



RoleSim*: Scaling axiomatic role-based similarity ranking on large graphs

Weiren Yu^{1,2} · Sima Iranmanesh² · Aparajita Haldar² · Maoyin Zhang¹ · Hakan Ferhatosmanoglu²

Received: 4 February 2021 / Revised: 28 May 2021 / Accepted: 7 July 2021 /
Published online: 11 August 2021

© The Author(s) 2021

Abstract

RoleSim and SimRank are among the popular graph-theoretic similarity measures with many applications in, *e.g.*, web search, collaborative filtering, and sociometry. While RoleSim addresses the automorphic (role) equivalence of pairwise similarity which SimRank lacks, it ignores the neighboring similarity information out of the automorphically equivalent set. Consequently, two pairs of nodes, which are not automorphically equivalent by nature, cannot be well distinguished by RoleSim if the averages of their neighboring similarities over the automorphically equivalent set are the same. To alleviate this problem: 1) We propose a novel similarity model, namely RoleSim*, which accurately evaluates pairwise role similarities in a more comprehensive manner. RoleSim* not only guarantees the automorphic equivalence that SimRank lacks, but also takes into account the neighboring similarity information outside the automorphically equivalent sets that are overlooked by RoleSim. 2) We prove the existence and uniqueness of the RoleSim* solution, and show its three axiomatic properties (*i.e.*, symmetry, boundedness, and non-increasing monotonicity). 3) We provide a concise bound for iteratively computing RoleSim* formula, and estimate the number of iterations required to attain a desired accuracy. 4) We induce a distance metric based on RoleSim* similarity, and show that the RoleSim* metric fulfills the triangular inequality, which implies the sum-transitivity of its similarity scores. 5) We present a threshold-based RoleSim* model that reduces the computational time further with provable accuracy guarantee. 6) We propose a single-source RoleSim* model, which scales well for sizable graphs. 7) We also devise methods to scale RoleSim* based search by incorporating its triangular inequality property with partitioning techniques. Our experimental results on real datasets demonstrate that RoleSim* achieves higher accuracy than its competitors while scaling well on sizable graphs with billions of edges.

Keywords Role-based similarity · Retrieval models and ranking · Web search · Link analysis

This article belongs to the Topical Collection: *Special Issue on Large Scale Graph Data Analytics*
Guest Editors: Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang

✉ Weiren Yu
weiren.yu@warwick.ac.uk

Extended author information available on the last page of the article.

1 Introduction

RoleSim, conceived by Jin et al. [9], is a promising role-oriented graph-theoretic measure that quantifies the similarity between two objects based on graph automorphism, with a proliferation of real-life applications [9, 10, 25], such as link prediction (social network), co-citation analysis (bibliometrics), motif discovery (bioinformatics), and collaborative filtering (information retrieval). It recursively follows a SimRank-like reasoning that “two nodes are assessed as role similar if they interact with automorphically equivalent sets of in-neighbors”. Intuitively, automorphically equivalent nodes in a graph are objects having similar roles that can be exchanged with minimum effect on the graph structure. Similar to the well-known SimRank measure [7], the recursive nature of RoleSim allows to capture the multi-hop neighboring structures that are automorphically equivalent in a network. Unlike SimRank that measures the similarity of two nodes from the paths connecting them, RoleSim quantifies similarities through the paths connecting their different roles. As a result, two nodes that are disconnected from each other will not be considered as dissimilar by RoleSim if they have similar roles. For evaluating similarity score $s(a, b)$ between nodes a and b , as opposed to SimRank whose similarity $s(a, b)$ takes the average similarity of all the neighboring pairs of (a, b) , RoleSim computes $s(a, b)$ by averaging only the similarities over the maximum bipartite matching of all the neighboring pairs of (a, b) . This subtle difference enables RoleSim to guarantee the automorphic equivalence, which SimRank lacks, in final scoring results. Therefore, RoleSim has been demonstrated as an effective similarity measure in a wide range of real applications. We summarize two of these applications below.

Application 1 (Similarity Search on the Web) Discovering web pages similar to a query page is an important task in information retrieval. In a Web graph, each node represents a web page, and an edge denotes a hyperlink from one page to another. RoleSim can be applied to measure the similarity of two web pages, based on the intuition that “two web pages are role-similar if they are pointed to by the automorphically equivalent sets of their in-neighboring pages”. This similarity measure produces more reliable similarity results than the SimRank model [10].

Application 2 (Social Network De-anonymization) Social network de-anonymization is a method to validate the strength of anonymization algorithms that protect a user’s privacy. RoleSim has been applied to de-anonymise node mappings based on the similarity information between a crawled network and an anonymised one. Based on the observation that “correct mappings tend to have higher similarity scores”, RoleSim iteratively evaluates pairwise node similarities between two networks, and captures the reasoning that “a pair of nodes between two networks is more likely to be a correct mapping if their neighbors are correct mappings”. RoleSim has demonstrated superior performance as compared with other existing de-anonymization algorithms [25].

Despite its popularity in real-world applications, RoleSim has a major limitation: with the aim to achieve automorphic equivalence, its similarity score $s(a, b)$ only considers the limited information of the average similarity scores over the automorphically equivalent set (*i.e.*, the maximum bipartite matching) of a ’s and b ’s in-neighboring pairs, but neglects the rest of the pairwise in-neighboring similarity information that is outside the automorphically equivalent set. Consequently, RoleSim does not always produce comprehensive similarity results because two pairs of nodes, which are not automorphically equivalent by nature, should be distinguishable from each other even though the average values of their

in-neighboring similarities over the set of the maximum bipartite matching are the same, as illustrated in Example 1.

Example 1 (Limitation of RoleSim) Consider the web graph G in Figure 1, where each node denotes a web page, and each edge depicts a hyperlink from one page to another. Using RoleSim, we evaluate pairs of similarities between nodes, as partially illustrated in the ‘RS’ column of the right table. It is discerned that node-pairs (1, 2) and (1, 3) have the same RoleSim similarity values, which is not reasonable. Because node 2 and node 3 are not strictly automorphically equivalent by nature, their similarities with respect to the same query node 1, *i.e.*, $s(1, 2)$ and $s(1, 3)$, should not be the same.

We notice that the main reason why $s(1, 2)$ and $s(1, 3)$ are assessed to be the same by the RoleSim model is that its similarity $s(a, b)$ considers only the average similarity scores over the maximum bipartite matching, denoted as $M_{a,b}$, of (a, b) ’s in-neighboring pairs $I_a \times I_b$, where I_a denotes the in-neighbor set of node a , and \times is the Cartesian product of two sets. Thus, the similarity information in the remaining in-neighboring pairs of (a, b) , *i.e.*, $I_a \times I_b - M_{a,b}$, are totally ignored. For example, if unfolding the in-neighboring pairs of (1, 2) and (1, 3) respectively, we see that, in the gray cells, $M_{1,2} = \{(4, 6), (5, 7)\}$ (*resp.* $M_{1,3} = \{(4, 9), (5, 10)\}$) is the maximum bipartite matching of (1, 2)’s (*resp.* (1, 3)’s) in-neighboring pairs $I_1 \times I_2$ (*resp.* $I_1 \times I_3$). The sum of the similarity values over $M_{1,2}$ is $0.488 + 0.360 = 0.848$, which is the same as that over $M_{1,3}$. Thus, RoleSim cannot distinguish $s(1, 2)$ from $s(1, 3)$.

Example 1 illustrates that, to effectively evaluate $s(a, b)$, relying only on the in-neighboring-pairs similarities in the maximum bipartite matching $M_{a,b}$ (*e.g.*, RoleSim) is not enough. Although RoleSim has the advantage of using intuitively the most influential pairs $M_{a,b}$ among all the in-neighboring pairs $I_a \times I_b$ for achieving automorphic equivalence, it *completely* ignores the similarity information outside $M_{a,b}$. For instance in Example 1, there are opportunities to take good advantage of the similarity values in the regions $I_1 \times I_2 - M_{1,2}$ and $I_1 \times I_3 - M_{1,3}$ which would be helpful to distinguish $s(1, 2)$ from $s(1, 3)$ further when the average similarities over $M_{1,2}$ and $M_{1,3}$ are the same.

Contributions Motivated by this, our main contributions are as follows:

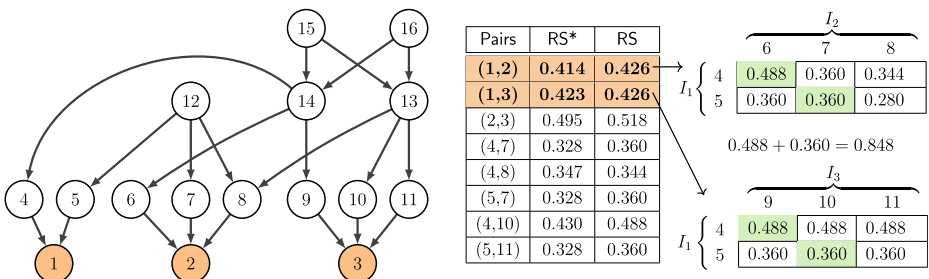


Figure 1 Limitation of RoleSim (RS) on a web graph, where node-pairs (1, 2) and (1, 3) have the same RoleSim score (0.426) since RS aggregates only the in-neighboring pairs that are automorphically equivalent (colored in green) whose sums are the same ($0.488 + 0.360 = 0.848$), while ignoring the remaining pairs

- 1) We first propose a novel similarity model, RoleSim*, which accurately evaluates pairwise role similarities in a more comprehensive fashion. Compared with the existing well-known similarity models (*e.g.*, SimRank and RoleSim), RoleSim* not only guarantees the automorphic equivalence that SimRank lacks, but also takes into consideration the pairwise similarities outside the automorphically equivalent sets that are overlooked by RoleSim. (Section 3.1)
- 2) We show three key properties of RoleSim*, *i.e.*, symmetry, boundedness, and non-increasing monotonicity of its iterative similarity scores. On the top of that, we prove the existence and uniqueness of the RoleSim* solution. (Section 3.2)
- 3) We derive an iterative formula for computing RoleSim* similarities. A concise upper bound for RoleSim* iterations is also established, which can estimate the total number of iterations required for attaining a desired accuracy. (Section 3.3)
- 4) To substantially accelerate the computation of RoleSim*, we also devise a threshold-based RoleSim* model based on two pruning strategies, and provide provable guarantees on accuracy which is controlled by a user-specified threshold parameter δ trading between speed and accuracy. (Section 4)
- 5) To scale RoleSim* similarity search well on large graphs with billions of edges, we propose a scalable algorithm for single-source RoleSim* retrieval, which avoids spending unnecessary time on repeated RoleSim* computations while caching important pairs through an unordered hash map. (Section 5)
- 6) We induce a distance metric based on our RoleSim* measure, and rigorously show that the RoleSim* distance metric fulfills the triangular inequality which other measures (*e.g.*, cosine distance) lack. This implies the sum-transitivity of the RoleSim* measure. (Section 6)
- 7) We discuss approaches to scale RoleSim* based search using the triangle inequality property and partitioning techniques (Section 7).
- 8) We conduct an experimental study to validate the effectiveness of our RoleSim* model. Our empirical results show that RoleSim* achieves higher accuracy than the existing competitors (*e.g.*, RoleSim and SimRank) while entailing comparable computational complexity bounds of RoleSim. We also devise an unsupervised experimental setting that quantifies the effectiveness of similarity measures, where RoleSim* outperforms the alternatives. (Section 8)

2 Related work

Graph-based similarity models have been popular since SimRank measure was proposed by Jeh and Widom [7]. SimRank is a node-pair similarity measure, which follows the recursive idea that “two nodes are considered as similar if they are pointed to by similar nodes”. Since then, there have been surges of studies focusing on optimization problems to accelerate SimRank computation as the naive SimRank computing method entails quadratic time in the number of nodes. According to assumptions on data updates, recent results can be divided into static algorithms [1, 4, 5, 12, 16, 22, 26, 32, 34, 37, 42], and dynamic algorithms on evolving graphs [8, 13, 19, 24, 28, 36, 40]. According to types of queries, these results are classified into single-source SimRank [8, 12, 19, 26, 40], single-pair SimRank [6, 15], all-pairs SimRank [1, 20, 34, 35], and partial-pairs SimRank [22, 39].

Recent years have witnessed an upsurge of interest in the semantic problems of pairwise similarity measures. Various SimRank and SimRank-like models have come into play.

Representative examples include C-Rank [31], SimFusion [38], Penetrating-Rank [41], RoleSim++ [25], RoleSim [9], MatchSim [18], SimRank* [37], ASCOS [3], CoSim-Rank [23], and SemSim [32]. In what follows, we will elaborate the pros and cons of these similarity measures and discuss their relations to this work.

C-Rank [31] C-Rank is a contribution-based ranking algorithm that integrates both content and link information of web pages through the concept of contribution, indicating that a page may contribute to enhancing the content quality of adjacent pages pointing to it via linkages. A C-Rank score of each page on a term is defined to be a linear combination of (i) its relevance score to the term and (ii) its contribution score that quantifies the degree of its overall contributions to other pages on the term. However, unlike similarity scores from the RoleSim family, C-Rank does not take into account the automorphic equivalence property for each pair of nodes. Our experimental evaluation demonstrates the accuracy of RoleSim* is superior to C-Rank with a little compromise in the computational time.

Penetrating-Rank [41] Zhao et al. [41] proposed Penetrating-Rank, which is a SimRank-based similarity measure that comprehensively considers both incoming and outgoing neighbouring information for similarity assessment. However, Penetrating-Rank is not an automorphic equivalence-based measure as role discovery is not the primary task of this model. Recently, the idea of Penetrating-Rank applied to SimRank shows some degree of resemblance to the idea of RoleSim++, which is a generalisation of RoleSim through exploitation of both in- and out-links of the graph structure.

