More is Simpler: Effectively and Efficiently Assessing Node-Pair Similarities Based on Hyperlinks

Weiren Yu[†♯], Xuemin Lin[♭†], Wenjie Zhang[†], Lijun Chang[†], Jian Pei[♮]

[†]The University of New South Wales, Australia [♭]East China Normal University, China
[♯]NICTA, Australia [♮]Simon Fraser University, Canada

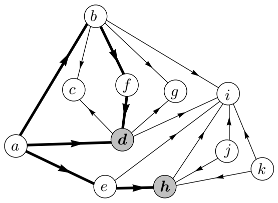
{weirenyu, lxue, zhangw, ljchang}@cse.unsw.edu.au jpei@cs.sfu.ca

ABSTRACT

Similarity assessment is one of the core tasks in hyperlink analysis. Recently, with the proliferation of applications, *e.g.,* web search and collaborative filtering, SimRank has been a well-studied measure of similarity between two nodes in a graph. It recursively follows the philosophy that "two nodes are similar if they are referenced (have incoming edges) from similar nodes", which can be viewed as an aggregation of similarities based on incoming paths. Despite its popularity, SimRank has an undesirable property, *i.e.,* "zero-similarity": It only accommodates paths with *equal* length from a common "center" node. Thus, a large portion of other paths are fully ignored. This paper attempts to remedy this issue. (1) We propose and rigorously justify SimRank*, a revised version of SimRank, which resolves such counter-intuitive "zero-similarity" issues while inheriting merits of the basic SimRank philosophy. (2) We show that the series form of SimRank* can be reduced to a fairly succinct and elegant closed form, which looks even simpler than SimRank, yet enriches semantics without suffering from increased computational cost. This leads to a fixed-point iterative paradigm of SimRank* in $O(Knm)$ time on a graph of n nodes and m edges for K iterations, which is comparable to SimRank. (3) To further optimize SimRank* computation, we leverage a novel clustering strategy via edge concentration. Due to its NP-hardness, we devise an efficient and effective heuristic to speed up SimRank* computation to $O(Kn\tilde{m})$ time, where $\tilde{m}$ is generally much smaller than m. (4) Using real and synthetic data, we empirically verify the rich semantics of SimRank*, and demonstrate its high computation efficiency.

1. INTRODUCTION

The task of assessing similarity between two nodes based on hyperlinks is a long-standing problem in information search. This type of similarity, also known as *link-based similarity*, is one of the fundamental primitives for hyperlink analysis in a graph, with a broad range of applications, *e.g.,* collaborative filtering [1], web page ranking [10], and graph clustering [24]. Intuitively, link-based similarity assessment aims to assign a relevance score to each node-pair based purely on the structure of a network, in contrast to *text-based similarity* that relies on the content of the Web. However, it is a complex challenge to find an appropriate link-based scoring function since a satisfactory general-purpose similarity measure should better simulate human judgement behavior, with simple and elegant formulations [17].



Node-Pairs	SR	PR	SR*	RWR
(h,d)	0	.049	.010	0
(a,f)	0	.075	.032	.032
(a,c)	0	0	.025	.024
(g,a)	0	0	.025	0
(g,b)	0	0	.075	0
(i,a)	0	0	.015	0
(i,h)	.044	.041	.031	0

Figure 1: Similarities on Citation Graph

Recently, SimRank [9] has received growing interest as a widely-accepted measure of similarity between two nodes. The triumph of SimRank is largely attributed to its succinct yet elegant philosophy: *Two nodes are similar if they are referenced by similar nodes.* The base case for this recursion is that each node is maximally similar to itself. SimRank was proposed by Jeh and Widom [9], and has gained tremendous popularity in many vibrant communities, *e.g.,* recommender systems [1], citation analysis [8], and k-nearest neighbor search [12]. Due to its self-referentiality, conventional methods for computing SimRank are iterative in nature. The state-of-the-art algorithm [17] needs $O(Knm)$ time on a graph of n nodes and m edges for K iterations.

While significant efforts have been devoted to optimizing SimRank computation (*e.g.,* [7,8,14,17]), the semantic issues of SimRank have attracted little attention. We observe that SimRank has an undesirable property, namely, "zero-similarity": SimRank score $s(i,j)$ only accommodates the paths with *equal length* from a common "source" node to both i and j. Thus, other paths for node-pair (i,j) are fully ignored by SimRank. as shown in Example 1.

Example 1. *Consider a citation network $\mathcal{G}$ in Figure 1, where each node represents a paper, and an edge a citation. Using the damping factor $C = 0.8$* [1]*, we compute SimRank similarity of node-pairs in $\mathcal{G}$. It can be noticed that many node-pairs in $\mathcal{G}$ have zero SimRank when they have no incoming paths of equal length from a common "source" node, as partly depicted in Column 'SR' of the table. For instance, $s(h,d) = 0$ as the in-link "source" a is not in the center of*

[1] As suggested in [9], C is empirically set around 0.6–0.8, which gives the rate of decay as similarity flows across edges.

Articles from this volume were invited to present their results at The 40th International Conference on Very Large Data Bases, September 1st - 5th 2014, Hangzhou, China.
Proceedings of the VLDB Endowment, Vol. 7, No. 1
Copyright 2013 VLDB Endowment 2150-8097/13/09... $ 10.00.

the paths: $h \leftarrow e \leftarrow \boxed{a} \rightarrow d$ [2], $h \leftarrow e \leftarrow \boxed{a} \rightarrow b \rightarrow f \rightarrow d$, *meaning that when we recursively compute the similarity of the in-neighbors prior to computing the similarity of the two nodes themselves, there is no likelihood for this recursion to reach the base case (a common in-link "source") that a node is maximally similar to itself. Similarly,* $s(a, g) = 0$ *as* a *has no in-neighbors, not to mention the fact that there is no such in-link "source" with equal distance to both* a *and* g*. In contrast,* $s(g, i) > 0$ *as there is an in-link "source"* b *(resp.* d*) in the center of* $g \leftarrow \boxed{b} \rightarrow i$ *(resp.* $g \leftarrow \boxed{d} \rightarrow i$*).* □

The "zero-SimRank" phenomenon in Example 1 is rather counter-intuitive. An evident example is $s(h, d) = 0$. We note in Figure 1 that h and d do have a common in-link "source" a, just except for the *equal-length* distance from a to both h and d. Hence, h and d should have some relevance. Another example is a path graph of length $2n$ as follows: $a_{-n} \leftarrow \cdots \leftarrow a_{-1} \leftarrow \boxed{a_0} \rightarrow a_1 \rightarrow \cdots \rightarrow a_n$, where each a_i $(i = 0, \pm 1, \cdots, \pm n)$ denotes a node. We notice that the SimRank $s(a_i, a_j) = 0$, for all $|i| \neq |j|$, which is quite against intuition since a_0 is the common root of all nodes a_i $(i = \pm 1, \cdots, \pm n)$. As will be shown in Section 3, SimRank does neglect all contributions of in-link paths without a "source" node in the center, and the "zero-similarity" issue refers not only to the problem that SimRank may produce "completely zero scores" *(i.e., "completely dissimilar" issue)*, but also to the problem that SimRank may miss the contributions of a large class of in-link paths (even though their scores are not zero) due to the "zero contributions" of such paths to SimRank scores *(i.e., "partially missing" issue)*. Indeed, as demonstrated by our experiments in Fig.6(d), both scenarios of "zero-similarity" *commonly* exist in real graphs, *e.g.,* on CitHepTh, 95+% node-pairs have "zero-SimRank" issues, among which 40+% are assessed as "completely dissimilar", and 55+% (though SimRank ≠0) "partially miss" contributions of many paths, adversely affecting assessment quality. These motivate us to revise the existing SimRank model.

A pioneering piece of work by Zhao *et al.* [23] proposes rudiments of a novel approach to refining the SimRank model. Observing that SimRank may incur some unwanted "zero-similarities", they suggested P-Rank, an extension of SimRank, by taking both in- and out-links into consideration for similarity assessment, as opposed to SimRank that merely considers in-links. Although P-Rank, to some degree, might reduce "zero-similarity" occurrences in practice, we argue that such a "zero-similarity" issue arises, not because of a biased overlook of SimRank against out-links, but because of the blemish in SimRank philosophy that may miss the contribution of a certain kind of paths (whose in-link "source" is not in the center). In other words, P-Rank can not, in essence, resolve the "zero-similarity" issue of SimRank. For instance, nodes h and d are similar in the context of P-Rank, as depicted in Col. 'PR' of Fig. 1, since there is an out-link "source" i in the center of the outgoing path $h \rightarrow \boxed{i} \leftarrow d$. However, if the edge $h \rightarrow i$ is replaced by $h \rightarrow l \rightarrow i$ with l being an inserted node, then the P-Rank of (h, d) is still zero, since in this case neither in- nor out-link "source" exists in the center of any incoming or outgoing paths of (h, d).

Our goal in this work is to propose an alternative model that can remedy SimRank "zero-similarity" issues in nature, while inheriting merits of the basic SimRank philosophy. Keeping with an elegant form and to support fast clustering strategies, our model is intended to be a refinement of SimRank for semantic richness, and takes into account contributions of many incoming paths (whose common "source" is not strictly in the center) that are neglected by SimRank. The major challenge with establishing this model is that it is notoriously difficult to effectively assess $s(a, b)$ by finding out *all* the possible incoming paths between a and b, regardless of whether there exists a common "source" with equal distance to both a and b. This problem is hard because such a task often requires traversing far more possible incoming paths to fetch the similarity information, which might not only destroy the simplicity of the original SimRank formulation, but also increase the computational difficulty of the model. Fortunately, we observe that our model can be "purified" as a fairly elegant closed form, and there are opportunities for the new model to assess similarities without suffering from high computational costs.

Contributions. Our main contributions are as follows.

- We propose SimRank*, a revision of SimRank, and justify its semantic richness. Our model provides a natural way of traversing more incoming paths that are largely ignored by SimRank for each node-pair, and thus enables counter-intuitive "zero-SimRank" nodes to be similar while inheriting the beauty of the SimRank philosophy. (Section 3)
- We show that the series form of SimRank* can be simplified into an elegant closed form, which looks more succinct yet has richer semantics than SimRank, without suffering from increased computational cost. This provides an iterative paradigm for computing SimRank* in $O(Knm)$ time on a graph of n nodes and m edges for K iterations, which is comparable to SimRank. (Subsects. 4.1–4.2)
- To further speed up SimRank* computation, as the existing technique [17] of partial sums memoization for SimRank optimization no longer applies, we leverage a novel clustering approach for SimRank* via edge concentration. Due to its NP-hardness, an efficient algorithm is devised to improve SimRank* computation to $O(Kn\tilde{m})$ time, where $\tilde{m}$ is generally much smaller than m. (Subsect. 4.3)

We evaluate the performance of SimRank* on real and synthetic data. The results show that (i) SimRank* achieves higher quality of similarity assessment, as compared with the state-of-the-art SimRank [17], P-Rank [23] and RWR [19]; (ii) Regarding computational efficiency, our algorithms are consistently faster than the baselines by several times.

Related Work. We categorize related work as follows.

Link-based Similarity. One of the most renowned link-based similarity metrics is SimRank, invented by Jeh and Widom [9]. It iteratively captures the notion that "two nodes are similar if they have *similar* in-neighbors", which weakens the philosophy of the rudimentary measures (*e.g.,* Coupling [11], Co-citation [18]) that "two nodes are similar if they have the *same* neighbors in common". The recursive nature of SimRank allows *two* nodes to be similar without common in-neighbors, which resembles PageRank [2] assigning a relevance score for *each* node. SimRank implies an unsatisfactory trait: The similarity of two nodes decreases as the number of their common in-neighbors increases. To address this issue, Fogaras and Rácz [7] introduce P-SimRank. They (1) incorporate Jaccard coefficients, and (2) interpret $s(a, b)$ as the probability that two random surfers, starting from a and b, will meet at a node. Antonellis *et al.* [1] propose SimRank++, by adding an evidence weight to compensate for the cardinality of in-neighbor matching. MatchSim [16]

[2] We abuse the notation $h \leftarrow e \leftarrow \boxed{a} \rightarrow d$ to denote the path of length 3, starting from h, taking 2 steps against the edge direction and 1 step along it, and finally arriving at d.

refines SimRank with maximum neighborhood matching. RoleSim [10] deploys generalized Jaccard coefficients to ensure automorphic equivalence for SimRank. However, none of them resolves the "zero-SimRank" issue. This issue surfaces in part in the motivating Example 1.2 of Zhao *et al.* [23] who propose P-Rank taking both in- and out-links into account. Our work differs from [23] in that (1) we show that the "zero-SimRank" issue is not caused by the ignorance of out-links in SimRank, and (2) we circumvent the "zero-similarity" issue by traversing more incoming paths of node-pairs that are neglected by the original SimRank.

