N_b^-| = 1$  and varying  $|N_a^-|$ ). The more symmetric the in-neighboring structure between nodes  $a$  and  $b$ , the higher the initial in-linkage similarity value  $E_{a,b}^-$ .

LEMMA 4. For any integers  $x = 0, 1, 2, \dots$  and  $y = 0, 1, 2, \dots$ , the function  $f(x, y)$  defined by Eq.(59) has the following properties:

- (1)  $f(x, y)$  is nonnegative, i.e.,  $0 \leq f(x, y) \leq 1$ . Particularly,  $f(x, y) = 1$  if and only if  $x = y$ .
- (2)  $f(x, y)$  is symmetric, i.e.,  $f(x, y) = f(y, x)$ .
- (3) When  $x + y$  (= constant) is fixed,  $f(x, y)$  is increasing as  $|x - y|$  is decreasing.
- (4) When  $x$  is fixed,  $f(x, y)$  is increasing w.r.t.  $y$  on the interval  $y \in [0, x]$ , and decreasing w.r.t.  $y$  on the interval  $y \in [x, +\infty)$ .

The properties of  $f(x, y)$  in Lemma 4 indicate that the in- and out-linkage matrices  $(E^-)_{i,j}$  and  $(E^+)_{i,j}$  in Definition 4 can be regarded as a rudimentary measure of the (local) similarity between node  $i \in G_A$  and  $j \in G_B$  that is based on  $i$ 's and  $j$ 's 1-hop in- and out-neighborhood structures only. For example, as depicted in Figure 9, Properties 3 and 4 imply that  $(E^-)_{i,j}$  (resp.  $(E^+)_{i,j}$ ) will assess nodes  $i$  and  $j$  as similar if their 1-hop in- (resp. out-) neighborhood structures (i.e., the in-degrees (resp. out-degrees) of  $i$  and  $j$ ) are similar. However, if we simply rely on  $(E^-)_{i,j}$  and  $(E^+)_{i,j}$  to evaluate the similarity between nodes  $i$  and  $j$  in two different graphs, the multi-hop neighborhood structures of  $i$  and  $j$  will be ignored. Therefore, in what follows, we will regard  $(E^-)_{i,j}$  and  $(E^+)_{i,j}$  as the (1-hop) seed similarity values, and propagate them iteratively through our ‘‘cosine-based’’ SimRank model that can effectively capture the multi-hop neighborhood information.

Utilising in- and out-linkage matrices  $E^-$  and  $E^+$ , we next generalise our ‘‘cosine-based’’ SimRank model to evaluate the similarity of nodes between two graphs.

DEFINITION 5. Let  $A$  (resp.  $B$ ) be the adjacency matrix of graph  $G_A$  (resp.  $G_B$ ), and  $E^+$  (resp.  $E^-$ ) be the out- (resp. in-) linkage matrix between  $G_A$  and  $G_B$  defined by Eq.(59). The generalised ‘‘cosine-based’’ SimRank similarity  $\tilde{S}_{a,b}$  between node  $a$  in  $G_A$  and node  $b$  in  $G_B$  is defined by

$$\tilde{S}_{a,b} = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \left( \beta \times \frac{e_a^\top (A^\top)^k E^+ B^k e_b}{\|A^k e_a\|_1 \|B^k e_b\|_1} + (1 - \beta) \times \frac{e_a^\top A^k E^- (B^\top)^k e_b}{\|(A^\top)^k e_a\|_1 \|(B^\top)^k e_b\|_1} \right) \quad (60)$$

where  $\beta \in [0, 1]$  is a user-controlled weight factor that balances the importance of in- and out-links. By default,  $\beta$  is set to 0.5, indicating the contributions from in- and out-links are of equal importance.

Definition 5 effectively quantifies the similarity between nodes of different graphs, which is a generalisation of Definition 2 that measures the similarity between nodes in the same graph. Similar to Definition 2, our extension in Definition 5 also considers multi-hop neighbourhood information of nodes  $a$  and  $b$  to evaluate the similarity between  $a$  and  $b$ . If we set  $\beta = 1$ ,  $A = B$ ,  $E^+ = E^- = I$  and replace  $L_2$ -norm  $\| \cdot \|_2$  with  $L_1$ -norm  $\| \cdot \|_1$  in Eq.(60), then  $\tilde{S}_{a,b}$  reduces to  $\hat{S}_{a,b}$  in Eq.(30). The differences between the model in Definition 2 and our extension in Definition 5 are three-fold:

- (1) To connect each  $k$ -hop neighbour of node  $a$  with that of node  $b$ , Definition 2 directly concatenates  $e_a^\top (A^\top)^k$  and  $A^k e_b$ , which can be viewed as using identity matrix  $I$  to link them:

$$e_a^\top (A^\top)^k A^k e_b = \underbrace{(e_a^\top (A^\top)^k)}_{\text{hop}_k(a)} \cdot \underbrace{(A^k e_b)}_{\text{hop}_k(b)} = \sum_{i \in G} (A^k e_a)_i \cdot (A^k e_b)_i$$

whereas Definition 5 utilises in-linkage matrix  $E^+$  (resp. out-linkage matrix  $E^-$ ) to concatenate  $e_a^\top (A^\top)^k$  and  $B^k e_b$  (resp.  $e_a^\top A^k$  and  $(B^\top)^k e_b$ ), e.g.,

$$\underbrace{(e_a^\top (A^\top)^k)}_{\text{hop}_k(a)} \cdot \underbrace{(B^k e_b)}_{\text{hop}_k(b)} = \sum_{i \in G_A} \sum_{j \in G_B} (A^k e_a)_i \cdot (E^+)_{i,j} \cdot (B^k e_b)_j$$

This subtle difference enables our extended model in Definition 5 to avoid producing zero similarity even if there are no connected paths between  $a$  and  $b$ , which is due to the introduction of  $E^+$  (resp.  $E^-$ ) that allows  $e_a^\top (A^\top)^k E^+ B^k e_b$  (resp.  $e_a^\top A^k E^- (B^\top)^k e_b$ ) not being zero even if there is no intersection between  $a$ 's and  $b$ 's  $k$ -hop neighbourhoods, as will be shortly shown by Theorem 11.

- (2) To normalise the terms  $e_a^\top (A^\top)^k E^+ B^k e_b$  and  $e_a^\top A^k E^- (B^\top)^k e_b$ , Eq.(60) utilises  $L_2$ -norm  $\| * \|_2$  as opposed to Eq.(30) using  $L_1$ -norm  $\| * \|_1$ . The reason is that the in- and out-linkage matrices  $E^-$  and  $E^+$  would maximally become two matrices of all 1s. Consequently,  $E^+ \leq 1 \cdot 1^\top$  implies

$$e_a^\top (A^\top)^k \underbrace{E^+}_{\leq 1 \cdot 1^\top} B^k e_b \leq \underbrace{(e_a^\top (A^\top)^k 1)}_{=\sum_{i \in G_A} (Ae_a)_i} \cdot \underbrace{(1^\top B^k e_b)}_{=\sum_{j \in G_B} (Be_b)_j} = \|A^k e_a\|_1 \cdot \|B^k e_b\|_1 \quad (61)$$

Similarly, it follows from  $E^- \leq 1 \cdot 1^\top$  that

$$e_a^\top A^k E^- (B^\top)^k e_b \leq (e_a^\top A^k 1) \cdot (1^\top (B^\top)^k e_b) = \|(A^\top)^k e_a\|_1 \cdot \|(B^\top)^k e_b\|_1 \quad (62)$$

Note that the equality signs “=” in Eqs.(61) and (62) are attainable when  $E^+$  and  $E^-$  are two matrices of all 1s, i.e.,  $G_A$  and  $G_B$  are two identical complete graphs. Thus,  $L_1$ -norm  $\| * \|_1$  is a tight bound for our extended model in Definition 5. In comparison, Definition 2 uses  $L_2$ -norm  $\| * \|_2$  for normalisation since the identity matrix  $I$  is used for concatenating  $e_a^\top (A^\top)^k$  and  $A^k e_b$ . As a result, according to the Cauchy–Schwarz inequality, we have

$$(e_a^\top (A^\top)^k) \cdot I \cdot (A^k e_b) = (e_a^\top (A^\top)^k) \cdot (A^k e_b) \leq \|A^k e_a\|_2 \cdot \|A^k e_b\|_2 \quad (63)$$

The “=” sign in the last inequality of Eq.(63) is attainable when  $A^k e_a = A^k e_b$  ( $\forall k$ ), i.e.,  $a = b$ .

- (3)  $\tilde{S}_{a,b}$  considers the contributions from both incoming and outgoing multi-hop neighbourhoods by introducing a user-specified weight factor  $\beta$  that balances the importance between in- and out-links, as opposed to Definition 2 that hinges only on incoming multi-hop neighbourhoods.

Based on the above analysis, we next provide the following theorem, which highlights the key advantage of our extended model in Definition 5 over its counterpart in Definition 2.

**THEOREM 11.** *To evaluate the similarity between nodes of two different graphs, our generalised “cosine-based” SimRank similarity model in Eq.(60) will alleviate the “zero-similarity” problem of the existing “cosine-based” SimRank model in Eq.(30).*

**PROOF.** For any graph  $G$  and node  $x$  in  $G$ , we denote by

$$\text{hop}_k^-(x|G) = \{i \in G | (A^k e_x)_i > 0\} \quad \text{and} \quad \text{hop}_k^+(x|G) = \{i \in G | ((A^\top)^k e_x)_i > 0\}$$

the  $k$ -hop in- and out-neighbor set of node  $x$  in graph  $G$ , respectively. Let  $G_A$  and  $G_B$  be two graphs, and  $a$  and  $b$  be two nodes in  $G_A$  and  $G_B$ , respectively. Applying  $\text{hop}_k^-(x|G)$  and  $\text{hop}_k^+(x|G)$  to Definition 5 yields

$$\begin{aligned} e_a^\top (A^\top)^k E^+ B^k e_b &= \sum_{i \in G_A} \sum_{j \in G_B} (\text{hop}_k^-(a|G_A))_i \cdot (E^+)_{i,j} \cdot (\text{hop}_k^-(b|G_B))_j \\ e_a^\top A^k E^- (B^\top)^k e_b &= \sum_{i \in G_A} \sum_{j \in G_B} (\text{hop}_k^+(a|G_A))_i \cdot (E^-)_{i,j} \cdot (\text{hop}_k^+(b|G_B))_j \end{aligned}$$

From the above equations, it can be discerned that  $\text{hop}_k^-(a|G_A) \cap \text{hop}_k^-(b|G_B) = \emptyset$  does not imply that  $\sum_{i \in G_A} \sum_{j \in G_B} (\text{hop}_k^-(a|G_A))_i \cdot (E^+)_{i,j} \cdot (\text{hop}_k^-(b|G_B))_j = 0$ . Thus, even if there are no connections between  $a$  and  $b$ , the terms  $e_a^\top (A^\top)^k E^+ B^k e_b$  and  $e_a^\top A^k E^- (B^\top)^k e_b$  could be non-zeros.

In contrast, setting  $G := G_A \cup G_B$  and applying  $\text{hop}_k^-(x|G)$  to Definition 2 yields

$$\begin{aligned} e_a^\top \left( \begin{bmatrix} A & \\ & B \end{bmatrix}^\top \right)^k \begin{bmatrix} A & \\ & B \end{bmatrix}^k e_b &= \sum_{i \in G} (\text{hop}_k^-(a|G))_i \cdot (\text{hop}_k^-(b|G))_i \\ &= |\text{hop}_k^-(a|G_A) \cap \text{hop}_k^-(b|G_B)| = 0 \end{aligned}$$

The last equality holds because  $a$  and  $b$  are in two different connected components  $G_A$  and  $G_B$  of  $G$ . Therefore, Eq.(60) alleviates the ‘‘zero-similarity’’ problem of Eq.(30).  $\square$

### 6.3 Efficient Computation of Generalised ‘‘Cosine-Based’’ SimRank Similarity

To compute our generalised ‘‘cosine-based’’ SimRank  $\tilde{S}_{a,b}$  in Eq.(60), we can use the following iterative method:

$$\tilde{S}_{a,b}^{(k)} = \tilde{S}_{a,b}^{(k-1)} + (1 - \gamma)\gamma^k \left( \beta(u^{(k)})^\top E^+ v^{(k)} + (1 - \beta)(u'^{(k)})^\top E^- v'^{(k)} \right) \quad \text{with } \tilde{S}_{a,b}^{(0)} = 0 \quad (64)$$

where the auxiliary vectors  $u^{(k)}$  and  $v^{(k)}$  are iteratively obtained by

$$\begin{cases} u^{(0)} = e_a \\ u^{(k)} = \frac{A u^{(k-1)}}{\|A u^{(k-1)}\|_1} \end{cases} \quad \begin{cases} v^{(0)} = e_b \\ v^{(k)} = \frac{B v^{(k-1)}}{\|B v^{(k-1)}\|_1} \end{cases} \quad \begin{cases} u'^{(0)} = e_a \\ u'^{(k)} = \frac{A^\top u'^{(k-1)}}{\|A^\top u'^{(k-1)}\|_1} \end{cases} \quad \begin{cases} v'^{(0)} = e_b \\ v'^{(k)} = \frac{B^\top v'^{(k-1)}}{\|B^\top v'^{(k-1)}\|_1} \end{cases} \quad (65)$$

It can be readily verified that the  $k$ -th iterative similarity  $\tilde{S}_{a,b}^{(k)}$  in Eq.(64) converges to the exact solution  $\hat{S}_{a,b}$  in Eq.(60) as  $k \rightarrow +\infty$ . The computational complexity of the above iterative method in Eqs.(64) and (65) is bounded by  $O(K(|E| + |V|))$  time and  $O(|E| + K|V|)$  memory for  $K$  iterations, which is comparable to our ‘‘cosine-based’’ SimRank computation via Eqs.(34) and (35).

