

Research

Fast computation of General SimRank on heterogeneous information network

Chuanyan Zhang¹ · Xiaoguang Hong^{2,3} · Yongqing Zheng^{2,3}

Received: 20 September 2022 / Accepted: 2 May 2024

Published online: 21 May 2024

© The Author(s) 2024 [OPEN](#)

Abstract

Similarity computation is a fundamental aspect of information network analysis, underpinning many research tasks including information retrieval, clustering, and recommendation systems. General SimRank (GSR), an extension of the well-known SimRank algorithm, effectively computes link-based global similarities incorporating semantic logic within heterogeneous information networks (HINs). However, GSR inherits the recursive nature of SimRank, making it computationally expensive to achieve convergence through iterative processes. While numerous rapid computation methods exist for SimRank, their direct application to GSR is impeded by differences in their underlying equations. To accelerate GSR computation, we introduce a novel approach based on linear systems. Specifically, we transform the pairwise surfer model of GSR on HINs into a new random walk model on a node-pair graph, establishing an equivalent linear system for GSR. We then develop a fast algorithm utilizing the local push technique to compute all-pair GSR scores with guaranteed accuracy. Additionally, we adapt the local push method for dynamic HINs and introduce a corresponding incremental algorithm. Experimental results on various real datasets demonstrate that our algorithms significantly outperform the traditional power method in both static and dynamic HIN contexts.

Keywords Similarity · General SimRank · Fast computation · Linear system · Heterogeneous information network

List of symbols

$G = (V, E)$	Graph with nodes V and edges E
n, m	$n = V , m = E $
$G^2 = (V^2, E^2)$	Node-pair graph of G
N, M	$N = V^2 , M = E^2 $
$T_G = (A, R)$	Network schema of G
R_{ij}	Semantic relation for A_i to $A_j, A_i, A_j \in A$
W_{ij}	Adjacency matrix about R_{ij}

Xiao Guang Hong contributed equally to this work.

✉ Chuanyan Zhang, chuanyan_zhang@sina.cn; Xiaoguang Hong, hxg@sdu.edu.cn; Yongqing Zheng, zhengyongqing@dareway.com.cn | ¹College of Information Science and Engineering, Qilu Normal University, Jinan 250200, Shandong, China. ²Software School, Shandong University, Jinan 250101, Shandong, China. ³Dareway Software Co., Ltd., Jinan 250000, Shandong, China.



\mathcal{G}^2	GSR node-pair graph
Q_{ij}	Similarity transmission matrix on \mathcal{G}^2 about $R_{i,j}$

1 Introduction

In recent years, many information systems have employed complex graph models to represent data, where vertices symbolize data objects and edges delineate the relationships among them. This structure is commonly referred to as an information network [1]. A fundamental task in analyzing these networks involves effectively capturing the similarities between objects. This challenge arises in various applications, including recommender systems [2], link prediction, neural graph learning and so on [3, 4].

To quantify object similarity, several link-based measures have been proposed, such as Random Walk with Restart [5], Personalized PageRank [6], and SimRank [7]. Among these, SimRank is particularly noted for its intuitive appeal and robust theoretical underpinnings: (1) it posits that “two objects are similar if they are referenced by similar objects”; (2) it is recursively defined based on a “pairwise surfer model,” leveraging the global topology of the graph. However, SimRank is tailored for homogeneous information networks, which consist of a single type of vertices and edges, and thus, it cannot be directly applied to heterogeneous information networks (HINs) that encompass a variety of objects and relationships [8].

General SimRank (GSR) [9], an extension of SimRank, adapts the original formula to accommodate HINs. It maintains the core principle that “two objects are similar if they are referenced by similar objects through the same semantic relations,” and it stipulates that “similarities exist only between objects of the same type.” GSR represents a link-based global similarity measure that maintains semantic consistency. Demonstrating its utility, GSR has been applied to recommendation systems and the exploration of interpretable machine learning methods [2]. The theoretical basis of GSR, known as the *constrained expected meeting distance*, retains the recursive nature of SimRank and requires numerous iterations to achieve convergence in its power solution. This results in high time and space complexities, which restrict its application in large-scale HINs. In this paper, we focus on developing a rapid all-pair solution to compute GSR scores for all node pairs within a HIN, aiming to overcome these limitations.

1.1 General SimRank

Given two nodes u and v of a graph $G = (V, E)$, the similarity of between them is denoted as $s(u, v)$. If G is a homogeneous network, SimRank computes $s(u, v)$ as follows [7]:

$$s(u, v) = \begin{cases} 1 & u = v \\ \frac{c}{|l(u)||l(v)|} \sum_{x \in l(u), y \in l(v)} s(x, y) & u \neq v \end{cases} \quad (1)$$

where $l(u)$ denotes the set of in-neighbors of u and $c \in (0, 1)$ is a decay factor (generally, $c = 0.8$). Obviously, SimRank calculates the similarities only in term of the structural information of a graph.

If G is a HIN whose network schema is $T_G = (A, R)$ with two mapping functions: $\phi : V \rightarrow A$; $\varphi : E \rightarrow R$, and $\phi(u) = \phi(v) = A_i$, GSR calculates $s(u, v)$ as follows [9]:

$$s(u, v) = \begin{cases} 1 & u = v \\ \frac{c}{|l(u)||l(v)|} \sum_{R_{j,i} \in R} \sum_{x \in l_j(u)} \sum_{y \in l_j(v)} s(x, y) & u \neq v \end{cases} \quad (2)$$

where $R_{j,i}$ is the semantic relation from node type A_j to A_i , and $l_j(u)$ is the set of in-neighbors of type A_j of u . If $|A| = 1$ and $|R| = 1$, Eq. (2) is equivalent to Eq. (1). Specifically, Eq. (2) is the normal form of GSR without considering multiple semantic relations and edge weights.

Let S_i be the similarity matrix about nodes of type A_i , i.e. $S_i(u, v) = s(u, v)$, the fast computation problem of General SimRank is defined as follows:

Definition 1 (Exact All-pair GSR Computation) Given a HIN $G(V, E)$ with the network schema $T_G(A, R)$ and an error bound ε , compute the approximate similarity matrix \hat{S}_i such that $\forall u, v, \|S_i(u, v) - \hat{S}_i(u, v)\| < \varepsilon$, where S_i is the ground true GSR matrix for all-pair nodes of type A_i and $S_i(u, v) = s(u, v)$.

1.2 Motivation

Due to the recursive nature of General SimRank, the *naive iterative method* (a.k.a. power method) costs high expensive both in time and space. To our best knowledge, there's no related works concentrating on the fast computation problem of GSR. Meanwhile, there exist many works which focus on the optimization techniques for SimRank computation [10–12]. Since SimRank can be considered as a special form of GSR, these techniques would help us in designing fast GSR computation method. Generally, the methods of SimRank can be divided into 3 categories: (1) *power method*; (2) *random walk method*; (3) *non-iterative method*. The power methods improve efficiency through matrix compression or pruning based on the naive iterative framework. They can calculate all-pair similarities with high accuracy. However, the improvement of efficiency is very limited and it is not suitable for dynamic networks [13]. Random walk methods mainly aim to solve the single source problem, e.g. top-k query [10, 14, 15]. It models the SimRank score $s(u, v)$ as the first meeting expectation between 2 groups of backward random walks starting from u and v respectively [16–18]. Non-iterative methods are the main research direction for SimRank computation. They try to design and solve a linear system of SimRank, which mainly starts from the matrix form of SimRank,

$$S = cW^T S W \vee I, \quad (3)$$

where S is the similarity matrix of SimRank, W is the column normalized adjacency matrix of graph, I is the unit matrix, and \vee denotes the entry-wise maximum. Due to the correction operator “ \vee ”, it is very difficult to design a linear recursive formula. Thus, many existing works relax Eq. (3) to an approximate linear system, i.e., $S = cW^T S W + (1 - c)I$, and then utilize different methods for accelerating SimRank computation [19–22]. Thus, it is difficult to guarantee the accuracy of the similarities. Recently, two works [11, 12] propose the equivalent linear equations based on node-pair graph separately which inspire us to build the linear system of GSR.

In this paper, we concentrate on fast all-pair GSR computation through linear system. However, we find that all existing methods of SimRank cannot be utilized to GSR since they have different matrix forms.

Theorem 1 (Matrix Form of General SimRank) Given a HIN $G(V, E)$ and the network schema $T_G(A, R)$, in matrix notation, General SimRank of Eq. (2) can be formulated as

$$S_i = \left(\sum_{R_{j,i} \in R} cW_{ji}^T S_j W_{ji} \right) \vee I \quad (4)$$

where W_{ji} is the adjacency matrix from nodes of A_j to nodes of A_i . If $e(x, u) \in E$, $\phi(x) = A_j$ and $\phi(u) = A_i$, $W_{ji}(x, u) = \frac{1}{|I(u)|}$; otherwise, $W_{ji}(x, u) = 0$.