There has also been work on link-based similarity (*e.g.,* [3, 13,19–21]). LinkClus [21] uses a hierarchical structure, called SimTree, for clustering multi-type objects. Blondel *et al.* [3] propose an appealing measure to quantify *graph* similarities. SimFusion [20] utilizes a reinforcement assumption for assessing similarities of multi-type objects in a heterogenous domain, as opposed to SimRank focusing solely on intra-type objects in a homogenous domain. Tong *et al.* [19] suggest Random Walk with Restart (RWR) for assessing node proximities, which is an excellent extension of Personalized PageRank (PPR). Leicht *et al.* [13] extend RWR by incorporating independent and sensible coefficients. However, RWR and its variants (PPR and [13]) also imply SimRank-like "zero-similarity" issues, as discussed in Subsect. 3.1.

Similarity Computation. The computational overheads of link-based similarity often arise from its recursive nature. To meet this challenge, Lizorkin *et al.* [17] propose three excellent optimization methods to SimRank (*i.e.,* essential node-pair selection, partial sums memoization, and threshold-sieved similarities). These substantially speed up SimRank computation from $O(Kd^2n^2)$ to $O(Knm)$ time, with d being the average in-degree of a graph. In contrast, our model performs even faster than SimRank, yet can enumerate more incoming paths missed by SimRank to enrich semantics since (1) our model can be simplified into a much simpler form than SimRank, and (2) the computation can be further accelerated via fine-grained memoization. Li *et al.* [14] use graph low-rank structure to compute SimRank via singular value decomposition (SVD), yielding $O(r^4n^2)$ time, with r ($\leq n$) being the rank of an adjacency matrix. However, it does not always reduce the complexity when r is large. In contrast, SimRank* needs $O(Kn\tilde{m})$ worst-case time, with $\tilde{m} \leq m$. He *et al.* [8] study the incremental SimRank with the focus on node updates for parallel computing on GPU.

2. PRELIMINARY

Below we briefly revisit two representations of SimRank: (1) the iterative form [9,17], and (2) the matrix form [8,14].

(1) Iterative Form. For a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes in $\mathcal{V}$ and edges in $\mathcal{E}$, let $\mathcal{I}(a)$ be the in-neighbor set of a, and $|\mathcal{I}(a)|$ the cardinality of $\mathcal{I}(a)$, then *the SimRank similarity* between nodes a and b, denoted as $s(a,b)$, is defined by (i) $s(a,b) = 0$, if $\mathcal{I}(a) = \varnothing$ or $\mathcal{I}(b) = \varnothing$; (ii) otherwise,

$$s(a,b) = \begin{cases} 1, & a = b; \\ \frac{C}{|\mathcal{I}(a)||\mathcal{I}(b)|} \sum_{j \in \mathcal{I}(b)} \sum_{i \in \mathcal{I}(a)} s(i,j), & a \neq b. \end{cases} \quad (1)$$

where $C \in (0,1)$ is a damping factor.

To solve $s(a,b)$, one can carry out the following iterations. (1) Start with $s_0(a,a) = 1$ and $s_0(a,b) = 0$ if $a \neq b$. (2) For $k = 0, 1, 2, \cdots$, iterate as indicated below: (i) $s_{k+1}(a,b) = 0$, if $\mathcal{I}(a) = \varnothing$ or $\mathcal{I}(b) = \varnothing$; (ii) otherwise,

$$s_{k+1}(a,b) = \begin{cases} 1, & a = b; \\ \frac{C}{|\mathcal{I}(a)||\mathcal{I}(b)|} \sum_{j \in \mathcal{I}(b)} \sum_{i \in \mathcal{I}(a)} s_k(i,j), & a \neq b. \end{cases} \quad (2)$$

The resulting sequence $\{s_k(a,b)\}_{k=0}^{\infty}$ converges to $s(a,b)$.

(2) Matrix Form. SimRank can be rewritten as

$$\mathbf{S} = C \cdot (\mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T) + (1 - C) \cdot \mathbf{I}_n, \quad (3)$$

where $\mathbf{S}$ is the *similarity matrix* whose entry $[\mathbf{S}]_{i,j}$ denotes SimRank score $s(i,j)$, $\mathbf{Q}$ is the *backward transition matrix* whose entry $[\mathbf{Q}]_{i,j} = 1/|\mathcal{I}(i)|$ if there is an edge from j to i, and 0 otherwise, $\mathbf{Q}^T$ denotes the transpose of matrix $\mathbf{Q}$.

Here, $\mathbf{I}_n$ is an $n \times n$ identity matrix. The term $(1-C) \cdot \mathbf{I}_n$ in Eq.(3) allows all diagonal entries of $\mathbf{S}$ being maximal, guaranteeing that each node is maximally similar to itself, which corresponds to the base case for $a = b$ in Eq.(1).

3. SIMRANK*: A REVISION OF SIMRANK

We first show that the "zero-similarity" issue (Example 1) is *rooted* in both SimRank and non-SimRank based metrics. We then propose our treatment, SimRank*, for this issue.

3.1 "Zero-SimRank" Issue

We shall abuse the following notions. (i) An *in-link path* ρ of node-pair (a,b) in $\mathcal{G}$ is a walk of length $(l_1 + l_2)$, denoted as $a = v_0 \leftarrow v_1 \leftarrow \cdots \leftarrow \boxed{v_{l_1}} \rightarrow v_{l_1+1} \rightarrow \cdots \rightarrow v_{l_1+l_2} = b$,[3] starting from a, taking l_1 steps against the directions of the edges $v_{i-1} \leftarrow v_i$ for every $i \in [1, l_1]$, and l_2 steps along the directions of $v_{i-1} \rightarrow v_i$ for every $i \in [l_1 + 1, l_1 + l_2]$, and finally arriving at b. (ii) The node v_{l_1} is called the *in-link "source"* of ρ. (iii) The *length* of in-link path ρ, denoted by $\mathsf{len}(\rho)$, is $(l_1 + l_2)$, *i.e.,* the number of edges in ρ.

Definition 1. *An in-link path ρ is* symmetric *if $l_1 = l_2$.*

For example in Figure 1, $\rho : h \leftarrow e \leftarrow \boxed{a} \rightarrow d$ is an in-link path of node-pair (h,d), with a being its in-link "source". $\mathsf{len}(\rho) = 2 + 1 = 3$. ρ is not symmetric since $l_1 = 2 \neq 1 = l_2$.

Clearly, in-link path ρ is *symmetric* if and only if there is an in-link "source" in the center of ρ. Any in-link path of *odd* length (*i.e.,* $l_1 + l_2$ is odd) is dissymmetric.

"Zero-SimRank" Issue. Based on the notion of in-link paths, we next show the "zero-SimRank" issue as follows:

Theorem 1. *For any two distinct nodes a and b in $\mathcal{G}$, the SimRank score $s(a,b) = 0$ if there does not exist any symmetric in-link path of node-pair (a,b). More importantly, even if $s(a,b) \neq 0$, SimRank $s(a,b)$ may still "partially miss" all the contributions of dissymmetric in-link paths for (a,b).*

As a proof of the theorem, we first extend the power property of an adjacency matrix. We then reinterpret SimRank based on its power series representation.

Extension of $\mathbf{A}^l$. Let $\mathbf{A}$ be the adjacency matrix of $\mathcal{G}$. There is an interesting property of $\mathbf{A}^l$ [4]: The entry $[\mathbf{A}^l]_{i,j}$[4] counts the number of paths of length l from node i to j. Such a property can be readily generalized as follows:

Lemma 1. *Let ρ be a "specific path" of length l, consisting of a sequence of nodes $i = v_0, v_1, \cdots, v_l = j$ with each edge being directed (1) from v_{k-1} to v_k, or (2) from v_k to v_{k-1}. Let $\bar{\mathbf{A}} = \prod_{k=1}^{l} \mathbf{A}_k$ with (1) $\mathbf{A}_k = \mathbf{A}$ if $\exists\ v_{k-1} \rightarrow v_k$ in ρ, or (2) $\mathbf{A}_k = \mathbf{A}^T$ if $\exists\ v_{k-1} \leftarrow v_k$ in ρ, for each $k \in [1, l]$. Then, the entry $[\bar{\mathbf{A}}]_{i,j}$ counts the number of specific paths ρ in $\mathcal{G}$.*

Lemma 1 can be proved by induction on l, which is similar to the proof of the power property of the adjacency matrix [4, pp.51]. We omit it here due to space limits.

[3] We allow a path from the "source" node to one end with repeated nodes to suit the existence of cycles in a graph.

[4] In the sequel, $[\mathbf{X}]_{i,j}$ denotes the (i,j)-entry of matrix $\mathbf{X}$.

Lemma 1 allows counting the number of "specific paths" whose edges are not all necessarily in the same direction. For instance, for the path $\rho : i \to \circ \leftarrow \circ \to \circ \to \circ \leftarrow j$ with $\circ$ denoting any node in $\mathcal{G}$, we can build $\bar{\mathbf{A}} = \mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{A}\mathbf{A}^T$, in which $\mathbf{A}$ (*resp.* $\mathbf{A}^T$) is at the positions 1,3,4 (*resp.* 2,5), corresponding to the positions of $\to$ (*resp.* $\leftarrow$) in ρ. Then, $[\bar{\mathbf{A}}]_{i,j}$ tallies the number of paths ρ in $\mathcal{G}$. If no such paths, $[\bar{\mathbf{A}}]_{i,j} = 0$. As another example, $[(\mathbf{A}^T)^{l_1} \cdot \mathbf{A}^{l_2}]_{i,j}$ tallies the number of in-link paths of length $(l_1 + l_2)$ for node-pair (i, j).

When all $\mathbf{A}_k$ $(\forall k \in [1, l])$ are set to $\mathbf{A}$, Lemma 1 reduces to the conventional power property of an adjacency matrix.

One immediate consequence of Lemma 1 is as follows:

Corollary 1. $\sum_{k=1}^{\infty} [(\mathbf{A}^T)^k \cdot \mathbf{A}^k]_{i,j}$ *counts the total number of all symmetric in-link paths of node-pair* (i, j) *in* $\mathcal{G}$.

Corollary 1 implies that if there are no nodes with equal distance to both i and j (*i.e.,* if no symmetric in-link paths for node-pair (i, j)), then $[(\mathbf{A}^T)^k \cdot \mathbf{A}^k]_{i,j} = 0$, $\forall k \in [1, \infty)$.

SimRank Reinterpretation. Leveraging Corollary 1, we show why SimRank has "zero-similarity" issue: $s(i, j) = 0$ if there are no nodes with equal distance to both i and j.

We first rewrite SimRank matrix $\mathbf{S}$ as a power series.

Lemma 2. *The SimRank* $\mathbf{S}$ *in Eq.*(3) *can be rewritten as*

$$\mathbf{S} = (1 - C) \cdot \sum_{l=0}^{\infty} C^l \cdot \mathbf{Q}^l \cdot (\mathbf{Q}^T)^l. \qquad (4)$$

Proof. According to [14, Eq.(4)], $\mathbf{S}$ has the closed form:

$$vec(\mathbf{S}) = (1 - C) \cdot (\mathbf{I}_n - C(\mathbf{Q} \otimes \mathbf{Q}))^{-1} vec(\mathbf{I}_n),$$

where $vec(\star)$ is a vectorization operator, $\otimes$ a tensor product.

Since $\|\mathbf{Q}\otimes\mathbf{Q}\|_\infty \le 1$, the identity $(\mathbf{I}_n - \mathbf{X})^{-1} = \sum_{k=0}^{\infty} \mathbf{X}^k$ implies that $vec(\mathbf{S}) = (1 - C) \cdot \sum_{k=0}^{\infty} C^k (\mathbf{Q} \otimes \mathbf{Q})^k \cdot vec(\mathbf{I}_n)$. Then, using tensor product properties $(\mathbf{Q} \otimes \mathbf{Q})^k = \mathbf{Q}^k \otimes \mathbf{Q}^k$ and $(\mathbf{Y} \otimes \mathbf{Z}) \cdot vec(\mathbf{I}_n) = vec(\mathbf{Z} \cdot \mathbf{I}_n \cdot \mathbf{Y}^T)$ with $\mathbf{Y} = \mathbf{Z} = \mathbf{Q}^k$, plus the linearity of $vec(\star)$, we can derive Eq.(4). □

The term $(1 - C)$ in Eq.(4) aims to normalize similarities in $[0, 1]$ as $\| \sum_{l=0}^{\infty} C^l \cdot \mathbf{Q}^l \cdot (\mathbf{Q}^T)^l \|_{\max} \le \sum_{l=0}^{\infty} C^l = \frac{1}{1-C}$. [5]

Lemma 2 reformulates SimRank in the form of weight sum of all *symmetric* in-link paths of length $2l$ for node-pair (i, j). To clarify this, as $\mathbf{Q}$ is the weighted (*i.e.,* row-normalized) matrix of $\mathbf{A}^T$, Lemma 2 implies that $[\mathbf{Q}^l \cdot (\mathbf{Q}^T)^l]_{i,j}$ can tally *the weight sum* (instead of *the number*) of in-link paths of length $2l$ for node-pair (i, j). Formally, we state this below:

Corollary 2. $[\mathbf{Q}^l \cdot (\mathbf{Q}^T)^l]_{i,j} = 0 \Leftrightarrow [(\mathbf{A}^T)^l \cdot \mathbf{A}^l]_{i,j} = 0.$

This, together with the component form of Eq.(4), *i.e.,*

$$[\mathbf{S}]_{i,j} = (1 - C) \cdot \sum_{l=0}^{\infty} C^l \cdot [\mathbf{Q}^l \cdot (\mathbf{Q}^T)^l]_{i,j}, \quad (\forall i, j \in [1, n]) \quad (5)$$

implies that $[\mathbf{S}]_{i,j}$ considers only contributions of *symmetric* in-link paths for (i, j), neglecting all *dissymmetric* ones. Consequently, $[\mathbf{S}]_{i,j} = 0$ if (i, j) has no symmetric paths. This proves the "zero-similarity" problem for SimRank.