**EXAMPLE 8.** Recall the two graphs  $G_A$  and  $G_B$  in Figure 8, with the in-linkage matrix  $E^-$  and out-linkage matrix  $E^+$  obtained in Example 7. Then, setting decay factor  $\gamma = 0.8$  and weight factor  $\beta = 0.5$ , we leverage Eqs.(64) and (65) to iteratively compute our generalised ‘‘cosine-based’’ SimRank similarity, e.g.,  $\hat{S}_{a,c'} = 0.284$ , between node  $a$  in  $G_A$  and node  $c'$  in  $G_B$  as follows:

| $k$ | $v^{(k)}$        | $u^{(k)}$                | $v'^{(k)}$         | $u'^{(k)}$               | $s^{(k)}(a, c')$ |
|-----|------------------|--------------------------|--------------------|--------------------------|------------------|
| 0   | $[1, 0, 0]^\top$ | $[0, 0, 1, 0, 0]^\top$   | $[1, 0, 0]^\top$   | $[0, 0, 1, 0, 0]^\top$   | 0                |
| 1   | $[0, 0, 0]^\top$ | $[.5, .5, 0, 0, 0]^\top$ | $[0, .5, .5]^\top$ | $[0, 0, 0, .5, .5]^\top$ | 0.150            |
| 2   | $[0, 0, 0]^\top$ | $[1, 0, 0, 0, 0]^\top$   | $[0, 0, 1]^\top$   | $[0, 0, 0, 0, 1]^\top$   | 0.220            |
| 3   | $[0, 0, 0]^\top$ | $[0, 0, 0, 0, 0]^\top$   | $[0, 0, 0]^\top$   | $[0, 0, 0, 0, 0]^\top$   | 0.284            |

It is discerned that, after 3 iterations, the auxiliary vectors  $v^{(3)}$ ,  $u^{(3)}$ ,  $v'^{(3)}$ ,  $u'^{(3)}$  become zeros. Thus, the resulting similarity score  $\tilde{S}_{a,c'}^{(3)} = 0.284$  is convergent, being the exact solution.

Similarly, the similarities for the remaining pairs of nodes across the two graphs  $G_A$  and  $G_B$  are

$$\hat{S} = \begin{matrix} & a' & b' & c' & d' & e' \\ \begin{matrix} a \\ b \\ c \end{matrix} & \begin{bmatrix} .334 & .284 & .284 & .195 & .1 \\ .195 & .335 & .310 & .335 & .195 \\ .1 & .195 & .284 & .284 & .334 \end{bmatrix} & \square \end{matrix}$$

Due to commonality of the algebraical structure between our original “cosine-based” SimRank (Eqs.(34) and (35)) and our extension (Eqs.(64) and (65)), our optimization techniques proposed in Section 4 for partial-pairs “cosine-based” SimRank computation can also be slightly modified to adapt to our generalised “cosine-based” SimRank model. We omit the details here and will demonstrate in Section 7 the efficiency of our Krylov-subspace based accelerative techniques when they are incorporated into our generalised “cosine-based” SimRank model.

## 7 EXPERIMENTAL STUDIES

### 7.1 Experimental Settings

We use both real-life datasets (WikiV, CaD, CitH, WebN, ComY, SocL) and synthetic data (SYN).

**(1) Real-life Data.** We use 8 real-life datasets. Six datasets are taken from Stanford Large Network Collection (SNAP)<sup>6</sup>, and two large datasets are crawled at .uk and .it domains, which are publicly available from The Laboratory for Web Algorithmics (LAW)<sup>7</sup>. Their details are described below:

|        | Datasets (Abbr.)                  | V          | E             | E / V | Type       |
|--------|-----------------------------------|------------|---------------|-------|------------|
| small  | Wikipedia (WikiV)                 | 8,297      | 103,689       | 12.49 | Directed   |
|        | DBLP Co-authorship (CaD)          | 15,683     | 55,064        | 5.31  | Undirected |
|        | HEP Citation Graph (CitH)         | 34,546     | 421,578       | 12.20 | Directed   |
| medium | Web Graph (WebN)                  | 325,729    | 1,497,134     | 4.59  | Directed   |
|        | Youtube Graph (ComY)              | 1,134,890  | 2,987,624     | 2.63  | Undirected |
|        | LiveJournal Friendship (SocL)     | 4,847,571  | 68,993,773    | 14.23 | Directed   |
| large  | .uk Domain Crawled in 2002 (UK02) | 18,520,486 | 298,113,762   | 16.09 | Directed   |
|        | .it Domain Crawled in 2004 (IT04) | 41,291,594 | 1,150,725,436 | 27.86 | Directed   |

**(2) Synthetic Data.** To produce SYN, we adopt a scale-free graph generator based on the Barabasi-Albert model<sup>8</sup>, including *growth* and *preferential attachment*. (a) *Growth* means that the number of nodes in the network increases over time; (b) *Preferential attachment* means that the more connected a node is, the more likely it is to receive new links. This generator takes as input two parameters: ( $|V|, |E|$ ).

**(3) Query Generator.** (i) To evaluate partial pairs of similarities  $\{s(x, y)\}_{x \in A, y \in B}$ , we select the set of queries  $(A, B)$  based on two criteria:

(a) *Importance coverage* is to ensure the selected  $(A, B)$  to comprehensively contain a broad range of any possible pairs. To this end, we first sort all nodes in  $V$  in descending order by PageRank (PR), and split them into 10 buckets: nodes with  $PR \in [0.9, 1]$  are in the first bucket; nodes with  $PR \in [0.8, 0.9)$  the second, etc. We next randomly select  $\lceil \frac{1}{10}|A| \rceil$  (resp.  $\lceil \frac{1}{10}|B| \rceil$ ) nodes from each bucket to  $A$  (resp.  $B$ ). Thus,  $(A, B)$  covers both important and non-important pairs.

(b) *Overlapping coverage* is to guarantee that  $(A, B)$  contains node-pairs with many multi-hop in-neighbors overlapped. To achieve this, we first sort node-pair  $(a, b)$  in descending order via a scoring function:<sup>9</sup>  $f_{a,b} := \sum_{k=1}^5 \frac{|\text{hop}_k(a) \cap \text{hop}_k(b)|}{|\text{hop}_k(a) \cup \text{hop}_k(b)|}$ . We then split all pairs into 5 buckets: pairs with

<sup>6</sup><http://snap.stanford.edu/data/index.html>

<sup>7</sup><http://law.di.unimi.it/index.php>

<sup>8</sup><http://graphstream-project.org/doc/Generators/>

<sup>9</sup>All paths of length up to 10 between  $a$  and  $b$  can be tallied by our queries, ensuring results accurate to 2 decimal places.

$f_{a,b} \in [4, 5]$  are in the first bucket; pairs with  $f_{a,b} \in [3, 4)$  the second, etc. For each bucket, we next sort node-pair  $(a, b)$  in descending order based on the value of  $g_{a,b} := \sum_{k=1}^5 |\text{hop}_k(a) \cap \text{hop}_k(b)|$ , and select top  $\lceil \frac{1}{5}|A||B| \rceil$  node-pairs from each bucket. Hence,  $(A, B)$  covers node-pairs with many multi-hop in-neighbors in common. (ii) Similarly, to evaluate single-source  $s(\star, q)$ , the query set for  $q$  can be sampled as “importance coverage”.

**(4) Algorithms.** We implement the following algorithms:

| Algorithms        | Description  |
|-------------------|--|
| SR <sup>#</sup>   | our “cosine-based” SimRank scheme (in Section 3.2)   |
| PSR <sup>#</sup>  | our enhanced SR <sup>#</sup> using Krylov subspace (in Section 4)                          |
| GSR <sup>#</sup>  | our generalised SR <sup>#</sup> to assess node similarity across two graphs (in Section 6) |
| AGSR <sup>#</sup> | our generalised SR <sup>#</sup> using Krylov subspace (in Section 6.3)                     |
| MSR               | Kusumoto’s scalable SimRank search algorithm [25]  |
| SMAT              | single-source SimRank (matrix decomposition) [11]  |
| OIP               | all-pairs SimRank (fine-grained clustering) [47]   |
| PSUM              | all-pairs SimRank (partial sums memoization) [34]  |
| JSR               | Jeh and Widom’s SimRank [19]   |
| LSR               | Li <i>et al.</i> ’s SimRank [29]   |
| SR*               | SimRank* (tally asymmetric paths) [49]   |
| SR <sup>++</sup>  | SimRank <sup>++</sup> (revised “evidence factor”) [1]                                      |
| RS                | RoleSim (automorphism equivalence) [22]  |
| RWR               | Random Walk with Restart   |
| COS               | classic cosine similarity  |

**(5) Parameters.** We set the following parameters by default, unless stated otherwise: (a) decay factor  $\gamma = 0.6$ , as suggested in [34]. (b) number of iterations  $k = 10$ , which guarantees the resulting output  $S^{(k)}$  accurate to at least 2 decimal places. (c) weight factor  $\beta = 0.5$ , indicating the contributions from in- and out-links of GSR<sup>#</sup> are of equal importance, as suggested in Definition 5.

**(6) Evaluation Metrics.** To evaluate the quality of similarity search, we use the following metrics:

(a) *Normalized Discounted Cumulative Gain (NDCG) at position  $p$*  is defined as follows:

$$\text{NDCG}_p := \frac{1}{\text{IDCG}_p} \sum_{i=1}^p \frac{2^{\text{rel}_i - 1}}{\log_2(1+i)},$$

where  $\text{rel}_i$  is the graded relevance at position  $i$ , and  $\text{IDCG}_p$  is the ideal DCG ranking.

(b) *Spearman’s  $\rho$  rank-order correlation* is defined by  $\rho := 1 - \frac{6}{n(n^2-1)} \sum_{i=1}^n d_i^2$ , where  $d_i$  is the difference of two ranks at position  $i$ , and  $n$  is the number of elements.

(c) *Kendall’s  $\tau$  coefficient* measures the ordinal association between two rankings, given by

$$\tau := \frac{1}{0.5n(n-1)} \times [(\# \text{ of concordant pairs}) - (\# \text{ of discordant pairs})].$$

where  $0.5n(n-1) = \binom{n}{2}$  is the binomial coefficient for the number of ways of selecting two items from  $n$  items.

**(7) Ground Truth.** (a) To label ground truth for similar users on WikiV, a manual evaluation is carried out by 50 professional members who have accumulated a long history of activity on Wikipedia. Each pair of users is considered by an evaluator, and is assigned a score on a scale from 1 to 4, with 1 meaning irrelevant, 4 meaning completely relevant, and 2 and 3 meaning “somewhere in between”. The judgement is based on evaluator’s knowledge and public votes on promotion of individuals to adminship. (b) To mark ground truth labels for similar authors on CaD, 30 members from 5 database groups are invited. Each pair of authors is given a score based on the collaboration distance between authors. The judgement relies on evaluator’s knowledge and “separations” of

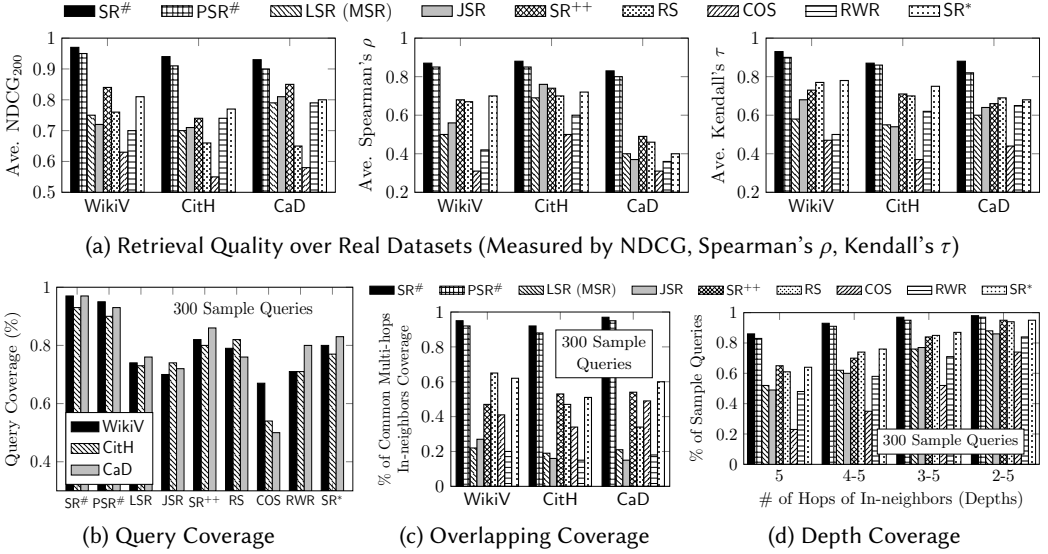


Fig. 10. Quantitative Results on Retrieval Quality over Real-life Datasets

Co-Author Path in Microsoft Academic Search.<sup>10</sup> (c) To establish the ground truth of similar articles on CitH, 28 research associates from the School of Physics are hired. Each pair of articles is assigned a score based on evaluator's knowledge on paper abstracts and citation relations.

All experiments are run with an Intel Core(TM) i7-4700MQ CPU @ 2.40GHz CPU and 64GB RAM, on Windows 8.