Proof Please refer to the proof in Appendix. □

1.3 Contributions

In this paper, we propose a novel method to calculate all-pair GSR scores effectively with an accuracy guarantee. Firstly, we derive a *equivalent linear system* of GSR based on the *node-pair graph* G^2 and study the relationship between GSR and PageRank. Then, we design a novel solution for the linear system of GSR based on *local push* algorithm. Compared with other methods for solving linear systems, the local push algorithm only involves the node pairs with large residuals in similarity computation. This *push-on-demand* manner not only avoids updating the whole solution at each iteration, but is also memory-efficient. Given an error bound ε , the iterative method has $O(KM)$ time complexity and $O(N)$ space complexity where K is the number of iterations, M and N are the number of edges and nodes of G^2 respectively. Based on the previous works [9, 13], $K = \log_{c\alpha} \varepsilon$, $\alpha \in (0, 1)$. Our solution has $O\left(\frac{\bar{s}}{(1-c)\varepsilon} M\right)$ time complexity and $O(nnz(N) + m + n)$ space cost where \bar{s} is the average GSR score, $nnz(N)$ is the number of non-zero element in all node pairs, and m and n

denote the edge and node number of G respectively. Especially, for each iteration of the power method, $O(M) = \Omega(M)$. Meanwhile, the lower bound of local push is far more less than $O(M)$. Further, We propose an incremental algorithm to calculate GSR scores in dynamic HINs, using current estimates and residuals of our local push method. In summary, we have made the following contributions:

- We propose a novel and fast method to calculate all pair GSR scores on HIN based on linear system.
- Based on the node pair graph, we design the equivalent linear system of GSR and solve it through local push method efficiently.
- An incremental algorithm is proposed to computer GSR scores on dynamic HINs.
- Experiments on several public datasets demonstrate that our methods significantly outperform the original power method.

2 Preliminaries

To make the paper self-contained, we then give some basic definitions, including HIN, network schema and node-pair graph, and then briefly introduce some useful operators on vector and matrix.

Definition 2 (Heterogeneous Information Network) An information network is defined as a directed graph $G = (V, E)$ with an object type mapping function $\phi : V \rightarrow A$ and a link type mapping function $\varphi : E \rightarrow R$, where each object $v \in V$ belongs to one particular object type $\phi(v) \in A$ and each link $e \in E$ belongs to a particular relation $\varphi(e) \in R$. When the types of objects $|A| > 1$ or the types of relations $|R| > 1$, the network is called *heterogeneous information network*; otherwise, it is a *homogeneous information network*.

Definition 3 (Network Schema) The network schema is a meta template for a heterogeneous network $G = (V, E)$ with the object type mapping $\phi : V \rightarrow A$ and the link mapping $\varphi : E \rightarrow R$, which is a directed graph defined over object types A , with edges as relations from R , denoted as $T_G = (A, R)$.

Generally, network schema provides a meta level (i.e., schema-level) description for better understanding the complex HIN. Meanwhile, node-pair graph provides a higher perspective to the *pairwise surfer model* of SimRank and GSR and it is defined as follows.

Definition 4 (Node-pair Graph) Given a directed graph $G = (V, E)$, its corresponding node-pair graph is denoted as $G^2 = (V^2, E^2)$ where $E^2 = \{e((a, b), (c, d)) | e(a, c), e(b, d) \in E\}$, $V^2 = \{(a, b) | a, b \in V\}$.

Definition 5 (Kronecker Product) Let $B \in \mathbb{R}^{p \times q}$, $C \in \mathbb{R}^{m \times n}$, the Kronecker Product of A and B is defined as:

$$B \otimes C = \begin{pmatrix} b_{11}C & \cdots & b_{1q}C \\ \vdots & \ddots & \vdots \\ b_{p1}C & \cdots & b_{pq}C \end{pmatrix}$$

Definition 6 (Vec Operator) For any matrix $D \in \mathbb{R}^{m \times n}$, \mathbf{r}_i denotes the row vector of D . Then $\text{vec}(D) \in \mathbb{R}^{1 \times mn}$ is defined as: $\text{vec}(D) = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_m)$.

Definition 7 (Uvec Operator) For any row vector set $\{\mathbf{r}_i | \mathbf{r}_i \in \mathbb{R}^{1 \times n_i}, i, n_i \in \mathbb{N}^*\}$, the operator $U\text{vec}\{\mathbf{r}_i\} \in \mathbb{R}^{1 \times \sum n_i}$ is defined as: $U\text{vec}\{\mathbf{r}_i\} = (\mathbf{r}_1, \mathbf{r}_2, \dots)$.

Obviously, the Kronecker product of two matrix B and C is an $pm \times qn$ matrix; the *vec* operator is to translate a matrix into a row vector and the *Uvec* operator is to combine row vectors of any size into one row vector.

3 The linear system of general SimRank

In this section, we propose a linear system of GSR based on the node-pair graph, then study its properties and compare it with P-PageRank [6].

3.1 General SimRank on node-pair graph

Based on Eq. (2), the GSR score $s(u, v)$ relies on the scores of all node pairs (x, y) , $x \in I_j(u)$, $y \in I_j(v)$, i.e., $s(x, y)$, which is to say “similarity propagates from (x, y) to (u, v) .” This transmission from the in-neighbor pair to the target node pair requires two edges separately on the original HIN G . While only one edge is needed for each transmission on node-pair graph since every node pair is modeled as one based on the Definition 4. This observation motivates us construct the node-pair graph of General SimRank.

Definition 8 (Node-pair Graph of General SimRank) Given a HIN $G = (V, E)$, the corresponding Node-pair Graph General SimRank, denoted as $\mathcal{G}^2 = (V^2, E^2)$, is defined as follows:

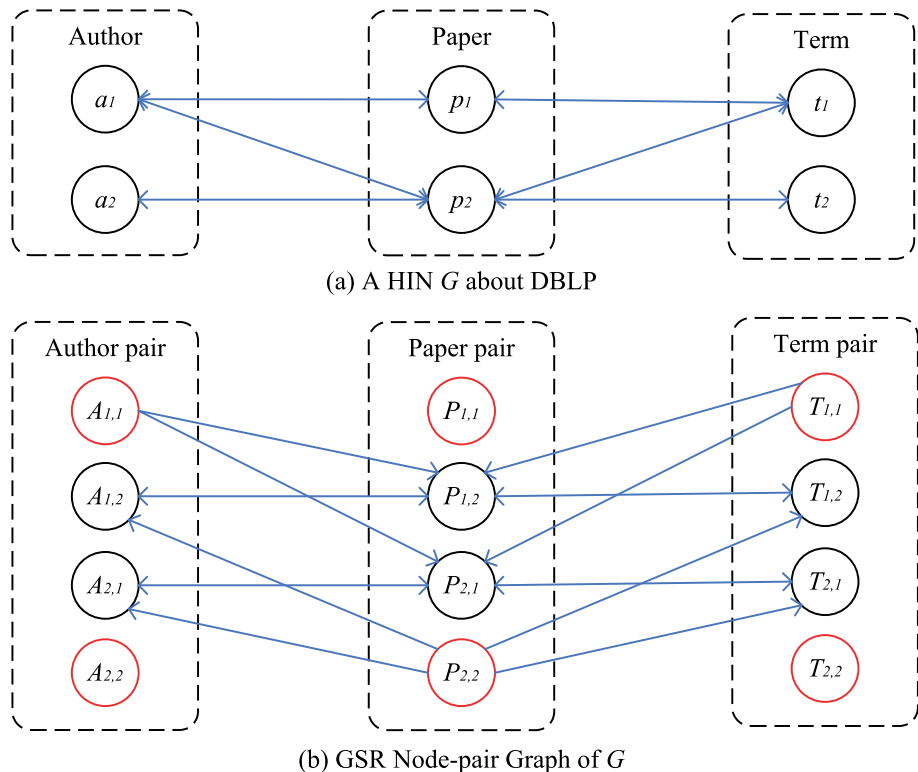
$$V^2 = \{(a, b) | a, b \in V \wedge \phi(a) = \phi(b)\}$$

$$E^2 = \{e((a, b), (c, d)) | e(a, c), e(b, d) \in E \wedge c \neq d\}.$$

Distinct from SimRank node-pair graph, each node of \mathcal{G}^2 represents a node pair (a, b) , $\phi(a) = \phi(b)$, which reflects the intuition that “similarity only exists between nodes of the same type.” In addition, we also declare that $c \neq d$ when we create an edge from node $n(a, b)$ to node $n(c, d)$ of \mathcal{G}^2 , which ensures that $s(u, u) \equiv 1$ for any $u \in V$ based on Eq. (2). Since the the GSR score $s(u, v)$ of node $n(u, v)$ is raised by its in-neighbors, i.e., $n(x, y)$, we should remove the all the edges pointing to the nodes of type $n(u, u)$ on \mathcal{G}^2 so that no similarity could be transmitted to them. Since these nodes only propagate similarities, not receive any similarity, we call them Source Nodes.

A toy HIN G and its \mathcal{G}^2 about DBLP are shown in Figure. 1. Since only the node pair with two nodes of the same type can be modeled as new nodes of GSR node-pair graph, we could construct 4 nodes on \mathcal{G}^2 : $\{n(a_1, a_1), n(a_1, a_2), n(a_2, a_1), n(a_2, a_2)\}$ for the two authors $\{a_1, a_2\}$ of G and the rest are the same. Then, we link the nodes of \mathcal{G}^2 . According to the definition of

Fig. 1 A toy example of HIN G about DBLP and its corresponding GSR node-pair graph \mathcal{G}^2



SimRank node-pair graph, there will be an edge $e((a_1, a_1), (p_1, p_1))$ for $e(a_1, p_1)$ of G . But in \mathcal{G}^2 , we remove it. Consequently, there's no edge linking to any source node and its GSR score is always equal to 1. Finally, we can say that "the GSR score of any node on \mathcal{G}^2 is raised by its in-neighbors."