RoleSim [10] RoleSim has been accepted as a promising role-based similarity model, due to its elegant intuition that “if two nodes are automorphically equivalent, they should share the same role and their role similarity should be maximal”. To speed up the RoleSim computation, an approximate heuristic, named Iceberg RoleSim, was devised to prune small similarity values below a threshold. Unlike SimRank that takes the average similarity of all the neighboring pairs of (a, b) , RoleSim computes $s(a, b)$ by averaging only the similarities over the maximum bipartite matching $M_{a,b}$. However, all the similarity information not included in the matching $M_{a,b}$ is completely ignored by RoleSim. In contrast, our RoleSim* model can effectively capture these information while guaranteeing automorphic equivalence.

RoleSim++ [25] RoleSim++, proposed by Shao et al. [25], takes good advantage of the direction information of both in- and out-links to model pairwise similarities, which is successfully used in the real-world de-anonymization application. It employs a novel matching algorithm, NeighborMatch, to find matchings for inner and outer neighbors, respectively. Moreover, a threshold-based model, α -RoleSim++, is proposed to eliminate tiny scores for speedup further. Our techniques of RoleSim* can also be slightly modified to tailor RoleSim++ to accommodate similarity contributions from non-automorphically equivalent pairs of in- and out-neighbours for semantic enhancement.

SimFusion [30] SimFusion exploits a unified relationship matrix (URM) to capture the inter- and intra-relationships among a set of heterogeneous data objects. A unified similarity matrix (USM), which is evaluated iteratively from the URM, characterises the latent relationships among heterogeneous data objects. However, as opposed to RoleSim*, SimFusion fails to capture automorphically equivalent relationships among the heterogeneous data objects.

MatchSim [18] Lin et al. [18] introduced MatchSim similarity model, which computes the similarity values between two objects based on the average similarity of their maximum matched neighbours. The key difference between MatchSim and RoleSim lies in the initialisation step – MatchSim starts with an identity matrix as its initial similarity and defines $s_0(a, b) = 1$ if $a = b$, and 0 otherwise, whereas RoleSim utilises a matrix of all ones to be starting similarity matrix which initialises all $s_0(*, *) = 1$. As a result, RoleSim exhibits the automorphism property that MatchSim lacks. However, similar to RoleSim, MatchSim totally neglects the neighboring similarity values outside the automorphically equivalent sets. The idea of RoleSim* can be applied to MatchSim in a similar way to resolve this problem.

SimRank* [37] & **ASCOS** [3] SimRank* and ASCOS are two variations of the SimRank model that addresses the zero-similarity problem of the SimRank measure. Nevertheless, these methods do not take into account the automorphically equivalent structure of nodes. The key idea of RoleSim* can be applied in a similar manner to SimRank* and ASCOS models to enrich the meaningful semantics of similarity assessment while effectively circumventing the zero SimRank problem.

CentSim [14] Li et al. [14] proposed CentSim, a centrality-based role similarity measure, which compares the centrality values of two nodes to evaluate their similarity. This measure employs several types of centrality including PageRank, Degree and Closeness for each node, and considers the weighted average of them for evaluating CentSim scores.

SemSim [32] Milo et al. [32] proposed a semantic-aware random walk-based model namely, SemSim, which is an extension of SimRank applied to heterogeneous information networks. SemSim aims to boost the quality of SimRank similarity scores by exploiting its node semantics and edge weights. Nonetheless, SemSim inherits the limitation of SimRank whose similarity values ignore the role-equivalent relationship between nodes.

Co-SimRank [23] Rothe and Schütze [23] presented Co-SimRank, a SimRank-like measure of pairwise similarity based on graph structure. A Co-SimRank score $s(a, b)$ of each pair (a, b) is computed from the inner product of two Personalised PageRank vectors corresponding to the seed node a and b , respectively. Co-SimRank distinguishes from SimRank in that the SimRank value $s(a, b)$ counts only the first hitting time of two random surfers starting at nodes a and b , whereas CoSimRank values tallies all the hitting times of the two random surfers. As a result, CoSimRank produces more complete similarity scores than SimRank. In comparison to RoleSim*, the values of CoSimRank do not look at the automorphically equivalent patterns of the graph. However, the intuition of RoleSim* can be extended to Co-SimRank for semantic enhancement.

SimRank There have also been a variety of studies on SimRank algorithms recently (e.g., [8, 21, 24, 27, 29]). Wang et al. [27] presents a fast probabilistic Monte-Carlo algorithm, ExactSim, to evaluate single-source and top- k SimRank results on large-scale graphs with over 10^6 nodes effectively. ExactSim provides high-probability guarantees to yield ground truths with provable accuracy. Lu et al. [21] proposed a matrix sampling approach in combination with the steepest descent technique, which not only guarantees the sparsity of the involved matrix, but also speeds up the rate of convergence for a single-pair SimRank retrieval. Wei et al. [29] proposes PRSim, which resorts to the distribution of the reverse PageRank to accelerate single-source SimRank queries, achieving sublinear query time on

power-law graphs with small index size. READS [8] precalculates \sqrt{c} -walks and squeezes random walks into compact trees. In the query processing, READS searches the walks commencing at the query node u , and retrieves all the \sqrt{c} -random walks which hit the \sqrt{c} -walks of u . TSF [24] constructs one-way graphs for indexing through sampling an in-neighbour from the in-links of each node. In the query processing, the one-way graphs are utilised to retrieve random walks for SimRank evaluation.

3 RoleSim*

3.1 RoleSim* formulation

The central intuition underpinning RoleSim* follows a recursive concept that two distinct nodes are assessed to be similar if they

1. interact with the automorphically equivalent sets of in-neighbors, and
2. are in-linked by similar nodes out of automorphically equivalent sets.

The starting point for this recursion is to assign each pair of nodes a similarity score 1, meaning that initially no pairs of nodes are thought of to be more (or less) similar than others.

Notations Before illustrating the mathematical definition to reify the RoleSim* intuition, we introduce the following notations.

Let $G = (V, E)$ be a directed graph with a set of nodes V and a set of edges E . Let I_a be all in-neighbors of node a , and $|I_a|$ the cardinality of the set I_a . For a pair of nodes (a, b) in G , we denote by $I_a \times I_b = \{(x, y) \mid x \in I_a \text{ and } y \in I_b\}$ all in-neighboring pairs of (a, b) , and $s(a, b)$ the RoleSim* similarity score between nodes a and b . Using $I_a \times I_b$ and $s(a, b)$, we define a weighted complete bipartite graph, denoted by $\mathcal{K}_{|I_a|, |I_b|} = (I_a \cup I_b, I_a \times I_b)$, with each edge $(x, y) \in I_a \times I_b$ carrying the weight $s(a, b)$. We denote by $M_{a,b} (\subseteq I_a \times I_b)$ the maximum weighted matching in bipartite graph $\mathcal{K}_{|I_a|, |I_b|}$.

Example 2 Recall graph G in Figure 1. For nodes 1 and 2, their in-neighbors are sets $I_1 = \{4, 5\}$ and $I_2 = \{6, 7, 8\}$, respectively. The set of all in-neighboring pairs of $(1, 2)$ is $I_1 \times I_2 = \{(4, 6), (4, 7), (4, 8), (5, 6), (5, 7), (5, 8)\}$. The maximum matching of bipartite graph $(I_1 \cup I_2, I_1 \times I_2)$ is $M_{1,2} = \{(4, 6), (5, 7)\}$ (see the pairs in bold font in $I_1 \times I_2$).

Other notations frequently used throughout this paper are listed in Table 1.

RoleSim* Formula. Based on our aforementioned intuition, we formally formulate the RoleSim* model as follows:

$$s(a, b) = \beta \left(\lambda \times \underbrace{\frac{1}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} s(x, y)}_{\text{Part 1: average similarity over maximum matching } M_{a,b}} \right) + (1 - \lambda) \times \underbrace{\frac{1}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} s(x, y)}_{\text{Part 2: average similarity over } (I_a \times I_b) - M_{a,b}} \Big) + (1 - \beta) \quad (1)$$

Table 1 Description of main symbols

Symbol	Description
G	directed graph $G = (V, E)$ with a set nodes V and a set of edges E
I_a	all in-neighbors of node a in G
O_a	all out-neighbors of node a in G
$M_{a,b}$	maximum weighted matching in bipartite graph $\mathcal{K}_{ I_a , I_b } = (I_a \cup I_b, I_a \times I_b)$
β	damping factor ($0 < \beta < 1$)
λ	relative weight balancing similarities inside and outside $M_{a,b}$ ($0 < \lambda < 1$)
K	total number of iterations
δ	user-specified threshold parameter ($\delta > 0$)
ϵ	error bound between $s_k(*, *)$ and $s_k^\delta(*, *)$
θ	user-specified threshold parameter ($\theta \geq 0$)
d_{max}	maximum degree in a graph
$s(a, b)$	RoleSim* similarity score between nodes a and b
$s_k(a, b)$	k -th iterative RoleSim* similarity score between nodes a and b
$s_k^\delta(a, b)$	δ -threshold based RoleSim* similarity $s_k(a, b)$
$\rho(a, b)$	Importance value of two nodes a and b

In (1), for every pair of nodes (a, b) , the set of their in-neighboring pairs, $I_a \times I_b$, is split into two subsets: $I_a \times I_b = M_{a,b} \cup (I_a \times I_b - M_{a,b})$. As a result, the definition of RoleSim* consists of two parts: Part 1 is the average similarity over maximum matching $M_{a,b}$, indicating the contribution from (a, b) interacting with the automorphically equivalent set, $M_{a,b}$, of (a, b) 's in-neighbors pairs. Part 2 is the average similarity over $(I_a \times I_b) - M_{a,b}$, corresponding to the contribution from (a, b) being pointed to by the rest of (a, b) 's in-neighbors pairs out of automorphically equivalent set $M_{a,b}$.

It is worth highlighting that the reason why we use the denominator $|I_a| + |I_b| - |M_{a,b}|$ instead of $|M_{a,b}|$ in (1) is to guarantee that RoleSim* covers the traditional RoleSim model as a special case when $\lambda = 1$. More specifically, since $|M_{a,b}| = \min\{|I_a|, |I_b|\}$, it follows that $|I_a| + |I_b| - |M_{a,b}| = \max\{|I_a|, |I_b|\}$. When we apply this to (1) and set $\lambda = 1$, Part 2 of (1) becomes zero, and (1) reduces to the following traditional RoleSim equation:

$$s(a, b) = \frac{\beta}{\max\{|I_a|, |I_b|\}} \sum_{(x,y) \in M_{a,b}} s(x, y) + (1 - \beta) \quad (2)$$

The reason why RoleSim in (2) uses $\max\{|I_a|, |I_b|\}$ ($= |I_a| + |I_b| - |M_{a,b}|$) as the denominator instead of $|M_{a,b}|$ ($= \min\{|I_a|, |I_b|\}$) is to differentiate similarity values of the pairs $s(a, b)$ and $s(a, c)$ when $|I_b| \neq |I_c|$. The larger the difference between $|I_b|$ and $|I_c|$, the more dissimilar the similarity values of $s(a, b)$ and $s(a, c)$ should be. For example, recall the similarities of $s(8, 3)$ and $s(8, 1)$ in Figure 1, their in-neighbouring grids are shown in Figure 2. Note that $|I_3| = 3$ and $|I_1| = 2$, which implies that the similarity values of $s(8, 3)$ and $s(8, 1)$ should be different. However, $|M_{8,3}| = |M_{8,1}| = 2$. Therefore, if we

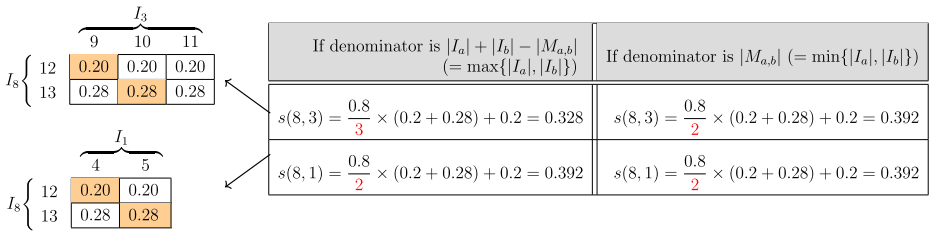


Figure 2 A demonstration on the affect of maximum matching denominator on similarity values

replace $|M_{a,b}|$ with $|I_a| + |I_b| - |M_{a,b}|$ in (1), the similarity values of $s(8, 3)$ and $s(8, 1)$ are considered as the same because

$$s(8, 3) = \frac{\beta}{|M_{8,3}|} (s(13, 9) + s(12, 10)) + (1 - \beta) = \frac{\beta}{2} (0.2 + 0.28) + (1 - \beta)$$

$$s(8, 1) = \frac{\beta}{|M_{8,1}|} (s(13, 4) + s(12, 5)) + (1 - \beta) = \frac{\beta}{2} (0.2 + 0.28) + (1 - \beta)$$

which is counter-intuitive to our common sense due to $|I_1| \neq |I_3|$. However, if we use (2), then the similarity values of $s(8, 3)$ and $s(8, 1)$ become

$$s(8, 3) = \frac{\beta}{|I_8| + |I_3| - |M_{8,3}|} (s(13, 9) + s(12, 10)) + (1 - \beta) = \frac{\beta}{3} (0.2 + 0.28) + (1 - \beta)$$

$$s(8, 1) = \frac{\beta}{|I_8| + |I_1| - |M_{8,1}|} (s(13, 9) + s(12, 10)) + (1 - \beta) = \frac{\beta}{2} (0.2 + 0.28) + (1 - \beta)$$

The larger the difference between $|I_3|$ and $|I_1|$, the more dissimilar the similarity scores of $s(8, 3)$ and $s(8, 1)$, which follows our intuition.

The relative weight of Part 1 and 2 is balanced by a user-controlled parameter $\lambda \in [0, 1]$. β is a damping factor between 0 and 1, which is often set to 0.6 or 0.8, implying that similarity propagation made with distant in-neighbors is penalised by an attenuation factor β across edges. When I_a (or I_b) = \emptyset , which implies the maximum matching $M_{a,b} = \emptyset$, we define Part 1 and Part 2 to be 0 in order to avoid the denominators of the fraction in Part 1 and 2 being zeros.

