High Quality Graph-Based Similarity Search

Weiren Yu[†], Julie A. McCann[†]
[†]Imperial College London, United Kingdom
{weiren.yu, j.mccann}@imperial.ac.uk

ABSTRACT

SimRank is an influential link-based similarity measure that has been used in many fields of Web search and sociometry. The best-of-breed method by Kusumoto *et al.* [7], however, does not always deliver high-quality results, since it fails to accurately obtain its diagonal correction matrix D. Besides, SimRank is also limited by an unwanted "connectivity trait": increasing the number of paths between nodes a and b often incurs a decrease in score $s(a, b)$. The best-known solution, SimRank++ [1], cannot resolve this problem, since a revised score will be zero if a and b have no common in-neighbors.

In this paper, we consider high-quality similarity search. Our scheme, SR$^{\#}$, is efficient and semantically meaningful: (1) We first formulate the exact D, and devise a "varied-D" method to accurately compute SimRank in linear memory. Moreover, by grouping computation, we also reduce the time of [7] from quadratic to linear in the number of iterations. (2) We design a "kernel-based" model to improve the quality of SimRank, and circumvent the "connectivity trait" issue. (3) We give mathematical insights to the semantic difference between SimRank and its variant, and correct an argument in [7]: "if D is replaced by a scaled identity matrix $(1-\gamma)I$, top-K rankings will not be affected much". The experiments confirm that SR$^{\#}$ can accurately extract high-quality scores, and is much faster than the state-of-the-art competitors.

Categories and Subject Descriptors

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

Keywords

Link Analysis; Graph-Based Similarity; High Quality Search

1. INTRODUCTION

The Web today is a huge, self-organized, and hyperlinked network. These salient features bring striking challenges to data management, and call for new search abilities to extract meaningful knowledge automatically from the gigantic Web. Link-based similarity search is a modern means to quantify node-to-node relationships based on graph topologies, with a wide range of successful applications, *e.g.,* link prediction, collaborative filtering, and co-citation analysis.

SimRank, conceived by Jeh and Widom [5], is one of the most influential similarity measures. The central idea underpinning SimRank is a simple recursion that "two nodes are assessed as similar if they are in-linked from similar nodes". For 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. Then, SimRank score between nodes a and b is defined by[1]

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

where $\gamma \in (0, 1)$ is a decay factor. Generally, $\gamma = 0.6$ [12] or 0.8 [5], which penalizes long paths relative to short ones.

In contrast with other similarity measures, SimRank has the following prominent features: (a) It takes a concise form that captures both direct and indirect neighbors recursively, unlike Bibliographic Coupling and Co-citation that focus only on direct neighbors. (b) It considers structural equivalence of two nodes, whereas Personalized PageRank focuses on reachability from every node to a query. Therefore, SimRank has attracted increasing attention in recent years [3, 4, 12].

1.1 The Quality of SimRank Search

Despite much effort devoted to fast SimRank computation (*e.g.,* [2, 7, 8, 12, 13]), the quality of SimRank search is still less desirable, due to the following two reasons:

(1) Superfluous Diagonal Correction Error $\epsilon_{\mathbf{diag}}$. The best-of-breed SimRank method by Kusumoto *et al.* [7] is based on the following "linearized SimRank formula":

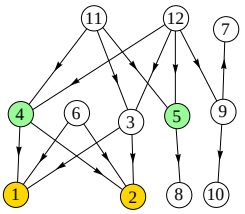
$$s(a,b) = e_a^\top D e_b + \gamma (Pe_a)^\top D (Pe_b) + \gamma^2 (P^2 e_a)^\top D (P^2 e_b) + \cdots \quad (2)$$

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, with $P_{a,b} = \begin{cases} 1/|N_b|, & \text{if } (a,b) \in E; \\ 0, & \text{if } (a,b) \notin E. \end{cases}$

According to [7], 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.* [7] have not obtained the exact D, but simply approximated D by $\tilde{D} := (1-\gamma)I$. Consequently, *the diagonal correction error* is produced:

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

[1]To avoid division by 0 in Eq.(1), $s(a, b) = 0$ if $|N_a||N_b| = 0$.



	$s(1,2)$	$s(4,5)$	$s(2,8)$	$s(8,10)$	$s(3,9)$
SR	0.24	0.30	0.12	0.18	0.30
SR^{++}	0.20	0.23	0	0	0.15
RS	0.34	0.24	0.05	0.07	0.12
SR$^{\#}$	0.38	0.24	0.14	0.10	0.17

	SR	SR^{++}	RS	SR$^{\#}$
$s(1,2) > s(4,5)$	✗	✗	✓	✓
$s(2,8) > s(8,10)$	✗	✗	✗	✓
$s(4,5) > s(3,9)$	✗	✓	✓	✓

Figure 1: SimRank++ (SR^{++}) and RoleSim (RS) may not resolve the "connectivity trait" problem of SimRank (SR)

where $s_{\tilde{D}}(a,b)$ is the estimated similarity when D is replaced by $\tilde{D}$ in Eq.(2). After D is estimated, [7] uses an iterative method that sums up only the first k terms of series $s_{\tilde{D}}(a,b)$, denoted as $s^{(k)}_{\tilde{D}}(a,b)$. This yields *the iterative error*:

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

Hence, the total error for approximating $s(a,b)$ by $s^{(k)}_{\tilde{D}}(a,b)$, in a nutshell, consists of two ingredients: ϵ_{diag} and ϵ_{iter}.

We argue ϵ_{diag} is far more serious than ϵ_{iter}, because ϵ_{iter} is guaranteed to converge by [7], and can be minimized by increasing the number of iterations. This increase, however, cannot minimize ϵ_{diag}. Worse still, there is no bound on ϵ_{diag} for Eq.(3). The only argument about ϵ_{diag} in [7] 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, as will be shown in Section 4.1, bears a blemish.

This motivates us to design an accurate and fast approach that has no ϵ_{diag} and can avoid computing the exact D.