Non-SimRank Based Metrics. Other measures, *e.g.,* Random Walk with Restart (RWR) and Personalized PageRank (PPR), also imply a SimRank-like "zero-similarity" issue.

As PPR is just a special vector form of RWR, our following discussion will mainly focus on RWR, which also suites PPR.

The "zero-similarity" issue for RWR, similar to SimRank, is that "nodes i and j are assessed as dissimilar $s_{\mathsf{rwr}}(i, j) = 0$ if there are no paths with one direction from i to j". For example in Figure 1, h and d are still dissimilar for RWR, as both $h \leftarrow e \leftarrow \boxed{a} \to d$ and $h \leftarrow e \leftarrow \boxed{a} \to b \to f \to d$ have two directions. However, $s_{\mathsf{rwr}}(a, f) \neq 0$ as there exists a path $\boxed{a} \to b \to f$ with one direction $(\to)$ from a to f. Thus, both RWR and SimRank may encounter "zero-similarity" issues. Indeed, in the language of in-link paths, while SimRank considers only *symmetric* in-link paths (whose "source" node is in the center), RWR merely tallies *unidirectional* in-link paths (whose "source" node is at one end), both of which are in a biased way to assess similarity.

To further clarify the "zero-similarity" issue for RWR, we can convert its closed form $\mathbf{S} = (1 - C) \cdot (\mathbf{I}_n - C \cdot \mathbf{W})^{-1}$ [19] into the power series form

$$[\mathbf{S}]_{i,j} = (1 - C) \cdot \sum_{k=0}^{\infty} C^k \cdot [\mathbf{W}^k]_{i,j}. \qquad (6)$$

As $\mathbf{W}$ is a weighted (*i.e.,* row-normalized) matrix of $\mathbf{A}$, we have $[\mathbf{W}^k]_{i,j} = 0 \;\Leftrightarrow\; [\mathbf{A}^k]_{i,j} = 0$. Thus, by Lemma 1, the drawback of RWR is clear: $[\mathbf{S}]_{i,j}$ only tallies the weight sum of paths with one direction from i to j, yet totally ignores in-link paths whose "source" node is not at node i.

In a nutshell, RWR may not resolve "zero-similarity" issues for SimRank, and vice versa. As will be seen in Figure 3, all nodes in the family tree G should have some relevances. Although RWR considers "Father and Me being similar" that is neglected by SimRank, it ignores "Me and Cousin being similar" that is accommodated by SimRank. Besides, both RWR and SimRank neglect "Me and Uncle being similar". Worse still, RWR fails to produce *symmetric* similarity $(s(i, j) \neq s(j, i))$. Since there is no path directed from Me to Father, RWR alleges "Me and Father being dissimilar". These call for a unified measure for similarity assessment.

3.2 SimRank*: A Remedy for SimRank

The reinterpretation of SimRank provides a new possible remedy to its "zero-similarity" problem.

SimRank* (Geometric Series Form). Since SimRank (*resp.* RWR) loses all *dissymmetric* (*resp. non-unidirectional*) in-link paths for node-pair (i, j), our treatment aims to compensate $s(i, j)$ for such a loss, by accommodating all *dissymmetric* (*resp. non-unidirectional*) in-link paths. Precisely, by adding the terms $[\mathbf{Q}^{l_1} \cdot (\mathbf{Q}^T)^{l_2}]_{i,j}$, $\forall l_1 \neq l_2$ (*resp.* $\forall l_1 \neq 0$), with appropriate weights, into the series form of SimRank (*resp.* RWR), we can derive a new treatment as follows:

$$\hat{\mathbf{S}} = (1 - C) \cdot \sum_{l=0}^{\infty} \frac{C^l}{2^l} \cdot \sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot \mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha}. \qquad (7)$$

Here, $\binom{l}{\alpha}$ is *the binomial coefficient* defined as $\binom{l}{\alpha} = \frac{l!}{\alpha!(l-\alpha)!}$. We call Eq.(7) *the geometric* [6] *series form of SimRank**.

To see how the geometric form of SimRank* Eq.(7) is derived and why it can perfectly resolve the "zero-similarity" problem for SimRank and RWR, we rewrite Eq.(7) as

$$[\hat{\mathbf{S}}]_{i,j} = (1 - C) \cdot \sum_{l=0}^{\infty} C^l \cdot [\hat{\mathbf{T}}_l]_{i,j} \quad \text{with} \qquad (8)$$

$$[\hat{\mathbf{T}}_l]_{i,j} = \frac{1}{2^l} \cdot \sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot [\mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha}]_{i,j}. \quad (\forall i, j \in [1, n])$$

[5] The matrix norm $\|\mathbf{X}\|_{\max} = \max_{i,j} |[\mathbf{X}]_{i,j}|$ is the maximum absolute entry of $\mathbf{X}$.

[6] Since $\{C^l\}$ in Eq.(7) is a *geometric* sequence, we abuse the term "*geometric*" for this series form, to distinguish Eq.(11).

Length	SimRank	RWR / PPR	α	SimRank*
1	N/A	$\boxed{i} \to j$	0	$\boxed{i} \to j$
			1	$i \leftarrow \boxed{j}$
2	$i \leftarrow \boxed{\bullet} \to j$	$\boxed{i} \to \circ \to j$	0	$\boxed{i} \to \circ \to j$
			1	$i \leftarrow \boxed{\bullet} \to j$
			2	$i \leftarrow \circ \leftarrow \boxed{j}$
3	N/A	$\boxed{i} \to \circ \to \circ \to j$	0	$\boxed{i} \to \circ \to \circ \to j$
			1	$i \leftarrow \boxed{\bullet} \to \circ \to j$
			2	$i \leftarrow \circ \leftarrow \boxed{\bullet} \to j$
			3	$i \leftarrow \circ \leftarrow \circ \leftarrow \boxed{j}$
4	$i \leftarrow \circ \leftarrow \boxed{\bullet} \to \circ \to j$	$\boxed{i} \to \circ \to \circ \to \circ \to j$	0	$\boxed{i} \to \circ \to \circ \to \circ \to j$
			1	$i \leftarrow \boxed{\bullet} \to \circ \to \circ \to j$
			2	$i \leftarrow \circ \leftarrow \boxed{\bullet} \to \circ \to j$
			3	$i \leftarrow \circ \leftarrow \circ \leftarrow \boxed{\bullet} \to j$
			4	$i \leftarrow \circ \leftarrow \circ \leftarrow \circ \leftarrow \boxed{j}$

$\circ$ – any node in $\mathcal{G}$ $\boxed{i}$, $\boxed{\bullet}$, $\boxed{j}$ – in-link "source"

Figure 2: In-link Paths of (i, j) for Length $l \in [1, 4]$ Counted by SimRank, RWR/PPR, and SimRank*

Below, to avoid ambiguity, we use $\hat{\mathbf{S}}$ to denote the exact SimRank* in Eq.(7), and $\mathbf{S}$ the exact SimRank in Eq.(4).

Comparing Eq.(8) with Eq.(5), we see that for a fixed l, SimRank* $\hat{s}(i, j)$ uses $\sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot [\mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha}]_{i,j}$ in $[\hat{\mathbf{T}}_l]_{i,j}$ to consider *all* in-link paths of length l for node-pair (i, j) in a comprehensive way, as opposed to SimRank $s(i, j)$ using $[\mathbf{Q}^l \cdot (\mathbf{Q}^T)^l]_{i,j}$ in Eq.(5) to accommodate only *symmetric* in-link paths of length $2l$ for node-pair (i, j) in a biased manner. As a result, SimRank* may find all (dissymmetric) in-link paths of two kinds, both of which are ignored by SimRank: (1) in-link paths of odd length; (2) in-link paths of even length whose in-link "source" is not in the center.

Though RWR via Eq.(6) using $[\mathbf{W}^l]_{i,j}$ may consider part of in-link paths of odd length that are missed by SimRank, they ignore (non-unidirectional) in-link paths of two kinds: (1) all symmetric ones that are accommodated by SimRank; (2) dissymmetric ones whose in-link "source" is not at an end, both of which can be found by SimRank*.

For instance, given a node-pair (i, j), Figure 2 compares all in-link paths of length $l \in [1, 4]$ considered by SimRank, RWR, and SimRank*. It can be seen from 'SimRank* Column' that only a small number of in-link paths can be accommodated by SimRank (in dark gray cells) and RWR (in light gray cells), relative to those of SimRank*.

Weighted Factors of Two Types. We next elaborate on two kinds of weighted factors adopted by SimRank* Eq.(8): (1) *length weights* $\{C^l\}_{l=0}^{\infty}$, (2) *symmetry weights* $\{\binom{l}{\alpha}\}_{\alpha=0}^{l}$.

Intuitively, *the length weight* C^l $(0 < C < 1)$ measures the importance of in-link paths of different lengths. Similar to the original SimRank (Eq.(5)), the outer summation over l in SimRank* (Eq.(8)) is to add up the contributions of in-paths of different length l. The length weight C^l aims at reducing the contributions of in-paths of *long* lengths relative to *short* ones, as $\{C^l\}_{l \in [0,\infty)}$ is a deceasing sequence *w.r.t.* length l.

The symmetry weight uses binomial $\binom{l}{\alpha}$ $(0 \le \alpha \le l)$ to assess the importance of in-link paths of a fixed length l, with α edges in one direction (from the "source" node to one end of the path) and $l - \alpha$ edges in the opposite direction. Here, α reflects the symmetry of in-link paths of length l. As depicted in Figure 2, when $\alpha = 0$ or l, in-link paths are totally dissymmetric, reducing to one single direction; when α is close to $\lfloor l/2 \rfloor$, the "source" node is near the center of in-link paths, being almost symmetric. To show the use of binomial $\binom{l}{\alpha}$ is reasonable, we consider the following issues.

(a) Why $\binom{l}{\alpha}$ is assigned only to $l + 1$ kinds of *in-link paths*, for a fixed l? Say, for $l = 4$ in Fig. 2, why neglect paths

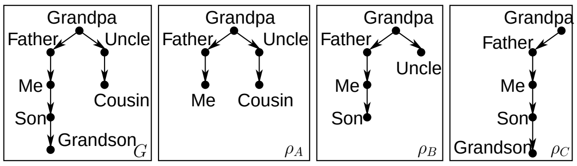


Figure 3: The more symmetric the in-link paths are, the larger contributions they will have to similarity

$\rho_1 : i \to \circ \leftarrow \circ \to \circ \leftarrow j$ and $\rho_2 : i \leftarrow \star \to \circ \leftarrow \diamond \to j$?

(b) Why use $\binom{l}{\alpha}$, instead of others, to weigh in-link paths?

(c) Why symmetric in-link paths are considered to be more important than less symmetric ones, for a fixed length?

For (a), as our SimRank* framework is *in-link oriented*,[7] the impact of *out-links* on similarity is not accommodated. Thus, for $l = 4$, the path ρ_1 is not considered since there are no *in-links* to nodes i and j in ρ_1. Even if i or j has in-links yet without *one common* in-link "source", *e.g.,* ρ_2, this path also has no contributions to similarity $\hat{s}(i, j)$. This is because in ρ_2 there are no in-links to nodes $\star$ and $\diamond$, thus the sub-path $\star \to \circ \leftarrow \diamond$ of ρ_2 has no contributions to $\hat{s}(\star, \diamond)$, which, iteratively, has no contributions to $\hat{s}(i, j)$. Hence, due to our *in-link oriented* framework for similarity assessment, for a fixed l, there are *at most* $l + 1$ kinds of *in-link paths* (where binomial weights $\binom{l}{\alpha}$ are assigned) having contributions to $\hat{s}(i, j)$, with $\alpha \in [0, l]$ edges in one direction and $l - \alpha$ edges in the opposite one, as shown in Figure 2.

For (b), there are 2 reasons for using $\binom{l}{\alpha}$ instead of others: (i) The binomial $\binom{l}{\alpha}$ can reduce the contributions of less symmetric in-link paths, relative to symmetric ones. Indeed, a larger (*resp.* smaller) weight is expected for an in-link path whose "source" is closer to the center (*resp.* either of ends). $\binom{l}{\alpha}$ happens to have this monotonicity: For a fixed l, when α increases from 0 to l, $\binom{l}{\alpha}$ first increases from 1 to a maximum value ($\alpha = \lfloor \frac{l}{2} \rfloor$, "source" at the center), and then "symmetrically" decreases back to 1 ($\alpha = l$, "source" at one end). (ii) The binomial $\binom{l}{\alpha}$ is an easy-to-compute math function, which enables the infinite series (Eq.(7)) to be simplified, as will be seen shortly, into the very succinct and elegant recurrence form (Eq.(13)). To our best knowledge, although some functions, like $e^{-(l-\frac{\alpha}{2})^2}$, have the similar monotonicity of $\binom{l}{\alpha}$, they would adversely complicate the form of Eq.(7) since it is even hard to compute $\sum_{\alpha=0}^{l} e^{-(l-\frac{\alpha}{2})^2}$ to determine the normalized weight factors, not to mention being able to simplify Eq.(7) into the elegant recurrence form. In contrast, $\sum_{\alpha=0}^{l} \binom{l}{\alpha} = 2^l$ enjoys a neat form. Inspired by these, we use $\binom{l}{\alpha}$, instead of others, as the preferred symmetric weight.