Fixed-Point Iteration To solve RoleSim* similarity $s(a, b)$ in (1), we adopt the following fixed-point iterative scheme:

$$s_0(a, b) = 1 \quad (\forall a, b) \tag{3}$$

$$s_{k+1}(a, b) = \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} s_k(x, y) + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} s_k(x, y) \right) + (1 - \beta) \tag{4}$$

where $s_k(a, b)$ denotes the RoleSim* score between nodes a and b at iteration k . Based on (3) and (4), we can iteratively compute all pairs of similarity scores $s_{k+1}(*, *)$ from those at the last iteration $s_k(*, *)$.

3.2 Axiomatic properties for RoleSim*

Symmetry, Boundedness, & Monotonicity Based on the definition of iterative similarity $s_k(a, b)$ in (3) and (4), we next show three axiomatic properties of RoleSim*, i.e., symmetry, boundedness, and non-increasing monotonicity, based on the following theorem.

Theorem 1 *The iterative RoleSim* $\{s_k(a, b)\}$ in (3) and (4) have the following key properties: for any node-pair (a, b) and each iteration $k = 0, 1, \dots$,*

1. **(Symmetry)** $s_k(a, b) = s_k(b, a)$
2. **(Boundedness)** $1 - \beta \leq s_k(a, b) \leq 1$
3. **(Monotonicity)** $s_{k+1}(a, b) \leq s_k(a, b)$

Proof 1. **(Symmetry)** By virtue of (3) and (4), $s_k(a, b) = s_k(b, a)$ follows immediately.
 2. **(Boundedness)** We will prove this by induction on k . For $k = 0$, it is apparent that $s_0(a, b) = 1 \in [1 - \beta, 1]$. For $k > 0$, we assume that $s_k(x, y) \leq 1$ holds, and will prove that $s_{k+1}(x, y) \leq 1$ holds as follows. Since

$$\begin{aligned}
 P_1 &:= \frac{1}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \underbrace{s_k(x, y)}_{\leq 1} \leq \frac{|M_{a,b}|}{|I_a| + |I_b| - |M_{a,b}|} \\
 &= \frac{\min\{|I_a|, |I_b|\}}{\max\{|I_a|, |I_b|\}} \leq 1 \\
 P_2 &:= \frac{1}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{s_k(x, y)}_{\leq 1} \leq \frac{|I_a \times I_b - M_{a,b}|}{|I_a| \times |I_b| - |M_{a,b}|} = 1
 \end{aligned}$$

Thus, (4) can be rewritten as

$$s_{k+1}(a, b) = \beta \times \left(\lambda \times \underbrace{P_1}_{\leq 1} + (1 - \lambda) \times \underbrace{P_2}_{\leq 1} \right) + (1 - \beta) \leq 1$$

On the other hand,

$$s_{k+1}(a, b) = \underbrace{\beta \times (\lambda \times P_1 + (1 - \lambda) \times P_2)}_{\geq 0} + (1 - \beta) \geq 1 - \beta$$

3. **(Monotonicity)** We will prove by induction on k . For $k = 0$, $s_0(a, b) = 1$. According to (4), it follows that

$$\begin{aligned}
 s_1(a, b) &= \beta \times \left(\lambda \times \underbrace{\frac{\min\{|I_a|, |I_b|\}}{\max\{|I_a|, |I_b|\}}}_{\leq 1} + (1 - \lambda) \times \underbrace{\frac{(|I_a| \times |I_b|) - |M_{a,b}|}{(|I_a| \times |I_b|) - |M_{a,b}|}}_{=1} \right) + (1 - \beta) \\
 &\leq \beta(\lambda + (1 - \lambda)) + (1 - \beta) = 1 = s_0(a, b)
 \end{aligned}$$

For $k > 0$, we assume that $s_{k+1}(a, b) \leq s_k(a, b)$ holds, and will prove that $s_{k+2}(a, b) \leq s_{k+1}(a, b)$ holds. According to (4), it follows that

$$\begin{aligned}
 s_{k+2}(a, b) &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \overbrace{s_{k+1}(x, y)}^{\text{(using hypothesis)} \leq s_k(x,y)} \right. \\
 &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{s_{k+1}(x, y)}_{\leq s_k(x,y)} \right) + (1 - \beta) \\
 &\leq \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} s_k(x, y) \right. \\
 &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} s_k(x, y) \right) + (1 - \beta) \\
 &= s_{k+1}(a, b)
 \end{aligned}$$

□

Theorem 1 indicates that, for every iteration $k = 0, 1, 2, \dots$, $\{s_k(a, b)\}$ is a bounded symmetric scoring function. Moreover, as $k \rightarrow \infty$, it can be readily verified that the exact solution $s(a, b)$ also is a bounded symmetric measure, which is similar to SimRank and RoleSim measures. In contrast, other measures (e.g., Hitting Time and Random Walk with Restart) are asymmetric.

It is worth noticing that, unlike SimRank iterative similarity values $\{s_k(a, b)\}$ that exhibit a non-decreasing trend (starting from 0 for any two distinct nodes a and b) w.r.t. the number of iterations k , RoleSim iterative similarity scores $\{s_k(a, b)\}$ show a non-increasing tendency (starting from 1 for any two distinct nodes a and b) w.r.t. k . This subtle difference makes many existing optimization techniques on SimRank not directly applicable to RoleSim*.

Existence & Uniqueness The bounded property and non-increasing property of RoleSim* iterative similarity values $\{s_k(a, b)\}$ w.r.t. k guarantee the existence and uniqueness of the exact RoleSim* solution $s(a, b)$ to (3) and (4), as indicated below:

Theorem 2 (Existence and Uniqueness) *There exists a unique solution $s(a, b)$ (i.e., the exact RoleSim score) to (3) and (4) such that the iterative RoleSim similarity $\{s_k(a, b)\}$ non-increasingly converges to it as the number of iterations k increases, i.e.,*

$$\lim_{k \rightarrow \infty} s_k(a, b) = s(a, b).$$

Proof (Existence) For each pair of nodes (a, b) , since the sequence $\{s_k(a, b)\}_k$ is lower-bounded by $(1 - \beta)$ (Property 2) and non-increasing (Property 3), by Monotone Convergence Theorem, $\{s_k(a, b)\}$ will converge to its infimum, denoted as $s(a, b)$, which is the exact RoleSim* solution, i.e., $\lim_{k \rightarrow \infty} s_k(a, b) = s(a, b)$.

(Uniqueness) For each pair of nodes (a, b) , suppose there exist two solutions, $s(a, b)$ and $\tilde{s}(a, b)$, that satisfy (4). We will prove that $s(a, b) = \tilde{s}(a, b)$. Let $\delta(a, b) := s(a, b) - \tilde{s}(a, b)$

and $\Delta := \max_{(a,b)}\{|\delta(a, b)|\}$. Then,

$$\begin{aligned} \delta(a, b) &= s(a, b) - \tilde{s}(a, b) \\ &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \overbrace{s(x, y) - \tilde{s}(x, y)}^{=\delta(x,y)} \right. \\ &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \overbrace{s(x, y) - \tilde{s}(x, y)}^{=\delta(x,y)} \right) \end{aligned}$$

Therefore, taking the absolute value of both sides and applying triangle inequality $|x + y| \leq |x| + |y|$ produces

$$\begin{aligned} |\delta(a, b)| &\leq \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \times \left| \sum_{(x,y) \in M_{a,b}} \delta(x, y) \right| \right. \\ &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \times \left| \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \delta(x, y) \right| \right) \\ &\leq \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \times \sum_{(x,y) \in M_{a,b}} \underbrace{|\delta(x, y)|}_{\leq \Delta} \right. \\ &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \times \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{|\delta(x, y)|}_{\leq \Delta} \right) \\ &\leq \beta(\lambda \times \Delta + (1 - \lambda) \times \Delta) = \beta \times \Delta \quad (\forall a, b) \end{aligned}$$

Thus, $\Delta = \max_{(a,b)}\{|\delta(a, b)|\} \leq \beta \times \Delta$, implying $\Delta = 0$, i.e., $s(a, b) = \tilde{s}(a, b)$. □

3.3 Iterative RoleSim* algorithm with guaranteed accuracy

In this section, we provide an iterative algorithm for retrieving RoleSim* similarity values, and give a concise error bound for the difference between iterative similarity scores as provided by our algorithm and actual (exact) scores.

Iterative Algorithm The fixed-point scheme in (3) and (4) implies an iterative algorithm for RoleSim* computation, as illustrated in Algorithm 1. It starts initialising all pairs of similarities to 1 (line 1), and carries out iterative computations of similarities for each pair of nodes (lines 3–15). If there are no in-neighbors for node a or b , $s(a, b)$ is set to $1 - \beta$ (lines 4–6). Otherwise, it finds maximum weighed matching $M_{a,b}$ in bipartite graph $(I_a \cup I_b, I_a \times I_b)$ (line 8), and averages the $(k - 1)$ -th iterative similarities over $M_{a,b}$ (resp. $(I_a \times I_b) - M_{a,b}$) to get w_1 (resp. w_2) (lines 9–14). Then, the weighted average of w_1 and w_2 is returned as score $s_k(a, b)$ at k -th iteration. This process continues till all pairs of similarities are computed for each iteration.

Algorithm 1 RoleSim* (G, β, λ, K).

```

Input : digraph  $G = (V, E)$ , decay factor  $\beta$ , relative weight  $\lambda$ , # of iterations  $K$ 
Output: RoleSim* scores  $s_K(*, *)$ .
1 initialise  $s_0(*, *) := 1$ 
2 for  $k := 1, 2, \dots, K$  do
3   foreach pair  $(a, b) \in V^2$  do
4      $I_a := \{x \in V \mid (x, a) \in E\}, \quad I_b := \{x \in V \mid (x, b) \in E\}$ 
5     if  $I_a = \emptyset$  or  $I_b = \emptyset$  then
6        $s_k(a, b) := 1 - \beta$ 
7     else
8        $M_{a,b} :=$  maximum matching in bipartite graph  $(I_a \cup I_b, I_a \times I_b)$ 
9       initialise  $t_1 := 0$  and  $t_2 := 0$ 
10      foreach  $(x, y) \in M_{a,b}$  do
11         $t_1 := t_1 + s_{k-1}(x, y)$ 
12      foreach  $(x, y) \in (I_a \times I_b) - M_{a,b}$  do
13         $t_2 := t_2 + s_{k-1}(x, y)$ 
14       $w_1 := \lambda / (|I_a| + |I_b| - |M_{a,b}|), \quad w_2 := (1 - \lambda) / (|I_a| \times |I_b| - |M_{a,b}|)$ 
15       $s_k(a, b) := \beta \times (w_1 \times t_1 + w_2 \times t_2) + (1 - \beta)$ 
16 return  $s_K(*, *)$ 

```

Complexity The computational cost of Algorithm 1 is shown in Theorem 3.

Theorem 3 It requires $O(K|E|^2)$ time and $O(|V|^2)$ memory for Algorithm 1 to retrieve RoleSim* similarity scores for $|V|^2$ node-pairs on graph $G = (V, E)$ with $|V|$ nodes and $|E|$ edges for K iterations.

Proof For each iteration k and each pair of nodes (a, b) , the computational time and memory required in each loop iteration of Algorithm 1 (lines 4–15) are described as follows:

Line	Time	Memory	Description
4	$O(I_a + I_b)$	$O(I_a + I_b)$	get in-neighborings for node a and b
6	$O(1)$	$O(1)$	initialise $s_k(a, b)$ if a and b have no in-neighbors
8	$O(I_a + I_b)$	$O(I_a \times I_b)$	finding the maximum matching in a weighted bipartite graph using Jonker-Volgenant algorithm [2]
9	$O(1)$	$O(1)$	initialise t_1 and t_2
10–11	$O(M_{a,b})$	$O(1)$	iteratively compute t_1
12–13	$O(I_a \times I_b - M_{a,b})$	$O(1)$	iteratively compute t_2
14–15	$O(1)$	$O(1)$	iteratively compute $s_k(a, b)$

Thus, for K iterations and $|V|^2$ node-pairs, the total time of Algorithm 1 is bounded by

$$\begin{aligned}
 & O\left(\sum_{(a,b) \in V^2} (K(2(|I_a| + |I_b|) + (|I_a| \times |I_b| - |M_{a,b}|) + |M_{a,b}|))\right) \\
 &= O\left(K \sum_{(a,b) \in V^2} (|I_a| \times |I_b|)\right) = O\left(K \underbrace{\sum_{a \in V} |I_a|}_{=|E|} \times \underbrace{\sum_{b \in V} |I_b|}_{=|E|}\right) = O(K|E|^2)
 \end{aligned}$$

Therefore, it entails $O(K|E|^2)$ time to retrieve $|V|^2$ pairs of RoleSim* scores.

Since $|V|^2$ pairs of similarities $s_{k-1}(*, *)$ at iteration $(k - 1)$ need to be prepared for retrieving $s_k(a, b)$ at next iteration k , the memory consumption of Algorithm 1 is bounded by $O(|V|^2)$. □

It is important to note that the $O(|V|^2)$ memory of Algorithm 1 hinders the scalability of RoleSim* computation on large graphs with millions of nodes. Therefore, in Section 5, on the top of Algorithm 1, we will propose a scalable algorithm for efficient RoleSim* similarity search on sizable graphs without loss of accuracy.

Error Bound We are now ready to investigate the error bound of the difference between the k -th iterative similarity $s_k(a, b)$ and exact one $s(a, b)$.

By virtue of the non-increasing monotonicity of $\{s_k(a, b)\}$, one can readily show that the exact $s(a, b)$ is the lower bound of all the iterative similarities $\{s_k(a, b)\}$, i.e., $s_k(a, b) \geq s(a, b) (\forall k)$. The following theorem further provides a concise upper bound to measure the closeness between $s_k(a, b)$ and $s(a, b)$.