(2) "Connectivity Trait" Problem. Another factor that plagues the quality of SimRank is the "connectivity trait": 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, 2, 11] only noticed a special case (1-hop neighbor) of the above phenomenon: "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 γ in Eq.(1) with the following "evidence factor":

$$\tilde{\gamma} := \gamma(1 - e^{-|N_a \cap N_b|}) \quad \text{or} \quad \tilde{\gamma} := \gamma \textstyle\sum_{i=1}^{|N_a \cap N_b|} \frac{1}{2^i} \qquad (4)$$

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

However, we observe a weakness of SimRank++ [1]: *SimRank++ score $\tilde{s}(a,b)$ is always zero if nodes a and b have no common (direct) in-neighbors.* This is because, by the definition in Eq.(4), if $N_a \cap N_b = \varnothing$, 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 [6], PSimRank [2], and MatchSim [11]) to quantify $s(a,b)$ 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 1.3.2 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. We evaluate the similarity of each node-pair by 4 measures: (a) SR (Jeh and Widom's SimRank [5]); (b) SR^{++} (SimRank++ [1]); (c) RS (RoleSim [6]); (d) SR$^{\#}$ (our method). The results are partly depicted in the table. We notice that SR^{++} and RS do not well resolve the SR "connectivity trait".*

For example, most people may 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 common $\{11,12\}$. However, although SR^{++} narrows the gap between $s(1,2)$ and $s(4,5)$, 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, $N_2 \cap N_8 = \varnothing$; neither has $(8,10)$. Thereby, their "evidence factors" $\tilde{\gamma} = 0$. However, there are 4* indirect *path-pairs in-linked from $(2,8)$:*

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

as opposed to only 1 from $(8,10)$: $8 \leftarrow 5 \leftarrow \boxed{12} \rightarrow 9 \rightarrow 10$. 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 degrees than high connectivities. Thus, RS can only partially resolve the SR "connectivity trait" problem. □

Example 1 suggests that the state-of-the-art methods (*e.g.,* SimRank++ [1] and RoleSim [6]) cannot solidly circumvent the "connectivity trait" problem of SimRank. Unfortunately, as illustrated by our statistical experiments in Section 5.2, 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 to resolve the "connectivity trait" problem.

1.2 Our Contributions

Our main contributions are summarized as follows:

- We formulate the exact diagonal correction matrix D, and propose a "varied-D" method to accurately compute SimRank with no ϵ_{diag} and in linear memory. Moreover, by grouping computation, we also optimize the algorithm [7] 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 "kernel-based" model and improve the search quality. (Section 3)
- We give mathematical insights to the semantic difference between Jeh and Widom's model [5] and its variant [9], and correct an argument [7]: if D is replaced by $(1-\gamma)I$, top-K rankings will not be affected much. (Section 4)

The comprehensive experiments verify that our methods (1) improve an accuracy of average NDCG$_{200}$ by ∼30% over SimRank on various real networks, and (2) are ∼10x faster than the state-of-the-art competitors on large datasets with 65.8M links for 1000 queries.

1.3 Related Work

1.3.1 SimRank Computation

Recent years have witnessed a surge of efficient methods to compute SimRank. They can be categorized as follows:

- *Single-source SimRank* [3, 7, 8]. Compute all $s(i,*)$.
- *All-pairs SimRank* [9, 12, 13, 16]. Compute all $s(*,*)$.
- *Single-pair SimRank* [2, 7, 10]. Compute $s(i,j)$.

Type	Algorithm	Error	Time	Memory
single source	Proposed	γ^{k+1}	$O(k\|E\|)$	$O(\|E\|+k\|V\|)$
	Kusumoto *et al.* [7]	$(\frac{\gamma^{k+1}}{1-\gamma})+\epsilon_{\text{diag}}$	$O(k^2\|E\|)$	$O(\|E\|+\|V\|)$
	Fujiwara *et al.* [3]	$\epsilon_{\text{rank-}r}+\epsilon_{\text{diag}}$	$O(r\|V\|^2)$	$O(r\|V\|^2)$
	Lee *et al.* [8]	γ^{k+1}	$O(d^{2k})$	$O(d^{2k}+\|V\|)$
all pairs	Proposed	γ^{k+1}	$O(k\|V\|\|E\|)$	$O(\|E\|+k\|V\|)$
	Kusumoto *et al.* [7]	$(\frac{\gamma^{k+1}}{1-\gamma})+\epsilon_{\text{diag}}$	$O(k^2\|V\|\|E\|)$	$O(\|E\|+\|V\|)$
	Yu *et al.* [13]	γ^{k+1}	$O(kd'\|V\|^2)$	$O(\|V\|^2)$
	Lizorkin *et al.* [12]	γ^{k+1}	$O(k\|V\|\|E\|)$	$O(\|V\|^2)$
	Yu *et al.* [16]	$\epsilon_{\text{rank-}r}+\epsilon_{\text{diag}}$	$O(r\|V\|^2)$	$O(\|V\|^2)$
	Li *et al.* [9]	$\epsilon_{\text{rank-}r}+\epsilon_{\text{diag}}$	$O(r^4\|V\|^2)$	$O(r^2\|V\|^2)$
	Jeh *et al.* [5]	γ^{k+1}	$O(k\|E\|^2)$	$O(\|V\|^2)$

Table 1: A comparison with previous deterministic methods (with low-rank $r \leq |V|$, degree $d = \frac{|E|}{|V|}$, and $d' \leq d$)

In Table 1, we briefly summarize the accuracy, time, and memory of previous works for each type of SimRank search.

Compared with the best-known method [7], our techniques not only well preserve the scalability of [7], but also achieve high accuracy and fast computational time. Furthermore, for high accuracy, our methods not only remove superfluous error ϵ_{diag} but also attain a better bound on ϵ_{iter} than [7].

1.3.2 SimRank "Connectivity Trait"

Fogaras *et al.* [2] is the first to notice one special case of the SimRank "connectivity trait" problem: "if two nodes a and b have β 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, which is rather tedious.

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.* [6] 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 SimRank variant models, such as MatchSim, SimRank*, SimFusion+ [15]. Our proposed methods for SimRank are also extensible to SimRank*. Due to space limitation, we omit it in this paper.

1.3.3 Semantics between SimRank and Its Variant

There are some interesting works (*e.g.,* [3, 4, 9, 14, 17]), based on the following model, to evaluate similarity $\tilde{S}$:

$$\tilde{S} = \gamma P^\top \tilde{S} P + (1-\gamma) I. \tag{5}$$

[7] argued that "the top-K rankings of $\tilde{S}$ in Eq.(5) and S in Eq.(1) are not affected much". However, we correct this argument, and provide new mathematical insights into the subtle difference of $\tilde{S}$ and S from a semantic perspective.

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

$$S = \max\{\gamma P^\top S P,\ I\}, \tag{6}$$

where $\max\{*\}$ denotes the matrix entry-wise maximum, *i.e.,* $(\max\{A,B\})_{i,j} = \max\{A_{i,j}, B_{i,j}\}$.