Based on the observation of GSR on \mathcal{G}^2 , we can rewrite the Eq. (2) of GSR as follows. Firstly, we separate V^2 of \mathcal{G}^2 into 2 disjoint sets. Let $V_D^2 = \{n_k = (u, v) | n_k \in V^2 \wedge u \neq v\}$, and $V_S^2 = V^2 - V_D^2$, i.e., the set of source nodes. Suppose each node $n_k = n(u, v) \in V^2$ has a value $s(n_k)$ and $s(n_k) = s(u, v)$. Then, $\phi(n_k) = A_i$ if and only if $\phi(u) = \phi(v) = A_i$. Finally, the Eq. (2) is rewritten as:

$$s_i(n_k) = \begin{cases} 1 & n_k \in V_S^2 \\ \frac{c}{|l(u)||l(v)|} \sum_{R_{j,i} \in R} \sum_{n_l \in l_j(n_k)} s(n_l) & n_k \in V_D^2 \end{cases} \tag{5}$$

where $l_j(n_k)$ is the in-neighbor set of n_k of type A_j , i.e., $l_j(n_k) = \{n_l | e(n_l, n_k), \phi(n_l) = A_j\}$. For example, the similarity of $A_{1,2}$ comes from $P_{1,2}$ and $P_{2,2}$ in Fig. 1; while for the source nodes, likely $A_{1,1}, A_{2,2}, P_{1,1}$ and so on, they cannot receive any similarity since they do not have any in-neighbor. Note that \mathcal{G}^2 and G share the same network schema based on the definition of GSR node-pair graph. Thus, for any node type $A_i \in A$ of T_G , there exists the same type A_i of \mathcal{G}^2 , and the semantic relations are the same too.

3.2 Equivalent linear system

In this section, we rewrite Eq. (5) into the matrix form and finally derive a equivalent linear system of General SimRank.

First, we construct the adjacency matrix W_{ji} from nodes of type A_j to nodes of type A_i on G^2 . Let $\phi(a) = A_j, \phi(b) = A_i$. If $\exists e(a, b) \in E, W_{ji}(a, b) = \frac{1}{|l(b)|}$; otherwise, $W_{ji}(a, b) = 0$. For any $n_k = n(u, v), n_k \in V_D^2, \phi(n_k) = A_i$, we set

$$Q_{ji}(n_l, n_k) = \frac{1}{|l(u)||l(v)|} = W_{ji}(x, u)W_{ji}(y, v)$$

where $n_l = (x, y), \phi(n_l) = A_j$.

For the node $n_k \in V_S^2$, we set

$$Q_{ji}(:, n_k) \equiv 0$$

which conforms to the definition that "no edge connects to source nodes on \mathcal{G}^2 ."

Finally, we calculate the similarity transmission matrix as follows:

$$Q_{ji} = \psi(W_{ji} \otimes W_{ji}) = \begin{pmatrix} \underbrace{W_{ji}(1, 1)W_{ji}(1, 1)}_{=0} & \dots & 0 & \dots \\ \vdots & \dots & 0 & \dots \\ \underbrace{W_{ji}(x, 1)W_{ji}(y, 1)}_{=0} & \dots & \underbrace{W_{ji}(x, k)W_{ji}(y, k)}_{=0} & \dots \\ \vdots & \ddots & \vdots & \dots \\ \underbrace{W_{ji}(p, 1)W_{ji}(p, 1)}_{=0} & \dots & 0 & \dots \end{pmatrix} \tag{6}$$

where $Q_{ji} \in \mathbb{R}^{p \times q}$ and the function ψ is

$$\psi(W_{ji} \otimes W_{ji}) = \psi(W_{ji}(x, u)W_{ji}(y, v)) = \begin{cases} 0 & u = v \\ \frac{1}{|l(u)||l(v)|} & u \neq v \end{cases} \tag{7}$$

Taking Fig. 1 as an example, let $\{A_1, A_2, A_2\}$ denote $\{Author, Paper, Term\}$. Then, we have the adjacency matrix W_{12} of G and the corresponding matrix Q_{12} of \mathcal{G}^2 as follows:

$$W_{12} = \begin{pmatrix} 0.5 & 0.25 \\ 0 & 0.25 \end{pmatrix}, Q_{12} = \begin{pmatrix} 0 & 0.125 & 0.125 & 0 \\ 0 & 0.125 & 0 & 0 \\ 0 & 0 & 0.125 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where $W_{12}(1, 2) = \frac{1}{|I(p_2)|} = 0.25, Q_{12}(2, 1) = W_{12}(1, 1)W_{12}(2, 1) = 0, Q_{12}(1, 2) = W_{12}(1, 1)W_{12}(1, 2) = 0.125$ and so on.

Further, with the matrix Q_{ij} , we can formulate the intuition of GSR into a linear system, which says “the GSR score of any node on \mathcal{G}^2 is raised by its in-neighbors.”

Theorem 2 Given the node-pair graph $\mathcal{G}^2 = (V^2, E^2)$ of a HING = (V, E) with $T_G = (A, R)$ and $Q_{ij}, 1 \leq i, j \leq |A|$, the matrix form of Eq. (5):

$$Vec(S_i) = c \sum_{R_{ij} \in R} Vec(S_i)Q_{ij} + Vec(I_i) \tag{8}$$

where I_i is a identity matrix of the same size as S_i .

Proof Suppose that $\bar{x}_i = Vec(S_i), \bar{I}_i = Vec(I_i)$ and $s(n_k) = \bar{x}_i(k)$. If $n_k \in V_S$, we have:

$$s(n_k) = \bar{x}_i(k) = \bar{I}_i(k) = 1$$

If $n_k \in V_D$, based on Eq. (5), we have:

$$\begin{aligned} s(n_k) &= c \sum_{R_{ij} \in R} \sum_{n_l \in I_j(n_k)} \frac{1}{|I(u)||I(v)|} s(n_l) \\ &= c \sum_{R_{ij} \in R} \sum_{n_l \in I_j(n_k)} Q_{ij}(l, k) s(n_l) \\ &= c \sum_{R_{ij} \in R} Vec(S_i)Q_{ij}(:, k) \end{aligned}$$

Since all the source nodes have no in-neighbors in \mathcal{G}^2 , we can get:

$$s_i(n_k) = \begin{cases} c \sum_{R_{ij} \in R} \underbrace{Vec(S_i)Q_{ij}(:, k)}_{=0} + \underbrace{\bar{I}_i(k)}_{=1} & n_k \in V_S^2 \\ c \sum_{R_{ij} \in R} \underbrace{Vec(S_i)Q_{ij}(:, k)}_{=0} + \underbrace{\bar{I}_i(k)}_{=0} & n_k \in V_D^2 \end{cases} \tag{9}$$

Finally, we have the matrix form of Eq. (5). □

For example, if we calculate the similarity between p_1 and p_2 in Fig. 1, denoted as $s_2(1, 1)$, we have $s_2(1, 1) = Vec(S_2)(1) = \bar{I}_i(1) = 1$ since $Vec(S_2)Q_{12}(:, 1) = 0$ and $Vec(S_2)Q_{32}(:, 1) = 0$. When we calculate $s_2(1, 2)$, only $Q_{12}(:, 2)$ and $Q_{32}(:, 2)$ are left to calculate $Vec(S_2)(2)$ due to $\bar{I}_i(2) = 0$.

The Eq. (8) is already a linear system for $\{S_i\}$. However, our problem studied in this paper is to computer all-pair GSR scores. Thus, we propose the complete linear form of GSR. First, we use the $Uvec$ operator to combine all the node vectors and unit vectors. Let $Z = |A|, \bar{x} = Uvec\{Vec(S_i) | 1 \leq i \leq Z\}$ and $\bar{I} = Uvec\{Vec(I_i) | 1 \leq i \leq Z\}$. Then, we build the complete matrix Q based on $\{Q_{ij}\}$ as follows:

$$Q = \begin{pmatrix} Q_{11} & \cdots & Q_{1Z} \\ \vdots & \ddots & \vdots \\ Q_{Z1} & \cdots & Q_{ZZ} \end{pmatrix} \tag{10}$$

where $Q_{ij} = 0$ if $R_{ij} \notin R$. We call Q the *similarity transmission matrix* of \mathcal{G}^2 .

Theorem 3 (The Linear System of General SimRank) *The General SimRank all-pair similarities \vec{x} is the solution of the following linear system:*

$$\vec{x} = c\vec{x}Q + \vec{I} \tag{11}$$

Proof Based on the definition of *Uvec* operator, we have:

$$\begin{aligned} \vec{x} &= (x_1, x_2, \dots, x_z) \\ \vec{I} &= (I_1, I_2, \dots, I_z) \end{aligned}$$

For any x_i , based on the Eq. (10), we have:

$$\vec{x}_i = c\vec{x} \begin{pmatrix} Q_{1i} \\ \vdots \\ Q_{zi} \end{pmatrix} + \vec{I}_i$$

Finally, we can get:

$$\vec{x}_i = c \sum_{j=1}^z \vec{x}_j Q_{ji} + \vec{I}_i$$

Since $Q_{ij} = 0$ if $R_{ij} \notin R$, any GSR \vec{x}_i of Eq. (11) satisfy the GSR form of Eq. (8). Thus, the solution of \vec{x} is also the GSR score vector. □

3.3 Properties of the linear system

In this section, we first show that the linear system of General SimRank is strictly diagonally, thus it has a unique solution. Then, we study the relationship between General SimRank and P-PageRank.

Firstly, we rewrite the linear system of Eq. (11) into

$$(I - cQ^T)\vec{x}^T = \vec{I}^T \tag{12}$$

where I is a unit matrix and Q^T is the transpose matrix of Q .

Next, we give the diagonal dominance property of the linear system.

Theorem 4 $I - cQ^T$ is strictly diagonally dominant.

Proof Let $H = I - cQ^T$.