For (c), the example below can explain, for a fixed length, why larger weights are assigned to more symmetric paths. Consider paths ρ_A, ρ_B and ρ_C of a family tree in Figure 3. Most people might feel ρ_A (Me and Cousin being similar) is more reliable than ρ_B (Uncle and Son being similar), which is more reliable than ρ_C (Grandpa and Grandson being similar). Thus, the more symmetric the in-link path is, the larger contribution it has to similarity assessment. In Figure 3, ρ_A should have the largest weight, ρ_B the second, ρ_C the third.

The efficacy of $(1 - C)$ and $\frac{1}{2^l}$ in Eq.(8) is to normalize $[\hat{\mathbf{S}}]_{i,j}$ and $[\hat{\mathbf{T}}_l]_{i,j}$, respectively, into $[0, 1]$. More specifically, one can readily verify that $\|\mathbf{Q}^{l_1} \cdot (\mathbf{Q}^T)^{l_2}\|_{\max} \le 1$, for $\forall l_1, l_2$.

[7] In order to highlight the essence of "zero-SimRank" issue, our SimRank* model, just like SimRank, PageRank, and RWR, is based on *incoming* edges for assessing similarity.

Thus, (i) $\| \sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot \mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha} \|_{\max} \leq \sum_{\alpha=0}^{l} \binom{l}{\alpha} = 2^l$, which implies $\|\hat{\mathbf{T}}_l\|_{\max} \leq 1$. (ii) Since $\| \sum_{l=0}^{\infty} C^l \cdot \hat{\mathbf{T}}_l \|_{\max} \leq \sum_{l=0}^{\infty} C^l = \frac{1}{1-C}$, it follows that $\|\mathbf{S}\|_{\max} \leq 1$.

Combining these two kinds of weights, the contribution of any in-link path for a given node-pair can be easily assessed. For example in Figure 1, $h \leftarrow e \leftarrow \boxed{a} \rightarrow d$ has a contribution rate of $(1-0.8) \cdot 0.8^3 \frac{1}{2^3} \binom{3}{2} = 0.0384$ for node-pair (h, d), Similarly, $h \leftarrow e \leftarrow \boxed{a} \rightarrow b \rightarrow f \rightarrow d$ has a contribution rate of $(1-0.8) \cdot 0.8^5 \frac{1}{2^5} \binom{5}{2} = 0.0205$. As opposed to SimRank only using length weight C^l, SimRank* considers both C^l and symmetry weight $\binom{l}{\alpha}$. Thus, our revision resolves "zero-SimRank" issues, as well as inherits SimRank philosophy.

Convergence of SimRank*. As SimRank* in Eq.(7) is an *infinite* series, it is unclear whether this series is convergent. This motivates us to study its convergence issue.

Let us first define *the k-th partial sum* of Eq.(7) as

$$\hat{\mathbf{S}}_k = (1-C) \cdot \sum_{l=0}^{k} \frac{C^l}{2^l} \cdot \sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot \mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha}. \qquad (9)$$

Leveraging $\hat{\mathbf{S}}_k$, we next show the convergence of Eq.(7).

Lemma 3. *Let $\hat{\mathbf{S}}$ and $\hat{\mathbf{S}}_k$ be defined by Eqs.(7) and (9), respectively. Then, the gap between $\hat{\mathbf{S}}$ and $\hat{\mathbf{S}}_k$ is bounded by*

$$\|\hat{\mathbf{S}} - \hat{\mathbf{S}}_k\|_{\max} \leq C^{k+1}. \quad (\forall k = 0, 1, \cdots) \qquad (10)$$

Proof. For each $k = 0, 1, \cdots$, we subtract Eq.(9) from Eq.(7), and then take $\| \star \|_{\max}$ norms on both sides to get

$$\|\hat{\mathbf{S}} - \hat{\mathbf{S}}_k\|_{\max} \leq (1-C) \sum_{l=k+1}^{\infty} \frac{C^l}{2^l} \cdot \underbrace{\sum_{\alpha=0}^{l} \binom{l}{\alpha}}_{=2^l} \cdot \underbrace{\|\mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha}\|_{\max}}_{\leq 1}$$

$$\leq (1-C) \sum_{l=k+1}^{\infty} C^l = (1-C) \cdot \frac{C^{k+1}}{(1-C)} = C^{k+1}. \quad \square$$

The convergence of SimRank* (Eq.(7)) follows directly from Lemma 3 and $\lim_{k \to \infty} C^{k+1} = 0 \ \ (0 < C < 1)$.

SimRank* (Exponential Series Form). In the *geometric* series form of SimRank* (Eq.(7)), Lemma 3 implies that, to guarantee the accuracy ϵ, the K-th partial sum $\hat{\mathbf{S}}_K$ with $K = \lceil \log_C \epsilon \rceil$ can be used to approximate the exact solution. However, there is a variant of SimRank* that can use only the K'-th partial sum with $K' \leq K$ to ensure the same ϵ:

$$\hat{\mathbf{S}}' = e^{-C} \cdot \sum_{l=0}^{\infty} \frac{C^l}{l!} \cdot \frac{1}{2^l} \sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot \mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha}. \qquad (11)$$

We call Eq.(11) the *exponential series form of SimRank**. It differs from Eq.(7) in the length weight $\frac{C^l}{l!}$ (which is an *exponential* sequence *w.r.t.* l) and its normalized factor e^{-C}.

The exponential series form of SimRank* is introduced to improve the rate of convergence for similarity computation. To clarify this, we define $\hat{\mathbf{S}}'_k$ as the k-th partial sum of $\hat{\mathbf{S}}'$ in Eq.(11). Analogous to Lemma 3, one can readily prove

$$\|\hat{\mathbf{S}}' - \hat{\mathbf{S}}'_k\|_{\max} \leq \tfrac{C^{k+1}}{(k+1)!}. \quad (\forall k = 0, 1, \cdots) \qquad (12)$$

Comparing Eq.(12) with Eq.(10), we see that for any fixed k, as $\frac{C^{k+1}}{(k+1)!} \leq C^{k+1}$, the convergence rate of $\hat{\mathbf{S}}'_k$ is *always* faster than that of $\hat{\mathbf{S}}_k$. Hence, to guarantee the same accuracy, the exponential SimRank* only needs to compute a tiny fraction of the partial sums of the geometric SimRank*.

The choice of length weight $\frac{C^l}{l!}$ for the exponential SimRank* (Eq.(11)) plays a key role in accelerating convergence. As suggested by the proof of Lemma 3, the bound C^{k+1} in Eq.(10) (*resp.* $\frac{C^{k+1}}{(k+1)!}$ in Eq.(12)) is actually derived from our choice of length weight C^l (*resp.* $\frac{C^l}{l!}$) for the geometric (*resp.* exponential) SimRank*. Thus, there might exist other length weights for speeding up the convergence of SimRank*, as there is no sanctity of the earlier choices of length weight. That is, apart from C^l and $\frac{C^l}{l!}$, other sequence, *e.g.,* $\frac{C^l}{l}$, that satisfies decreasing monotonicity *w.r.t.* length l can be regarded as another possible candidate for length weight, since the efficacy of the length weight is to reduce the contributions of in-link paths of long lengths relative to short ones. The reasons why we select C^l and $\frac{C^l}{l!}$, instead of others, are two-fold: (1) The normalized factor of length weight should have a simple form, *e.g.,* $\sum_{l=0}^{\infty} \frac{C^l}{l!} = e^C$. (2) Once selected, the length weight should enable the series form of SimRank* to be simplified into a very elegant form, *e.g.,* using $\frac{C^l}{l!}$ allows Eq.(11) being simplified, as will be seen in Eq.(15), into a neat closed form. In contrast, $\frac{C^l}{l}$ is not a preferred length weight as its series version may not be simplified into a neat recursive (or closed) form, though the form $\sum_{l=0}^{\infty} \frac{C^l}{l} = \ln \frac{1}{(1-C)}$ is simple for normalized factor.

4. EFFICIENTLY COMPUTING SIMRANK*

At first glance, the series form of SimRank* (Eq.(7)) is more complicated than that of SimRank (Eq.(4)). A brute-force way of computing the first k-th partial sums of Eq.(7) requires $O(k \cdot l^2 \cdot n^3)$ time, involving l^2 matrix multiplications in the inner summation for each fixed l in the outer summation, which seems much more expensive than SimRank.

In this section, we first reformulate the series forms of SimRank* into elegant recursive and closed forms. We then propose efficient techniques for computing SimRank*.

4.1 Recursive & Closed Forms of SimRank*

The series forms of SimRank* (Eqs.(7) and (11)) are tedious, and suffer from high complexity if calculated directly.

The main result of this subsection is to derive an elegant recursive form for Eq.(7) and a closed form for Eq.(11), which will be useful for efficient SimRank* computation.

Recursive Form of Geometric SimRank*. We first show a recursive form for the geometric SimRank* of Eq.(7).

Theorem 2. *The SimRank* geometric series $\hat{\mathbf{S}}$ in Eq.(7) takes the following elegant recursive form:*

$$\hat{\mathbf{S}} = \tfrac{C}{2} \cdot (\mathbf{Q} \cdot \hat{\mathbf{S}} + \hat{\mathbf{S}} \cdot \mathbf{Q}^T) + (1-C) \cdot \mathbf{I}_n. \qquad (13)$$

To prove Theorem 2, the following lemma is needed.

Lemma 4. *For each $k = 0, 1, \cdots$, the k-th partial sum $\hat{\mathbf{S}}_k$ defined by Eq.(9) satisfies the following iteration:*

$$\begin{cases} \hat{\mathbf{S}}_0 = (1-C) \cdot \mathbf{I}_n, \\ \hat{\mathbf{S}}_{k+1} = \frac{C}{2} \cdot (\mathbf{Q} \cdot \hat{\mathbf{S}}_k + \hat{\mathbf{S}}_k \cdot \mathbf{Q}^T) + (1-C) \cdot \mathbf{I}_n. \end{cases} \qquad (14)$$

Proof. For $k = 0$, it is obvious from Eq.(9) that $\hat{\mathbf{S}}_0 = (1-C) \cdot \mathbf{I}_n$, which satisfies Eq.(14). For $k = 1, 2, \cdots$, substituting Eq.(9) into the right-hand side of Eq.(14) yields

$$\hat{\mathbf{S}}_{k+1} = \frac{C}{2} \cdot \Big((1-C) \sum_{l=0}^{k} \frac{C^l}{2^l} \overbrace{\sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot \mathbf{Q}^{\alpha+1} \cdot (\mathbf{Q}^T)^{l-\alpha}}^{= \sum_{\alpha=1}^{l+1} \binom{l}{\alpha-1} \cdot \mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha+1}} +$$

$$
\begin{aligned}
&+(1-C)\sum_{l=0}^{k}\frac{C^l}{2^l}\sum_{\alpha=0}^{l}\binom{l}{\alpha}\cdot\mathbf{Q}^{\alpha}\cdot(\mathbf{Q}^T)^{l-\alpha+1}\Big)+(1-C)\cdot\mathbf{I}_n \\
=\ & \frac{C}{2}(1-C)\Big(\sum_{l=0}^{k}\frac{C^l}{2^l}\Big(\sum_{\alpha=1}^{l}\Big(\binom{l}{\alpha-1}+\binom{l}{\alpha}\Big)\mathbf{Q}^{\alpha}(\mathbf{Q}^T)^{l-\alpha+1}\Big)+ \\
&+\mathbf{Q}^{l+1}+(\mathbf{Q}^T)^{l+1}\Big)+(1-C)\cdot\mathbf{I}_n \\
=\ & \frac{C}{2}\cdot(1-C)\Big(\sum_{l=0}^{k}\frac{C^l}{2^l}\cdot\sum_{\alpha=0}^{l+1}\binom{l+1}{\alpha}\cdot\mathbf{Q}^{\alpha}(\mathbf{Q}^T)^{l-\alpha+1}\Big)+(1-C)\mathbf{I}_n \\
=\ & (1-C)\cdot\sum_{l=0}^{k+1}\frac{C^l}{2^l}\cdot\sum_{\alpha=0}^{l}\binom{l}{\alpha}\cdot\mathbf{Q}^{\alpha}\cdot(\mathbf{Q}^T)^{l-\alpha}.
\end{aligned}
$$

Thus, $\hat{\mathbf{S}}_{k+1}$ in Eq.(14) also takes the form of Eq.(9). □

One consequence of Lemma 4 is the proof of Theorem 2.