## 7.2 Evaluation on Retrieval Quality

**7.2.1 Quantitative Results on Retrieval Quality.** We first evaluate the high search quality of SR# and PSR# against SR++, JSR, LSR<sup>11</sup>, RS, COS, RWR, SR\* on real WikiV, CitH, CaD. For each dataset, we randomly issue 300 queries for  $s(*, q)$  and  $s(*, *)$  via *importance coverage* criterion, and use 3 metrics (NDCG, Kendall, Spearman) to evaluate each method, respectively. Figure 10a shows the average quantitative results. (1) In all cases, SR# exhibits higher semantic quality than the other methods. This is because SR# can avoid “connectivity trait” issue by utilizing a “cosine” kernel *recursively* in a SimRank-like style, whereas COS considers only *direct* overlapped in-neighbors, and JSR and LSR both have a “connectivity trait” problem. (2) In several cases, the NDCG<sub>200</sub> of SR++ (on CitH) and RS (on CaD) may even worse than that of JSR and LSR. This is because, for SR++, its evidence factor will be 0 whenever there are no common *direct* in-neighbors; for RS, automorphism equivalence has priority over connectivity in similarity assessment. Thereby, SR++ and RS may not resolve the “connectivity trait” problem. SR\* is capable of resolving the zero-similarity problem of SimRank. Thus, its NDCG<sub>200</sub> is higher than JSR and LSR. However, SR\* also has the “connectivity trait” problem of SimRank, leading to its retrieval quality lower than SR# and PSR#.

Figure 10b compares the percentage of queries from the 300 queries sample (based on *importance coverage* criterion) that SR#, SR++, JSR, LSR, RS, COS, RWR, SR\* provide similarities for on real WikiV, CitH, CaD, respectively. For each dataset, SR# substantially improves the coverage of JSR/LSR ( $\sim 0.73$ ) and SR++/RS/SR\* ( $\sim 0.81$ ) to  $\sim 0.95$ . This can be considered as expected, since (a) the “connectivity trait” problem of JSR and LSR will downgrade similarities of node-pairs with

<sup>10</sup><http://academic.research.microsoft.com/VisualExplorer>

<sup>11</sup>For semantics evaluation, MSR produces the same similarity values as LSR, since [25] approximates  $D$  by  $(1 - \gamma)I$ .

| #  | SR <sup>#</sup> / PSR <sup>#</sup> | JSR                   | LSR                   | SR <sup>++</sup>    | COS                  | RWR                 | RS              |
|----|------------------------------------|-----------------------|-----------------------|---------------------|----------------------|---------------------|-----------------|
| 1  | Jens Graupmann                     | Sergej Sizov          | Ioannis Aekaterinidis | Sergej Sizov        | Jens Graupmann       | Martin Theobald     | Wolfgang Bock   |
| 2  | Michael Biber                      | Ioannis Aekaterinidis | Sergej Sizov          | Michael Biber       | Michael Biber        | Matthias Bender     | Peter H. Cramer |
| 3  | Patrick Zimmer                     | Michael Biber         | Jens Graupmann        | Patrick Zimmer      | Patrick Zimmer       | Christian Zimmer    | Thomas Licht    |
| 4  | Sergej Sizov                       | Patrick Zimmer        | Michael Biber         | Jens Graupmann      | Matthias Bender      | Sebastian Michel    | Gerd Pttjer     |
| 5  | Matthias Bender                    | Jens Graupmann        | Patrick Zimmer        | Matthias Bender     | Christian Zimmer     | Jens Graupmann      | Bernd Wallat    |
| 6  | Christian Zimmer                   | Matthias Bender       | Christian Zimmer      | Christian Zimmer    | Sergej Sizov         | Ralf Schenkel       | Peter Weging    |
| 7  | Dimitrios Mavroeidis               | Christian Zimmer      | Matthias Bender       | Stefan Siersdorfer  | Peter Triantafillou  | Peter Triantafillou | Michael Weigt   |
| 8  | George Tsatsaronis                 | Stefan Siersdorfer    | Stefan Siersdorfer    | Sebastian Michel    | Martin Theobald      | Surajit Chaudhuri   | Martin R. Frank |
| 9  | Peter Triantafillou                | Sebastian Michel      | Sebastian Michel      | Peter Triantafillou | Dimitrios Mavroeidis | Michael Biber       | Xiaofeng Meng   |
| 10 | Stefan Siersdorfer                 | Peter Triantafillou   | Peter Triantafillou   | Ralf Schenkel       | George Tsatsaronis   | Patrick Zimmer      | Tok Wang Ling   |

(a) Top-10 Similar Authors w.r.t. Query = “Gerhard Weikum”

| #   | SR <sup>#</sup> / PSR <sup>#</sup> | JSR / LSR       | SR <sup>++</sup> | COS                | RWR                 | RS                       |
|-----|------------------------------------|-----------------|------------------|--------------------|---------------------|--------------------------|
| 1   | Nitin Thaper                       | SungRan Cho     | SungRan Cho      | Amit Marathe       | Nick Koudas         | Xiaofeng Meng            |
| 2   | Piotr Indyk                        | Amit Marathe    | Amit Marathe     | Nitin Thaper       | S. Muthukrishnan    | Shahram Ghandeharizadeh  |
| 3   | Amit Marathe                       | Nitin Thaper    | Nitin Thaper     | Piotr Indyk        | Sudipto Guha        | Dimitrios Georgakopoulos |
| 4   | Sudipto Guha                       | Piotr Indyk     | Piotr Indyk      | Sudipto Guha       | Graham Cormode      | Michael Stonebraker      |
| 5   | SungRan Cho                        | P. G. Ipeirotis | P. G. Ipeirotis  | SungRan Cho        | Flip Korn           | Mikal Ziane              |
| 6   | Ting Yu                            | David Toman     | David Toman      | Ting Yu            | H. V. Jagadish      | Achim Pick               |
| 7   | David Toman                        | Luis Gravano    | Luis Gravano     | P. G. Ipeirotis    | Siheh Amer-Yahia    | Alexandra Poulouvasilis  |
| 8   | P. G. Ipeirotis                    | Irina Rozenbaum | Irina Rozenbaum  | David Toman        | Ting Yu             | Martin R. Frank          |
| 9   | Luis Gravano                       | Yunyue Zhu      | Ting Yu          | Rui Zhang          | Y. Papakonstantinou | Jeanette P. Schmidt      |
| 10  | Tianqiu Wang                       | Ting Yu         | Yunyue Zhu       | Luis Gravano       | Amit Marathe        | Charles T. Melson        |
| ... | ...                                | ...             | ...              | ...                | ...                 | ...                      |
| 13  | Michael Rabinovich                 | Sudipto Guha    | Sudipto Guha     | Andrey Balmin      | Theodore Johnson    | Mary Ann Walker          |
| ... | ...                                | ...             | ...              | ...                | ...                 | ...                      |
| 39  | N. Wiwatwattana                    | Michail Vlachos | Xiaosong Ma      | Monica Scannapieco | Nitin Thaper        | Rory Patterson           |

(b) Top-10 Similar Authors w.r.t. Query = “Divesh Srivastava”

Fig. 11. Qualitative Comparison: Retrieval Quality on DBLP Coauthorship Dataset

high connectivity, and (b) SR<sup>++</sup> can only partially fix the “connectivity trait” issue within 1-hop in-neighborhood. Note that SR<sup>#</sup> also has the “connectivity trait” problem of SimRank, but its coverage is not as low as JSR/LSR. This is because SR<sup>#</sup> fixes the zero-similarity problem of JSR/LSR, thereby enhancing its retrieval quality to  $\sim 0.81$ .

Figure 10c depicts the percentage of queries with overlapped multi-hop in-neighbors from the 300 queries sample (via *overlapping coverage* criterion) that SR<sup>#</sup>, SR<sup>++</sup>, JSR, LSR, RS, COS, RWR, SR<sup>\*</sup> are able to identify on real WikiV, CitH, CaD, respectively. (1) For each dataset, SR<sup>#</sup> significantly achieves  $\sim 0.95$  coverage of common multi-hop in-neighbors, much superior to JSR/LSR ( $\sim 0.20$ ), SR<sup>++</sup> ( $\sim 0.51$ ), COS ( $\sim 0.41$ ), and SR<sup>\*</sup> ( $\sim 0.6$ ). The reason is that our “cosine” kernel provides an appropriate normalization factor  $\| * \|_2$  that can recursively fix the “connectivity trait” problem. In contrast, the  $\| * \|_1$  normalization factor of JSR/LSR excessively squeezes the range of similarity  $[0, 1]$ . (2) Under *overlapping coverage* criterion, COS ( $\sim 0.41$ ) outperforms JSR/LSR ( $\sim 0.20$ ) since COS is not limited by the “connectivity trait” problem.

To further evaluate the search depth of SR<sup>#</sup>, SR<sup>++</sup>, JSR, LSR, RS, COS, RWR, SR<sup>\*</sup>, we first apply *overlapping coverage* criterion to generate 2,000 queries, and then generate 300 queries via *importance coverage* criterion, and classify them into 4 groups, e.g., “4-5” collects queries (*a, b*) whose path length between *a* and *b* is 8–10. Figure 10d depicts the search depth of all the methods on WikiV. (1) For each group, SR<sup>++</sup> and COS have the lowest quality of depth search among all the methods, since they cannot capture the paths of length  $> 2$  between nodes. (2) SR<sup>#</sup> achieves the highest quality, and its superiority is more pronounced in the groups that have longer paths. This tells us that the “connectivity trait” issue has a large influence on node-pairs with long paths.

**7.2.2 Qualitative Case Study: Retrieval Quality on DBLP.** Figure 11 illustrates the retrieval quality for finding the top-K most similar coauthors with respect to any given query on DBLP collaboration graph (CaD). We use different similarity models for ranking. Due to space limitations, we just detail



the qualitative results on top-15 coauthors with respect to the query (Gerhard Weikum, and Divesh Srivastava), respectively.

When the query is “Gerhard Weikum”, the qualitative ranking results of  $SR^\#$ ,  $PSR^\#$ ,  $SR^{++}$ , JSR, LSR, RS, COS, RWR are shown in Figure 11a. We discern that (1) for top-10 rankings,  $SR^\#$  and  $PSR^\#$  produce the same results, implying the accuracy loss of our Krylov subspace-based method  $PSR^\#$  for dimensionality reduction is negligibly small for top-K rankings as it compromises only a little (or no) accuracy for achieving substantial speedups. (2) Moreover, the most similar coauthors of Gerhard Weikum are effectively captured by  $SR^\#$  and  $PSR^\#$ , highlighting the better search quality of these ranking models. For instance, Jens Graupmann is ranked 1st by  $SR^\#$  and  $PSR^\#$ , which is consistent with the fact that Jens has many common coauthors with Gerhard as they had close collaborations on the development of the SphereSearch Engine at the Max Planck Institute for Informatics in 2005. (3) Jens ranks lower in JSR and LSR, since both Jeh’s and Li *et al.*’s SimRank models adopt an unreasonable weight factor in the denominator for normalising the common coauthors in the nominator, leading to the “connectivity trait” problems for similarity ranking. (4) Jehs ranks slightly higher in  $SR^{++}$  than in JSR, indicating that  $SR^{++}$  partially alleviates the “connectivity trait” problem by introducing an evidence factor in the one-hop neighbourhood. In comparison,  $SR^\#$  and  $PSR^\#$  can well resolve the “connectivity trait” problem of SimRank by refining the weight factor for the neighbourhood of multiple hops, thereby exhibiting more reliable search quality than  $SR^{++}$ . (5) Sergej ranks lower than Martin Theobald by RWR since Sergej had fewer papers co-authored with Gerhard. However, there are many multi-hop common co-authored papers between Sergej and Gerhard, which can be effectively evaluated by  $SR^\#$ . (6) Sergej ranks lower in COS than in  $SR^\#$ , LSR, and JSR as COS captures only one-hop common neighbouring information between Sergej and Gerhard. (7) RS is based on automorphically structural equivalence of nodes for similarity assessment rather than connectivity of paths between nodes. Therefore, the top-K ranking results of RS differ from all other ranking models, which reflects the “role-based” similarity information.

When the query is “Divesh Srivastava”, the results are illustrated in Figure 11b. We notice that (1) Nitin Thaper and Divesh have many one-hop and multi-hop coauthors in common, which can be well taken into consideration by  $SR^\#$ , highlighting its better search quality for ranking. (2) The rankings of Nitin in JSR and LSR are lower than in  $SR^\#$ , due to the “connectivity trait” problems of JSR and LSR in normalising the number of common coauthors. (3) Nitin ranks higher in COS but lower in RWR. This is because Nitin and Divesh have more one-hop common coauthors but fewer coauthored papers. (4) For the query “Divesh”, the top-30 rankings of JSR and LSR are the same, indicating that Kusumoto *et al.*’s strategy of approximating  $D$  with  $(1 - \gamma)I$  for SimRank, in a few cases, may get good approximation results for the purpose of top-K rankings. However, this is not always the case. For example, when the query is “Gerhard”, even the top-10 rankings of JSR and LSR are different. In this case, our varied- $D$  model proposed in Section 2.3 provides an accurate way of evaluating Jeh and Widom’s SimRank model, which makes JSR scalable on any sizable graphs. (5) The ranking of  $SR^{++}$  is almost the same as LSR except a slightly ranking difference in the 9th and 10th places, due to the evidence factor used by  $SR^{++}$  that can in part alleviate the “connectivity trait” problem of SimRank. However, the effect of evidence factor on ranking quality improvement is not as obvious as  $SR^\#$  because the evidence factor of  $SR^{++}$  is only effective for one-hop common neighbours.