If $Q(k, k) = 0$, we have $H(k, k) = 1$. Since $\sum_i Q^T(k, i) = 1$ or $\sum_i Q^T(k, i) = 0$, we can get:

$$H(k, k) > c \sum_{i \neq k} Q^T(k, i) = \sum_{i \neq k} |H(k, i)|$$

If $Q(k, k) \neq 0$, we have $H(k, k) = 1 - cQ(k, k)$. Since $1 > c \sum_i Q^T(k, i)$, we have

$$1 - cQ(k, k) > c \sum_i Q^T(k, i) - cQ(k, k) = \sum_{i \neq k} |H(k, i)|$$

Summarily, we have $H(k, k) > \sum_{i \neq k} |H(k, i)|$. □

According to the Levy-Desplanques theorem [11], strictly diagonal dominant matrices are non-singular. Therefore, there always exists a unique solution \vec{x} for Eq. (11).

Furthermore, we study the relationship between General SimRank and PageRank. PageRank is a hyperlink based ranking method for web pages, which is based on the assumption that *the existence of a hyperlink $u \rightarrow v$ implies that u votes for the quality of v* . If we utilize a directed graph $G = (V, E)$ to model the pages and hyperlinks, *the PageRank score of a node is raised by its in-neighbors*, which is the same as the intuition of GSR on the node-pair graph. Personalized PageRank (PPR) enters user preferences by assigning more importance to edges in the neighborhood of certain pages at the user's selection.

Given $G = (V, E)$, let A denote the adjacency matrix of the web-graph with normalized rows, and let \mathbf{r} be the so called *preference vector* including a probability distribution over V . PageRank vector $\overline{\mathbf{pr}}$ is defined as the solution of the following equation

$$\overline{\mathbf{pr}} = c\overline{\mathbf{pr}}A + (1 - c)\overline{\mathbf{q}} \quad (13)$$

If $\overline{\mathbf{q}}$ is uniform over V , then $\overline{\mathbf{pr}}$ is referred to as the *global PageRank vector*. Otherwise, it will be referred to as *personalized PageRank vector*. For example, $\overline{\mathbf{pr}}_t$ is called *individual PageRank vector* for node t with an indicator vector $\overline{\mathbf{r}}_t$ in Eq. (13) where the t^{th} coordinate is 1 and all others are 0.

Given the GSR node-pair graph $\mathcal{G}^2 = (V^2, E^2)$, let $n_l = (u, v)$, $n_k = (x, y)$, and $e(n_k, n_l) \in E^2$. We have $Q(k, l) = \frac{1}{|l(u)||l(v)|}$, while $A(k, l) = \frac{1}{|l(x)||l(y)|}$. Suppose \mathcal{G}' is the reverse graph of \mathcal{G}^2 . We can get the following theorem.

Theorem 5 *The General SimRank $\overline{\mathbf{x}}$ of Eq. (11) is $\frac{\overline{\mathbf{pr}}}{1-c}$ where $\overline{\mathbf{pr}}$ is the Personalized PageRank solution on \mathcal{G}' with $\overline{\mathbf{q}} = \overline{\mathbf{I}}$.*

Proof Since \mathcal{G}' is the reverse graph of \mathcal{G}^2 , we have $A = Q$. The PPR equation on \mathcal{G}' can be rewritten into

$$\overline{\mathbf{pr}} = c\overline{\mathbf{pr}}Q + (1 - c)\overline{\mathbf{I}}$$

Then, we multiply $\frac{1}{1-c}$ on the both sides,

$$\frac{\overline{\mathbf{pr}}}{1-c} = c\frac{\overline{\mathbf{pr}}}{1-c}Q + \overline{\mathbf{I}} \quad (14)$$

We can see that the equation (14) has the same form as equation (11) of GSR. So we have $\overline{\mathbf{x}} = \frac{\overline{\mathbf{pr}}}{1-c}$. \square

4 Local push algorithm for the linear lystem

In this section, we show our local push based algorithm for all-pair General SimRank computation. Given the linear system of Eq. (11), a straightforward approach is to solve the inverse matrix Q . However, this process is very expensive due to the large size of $Q \in \mathbb{R}^{N \times N}$, $N = \sum_i n_i^2$, $n_i = |\{u | u \in V \wedge \phi(u) = A_i\}|$. Based on Theorem 5, there exists a linear relation between the linear system of GSR and PPR. Hence, we can adopt local push, an optimization technique for calculating PPR to solving Eq. (11).

4.1 Local push algorithm

Local push (LP), as a kind of iterative method, constantly updates 2 vectors $\{\overline{\mathbf{p}}, \overline{\mathbf{r}}\}$, where $\overline{\mathbf{p}}$ is the approximate solution and $\overline{\mathbf{r}}$ is called residual vector. On each step, each node pushes its residual to its neighbors [23]. Given Eq. (11) on \mathcal{G}^2 , the corresponding residual vector $\overline{\mathbf{r}}$ is:

$$\begin{aligned} \overline{\mathbf{r}} &= (\overline{\mathbf{x}} - \overline{\mathbf{p}})(\overline{\mathbf{I}} - cQ) \\ &= \overline{\mathbf{I}} - \overline{\mathbf{p}}(\overline{\mathbf{I}} - cQ) \end{aligned} \quad (15)$$

In addition, the ground truth $\overline{\mathbf{x}}$ can also be represented by $\overline{\mathbf{p}}$ and $\overline{\mathbf{r}}$. Left multiplying $(I - cQ)^{-1}$ on both sides of Eq. (15), we obtain

$$\overline{\mathbf{x}} = \overline{\mathbf{p}} + \overline{\mathbf{r}}(I - cQ)^{-1} \quad (16)$$

which indicates that a more accurate $\overline{\mathbf{p}}$ can be obtained by iteratively reducing the residual vector $\overline{\mathbf{r}}$.

Suppose $\vec{p}^{(t)}$ be the estimate at iteration t , and $\vec{r}^{(t)}$ be the current residual vector which is obtained by Eq. (15). The idea of local push for solving \vec{p} is: pick one coordinate i , transform r_i to p_i , and update \vec{r} using the updated \vec{p} . That is:

$$\vec{p}^{(t+1)} = \vec{p}^{(t)} + r_i \times \vec{e}_i \tag{17}$$

$$\vec{r}^{(t+1)} = \vec{r}^{(t)} - r^{(t)} \times \vec{e}_i + cr_i^{(t)} \vec{e}_i Q \tag{18}$$

where \vec{e}_i is an unit vector with 1 for the i^{th} dimension, and Eq. (18) is obtained by substituting Eq. (17) into Eq. (15). Equation (18) shows how $r^{(t+1)}$ is updated by $r^{(t)}$: first node i removes its residual $r^{(t)}$, and then $\forall j \in Out(j)$, node i pushes $\frac{c \times r_i^{(t)}}{|I(j)|}$ to node j .

Given a HIN $G = (V, E)$ and $T_G = (A, R)$, the LP based algorithm to compute all-pair GSR $\{S_j\}$ is shown in Algorithm 1.

Algorithm 1 Local Push Algorithm for all-pair GSR Computation

```

Require:  $G = (V, E), T_G = (A, R), c, r_{max}$ 
Ensure:  $\{P_i\}, 1 \leq i \leq |A|$ 
1: for  $(1 \leq i \leq |A|)$  do
2:    $P_i \leftarrow 0, R_i \leftarrow 1$ 
3: end for
4: while  $\max_{(u,v)}\{|R_i(u, v)|\} > r_{max}$  do
5:    $P_i(u, v) \leftarrow P_i(u, v) + R_i(u, v)$ 
6:    $tmp \leftarrow R_i(u, v), R_i(u, v) \leftarrow 0$ 
7:   for  $(x \in Out(u), y \in Out(v))$  do
8:     if  $x \neq y \wedge \phi(x) = \phi(y) = A_j$  then
9:        $R_j(x, y) \leftarrow R_j(x, y) + \frac{c \times tmp}{|I(x)||I(y)|}$ 
10:    end if
11:   end for
12: end while

```

Since each vertex of \mathcal{G}^2 is a pair of nodes in G , we use two matrix set $\{P_i\}$ and $\{R_i\}$ to represent the vector \vec{p} and \vec{r} respectively, i.e., $Uvec\{vec(P_i)\} = \vec{p}$ and $Uvec\{vec(R_i)\} = \vec{r}$.

4.2 Analysis of local push algorithm

4.2.1 Accuracy

Based on Eq. (16), we have $\|x - p\|_{max} = \|\vec{r}(I - cQ^{-1})\|_{\infty}$. Since the Q is a column normalized matrix, $\|Q\|_{\infty} = 1$. Thus, we have $\|x - p\|_{max} \leq \frac{\|\vec{r}\|_{\infty}}{(1-c)\|Q\|_{\infty}} = \frac{r_{max}}{1-c}$.

Given an error bound ϵ , we can set $r_{max} = (1 - c)\epsilon$ for Algorithm 1 so that $\|x - p\|_{max} \leq \epsilon$.

4.2.2 Time complexity

The proposed Algorithm 1 is different from local push for Personalized PageRank. For \vec{pr} of Eq. (13), it always have $\sum_i \vec{pr}_i = 1$. While given a node u , $\sum_{v \in V} s(u, v) \geq 1$. Thus, their convergence rates are different. We cannot reference the time complexity of local push based P-PageRank computation method directly.