PROOF OF THEOREM 2. Lemma 3 implies the convergence of SimRank*, *i.e.,* the existence of $\lim_{k\to\infty}\hat{\mathbf{S}}_k$. Thus, taking limits on both sides of Eq.(14) as $k\to\infty$ yields Eq.(13). □

Closed Form of Exponential SimRank*. We next present a closed formula for the exponential SimRank* of Eq.(11).

THEOREM 3. *The exponential series form of SimRank* in Eq.*(11) *neatly takes the following closed form:*

$$\hat{\mathbf{S}}' = e^{-C}\cdot e^{\frac{C}{2}\mathbf{Q}}\cdot e^{\frac{C}{2}\mathbf{Q}^T}.\ ^{8} \tag{15}$$

PROOF. We utilize the factorial formula $\binom{l}{\alpha}=\frac{l!}{\alpha!(l-\alpha)!}$ to simplify the series form of Eq.(11) into the closed form:

$$
\begin{aligned}
\hat{\mathbf{S}}' &= e^{-C}\cdot\sum_{l=0}^{\infty}\sum_{\alpha=0}^{l}\frac{1}{2^l}\cdot\frac{C^{\alpha}}{\alpha!}\mathbf{Q}^{\alpha}\cdot\frac{C^{l-\alpha}}{(l-\alpha)!}(\mathbf{Q}^T)^{l-\alpha} \\
&= e^{-C}\cdot\sum_{\alpha=0}^{\infty}\frac{C^{\alpha}}{\alpha!}\mathbf{Q}^{\alpha}\cdot\underbrace{\sum_{l=\alpha}^{\infty}\frac{1}{2^l}\cdot\frac{C^{l-\alpha}}{(l-\alpha)!}(\mathbf{Q}^T)^{l-\alpha}}_{=\sum_{l=0}^{\infty}\frac{1}{2^{l+\alpha}}\cdot\frac{t^l}{l!}(\mathbf{Q}^T)^l} \\
&= e^{-C}\cdot\left(\sum_{\alpha=0}^{\infty}\frac{1}{2^{\alpha}}\cdot\frac{C^{\alpha}}{\alpha!}\mathbf{Q}^{\alpha}\right)\cdot\left(\sum_{l=0}^{\infty}\frac{1}{2^l}\cdot\frac{C^l}{l!}(\mathbf{Q}^T)^l\right) \\
&= e^{-C}\cdot e^{\frac{C}{2}\mathbf{Q}}\cdot e^{\frac{C}{2}\mathbf{Q}^T},
\end{aligned}
$$

where the second equality is obtained by interchanging the order of double summation $\sum_{l=0}^{\infty}\sum_{\alpha=0}^{l}f(l,\alpha)=\sum_{\alpha=0}^{\infty}\sum_{l=\alpha}^{\infty}f(l,\alpha)$. □

The utility of Theorem 3 will be appreciated in Subsect. 4.3 for optimizing the exponential SimRank* computation.

4.2 SimRank* Computation

Having formulated SimRank* into the very elegant forms, we next develop efficient techniques to speed up the computation of SimRank*.

Due to high commonalities between the geometric SimRank* $\hat{\mathbf{S}}$ (in Eq.(7)) and its exponential variant $\hat{\mathbf{S}}'$ (in Eq.(11)), we shall mainly focus on geometric SimRank* computation, which is readily applicable to its exponential variant as well.

Algorithm. To compute the SimRank* series $\hat{\mathbf{S}}$ in Eq.(7), the closed form Eq.(13) provides an easy yet effective way: One can use the iterative paradigm Eq.(14) to compute $\hat{\mathbf{S}}_k$, with accuracy guaranteed by Lemma 3.

Complexity. The computational time of performing Eq.(14) is $O(Knm)$ for K iterations on a graph of n nodes and m edges, which is dominated by the cost of matrix multiplication $\mathbf{Q}\cdot\hat{\mathbf{S}}_k$ per iteration. Due to $\hat{\mathbf{S}}_k$ symmetry, the result of $\hat{\mathbf{S}}_k\cdot\mathbf{Q}^T$ can be obtained from the transpose of the calculated matrix $\mathbf{Q}\cdot\hat{\mathbf{S}}_k$. Thus, for each iteration, Eq.(14) requires only one matrix multiplication (corresponding to performing only a *single* summation of Eq.(14)), as opposed to its counterpart of computing SimRank via Eq.(3) that needs two matrix multiplications for $\mathbf{Q}\cdot\mathbf{S}_k\cdot\mathbf{Q}^T$ (corresponding to performing a *double* summation of Eq.(2) regardless of whether memoization [17] is used). From this perspective, despite the traversal of more in-link paths, SimRank* runs even faster (up to a constant factor) than SimRank, which is a substantial improvement achieved by Theorem 2.

4.3 Optimizations

To accelerate SimRank* iterations in Eq.(14), the conventional optimization techniques [17] for SimRank cannot be effectively applied to SimRank*. Indeed, Lizorkin *et al.* [17] proposed three appealing approaches for optimizing SimRank computation, *i.e.,* essential node-pair selection, partial sums memoization, and threshold-sieved similarities, among which only the threshold-sieved similarities method can be ported to SimRank* that allows eliminating node-pairs of small similarities in the computation. Essential node-pair selection no longer applies because SimRank utilizes a “zero-similarity” set as a pruning rule to speed up its computation, whereas SimRank* regards the existence of such a set as an issue of the SimRank philosophy and attempts to fix it. Partial sums memoization plays a vital role in significantly speeding up the computation of SimRank to $O(Knm)$ time. To see why it does not work in SimRank*, let us compare the component forms of SimRank and SimRank*, respectively, in Eqs.(16) and (17):

$$s_{k+1}(a,b)=\frac{C}{|\mathcal{I}(a)||\mathcal{I}(b)|}\sum_{x\in\mathcal{I}(a)}\overbrace{\sum_{y\in\mathcal{I}(b)}s_k(x,y)}^{=\text{Partial}^{s_k}_{\mathcal{I}(b)}(x)}. \tag{16}$$

$$\hat{s}_{k+1}(a,b)=\frac{C}{2|\mathcal{I}(b)|}\underbrace{\sum_{y\in\mathcal{I}(b)}\hat{s}_k(a,y)}_{=\text{Partial}^{\hat{s}_k}_{\mathcal{I}(b)}(a)}+\frac{C}{2|\mathcal{I}(a)|}\sum_{x\in\mathcal{I}(a)}\hat{s}_k(x,b). \tag{17}$$

For SimRank, if $\mathcal{I}(a)$ and $\mathcal{I}(\star)$ have some node, say i, in common, then the partial sum $\text{Partial}^{s_k}_{\mathcal{I}(b)}(i)$ in Eq.(16), once memoized, can be reused in both $\hat{s}_{k+1}(a,b)$ and $\hat{s}_{k+1}(\star,b)$ computation. In contrast, for SimRank*, no matter whether $\mathcal{I}(a)\cap\mathcal{I}(\star)\neq\varnothing$, the partial sum $\text{Partial}^{\hat{s}_k}_{\mathcal{I}(b)}(a)$ in Eq.(17) for computing $\hat{s}_{k+1}(a,b)$, if memoized, has no chance to be reused again in computing other similarities $\hat{s}_{k+1}(\star,b)$, with $\star$ denoting any node in $\mathcal{G}$ except a.

Fine-grained Memoization. Instead of memoizing the results of $\sum_{y\in\mathcal{I}(b)}\hat{s}_k(a,y)$ over the *whole* set $\mathcal{I}(b)$ in Eq.(17), we use *fine-grained* memoization for optimizing SimRank* by caching a partial sum, in part, over a *subset* as follows:

$\text{Partial}^{\hat{s}_k}_{\Delta}(a)\triangleq\sum_{y\in\Delta}\hat{s}_k(a,y)$ with $\Delta\subseteq\mathcal{I}(\star)$.

Our observation is that there may be duplicate additions among $\sum_{y\in\mathcal{I}(\star)}\hat{s}_k(a,y)$ over *different* in-neighbor sets $\mathcal{I}(\star)$. Thus, once memoized, the result of $\text{Partial}^{\hat{s}_k}_{\Delta}(a)$ can be shared among many sums $\sum_{y\in\mathcal{I}(\star)}\hat{s}_k(a,y)$ for computing $\hat{s}_{k+1}(a,\star)$. As an example in Figure 1, $\mathcal{I}(h)$ and $\mathcal{I}(i)$ have three nodes $\{e,j,k\}$ in common, and thus, once memoized, the resulting fine-grained partial sum $\text{Partial}^{\hat{s}_k}_{\{e,j,k\}}(a)$ can be shared between $\sum_{y\in\mathcal{I}(h)}\hat{s}_k(a,y)$ and $\sum_{y\in\mathcal{I}(i)}\hat{s}_k(a,y)$ for computing

[8] $e^{\mathbf{X}}\triangleq\mathbf{I}+\mathbf{X}+\frac{\mathbf{X}^2}{2!}+\cdots=\sum_{k=0}^{\infty}\frac{\mathbf{X}^k}{k!}$, for a square matrix $\mathbf{X}$.

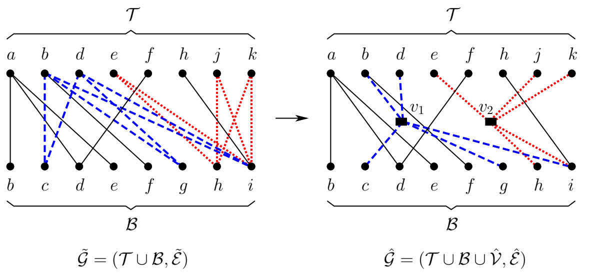

$\mathcal{T}$
a b d e f h j k
b c d e f g h i
$\mathcal{B}$
v_1
v_2
$\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$
$\hat{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B} \cup \hat{\mathcal{V}}, \hat{\mathcal{E}})$

Figure 4: Compression of Induced Bigraph $\tilde{\mathcal{G}}$ into $\hat{\mathcal{G}}$ via Edge Concentration

both $\hat{s}_{k+1}(a,h)$ and $\hat{s}_{k+1}(a,i)$ via Eq.(17), for any fixed a. However, it seems hard to find perfect fine-grained subsets $\Delta \subseteq \mathcal{I}(\star)$ for maximal computation sharing, since there may be many arbitrarily overlapped in-neighbor sets in a graph. To overcome this difficulty, we shall deploy efficient techniques of bipartite graph compression via edge concentration for finding such fine-grained subsets.

Induced Bigraph. We first construct *an induced bipartite graph (bigraph)* from $\mathcal{G}$, which is defined as follows.

DEFINITION 2. *An induced bipartite graph (bigraph) from a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a bipartite graph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$, such that its two disjoint node sets $\mathcal{T} = \{x \in \mathcal{V} \mid \mathcal{O}(x) \neq \varnothing\}$, $\mathcal{B} = \{x \in \mathcal{V} \mid \mathcal{I}(x) \neq \varnothing\}$,* [9] *and for each $u \in \mathcal{T}$ and $v \in \mathcal{B}$, $(u,v) \in \tilde{\mathcal{E}}$ if and only if there is an edge from u to v in $\mathcal{G}$.*

Intuitively, an induced bigraph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$ visualizes the neighborhood structure of $\mathcal{G}$ from a different perspective. For any $x \in \mathcal{B}$, the nodes in $\mathcal{T}$ that are connected with x correspond to the in-neighbors of x in $\mathcal{G}$. Note that when node x has both in- and out-neighbors in $\mathcal{G}$, label x that appears in both $\mathcal{T}$ and $\mathcal{B}$ will be regarded as two distinct nodes despite the same label. To avoid ambiguity, we shall use $x \in \mathcal{T}$ and $x \in \mathcal{B}$ to distinguish them. Each directed edge in $\mathcal{G}$ is mapped to one edge in $\tilde{\mathcal{G}}$, and thus, $|\mathcal{E}| = |\tilde{\mathcal{E}}|$. For instance, the left part of Figure 4 shows the induced bigraph $\tilde{\mathcal{G}}$ from $\mathcal{G}$ of Figure 1. From $\tilde{\mathcal{G}}$, we can clearly see that b and d in $\mathcal{B}$ are both connected with a in $\mathcal{T}$, meaning that, in $\mathcal{G}$, b and d both have an in-neighbor a.

Biclique Compression via Edge Concentration. Based on the induced bigraph $\tilde{\mathcal{G}}$, we next introduce the notion of *bipartite cliques (bicliques)*.

DEFINITION 3. *Given an induced bigraph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$, a pair of two disjoint subsets $\mathcal{X} \subseteq \mathcal{T}$ and $\mathcal{Y} \subseteq \mathcal{B}$ is called a biclique if $(x,y) \in \tilde{\mathcal{E}}$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

Intuitively, a biclique $(\mathcal{X}, \mathcal{Y})$ is a complete bipartite subgraph of $\tilde{\mathcal{G}}$, which has $|\mathcal{X}| + |\mathcal{Y}|$ nodes and $|\mathcal{X}| \times |\mathcal{Y}|$ edges. Each biclique $(\mathcal{X}, \mathcal{Y})$ in $\tilde{\mathcal{G}}$ tells us that in $\mathcal{G}$, all nodes $y \in \mathcal{Y}$ have the common in-neighbor set $\mathcal{X}$. For example, there are two bicliques in Figure 4: $(\{b,d\}, \{c,g,i\})$ in dashed line, and $(\{e,j,k\}, \{h,i\})$ in dotted line. Biclique $(\{b,d\}, \{c,g,i\})$ in $\tilde{\mathcal{G}}$ implies that in $\mathcal{G}$, three nodes c, g, i all have two in-neighbors $\{b,d\}$ in common.