Kusumoto *et al.* [7] have showed that there exists a unique diagonal matrix D such that Eq.(6) can be converted to

$$S = \gamma P^\top S P + D, \tag{7}$$

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

However, [7] 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:*

$$S_{\tilde{D}} = \gamma P^\top S_{\tilde{D}} P + \tilde{D}, \tag{8}$$

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

$$\|\Delta S\|_{\max} \leq \tfrac{1}{1-\gamma} \|\Delta D\|_{\max}. \text{ [2]} \tag{9}$$

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

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

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

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

□

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 matrices X and Y, their entry-wise product $X \circ Y$ is defined as*

$$(X \circ Y)_{i,j} = X_{i,j} Y_{i,j}.$$

Let $diag(Z)$ be a diagonal matrix whose diagonal entries are those of Z, *i.e.,* $(diag(Z))_{i,i} = Z_{i,i}$.

Using this notation, Eq.(6) can be represented as

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

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

$$D = I - \gamma diag(P^\top S P). \tag{12}$$

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

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

$$\overrightarrow{diag}(X^\top Z Y) = (X \circ Y)^\top \overrightarrow{diag}(Z).$$

[2] $\|*\|_{\max}$ returns the maximum element of a matrix.

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

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

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

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

PROOF. Taking $\overrightarrow{diag}(*)$ on both sides of Eq.(2) produces

$$\overrightarrow{diag}(S) = \overrightarrow{diag}(D) + \gamma \overrightarrow{diag}(P^\top D P) + \gamma^2 \overrightarrow{diag}((P^\top)^2 D P^2) + \cdots \quad (14)$$

By SimRank definition Eq.(6), we have $S_{i,i} = 1$ $(\forall i \in V)$, which implies that $\overrightarrow{diag}(S) = \vec{1}$.

Applying Lemma 2 to the right-hand side of Eq.(14) yields

$$\vec{1} = \left(I + \gamma(P \circ P) + \gamma^2 (P^2 \circ P^2) + \cdots\right)^\top \overrightarrow{diag}(D), \quad (15)$$

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

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 [7] will innately produce ϵ_{diag}.

Theorem 1 also implies that the estimation $D \approx (1-\gamma)I$ in [7] 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

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

which suggests that the approximation $D \approx (1-\gamma)I$ in [7] is equivalent to the approximation $P^k \circ P^k \approx P^k$. Clearly, most people will not agree that $((P^k)_{i,j})^2 \approx (P^k)_{i,j}$ is reasonable. In Section 4.2, we will further discuss $D \approx (1-\gamma)I$ from the viewpoint of semantics.

One benefit of Theorem 1 is that it narrows the boundaries for the range of D in [7], based on the following corollary.

COROLLARY 1. $(1-\gamma)I \le D \le I - \gamma diag(P^\top P)$ [3]

PROOF. Since $0 \le P_{i,j} \le 1$, we can readily show that

$$(P \circ P)^k \le (P^k \circ P^k) \le P^k.$$

Applying this to Eq.(13) yields

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

Since $\sum_{k=0}^{+\infty} X^k = (I - X)^{-1}$, it follows that $(I - \gamma P)^\top \vec{1} \le \overrightarrow{diag}(D) \le (I - \gamma(P \circ P))^\top \vec{1}$. Then, applying Lemma 2 on both sides, we obtain the results. □

In comparison, the best-known bounds for the range of D in Proposition 2 of [7] (*i.e.,* $(1-\gamma)I \le D \le I$) are loose, and independent of P, which is isolated from graph structures.

2.3 A "Varied-D" Iterative Model

Another important consequence of Theorem 1 is to derive an accurate SimRank algorithm without ϵ_{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:

$$S^{(k)} := D_k + \gamma P^\top D_{k-1} P + \cdots + \gamma^k (P^\top)^k D_0 P^k, \quad (16)$$

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.* [7], our "varied-D" model Eq.(16) replaces all Ds by a convergent sequence $\{D_k\}$. The main advantage of our replacement is that Eq.(16) can avoid determining the *exact* D beforehand, and thereby, will not produce the superfluous error ϵ_{diag}.

The correctness of our "varied-D" model can be verified by taking limits $k \to \infty$ on both sides of Eq.(16). As $k \to \infty$, $D_k \to D$ and $S^{(k)} \to S$.[4] Thus, Eq.(16) converges to Eq.(2).

2.3.1 Finding D_k in "Varied-D" Model

The challenging problem in our "varied-D" Eq.(16) is to determine the diagonal matrix D_k. Our main idea is based on two observations: (a) $S^{(k)}$ in Eq.(16) can be iterated as

$$S^{(l)} = \gamma P^\top S^{(l-1)} P + D_l \quad \text{with} \quad S^{(0)} = D_0. \quad (17)$$

(b) To ensure $diag(S^{(l)}) = I$, D_l in Eq.(17) must satisfy

$$D_l = I - \gamma diag(P^\top S^{(l-1)} P). \quad (18)$$

Coupling these observations, we can compute D_k in Eq.(16).

THEOREM 2. *The diagonal correction matrices in Eq.(16) can be iteratively obtained as follows:*

$$(D_k)_{i,i} = 1 - \sum_{l=1}^{k} (h_l \circ h_l)^\top \overrightarrow{diag}(D_{k-l}) \quad \textit{with} \quad D_0 = I, \quad (19)$$

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

$$\begin{cases} h_0 = e_i \\ h_l = \sqrt{\gamma} P h_{l-1} \quad (l = 1, 2, \cdots, k) \end{cases} \quad (20)$$

PROOF. First, we derive a complete matrix formula of D_k. By Lemma 2, Eq.(19) can be converted to

$$(D_k)_{i,i} = 1 - \sum_{l=1}^{k} h_l{}^\top D_{k-l} h_l \quad (21)$$

Successive substitution applied to Eq.(20) yields $h_l = \sqrt{\gamma^l} P^l e_i$. Then, substituting this back into Eq.(21) produces

$$D_k = I - \sum_{l=1}^{k} \gamma^l diag\left((P^l)^\top D_{k-l} (P^l)\right) \quad (22)$$

Next, we show that D_k in Eq.(22) satisfies Eqs.(16)–(18). It follows from Eq.(16) that

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

Thus, the above equation implies that

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

Applying Eq.(22) to the right-hand side yields Eq.(18). □

Theorem 2 provides a simple efficient way to compute D_k.

Algorithm 1: Compute Diagonal Matrix D_k

```
initialize t := 0, h_0 := e_i, D_0 := I ;
for l := 1, 2, ··· , k do
    compute h_l := √γ P h_{l-1} ;
    update t := t + (h_l ∘ h_l)^⊤ diag(D_{k-l}) ;
return (D_k)_{i,i} := 1 − t ;
```

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

[3] For matrices A and B, $A \le B$ refers to $A_{i,j} \le B_{i,j}$, $\forall i, j$.

[4] The convergence of $S^{(k)}$ will be proved in Section 2.3.2.