**7.2.3 Effects of  $m$  on Retrieval Quality of  $PSR^\#$ .** Figure 12 depicts the accuracy of  $PSR^\#$  under different low dimensionality  $m \in \{5, 10, 15\}$  on six real datasets, as compared with  $SR^\#$ . We fix the number of iterations  $k = 40$  and randomly pick up  $(|A|, |B|) = (100, 50)$  queries for small and medium datasets (CitH, WebN, ComY, SocL), and  $(|A|, |B|) = (100, 10)$  queries for large datasets (UK02 and

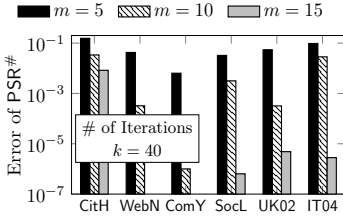


Fig. 12. Varying  $m$  for  $\text{PSR}^\#$  on Six Real Datasets

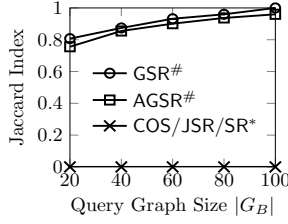


Fig. 13. Varying  $|G_B|$  for  $\text{GSR}^\#$  and  $\text{AGSR}^\#$  on CaD

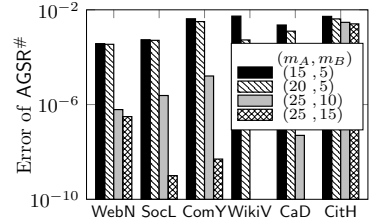


Fig. 14. Varying  $(m_A, m_B)$  for  $\text{AGSR}^\#$  on Six Real Datasets

IT04). The error of  $\text{PSR}^\#$  is measured by the maximum difference of similarities between  $\text{PSR}^\#$  at low dimensionality  $m$  and  $\text{SR}^\#$  over the query sets  $(A, B)$ . From the results, we can discern that (1) on each dataset, the error of  $\text{PSR}^\#$  decreases when the dimensionality  $m$  is growing. Particularly, on WebN and ComY, when  $m$  increases to 15, the error of  $\text{PSR}^\#$  drops to 0, which implies that  $\text{PSR}^\#$  can produce the exact  $\text{SR}^\#$  similarity scores without any loss of accuracy by resorting to only a 15-dimensional Krylov subspace, highlighting the effectiveness of our dimensionality reduction techniques. (2) Moreover, when  $m$  increases from 5 to 15, we see that the errors of  $\text{PSR}^\#$  reduced on medium and large datasets (e.g., ComY, UK02) are more apparent than those on small datasets (CitH). Hence, on large datasets, the accuracy of  $\text{PSR}^\#$  is more sensitive to  $m$ .

**7.2.4 Effects of  $|G_B|$  on Retrieval Quality of  $\text{GSR}^\#$  and  $\text{AGSR}^\#$ .** Next, we are going to evaluate the search quality of (1) our generalised “cosine-based” SimRank similarity algorithm  $\text{GSR}^\#$  across two graphs  $G_A$  and  $G_B$ , and (2) its accelerative version  $\text{AGSR}^\#$  which is based on our Arnoldi dimensionality reduction techniques. We regard the graph in each real dataset as  $G_A$ , and randomly sample a subgraph from  $G_A$ , denoted as  $G_B$ . To obtain good (unbiased) sample graph  $G_B$ , our sampling strategy follows the “forest fire” evolutionary approach [28]. We pick a seed node at random, and start “burning” its outgoing edges and the corresponding nodes. If an edge gets burned, the node at the other endpoint has a chance to burn its own edges, and so on recursively till the size of each sample graph meets our expectations.

Figure 13 compares the semantic accuracy of our generalised “cosine-based” SimRank similarity algorithms ( $\text{GSR}^\#$  and  $\text{AGSR}^\#$ ) with that of other models (e.g., COS, JSR,  $\text{SR}^\#$ ) for assessing pairwise similarities across two graphs  $G_A$  and  $G_B$ . For each graph  $G_A$  in the real datasets, we sample several graphs for  $G_B$ , with their vertex sizes ranging varying from 20 to 100. The results on CaD are reported in Figure 13. Note that the results on other real datasets are similar and omitted here. To measure the semantic accuracy among the different algorithms, we regard the node set  $V_B$  of graph  $G_B$  as the ground truth, and expect to find the node subset  $P(\mathcal{J})$  in graph  $G_A$  that best pairs all nodes in  $V_B$  through the scoring results of each search algorithm  $\mathcal{J}$ . If the retrieved set  $P(\mathcal{J})$  is close to the ground truth  $V_B$ , it indicates that the algorithm  $\mathcal{J}$  has good semantics in producing reliable similarity scores for pairing nodes between  $G_A$  and  $G_B$ . To be specific, we run  $\text{GSR}^\#$  and  $\text{AGSR}^\#$  to evaluate the pairs of similarities  $S_{A,B} := \{S_{a,b}\}_{a \in G_A \text{ and } b \in G_B}$  across graphs  $G_A$  and  $G_B$ , and perform other algorithms (COS, JSR,  $\text{SR}^\#$ ) to assess  $S_{A,B}$  between two node sets  $A$  and  $B$  in the union graph  $(G_A \cup G_B)$ . Based on the similarity results  $S_{A,B}(\mathcal{J})$  from each algorithm  $\mathcal{J}$ , we first construct a complete bipartite graph  $B(\mathcal{J}) = (V_A \cup V_B, E)$  with nodes of  $G_A$  (denoted as  $V_A$ ) on one side and nodes of  $G_B$  (denoted as  $V_B$ ) on the other side. Each edge  $(a, b)$  in the bipartite graph  $B(\mathcal{J})$  carries a weight value of  $S_{a,b}(\mathcal{J})$ , which is the similarity score  $S_{a,b}$  returned by algorithm  $\mathcal{J}$ . We then apply the Hungarian algorithm on  $B(\mathcal{J})$  to find a matching of maximum weight in  $B(\mathcal{J})$ .

that pairs each node in  $G_B$  with a node in  $G_A$ , which produces a node subset of  $V_A$ , denoted as

$$P(\mathcal{J}) := \{x \in V_A \mid x \text{ is paired with node in } V_B \text{ via the maximum weight matching of } B(\mathcal{J})\}$$

We finally use Jaccard index, defined by  $\text{Jaccard}(P(\mathcal{J}), V_B) := \frac{|P(\mathcal{J}) \cap V_B|}{|P(\mathcal{J}) \cup V_B|}$ , to measure the closeness between the set  $P(\mathcal{J})$  and the ground truth  $V_B$ . The higher Jaccard similarity, the larger overlap between  $P(\mathcal{J})$  and  $V_B$ , meaning that  $S_{A,B}(\mathcal{J})$  returned by algorithm  $\mathcal{J}$  is more reliable since the maximum weight matching over  $S_{A,B}(\mathcal{J})$  can better pair the nodes between  $G_A$  and  $G_B$ .

From the results in Figure 13, we discern that (1) with the increasing size of the query graph  $G_B$  from 20 to 100, the Jaccard indices of  $\text{GSR}^\#$  and  $\text{AGSR}^\#$  also increase and are stable between 0.8 and 1, being consistently much higher than the other competitors. This indicates that our generalised “cosine-based” SimRank similarity model yields semantically meaningful scoring results across two graphs. (2) The semantic accuracy of  $\text{GSR}^\#$  is slightly higher than that of  $\text{AGSR}^\#$ . This is because  $\text{AGSR}^\#$  is a dimensionality reduction algorithm which would sacrifice a little accuracy for fast speedup. (3) The other similarity search algorithms (COS, JSR,  $\text{SR}^*$ ) always produce zero Jaccard scores regardless of the size of the query graph  $G_B$ , implying that the similarity scores produced by these algorithms cannot find the best matching to better pair the pairwise nodes across two graphs. These undesirable semantic results are due to the fact that COS, JSR,  $\text{SR}^*$  always produce zero similarity values between two nodes if they belong to two different disconnected components.

**7.2.5 Effects of  $(m_A, m_B)$  on Retrieval Quality of  $\text{AGSR}^\#$ .** Figure 14 depicts the maximum error of the similarity values computed by  $\text{AGSR}^\#$  relative to those evaluated by  $\text{GSR}^\#$  over all the query pairs on every real dataset. Varying low dimensionality  $m_A$  for graph  $G_A$  from 15 to 25 and  $m_B$  for graph  $G_B$  from 5 to 15, we observe that (1) for all  $m_A$  and  $m_B$ , the errors of  $\text{AGSR}^\#$  are consistently below 0.01, being acceptably small in practice. This indicates our Arnoldi-based dimensionality reduction techniques, when applied to  $\text{GSR}^\#$ , compromise only a little accuracy for a dramatic increase (up to over one order of magnitude) in the speed of similarity retrieval. (2) On each dataset, a mild increase in  $(m_A, m_B)$  may also result in orders-of-magnitude decrease in the error of  $\text{AGSR}^\#$ .

**7.2.6 Effects of  $k$  on Retrieval Quality of  $\text{PSR}^\#$ .** Figure 15 analyses the sensitivity of retrieval quality relevant to  $k$  for ranking top-100 similar authors *w.r.t.* a given query on DBLP (CaD). Due to space limitations, we only report the results *w.r.t.* two queries (“Gerhard Weikum” and “Divesh Srivastava”) since tendencies for other queries and datasets are similar. For each query, we vary  $k$  from 5 to 20 for  $\text{PSR}^\#$ , and compare their top-100 similarity rankings with those using the baseline  $\text{SR}^\#$ . To ensure the similarity scores of  $\text{SR}^\#$  to be accurate enough as a baseline, we set the maximum number of iterations for  $\text{SR}^\#$  to 100 so that  $\text{SR}^\#$  values can achieve a high precision of  $\gamma^{100+1} = 0.6^{100+1} < 10^{-22}$ . In the ranking tables, the authors highlighted in red mean that their ranking positions in  $\text{PSR}^\#$  for a given  $k$  are perturbed from the baseline  $\text{SR}^\#$ , where the numbers in the bracket of each red cell represent the relative deviation of the ranking positions of  $\text{PSR}^\#$  with different  $k$  *w.r.t.*  $\text{SR}^\#$ . We notice that (1) for query “Gerhard Weikum” (*resp.* “Divesh Srivastava”), when  $k$  increases from 5 to 20, the top-30 (*resp.* top-15) rankings of  $\text{PSR}^\#$  are exactly the same as the baseline  $\text{SR}^\#$ . This indicates that  $k$  is insensitive to the quality of top-K ranking particularly when top-K is very small (*e.g.*, top-15). (2) However, for the rankings below 30, the deviation in  $\text{PSR}^\#$  rankings is more sensitive to  $k$ . Generally, the smaller  $k$ , the more ranking positions deviated in  $\text{PSR}^\#$ . For example, for query “Gerhard Weikum” in Figure 15a, in  $\text{PSR}^\#$ , there are more than 15 red cells for  $k = 5$ , 4 for  $k = 10$ , and only 2 for  $k = 20$ , respectively. This is consistent with our intuition since a small number of iterations  $k$  achieve less accurate scores for  $\text{PSR}^\#$ , thereby leading to more perturbations in the ranking quality. Similar trends hold for query “Divesh Srivastava” in Figure 15b.

The choice of  $k$  often depends on the dataset, our requirements for accuracy and computational time, and the size of top-K ranking. For example, from our DBLP experiments in Figure 15, if we

| #   | SR <sup>#</sup>   | PSR <sup>#</sup>        |                         |                         |
|-----|-------------------|-------------------------|-------------------------|-------------------------|
|     |                   | k = 5                   | k = 10                  | k = 20                  |
| 1   | Nitin Thaper      | Nitin Thaper            | Nitin Thaper            | Nitin Thaper            |
| 2   | Piotr Indyk       | Piotr Indyk             | Piotr Indyk             | Piotr Indyk             |
| ... | ...               | ...                     | ...                     | ...                     |
| 30  | H. V. Jagadish    | H. V. Jagadish          | H. V. Jagadish          | H. V. Jagadish          |
| 31  | Xiaosong Ma       | Jungchul Woo [↑ 1]      | Xiaosong Ma             | Xiaosong Ma             |
| 32  | Jungchul Woo      | Chulyun Kim [↑ 1]       | Jungchul Woo            | Jungchul Woo            |
| 33  | Chulyun Kim       | Xiaosong Ma [↓ 2]       | Chulyun Kim             | Chulyun Kim             |
| 34  | Theodore Johnson  | Theodore Johnson        | Theodore Johnson        | Theodore Johnson        |
| ... | ...               | ...                     | ...                     | ...                     |
| 43  | V. Shkapenyuk     | V. Shkapenyuk           | V. Shkapenyuk           | V. Shkapenyuk           |
| 44  | Fang Du           | A. K. H. Tung [↑ 6]     | Fang Du                 | Fang Du                 |
| 45  | E. D. Lazowska    | Tamraparni Dasu [↑ 2]   | Stelios Paparizos [↑ 1] | Stelios Paparizos [↑ 1] |
| 46  | Stelios Paparizos | Stelios Paparizos       | E. D. Lazowska [↓ 1]    | E. D. Lazowska [↓ 1]    |
| 47  | Tamraparni Dasu   | Fang Du [↓ 3]           | Tamraparni Dasu         | Tamraparni Dasu         |
| 48  | Cinda Heeren      | E. D. Lazowska [↓ 3]    | Cinda Heeren            | Cinda Heeren            |
| 49  | Leonard Pitt      | Cinda Heeren [↓ 1]      | Leonard Pitt            | Leonard Pitt            |
| 50  | A. K. H. Tung     | Leonard Pitt [↓ 1]      | A. K. H. Tung           | A. K. H. Tung           |
| 51  | Alin Deutsch      | Alin Deutsch            | Alin Deutsch            | Alin Deutsch            |
| ... | ...               | ...                     | ...                     | ...                     |
| 111 | Shyh-Kwei Chen    | Shyh-Kwei Chen          | Shyh-Kwei Chen          | Shyh-Kwei Chen          |
| 112 | S. Seshadri       | Elaine Qing Chang [↑ 5] | S. Seshadri             | S. Seshadri             |
| 113 | Srinivas Ashwin   | Terence Ho [↑ 5]        | Srinivas Ashwin         | Srinivas Ashwin         |
| 114 | Varun Kacholia    | Beng Chin Ooi [↑ 13]    | Varun Kacholia          | Varun Kacholia          |
| 115 | Rushi Desai       | S. Seshadri [↓ 3]       | H. Karambelkar [↑ 1]    | Rushi Desai             |
| 116 | H. Karambelkar    | Srinivas Ashwin [↓ 3]   | Rushi Desai [↓ 1]       | H. Karambelkar          |
| 117 | Elaine Qing Chang | R. Ramakrishnan [↑ 7]   | Elaine Qing Chang       | Elaine Qing Chang       |