Bicliques are introduced to compress bigraph $\tilde{\mathcal{G}}$ for optimizing SimRank* computation. It is important to notice that for any fixed node a, the total cost [10] of performing the sums $\sum_{y \in \mathcal{I}(\star)} \hat{s}_k(a,y)$ over all in-neighbor sets $\mathcal{I}(\star)$ (via Eq.(17)) is equal to the number $|\tilde{\mathcal{E}}|$ of edges of bigraph $\tilde{\mathcal{G}}$. Therefore, our goal of minimizing the cost of summations for SimRank* is equivalent to the problem of minimizing the number of edges in the compressed graph of $\tilde{\mathcal{G}}$. Unfortunately, this bigraph compression problem, also known as *edge concentration* (EC), has been proved to be NP-hard [15]. The main ingredient of EC is to group sets of edges in $\tilde{\mathcal{G}}$ together, so that the compressed graph contains fewer edges which often implies less cost of summations for SimRank*, while retaining the same information as $\tilde{\mathcal{G}}$. To compress $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$, we first leverage Buehrer and Chellapilla's algorithm [5] for finding collections of bicliques in $\tilde{\mathcal{G}}$. Their algorithm is based on the heuristic of frequent itemset mining, and requires $O(|\tilde{\mathcal{E}}| \log(|\mathcal{T}| + |\mathcal{B}|))$ time to identify bicliques. We then replace edges of each biclique $(\mathcal{X}, \mathcal{Y})$ with a special node, called *an edge concentration node*, whose "fan-in" is all nodes in $\mathcal{X}$ and whose "fan-out" is all nodes in $\mathcal{Y}$. Finally, the compressed graph, denoted as $\hat{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B} \cup \hat{\mathcal{V}}, \hat{\mathcal{E}})$, can be obtained from bigraph $\tilde{\mathcal{G}}$, where (i) $\mathcal{T}$ and $\mathcal{B}$ are the same as those of $\tilde{\mathcal{G}}$, (ii) $\hat{\mathcal{V}}$ is the set of edge concentration nodes, and (iii) $\hat{\mathcal{E}}$ is the set of edges in $\hat{\mathcal{G}}$. In practice, $|\hat{\mathcal{E}}|$ is typically much smaller than $|\tilde{\mathcal{E}}|$, since $|\mathcal{X}| \times |\mathcal{Y}|$ edges of each biclique in $\tilde{\mathcal{G}}$ are reduced to $|\mathcal{X}| + |\mathcal{Y}|$ edges in $\hat{\mathcal{G}}$, which is a substantial improvement achieved by edge concentration. For example, the right part of Figure 4 depicts the resultant graph $\hat{\mathcal{G}}$ of applying this approach to $\tilde{\mathcal{G}}$. We can see that the number of edges in $\hat{\mathcal{G}}$ is decreased by 2 via edge concentration, meaning that the cost of computing SimRank* in $\hat{\mathcal{G}}$ can be reduced by 2 operations, by adding two edge concentration nodes v_1 and v_2.

Algorithm. Based on compressed graph $\hat{\mathcal{G}}$, we next present an algorithm for computing SimRank*, by using fine-grained memoization. The algorithm, referred to as memo-gSR*, is shown in Algorithm 1. It takes as input a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a damping factor C, and the number of iterations K, and returns all-pairs of SimRank* similarities $\hat{s}(\star,\star)$.

To present the algorithm, we need the following notations. For a compressed graph $\hat{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B} \cup \hat{\mathcal{V}}, \hat{\mathcal{E}})$, we shall abuse (i) $\Delta(v)$ $(v \in \hat{\mathcal{V}})$ to denote all the "fan-in" nodes of concentration node v in $\mathcal{T}$, *i.e.*, $\Delta(v) = \{x \in \mathcal{T} \mid \exists (x,v) \in \hat{\mathcal{E}}\}$; and (ii) $\mathcal{N}(x)$ $(x \in \mathcal{B})$, to denote all the nodes in $\mathcal{T} \cup \hat{\mathcal{V}}$ that are connected with $x \in \mathcal{B}$, *i.e.*, $\mathcal{N}(x) = \{y \in \mathcal{T} \cup \hat{\mathcal{V}} \mid \exists (y,x) \in \hat{\mathcal{E}}\}$.

The algorithm memo-gSR* runs in two phases.

(1) Preprocessing (lines 1–2). The algorithm first constructs an induced bigraph $\tilde{\mathcal{G}}$ from $\mathcal{G}$ (line 1). Based on $\mathcal{G}$, it then compresses $\tilde{\mathcal{G}}$ into $\hat{\mathcal{G}}$, by invoking the algorithm [5] to replace bicliques of $\tilde{\mathcal{G}}$ with "stars" via edge concentration (line 2).

(2) Updating (lines 3–19). The algorithm then iteratively computes all $\hat{s}_k(\star,\star)$ based on $\hat{\mathcal{G}}$. For every iteration k, (i) it first uses fine-grained memoization to add up $\hat{s}_k(a,\star)$ for each fixed a, with $\star$ being each node in the "fan-in" set $\Delta(v)$ of an edge concentration node v in $\hat{\mathcal{V}}$ (lines 5–7). (ii) Using the memoized $\text{Partial}_{\Delta(\star)}^{\hat{s}_k}(a)$, it then computes the partial sums $\text{Partial}_{\mathcal{I}(\star)}^{\hat{s}_k}(a)$ over different in-neighbor set $\mathcal{I}(\star)$ of $\mathcal{G}$ (lines 8–10). Due to fine-grained memoization, the intermediate results $\text{Partial}_{\Delta(\star)}^{\hat{s}_k}(a)$, for any fixed node

[9] The notation $\mathcal{O}(x)$ denotes the out-neighbor set of node x.

[10] Here, the total cost refers to the number of additions plus assignment operations. For example, the cost of performing $\sum_{y \in \mathcal{I}(h)} \hat{s}_k(a,y) = \hat{s}_k(a,e) + \hat{s}_k(a,j) + \hat{s}_k(a,k)$ is 3, including 2 additions and 1 assignment operation to store the result, which is equal to the number of edges that are connected with node $h \in \mathcal{B}$ in the left part of Figure 4.

Algorithm 1: memo-gSR* $(\mathcal{G}, C, K)$

```
Input  : graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, damping factor $C$, iteration $K$.
Output: SimRank* scores $\hat{s}_K(\star,\star)$.
1  build an induced bigraph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$ from $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ;
2  generate a compressed graph $\hat{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B} \cup \hat{\mathcal{V}}, \hat{\mathcal{E}})$ from $\tilde{\mathcal{G}}$ ;
3  initialize $\hat{s}_0(x,y) \leftarrow \left\{ \begin{smallmatrix} 1, & x=y \\ 0, & x\neq y \end{smallmatrix} \right. \quad \forall x, y \in \mathcal{V}$ ;
4  for $k \leftarrow 0, 1, \cdots, K-1$ do
5      foreach node $v \in \hat{\mathcal{V}}$ in $\hat{\mathcal{G}}$ do
6          foreach node $a \in \mathcal{V}$ in $\mathcal{G}$ do
7              $\mathrm{Partial}_{\Delta(v)}^{\hat{s}_k}(a) \leftarrow \sum_{y \in \Delta(v)} \hat{s}_k(a,y)$ ;
8      foreach node $x \in \hat{\mathcal{B}}$ in $\hat{\mathcal{G}}$ do
9          foreach node $a \in \mathcal{V}$ in $\mathcal{G}$ do
10             $\mathrm{Partial}_{\mathcal{I}(x)}^{\hat{s}_k}(a) \leftarrow \sum_{y \in \mathcal{N}(x) \cap \mathcal{T}} \hat{s}_k(a,y)$
                   $+ \sum_{y \in \mathcal{N}(x) \cap \hat{\mathcal{V}}} \mathrm{Partial}_{\Delta(y)}^{\hat{s}_k}(a)$ ;
11     free $\mathrm{Partial}_{\Delta(v)}^{\hat{s}_k}(a) \quad \forall v \in \hat{\mathcal{V}}, a \in \mathcal{V}$ ;
12     foreach node $x \in \mathcal{V}$ in $\mathcal{G}$ do
13         foreach node $y \in \mathcal{V}$ in $\mathcal{G}$ do
14             initialize $t_1 \leftarrow 0$, $t_2 \leftarrow 0$ ;
15             if $\mathcal{I}(x) \neq \varnothing$ then $t_1 \leftarrow \frac{C}{2|\mathcal{I}(x)|} \mathrm{Partial}_{\mathcal{I}(x)}^{\hat{s}_k}(y)$ ;
16             if $\mathcal{I}(y) \neq \varnothing$ then $t_2 \leftarrow \frac{C}{2|\mathcal{I}(y)|} \mathrm{Partial}_{\mathcal{I}(y)}^{\hat{s}_k}(x)$ ;
17             compute $\hat{s}_{k+1}(x,y) \leftarrow t_1 + t_2 + \left\{ \begin{smallmatrix} 1-C, & x=y \\ 0, & x\neq y \end{smallmatrix} \right.$ ;
18     free $\mathrm{Partial}_{\mathcal{I}(x)}^{\hat{s}_k}(y) \quad \forall x \in \mathcal{V}, y \in \mathcal{V}$ ;
19 return $\hat{s}_K(\star,\star)$ ;
```

a, can be reused among many partial sums $\mathrm{Partial}_{\mathcal{I}(\star)}^{\hat{s}_k}(a)$ computations. Hence, the cost of computing SimRank* is reduced. (iii) By Eq.(17), these partial sums can be used for computing $\hat{s}_k(\star,\star)$ (lines 12–17). Due to $\hat{s}_k(\star,\star)$ symmetry, the second summation in Eq.(17) can be computed from $\mathrm{Partial}_{\Delta(a)}^{\hat{s}_k}(x)$ (line 16). Once processed, the memoized results are freed (lines 11 and 18). After K iterations, SimRank* scores $\hat{s}_K(\star,\star)$ of all-pairs are returned (line 19).

Correctness & Complexity. One can verify that the algorithm correctly computes $\hat{s}_K(\star,\star)$, which satisfies Eq.(17). Besides, memo-gSR* is in $O(Kn\tilde{m})$ time, where $\tilde{m}$ is the number of edges in the compressed graph $\hat{\mathcal{G}}$. Here, $\tilde{m}$ is *always* smaller than m, and in practice, $\tilde{m} \ll m$, depending on the number of bicliques, and biclique density in $\tilde{\mathcal{G}}$. This is because edge concentration compresses bicliques (dense subgraphs) in $\tilde{\mathcal{G}}$ such that for each biclique $(\mathcal{X}_i, \mathcal{Y}_i)$, the number of its edges $|\mathcal{X}_i| \cdot |\mathcal{Y}_i|$ can be reduced to $|\mathcal{X}_i| + |\mathcal{Y}_i|$. Thus, $\tilde{m} \leq m - N \cdot \sum_{i=1}^{N} (|\mathcal{X}_i| \cdot |\mathcal{Y}_i| - (|\mathcal{X}_i| + |\mathcal{Y}_i|))$, where N is the number of bicliques in $\tilde{\mathcal{G}}$. Since $|\mathcal{X}_i|, |\mathcal{Y}_i| \geq 2$, it holds that $|\mathcal{X}_i| + |\mathcal{Y}_i| \leq |\mathcal{X}_i| \cdot |\mathcal{Y}_i|$. Hence, $\tilde{m}$ is *always* less than m. Moreover, the construction of $\hat{\mathcal{G}}$ is in $O(|\tilde{\mathcal{E}}| \log(|\mathcal{T}| + |\mathcal{B}|)) = O(\tilde{m} \log(2n))$ time [5] (lines 1–2). For each iteration, $\mathrm{Partial}_{\mathcal{I}(\star)}^{\hat{s}_k}(\star)$ are in $O(n\tilde{m})$ time (lines 5–11), and $\hat{s}_k(\star,\star)$ in $O(n^2)$ time (lines 12–18). Thus, the total time is $O(Kn\tilde{m})$, as opposed to $O(Knm)$ of original iterations in Lemma 4.

EXAMPLE 2. *Recall the graph $\mathcal{G}$ of Fig. 1.* memo-gSR* *computes SimRank* scores of all-pairs in $\mathcal{G}$ as follows:*

First, using Definition 2 and the algorithm [5], it builds bigraph $\tilde{\mathcal{G}}$ and compressed graph $\hat{\mathcal{G}}$, as shown in Fig. 4.