THEOREM 3. *Given the total iteration number $k = 1, 2, \cdots$, Algorithm 1 is in $O(k|V|)$ memory and $O(k(|E|+|V|))$ time.*

In contrast to the linear-memory SimRank method in [7], Theorem 3 implies that our "varied-D" method to compute D_k will not compromise the scalability of [7] for high quality search, since D_k can be computed in linear memory as well.

2.3.2 Fast Convergence of "Varied-D" Model

Besides no ϵ_{diag} and no need to precompute the exact D, our "varied-D" model Eq.(16) also converges faster than [7].

THEOREM 4. *Let $S^{(k)}$ and S be the k-th iterative and the exact SimRank in Eqs.*(16) *and* (7)*, respectively. Then,*

$$\|S^{(k)} - S\|_{\max} \leq \gamma^{k+1}. \tag{23}$$

PROOF. We subtract Eq.(7) from Eq.(17) to obtain, $\forall k$,

$$S^{(k)} - S = \gamma P^\top (S^{(k-1)} - S)P + (D^{(k)} - D). \tag{24}$$

We notice from Eq.(18) that $(S^{(k)})_{i,i} = S_{i,i} = 1$, $\forall i \in V$. Thus, when $i \neq j$, it follows from Eq.(24) that, $\forall i, j \in V$,

$$\begin{aligned}(S^{(k)} - S)_{i,j} &= \gamma (P^\top)_{i,*} (S^{(k-1)} - S) P_{*,j}. \\ &\leq \gamma \|S^{(k-1)} - S\|_{\max} \leq \cdots \leq \gamma^k \|I - S\|_{\max}\end{aligned} \tag{25}$$

By Eq.(1), $\|I - S\|_{\max} \leq \gamma$. Thus, Eq.(23) holds. □

In comparison to the bound $\left(\frac{\gamma^{k+1}}{1-\gamma}\right)$ (see Eq.(10) of [7]), Theorem 4 shows that our "varied-D" model not only eliminates ϵ_{diag}, but also has a better bound on ϵ_{iter} than [7]. Thus, our "varied-D" model achieves both high-quality and fast convergence rate at the same time.

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

Having determined D_k in our "varied-D" model Eq.(16), we next propose our method to efficiently compute $S^{(k)}$.

The method [7] requires $O(k^2|E|)$ and $O(k^2|V||E|)$ time, respectively, to compute single-source and all-pairs SimRank. If we merely apply the method [7] and replace D with D_k, then our "varied-D" Eq.(16) to compute $S^{(k)}$ will retain the same complexity as [7] except with no ϵ_{diag}, as follows:

Procedure 2: Single-Source "Varied-D" SimRank(i)

1 initialize $h := \vec{0},\ x := e_i$;
2 **for** $l := 0, 1, \cdots, k$ **do**
3 update $h := h + \gamma^l (P^\top)^l (D_{k-l}) x,\quad x := Px$;
4 **return** $(S^{(k)})_{i,*} := h$;

However, we observe that there exist many duplicate products in [7]. Precisely, to obtain the result of the sums

$$(S^{(k)})_{i,*} = D_k x_0 + \gamma P^\top D_{k-1} x_1 + \cdots + \gamma^k (P^\top)^k D_0 x_k, \tag{26}$$

the method [7] *separately* computes every $\left(\gamma^l (P^\top)^l (D_{k-l}) x_l\right)$ and then adds them together. Its main limitation is that, to compute any power of $(P^\top)$, [7] has to go through all of the previous powers from scratch. As a result, there are l matrix-vector products to compute each h in Line 3, leading to $\sum_{l=1}^{k} l = O(k^2)$ products for k iterations in total.

We now propose an efficient method for Procedure 2, 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.(26) as follows:

$$\begin{aligned}(S^{(k)})_{i,*} = D_k x_0 + \gamma P^\top (D_{k-1} x_1 + \gamma P^\top (D_{k-2} x_2 + \cdots \\ \cdots + \gamma P^\top (D_1 x_{k-1} + \gamma P^\top (D_0 x_k))))\end{aligned} \tag{27}$$

and obtain the result by starting with the innermost brackets and working outwards. In contrast with the method [7], Eq.(27) has only $O(k)$ matrix-vector products in k brackets, as opposed to $O(k^2)$ products in Procedure 2.

Based on Eq.(27), we give an efficient way of Procedure 2.

Algorithm 3: Optimized Single-Source SimRank(i)

1 initialize $x_0 := e_i$;
2 **for** $l := 1, 2, \cdots, k$ **do**
3 update $x_l := P x_{l-1}$;
4 initialize $y_0 := \overrightarrow{diag}(D_0) \circ x_k$;
5 **for** $l := 1, 2, \cdots, k$ **do**
6 update $y_l := \overrightarrow{diag}(D_l) \circ x_{k-l} + \gamma P^\top y_{l-1}$;
7 **return** $(S^{(k)})_{i,*} := y_k$;

Algorithm 3 can reduce not only the time of single-source SimRank from $O(k^2|E|)$ [7] to $O(k|E|)$, but also the time of all-pairs SimRank from $O(k^2|V||E|)$ [7] to $O(k|V||E|)$, since all-pairs SimRank runs $|V|$ times of single-source SimRank.

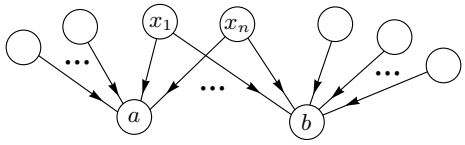


Figure 2: SimRank "Connectivity Trait" Problem

3. ENHANCING SIMRANK QUALITY

After the superfluous ϵ_{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 δ 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 δ:

$$s_\delta(a, b) = \gamma \cdot \frac{|N_a \cap N_b| + \delta}{(|N_a| + \delta)(|N_b| + \delta)} \sim \gamma \cdot \frac{\delta}{\delta^2} \to 0. \quad (\delta \to \infty) \tag{28}$$

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

3.2 Our Kernel-Based SimRank Model

To avoid the order inconsistency between denominator and numerator in Eq.(28), 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*

$$\hat{S}_{a,b} = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \frac{e_a^\top (A^k)^\top A^k e_b}{\|A^k e_a\|_2 \|A^k e_b\|_2}, \tag{29}$$

where $\|x\|_2 := \sqrt{\sum_i |x_i|^2}$ denotes the L_2-norm of vector x.