(a) Query = “Gerhard Weikum”

| #   | SR <sup>#</sup>     | PSR <sup>#</sup>          |                           |                         |
|-----|---------------------|---------------------------|---------------------------|-------------------------|
|     |                     | k = 5                     | k = 10                    | k = 20                  |
| 1   | Jens Graupmann      | Jens Graupmann            | Jens Graupmann            | Jens Graupmann          |
| 2   | Michael Biber       | Michael Biber             | Michael Biber             | Michael Biber           |
| ... | ...                 | ...                       | ...                       | ...                     |
| 16  | A. P. Buchmann      | A. P. Buchmann            | A. P. Buchmann            | A. P. Buchmann          |
| 17  | Andreas Thor        | Daniel A. Keim [↑ 1]      | Andreas Thor              | Andreas Thor            |
| 18  | Daniel A. Keim      | Andreas Thor [↓ 1]        | Daniel A. Keim            | Daniel A. Keim          |
| 19  | Michael Gillmann    | Gautam Das [↑ 5]          | Michael Gillmann          | Michael Gillmann        |
| 20  | Wolfgang Wonner     | Peter Muth [↑ 5]          | Wolfgang Wonner           | Wolfgang Wonner         |
| 21  | Harald Schning      | Achim Pick [↑ 5]          | Harald Schning            | Harald Schning          |
| 22  | Peter Zaback        | Michael Gillmann [↓ 3]    | Peter Zaback              | Peter Zaback            |
| 23  | Erhard Rahm         | Wolfgang Wonner [↓ 3]     | Gautam Das [↑ 1]          | Gautam Das [↑ 1]        |
| 24  | Gautam Das          | Harald Schning [↓ 3]      | Erhard Rahm [↓ 1]         | Erhard Rahm [↓ 1]       |
| 25  | Peter Muth          | Peter Zaback [↓ 3]        | Martin L. Kersten [↑ 2]   | Martin L. Kersten [↑ 2] |
| 26  | Achim Pick          | Erhard Rahm [↓ 3]         | Peter Muth [↓ 1]          | Peter Muth [↓ 1]        |
| 27  | Martin L. Kersten   | Martin L. Kersten         | Achim Pick [↓ 1]          | Achim Pick [↓ 1]        |
| 28  | Patrick Valduriez   | Patrick Valduriez         | Patrick Valduriez         | Patrick Valduriez       |
| ... | ...                 | ...                       | ...                       | ...                     |
| 88  | Utkarsh Srivastava  | Prasanna Ganesan [↑ 3]    | Utkarsh Srivastava        | Utkarsh Srivastava      |
| 89  | Bin Chen            | Panagiotis Karras [↓ 21]  | Bin Chen                  | Bin Chen                |
| 90  | Kyuseok Shim        | Shawn R. Jeffery [↑ 2]    | Kyuseok Shim              | Kyuseok Shim            |
| 91  | Prasanna Ganesan    | Hans-Peter Kriegel [↓ 11] | Y. Papakonstantinou [↑ 2] | Prasanna Ganesan        |
| 92  | Shawn R. Jeffery    | Bryan Smith [↓ 7]         | Prasanna Ganesan [↓ 1]    | Peter A. Boncz [↑ 3]    |
| 93  | Y. Papakonstantinou | Meikel Pss [↓ 6]          | Peter A. Boncz [↑ 2]      | Y. Papakonstantinou     |

(b) Query = “Divesh Srivastava”

Fig. 15. Sensitivity Analysis of Retrieval Quality w.r.t.  $k$  on DBLP Dataset

want to retrieve top-30 most similar authors, we can choose  $k = 10$  because (1) when  $k$  increases from 5 to 10, the accuracy of PSR<sup>#</sup> is significantly increased. (2) However, when  $k$  continues to grow from 10 to 20, there is only a slight improvement in PSR<sup>#</sup> accuracy (e.g., for query “Gerhard

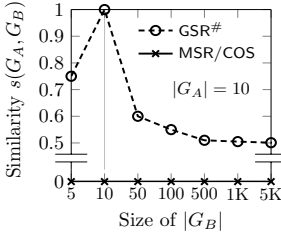


Fig. 16. Similarity of GSR# on Graphs with Different Scales

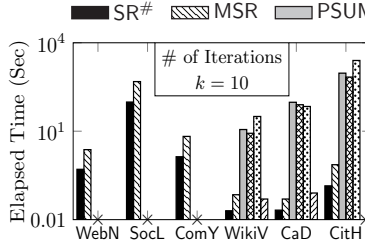


Fig. 17. Time for Single-Source Similarity Search on Real Datasets

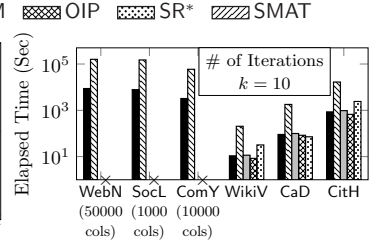


Fig. 18. Time for Partial-Pairs & All-Pairs Search (WikiV, CaD, CitH)

Weikum”, there are 4 red cells for  $k = 10$  and 2 for  $k = 20$ ), but more computational time is required as the number of iterations  $k$  is doubled.

**7.2.7 Case Study of GSR# on Two Graphs with Different Scales.** We next provide a case study, showing the effectiveness of GSR# for dealing with the situation where “two nodes may not share a similar local structure, but GSR# is capable of assessing them as similar”. Formally, let us experimentally consider the following case: “Two fully connected graphs  $G_A$  and  $G_B$  are of different scales, but GSR# is able to effectively capture their similarity.”

We fix the number of nodes in  $G_A$  with  $|V_A| = 10$ , and vary the number of nodes in  $G_B$  with  $|V_B|$  ranging from 5 to 5K. Since  $G_A$  and  $G_B$  are fully connected graphs, the generalised “cosine-based” SimRank matrix between  $G_A$  and  $G_B$ , denoted as  $S(G_A, G_B)$ , is of size  $|V_A| \times |V_B|$  whose elements are of the same similarity scores. Thus, Figure 16 reports the same value of  $S(G_A, G_B)$  for every pair of graphs  $(G_A, G_B)$ , where  $x$ -axis denotes the node size of  $G_B$ , and  $y$ -axis denotes the similarity value of  $S(G_A, G_B)$ . From the results, we see that GSR# can effectively capture the similarity between two graphs of different scales, as opposed to the existing SimRank that assesses them as totally dissimilar. This is because GSR# introduces the in- and out-linkage matrices as the seed similarity scores, and propagates them iteratively through our “cosine-based” SimRank model. In contrast, SimRank only relies on the local connectivity structure of two nodes. Moreover, for the fixed  $|V_A| = 10$ , when  $|V_B|$  varies from 10 to 5K, the similarity score  $S(G_A, G_B)$  decreases from 1 to 0.5; When  $|V_B|$  varies from 5 to 10, the similarity score  $S(G_A, G_B)$  increases from 0.75 to 1. This is consistent with our basic intuition - the more noticeable the difference in scales between  $G_A$  and  $G_B$ , the less similarity scores they will have. The highest similarity is reached when  $G_A$  and  $G_B$  are of the same size (i.e.,  $|V_A| = |V_B| = 10$ ).

### 7.3 Evaluation on Time Efficiency

**7.3.1 Computational Time of SR#.** Figure 17 illustrates the running time of SR#, MSR, PSUM, OIP, SR\*, SMAT for single-source  $s(*, q)$  on 6 real datasets. (1) In all cases, SR# always substantially outperforms the other methods. This is because SR# can eliminate duplicate computations for maximal sharing, whereas MSR computes each term separately. (2) PSUM, OIP, SR\*, SMAT will crash on large WebN, SocL, ComY, due to the memory allocation. On WikiV and CaD, they are 3-4 orders of magnitude slower than SR#, since their iterative models to compute  $s(*, q)$  rely on all-pairs outputs of the previous iteration.

Figure 18 shows the time of SR#, MSR, PSUM, OIP, SR\* for all-pairs  $s(*, *)$  on 6 real datasets. (1) Only SR# and MSR survive on all datasets, whereas PSUM, OIP, SR\* fail on large WebN, SocL, ComY, due to the memory allocation. Notice that, for WebN, SocL, ComY, we report the time for 50K, 1K, 10K columns (randomly chosen from the entire similarity matrix), respectively, though SR#, MSR can potentially run for weeks. (2) MSR is slower than others as it sacrifices speed for

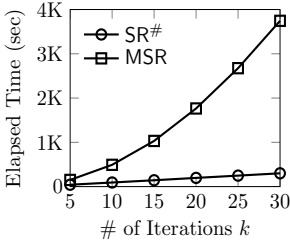


Fig. 19. Time vs. # of Iterations  $k$  on SocL

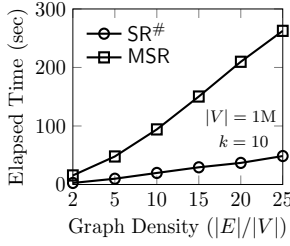


Fig. 20. Time vs. Graph Density  $|E|/|V|$  on SYN

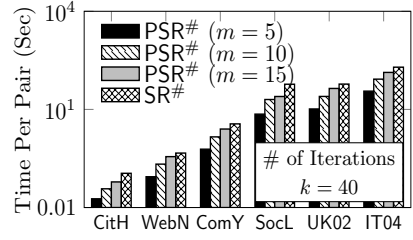


Fig. 21. Average Time Per Node-Pair for PSR# and SR# on Real Datasets

scalability. In contrast, SR# not only scales well on large graphs, but also has comparable speed to those of PSUM, OIP, SR\*.

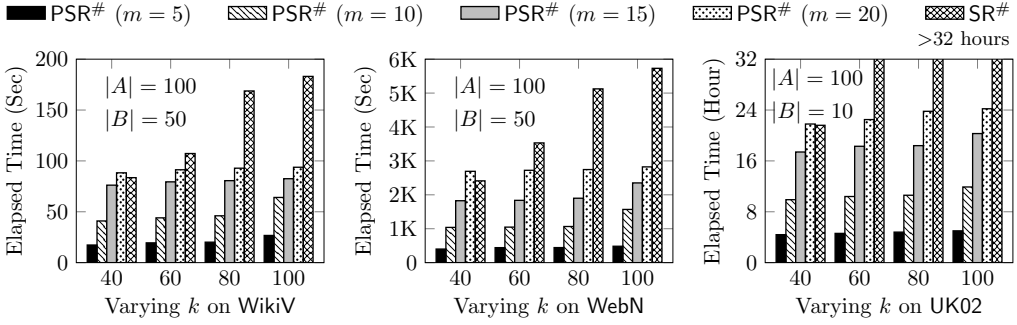
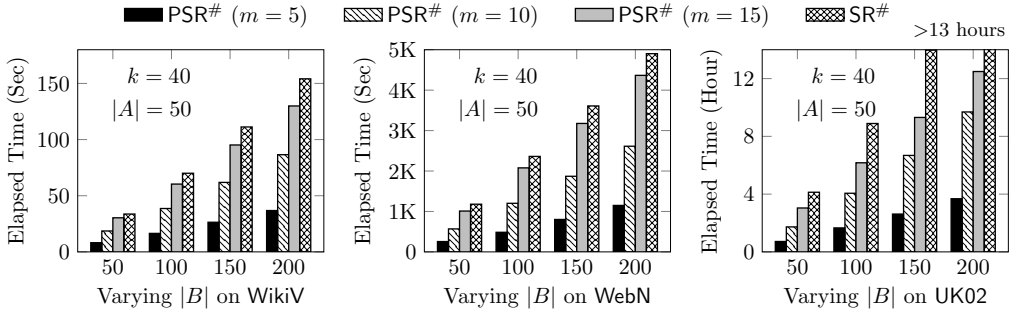
Figure 19 presents the impact of the number of iterations  $k$  on the time of SR# and MSR. When  $k$  grows from 5 to 30, the time of SR# does not increase significantly (just from 42s to 301s), as opposed to the time of MSR growing from 152s to 3744s. The reason is that MSR contains many duplicate computations among different iterations, whereas SR# can merge these results after rearranging the computation order. It is consistent with our analysis in Subsection 2.4.

Figure 20 demonstrates the impact of network density on the computational time of SR# and MSR on synthetic data. Fixing  $|V| = 1,000,000$  and  $k = 10$ , we generate a synthetic dataset by increasing the graph density from 2 to 25. (1) When the density increases, the time of both algorithms will increase. (2) For dense graphs, the speedup for SR# is significantly higher than MSR, due to the number of iterations with a huge influence on MSR compared with SR#. This is in agreement with the complexity of SR# and MSR.

**7.3.2 Computational Time of PSR#.** To evaluate the high efficiency and scalability of our enhanced version PSR# for partial-pairs similarity assessment on large-scale graphs, we conduct the following additional experiments, and adopt more sizable real datasets (e.g., UK02 and IT04) with up to billions of edges. Since our experiments in Subsection 7.3.1 have demonstrated that SR# outperforms the existing competitors (SR++, JSR, LSR, RS, COS, RWR, SR\*), we only compare PSR# with SR# (the best-known competitor) in our additional experiments.