Theorem 6 (Time Complexity of Local Push for GSR) *Let \bar{s} be the average GSR score on $G = (V, E)$. Then the time complexity of Algorithm 1 is $O(\frac{\bar{s}M}{(1-c)\epsilon})$ where M is the edge number of \mathcal{G}^2 .*

Proof Suppose $\forall n_l \in V^2, n_l = (u, v)$. When we run algorithm 1 for n_l , the time of operations $P_i(u, v) = P_i(u, v) + R_i(u, v)$ is at most $\lfloor \frac{s(u,v)}{(1-c)\epsilon} \rfloor$, since its increases from 0 to $s(u, v)$ by at least $r_{max} = (1 - c)\epsilon$. Furthermore, the time to propagate the new residual score is $O(|Out(n_l)|)$ since each out-neighbor of n_l must receive some of its score. Then, the running time for all nodes in \mathcal{G}^2 is at most:

$$\begin{aligned} \sum_{n_l \in V^2} \frac{s_l}{(1-c)\epsilon} O(|Out(n_l)|) &\approx \frac{\bar{s}}{(1-c)\epsilon} \sum_{n_l \in V^2} O(|Out(n_l)|) \\ &= O\left(\frac{\bar{s}M}{(1-c)\epsilon}\right) \end{aligned}$$

where M is the edge number of \mathcal{G}^2 . □

4.2.3 Space complexity

The original iterative method of GSR computation needs $O(N)$ space to maintain all the node pairs. The space cost of Algorithm 1 is dominated by the size of $\{P_i\}$ and $\{R_i\}$, whose size also are $O(N)$. In practice, we can construct 2 unified matrix P and R to represent the two matrix sets respectively. Further, we find that most elements in P and R are zeros when the desired error bound is achieved. The sparsity motivates us to use 2D hash tables to maintain P and R , which is usually used by current SimRank solutions [12]. Finally, the space cost of local push for GSR is $O(nnz(P) + nnz(R) + n + m) = O(nnz(R) + n + m)$ since there always exist residuals that have not been pushed in Algorithm 1, i.e. $nnz(R) \gg nnz(P)$.

4.3 Incremental algorithm for dynamic HIN

The local push algorithm can be extended to deal with dynamic HINs naturally. In this section, we show how to calculate GSR scores on dynamic HINs by extending local push.

The idea of incremental local push (ILP) algorithm is that: given $\{P_i\}$ and $\{R_i\}$ of the original HIN, only a few elements of $\{R_i\}$ exceed r_{max} due to the HIN change, including edge deletion and insertion. Hence, we can run the local push using the current $\{P_i\}$ and an updated $\{R'_i\}$. In the following, we first deduce an invariant for $\{P_i\}$ and $\{R_i\}$, then we show how to update $\{R_i\}$ to $\{R'_i\}$.

Theorem 7 *Given $\{P_i\}$ and $\{R_i\}$ in Algorithm 1, for $\forall u, v \in V \wedge \phi(u) = \phi(v) = A_i$, the following invariant holds at any step:*

$$P_i(u, v) + R_i(u, v) = \begin{cases} \frac{c \sum_{R_{j,i} \times y} P_j(x, y)}{|I(u)||I(v)|} & u \neq v \\ 1 & u = v \end{cases} \quad (19)$$

Proof Based on Eq. (15), we have: $\vec{r} + \vec{p} = \vec{I} + c\vec{p}Q$, so the proof can be completed by formulating each coordinate of \vec{p} and \vec{r} . □

When the original HIN is changed, we always have $P_i(u, v) = 1$ and $R_i(u, v) = 0$ if $u = v$ based on Eq. (19). Therefore, we only need to consider the the node pair $(u, v), u \neq v$ of R_i . Suppose an edge $e(a, b), \phi(a) = A_k, \phi(b) = A_i$ is inserted into G , the only entries of $\{R_i\}$ that are not satisfied with Eq. (19) would be $R_i(b, *)$ and $R_i(*, b)$ due to the change of in-degree. Let $\forall v \in V \wedge v \neq b \wedge \phi(v) = A_i$, and $R'_i(b, v)$ be the updated $R_i(b, v)$. Then the incremental amount Δ for updating $R_i(b, v)$ can be calculated as:

$$\begin{aligned}
\Delta &= R'_i(b, v) - R_i(b, v) \\
&= \left\{ \frac{c \sum \sum P_j(x, y) + c \sum_{a' \in k(v)} P_k(a, a')}{(|I(b)| + 1)|I(v)|} - P_i(b, v) \right\} \\
&\quad - \left\{ \frac{c \sum \sum P_j(x, y)}{|I(b)||I(v)|} - P_i(b, v) \right\} \\
&= \frac{c \sum_{a' \in k(v)} P_k(a, a')}{(|I(b)| + 1)|I(v)|} - \frac{c \sum \sum P_j(x, y)}{(|I(b)| + 1)|I(b)||I(v)|} \\
&= \frac{c \sum_{a' \in k(v)} P_k(a, a')}{(|I(b)| + 1)|I(v)|} - \frac{P_i(b, v) + R_i(b, v)}{|I(b)| + 1}
\end{aligned} \tag{20}$$

For the situation of edge deletion, the derivation is similar. The ILP algorithm for dynamic HIN is shown in Algorithm 2.

Algorithm 2 Incremental Local Push for Dynamic HIN

Require: $e(a, b) \in G, T_G, \{P_i\}, \{R_i\}, c, r_{max}$
Ensure: $\{P_i\}$

- 1: **if** $e(a, b)$ is inserted into G **then**
- 2: $G \leftarrow G \cup e(a, b)$
- 3: **for** $(v \in V \wedge \phi(v) = A_i \wedge v \neq b)$ **do**
- 4: calculate Δ based on Eq. (20)
- 5: $R_i(v, b) \leftarrow R_i(v, b) + \Delta$
- 6: $R_i(b, v) \leftarrow R_i(b, v) + \Delta$
- 7: **end for**
- 8: **else**
- 9: $G \leftarrow G / e(a, b)$
- 10: **for** $(v \in V \wedge \phi(v) = A_i \wedge v \neq b)$ **do**
- 11: $\Delta \leftarrow \frac{c \sum_{a' \in k(v)} P_k(a, a')}{|I(b)| - 1} - \frac{P_i(b, v) + R_i(b, v)}{|I(b)| - 1}$
- 12: $R_i(v, b) \leftarrow R_i(v, b) - \Delta$
- 13: $R_i(b, v) \leftarrow R_i(b, v) - \Delta$
- 14: **end for**
- 15: **end if**
- 16: **return** $LP(G, T_G, \{P_i\}, \{R_i\}, c, r_{max})$

Since ILP utilizes Algorithm 1 as return, it has the same accuracy guarantee as Algorithm. In addition, the space cost of ILP is the same as the static version.

Theorem 8 For each edge insertion/deletion in $G = (V, E)$, ILP has $O(m' + \frac{\bar{d}^2}{c(1-c)^2\varepsilon})$ where m' is the edge number of updated semantic relation and \bar{d} is the average degree of G .

Proof Given a homogeneous graph $G = (V, E)$, the time cost of Local Push to maintain a PPR vector to a target node t with an additive error ε as k edges are update is $O(k + \frac{\bar{d}}{\varepsilon})$.

Based on Theorem 5, we can reduce the ILP to the existing result of maintaining a PPR vector in dynamic graphs. The first difference is that one single edge update of a specific semantic relation in G would result in $2m' + 1$ edge updates in \mathcal{G}^2 . Second, computing the SimRank vector with an error bound ε is equivalent to computing a PageRank vector to t by local push, such that $\|S_i(u, v) - S_i(\hat{u}, v)\| < c(1 - c)\varepsilon$. Further, based on Algorithm 1, $r_{max} = (1 - c)\varepsilon$. Let \bar{d}^2 be the average degree of \mathcal{G} . Therefore, we infer that the time cost of one edge update in G is $O(m' + \frac{\bar{d}^2}{c(1-c)^2\varepsilon})$. \square

Table 1 Data sets

Data sets	Node	Edge	Node type
MovieLens-100K (ML) ^a	2625	100,000	2
IMDb ^b	8547	16,651	9
email-Enron (EN) ^c	36,692	183,831	1
DBLP ^d	164,341	963,258	5

These public data sets can be obtained from the following websites

^a<https://grouplens.org/datasets/movielens/>

^b<https://datasets.imdbws.com/>

^c<https://snap.stanford.edu/data/index.html>

^d<https://dblp.org/xml/>

Table 2 Accuracy results ($\epsilon = 10^{-3}$)

Dataset	LP		ILP		BiCG	
	Max error	Mean error	Max error	Mean error	Max error	Mean error
ML	5.62×10^{-4}	2.39×10^{-5}	7.26×10^{-4}	6.35×10^{-5}	3.26×10^{-5}	1.32×10^{-7}
IMDb	9.45×10^{-4}	5.13×10^{-6}	4.83×10^{-5}	3.78×10^{-5}	4.15×10^{-5}	2.81×10^{-7}
EN	5.01×10^{-4}	1.62×10^{-6}	8.12×10^{-4}	4.02×10^{-6}	1.62×10^{-5}	4.53×10^{-8}
DBLP	7.98×10^{-4}	3.22×10^{-6}	5.91×10^{-4}	1.18×10^{-6}	2.74×10^{-5}	1.29×10^{-8}

5 Experiment

5.1 Experimental settings

This section experimentally evaluates the proposed solutions against the power method. All experiments are conducted on a Linux machine with Xeon(R) CPU E5-2670 V2 @ 2.50GHZ and 128GB memory.

Baselines. We evaluate 4 algorithms: (1) the traditional iterative method (IM), (2) local push based algorithm 1 for static graphs (LP), (3) incremental local push algorithm for dynamic graphs (ILP), (4) Bi-Conjugate Gradient method (BiCG) for the linear system of Eq. (11) which can be easily rewritten in the form of $Ax = y$ through matrix transpose [11]. We set $c = 0.80$ for all algorithms.

Accuracy Metrics. We use the power method to calculate the ground-truth GSR score of each node pair. Besides the maximum error, we also use mean error to evaluate the accuracy for all-pair GSR computation, i.e., $ME = \frac{1}{N} \sum_i \sum_{u,v} |\hat{S}_i(u,v) - S_i(u,v)|$.