To prevent division by zero in Eq.(29), 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 weighted cosine similarities between a's and b's multi-hop in-neighbor sets. This can be seen more clearly when we rewrite Eq.(29) as

$$\hat{S}_{a,b} = (1-\gamma)\sum_{k=0}^{\infty}\gamma^k \phi(A^k e_a, A^k e_b) \quad \text{with } \phi(x,y) := \tfrac{x^\top y}{\|x\|_2\|y\|_2}. \quad (30)$$

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.(30) 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.(30) 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 γ^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.(29) can circumvent the SimRank "connectivity trait" problem.*

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

$$e_a^\top (A^k)^\top A^k e_b = |\mathsf{hop}_k(a) \cap \mathsf{hop}_k(b)|,$$
$$\|A^k e_a\|_2 = \sqrt{e_a^\top (A^k)^\top A^k e_a} = \sqrt{|\mathsf{hop}_k(a)|}.$$

Plugging these into Eq.(29) produces

$$\hat{S}_{a,b} = (1-\gamma)\sum_{k=0}^{\infty}\gamma^k \frac{|\mathsf{hop}_k(a) \cap \mathsf{hop}_k(b)|}{\sqrt{|\mathsf{hop}_k(a)| \cdot |\mathsf{hop}_k(b)|}}. \quad (31)$$

When inserting the following δ paths between a and b:

$$a \underbrace{\leftarrow \circ \leftarrow \cdots \leftarrow \circ \leftarrow}_{k_1 \text{ edges}} \boxed{\circ} \underbrace{\rightarrow \circ \rightarrow \cdots \rightarrow \circ \rightarrow}_{k_2 \text{ edges}} b \quad (32)$$

we notice that, only for $k_1 = k_2$, the k_1-th term of the series Eq.(31) will be changed to a function of δ:

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

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

$$f'(\delta) = f(\delta)\big(\tfrac{1}{|\mathsf{hop}_{k_1}(a) \cap \mathsf{hop}_{k_1}(b)|+\delta} - \tfrac{1}{2(|\mathsf{hop}_{k_1}(a)|+\delta)} - \tfrac{1}{2(|\mathsf{hop}_{k_1}(b)|+\delta)}\big).$$

Since $f(\delta) > 0$ and $|\mathsf{hop}_{k_1}(a)| \geq |\mathsf{hop}_{k_1}(a) \cap \mathsf{hop}_{k_1}(b)|$ and $|\mathsf{hop}_{k_1}(b)| \geq |\mathsf{hop}_{k_1}(a) \cap \mathsf{hop}_{k_1}(b)|$, we can obtain $f'(\delta) > 0$. Thus, $f(\delta)$ increases *w.r.t.* δ, which implies that paths (32) insertion will not decrease $\hat{S}_{a,b}$. □

Indeed, by using $Pe_b = Ae_b/\|Ae_b\|_1$ [5] to the original SimRank Eq.(2), we notice that both Eqs.(2) and (29) tally the same paths in-linked from a and b. The difference is norms $\|*\|_2$ and $\|*\|_1$ used by Eq.(29) and Eq.(2)[6], 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.(30).

[5] $\|x\|_1 := \sum_i |x_i|$ denotes the L_1-norm of vector x.
[6] P is associated with $\frac{1}{|N_a||N_b|}$ $(= \frac{1}{\|Pe_a\|_1\|Pe_b\|_1})$ in Eq.(1).

EXAMPLE 2. *Recall the δ paths $\{a \leftarrow x \rightarrow b\}$ to be added into G in Figure 2. After insertion, $\hat{S}_{a,b}(\delta)$ in Eq.(29) can circumvent the "connectivity trait" problem. This is because*

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

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

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

Therefore, it follows from Eq.(29) that

$$\hat{S}_{a,b}(\delta) = (1-\gamma)\gamma \cdot \tfrac{|N_a \cap N_b|+\delta}{\sqrt{|N_a|+\delta}\sqrt{|N_b|+\delta}} \to (1-\gamma)\gamma \quad (\delta \to \infty) \quad (33)$$

Comparing this with Eq.(28), $\hat{S}_{a,b}(\delta)$ is not eventually decreasing w.r.t. δ, which is due to norm $\|*\|_2$ used in Eq.(29). □

In contrast to SimRank++ [1] and PSimRank [2] whose revised weight factors rely only on common N_a and N_b, our method Eq.(29), even if $N_a \cap N_b = \varnothing$, can evaluate $s(a,b)$ from common multi-hops neighbors $\mathsf{hop}_k(a) \cap \mathsf{hop}_k(b)$.

To compute the cosine-based SimRank score $\hat{S}_{a,b}$, if $a = b$, Eq.(29) 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

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

Eqs.(34)–(35) provide an algorithm to compute $\hat{S}_{a,b}^{(k)}$, which is in $O(k|E|)$ time and $O(|E|+k|V|)$ memory for k iterations.

4. SEMANTIC DIFFERENCE

Apart from Jeh and Widom's SimRank model [5]:

$$S = \max\{\gamma P^\top S P,\ I\}, \quad (36)$$

recent years have witnessed many studies (*e.g.,* [3, 4, 9, 14]) to compute similarity, based on Li *et al.*'s model [9]:

$$\tilde{S} = \gamma P^\top \tilde{S} P + (1-\gamma) I. \quad (37)$$

In this section, we explore their relationship from a semantic perspective, and correct two arguments in [9] and [7].

4.1 A Fly in the Ointment of [7, 9]

There are only two works [7, 9] that have mentioned the relationship between $\tilde{S}$ and S. (a) Li *et al.* [9] argued that *"$\tilde{S}$ affects only the absolute similarity value of S, but not the relative similarity ranking of S."* (b) The recent work by Kusumoto *et al.* [7] states that "*$\tilde{S}$ does not much affect the top-K ranking of S.* [7]" However, either of them implies a limitation, as disproved by the following counterexample.

EXAMPLE 3. *Consider graph G in Figure 3, for $\gamma = 0.6$, the top-10 similarity rankings by S and $\tilde{S}$ are shown in part:*

node pairs	$(3,3)$	$(6,6)$	...	$(1,2)$	$(7,8)$
rank by S	*1*	*1*	...	*9*	*9*
rank by $\tilde{S}$	*4*	*3*	...	*10*	*9*

[7] In essence, $S \approx \tilde{S}$ is equivalent to $D \approx (1-\gamma)I$.

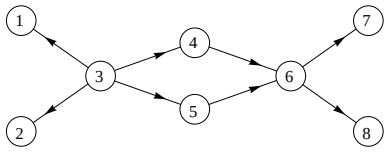


Figure 3: A Citation Graph (A Counterexample)

From the table, we can discern the following:
(a) $\tilde{S}$ does not preserve the relative similarity rankings of S;
(b) At least 4 out of top-10 rankings of S are affected by $\tilde{S}$.
Thus, neither of the statements by [7, 9] is correct. □

4.2 Semantic Relationship Between S and $\tilde{S}$

To "rekindle" the semantic relationship between S and $\tilde{S}$, let us first introduce the following notation:

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

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

This notation is introduced to bring new insights into S.