Figure 21 compares the computational time of SR# and PSR# on 6 real datasets. For each medium (resp. large) dataset, we randomly choose  $(|A|, |B|) = (100, 50)$  (resp.  $(|A|, |B|) = (100, 10)$ ) pairs as queries for partial-pairs similarity search. On each dataset, fixing the number of iterations  $k = 40$ , we vary low dimensionality  $m \in \{5, 10, 15\}$  for PSR#, and compare their average CPU time required for retrieving the similarity of each pair of nodes. Our results show that (1) on each dataset, for any low dimensionality  $m \in \{5, 10, 15\}$ , PSR# is consistently 2–8 times faster than SR#. For instance, PSR# ( $m = 5$ ) is 8.2 times faster than SR# on SocL. This is because PSR# constructs a low-order Krylov subspace ( $2m \times 2m$ , with  $m \ll n$ ) for efficient “cosine-based” SimRank search through Arnoldi iterations, which substantially reduces the search time, as opposed to SR# that performs all the iterations in the original space ( $n \times n$ ). (2) On every dataset, when  $m$  increases from 5 to 15, the running time of PSR# also grows, due to the increasing size of the Krylov subspace. This is well consistent with our time complexity analysis of Algorithm 3 in Section 4.3.

Figure 22 shows the impact of low dimensionality ( $m$ ) and the number of iterations ( $k$ ) on the computational time of PSR# over real datasets, as compared with the time of SR#. Fixing the size of the query sets  $(|A|, |B|)$  on each dataset, we vary the number of iterations  $k$  from 40 to 100. Due to similar trends, we report the results only on three datasets (WikiV, WebN, and UK02). It can be discerned that (1) on each dataset, with the growing number of iterations  $k$ , the computational


 Fig. 22. Effects of Iteration Number  $k$  and Low Order  $m$  for  $\text{PSR}^\#$  on CPU Time over Real-life Datasets

 Fig. 23. Effects of Query Size  $|A| \times |B|$  and Low Order  $m$  for  $\text{PSR}^\#$  on CPU Time over Real-life Datasets

time of  $\text{PSR}^\#$  for any fixed  $m$  increases quite gently, and remains almost stable when  $m$  is small, whereas the time of  $\text{SR}^\#$  increases rapidly in particular when  $k$  becomes large. This is because the iterations performed by  $\text{PSR}^\#$  are in the small Krylov subspace, whose low dimensionality ( $m$ ) is far less than the number of nodes ( $n$ ) of the graph. As a result, the time for performing the iterations on the small Krylov subspace ( $2m \times 2m$ ) is much smaller than the time for iteratively generating the Krylov space on the original space ( $n \times n$ ), leading to insensitivity of  $k$  to the computational time of  $\text{PSR}^\#$ . In comparison, the time of  $\text{SR}^\#$  is more sensitive to  $k$  because more iterations performed by  $\text{SR}^\#$  are on the large original space ( $n \times n$ ). (2) On each dataset, given the number of iterations  $k$ ,  $\text{PSR}^\#$  becomes slower with the growth of dimensionality  $m$ , but for any  $m \in \{5, 10, 15, 20\}$ ,  $\text{PSR}^\#$  still runs much faster than  $\text{SR}^\#$ . The larger  $k$ , the bigger difference of time between  $\text{PSR}^\#$  and  $\text{SR}^\#$ . This conforms to our time complexity analysis of Algorithm 3 in Section 4.3. (3) With the increasing size of the datasets, the superiority of  $\text{PSR}^\#$  over  $\text{SR}^\#$  becomes more noticeable. For instance, when  $k = 60$ , on large UK02, the gap of the running time between  $\text{PSR}^\# (m = 5)$  and  $\text{SR}^\#$  is far more than 27.4 hours because  $\text{PSR}^\# (m = 5)$  only spends 4.7 hours, whereas  $\text{SR}^\#$  does not stop running after 32 hours. In comparison, on small WikiV, the time difference between  $\text{PSR}^\# (m = 5)$  and  $\text{SR}^\#$  is dramatically decreased to only 1.4 minutes. This highlights the scalability of  $\text{PSR}^\#$  on the growing number of iterations  $k$  over massive graphs. (4) When  $m = 20$  and  $k = 40$ , we can see from each dataset that  $\text{PSR}^\#$  is even slower than  $\text{SR}^\#$  as expected because  $\text{PSR}^\#$  not only requires 40 iterations to create the Krylov subspace that is not small, but also performs additional iterations on this Krylov subspace to get the results. If  $m$  was increased above  $\lceil k/2 \rceil$ , this would make  $\text{PSR}^\#$  establish a “high-dimensional subspace” to approximate the (low) original space, leading to the “dimensionality expansion” efficacy of  $\text{PSR}^\#$ . In practice,  $m$  is often set to be moderately smaller than  $\lceil k/2 \rceil$ .

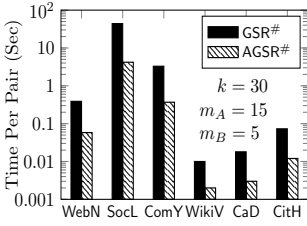


Fig. 24. Time of GSR<sup>#</sup> and AGSR<sup>#</sup> on Real Datasets

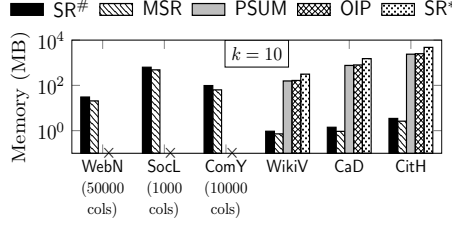


Fig. 25. Memory of SR<sup>#</sup> for Single Source & All-Pairs Search on Real Datasets

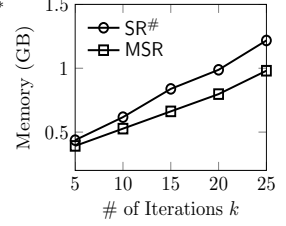


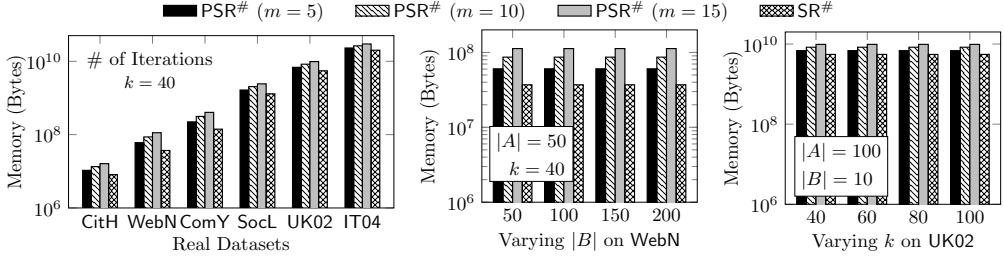
Fig. 26. Effects of  $k$  on Memory of SR<sup>#</sup> over SocL

To evaluate partial-pairs similarities  $\{s(a, b)\}_{\forall a \in A \text{ and } \forall b \in B}$  between two query sets  $A$  and  $B$ , Figure 23 investigates how the computational time of PSR<sup>#</sup> is affected by the size of queries ( $|A| \times |B|$ ) and low dimensionality ( $m$ ). We fix the size of the query set  $|A| = 50$  and the number of iterations  $k = 40$ , and vary the size  $|B|$  of the query set from 50 to 200, and low dimensionality  $m$  from 5 to 15 for PSR<sup>#</sup>. Due to similar trend, we report the results on three datasets (WikiV, WebN, and UK02). It is observed that (1) on each dataset, when  $m$  is fixed, the computational time of PSR<sup>#</sup> increases linearly when the query size  $|B|$  is growing. This agrees well with our computational complexity analysis of Algorithm 3 in Section 4.3. (2) More importantly, the larger  $m$ , the more rapid growth rate of the time for PSR<sup>#</sup>, highlighting the higher scalability and computational efficiency of our Krylov subspace-based approach for smaller setting of low dimensionality  $m$ . (3) In all cases, when  $|B|$  increases, the growth rate of the time for SR<sup>#</sup> is the fastest as compared with those of PSR<sup>#</sup> for any  $m \in \{5, 10, 15\}$ . Therefore, SR<sup>#</sup> is not scalable well to sizable query sets on large graphs, as expected. (4) On large datasets, the disparity of the time between PSR<sup>#</sup> and SR<sup>#</sup> is more pronounced. This indicates the effectiveness of our dimensionality reduction techniques that make PSR<sup>#</sup> highly scalable over sizable graphs.

**7.3.3 Computational Time of GSR<sup>#</sup> and AGSR<sup>#</sup>.** Figure 24 compares the computational time of GSR<sup>#</sup> and AGSR<sup>#</sup> on real datasets. We randomly pick 150 pairs  $\{(a, b)\}_{a \in G_A \text{ and } b \in G_B}$  across two graphs  $G_A$  and  $G_B$  for similarity assessment through GSR<sup>#</sup> and AGSR<sup>#</sup>, respectively. For AGSR<sup>#</sup>, since  $|G_A| > |G_B|$  on every real dataset, we set low dimensionality  $m_A = 15$  for  $G_A$  and  $m_B = 5$  for  $G_B$ . For each graph  $G_A$  in the real datasets, we sample several graphs for  $G_B$ , with their vertex sizes ranging over  $\{10, 30, 100, 300, 1000, 3000\}$ . Due to similar tendency of the performance over different  $G_B$ , we just report the results of GSR<sup>#</sup> and AGSR<sup>#</sup> in the case of  $|G_B| = 30$ . The results on average time for each node-pair taken by GSR<sup>#</sup> and AGSR<sup>#</sup> are reported in Figure 24, respectively. We see that (1) both GSR<sup>#</sup> and AGSR<sup>#</sup> scale well on each dataset. The larger dataset, the more running time required, which conforms with our time complexity analysis in Section 6.3. Even on large dataset SocL with 68 million edges, it takes only 4.2 seconds for AGSR<sup>#</sup> and 43.9 seconds for GSR<sup>#</sup>, highlighting the scalability of our generalised “cosine-based” SimRank algorithms on large graphs. (2) AGSR<sup>#</sup> is constantly 5.1–10.4 times faster than GSR<sup>#</sup>, which is due to the effectiveness of our Arnoldi dimensionality reduction method. Noticeably, the speedup achieved by AGSR<sup>#</sup> is more apparent on larger datasets. For instance, on small CaD and CitH, AGSR<sup>#</sup> is about 6.1–6.7 times faster than GSR<sup>#</sup>, whereas on large ComY and SocL, the speedup of AGSR<sup>#</sup> increases to 8.9–10.4 times. This is because, with an increase in graph size, GSR<sup>#</sup> spends more time for every iteration on the graph with the original dimension, in contrast with AGSR<sup>#</sup> that only iterates  $m_A$  and  $m_B$  times on the original graph to construct a low-dimensional Krylov subspace and performs the rest of the iterations in the small Krylov subspace, thereby saving much computational cost.

## 7.4 Evaluation on Memory Efficiency




Fig. 27. Memory of  $PSR^\#$  on Real-life Datasets

**7.4.1 Memory Efficiency of  $SR^\#$ .** Figure 25 shows the memory of  $SR^\#$ ,  $MSR$ ,  $PSUM$ ,  $OIP$ ,  $SR^\#$  on six real datasets. The results are quite similar for both single-source and all-pairs similarity computations. (1) For large  $WebN$ ,  $SocL$  and  $ComY$ , only  $SR^\#$  and  $MSR$  survive, highlighting their scalability. (2) For each dataset,  $SR^\#$  requires slightly more memory than  $MSR$  because it requires to store the iterative diagonal correction matrix  $D_k$ . This additional memory is negligible, as compared with the memory requirements for these datasets.

Figure 26 reports the impact of the iteration number  $k$  on the memory of  $SR^\#$  and  $MSR$  on  $SocL$ . The results on other datasets are similar, and thus are omitted here. (1) When  $k$  varies from 5 to 25, the memory requirements of  $SR^\#$  and  $MSR$  increase, since they need to memorize the  $k$  intermediate vectors from previous iterations, as expected. (2) The disparity in the memory between  $SR^\#$  and  $MSR$  is due to storing  $D_k$ .

**7.4.2 Memory Efficiency of  $PSR^\#$ .** Figure 27 depicts the memory of  $SR^\#$  with  $PSR^\#$  under different settings of low dimensionality  $m \in \{5, 10, 15\}$  on real datasets. We first compare the memory of  $SR^\#$  and  $PSR^\#$  on six real datasets. Fixing the number of iterations  $k = 40$ , we randomly select  $(|A|, |B|) = (100, 50)$  (*resp.*  $(|A|, |B|) = (100, 10)$ ) pairs as queries for each medium (*resp.* large) real dataset. The results are illustrated in Figure 27 (left). When increasing  $m$  from 5 to 15 for  $SR^\#$  on each dataset, we can discern that (1) the memory of  $PSR^\#$  is slightly increasing, due to the increasing size of the upper block Hessenberg matrix  $H_m$  that entails more space to be memorised after Arnoldi decomposition. (2) In all cases,  $PSR^\#$  has consistently comparable memory usage with  $SR^\#$ . With the growing size of datasets, the memory gap between  $PSR^\#$  and  $SR^\#$  becomes negligibly small.

Next, on each real dataset, we vary the size of queries and see how this affects the memory of  $PSR^\#$ . Due to the similar tendency on each dataset, we only report the results on  $WebN$ , as shown in Figure 27 (middle). We can see that, when the query size  $|B|$  increases from 50 to 200, the memory usage of  $PSR^\#$  (for each fixed  $m$ ) remains quite stable. This is because the size of our Krylov subspace is mainly determined by the low dimensionality  $m$  and the number of nodes  $n$  in the graph. When the similarity over each query is computed, the memory used by our Krylov subspace *w.r.t.* to this query will be released for generating the Krylov space *w.r.t.* the next query. Thus, no additional space is required to compute similarities for different queries.