Data sets. we test the baselines on 4 graph data sets that are publicly available and commonly used in the literature. Table 1 shows the statistics of each graph. Besides 3 HINs, we also evaluate the algorithms on homogeneous information network, i.e. email-Enron.

5.2 Experimental results

5.2.1 Effectiveness

In our experimental evaluation, we assessed the accuracy of the BiCG, LP, and ILP algorithms across all datasets with a precision threshold of $\epsilon = 0.001$. The ground truth for GSR was computed using the Iterative Method (IM) with an accuracy of 10^{-4} . For testing the ILP, we randomly generated 2000 edge updates per dataset, which included an equal split of 50% edge insertions and 50% edge deletions. Each insertion involved adding a new edge, while each deletion involved removing an existing edge. The outcomes of these tests are presented in Table 2.

The results clearly indicate that the maximum errors for both LP and ILP are well within the specified error bounds, underscoring the effectiveness of these algorithms. Notably, BiCG exhibited the smallest mean and maximum errors, attributed to its continuous refinement of all elements in the similarity matrix until the maximum error falls below ϵ . Further insights from our experiments with LP and ILP include:

Fig. 2 Time cost on static HINs

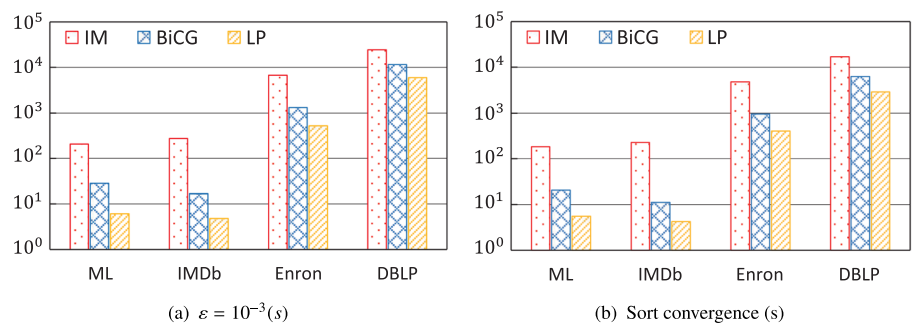
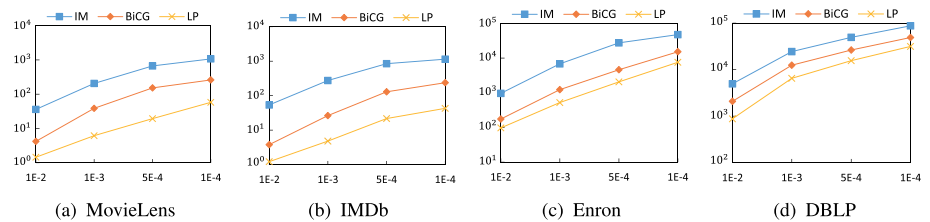


Fig. 3 CPU Time varying ϵ (s)



1. The error magnitudes of ILP on dynamic graphs are comparable to those of LP on static graphs. This similarity is due to Algorithm 2, which utilizes LP as its subroutine.
2. Both LP and ILP algorithms tend to perform better on larger datasets (e.g., DBLP) compared to smaller ones (e.g., MovieLens and IMDB). We hypothesize that nodes with higher in-degrees in larger datasets have increased chances of accumulating residuals from their neighbors, even when their individual errors are below ϵ . Our observations indicate that maximum errors predominantly occur in nodes with fewer in-neighbors, whereas nodes with higher in-degrees exhibit significantly lower errors. Generally, nodes in larger-scale networks have more neighbors, which likely contributes to the superior performance of LP in terms of mean error on these datasets.
3. The efficacy of our LP and ILP algorithms is also validated on homogeneous graphs, as demonstrated by the results on the email-Enron dataset.

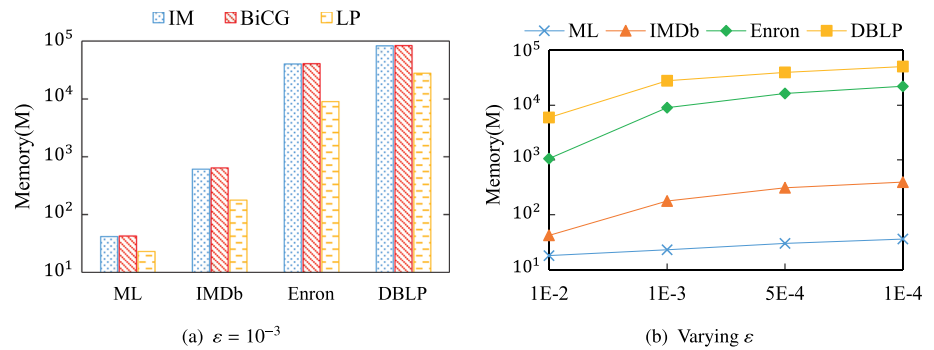
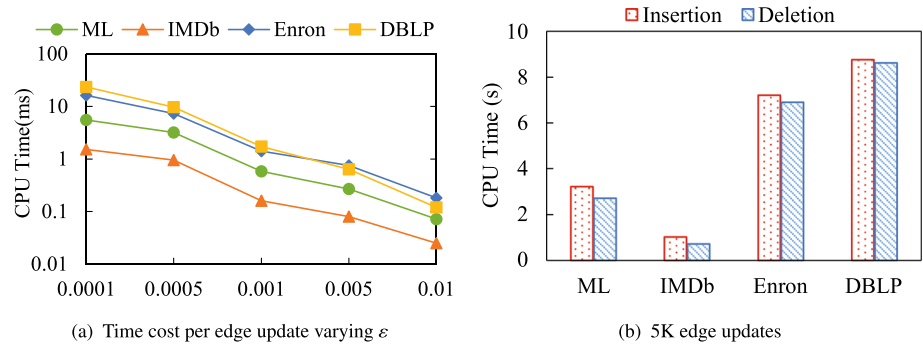
These findings highlight the robustness and adaptability of the LP and ILP algorithms across various types of graphs and datasets, proving their utility in both static and dynamic contexts.

5.2.2 Efficiency on static graphs

Time Efficiency. We conducted evaluations focusing on two main aspects: (1) the running time required for the IM, BiCG and LP algorithms to achieve the error bound ϵ and reach convergence across all datasets, and (2) the effectiveness of these algorithms with respect to varying error bounds ϵ on the same dataset. We set $\epsilon = 0.001$ for all datasets to compare the time efficiency of each algorithm in reaching this error threshold. The results are depicted in Fig. 2a. It is evident that LP significantly outpaces IM, improving by 1 – 2 orders of magnitude. BiCG, a state-of-the-art technique for solving linear systems, also outperforms IM and is surpassed by LP by approximately an order of magnitude. This performance differential is attributed to BiCG’s inclusion of entries with errors already below ϵ in each iteration, whereas LP focuses only on node pairs with residuals exceeding $(1 - c)\epsilon$, effectively minimizing unnecessary computations.

Additionally, we examined the time required to achieve sort convergence, which is crucial for applications such as similarity search. The findings, illustrated in Fig. 2b, show that LP consistently exceeds the performance of both IM and BiCG across all scenarios. Notably, all algorithms reach sort convergence faster than the set error threshold of $\epsilon = 0.001$. These observations affirm Theorem 6, suggesting that LP’s time cost is linearly related to dataset size.

The parameter ϵ plays a dual role, influencing both result quality and computational time. We assessed the impact of varying ϵ from 0.01 to 0.0001 on the running times of the three methods across all datasets, with the results presented in Fig. 3. LP consistently outperformed IM by more than an order of magnitude across all tested values of ϵ and was approximately 2 to 6 times faster than BiCG. Furthermore, as ϵ decreased, the time cost for all algorithms increased, indicating that higher accuracy demands more computational effort. Particularly, each tenfold increase in accuracy led

Fig. 4 Memory cost on static HINs**Fig. 5** CPU time of ILP on dynamic HINs

to a nearly proportional increase in LP's computation time, aligning with the time complexity analysis of LP discussed in Theorem 6.

Memory Cost. We next assess the space efficiency of the algorithms under consideration. With $\epsilon = 0.001$ set uniformly across all datasets, the space consumption results for the three algorithms are depicted in Fig. 4a. It is observed that the space requirement for LP is less than that for both the IM and the BiCG method. This efficiency in LP arises because it prunes unnecessary computational node pairs and utilizes 2D hash tables to manage sparse matrices effectively. In contrast, both IM and BiCG necessitate $O(N)$ space to store all node pairs in the GSR. However, the space cost for BiCG is greater than that for IM because BiCG requires additional vectors for search and residuals to determine subsequent search directions.

Further, we explore the impact of varying ϵ from 0.01 to 0.0001 on the memory cost by running LP on the same datasets to gauge its robustness concerning memory usage. These results are showcased in Fig. 4b. As ϵ decreases, there is a slight increase in space cost since LP involves more entries to maintain accuracy. Additionally, it is observed that larger-scale graphs, such as DBLP and Enron, are more sensitive to changes in ϵ compared to smaller graphs like MovieLens. This sensitivity is attributed to the lower average GSR scores in larger-scale graphs, where a significant proportion of entries exhibit small residuals, thus affecting the overall space efficiency.

5.2.3 Efficiency on dynamic graphs

We assess the time cost of the ILP algorithm on dynamic networks by examining two aspects: (1) the running time per edge update as ϵ varies, and (2) the scalability when the number of updated edges changes. Initially, we simulate 10,000 edge updates for each dataset, comprising an equal split of 50% edge insertions and 50% edge deletions. We then compute the average running time per edge update as ϵ is adjusted from 0.0001 to 0.01. The results are displayed in Fig. 5a.