Then, it iteratively computes SimRank via fine-grained memoization based on $\hat{\mathcal{G}}$. For example, to compute $\hat{s}_{k+1}(a,i)$ and $\hat{s}_{k+1}(a,h)$, it first memoizes the fine-grained partial sums over the "fan-in" sets of v_1 and v_2 (lines 5–7):*

$$Partial_{\Delta(v_1)}^{\hat{s}_k}(a) \leftarrow \hat{s}(b,a) + \hat{s}(d,a),$$

$$Partial_{\Delta(v_2)}^{\hat{s}_k}(a) \leftarrow \hat{s}(e,a) + \hat{s}(j,a) + \hat{s}(k,a).$$

Using memoized $Partial_{\Delta(v_1)}^{\hat{s}_k}(a)$ and $Partial_{\Delta(v_2)}^{\hat{s}_k}(a)$, it then computes $Partial_{\mathcal{I}(i)}^{\hat{s}_k}(a)$ and $Partial_{\mathcal{I}(h)}^{\hat{s}_k}(a)$ (lines 8–10)

$$Partial_{\mathcal{I}(i)}^{\hat{s}_k}(a) \leftarrow Partial_{\Delta(v_1)}^{\hat{s}_k}(a) + Partial_{\Delta(v_2)}^{\hat{s}_k}(a) + \hat{s}(h,a),$$

$$Partial_{\mathcal{I}(h)}^{\hat{s}_k}(a) \leftarrow Partial_{\Delta(v_2)}^{\hat{s}_k}(a).$$

Finally, since $\mathcal{I}(a) = 0$, $\hat{s}_{k+1}(a,i)$ and $\hat{s}_{k+1}(a,h)$ can be obtained as follows (lines 12–17):

$$\hat{s}_{k+1}(a,x) \leftarrow \tfrac{C}{2|\mathcal{I}(x)|} Partial_{\mathcal{I}(x)}^{\hat{s}_k}(a) \qquad (x \in \{i,j\})$$

The rest of the results are shown in Col. 'SR' in Fig. 1.* □

Exponential SimRank* Optimization. The aforementioned optimization methods for (geometric) SimRank* computation can be readily extended to exponential SimRank*.

To shed light on this, we recall the exponential SimRank* series in Eq.(11) and its closed form Eq.(15) in Theorem 3. Similar to the proof of Theorem 3, one can readily show that the k-th partial sum of $\hat{\mathbf{S}}'$ defined by

$$\hat{\mathbf{S}}'_k \triangleq e^{-C} \cdot \sum_{l=0}^{k} \frac{C^l}{l!} \cdot \frac{1}{2^l} \sum_{\alpha=0}^{l} \binom{l}{\alpha} \cdot \mathbf{Q}^{\alpha} \cdot (\mathbf{Q}^T)^{l-\alpha} \tag{18}$$

can be represented as the product of the k-th partial sum of matrix exponential $(e^{\frac{C}{2}\mathbf{Q}})$ and its transpose $(e^{\frac{C}{2}\mathbf{Q}})^T$, *i.e.*,

$$\hat{\mathbf{S}}'_k = e^{-C} \cdot \mathbf{T}_k \cdot {\mathbf{T}_k}^T, \text{ with } \mathbf{T}_k \triangleq \textstyle\sum_{i=0}^{k} (\frac{C}{2}\mathbf{Q})^i / i!.$$

Thus, computing $\hat{\mathbf{S}}'_k$ amounts to solving $\mathbf{T}_k$ that can be iteratively derived as follows:

$$\left\{ \begin{array}{l} \mathbf{R}_{k+1} = \mathbf{Q} \cdot \mathbf{R}_k \\ \mathbf{T}_{k+1} = \mathbf{T}_k + \frac{C^k}{2^k \cdot k!} \cdot \mathbf{R}_k \end{array} \right. \text{ with } \left\{ \begin{array}{l} \mathbf{R}_0 = \mathbf{I}_n \\ \mathbf{T}_0 = \mathbf{0}_n \end{array} \right. , \tag{19}$$

where $\mathbf{R}_k$ is an auxiliary matrix used for computing $\mathbf{T}_k$.

It is worth noting that the matrix equation $\mathbf{R}_{k+1} = \mathbf{Q} \cdot \mathbf{R}_k$ in Eq.(19) can be rewritten, in the component form, as

$$\begin{aligned} [\mathbf{R}_{k+1}]_{(a,b)} &= [\mathbf{Q} \cdot \mathbf{R}_k]_{(a,b)} = \textstyle\sum_{y=1}^{n} [\mathbf{Q}]_{(a,y)} \cdot [\mathbf{R}_k]_{(y,b)} \\ &= \tfrac{1}{|\mathcal{I}(a)|} \textstyle\sum_{y \in \mathcal{I}(a)} [\mathbf{R}_k]_{(y,b)}, \end{aligned}$$

which takes the similar form of the single summation in Eq.(17) except for the coefficient $\frac{C}{2}$. Thus, our previous optimization approach of fine-grained partial sums sharing used for Eq.(17) can be applied in a similar way to Eq.(19), for improving the computational efficiency. For the interest of space, we omit the detailed algorithm here.

5. EXPERIMENTAL EVALUATION

Our comprehensive empirical studies on real and synthetic data evaluate (i) the semantic richness and relative order of SimRank*; (ii) the computational efficiency of SimRank*.

Experimental Setting. We use the following datasets.

(1) Real data. For semantics and relative order evaluation, we use two graphs: CITHEPTH(*directed*), DBLP(*undirected*).

(a) CITHEPTH[11], a citation network, where nodes are papers labeled with titles, and an edge a citation. The data is collected from the arXiv, with papers from 1993 to 2003.

(b) DBLP[12], a collaboration graph, where nodes are authors, and edges co-authorships. The graph is derived from

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

[12] http://dblp.uni-trier.de/~ley/db/

Dataset	$\lvert\mathcal{G}\rvert$ ($\lvert\mathcal{V}\rvert$, $\lvert\mathcal{E}\rvert$)	Density ($\lvert\mathcal{E}\rvert/\lvert\mathcal{V}\rvert$)
CitHepTh	451K (33K, 418K)	12.6
DBLP	102K (15K, 87K)	5.8
D05	21K (4K, 17K)	4.3
D08	85K (13K, 72K)	5.5
D11	103K (14K, 89K)	6.3
Web-Google	5.8M (873K, 4.9M)	5.6
CitPatent	19.8M (3.6M, 16.2M)	4.5

Figure 5: Details of Real Datasets

6-year publications (2002–2007) in seven major conferences: SIGMOD, PODS, VLDB, ICDE, SIGKDD, SIGIR, WWW.

For computational efficiency evaluation, we use five graphs: *(c)* D05, D08, D11, three co-authorship graphs, which are constructed from 9-year DBLP publications (2003–2011) in 7 major conferences (as remarked in the first DBLP dataset). Each graph is built by choosing every 3 years as a time step. *(d)* Web-Google, a web graph, where nodes are pages, and edges links. The data is from Google Programming Contest. *(e)* CitPatent, a U.S. patent network, in which nodes are patents, and edges are citations made by patents. This data is maintained by the National Bureau of Economic Research.

The size $|G|(|\mathcal{V}|, |\mathcal{E}|)$ of the graphs are shown in Figure 5.

(2) Synthetic data. To produce synthetic networks, we use a generator GTgraph[13] that is controlled by $|\mathcal{V}|$ and $|\mathcal{E}|$.

(3) Baselines. We implement the following algorithms in Visual C++ 9.0. (a) our geometric SimRank* algorithm memo-gSR* and it exponential variant memo-eSR* via fine-grained memoization (Section 4.3); (b) our conventional iterative SimRank* algorithm iter-gSR* which, as a comparison to memo-gSR*, computes similarities without memoization (Section 4.2); (c) psum-SR [17] and psum-PR [23] algorithms that compute SimRank and P-Rank similarities via partial sums memoization, respectively; (d) mtx-SR algorithm [14] that computes SimRank using singular value decomposition. (e) RWR [19] measures the node proximity *w.r.t.* a query.

(4) Test Queries. To serve the ranking purpose, we select 500 query nodes from each graph, based on the following: For each graph, we first sort all nodes in order of their in-degree into 5 groups, and then randomly choose 100 nodes from each group, aiming to guarantee that the selected nodes can systematically cover a broad range of all possible queries. Here, we mainly focus on single-node queries, since a multi-node query can be fairly factorized into multiple single-node queries via Linearity Theorem [6]. For every experiment, the average performance is reported over all test queries.

(5) Parameters. We set the following default parameters: (a) $C = 0.6$, which is the typical decay factor used in [9]. (b) $K = 5$, which is the total number of iterations, being the time-accuracy trade-off. Besides, for all the methods, we clip similarity values at 10^{-4}, to discard far-apart nodes with scores less than 10^{-4} for storage. It can greatly reduce space cost with minimal impact on accuracy, as shown in [17].

(6) Effectiveness Metrics. To evaluate semantics and relative ordering, we consider both node and node-pair ranking. We adopt three metrics [6, 14]: *Kendall's* τ, *Spearman's* ρ, and *Normalized Discounted Cumulative Gain* (NDCG).
(a) *Kendall's* τ is defined as $\tau = \frac{2}{N(N-1)} \sum_{\{i,j\} \in P} \bar{K}_{i,j}(\tau_1, \tau_2)$, with $\bar{K}_{i,j}(\tau_1, \tau_2) = 1$ if i and j are in the same order in τ_1 and τ_2, and otherwise 0. Here, τ_1 and τ_2 are the rankings of elements in two lists, P is the set of unordered pairs in τ_1 and τ_2, and N is the number of elements in a ranking list.
(b) *Spearman's* ρ is given by $\rho = 1 - \frac{6 \sum_{i=1}^{N} d_i^2}{N(N^2-1)}$, where d_i is the ranking difference between the i-th elements in two lists.
(c) *NDCG at position* p *w.r.t. query* q is given by $\mathrm{NDCG}_p(q) = \frac{1}{\mathrm{IDCG}_p(q)} \sum_{i=1}^{p} \frac{2^{s(i,q)}-1}{\log_2(1+i)}$, where $s(i,q)$ is the similarity score between nodes i and q, and $\mathrm{IDCG}_p(q)$ is a normalized factor ensuring the "true" NDCG ordering to be 1.

(5) Ground Truth. (a) To validate similar authors on DBLP, we invite 20+ experts from database and data mining areas to assess the "true" relevance of each retrieved co-authorship. They may also refer to Co-Author Path in Microsoft Academic Search[14] to see "separations" between collaborators. (b) To evaluate similar papers on CitHepTh, we hire 15+ researchers from the physical department for judging the "true" relevance of the retrieved co-citations. Their assessment may hinge on paper contents, H-index, and #-citations in `www.ScienceDirect.com`. For all the ground truths, the final results are rendered by a majority vote of feedbacks.

All experiments are run on a machine powered by an Intel Core(TM) 3.10GHz CPU with 8GB RAM, on Windows 7.

Experimental Results. We next present our findings.

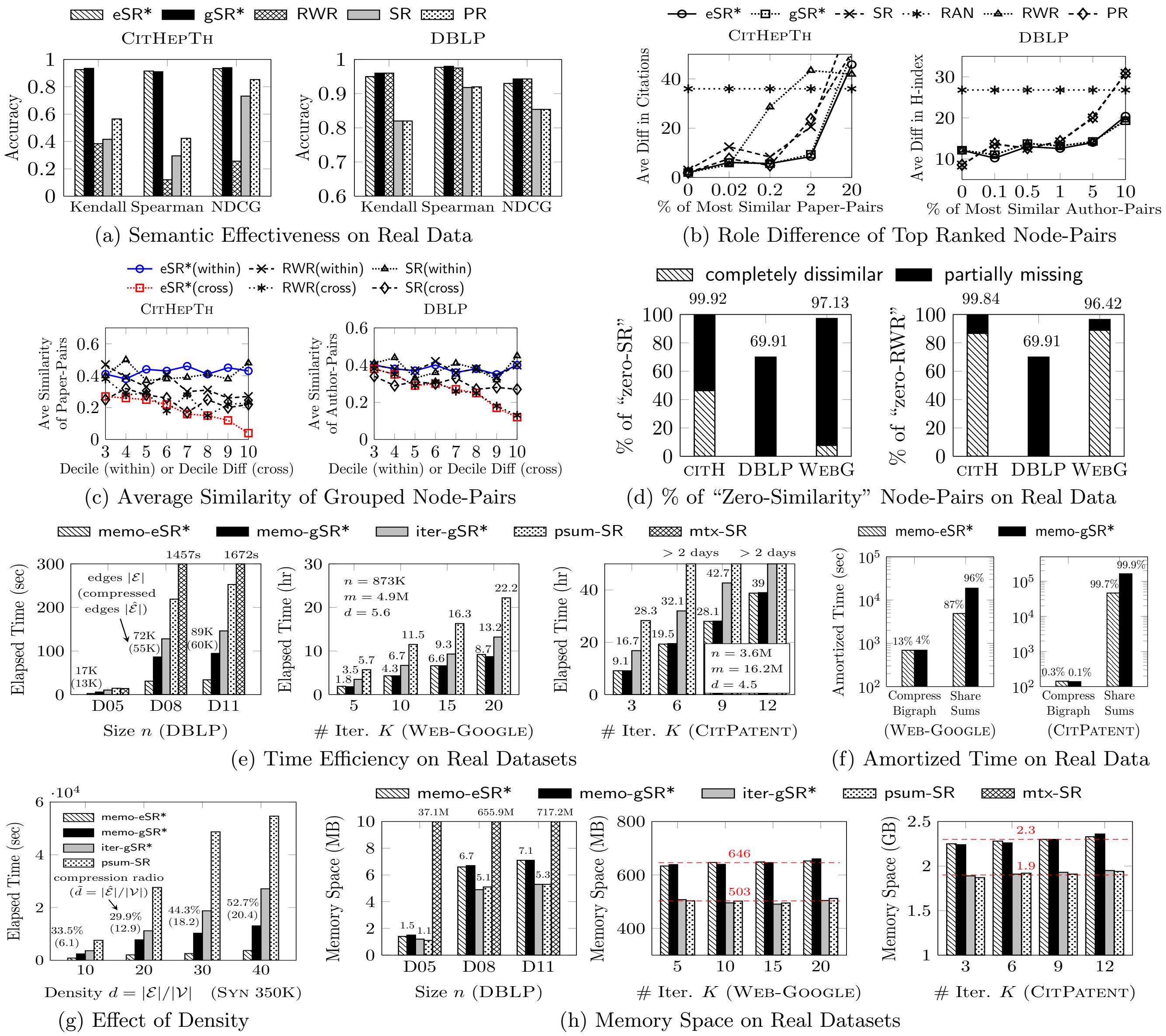
Exp-1: Semantics & Relative Order. We first run the algorithms on *directed* CitHepTh and *undirected* DBLP. By randomly issuing 500+ queries, we evaluate the average semantic accuracy for each algorithm via three metrics (Kendall, Spearman, NDCG). Fig.6(a) depicts the results. (Due to space limits, many case studies are reported in [22] to further exemplify the quantitative results in Fig.6(a).) (1) On CitHepTh, memo-gSR* and memo-eSR* have higher accuracy (*e.g.,* Spearman's $\rho \approx 0.91$) than psum-SR (0.29), RWR (0.12) and psum-PR (0.42) on average, *i.e.,* the semantics of SimRank* is effective. This is because SimRank* considers *all* in-link paths for assessing similarity, whereas SimRank and RWR, respectively, counts only limited *symmetric* and *unidirectional* paths. (2) On DBLP, the accuracy of RWR is the same as memo-gSR* and memo-eSR*, due to the *undirectedness* of DBLP. This tells us that, regardless of edge directions, both SimRank* and RWR count the path of *all* lengths, as opposed to SimRank considering only the *even*-length paths. Likewise, psum-PR and psum-SR produce the same results on undirected DBLP. (3) On both datasets, memo-gSR* and memo-eSR* keep almost the same accuracy, implying that the relative order of the geometric SimRank* is well maintained by its exponential counterpart.