Theorem 4 (Error Bound for Iterative RoleSim*) *For every iteration $k = 0, 1, 2, \dots$, the difference between $s_k(a, b)$ and $s(a, b)$ is bounded by*

$$s_k(a, b) - s(a, b) \leq \beta^{k+1} \quad (\forall a, b) \tag{5}$$

Proof We prove this by induction on k . For $k = 0$, $s_0(a, b) = 1$. According to Property 2 of Theorem 1, $1 - \beta \leq s_k(a, b) \leq 1$, implying that $1 - \beta \leq s(a, b) \leq 1$. Thus, $s_0(a, b) - s(a, b) \leq \beta$ holds.

For $k > 0$, we assume that $s_k(a, b) - s(a, b) \leq \beta^{k+1}$ holds, and will prove that $s_{k+1}(a, b) - s(a, b) \leq \beta^{k+2}$ holds. Subtracting (4) from (1) produces

$$\begin{aligned}
 s_{k+1}(a, b) - s(a, b) &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \overbrace{s_k(x, y) - s(x, y)}^{\leq \beta^{k+1}} \right. \\
 &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \overbrace{s_k(x, y) - s(x, y)}^{\leq \beta^{k+1}} \right) \\
 &\leq \beta(\lambda \times \beta^{k+1} + (1 - \lambda) \times \beta^{k+1}) = \beta^{k+2} \quad (\forall a, b)
 \end{aligned}$$

□

Theorem 4 derives a concise exponential upper bound for the difference between the k -th iterative similarity $s_k(a, b)$ and exact $s(a, b)$. Combining this bound with the non-increasing

monotonicity $s_k(a, b) \geq s(a, b)$, we can obtain that the k -th iterative error $s_k(a, b) - s(a, b)$ is between 0 and β^{k+1} . Moreover, Theorem 4 also implies that, given desired accuracy $\epsilon > 0$, the total number of iterations required for computing RoleSim* similarity is $k = \lceil \log_{\beta} \epsilon \rceil$.

It is worth noticing that the equality sign in our error estimation (5) is reachable, highlighting the tightness of the bound, as illustrated in Example 3.

Example 3 Consider the graph G in Figure 3. Given $\beta = 0.8$ and $\lambda = 0.7$, let us evaluate the RoleSim* similarity $s(c, d)$ iteratively via (3) and (4). For iteration $k = 0$, it is apparent that $s_0(*, *) = 1$. When $k = 1$, it follows from $|I_c| = |I_d| = |M_{c,d}| = 2$ and (4) that

$$s_1(c, d) = 0.8 \times \left(\frac{0.7}{2+2-2} \times (1 + 1) + \frac{1-0.7}{2 \times 2-2} \times (1 + 1) \right) + (1 - 0.8) = 1$$

Since the exact solution is $s(c, d) = 0.36$, when $k = 1$, we have

$$s_1(c, d) - s(c, d) = 1 - 0.36 = 0.64 = 0.8^{1+1} = \beta^{k+1}$$

Therefore, the equality in (5) is attainable on G when $(k, \beta) = (1, 0.8)$.

4 Threshold-based RoleSim*

In this section, we propose our threshold-based RoleSim* model that substantially speeds up the computation of RoleSim* similarities with only a little sacrifice in accuracy. We will establish provable error bounds on our threshold-based RoleSim* model with respect to a user-specified threshold parameter δ , which is a speed-accuracy tradeoff.

Through the iterative computation of RoleSim* via (3) and (4), we notice that there are a significant number of node-pairs whose iterative similarity scores $s_k(*, *)$ are very close to their convergent values $s(*, *)$ and thus will not change much in subsequent iterations as k grows. To accelerate RoleSim* computation, we have the following two observations for eliminating such pairs from the unnecessary RoleSim* computations, with guaranteed accuracy.

Observation 1 *If the RoleSim* similarity scores of two adjacent iterations, $s_{k-1}(*, *)$ and $s_k(*, *)$, become quite close to each other after some iterations, then the RoleSim* iterative sequence $\{s_k(*, *)\}$ from some iteration k_0 onwards are very close to the exact solution $s(*, *)$ as well.*

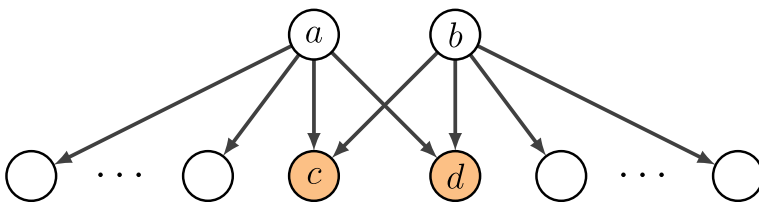


Figure 3 The equality sign in our estimate bound (5) for iterative RoleSim* computation is attainable, e.g., it can be verified from graph G that there exist $(k, \beta) = (1, 0.8)$ such that $s_1(c, d) - s(c, d) = 1 - 0.36 = 0.8^{1+1} = \beta^{k+1}$ holds

This observation is based on Cauchy Convergence Criterion to test whether a sequence has a limit. Precisely, for any small user-specified threshold $\delta > 0$, this criterion implies that

$$\lim_{k \rightarrow +\infty} s_k(*, *) = s(*, *) \Leftrightarrow \exists k_0 \text{ s.t. } |s_k(*, *) - s_{k+1}(*, *)| < \delta \quad (\forall k > k_0)$$

We apply this criterion to skip unnecessary iterative computations for node-pairs whose RoleSim* scores of two consecutive iterations are very small. More specifically, after some iterations, once the gap between $s_{k-1}(*, *)$ and $s_k(*, *)$ is below the threshold δ , instead of employing (4) to iteratively compute $s_{k+1}(*, *)$ from $s_k(*, *)$, we simply supersede $s_{k+1}(*, *)$ by the value of $s_k(*, *)$. Therefore, we define the following threshold-based RoleSim* similarity $\bar{s}_k^\delta(*, *)$ based on Observation 1:

$$\bar{s}_0^\delta(a, b) = 1$$

$$\bar{s}_{k+1}^\delta(a, b) = \begin{cases} \bar{s}_k^\delta(a, b) & \text{if } \bar{s}_{k-1}^\delta(a, b) - \bar{s}_k^\delta(a, b) < \delta \text{ and } k \geq 1 \\ \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \bar{s}_k^\delta(x, y) \right) & \text{otherwise} \end{cases} \tag{6a}$$

$$\left(+ \frac{1-\lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \bar{s}_k^\delta(x, y) \right) + (1 - \beta) \tag{6b}$$

To quantify the difference between the threshold-based RoleSim* similarity $\bar{s}_k^\delta(*, *)$ in (6) and the conventional one $s_k(*, *)$ in (4) at each iteration k , we show the following theorem.

Theorem 5 *Given a threshold δ , for any number of iterations $k = 0, 1, 2, \dots$, there exists a positive integer k_0 such that for any $k > k_0$ and two nodes (a, b) ,*

$$\bar{s}_k^\delta(a, b) - s_k(a, b) \leq \epsilon_0 \quad \text{with} \quad \epsilon_0 = \frac{\beta(1-\beta^{k-k_0})}{1-\beta} \delta \tag{7}$$

where k_0 is the minimum integer that guarantees $\bar{s}_{k_0-1}^\delta(a, b) - \bar{s}_{k_0}^\delta(a, b) < \delta$.

Proof When $k = 0$, it is apparent that $s_0(a, b) = \bar{s}_0^\delta(a, b) = 1$.

Let k_0 be the minimum integer that guarantees $\bar{s}_{k_0-1}^\delta(a, b) - \bar{s}_{k_0}^\delta(a, b) < \delta$ for any two nodes a and b . When $0 < k < k_0$, in the case of $\bar{s}_{k-1}^\delta(a, b) - \bar{s}_k^\delta(a, b) \geq \delta$, $\bar{s}_k^\delta(a, b)$ is iteratively computed from (6a). Hence,

$$\bar{s}_k^\delta(a, b) = s_k(a, b) \quad (\forall k = 0, 1, \dots, k_0, \quad \forall a, b).$$

When $k \geq k_0$, in the case of $\bar{s}_{k-1}^\delta(a, b) - \bar{s}_k^\delta(a, b) < \delta$ for any nodes a and b , $\bar{s}_{k+1}^\delta(a, b)$ is directly obtained by (6b), thereby leading to its deviation from $s_{k+1}(a, b)$ since the k_0 -th

iteration. In this case, to quantify the gap between $\bar{s}_{k+1}^\delta(a, b)$ and $s_{k+1}(a, b)$, we notice that

$$\begin{aligned} & s_{k_0}(a, b) - s_{k_0+1}(a, b) \\ &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \underbrace{(s_{k_0-1}(x, y) - s_{k_0}(x, y))}_{\leq \delta} \right) \\ & \quad + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{(s_{k_0-1}(x, y) - s_{k_0}(x, y))}_{\leq \delta} \Big) \\ &\leq \beta \times \left(\frac{\lambda \times (|M_{a,b}| \times \delta)}{|I_a| + |I_b| - |M_{a,b}|} + \frac{(1 - \lambda) \times (|I_a| \times |I_b| - |M_{a,b}|) \times \delta}{|I_a| \times |I_b| - |M_{a,b}|} \right) \\ &\leq \beta \times (\lambda \times \delta + (1 - \lambda) \times \delta) = \beta \times \delta \end{aligned}$$

Similarly,

$$\begin{aligned} & s_{k_0+1}(a, b) - s_{k_0+2}(a, b) \\ &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \underbrace{(s_{k_0}(x, y) - s_{k_0+1}(x, y))}_{\leq \beta \times \delta} \right) \\ & \quad + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{(s_{k_0}(x, y) - s_{k_0+1}(x, y))}_{\leq \beta \times \delta} \Big) \\ &\leq \beta^2 \times \delta \end{aligned}$$

Iteratively, we can obtain that

$$s_{k_0+i-1}(a, b) - s_{k_0+i}(a, b) \leq \beta^i \times \delta \quad (\forall i = 0, 1, 2, \dots)$$

Since $\bar{s}_k^\delta(a, b) = \bar{s}_{k_0}^\delta(a, b) = s_{k_0}(a, b) \quad (\forall k \geq k_0)$, it follows that

$$\begin{aligned} & \bar{s}_k^\delta(a, b) - s_k(a, b) = s_{k_0}(a, b) - s_k(a, b) \\ &= \underbrace{(s_{k_0}(a, b) - s_{k_0+1}(a, b))}_{\leq \beta \times \delta} + \underbrace{(s_{k_0+1}(a, b) - s_{k_0+2}(a, b))}_{\leq \beta^2 \times \delta} + \dots + \underbrace{(s_{k-1}(a, b) - s_k(a, b))}_{\leq \beta^{k-k_0} \times \delta} \\ &\leq \delta \times \sum_{i=1}^{k-k_0} \beta^i = \delta \times \frac{\beta(1 - \beta^{k-k_0})}{1 - \beta} \quad (\forall k \geq k_0) \end{aligned}$$

□

Theorem 5 indicates that the threshold δ is a user-controlled parameter, which is a speed-accuracy trade-off. A small setting of δ ensures a high accuracy of $\bar{s}_k^\delta(*, *)$, but at the cost of more time for iterations, since only a small number of node-pairs can be pruned. In contrast, larger δ can discard more pairs of nodes from iterative computations, but would produce a larger error bound ϵ_1 between $\bar{s}_k^\delta(*, *)$ and $s_k(*, *)$.

Example 4 Consider the graph G in Figure 4a. Given threshold $\delta = 0.01$, decay factor $\beta = 0.6$, and relative weight $\lambda = 0.8$, for pair $(a, b) = (2, 3)$, in Figure 4b, we see that

$$\bar{s}_3^{0.01}(2, 3) - \bar{s}_4^{0.01}(2, 3) = 0.7139 - 0.7077 = 0.0062 < \delta = 0.01.$$

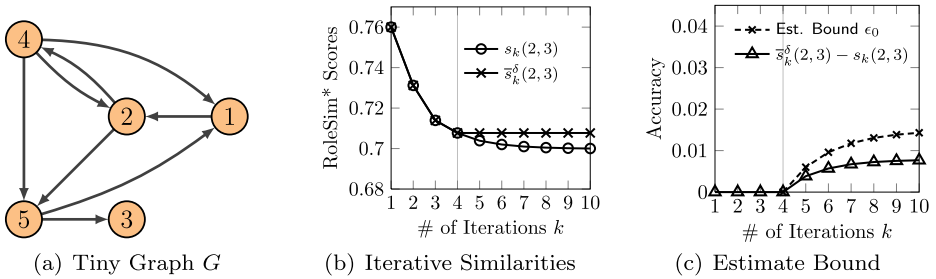


Figure 4 Error bound on the gap $\bar{s}_k^\delta(2, 3) - s_k^\delta(2, 3)$ based on Observation

Thus, there exists an integer $k_0 = 4$, such that the error bound in (7) holds for all $k > k_0$, as depicted in Figure 4c. For example, when $k = 10$, we have

$$\bar{s}_{10}^{0.01}(2, 3) - s_{10}(2, 3) = 0.7077 - 0.7 = 0.0077 \leq \epsilon_0$$

$$\text{with } \epsilon_0 = \frac{\beta(1-\beta^{(k-k_0)})}{1-\beta} \delta = \frac{0.6 \times (1-0.6^{10-4})}{1-0.6} \times 0.01 = 0.0143. \quad \square$$

On the top of Observation 1, to enable a further speedup in the computation of RoleSim*, our second observation for discarding unnecessary RoleSim* iterations is the following:

Observation 2 For a given threshold δ , after some iterations, if the RoleSim* similarity score $s_k(*, *)$ is within a small δ -neighbourhood of $(1 - \beta)$, then the RoleSim* iterative sequence $\{s_k(*, *)\}$ from some iteration k_1 onwards is also within the δ -neighbourhood of $(1 - \beta)$.