THEOREM 6. *The similarity S in Jeh and Widom's model Eq.(36) can be characterized as follows:*

$$S = I + \gamma(P^\top P)_{off} + \gamma^2(P^\top(P^\top P)_{off}P)_{off} + \cdots + \gamma^k \underbrace{(P^\top \cdots (P^\top (P^\top P)_{off} P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} + \cdots \quad (38)$$

PROOF. Applying $(*)_{off}$, Eq.(36) can be iterated as

$$S_k = \gamma(P^\top S_{k-1} P)_{off} + I. \quad (39)$$

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

$$R_k = \gamma^k \underbrace{(P^\top \cdots (P^\top (P^\top P)_{off} P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} + R_{k-1}. \quad (40)$$

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

$$\begin{aligned} S_{k+1} &= \gamma(P^\top S_k P)_{off} + I \qquad \text{(using the hypothsis } S_k = R_k\text{)} \\ &= \gamma^{k+1}\big(P^\top \underbrace{(P^\top \cdots (P^\top P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} P\big)_{off} + \underbrace{\gamma(P^\top R_{k-1} P)_{off} + I}_{=\{\text{using Eq.(39)}\}} \\ &= \gamma^{k+1} \underbrace{\big(P^\top (P^\top \cdots (P^\top P)_{off} \cdots P)_{off} P\big)_{off}}_{(k+1) \text{ nested } (*)_{off}} + R_k = R_{k+1}. \end{aligned}$$ □

By Theorem 6, S in Eq.(36) is the weighted sums of

$$\underbrace{(P^\top \cdots (P^\top (P^\top P)_{off} P)_{off} \cdots P)_{off}}_{k \text{ nested } (*)_{off}} \qquad \forall k = 1, 2, \cdots \quad (41)$$

In contrast, $\tilde{S}$ in Eq.(37) is the weighted sums of the terms

$$\underbrace{(P^\top \cdots (P^\top (P^\top P) P) \cdots P)}_{k \text{ nested brackets}} \qquad \forall k = 1, 2, \cdots \quad (42)$$

To find out the semantic difference between S and $\tilde{S}$, we merely need to compare the paths tallied by (41) and (42):

THEOREM 7. *Given a graph G, the terms in Eq.(41) tally the following paths in G:*

$$x_0 \underbrace{\leftarrow x_1 \leftarrow \cdots \leftarrow x_{k-1} \leftarrow}_{k \text{ edges}} \boxed{x_k} \underbrace{\rightarrow x_{k+1} \rightarrow \cdots \rightarrow x_{2k-1} \rightarrow}_{k \text{ edges}} x_{2k} \quad (43)$$

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

Figure 4: Different Paths Tallied by S and $\tilde{S}$

where $x_0, \cdots, x_{2k}$ can be any nodes, but with no repetition of nodes x_i and x_{2k-i} allowed, $\forall i \in \{0, 1, \cdots, 2k\} - \{k\}$.
In comparison, the terms in Eq.(42) tally the paths of (43) in G without having such a constraint on nodes x_i and x_{2k-i}.

PROOF. By the power property of the adjacency matrix, $\big((P^k)^\top P^k\big)_{i,j}$ tallies the paths of (43) between i and j.

To show the terms in Eq.(41) tally the paths of (43) with the additional constraint, we use induction on k as follows.

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

Assume that, for the fixed k, the term

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

tallies the length-$2k$ paths (43) 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 (43), and $x_{2k} \rightarrow j$, which is

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

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

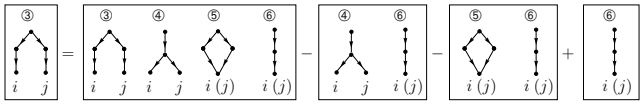
EXAMPLE 4. *Recall the graph in Figure 3. By Theorem 7, the path $7 \leftarrow 6 \leftarrow 5 \leftarrow \boxed{3} \rightarrow 4 \rightarrow 6 \rightarrow 8$ is tallied by the term $\big((P^3)^\top P^3\big)$, but not by $(P^\top(P^\top(P^\top P)_{off}P)_{off}P)_{off}$.*
Indeed, regarding the term $(P^\top(P^\top(P^\top P)_{off}P)_{off}P)_{off}$, the innermost $(*)_{off}$ disallows paths with repetition of nodes 5 and 4; the second nested $(*)_{off}$ disallows the repetition of nodes 6 and 6 (which the considered path violates); the outermost $(*)_{off}$ disallows the repetition of nodes 7 and 8. □

In light of Theorem 7, 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 4 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 $(*)_{off}$ definition to expand, *e.g.,* the term $(P^\top(P^\top P)_{off}P)_{off}$ as follows:

$$\underbrace{\big((P^\top(P^\top P)_{off}P)_{off}\big)_{i,j}}_{③} = \underbrace{\big((P^2)^\top P^2\big)_{i,j}}_{③\,④\,⑤\,⑥} - \underbrace{\big(P^\top diag(P^\top P)P\big)_{i,j}}_{④\,⑥} - \underbrace{\big(diag((P^2)^\top P^2)\big)_{i,j}}_{⑤\,⑥} + \underbrace{\big(diag(P^\top diag(P^\top P)P)\big)_{i,j}}_{⑥}$$

where a circled number beneath each term is associated with a path numbered in the upper-left corner of Figure 4.



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 $2k$ ($k = 1, 2, \cdots$), with first k contiguous edges oriented in one direction, and next k contiguous edges in the opposite direction.*

5. EXPERIMENTAL STUDIES

5.1 Experimental Setting

(1) Real Data. The details are described below:

Dataset	$\|V\|$	$\|E\|$	$\|E\|/\|V\|$	Type
WikiV	7,115	103,689	14.57	Directed
CaD	15,683	55,064	5.31	Undirected
CitH	34,546	421,578	12.20	Directed
WebN	325,729	1,497,134	4.59	Directed
ComY	1,134,890	2,987,624	2.63	Undirected
SocL	4,847,571	68,993,773	14.23	Directed

(a) WikiV, a Wikipedia who-votes-on-whom graph[8], where nodes are users, and an edge $i \to j$ means user i voted on j.

(b) CaD, a collaboration graph, where each node is an author, and edges co-authorships. The graph is derived from 6-year publications (2006–2011) in seven major conferences.

(c) CitH, a citation graph from arXiv high energy physics theory, where each node is a paper labeled with meta information (*e.g.,* title, authors, abstract) and an edge a citation.

(d) WebN, a web graph from University of Notre Dame, where a node is a page (from `nd.edu`) and an edge a link.

(e) ComY, an undirected Youtube social graph, where a node is a user and an edge a friendship.

(f) SocL, a friendship graph of a LiveJournal community, where $i \to j$ is a recommendation of user j from user i.