Finally, on every dataset, we investigate the effect of the number of iterations ( $k$ ) on the memory of  $PSR^\#$ . We fixed query size  $(|A|, |B|) = (100, 10)$  and report the results on  $UK02$  only. The results on other datasets are similar, and omitted here for space interests. It can be observed that, when  $k$  grows from 40 to 100, the memory of  $PSR^\#$  for any fixed  $m$  is insensitive to the increasing number of iterations, as expected. The reason is that, once constructed, our Krylov subspace can be reused for all the subsequent iterations. Consequently, increasing  $k$  leads to extra time to perform more iterations *w.r.t.* the upper block Hessenberg matrix  $H_m$  on the small Krylov subspace, which is insensitive to memory space.

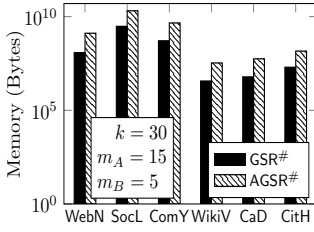


Fig. 28. Memory of GSR<sup>#</sup> and AGSR<sup>#</sup> on Six Real Datasets

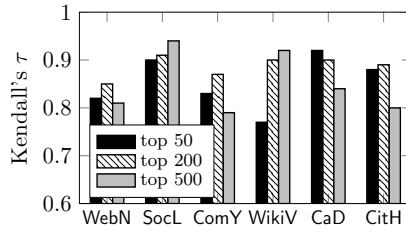


Fig. 29. Ranking Comparison of Relative Order Between LSR and JSR

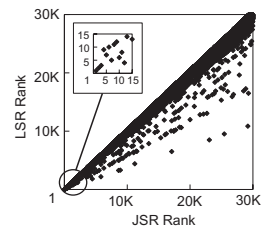


Fig. 30. Top-30K Rankings by LSR and JSR on WikiV

**7.4.3 Memory Efficiency of GSR<sup>#</sup> and AGSR<sup>#</sup>.** Figure 28 reports the memory consumption required by GSR<sup>#</sup> and AGSR<sup>#</sup>, respectively, on each real dataset. It is discerned that (1) when the size of the graph is growing, the memory requirement for each algorithm also increases. This agrees well with our space complexity analysis in Section 6.3. (2) Moreover, on each dataset, the memory required by GSR<sup>#</sup> is always a bit smaller than that by AGSR<sup>#</sup>. The additional space consumed by AGSR<sup>#</sup> is due to the temporary storage of the generated Krylov subspace. In comparison, GSR<sup>#</sup> takes less memory space but entails far more time for similarity search as it performs all the iterations in the original space, with no need of the extra memory to build the Krylov subspace.

## 7.5 Evaluation of Semantic Difference Between LSR and JSR

**7.5.1 Relative Order Comparison.** Figure 29 compares the relative order between LSR and JSR for the top  $K$  results on 6 real datasets ( $K = 50, 200, 500$ ). The order gap is measured by Kendall's  $\tau$ . (1) For different graphs, the quality of the relative order is irrelevant to top  $K$  size. For instance, on SocL, top 500 (0.94) is better preserved than top 200 (0.91) and top 50 (0.9), whereas on CaD, top 500 (0.84) is worse than both top 200 (0.9) and top 50 (0.92). (2) On each dataset, the average Kendall's  $\tau$  for top 50 is 0.77–0.92, which indicates that LSR does not maintain the relative rank of JSR, even for top 50. Thus, approximating  $D$  by  $(1 - \gamma)I$  would adversely affect the top  $K$  ranking.

Further, a qualitative result on WikiV is depicted in Figure 30, where  $x$  (*resp.*  $y$ ) axis is the ranking by JSR (*resp.* LSR). Other datasets also statistically exhibit similar phenomena. (1) Many points below the diagonal imply that low-ranked node-pairs by JSR have greater likelihood to get promoted to a high rank of LSR. This association does not imply a (near) linear relationship between the rankings of JSR and LSR. (2) For high top- $K$  ranking (*e.g.*,  $K = 15$ ), the top 15 of JSR may be inconsistent with those of LSR. Hence, the relative order preservation of JSR and LSR hinges on network structure.

**7.5.2 Relative Error Comparison.** To validate the correctness of our formulae in Theorem 10 that effectively convert from Li *et al.*'s SimRank similarity  $\tilde{S}$  to Jeh and Widom's SimRank similarity  $S$ , Figure 32 compares the  $k$ -th iterative error of our conversion formulae in Theorem 10 with that of the naive iterative model performed by JSR. Note that the  $k$ -th iterative error of our formulae in Theorem 10 arises from our replacement of the infinite sums with the first  $k$ -th partial sums when  $S$  is converted from  $\tilde{S}$  via Theorem 10. To measure the error, the exact solution  $S$  to Jeh and Widom's SimRank model is obtained by performing JSR after  $k = 100$  iterations because the accuracy bound proposed by Lizorkin *et al.* [34],  $|S(a, b) - S_k(a, b)| \leq \gamma^{k+1} = 0.6^{101} = 3.91 \times 10^{-23}$ , implies that  $S_k$  with  $k = 100$  can guarantee similarity values accurate to at least 20 decimal places for  $\gamma = 0.6$ . When varying the number of iterations  $k$  from 1 to 20 on WikiV, we notice that (1) the  $k$ -th iterative errors of both methods decrease to 0, indicating the correctness of our conversion formulae in Theorem 10. (2) Moreover, for each iteration  $k$ , the  $k$ -th iterative error of our conversion formulae is consistently smaller than that of JSR. The reason that, unlike Theorem 10 that starts from Li

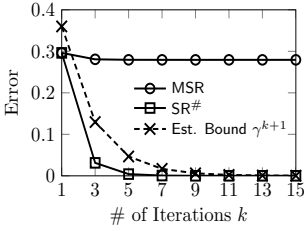


Fig. 31. Absolute Error of  $(\epsilon_{\text{diag}} + \epsilon_{\text{iter}})$  vs.  $k$

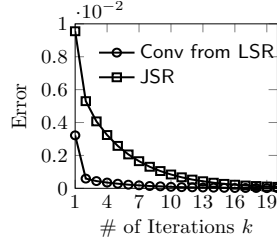


Fig. 32. Relative Error to Convert from LSR to JSR

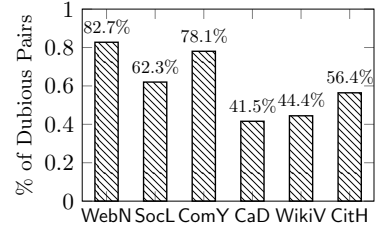


Fig. 33. % of Pairs with "Connectivity Trait" Problems on Real Datasets

*et al.*'s SimRank similarity  $\tilde{S}$  to get Jeh and Widom's solution  $S$ , JSR starts from the identity matrix  $I$  which is less close to  $S$  than  $\tilde{S}$  to  $S$ .

**7.5.3 Absolute Error Comparison.** Figure 31 tests  $(\epsilon_{\text{diag}} + \epsilon_{\text{iter}})$  of MSR and  $\text{SR}^\#$  w.r.t.  $k$ . When  $k$  increases from 1 to 15, the error of each algorithm decreases. While the error of  $\text{SR}^\#$  approaches 0, MSR levels off at 0.28. The large disparity between their convergent solutions is due to the approximation of  $D$  by  $(1 - \gamma)I$  in MSR; our "varied- $D$ " iterative model can guarantee the error to be 0 when  $k$  increases. The  $\text{SR}^\#$  curve is always below the Est. Bound curve, as expected.

**7.5.4 Significance of The "Connectivity Trait" Problem.** Figure 33 statistically shows the percentage of node-pairs with the "connectivity trait" problem over all real datasets. From the results, we see that the percentages are all high (e.g., 82.7% on WebN, 62.3% on SocL, 78.1% on ComY), showing the seriousness of this problem in real scenarios. Fortunately, all these node-pairs having "connectivity trait" problems can be resolved by  $\text{SR}^\#$ .

## 8 RELATED WORK

### 8.1 SimRank Computation

In recent years, many efficient methods have been proposed for SimRank computation. There are two ways to categorize them:

(a) According to accuracy, previous works can be classified into *deterministic* methods [1, 11, 15, 19, 25, 34, 47], and *probabilistic* methods [8, 9, 26, 37, 40, 43, 46]. Deterministic methods are accurate yet cost-inhibitive, whereas probabilistic methods are scalable on large graphs but produce random errors. Among the existing deterministic methods, Kusumoto *et al.* [25] is an efficient algorithm for accurate SimRank similarity search, which is based on an appealing "linearized SimRank formula". However, this formula is based on an assumption that the exact value of the diagonal correction matrix  $D$  is precomputed in advance. In practice, it is difficult to accurately obtain  $D$ , leading to the superfluous error  $\epsilon_{\text{diag}}$ . In contrast, our "varied- $D$ " iterative method does not need to obtain the exact value of  $D$  while accurately computing similarities within linear memory.

(b) According to queries, existing similarity search algorithms can be categorized as follows:

- *Single-Source* [11, 21, 25, 26, 43]. Given a query  $i \in V$ , retrieve  $s(*, i) := \{s(x, i) \mid \forall x \in V\}$ .
- *All-Pairs* [29, 34, 47, 50]. Retrieve all similarities  $s(*, *) := \{s(x, y) \mid \forall (x, y) \in V^2\}$ .
- *Single-Pair* [9, 25, 30]. Given a pair of queries  $(i, j) \in V^2$ , retrieve  $s(i, j)$ .
- *Partial-Pairs* [39, 51, 55]. Given two subsets  $(A, B) \subseteq V^2$ , retrieve  $s(A, B) := \{s(i, j) \mid \forall (i, j) \in A \times B\}$ .

In Table 1, we summarize the accuracy, computational time, and memory space of the previous deterministic methods for each type of SimRank search. Compared with the existing method [25], our techniques not only well preserve the scalability of [25], but also achieve high accuracy and

| Type          | Algorithm                   | Error  | Time  | Memory                            | Core Techniques                          |
|---------------|-----------------------------|--|---|-----------------------------------|--|
| single source | Proposed                    | $\gamma^{k+1}$   | $O(k E )$   | $O( E  + k V )$                   | “varied- $D$ ” linearization             |
|               | Kusumoto <i>et al.</i> [25] | $(\frac{\gamma^{k+1}}{1-\gamma}) + \epsilon_{\text{diag}}$ | $O(k^2 E )$   | $O( E  + k V )$                   | linearization                            |
|               | Fujiwara <i>et al.</i> [11] | $\epsilon_{\text{rank-}r} + \epsilon_{\text{diag}}$        | $O(r V ^2 + r V )$  | $O(r V ^2)$                       | rank- $r$ matrix decomposition           |
|               | Lee <i>et al.</i> [26]      | $\gamma^{k+1}$   | $O(d^{2k})$   | $O(d^{2k} +  V )$                 | random surfer pair (iterative)           |
|               | Wang <i>et al.</i> [42]     | prob.  | $O(\frac{\log V }{\epsilon^2} +  E  \log(\frac{1}{\epsilon}))$              | $O( V  \log(\frac{1}{\epsilon}))$ | Monte-Carlo sampling                     |
|               | Lu <i>et al.</i> [35]       | prob.  | $O( E  + \frac{ E + V }{\epsilon})$   | $O(\frac{ E + V }{\epsilon})$     | sampling + steepest decent               |
|               | Tian <i>et al.</i> [40]     | prob.  | $O( E  \log^2(\frac{1}{\epsilon}) +  V  \log(\frac{ V }{\delta})/\epsilon)$ | $O(\frac{ E + V }{\epsilon})$     | sampling + indexing                      |
| all pairs     | Proposed                    | $\gamma^{k+1}$   | $O(k V  E )$  | $O( E  + k V )$                   | “varied- $D$ ” linearization             |
|               | Kusumoto <i>et al.</i> [25] | $(\frac{\gamma^{k+1}}{1-\gamma}) + \epsilon_{\text{diag}}$ | $O(k^2 V  E )$  | $O( E  + k V )$                   | linearization                            |
|               | Yu <i>et al.</i> [47]       | $\gamma^{k+1}$   | $O(kd' V ^2)$   | $O( V ^2)$                        | fine-grained partial sums memoization    |
|               | Lizorkin <i>et al.</i> [34] | $\gamma^{k+1}$   | $O(k \min\{ V  E , \frac{ V ^2}{\log_2 V }\})$                              | $O( V ^2)$                        | partial sums memoization                 |
|               | Li <i>et al.</i> [29]       | $\epsilon_{\text{rank-}r} + \epsilon_{\text{diag}}$        | $O(r^4 V ^2)$   | $O(r^2 V ^2)$                     | tensor product + rank- $r$ decomposition |
|               | Jeh <i>et al.</i> [19]      | $\gamma^{k+1}$   | $O(k E ^2)$   | $O( V ^2)$                        | naive iteration                          |
|               | Proposed                    | $\gamma^{k+1}$   | $O(k E )$   | $O( E  + k V )$                   | “varied- $D$ ” linearization             |
| single pair   | Kusumoto <i>et al.</i> [25] | $(\frac{\gamma^{k+1}}{1-\gamma}) + \epsilon_{\text{diag}}$ | $O(k E )$   | $O( E  + k V )$                   | linearization                            |
|               | He <i>et al.</i> [16]       | $\gamma^{k+1}$   | $O(k E ^2 -  E )$   | $O( V ^2)$                        | random surfer pair + position matrix     |
|               | Li <i>et al.</i> [30]       | $\gamma^{k+1}$   | $O(kd^2 \min\{d^k,  V ^2\})$  | $O( V ^2)$                        | random surfer pair (iterative)           |
|               | Lu <i>et al.</i> [35]       | prob.  | $O( E  + \frac{ E + V }{\epsilon})$   | $O(\frac{ E + V }{\epsilon})$     | sampling + steepest decent               |

Table 1. A comparison of our approach with previous methods for similarity search, where  $r (\leq |V|)$  is the low rank,  $d = |E|/|V|$  is the average degree of the graph, and  $d' \leq d$

fast computational time. Furthermore, for achieving high accuracy, our methods not only remove superfluous error  $\epsilon_{\text{diag}}$  but also attain a better bound on  $\epsilon_{\text{iter}}$  than [25].