This observation comes from the non-increasing property and lower bound of the RoleSim* iterative sequence $\{s_k(*, *)\}_{k=1}^\infty$ that we derived in Theorem 1. We notice that there are a number of pairs whose iterative RoleSim* similarity scores are very close to the lower bound $(1 - \beta)$, but have not converged to the exact value of $(1 - \beta)$ yet. Iteratively computing such pairs via (4) till convergence is cost-inhibitive. We observe that, when $s_k(*, *)$ becomes close to $(1 - \beta)$, the value of $s_{k+1}(*, *)$ in the subsequent iteration is even closer to $(1 - \beta)$ than $s_k(*, *)$. As a result, there are opportunities to terminate earlier the iterative computations of $s_{k+i}(*, *)$ by simply replacing the value of $s_{k+i}(*, *)$ with $(1 - \beta)$ for all $i = 1, 2, \dots$, once $s_k(*, *)$ falls into the δ -neighborhood of $(1 - \beta)$, as illustrated below:

$$\underline{s}_0^\delta(a, b) = 1$$

$$\underline{s}_{k+1}^\delta(a, b) = \begin{cases} 1 - \beta & \text{if } \underline{s}_k^\delta(a, b) < (1 - \beta) + \delta \\ \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \underline{s}_k^\delta(x, y) \right) & \text{otherwise} \\ + \frac{1-\lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underline{s}_k^\delta(x, y) \end{cases} + (1 - \beta) \quad (8b)$$

To distinguish $\bar{s}_k^\delta(*, *)$ in (6), we denote by $\underline{s}_k^\delta(*, *)$ in (8) the threshold-based RoleSim* similarity based on Observation 2. By definition, it is discerned that $\underline{s}_k^\delta(*, *) \leq s_k(*, *) \leq \bar{s}_k^\delta(*, *)$. The following theorem provides the bound for the difference between $s_k(*, *)$ in (4) and $\underline{s}_k^\delta(*, *)$ in (8).

Theorem 6 Given a threshold δ , for any number of iterations $k = 0, 1, 2, \dots$, there exists a positive integer k_1 such that for any $k \geq k_1$ and two nodes (a, b) , it follows that

$$s_k(a, b) - s_k^\delta(a, b) \leq \epsilon_1 \quad (\forall k \geq k_1) \tag{9}$$

with $\epsilon_1 = \delta - \frac{\rho^{k_1} - \rho^k}{1 - \rho} \times \xi$, $\rho = \beta(1 - \lambda)$, $\xi = 1 - \max_{(a,b)} \{s_1(a, b)\}$

where k_1 is the minimum integer that guarantees $s_{k_1}^\delta(a, b) < 1 - \beta + \delta$.

Proof We first find the lower bound on the gap between $s_k(a, b)$ and $s_{k+i}(a, b)$. By definition of (4), when $k = 0$, it follows from $s_0(*, *) = 1$ that

$$s_0(a, b) - s_1(a, b) = 1 - \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \times |M_{a,b}| + (1 - \lambda) \right) - (1 - \beta)$$

Plugging $|M_{a,b}| = \min(|I_a|, |I_b|)$ and $|I_a| + |I_b| - |M_{a,b}| = \max(|I_a|, |I_b|)$ to the above equation yields

$$s_0(a, b) - s_1(a, b) = \xi_{a,b} \quad \text{with} \quad \xi_{a,b} = \lambda\beta \times \left(1 - \frac{\min(|I_a|, |I_b|)}{\max(|I_a|, |I_b|)} \right)$$

Let $\xi = \min_{a,b} \{\xi_{a,b}\} = 1 - \max_{a,b} \{s_1(a, b)\}$, we have

$$s_0(a, b) - s_1(a, b) \geq \xi$$

When $k = 1$, it follows that

$$\begin{aligned} s_1(a, b) - s_2(a, b) &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \underbrace{(s_0(x, y) - s_1(x, y))}_{\geq \xi} \right. \\ &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{(s_0(x, y) - s_1(x, y))}_{\geq \xi} \right) \\ &\geq \beta\xi \times \left(\underbrace{\frac{\lambda|M_{a,b}|}{|I_a| + |I_b| - |M_{a,b}|}}_{\geq 0} + \underbrace{\frac{(1 - \lambda) \times (|I_a| \times |I_b| - |M_{a,b}|)}{|I_a| \times |I_b| - |M_{a,b}|}}_{=1-\lambda} \right) \\ &\geq \rho \times \xi \quad \text{with} \quad \rho = \beta(1 - \lambda) \end{aligned}$$

Similarly, when $k = 2$, we have

$$\begin{aligned} s_2(a, b) - s_3(a, b) &= \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} \underbrace{(s_1(x, y) - s_2(x, y))}_{\geq \rho \times \xi} \right. \\ &\quad \left. + \frac{1 - \lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} \underbrace{(s_1(x, y) - s_2(x, y))}_{\geq \rho \times \xi} \right) \\ &\geq \beta(1 - \lambda)\rho \times \xi = \rho^2 \times \xi \end{aligned}$$

Iteratively, we have

$$s_k(a, b) - s_{k+1}(a, b) \geq \rho^k \times \xi \quad (\forall k = 0, 1, \dots)$$

Therefore,

$$\begin{aligned}
 & s_k(a, b) - s_{k+i}(a, b) \\
 &= \underbrace{(s_k(a, b) - s_{k+1}(a, b))}_{\geq \rho^k \times \xi} + \underbrace{(s_{k+1}(a, b) - s_{k+2}(a, b))}_{\geq \rho^{k+1} \times \xi} + \dots + \underbrace{(s_{k+i-1}(a, b) - s_{k+i}(a, b))}_{\geq \rho^{k+i-1} \times \xi} \\
 &\geq \sum_{j=k}^{k+i-1} \rho^j \times \xi = \frac{\rho^k(1 - \rho^i)}{1 - \rho} \times \xi \tag{10}
 \end{aligned}$$

Next, capitalising on the lower bound for $s_k(a, b) - s_{k+i}(a, b)$, we are going to find the upper bound for $s_{k_1+i}(a, b) - \underline{s}_{k_1+i}^\delta(a, b)$. Let k_1 be the minimum integer that guarantees $\underline{s}_{k_1}^\delta(a, b) \leq (1 - \beta) + \delta$ for any two nodes a and b . Then, when $k = k_1$, we have

$$s_{k_1}(a, b) = \underline{s}_{k_1}^\delta(a, b) \leq (1 - \beta) + \delta \Rightarrow 1 - \beta \geq s_{k_1}(a, b) - \delta$$

Iteratively, when $k = k_1 + i$ ($i \geq 1$), it follows from $\underline{s}_{k_1+i}^\delta(a, b) = 1 - \beta$ that

$$\begin{aligned}
 & s_{k_1+i}(a, b) - \underline{s}_{k_1+i}^\delta(a, b) \\
 &= s_{k_1+i}(a, b) - \underbrace{(1 - \beta)}_{\geq s_{k_1}(a, b) - \delta} \leq \delta - \underbrace{(s_{k_1}(a, b) - s_{k_1+i}(a, b))}_{\geq \frac{\rho^{k_1}(1 - \rho^i)}{1 - \rho} \times \xi \text{ by (10)}} \leq \delta - \frac{\rho^{k_1}(1 - \rho^i)}{1 - \rho} \times \xi
 \end{aligned}$$

Thus,

$$s_k(a, b) - \underline{s}_k^\delta(a, b) \leq \epsilon_1 \quad \text{with} \quad \epsilon_1 = \delta - \frac{\rho^{k_1} - \rho^k}{1 - \rho} \times \xi \quad (\forall k \geq k_1)$$

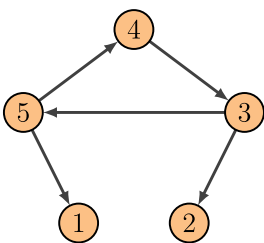
□

Example 5 Consider the digraph G in Figure 5a. Given a threshold $\delta = 0.1$, decay factor $\beta = 0.2$, and relative weight $\lambda = 0.55$, for node-pair $(a, b) = (2, 3)$, it is discerned that, when k grows to 3, the value of $\bar{s}_k^\delta(2, 3)$ will fall into the δ -neighborhood of $(1 - \beta)$, i.e., $\bar{s}_3^{0.1}(2, 3) = 0.899 < 1 - \beta + \delta = 1 - 0.2 + 0.1 = 0.9$. Thus, there exists an integer $k_1 = 3$, such that the error bound in (9) holds for all $k > k_1$, as shown in Figure 5c. Then, e.g., when $k = 4$ ($> k_1$), we have

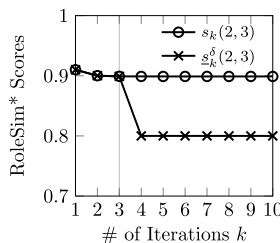
$$s_4(2, 3) - \underline{s}_4^\delta(2, 3) = 0.89889 - 0.8000 = 0.09889 \leq \epsilon_1$$

$$\text{where } \rho = \beta(1 - \lambda) = 0.2 \times (1 - 0.55) = 0.09, \quad \xi = 1 - \max_{(a,b)} \{s_1(a, b)\} = 0.09,$$

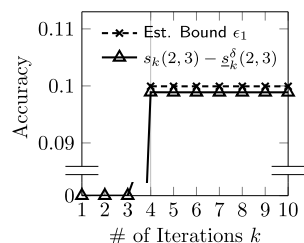
$$\text{and } \epsilon_1 = \delta - \frac{\rho^{k_1} - \rho^k}{1 - \rho} \xi = 0.1 - \frac{0.09^3 - 0.09^4}{1 - 0.09} \times 0.09 = 0.09993.$$



(a) Tiny Graph G



(b) Iterative Similarities



(c) Estimate Bound

Figure 5 Error bound on the gap $s_k^\delta(2, 3) - \underline{s}_k^\delta(2, 3)$ based on Observation 2

Putting Them All Together Combining Observations 1 and 2, we next propose the following complete scheme for threshold-based RoleSim* retrieval. To differentiate the notation from $\bar{s}_k^\delta(*, *)$ in (6) and $\underline{s}_k^\delta(*, *)$ in (8), we denote by $s_k^\delta(*, *)$ the threshold-based RoleSim* similarity for our complete scheme combining both Observations 1 and 2, which is defined as follows:

$$s_0^\delta(a, b) = 1$$

$$s_k^\delta(a, b) = \begin{cases} s_k^\delta(a, b) & \text{if } s_{k-1}^\delta(a, b) - s_k^\delta(a, b) < \delta \text{ and } k \geq 1 & (11a) \\ 1 - \beta & \text{if } s_k^\delta(a, b) < (1 - \beta) + \delta & (11b) \\ \beta \times \left(\frac{\lambda}{|I_a| + |I_b| - |M_{a,b}|} \sum_{(x,y) \in M_{a,b}} s_k^\delta(x, y) \right) & \text{otherwise} & (11c) \\ + \frac{1-\lambda}{|I_a| \times |I_b| - |M_{a,b}|} \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} s_k^\delta(x, y) \end{cases} + (1 - \beta)$$

By virtue of Theorems 5 and 6, the following upper bound on the difference between $s_k^\delta(*, *)$ and $s_k(*, *)$ is immediate.

Corollary 1 (Error Bound for Threshold-Based RoleSim* Iteration) *Given a threshold δ , for any number of iterations $k = 0, 1, 2, \dots$, there exist two positive integers k_0 and k_1 such that for any two nodes (a, b) ,*

$$|s_k(a, b) - s_k^\delta(a, b)| \leq \epsilon \text{ with } \epsilon = \begin{cases} \min \{\epsilon_0, \epsilon_1\} & \text{if } k \geq \max \{k_0, k_1\} \\ \epsilon_0 & \text{if } k_0 \leq k \leq k_1 \\ \epsilon_1 & \text{if } k_1 \leq k \leq k_0 \\ 0 & \text{if } 0 \leq k \leq \min \{k_0, k_1\} \end{cases}$$

where $\epsilon_0 = \frac{\beta(1-\beta^{k-k_0})}{1-\beta} \delta$, and $\epsilon_1 = \delta - \frac{\rho^{k_1-\rho^k} \xi}{1-\rho}$ with $\rho = \beta(1 - \lambda)$ and $\xi = 1 - \max_{a,b} \{s_1(a, b)\}$; k_0 (resp. k_1) is the minimum positive integer that ensures $s_{k_0-1}^\delta(a, b) - s_{k_0}^\delta(a, b) < \delta$ (resp. $s_{k_1}^\delta(a, b) < 1 - \beta + \delta$) holds.

5 Scaling RoleSim* search on large graphs

In this section, we propose efficient techniques that enable RoleSim* similarity search to scale well on sizable graphs with billions of edges. It is noticed that our iterative method for RoleSim* search by Algorithm 1 needs to memoise all $|V|^2$ pairs of similarities $\{s_k(*, *)\}$ at iteration k for computing any similarity at iteration $(k + 1)$. On small graphs, this algorithm runs very fast for all-pairs search. However, real graphs are often large with millions of nodes. The $O(|V|^2)$ memory required by Algorithm 1 would jeopardise its scalability over massive graphs. Moreover, in many real-world applications, users are often interested in partial-pairs similarity search. For instance, in a DBLP collaboration network, one would like to find who are Prof. Jennifer Widom’s close collaborators. In a social graph, one wants to know who are Thomas’s close friends on Instagram. In a web graph, one wishes to identify which web pages are relevant to a given query page. These applications call for a need to devise a scalable method that retrieves partial-pairs RoleSim* similarities within a small amount of memory. Formally, we are ready to solve the following RoleSim* search problem:

PROBLEM (Single-source RoleSim* Similarity Search).

Given: a graph $G = (V, E)$, a query node $q \in V$, and a desired depth K^1

¹The desired depth K is equivalent to the total number of iterations in Algorithm 1.