(2) Synthetic Data. To produce SYN, we adopt a scale-free graph generator based on the Barabasi-Albert model[9]. This generator takes as input two parameters: $(|V|, |E|)$.

(3) Query Generator. (i) To evaluate all-pairs $s(*,*)$, we generate the query-pair set (A, B), by using 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) has both important and non-important pairs.

(b) *Overlapping coverage* is to ensure 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:[10] $f_{a,b} := \sum_{k=1}^{5} \frac{|\mathsf{hop}_k(a) \cap \mathsf{hop}_k(b)|}{|\mathsf{hop}_k(a) \cup \mathsf{hop}_k(b)|}$. We then split all pairs into 5 buckets: pairs with $f_{a,b} \in [4, 5]$ 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 $g_{a,b} := \sum_{k=1}^{5} |\mathsf{hop}_k(a) \cap \mathsf{hop}_k(b)|$, 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".

[8] http://snap.stanford.edu/data/index.html

[9] http://graphstream-project.org/doc/Generators/

[10] All paths of length up to 10 between a and b can be tallied by our queries, ensuring results accurate to 2 decimal places.

(4) Algorithms. We implement the following, all in VC++.

Name	Description
SR#	our scheme ("cosine" kernel + computation sharing)
MSR	the state-of-the-art SimRank [7]
OIP	all-pairs SimRank (fine-grained clustering) [13]
PSUM	all-pairs SimRank (partial sums memoization) [12]
SMAT	single-source SimRank (matrix decomposition) [3]
JSR	Jeh and Widom's SimRank [5]
LSR	Li *et al.*'s SimRank [9]
SR++	SimRank++ (revised "evidence factor") [1]
RS	RoleSim (automorphism equivalence) [6]
RWR	Random Walk with Restart
COS	classic cosine similarity

(5) Parameters. We set (a) $\gamma = 0.6$, as suggested in [12]. (b) $k = 10$, guaranteeing $S^{(k)}$ accurate to 2 decimal places.

(6) Evaluation Metrics. To evaluate the semantic quality of the similarity search, we consider four metrics:

(a) Normalized Discounted Cumulative Gain at position p: $\text{NDCG}_p := \frac{1}{\text{IDCG}_p} \sum_{i=1}^{p} \frac{2^{\mathsf{rel}_i} - 1}{\log_2(1+i)}$, where rel_i is the graded relevance at position i, and IDCG_p is the ideal DCG ranking.

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

(c) Kendall's $\tau := \frac{(\#\text{ of concordant pairs}) - (\#\text{ of discordant pairs})}{0.5n(n-1)}$.

(d) Query coverage is the queries from our query sample.

(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 Co-Author Path in Microsoft Academic Search.[11] (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 32GB RAM, on Windows 7.

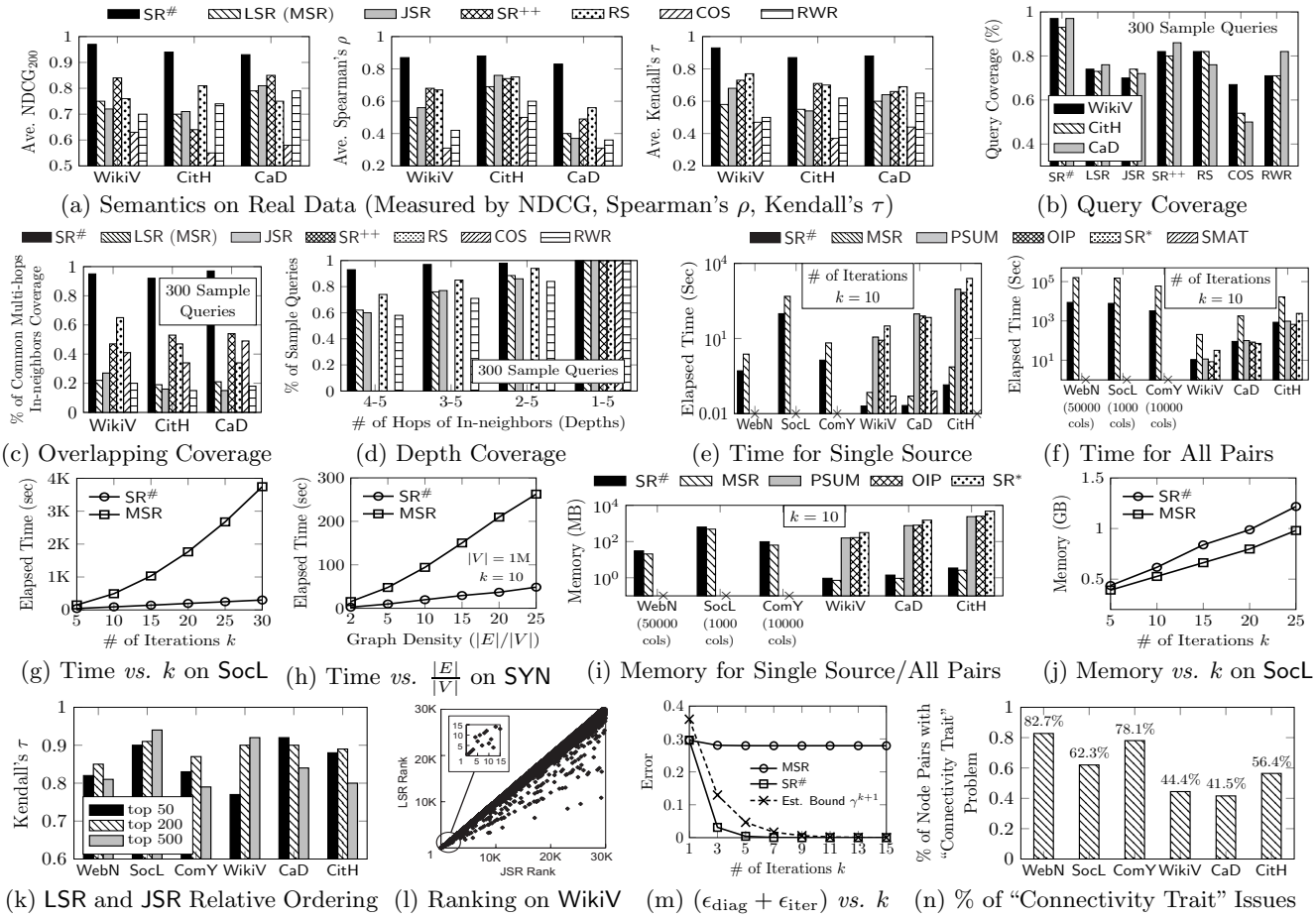
5.2 Experimental Results

5.2.1 Semantic Quality

We first evaluate the high semantic quality of SR# against SR++, JSR, LSR[12], RS, COS, RWR 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. Fig. 5a 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_{200} of SR++ (on CitH) and RS (on CaD) may even worse than that of JSR and LSR. This is because, for SR++, its evi-

[11] http://academic.research.microsoft.com/VisualExplorer

[12] For semantics evaluation, MSR produces the same similarity values as LSR, since [7] approximates D by $(1-\gamma)I$.