Recent years have witnessed an upsurge of interest in link-based pairwise similarity retrieval. Lu *et al.* [35] devised a matrix sampling approach for fast single-pair and single-source SimRank computation, which guarantees the number of nonzeros in the sparsified matrix is  $O(|V|/\epsilon)$ . On top of that, a steepest descent method is employed to speed up the computation of single-source SimRank in  $O(1/\epsilon^2)$  time. Wang *et al.* [42] suggests a probabilistic Monte-Carlo algorithm, ExactSim, to retrieve top-K single-source SimRank on large graphs of  $10^6$  nodes quickly, with high probability accuracy guaranteed. Recently, a variation of SimRank, namely SimRank\* [49], is proposed to address the undesirable zero-similarity problems of SimRank. Moreover, a scalable clustering strategy via edge concentration is designed to accelerate similarity retrieval. Youngmann *et al.* [46] presented SemSim, which enriches SimRank similarity with semantics. SemSim is applied on a weighted attributed graph. Thus, our “cosine-based” SimRank model can be extended to attributed graphs by integrating the idea of SemSim for achieving semantic richness. READS [21] provides a random walk based scheme to dynamically compute SimRank. It precomputes  $\sqrt{c}$ -walks and squeezes random walks into compact trees. In the query processing, READS searches the walks starting at query node  $u$ , and retrieves all the  $\sqrt{c}$ -random walks that meet the  $\sqrt{c}$ -walks of  $u$ .

## 8.2 “Connectivity Trait” Problem

Fogaras *et al.* [9] is the first to notice one special case of the SimRank “connectivity trait” problem: “if two nodes  $a$  and  $b$  have  $\beta$  common (direct) in-neighbors, then  $s(a, b) \leq 1/\beta$ .” To address this problem, they employed an unwieldy method that divides the entire search space into three probabilities:  $\frac{|N_a^- \cap N_b^-|}{|N_a^- \cup N_b^-|}$ ,  $\frac{|N_a^- - N_b^-|}{|N_a^- \cup N_b^-|}$ , and  $\frac{|N_b^- - N_a^-|}{|N_a^- \cup N_b^-|}$ . However, this complicates the revised SimRank equation.

Recently, Antonellis *et al.* [1] gave an excellent revision, called SimRank++, by introducing the “evidence factor”  $\tilde{\gamma}$ . Unfortunately,  $\tilde{\gamma}$  can only, in part, alleviate a special case of the “connectivity trait” problem, since, if  $|N_a^- \cap N_b^-| = 0$ , then  $\tilde{\gamma} = 0$  has no recursive impact on SimRank any more.

Jin *et al.* [22] also gave an excellent exposition on “role similarity”. Their proposed model, namely RoleSim, has the advantage of utilizing “automorphic equivalence” to improve the quality

of similarity search in “role” based applications. Their initial intention, however, was not to deal with the SimRank “connectivity trait” problem.

There is also a SimRank-like “connectivity trait” problem in other link-based similarity models, such as Penetrating-Rank (P-Rank) [53], Random Walk with Restart (RWR) [41], ASCOS++ [2], SimRank\* [49], SimFusion [44]. Our proposed solutions for fixing the SimRank “connectivity trait” problem are also extensible to these models.

### 8.3 Semantics between SimRank and Its Variant

There are some interesting works (e.g., [11, 15, 29, 48, 51]) based on Li *et al.*'s model:  $\tilde{S} = \gamma P^T \tilde{S} P + (1 - \gamma)I$ . [25] argued that “the top-K rankings of  $\tilde{S}$  and  $S$  in Eq.(1) are not affected much”. However, we correct this argument, provide new mathematical insights into the subtle difference of  $\tilde{S}$  and  $S$ .

## 9 CONCLUSIONS

In this article, we consider the problem of high-quality link-based similarity search. Firstly, we devise a “varied- $D$ ” model to compute SimRank with no  $\epsilon_{\text{diag}}$  in linear memory. We also speed up the computational time from quadratic [25] to linear in terms of  $k$ . Secondly, we devise a “cosine-based” SimRank model for similarity search to circumvent the “connectivity trait” problem of SimRank. Thirdly, we propose an efficient dimensionality reduction method via block Arnoldi–Ruhe iterations, which drastically speeds up partial-pairs “cosine-based” SimRank similarity join on sizable graphs, with provable guarantees on accuracy. Fourthly, we give new insights into the semantic difference between Jeh and Widom’s SimRank and its variant, and correct an argument in [25]. Fifthly, we also design a novel approach that allows us to make instant conversion from Li *et al.*'s SimRank to Jeh and Widom’s SimRank, without any loss of accuracy, and provide key intuitions underpinning our conversion formulae. Sixthly, we notice that, if two nodes are in two distinct graphs (or two disconnected components), the existing SimRank model as well as the “cosine-based” model would assess their nodes as completely dissimilar. To alleviate this problem, we propose our generalised “cosine-based” SimRank model to effectively quantify the similarity for nodes across two distinct graphs (or two disconnected components). Finally, our extensive experiments on a variety of real datasets validate the superiority of our proposed approaches in terms of high accuracy, scalability, and computational efficiency.

In our future work, we will extend the “cosine-based” model to retrieve the similarity between subgraphs, where the query may not be a node in the given graph. Another avenue is incorporating semantic and structural information of the graph through ontology matching and entity resolution to design a more comprehensive similarity model.

**Acknowledgments.** This work was supported by the National Natural Science Foundation of China (NSFC 61972203), and Natural Science Foundation of Jiangsu Province (BK20190442).

## REFERENCES

- [1] I. Antonellis, H. G. Molina, and C. Chang. SimRank++: Query rewriting through link analysis of the click graph. *PVLDB*, 1(1), 2008.
- [2] H. Chen and C. L. Giles. ASCOS++: an asymmetric similarity measure for weighted networks to address the problem of simrank. *ACM Trans. Knowl. Discov. Data*, 10(2):15:1–15:26, 2015.
- [3] P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012.
- [4] W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst.*, 18(3):288–321, 2000.
- [5] N. Craswell and M. Szummer. Random walks on the click graph. In *ACM SIGIR*, pages 239–246. ACM, 2007.
- [6] P. Dey, K. Goel, and R. Agrawal. P-Simrank: Extending simrank to scale-free bipartite networks. In *WWW*, pages 3084–3090. ACM, 2020.

- [7] S. Elsworth and S. Guttel. The block rational arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 41(2):365–388, 2020.
- [8] D. Fogaras and B. Rácz. A scalable randomized method to compute link-based similarity rank on the web graph. In *EDBT Workshops*, 2004.
- [9] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, 2005.
- [10] S. Fox, K. Karnawat, M. Mydland, S. T. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168, 2005.
- [11] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, and M. Onizuka. Efficient search algorithm for SimRank. In *ICDE*, 2013.
- [12] P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Trans. Inf. Syst.*, 21(1):64–93, 2003.
- [13] M. R. Hamedani and S. Kim. SimCC-AT: A method to compute similarity of scientific papers with automatic parameter tuning. In *ACM SIGIR*, pages 1005–1008. ACM, 2016.
- [14] M. R. Hamedani and S. Kim. Pairwise normalization in SimRank variants: problem, solution, and evaluation. In *SAC*, pages 534–541. ACM, 2019.
- [15] G. He, H. Feng, C. Li, and H. Chen. Parallel SimRank computation on large graphs with iterative aggregation. In *KDD*, 2010.
- [16] J. He, H. Liu, J. X. Yu, P. Li, W. He, and X. Du. Assessing single-pair similarity over graphs by aggregating first-meeting probabilities. *Inf. Syst.*, 42:107–122, 2014.
- [17] B. M. Hill and A. Debons. Bibliographic coupling. *J. Am. Soc. Inf. Sci.*, 23(4):286, 1972.
- [18] M. E. Houle, V. Oria, S. Satoh, and J. Sun. Annotation propagation in image databases using similarity graphs. *ACM Trans. Multim. Comput. Commun. Appl.*, 10(1):7:1–7:21, 2013.
- [19] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *KDD*, 2002.
- [20] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279. ACM, 2003.
- [21] M. Jiang, A. W. Fu, R. C. Wong, and K. Wang. READS: A random walk approach for efficient and accurate dynamic simrank. *PVLDB*, 10(9):937–948, 2017.
- [22] R. Jin, V. E. Lee, and H. Hong. Axiomatic ranking of network role similarity. In *KDD*, 2011.
- [23] I. M. Kloumann, J. Ugander, and J. M. Kleinberg. Block models and personalized PageRank. *Proc. Natl. Acad. Sci. USA*, 114(1):33–38, 2017.
- [24] O. Kurland and L. Lee. PageRank without hyperlinks: Structural reranking using links induced by language models. *ACM Trans. Inf. Syst.*, 28(4):18:1–18:38, 2010.
- [25] M. Kusumoto, T. Maehara, and K. Kawarabayashi. Scalable similarity search for SimRank. In *SIGMOD*, 2014.
- [26] P. Lee, L. V. S. Lakshmanan, and J. X. Yu. On top- $k$  structural similarity search. In *ICDE*, 2012.
- [27] R. Lempel and S. Moran. SALSA: The stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, 2001.
- [28] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *SIGKDD*, pages 631–636. ACM, 2006.
- [29] 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.
- [30] P. Li, H. Liu, J. X. Yu, J. He, and X. Du. Fast single-pair SimRank computation. In *SDM*, 2010.
- [31] X. Li, Y. Chen, B. Pettit, and M. de Rijke. Personalised reranking of paper recommendations using paper content and user behavior. *ACM Trans. Inf. Syst.*, 37(3):31:1–31:23, 2019.
- [32] Z. Lin, M. R. Lyu, and I. King. PageSim: A novel link-based measure of web page similarity. In *WWW*, pages 1019–1020. ACM, 2006.
- [33] Z. Lin, M. R. Lyu, and I. King. MatchSim: A novel similarity measure based on maximum neighborhood matching. *Knowl. Inf. Syst.*, 32(1), 2012.
- [34] D. Lizorkin, P. Velikhov, M. N. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for SimRank computation. *VLDB J.*, 19(1), 2010.
- [35] J. Lu, Z. Gong, and Y. Yang. A matrix sampling approach for efficient SimRank computation. *Inf. Sci.*, 556:1–26, 2021.
- [36] D. Rafiei and F. Deng. Similarity join and similarity self-join size estimation in a streaming environment. *IEEE Trans. Knowl. Data Eng.*, 32(4):768–781, 2020.
- [37] J. Shi, T. Jin, R. Yang, X. Xiao, and Y. Yang. Realtime index-free single source simrank processing on web-scale graphs. *PVLDB*, 13(7):966–978, 2020.
- [38] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Am. Soc. Inf. Sci.*, 24(4), 1973.
- [39] W. Tao, M. Yu, and G. Li. Efficient top- $k$  Simrank-based similarity join. *PVLDB*, 8(3):317–328, 2014.
- [40] B. Tian and X. Xiao. SLING: A near-optimal index structure for simrank. In *SIGMOD*, pages 1859–1874, 2016.
- [41] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, 2006.

- [42] H. Wang, Z. Wei, Y. Yuan, X. Du, and J. Wen. Exact single-source SimRank computation on large graphs. In D. Maier, R. Pottinger, A. Doan, W. Tan, A. Alawini, and H. Q. Ngo, editors, *SIGMOD*, pages 653–663. ACM, 2020.
- [43] Z. Wei, X. He, X. Xiao, S. Wang, Y. Liu, X. Du, and J. Wen. PRSim: Sublinear time SimRank computation on large power-law graphs. In *SIGMOD*, pages 1042–1059. ACM, 2019.
- [44] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. SimFusion: Measuring similarity using unified relationship matrix. In *SIGIR*, 2005.
- [45] J. J. Xu and H. Chen. CrimeNet Explorer: A framework for criminal network knowledge discovery. *ACM Trans. Inf. Syst.*, 23(2):201–226, 2005.
- [46] B. Youngmann, T. Milo, and A. Somech. Boosting SimRank with semantics. In *EDBT*, pages 37–48, 2019.
- [47] W. Yu, X. Lin, and W. Zhang. Towards efficient SimRank computation on large networks. In *ICDE*, 2013.
- [48] W. Yu, X. Lin, and W. Zhang. Fast incremental SimRank on link-evolving graphs. In *ICDE*, pages 304–315, 2014.
- [49] W. Yu, X. Lin, W. Zhang, J. Pei, and J. A. McCann. SimRank\*: Effective and scalable pairwise similarity search based on graph topology. *VLDB J.*, 28(3):401–426, 2019.
- [50] W. Yu and J. A. McCann. Sig-SR: SimRank search over singular graphs. In *SIGIR*, 2014.
- [51] W. Yu and J. A. McCann. Efficient partial-pairs SimRank search for large networks. *PVLDB*, 8(5):569–580, 2015.
- [52] W. Yu and J. A. McCann. High quality graph-based similarity search. In *ACM SIGIR*, pages 83–92. ACM, 2015.
- [53] P. Zhao, J. Han, and Y. Sun. P-Rank: A comprehensive structural similarity measure over information networks. In *CIKM*, 2009.
- [54] W. Zheng, L. Zou, L. Chen, and D. Zhao. Efficient SimRank-based similarity join. *ACM Trans. Database Syst.*, 42(3):16:1–16:37, 2017.
- [55] W. Zheng, L. Zou, Y. Feng, L. Chen, and D. Zhao. Efficient SimRank-based similarity join over large graphs. *PVLDB*, 6(7):493–504, 2013.