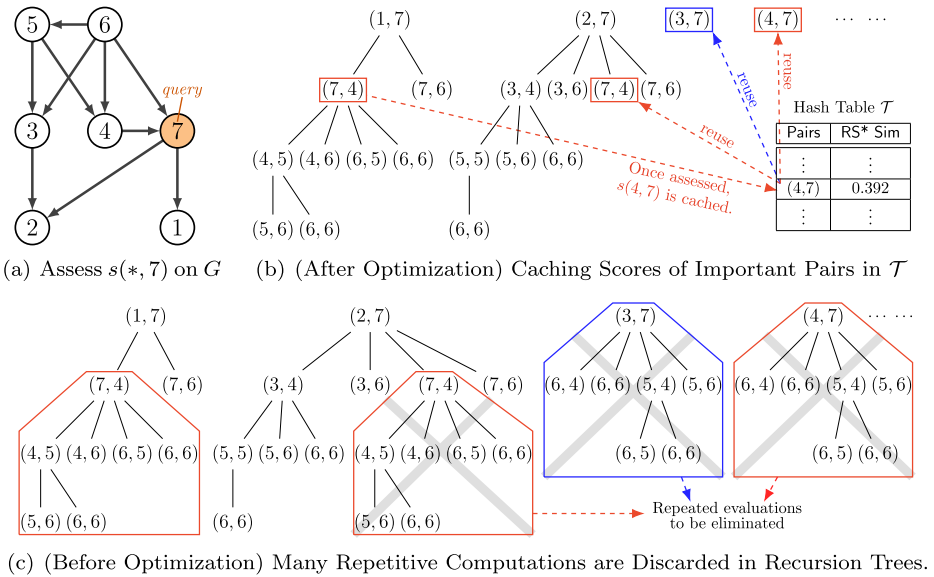


Figure 6 Single-source RoleSim* method that caches the similarities of important pairs eliminates many unnecessary recomputations in DFS backtracking

Retrieve: $|V|$ pairs of RoleSim* similarities $\{s_K(*, q)\}$ between all nodes in G and query q in a scalable manner.

To avoid using $O(|V|^2)$ memory, the central idea underpinning our method is judiciously implementing caching techniques on only a small portion of node-pairs that involve heavily repetitive similarity computations. More specifically, to evaluate each pair (u, q) 's similarity for single-source $\{s(*, q)\}$ retrieval, we start at each root pair (u, q) , and employ a depth-first search (DFS) to traverse all the in-neighboring pairs within k hops from the root (u, q) , recursively, against the in-coming edges of the graph in a depthward movement before backtracking when a desired depth k is reached or a “dead end” (*i.e.*, a pair (x, y) with either node x or y having no in-neighbours) occurs in any iteration. The iterative recurrence for retrieving each root pair (u, q) can be diagrammed by a recursion tree. For example, given graph G in Figure 6a with query node $q = 7$ and desired depth $K = 4$, the recursion tree for the recurrence to retrieve each $s(x, 7)$ ($\forall x \in V$) through DFS is depicted in Figure 6c, respectively. We have the following two observations:

- (1) There are a number of repeated computations among these recursion trees. For instance, $s(4, 7)$ is repetitively evaluated three times (circled in red). If the result of $s(4, 7)$ is cached and reused in subsequent recurrence, a number of unnecessary RoleSim* computations can be avoided.
- (2) When breaking down the traversal of $s(3, 7)$ and $s(4, 7)$, we notice that their unfolded recurrence structures (circled in blue) are exactly the same, which is due to the same in-neighboring structures of nodes 3 and 4, *i.e.*, $I(3) = I(4)$. If the previously cached results of $s(4, 7)$ can be used again for evaluating any other $\{s(x, 7)\}$ (for all $x \in V - \{4\}$) with the same in-neighboring structure of node 7), many duplicate computations can be skipped further.

It is worth mentioning that the parameter K here is the desired depth of search to control the height of the traversed recursion tree, which provides a user-controlled effect of the

speed and accuracy for computing RoleSim* similarity. For example, when K is small, the computation of $s_K(v, q)$ is fast, but this would increase the error between $s_K(v, q)$ and the exact solution $s(v, q)$. When K is large, $s_K(v, q)$ approaches $s(v, q)$, which achieves high accuracy, but will take more time, as it requires more steps to traverse the recursion tree. Moreover, the user-specified K also effectively avoid ending up an infinite loop of a circle while traversing the graph for RoleSim* retrieval.

Based on these observations, we devise a caching approach in backtracking of DFS to minimise duplicate RoleSim* similarity computations. Different from Algorithm 1 that requires $O(|V|^2)$ memory space to cache all-pairs similarities, we select only the “important” pairs for memoization. We first define the “importance” of a node-pair as follows.

Definition 1 Let (x, y) be a pair of nodes, and $|O_x|$ be the out-degree of node x , then the importance of the pair (x, y) , denoted as $\rho(x, y)$, is defined as

$$\rho(x, y) := |O_x| \times |O_y|$$

Intuitively, Definition 1 uses degree centrality to evaluate the “importance” of a pair since a pair (x, y) is likely to be “important” if nodes x and y are linked to a large number of nodes.

According to Definition 1, during DFS backtracking, when each pair (x, y) is visited, we first check if $\rho(x, y) \geq \theta$ to determine whether this pair is worthy of being cached, where θ is user-specified threshold between 0 and d_{\max}^2 , which is a space-speed tradeoff. When θ is set to 0, all pairs in the recursion tree are memoised, which in the worst case will reduce to the case of Algorithm 1. When $\theta > d_{\max}^2$, no caching techniques apply, which degrades to the naive recursive retrieval of RoleSim* similarities, being rather cost-inhibitive. In other words, the selection of θ value is a trade-off problem between memory and execution time, the smaller the θ , the larger the number of memoised pairs in memory and consequently the lower the execution time. Therefore, an appropriate selection of θ plays an important role in providing a good balance between the computational time and memory space (*i.e.*, the number of memoised pairs to be retrieved).

To avoid caching insignificant pairs, we often set θ to the first quartile of the pairwise out-degree set $\{\rho(x, y)\}_{(x,y) \in V^2}$ of the graph, which guarantees more than slightly important pairs to be cached using a moderate amount of memory. This is because there are close relationships between θ and the number of retrieved pairs N . As demonstrated by our extensive experiments in Figure 7, when θ is less than the first quartile of the pairwise out-degree set $\{\rho(x, y)\}_{(x,y) \in V^2}$ on each dataset³ (*e.g.*, DBLP, Bitcoin- α and P2P), there are a large number of memoised pairs with huge space requirement, but the computation is very fast. When θ is larger than the first quartile of $\{\rho(x, y)\}_{(x,y) \in V^2}$, the execution time increases significantly whereas the number of retrieved pairs decreases sharply. Only when θ is around the first quartile of $\{\rho(x, y)\}_{(x,y) \in V^2}$ (*e.g.*, $\theta \approx 10$ on DBLP, 5 on Bitcoin- α , 60 on P2P), there is a good balance between the computational time and memory space (with the balancing point circled in red). Thus, we empirically set θ to the first quartile of the pairwise out-degree set $\{\rho(x, y)\}_{(x,y) \in V^2}$ of a graph.

Once we decide that a visited pair (x, y) deserves to be cached, we next employ an unordered hash table \mathcal{T} for memoising. Precisely, we first check whether the key (*i.e.*, node-pair (x, y)) exists in the hash table. If not, we compute the RoleSim* similarity $s(x, y)$

² $d_{\max} := \max_{x \in V} \{|O_x|\}$ is the maximum out-degree of the graph.

³<https://snap.stanford.edu/data/index.html>

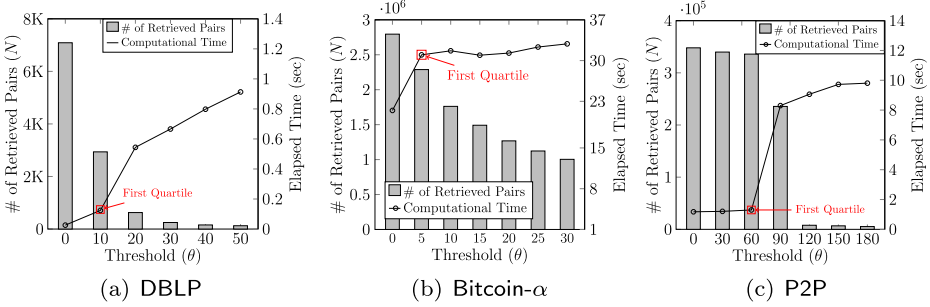


Figure 7 An appropriate choice of θ provides a good balance between computational time and memory space (i.e., # of memorised pairs)

once, and add $\langle \text{key}, \text{value} \rangle := \langle (x, y), s(x, y) \rangle$ to the hash table. Otherwise, we just retrieve the cached similarity value $s(x, y)$ corresponding to the key (x, y) from the hash table instead of computing the similarity $s(x, y)$ again, thus significantly boosting the performance for single-source RoleSim* search. Note that, due to the symmetry of RoleSim* similarity $s(x, y) = s(y, x)$, when hashing the pair (x, y) , we will swap x and y beforehand if $x > y$, to avoid both (x, y) and (y, x) being hashed.

Algorithm 2 Single-Source RoleSim* ($G, q, \beta, \lambda, K, \theta$)

```

Input : graph  $G$ , query  $q$ , decay factor  $\beta$ , relative weight  $\lambda$ , depth  $K$ , threshold  $\theta$ 
Output: single-source RoleSim* scores  $s_K(*, q)$ .
1 initialise unordered hash table  $\mathcal{T} := \emptyset$ 
2 if  $I_q = \emptyset$  then  $s_K(*, q) := 1 - \beta$ 
3 else
4   foreach node  $u \in V$  do
5      $s_K(u, q) := \text{Single-Pair}(G, u, q, \mathcal{T}, \beta, \lambda, K, \theta)$ 
6 return  $s_K(*, q)$ 

Function Single-Pair ( $G, a, b, \mathcal{T}, \beta, \lambda, k, \theta$ ):
7   initialize  $s := 0$ 
8   if  $k = 0$  or  $I_a = \emptyset$  or  $I_b = \emptyset$  then return  $s := 1 - \beta$ 
9   initialise  $t_1 := 0$  and  $t_2 := 0$ 
10   $M_{a,b} :=$  maximum matching in bipartite graph  $(I_a \cup I_b, I_a \times I_b)$ 
11  foreach  $(x, y) \in I_a \times I_b$  do
12    if  $\mathcal{T}.\text{containsKey}(x, y, k)$  then
13       $t := \mathcal{T}(x, y, k)$ 
14    else
15       $t := \text{Single-Pair}(G, x, y, \mathcal{T}, \beta, \lambda, k - 1, \theta)$ 
16    if  $(x, y) \in M_{a,b}$  then  $t_1 := t_1 + t$  else  $t_2 := t_2 + t$ 
17   $w_1 := \lambda / (|I_a| + |I_b| - |M_{a,b}|)$ ,  $w_2 := (1 - \lambda) / (|I_a| \times |I_b| - |M_{a,b}|)$ 
18   $s := \beta \times (w_1 \times t_1 + w_2 \times t_2) + (1 - \beta)$ 
19  if  $|O_a| \times |O_b| \geq \theta$  then  $\mathcal{T}.\text{add}((a, b, k), s)$ 
20  return  $s$ 
    
```


Single-Source Algorithm The single-source RoleSim* algorithm, referred to as SSRS*, is shown in Algorithm 2. It works as follows. First, it starts by building a hash table \mathcal{T} (line 1). Next, it invokes a Single-Pair function to evaluate the RoleSim* similarity between each node $u \in G$ and query q according to whether the similarity value of pair (u, q) is memoised in hash table \mathcal{T} (lines 2-6). If pair (u, q) is at the last level or has no in-neighbours, the similarity is set to $(1 - \beta)$ (line 8). Otherwise, it enumerates all the in-neighboring pairs (a, b) in $I_u \times I_q$ (line 11). If (a, b) exists in hash table \mathcal{T} , it retrieves the similarity of (a, b) from \mathcal{T} directly with no need for recomputation (line 13); otherwise, it recursively computes the similarity of (a, b) (line 15). Using all the similarities of the in-neighboring pairs of (u, q) , it then computes similarity $s(u, q)$ according to (4) (line 16-18), and memoises the resulting score if (u, q) is an important pair (line 19). Finally, it returns $s(u, q)$ to the main function (line 20).

Computational Complexity Analysing the computational time for retrieving single-source RoleSim* query, we show the following theorem:

Theorem 7 *Let N be the number of pairs whose RoleSim* similarity scores are retrieved from the hash table in the traversal of the recursion tree with K levels. Assume that the network G is scale-free and follows power-law degree distribution. We denote by $p_{in}(d)$ and $p_{out}(d)$ the fraction of nodes in G having in-degree and out-degree d , respectively, which satisfy $p_{in}(d) \propto d^{-\gamma_{in}}$ and $p_{out}(d) \propto d^{-\gamma_{out}}$, where γ_{in} and γ_{out} are the power-law exponents whose values are typically $2 \sim 3$. Let d_{in} and d_{out} be the maximum in-degree and out-degree of G , respectively. Then, the average computational time for retrieving RoleSim* similarities between all nodes and a query for K levels is bounded by $O(\rho_{out}^{2(K-1)} (|V|\rho_{out}^2 - \frac{N}{K}))$ with $\rho_{out} := O\left(\frac{(d_{out})^{2-\gamma_{out}} - 1}{2-\gamma_{out}}\right)$.*

Proof We first analyse the computational cost for a single-pair RoleSim* query without using any memoisation optimisation. Since the graph G follows power-law degree distribution, the expected value of the number of in-neighbours of each node is bounded by

$$\begin{aligned} \sum_{d=1}^{d_{in}} d \cdot p_{in}(d) &= \sum_{d=1}^{d_{in}} d \cdot C_{in} \cdot d^{-\gamma_{in}} = C_{in} \cdot \sum_{d=1}^{d_{in}} d^{1-\gamma_{in}} \leq C_{in} \int_1^{d_{in}} x^{1-\gamma_{in}} dx \\ &= \frac{C_{in}}{2-\gamma_{in}} \cdot x^{2-\gamma_{in}} \Big|_1^{d_{in}} = \rho_{in} \quad \text{with} \quad \rho_{in} := \frac{C_{in}}{2-\gamma_{in}} (d_{in}^{2-\gamma_{in}} - 1) \end{aligned}$$

where C_{in} is a constant. Similarly, the expected value of the number of out-neighbouring pairs of any pair is bounded by $\rho_{out} := \frac{C_{out}}{2-\gamma_{out}} ((d_{out})^{2-\gamma_{out}} - 1)$, where C_{out} is a constant. We notice that, to compute each pair of similarity at any level i , the expected value of the number of in-neighbouring pairs that we need to retrieve at level $(i - 1)$ is $O(\rho_{in}^2)$. Since the expected value of the number of node-pairs at level i is bounded by $O(\rho_{out}^{2(i-1)})$ in the average case, the total computational time for evaluating any single-pair similarity at the top level in the average case is bounded by

$$O\left(\sum_{i=1}^K \rho_{out}^{2(i-1)} \cdot \rho_{in}^2\right) = O\left(\frac{\rho_{in}^2}{\rho_{out}^2 - 1} \cdot (\rho_{out}^{2K} - 1)\right) = O(\rho_{out}^{2K})$$

which implies that $O(|V|\rho_{out}^{2K})$ time is required for single-source retrieval of $|V|$ nodes w.r.t. a query.