(a) Semantics on Real Data (Measured by NDCG, Spearman's ρ, Kendall's τ) (b) Query Coverage

(c) Overlapping Coverage (d) Depth Coverage (e) Time for Single Source (f) Time for All Pairs

(g) Time *vs.* k on SocL (h) Time *vs.* $\frac{|E|}{|V|}$ on SYN (i) Memory for Single Source/All Pairs (j) Memory *vs.* k on SocL

(k) LSR and JSR Relative Ordering (l) Ranking on WikiV (m) $(\epsilon_{\text{diag}} + \epsilon_{\text{iter}})$ *vs.* k (n) % of "Connectivity Trait" Issues

Figure 5: Performance Evaluations on Real and Synthetic Datasets

dence 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.

Fig. 5b compares the percentage of queries from the 300 queries sample (based on *importance coverage* criterion) that SR$^{\#}$, SR^{++}, JSR, LSR, RS, COS, RWR provide similarities for on real WikiV, CitH, CaD, respectively. For each dataset, SR$^{\#}$ substantially improves the coverage of JSR/LSR ($\sim$0.73) and SR^{++}/RS ($\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 high connectivity, and (b) SR^{++} can only partially fix the "connectivity trait" issue within 1-hop in-neighborhood.

Fig. 5c depicts the percentage of queries with overlapped multi-hop in-neighbors from the 300 queries sample (via *overlapping coverage* criterion) that SR$^{\#}$, SR^{++}, JSR, LSR, RS, COS, RWR are able to identify on real WikiV, CitH, CaD, respectively. (1) For each dataset, SR$^{\#}$ significantly achieves $\sim$0.95 coverage of common multi-hop in-neighbors, much superior to JSR/LSR ($\sim$0.20), SR^{++}($\sim$0.51) and COS($\sim$0.41). 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$^{\#}$, SR^{++}, JSR, LSR, RS, COS, RWR, 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. Fig. 5d depicts the search depth of all the methods on WikiV. (1) For each group, SR^{++} 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$^{\#}$ 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.

5.2.2 Time Efficiency

Fig. 5e 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.

Fig. 5f shows the time of SR$^{\#}$, MSR, PSUM, OIP, SR* for all-pairs $s(*, *)$ on 6 real datasets. (1) Only SR$^{\#}$, MSR survive on all datasets, whereas PSUM, OIP, SR* fail on large

WebN, SocL, ComY, due to the memory allocation. (2) MSR is slower than others as it sacrifices speed for scalability. In contrast, SR# not only scales well on large graphs, but also has comparable speed to those of PSUM, OIP, SR*.

Fig. 5g presents the impact of the iteration number 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.

Fig. 5h 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.

5.2.3 Memory Efficiency

Fig. 5i shows the memory of SR#, MSR, PSUM, OIP, SR* on six real datasets. (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 D_k.

Fig. 5j reports the impact of the iteration number k on the memory of SR# and MSR on SocL. (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.

5.2.4 Relative Order

Fig. 5k 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 τ. (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 τ 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 Fig. 5l, where x (y) axis is the ranking by JSR (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.

Fig. 5m tests ($\epsilon_{\text{diag}} + \epsilon_{\text{iter}}$) of MSR and SR# *w.r.t.* k. (1) When k increases from 1 to 15, the error of each algorithm decreases. While the error of 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. (2) The SR# curve is always below the Est. Bound curve, showing the correctness of Theorem 4.

Fig. 5n 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.

6. CONCLUSIONS

We consider the problem of high-quality similarity search. Observing that (1) the best-of-breed SimRank [7] produces diagonal correction error ϵ_{diag} and (2) SimRank++ [1] does not well resolve the "connectivity trait" problem, we proposed our scheme, SR#. First, we characterize the exact D, and devise a "varied-D" model to compute SimRank with no ϵ_{diag} in linear memory. We also speed up the computational time from quadric [7] to linear in terms of k. Next, we devise a "kernel-based" model to circumvent the "connectivity trait" problem. Finally, we give new insights into the semantic difference between Jeh and Widom's SimRank and its variant, and correct an argument in [7]. We empirically show that SR# (1) improves an accuracy of average NDCG_{200} by ∼30% on real graphs, and (2) can be ∼10x faster than [7] on SocL with 65.8M links for 1000 queries.

Acknowledgment. This work forms part of the Big Data Technology for Smart Water Network research project funded by NEC Corporation, Japan.

7. 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] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, 2005.
[3] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, and M. Onizuka. Efficient search algorithm for SimRank. In *ICDE*, 2013.
[4] G. He, H. Feng, C. Li, and H. Chen. Parallel SimRank computation on large graphs with iterative aggregation. In *KDD*, 2010.
[5] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *KDD*, 2002.
[6] R. Jin, V. E. Lee, and H. Hong. Axiomatic ranking of network role similarity. In *KDD*, 2011.
[7] M. Kusumoto, T. Maehara, and K. Kawarabayashi. Scalable similarity search for SimRank. In *SIGMOD*, 2014.
[8] P. Lee, L. V. S. Lakshmanan, and J. X. Yu. On top-k structural similarity search. In *ICDE*, 2012.
[9] 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.
[10] P. Li, H. Liu, J. X. Yu, J. He, and X. Du. Fast single-pair SimRank computation. In *SDM*, 2010.
[11] Z. Lin, M. R. Lyu, and I. King. MatchSim: A novel similarity measure based on maximum neighborhood matching. *Knowl. Inf. Syst.*, 32(1), 2012.
[12] D. Lizorkin, P. Velikhov, M. N. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for SimRank computation. *VLDB J.*, 19(1), 2010.
[13] W. Yu, X. Lin, and W. Zhang. Towards efficient SimRank computation on large networks. In *ICDE*, 2013.
[14] W. Yu, X. Lin, and W. Zhang. Fast incremental simrank on link-evolving graphs. In *ICDE*, pages 304–315, 2014.
[15] W. Yu, X. Lin, W. Zhang, Y. Zhang, and J. Le. SimFusion+: Extending SimFusion towards efficient estimation on large and dynamic networks. In *SIGIR*, 2012.
[16] W. Yu and J. A. McCann. Sig-SR: SimRank search over singular graphs. In *SIGIR*, 2014.
[17] W. Yu and J. A. McCann. Efficient partial-pairs SimRank search for large networks. *PVLDB*, 8(5):569–580, 2015.