It is readily apparent that the time cost per edge update is measured in microseconds. When compared to the results of LP shown in Fig. 2, it is evident that ILP operates at a significantly different level of time complexity, highlighting its high efficiency. Further observations reveal that as ϵ increases, the efficiency of ILP significantly improves. Specifically, a tenfold increase in ϵ typically enhances the efficiency of ILP by approximately an order of magnitude. This scaling behavior

supports our theoretical analysis of ILP's time complexity in Theorem 8, which is $O(\frac{1}{\epsilon})$. Performance comparisons across datasets show that ILP is more effective on IMDb than on MovieLens, while its efficiency on Enron and DBLP is comparable. This suggests that the efficiency of ILP is predominantly influenced by the average degree \bar{d} , c , ϵ , rather than by the number of updates m' . According to the data in Table 1, the average degree \bar{d} of MovieLens is nearly 20 times greater than that of IMDb, and the average degrees of Enron and DBLP are on similar levels. Notably, ILP performs better on DBLP than on Enron when $\epsilon > 0.001$. We attribute this to the fact that increasing ϵ leads to the removal of a large number of node pairs with small residuals, particularly in the large-scale DBLP network. This observation underscores the sensitivity of ILP's performance to network structure and the chosen error threshold.

Fixing the number of edge updates at 5,000, we compare the CPU time required for edge insertions and deletions across all datasets, with the results presented in Fig. 5b. It is readily apparent that ILP processes edge deletions more quickly than insertions. Notably, the time difference between edge insertion and deletion becomes more pronounced with smaller datasets. This is because edge insertions tend to increase the average degree \bar{d} of the network, whereas deletions decrease it. When the number of edge updates is fixed, the impact of changes in \bar{d} is more pronounced in smaller datasets.

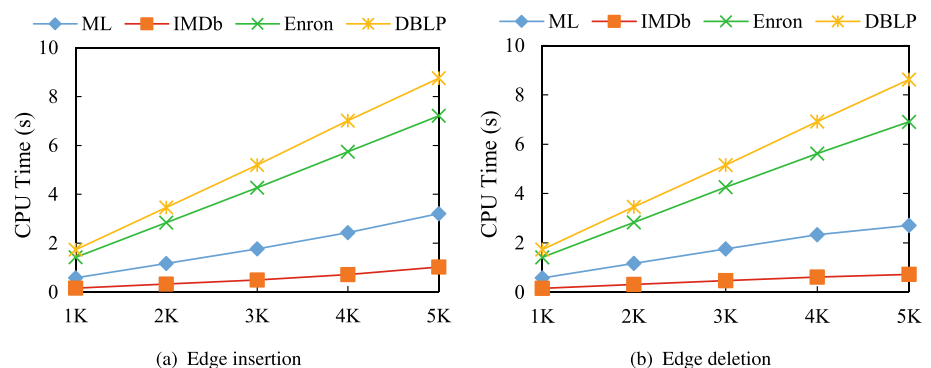
In our final set of experiments, we evaluate the scalability of ILP by varying the number of updated edges from 1000 to 5000 across all datasets. With ϵ fixed at 10^{-3} , the results are displayed in Fig. 6, which includes separate charts for insertions (a) and deletions (b). Generally, the time cost of ILP exhibits a linear relationship with the number of updated edges. Furthermore, it is observed that as the number of updated edges increases, the time required for edge insertions exceeds that for deletions. This consistent pattern highlights the relative computational demands of these operations within the ILP framework.

6 Related work

Similarity Measure. Measuring the similarity of objects in information networks is a fundamental problem that has attracted significant attention due to its broad applications [1]. Research in this area generally falls into one of two categories: (1) Content-based similarity measures, which treat each object as a bag of attributes or as a vector of feature weights, exemplified by approaches like Similarity Join [24]. (2) Link-based similarity measures, which focus on the connections between objects, such as personalized PageRank [6], Random Walk with Restart (RWR) [5], and SimRank [7, 25]. According to evaluations in Sun et al. [8], link-based similarity measures tend to correlate better with human judgments compared to content-based measures.

Most existing similarity measures that utilize link information are defined on homogeneous networks and cannot be directly applied to Heterogeneous Information Networks (HINs) due to the varied semantics of the edges. To account for the diverse semantics in HINs, PCRW introduces a learnable proximity measure based on random walk with restart, defined by a weighted combination of simple "path experts," each following a specific sequence of labeled edges. Sun Y. et al. define meta paths on network schemas and propose PathSim to measure the similarity of same-typed objects through symmetric path instances [8]. Building on the concept of meta paths, several variants have been proposed, such as HeteSim [26] and HeteRank [27]. However, these approaches primarily provide local similarities. Some studies attempt to integrate these local similarities into a unified global similarity measure using meta paths or other structures, such as meta trees [28] and

Fig. 6 The scalability of ILP on dynamic HINs



meta graphs [3]. Nevertheless, these methods often struggle to ensure semantic consistency, which requires that each step of the similarity calculation be logically interpretable so that the final results align with semantic logic.

General SimRank [9], an extension of the well-known SimRank, offers a comprehensive solution for similarity computation in HINs. GSR calculates the similarity between objects of the same type based on the similarities of their in-neighbors, providing globally consistent and semantically aware similarity scores without the need for learning.

In contemporary research, deep learning has become the mainstream technology for studying HINs, encompassing tasks from node embedding to link prediction, recommendation, and clustering. Node embeddings can also measure similarity. However, there are drawbacks. Firstly, graph embedding inevitably leads to information loss. Secondly, graph embeddings are often not an end in themselves but are used as inputs for subsequent deep learning models designed to address practical problems. Despite the successes of deep learning, challenges such as interpretability and learning costs remain significant [4]. Structure-based learning methods may offer a promising route to interpretable machine learning, combining the infinite fitting capabilities of deep neural networks with the interpretability of knowledge graphs. From this perspective, link-based similarity measures are crucial to structure learning models, akin to the roles of conditional probability and vector inner products in traditional learning models [2].

Fast Computation. To the best of our knowledge, there is currently no research specifically focused on the fast computation of General SimRank (GSR). However, the computation of SimRank, which can be considered a specific form of GSR, has been extensively studied over many years with numerous approaches proposed. Insights from these studies may be applicable to GSR computation. Generally, methods for calculating SimRank can be categorized into three groups:

- **Power Method:** This approach is based on a naive iterative method that utilizes the adjacency matrix to compute the similarity matrix. Due to the sparsity of the adjacency matrix, power methods often employ matrix compression techniques to reduce space complexity and utilize compact matrix computing technologies to enhance computational efficiency [20]. Additionally, to reduce the time cost, some studies cache frequently used partial results [14]. Although the power method is highly accurate, its space and time costs exceed linear complexity, making it challenging to apply to large-scale or dynamic graphs.
- **Random Walk Method:** This method posits that the SimRank score $s(u, v)$, is equivalent to the expected first meeting time $E(c^t)$ of two random walks starting from nodes u and v , respectively [15]. Random walk-based methods typically involve two phases: generating indices for random walks and calculating similarity based on the meeting times and steps for queries. Efficiency improvements have been made through index structure optimization and modifications to the Monte Carlo sampling strategy [16–18]. While the random walk method is often used for single-source queries with linear time complexity [10], its performance can be variable in terms of stability and accuracy due to the randomness of sampling.
- This is a prominent research direction for SimRank computation. Some approaches approximate the SimRank formula as a linear system, which is then solved using linear algebra techniques such as eigen-decomposition or Singular Value Decomposition (SVD) [19–22]. Due to the use of the Kronecker product, the space complexity becomes $O(r^2n^2)$, where r is the rank of the adjacency matrix. Methods like PBiCG [11] and FLP-SR [12] reformulate the SimRank equation on G^2 and derive equivalent linear systems by eliminating the nonlinear operator v , allowing for the calculation of all-pair SimRank scores with high accuracy.

In summary, while existing methods provide various ways to compute SimRank, each has its own set of trade-offs concerning accuracy, efficiency, and complexity. These insights could guide the development of efficient algorithms for computing GSR in both static and dynamic contexts.

7 Conclusion

This paper tackles the challenge of computing exact all-pair General SimRank (GSR) scores in both static and dynamic HINs. Traditional optimization techniques for SimRank are not directly applicable to GSR due to its distinct matrix formulation. To address this, we have developed a novel linear system specifically for GSR and introduced a local push-based algorithm to compute all-pair GSR scores efficiently. A key feature of our algorithm is that it does not require the computation of all GSR scores, employing a push-on-demand approach that significantly reduces unnecessary

computational costs while maintaining guaranteed accuracy. Additionally, we have adapted our algorithm to monitor GSR scores in dynamic HINs effectively. Our experimental results demonstrate that our algorithms significantly outperform existing power iteration solutions in terms of both time and memory efficiency.

Acknowledgements We sincerely thank the editors and reviewers for valuable suggestions and constructive comments. This work was supported by the Shandong Taishan Industry Leading Talent Project NO.tscx202211010; National Natural Science Foundation of China under Grant 2018IM020200 and MSTIP of Shandong Province of China Grant 2019JZZY010109.

Author contributions Prof. Xiaoguang Hong and Yongqing Zheng identified the problem and proposed the basic solution idea. Dr. Chuanyang Zhang designed the experiments and wrote the main text of the manuscript. All authors had reviewed the manuscript.

Funding This paper is fully supported by the following funds: Shandong Taishan Industry Leading Talent Project NO.tscx202211010; National Natural Science Foundation of China under Grant 2018IM020200; MSTIP of Shandong Province of China Grant 2019JZZY010109.