However, after using memoisation, this computational cost is significantly reduced. Generally, the amount of computational cost reduction depends on the number of memoised pairs and the position at which the memoised pairs appear. For ease of our analysis, we denote by $C(i)$ the computational cost that can be saved by a pair at level i whose similarity value is obtainable directly from the hash table. Since the expected value of the number of out-neighboring pairs of similarities that need to be retrieved at level $(i + 1)$ is bounded by $O(\rho_{\text{out}}^2)$ recursively till level K , the total computational cost of $C(i)$ in the worst case is:

$$C(i) = 1 + \rho_{\text{out}}^2 + \rho_{\text{out}}^4 + \dots + \rho_{\text{out}}^{2(K-i)} = \frac{\rho_{\text{out}}^{2(K-i+1)} - 1}{\rho_{\text{out}}^2 - 1}$$

Then, the average computational cost, denoted as \bar{C} , which can be saved by a pair through retrieving its similarity score from the hash table, is as follows:

$$\begin{aligned} \bar{C} &= \frac{1}{K} \sum_{i=1}^K i \times C(i) = \frac{1}{K} \left(\sum_{i=1}^K i \times \left(\frac{\rho_{\text{out}}^{2(K-i+1)} - 1}{\rho_{\text{out}}^2 - 1} \right) \right) \\ &= \frac{1}{K} \left(\frac{1}{\rho_{\text{out}}^2 - 1} \left(\sum_{i=1}^K i \times (\rho_{\text{out}}^2)^{(K-i+1)} - \sum_{i=1}^K i \right) \right) \\ &= \frac{1}{K} \left(\frac{1}{\rho_{\text{out}}^2 - 1} \left(\frac{\rho_{\text{out}}^4 (\rho_{\text{out}}^{2K} - 1)}{(\rho_{\text{out}}^2 - 1)^2} - \frac{K \rho_{\text{out}}^2}{(\rho_{\text{out}}^2 - 1)} - \frac{(1+K)K}{2} \right) \right) = O \left(\frac{\rho_{\text{out}}^{2(K-1)}}{K} \right). \end{aligned}$$

It follows that the average computational cost for a single-source query is

$$\begin{aligned} O(|V| \rho_{\text{out}}^{2K}) - N \times \bar{C} &= O(|V| \rho_{\text{out}}^{2K} - N \times \frac{\rho_{\text{out}}^{2(K-1)}}{K}) \\ &= O(\rho_{\text{out}}^{2(K-1)} \left(|V| \rho_{\text{out}}^2 - \frac{N}{K} \right)) \quad \text{with} \quad \rho_{\text{out}} := O \left(\frac{(d_{\text{out}})^{2-\gamma_{\text{out}}} - 1}{2 - \gamma_{\text{out}}} \right) \end{aligned}$$

□

6 “Sum-transitivity” of RoleSim* similarity

In this section, we investigate the transitive property of the proposed RoleSim* similarity measure. Intuitively, when a similarity measure $s(*, *)$ fulfils the transitive property, it means that, for any three nodes a, b, c in the graph, if a is similar to b and b is similar to c , it implies that a is likely to be similar to c . The transitivity feature is useful in many real applications, e.g., for predicting and recommending links in a graph.

Before showing the transitive property of RoleSim*, let us induce a distance $d(a, b) := 1 - s(a, b)$ from the RoleSim* measure. Due to $s(*, *) \in [1 - \beta, 1]$, the distance $d(*, *)$ is between 0 and β . In what follows, we will show that $d(*, *)$ satisfies the triangular inequality, which is an indication of $s(*, *)$ transitivity.

We first provide the following two lemmas, which will lay the foundation for our proof of the RoleSim* triangular inequality.

Lemma 1 *Let $s_k(*, *)$ be the k -th iterative RoleSim* similarity via (3) and (4). For any 3 nodes (a, b, c) in a graph, if $s_k(a, b) + s_k(b, c) - s_k(a, c) \leq 1$ holds at iteration k , the following inequality holds:*

$$\frac{\sum_{(x,y) \in M_{a,b}} s_k(x, y)}{|I_a| + |I_b| - |M_{a,b}|} + \frac{\sum_{(y,z) \in M_{b,c}} s_k(y, z)}{|I_b| + |I_c| - |M_{b,c}|} - \frac{\sum_{(x,z) \in M_{a,c}} s_k(x, z)}{|I_a| + |I_c| - |M_{a,c}|} \leq 1 \tag{12}$$

Proof Without loss of generality, we only consider the case of $|I_a| \leq |I_b| \leq |I_c|$. The proofs for other cases are similar, and omitted here due to space limitation. In this case, we have

$$|I_a| + |I_b| - |M_{a,b}| = \max\{|I_a|, |I_b|\} = |I_b|.$$

Hence, the left-hand side (LHS) of (12) can be rewritten as

$$\begin{aligned} & \text{LHS of (12)} \\ &= \frac{1}{|I_b|} \sum_{(x,y) \in M_{a,b}} s_k(x, y) + \frac{1}{|I_c|} \sum_{(y,z) \in M_{b,c}} s_k(y, z) - \frac{1}{|I_c|} \sum_{(x,z) \in M_{a,c}} s_k(x, z) \\ &= \underbrace{\left(\frac{1}{|I_b|} - \frac{1}{|I_c|} \right) \sum_{(x,y) \in M_{a,b}} s_k(x, y)}_{\text{Part 1}} \\ & \quad + \frac{1}{|I_c|} \underbrace{\left(\sum_{(x,y) \in M_{a,b}} s_k(x, y) + \sum_{(y,z) \in M_{b,c}} s_k(y, z) - \sum_{(x,z) \in M_{a,c}} s_k(x, z) \right)}_{\text{Part 2}} \end{aligned} \tag{13}$$

We first find an upper bound on Part 1. Since $\sum_{(x,y) \in M_{a,b}} s_k(x, y) \leq \sum_{(x,y) \in M_{a,b}} 1 = |M_{a,b}|$, it follows that

$$\text{Part 1} \leq \left(\frac{1}{|I_b|} - \frac{1}{|I_c|} \right) \times |M_{a,b}| = \left(\frac{1}{|I_b|} - \frac{1}{|I_c|} \right) \times |I_a| \tag{14}$$

To get an upper bound for Part 2, let

$$\begin{aligned} \tilde{I}_b &= \{y \mid \forall x \in I_a, \exists y \in I_b, \text{ s.t. } (x, y) \in M_{a,b}\} \\ \tilde{M}_{a,c} &= \{(x, z) \mid \exists y \in I_b, \text{ s.t. } (x, y) \in M_{a,b} \wedge (y, z) \in M_{b,c}\} \end{aligned}$$

Then, $M_{b,c}$ can be partitioned into two parts: $M_{b,c} = M_{b,c}^{(1)} \cup M_{b,c}^{(2)}$ where

$$\begin{aligned} M_{b,c}^{(1)} &= \{(y, z) \in M_{b,c} \mid y \in \tilde{I}_b, z \in I_c\} \\ M_{b,c}^{(2)} &= \{(y, z) \in M_{b,c} \mid y \in I_b - \tilde{I}_b, z \in I_c\} \end{aligned}$$

Therefore,

$$\begin{aligned}
 \text{Part 2} &= \sum_{(x,y) \in M_{a,b}} s_k(x, y) + \sum_{(y,z) \in M_{b,c}} s_k(y, z) - \sum_{(x,z) \in M_{a,c}} s_k(x, z) \\
 &= \left(\sum_{(x,y) \in M_{a,b}} s_k(x, y) + \sum_{(y,z) \in M_{b,c}^{(1)}} s_k(y, z) \right) + \sum_{(y,z) \in M_{b,c}^{(2)}} \underbrace{s_k(y, z)}_{\leq 1} - \sum_{(x,z) \in M_{a,c}} s_k(x, z) \\
 &\leq \underbrace{\left(\sum_{(x,z) \in M_{a,c}} s_k(x, z) + |I_a| \right)}_{\leq \sum_{(x,z) \in M_{a,c}} s_k(x, z)} + \left(\underbrace{|M_{b,c}|}_{=|I_b|} - \underbrace{|\tilde{I}_b|}_{=|I_a|} \right) - \sum_{(x,z) \in M_{a,c}} s_k(x, z) \leq |I_b| \tag{15}
 \end{aligned}$$

Substituting (14) and (15) into (13) produces

$$\text{LHS of (12)} \leq \left(\frac{1}{|I_b|} - \frac{1}{|I_c|} \right) |I_a| + \frac{|I_b|}{|I_c|} = \frac{|I_a|}{|I_b|} + \frac{|I_b| - |I_a|}{|I_c|} \leq 1$$

□

Lemma 2 For any 3 nodes (a, b, c) in a graph, if $s_k(a, b) + s_k(b, c) - s_k(a, c) \leq 1$ holds at iteration k, then it follows that

$$\frac{\sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} s_k(x, y)}{|I_a| \times |I_b| - |M_{a,b}|} + \frac{\sum_{(y,z) \in (I_b \times I_c) - M_{b,c}} s_k(y, z)}{|I_b| \times |I_c| - |M_{b,c}|} - \frac{\sum_{(x,z) \in (I_a \times I_c) - M_{a,c}} s_k(x, z)}{|I_a| \times |I_c| - |M_{a,c}|} \leq 1 \tag{16}$$

Proof For each $x \in I_a$, there exist $y_x \in I_b$ and $z_x \in I_c$ such that $(x, y_x) \in M_{a,b}$ and $(x, z_x) \in M_{a,c}$. Then, for each $z \in I_c - \{z_x\}$, there exists $y \in I_b$ such that

$$s_k(x, y) + s_k(y, z) - s_k(x, z) \leq 1$$

Summing both sides of the inequality over all $z \in I_c - \{z_x\}$ and all $y \in I_b$ yields

$$\underbrace{\sum_{y \in I_b} \sum_{z \in I_c - \{z_x\}} s_k(x, y)}_{\text{Part 1}} + \underbrace{\sum_{y \in I_b} \sum_{z \in I_c - \{z_x\}} s_k(y, z)}_{\text{Part 2}} - \underbrace{\sum_{y \in I_b} \sum_{z \in I_c - \{z_x\}} s_k(x, z)}_{=|I_b| \times \sum_{z \in I_c - \{z_x\}} s_k(x, z)} \leq (|I_c| - 1) \times |I_b|$$

where

$$\text{Part 1} = (|I_c| - 1) \times \sum_{y \in I_b} s_k(x, y) \geq (|I_c| - 1) \times \sum_{y \in I_b - \{y_x\}} s_k(x, y)$$

$$\begin{aligned}
 \text{Part 2} &= \sum_{(y,z) \in (I_b \times I_c)} s_k(y, z) - \sum_{y \in I_b} s_k(y, z_x) \geq \sum_{(y,z) \in (I_b \times I_c) - M_{b,c}} s_k(y, z) \\
 &\leq \sum_{(y,z) \in M_{b,c}} s_k(y, z)
 \end{aligned}$$

Therefore, it follows that

$$(|I_c| - 1) \times \sum_{y \in I_b - \{y_x\}} s_k(x, y) + \sum_{(y,z) \in (I_b \times I_c) - M_{b,c}} s_k(y, z) - |I_b| \times \sum_{z \in I_c - \{z_x\}} s_k(x, z) \leq (|I_c| - 1) \times |I_b|$$

Summing both sides of the inequality over all $x \in I_a$ produces

$$\begin{aligned}
 &= \sum_{(x,y) \in (I_a \times I_b) - M_{a,b}} s_k(x,y) \\
 (|I_c| - 1) \times &\underbrace{\sum_{x \in I_a} \sum_{y \in I_b - \{y_x\}} s_k(x,y)} + |I_a| \times \sum_{(y,z) \in (I_b \times I_c) - M_{b,c}} s_k(y,z) \\
 - |I_b| \times &\underbrace{\sum_{x \in I_a} \sum_{z \in I_c - \{z_x\}} s_k(x,z)} \leq |I_a| \times (|I_c| - 1) \times |I_b| \\
 &= \sum_{(x,z) \in (I_a \times I_c) - M_{a,c}} s_k(x,z)
 \end{aligned}$$

Since $\bigcup_{x \in I_a} \{(x, y_x)\} = M_{a,b}$ and $\bigcup_{x \in I_a} \{(x, z_x)\} = M_{a,c}$, we divide both sides of the inequality by $(|I_a| \times (|I_c| - 1) \times |I_b|)$ to get LHS of(16) ≤ 1 . □