Fig.6(b) further validates that node-pairs with *high* SimRank* scores do have *similar* roles. On CitHepTh, we use #-citation as a proximity measure for co-citation role; on DBLP, we use H-index for coauthor role, since if a paper is highly cited, it will increase the H-index of every co-author. From the results, we see that on CitHepTh, for the top 2% similar paper-pairs, the average difference in their #-citation is 8 for memo-gSR* and memo-eSR*, which is lower than psum-SR (21), psum-PR (24), RWR (43), and the random-pair difference RAN (38). A *lower* average difference in #-citation (*resp.* H-index) indicates that papers (*resp.* authors) are reliably *similar.* As we increase the search to top 20% similar paper-pairs on CitHepTh, SimRank* can constantly find *reliable* similarity, whereas SimRank converges to random scoring. Thus, node-pairs with higher SimRank* scores will have similar roles. A similar result is shown on DBLP.

Fig.6(c) confirms that nodes with similar roles do have high SimRank* scores. On CitHepTh (*resp.* DBLP), we group the papers (*resp.* authors) into 10 roles based on the #-citation (*resp.* H-index), from top 10% to bottom 10%.

[13] http://www.cse.psu.edu/~madduri/software/GTgraph/index.html

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



(a) Semantic Effectiveness on Real Data

(b) Role Difference of Top Ranked Node-Pairs

(c) Average Similarity of Grouped Node-Pairs

(d) % of "Zero-Similarity" Node-Pairs on Real Data

(e) Time Efficiency on Real Datasets

(f) Amortized Time on Real Data

(g) Effect of Density

(h) Memory Space on Real Datasets

Figure 6: Performance Evaluation on Real and Synthetic Datasets

For each node-pair, if two nodes are within the same role, we average out their similarity score for this role. We also average out #-node-pairs not within the same role (across roles). We see that *e.g.,* on DBLP, the average SimRank* similarity within the same role is stable around 0.4, in contrast with SimRank fluctuating between 0.35 and 0.45, due to many dissymmetric paths completely neglected by SimRank. For the author-pairs across roles, the x-axis denotes the difference of role decile for two authors in a pair. The decreasing line of memo-eSR* and RWR indicates that role similarity correctly decreases as H-index gets less similar. For psum-SR, the average across-role similarity is round 0.3, approaching random scoring. This tells that SimRank* scores are more reliable than others to reflect nodes with similar roles. The result is more pronounced on CitHepTh.

Fig.6(d) shows the "zero-similarity" issues for SimRank and RWR *commonly* exist in real graphs. The results on *e.g.,* CitHepTh show that more than 95% of node-pairs have "zero-SimRank" issues, among which about 40% are assessed as "completely dissimilar" (*i.e.,* SimRank=0), and about 55% have "partially missing" issue (SimRank $\neq$0, but miss the contributions of the dissymmetric in-links paths). It shows the necessity for our revision of SimRank and RWR.

Exp-2: Time Efficiency. We next evaluate (1) the CPU time of SimRank* on real data, and (2) the impact of graph density on CPU time on synthetic data.

Fixing accuracy ϵ = .001 on DBLP, and varying K on Web-Google and CitPatent, we compare the CPU time of the five algorithms. The results are shown in Figure 6(e), telling the following. (1) In all the cases, memo-gSR* and memo-eSR* outperform iter-gSR*, psum-SR and mtx-SR, *i.e.,* our fine-grained memoization approach is efficient. Indeed, mtx-SR is the slowest on D05, D08, D11 due to its cost-inhibitive SVD. On Web-Google, memo-gSR* (memo-eSR*) is on average 1.6X and 2.6X faster than iter-gSR* and psum-SR, respectively. On CitPatent, the speedup of memo-gSR* (memo-eSR*) is on average 1.7X and 3.1X better than iter-gSR* and psum-SR, respectively. When $K \geq 6$, psum-SR takes too long to finish computations in two days on large CitPatent, which is practically unacceptable. In contrast, memo-gSR* (memo-eSR*) just needs about 19.5 hours for $K = 6$. This is because SimRank* takes a simpler form than SimRank, in which one just needs to compute one *single* summation per iteration, in contrast to a *double* summation of psum-SR. (2) Given $\epsilon = .001$ on DBLP, the speedup of memo-eSR* is more pronounced, 6.8X, 4.2X, 2.7X faster than psum-SR, iter-gSR*, memo-gSR* on average, respectively. This is because the closed matrix form of memo-eSR* accelerates the convergence of SimRank*, thus yielding less iterations for attaining the same accuracy ϵ.

Figure 6(f) further shows the amortized time for each phase of memo-eSR* and memo-gSR* on Web-Google and CitPatent (given $\epsilon = .001$), with x-axis being two phases. From the results, (1) for memo-eSR* and memo-gSR*, the time for "Compress Bigraph" is about one order of magnitude less than the time for "Share Sums" on Web-Google, and 2.5 orders of magnitude less on CitPatent. This tells that the preprocessing does not incur much extra time, confirming our complexity analysis in Subsect. 4.3. (2) "Compress Bigraph" takes up larger portions (13% on Web-Google, and 0.3% on CitPatent) in the total time of memo-eSR*, than those (4% on Web-Google, and 0.1% on CitPatent) in the total time of memo-gSR*. This is because memo-eSR* and memo-gSR* takes (almost) the same time for "Compress Bigraph", whereas, for "Share Sums", memo-eSR* needs less time (3.8X on Web-Google, 3.5X on CitPatent) than memo-gSR*, due to the convergence speedup of memo-eSR*.

Fixing $n = 350$K, varying m from 3.5M to 14M on synthetic data, Figure 6(g) shows the impact of graph density $d = m/n$ on CPU time. The results show that (1) given $\epsilon = .001$, memo-eSR* outperforms memo-gSR*, iter-gSR*, and psum-SR by 3.5X, 6.1X, and 14X speedups, respectively, as m increases. (2) The speedups of memo-eSR* and memo-gSR* are sensitive to graph density. This is because when graphs become denser, there is a higher likelihood that in-neighbor sets will overlap one another for fine-grained partial sums sharing. The biggest speedups are observed for higher density — with nearly 1.5 orders of magnitude speedup at $d = 40$, and its compression ratio is 52.7%. [15]

Exp-3: Memory Space. Lastly, we evaluate the space requirement of memo-eSR* and memo-gSR* against iter-gSR*, psum-SR and mtx-SR on real data. We only use mtx-SR on small DBLP since its memory space will explode on large Web-Google, due to its SVD destroying graph sparsity.

The results are reported in Figure 6(h). We observe that (1) in all the cases, memo-eSR* and memo-gSR* take almost the same space, both of which fairly retain the same orders of magnitude as iter-gSR* and psum-SR. Indeed, both memo-eSR* and memo-gSR* only need 28.2%, 29.3%, and 19.2% more space on average on DBLP, Web-Google, and CitPatent, respectively, as compared with iter-gSR* and psum-SR. The extra space of memo-eSR* and memo-gSR* is used for fine-grained SimRank* memoization. This tells that memo-eSR* and memo-gSR* do not need to sacrifice much space for achieving high time efficiency. (2) On DBLP, memo-eSR* and memo-gSR* require far less space than mtx-SR by at least one order of magnitude, since mtx-SR using SVD may produce very dense matrices. (3) On Web-Google and CitPatent, the space of memo-eSR* and memo-gSR* is stable as K grows because the memoized partial sums are immediately released after each iteration.

[15] Here, the compression ratio is defined by $(1 - \frac{\tilde{m}}{m}) \times 100\%$, where $\tilde{m}$ is the number of edges in the compressed graph $\hat{\mathcal{G}}$.

6. CONCLUSION

We have proposed SimRank*, a refinement of SimRank, for effectively assessing link-based similarities. In contrast to SimRank only considering contributions of *symmetric* in-link paths, SimRank* can tally contributions of *all* in-link paths between two nodes, thus resolving the "zero-SimRank" issue for semantic richness. We have also converted the series form of SimRank* into two elegant forms: the geometric SimRank* and its exponential variant, both of which look even simpler than SimRank, yet without suffering from increased computational cost. Finally, we have developed a fine-grained memoization strategy via edge concentration, with an efficient algorithm speeding up SimRank* computation from $O(Knm)$ to $O(Kn\tilde{m})$ time, where $\tilde{m}$ is generally much smaller than m. Our experimental results on real and synthetic data show richer semantics and higher computation efficiency of SimRank*.

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] P. Berkhin. Survey: A survey on PageRank computing. *Internet Mathematics*, 2(1), 2005.

[3] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. V. Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Rev.*, 46(4), 2004.

[4] R. Brualdi and D. Cvetkovic. *A Combinatorial Approach to Matrix Theory and Its Applications*. Discrete Mathematics and Its Applications. Taylor & Francis, 2008.

[5] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM*, 2008.

[6] S. Chakrabarti. Dynamic personalized PageRank in entity-relation graphs. In *WWW*, pages 571–580, 2007.

[7] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, 2005.

[8] G. He, H. Feng, C. Li, and H. Chen. Parallel SimRank computation on large graphs with iterative aggregation. In *KDD*, 2010.

[9] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *KDD*, 2002.

[10] R. Jin, V. E. Lee, and H. Hong. Axiomatic ranking of network role similarity. In *KDD*, 2011.

[11] M. M. Kessler. Bibliographic coupling between scientific papers. *Amer. Doc.*, 14(1):10–25, 1963.

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

[13] E. A. Leicht, P. Holme, and M. E. J. Newman. Vertex similarity in networks. *Physical Review E*, 73(2), 2006.

[14] 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.

[15] X. Lin. On the computational complexity of edge concentration. *Discrete Applied Mathematics*, 101(1-3):197–205, 2000.

[16] Z. Lin, M. R. Lyu, and I. King. MatchSim: A novel similarity measure based on maximum neighborhood matching. *Knowl. Inf. Syst.*, 32(1), 2012.

[17] D. Lizorkin, P. Velikhov, M. N. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for SimRank computation. *PVLDB*, 1(1), 2008.

[18] 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.

[19] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, 2006.

[20] 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.

[21] X. Yin, J. Han, and P. S. Yu. LinkClus: Efficient clustering via heterogeneous semantic links. In *VLDB*, 2006.

[22] W. Yu, X. Lin, W. Zhang, L. Chang, and J. Pei. More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks. http://www.cse.unsw.edu.au/˜weirenyu/pubs/20130428.pdf UNSW-CSE-TR-201304, University of New South Wales, 2013.

[23] P. Zhao, J. Han, and Y. Sun. P-Rank: A comprehensive structural similarity measure over information networks. In *CIKM*, 2009.

[24] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural / attribute similarities. *PVLDB*, 2(1), 2009.