Data availability All data used in this paper are public, and the sources are listed in Table 1. All models and code generated during the study are available from the corresponding author (Chuanyang Zhang) by request.

Declarations

Ethics approval and consent to participate Not applicable

Competing interests We declare that the authors have no Conflict of interest as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A

Proofs

Proof of Theorem 1 Generally, for any node pair (u, v) , $\phi(u) = \phi(v) = A_i$, we have

$$s^0(u, v) = \begin{cases} 1 & u = v \\ 0 & u \neq v \end{cases} \quad (21)$$

where $s^j(u, v)$ is the similarity result after j^{th} iteration based on the naive iterative method.

If $u = v$, based on Eq. (2) of GSR, for $k = 0, 1, 2, \dots$, we can obtain: $s^k(u, v) \equiv 1$.

If $u \neq v$, we can obtain:

$$\begin{aligned} s^{k+1}(u, v) &= \frac{c}{|I(u)||I(v)|} \sum_{R_{ji} \in R} \sum_{x \in I_j(u)} \sum_{y \in I_j(v)} s^k(x, y) \\ &= c \sum_{R_{ji} \in R} \sum_{x \in I_j(u)} \sum_{y \in I_j(v)} \frac{1}{|I(u)|} s^k(x, y) \frac{1}{|I(v)|} \\ &= c \sum_{R_{ji} \in R} \sum_{x \in I_j(u)} \sum_{y \in I_j(v)} W_{ji}(x, u) s^k(x, y) W_{ji}(y, v) \\ &= c \sum_{R_{ji} \in R} W_{ji}^T(u, :) S_j^k W_{ji}(:, v) \end{aligned} \quad (22)$$

Finally, in matrix notation, the iterative computation of General SimRank can be formulated as:

$$\begin{cases} S_i^0 = I \\ S_i^{k+1} = (c \sum_{R_{ji} \in R} W_{ji}^T S_j^k W_{ji}) \vee I \end{cases} \quad (23)$$

Theorem 3 of the work [9] shows that there exists a unique solution of GSR. Thus, we can infer that $S_i = (c \sum_{R_{ji} \in R} W_{ji}^T S_j W_{ji}) \vee I$ is the equivalent form of Eq. (2). □

References

- Shi C, Li Y, Zhang J, Sun Y, Yu PS. A survey of heterogeneous information network analysis. *IEEE Trans Knowl Data Eng.* 2017;29(1):17–37. <https://doi.org/10.1109/TKDE.2016.2598561>.
- Zhang C, Hong X. Challenging the long tail recommendation on heterogeneous information network. In: 2021 International Conference on Data Mining, ICDM 2021—Workshops, Auckland, New Zealand, December 7–10, 2021. p. 94–101. <https://doi.org/10.1109/ICDMW53433.2021.00018>.
- Fang Y, Lin W, Zheng VW, Wu M, Shi J, Chang KC, Li X. Metagraph-based learning on heterogeneous graphs. *IEEE Trans Knowl Data Eng.* 2021;33(1):154–68. <https://doi.org/10.1109/TKDE.2019.2922956>.
- Zhang Z, Cui P, Zhu W. Deep learning on graphs: a survey. *IEEE Trans Knowl Data Eng.* 2022;34(1):249–70. <https://doi.org/10.1109/TKDE.2020.2981333>.
- Yoon M, Jung J, Kang U. TPA: fast, scalable, and accurate method for approximate random walk with restart on billion scale graphs. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16–19, 2018. p. 1132–43. <https://doi.org/10.1109/ICDE.2018.00105>.
- Zhang H, Lofgren P, Goel A. Approximate personalized pagerank on dynamic graphs. In: Krishnapuram B, Shah M, Smola AJ, Aggarwal CC, Shen D, Rastogi R, editors. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13–17, 2016. p. 1315–24. <https://doi.org/10.1145/2939672.2939804>.
- Jeh G, Widom J. SimRank: a measure of structural-context similarity, 2002. p. 538–43. <https://doi.org/10.1145/775047.775126>
- Sun Y, Han J, Yan X, Yu PS, Wu T. Pathsim: meta path-based top-k similarity search in heterogeneous information networks. *Proc VLDB Endow.* 2011;4(11):992–1003.
- Zhang C, Hong X, Peng Z. GSimRank: A general similarity measure on heterogeneous information network. In: Wang X, Zhang R, Lee Y, Sun L, Moon Y, editors. Web and Big Data—4th International Joint Conference, APWeb-WAIM 2020, Tianjin, China, September 18–20, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12317, 2020. p. 588–602. https://doi.org/10.1007/978-3-030-60259-8_43.
- Wang H, Wei Z, Liu Y, Yuan Y, Du X, Wen J. ExactSim: benchmarking single-source SimRank algorithms with high-precision ground truths. *VLDB J.* 2021;30(6):989–1015. <https://doi.org/10.1007/s00778-021-00672-7>.
- Lu J, Gong Z, Lin X. A novel and fast SimRank algorithm. *IEEE Trans Knowl Data Eng.* 2017;29(3):572–85. <https://doi.org/10.1109/TKDE.2016.2626282>.
- Wang Y, Lian X, Chen L. Efficient SimRank tracking in dynamic graphs. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16–19, 2018. p. 545–56. <https://doi.org/10.1109/ICDE.2018.00056>.
- Lizorkin D, Velikhov P, Grinev MN, Turdakov D. Accuracy estimate and optimization techniques for SimRank computation. *VLDB J.* 2010;19(1):45–66. <https://doi.org/10.1007/s00778-009-0168-8>.
- Yu W, Lin X, Zhang W. Towards efficient SimRank computation on large networks. In: Jensen CS, Jermaine CM, Zhou X, editors. 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8–12, 2013. p. 601–12. <https://doi.org/10.1109/ICDE.2013.6544859>.
- Tian B, Xiao X. SLING: A near-optimal index structure for SimRank. In: Özcan F, Koutrika G, Madden S, editors. Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26–July 01, 2016. p. 1859–74. <https://doi.org/10.1145/2882903.2915243>.
- Jiang M, Fu AW, Wong RC, Wang K. READS: a random walk approach for efficient and accurate dynamic SimRank. *Proc VLDB Endow.* 2017;10(9):937–48. <https://doi.org/10.14778/3099622.3099625>.
- Liu Y, Zheng B, He X, Wei Z, Xiao X, Zheng K, Lu J. Probesim: scalable single-source and top-k SimRank computations on dynamic graphs. *Proc VLDB Endow.* 2017;11(1):14–26. <https://doi.org/10.14778/3151113.3151115>.
- Song J, Luo X, Gao J, Zhou C, Wei H, Yu JX. Uniwalk: unidirectional random walk based scalable SimRank computation over large graph. *IEEE Trans Knowl Data Eng.* 2018;30(5):992–1006. <https://doi.org/10.1109/TKDE.2017.2779126>.
- Li C, Han J, He G, Jin X, Sun Y, Yu Y, Wu T. Fast computation of SimRank for static and dynamic information networks. In: Manolescu I, Spaccapetra S, Teubner J, Kitsuregawa M, Léger A, Naumann F, Ailamaki A, Özcan F, editors. EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22–26, Proceedings. ACM International Conference Proceeding Series, vol. 426. 2010. p. 465–76. <https://doi.org/10.1145/1739041.1739098>.
- Yu W, Zhang W, Lin X, Zhang Q, Le J. A space and time efficient algorithm for SimRank computation. *World Wide Web.* 2012;15(3):327–53. <https://doi.org/10.1007/s11280-010-0100-6>.

21. Fujiwara Y, Nakatsuji M, Shiokawa H, Onizuka M. Efficient search algorithm for SimRank. In: Jensen CS, Jermaine CM, Zhou X, editors. 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8–12, 2013. p. 589–600. <https://doi.org/10.1109/ICDE.2013.6544858>.
22. Maehara T, Kusumoto M, Kawarabayashi K. Scalable SimRank join algorithm. In: Gehrke J, Lehner W, Shim K, Cha SK, Lohman GM, editors. 31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13–17, 2015. p. 603–14. <https://doi.org/10.1109/ICDE.2015.7113318>.
23. Bressan M, Pretto L. Local computation of pagerank: the ranking side. In: Macdonald C, Ounis I, Ruthven I, editors. Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24–28, 2011. p. 631–40. <https://doi.org/10.1145/2063576.2063670>.
24. Aumüller M, Ceccarello M. Implementing distributed similarity joins using locality sensitive hashing. In: Stoyanovich J, Teubner J, Guagliardo P, Nikolic M, Pieris A, Mühlrig, J, Özcan F, Schelter S, Jagadish HV, Zhang M, editors. Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29–April 1, 2022. p. 1–78190. <https://doi.org/10.5441/002/edbt.2022.07>.
25. Antonellis I, Garcia-Molina H, Chang C. SimRank++: query rewriting through link analysis of the click graph. Proc VLDB Endow. 2008;1(1):408–21. <https://doi.org/10.14778/1453856.1453903>.
26. Shi C, Kong X, Huang Y, Yu PS, Wu B. Hetesim: a general framework for relevance measure in heterogeneous networks. IEEE Trans Knowl Data Eng. 2014;26(10):2479–92. <https://doi.org/10.1109/TKDE.2013.2297920>.
27. Zhang M, Wang J, Wang W. HeteRank: a general similarity measure in heterogeneous information networks by integrating multi-type relationships. Inf Sci. 2018;453:389–407. <https://doi.org/10.1016/j.ins.2018.04.022>.
28. Zhou Y, Huang J, Sun H, Sun Y, Qiao S, Wambura SM. Recurrent meta-structure for robust similarity measure in heterogeneous information networks. ACM Trans Knowl Discov Data. 2019;13(6):64–16433. <https://doi.org/10.1145/3364226>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.