Example 6 Recall the graph G in Figure 1, and three node-pairs $(1, 2), (2, 3), (3, 1)$ in G . For each pair (e.g., $(1, 2)$), all the RoleSim* similarities of its in-neighboring pairs are tabularised as a grid (e.g., $I_1 \times I_2$) in Figure 8, respectively. The green cells in each grid (e.g., $I_1 \times I_2$) correspond to the similarities over the maximum bipartite matching (e.g., $M_{1,2}$); and the remaining cells in orange denote the similarities out of the bipartite matching (i.e., in $I_1 \times I_2 - M_{1,2}$). Lemma 1 indicates that the similarity values in the green cells satisfy

$$\begin{aligned}
 &\frac{\sum_{(x,y) \in M_{1,2}} s_k(x,y)}{|I_1| + |I_2| - |M_{1,2}|} + \frac{\sum_{(y,z) \in M_{2,3}} s_k(y,z)}{|I_2| + |I_3| - |M_{2,3}|} - \frac{\sum_{(x,z) \in M_{1,3}} s_k(x,z)}{|I_1| + |I_3| - |M_{1,3}|} \\
 = &\frac{0.430 + 0.328}{2 + 3 - 2} + \frac{0.430 + 0.328 + 0.347}{3 + 3 - 3} - \frac{0.430 + 0.328}{2 + 3 - 2} = 0.137 \leq 1
 \end{aligned}$$

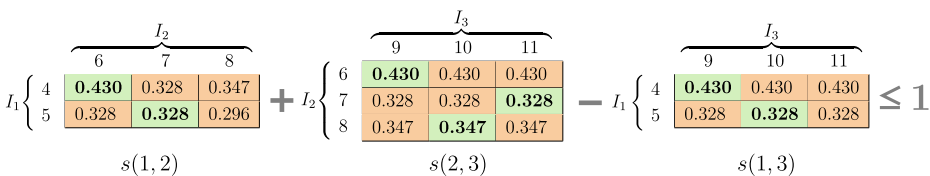


Figure 8 An illustrative example of Lemmas 1 and 2. The similarity grids for the in-neighbouring pairs of three node-pairs $(1, 2), (1, 3), (2, 3)$ in Figure 1 are visualised, respectively, to picturise RoleSim* triangular inequality

Similarly, Lemma 2 implies that the similarity values in the orange cells satisfy

$$\begin{aligned}
 & \frac{\sum_{(x,y) \in (I_1 \times I_2) - M_{1,2}} s_k(x,y)}{|I_1| \times |I_2| - |M_{1,2}|} + \frac{\sum_{(y,z) \in (I_2 \times I_3) - M_{2,3}} s_k(y,z)}{|I_2| \times |I_3| - |M_{2,3}|} - \frac{\sum_{(x,z) \in (I_1 \times I_3) - M_{1,3}} s_k(x,z)}{|I_1| \times |I_3| - |M_{1,3}|} \\
 &= \underbrace{\frac{0.328 + 0.347 + 0.328 + 0.296}{2 \times 3 - 2}}_{\textcircled{1}} - \underbrace{\frac{0.430 + 0.430 + 0.328 + 0.328}{2 \times 3 - 2}}_{\textcircled{3}} \\
 &+ \underbrace{\frac{0.430 + 0.430 + 0.328 + 0.328 + 0.347 + 0.347}{3 \times 3 - 3}}_{\textcircled{2}} = 0.335 \leq 1
 \end{aligned}$$

Leveraging Lemmas 1 and 2, we are now ready to show the sum-transitivity of the RoleSim* similarity distance, which is the main result in this subsection:

Theorem 8 (RoleSim* Triangle Inequality) *We denote by $d(a, b) := 1 - s(a, b)$ the closeness between nodes a and b . Then, for any three nodes a, b, c in a graph, the following triangle inequality holds, i.e.,*

$$d(a, b) + d(b, c) \geq d(a, c) \tag{17}$$

Proof By the definition of $d(a, b) := 1 - s(a, b)$, based on the fact that

$$\begin{aligned}
 d(a, b) + d(b, c) \geq d(a, c) &\iff 1 - s(a, b) + 1 - s(b, c) \geq 1 - s(a, c) \\
 &\iff s(a, b) + s(b, c) - s(a, c) \leq 1 \tag{18}
 \end{aligned}$$

in what follows we will prove (18) holds by induction on k . For $k = 0$, by virtue of (3), it is apparent that

$$s_0(a, b) + s_0(b, c) - s_0(a, c) = 1 + 1 - 1 = 1 \leq 1.$$

For $k > 0$, we assume that $s_k(a, b) + s_k(b, c) - s_k(a, c) \leq 1$ holds, and will prove that $s_{k+1}(a, b) + s_{k+1}(b, c) - s_{k+1}(a, c) \leq 1$ holds.

Let P_1 and P_2 be left-hand side (LHS) of (12) and (16), respectively. According to Lemmas 1 and 2, it follows from $P_1 \leq 1$ and $P_2 \leq 1$ that

$$\begin{aligned}
 s_{k+1}(a, b) + s_{k+1}(b, c) - s_{k+1}(a, c) &= \beta(\lambda P_1 + (1 - \lambda)P_2) + (1 - \beta) \\
 &\leq \beta(\lambda + (1 - \lambda)) + (1 - \beta) \leq 1
 \end{aligned}$$

□

7 Scaling RoleSim* search using triangle inequality and partitioning

RoleSim* based similarity search can be also scaled via pruning using the triangular inequality property. With some pre-computation, one can prune the node-pairs that need not be evaluated and eliminate nodes from the candidate list to produce a top- k similar nodes rank list with less computation. We first discuss a strategy for retrieving approximate node-pair results based on graph partitioning. We then discuss exact single-source computation to obtain the most similar nodes to a query q , by indexing based on the distance to some chosen keys.

First, we consider a simple graph partitioning-based strategy, where graph G is partitioned using a vertex separator method to produce parts of roughly equal sizes. We refer to the set of vertex separators (also called vertex cut or separating set) as V_S . Nodes in V_S are the nodes that, when removed from G , separate it into its partitions. For our purposes, we include the set of nodes V_S with their corresponding edge connections into each of the partitions. That is, nodes in the V_S are the only nodes that are present in every subgraph constructed via this partitioning approach.

Example 7 Consider the digraph G in Figure 9a, where a 2-way partitioning has been performed resulting in vertex separators $V_S = \{3, 7\}$. Subgraphs G_1 and G_2 are constructed such that $V_{G_1} = \{1, 2, 3, 7\}$ and $V_{G_2} = \{3, 4, 5, 6, 7\}$. Using our single-source approach, all pairs of similarities from nodes in V_S to nodes in G can be pre-computed. Given a decay factor $\beta = 0.8$, and relative weight $\lambda = 0.7$, the exact RoleSim* similarities are computed (after k iterations) and cached for all $s(v, *)$ where $v \in V_S$. Now, for node-pair $(2, 1)$, the exact similarity score of $s(2, 1)$ from G_1 may be approximately computed as $s_P(2, 1)$ where the pre-computed value of $s(3, 7)$ is used in every iteration and only the pruned graph G_1 is considered. The similarity score $s_{PA}(2, 1)$ also uses the pre-computed $s(3, 7)$, but also allows access to neighboring nodes if they are present in G_2 . As seen from Figure 9b and c, $s_{PA}(2, 1)$ is more accurate but less efficient due to accessing a larger graph.

Consider a query node q . To compute its similarity to any node n in G , there are three different cases:

1. One or both nodes are vertex separators, i.e., $q \in V_S$ or $n \in V_S$: this means an exact value for $s(q, n)$ is already pre-computed
2. Both nodes are in the same partition, say $q, n \in G_1$: this means an approximate value for $s(q, n)$ can be computed considering G_1 as the input graph and discarding all nodes/edges from G_2
3. Both nodes are in different partitions, say $q \in G_1$ and $n \in G_2$: this means an approximate (lower bound) value for $s(q, n)$ can be directly computed from the pre-computed exact values of $s(v, *)$ by making use of the triangle inequality property of RoleSim*

While case 1 returns the exact value, case 2 and case 3 lead to approximate results. This is depicted in Figure 9d, where numerous additional similarity computations (Figure 9e) are avoided by retrieving the values stored for nodes in V_S and using these to compute approximate results. In case 2 when G is pruned into G_1 , the nodes/edges connections that are discarded no longer contribute to the similarity computation during the RoleSim* traversals, leading to inexact results. The case 3, considering $q \in G_1$ and $n \in G_2$ and pre-computed exact values of $s(v, *)$, is explained in further detail:

Using triangle inequality (18), we know that $\forall v \in V_S$:

$$s(q, n) \geq s(q, v) + s(v, n) - 1$$

Hence, using the pre-computed values $s(v, *)$, we know that $s(q, n)$ must have its lower bound as the largest of these values, or:

$$s(q, n) \geq \max_{\forall v \in V_S} (s(q, v) + s(v, n) - 1)$$

This can be extended to k -way partitioning. The vertex separator set is taken as one end node of each of the edge cuts in any k -way partitioning scheme, and included into each of the subgraphs.

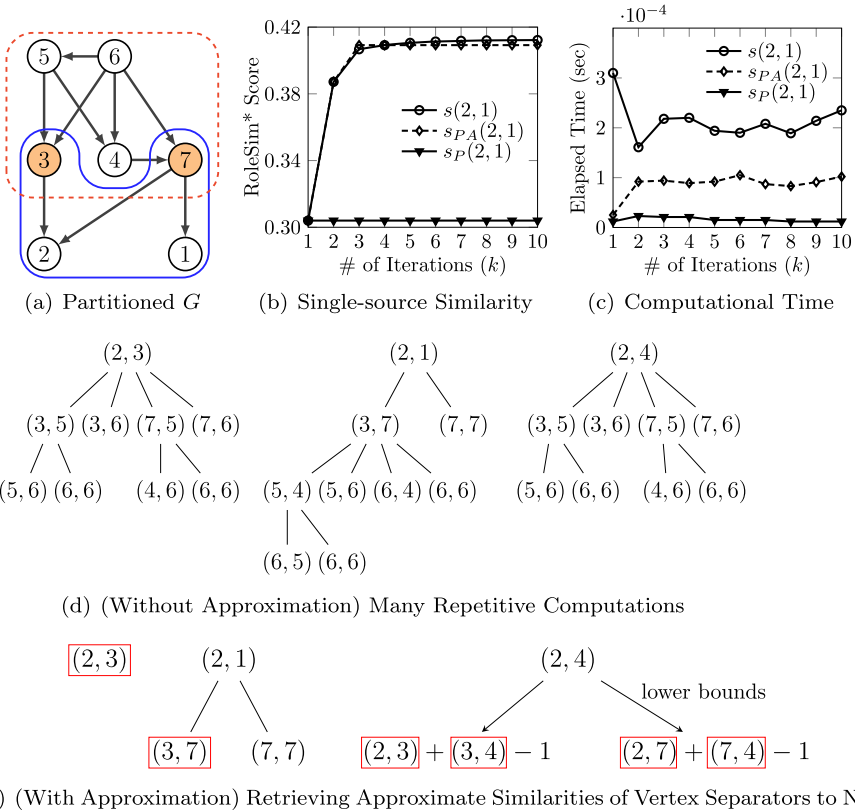


Figure 9 Approximate RoleSim* node-pair similarity retrieval that caches the similarities between vertex separators to all nodes and uses triangle inequality to compute approximate results for node-pairs (2, 3) (case 1), (2, 1) (case 2), (2, 4) (case 3), where two partitions are encircled in red and blue respectively

Next, we discuss the challenge of returning an exact solution for the most similar nodes to a query q . The vertex separator set V_S is taken as a set of keys whose similarities to all nodes are pre-computed. From the triangle inequality, we know that:

$$d(q, n) \geq |d(v, n) - d(q, v)| \tag{19}$$

Hence, a lower bound may be obtained on $d(q, n)$, by computing:

$$d(q, n) \geq \max_{v \in V_S} |d(v, n) - d(q, v)| \tag{20}$$

Given a threshold s_{\min} , we want to find all nodes n_i such that $s(q, n_i) \geq s_{\min}$. That is, $d(q, n_i) \leq \xi$ where $\xi = 1 - s_{\min}$. The lower bounds $d(q, n_i)$ for all nodes n_i are given by (20). Any lower bound greater than ξ allows the node n_i to be pruned from the candidate list. We denote the resulting candidate set as V_k . In large graphs, with careful partitioning to select a small number of vertex separators ($|V_k| \ll |V|$), the number of distance computations is substantially reduced.

One can use this approach to prune partitions while identifying the top- k similar nodes to query q . Consider subgraphs G_j obtained by partitioning G , each containing a single vertex separator v . Using pre-computed values for $d(v, *)$, (19) gives the lower bounds of $d(q, n_i)$

	Datasets	(Abbr.)	V	E	E / V	Type
small	DBLP	(DBLP)	2,372	7,106	2.99	Undirected
	Amazon	(AMZ)	5,086	8,970	1.76	Directed
	HEP-Citation	(CIT)	34,546	421,578	12.20	Directed
medium	P2P-Gnutella	(P2P)	62,586	147,892	2.36	Directed
	Email-EuAll	(EML)	265,214	420,045	1.58	Directed
	Web-Google	(WEB)	875,713	5,105,039	5.82	Directed
large	YouTube	(YOU)	1,134,890	2,987,624	2.63	Undirected
	LiveJournal	(LJ)	4,847,571	68,993,773	14.23	Directed

for $n_i \in V_{G_j}$ ($j = 1, \dots, N_P$ respectively for each of the N_P partitions). The minimum of all these distances within a partition gives a lower bound value that helps to index the partitions:

$$d(q, n_i) \geq \min_{n_i \in V_{G_j}} |d(v, n_i) - d(q, v)|$$

Let us denote these lower bounds as ξ_{G_j} for each subgraph G_j . Without loss of generality, suppose $\xi_{G_1} > \xi_{G_2}$ for a 2-way partitioning of G . Thus, every node in G_1 is necessarily at least ξ_{G_1} distance away from q . We first compute exact distances for all nodes in G_2 . If k such nodes all have a distance to q that is smaller than ξ_{G_1} , then nodes of G_1 need not be considered, and the resulting top- k can be directly returned. If not, that is if any nodes of G_2 are at a distance higher than the lower bound of the next partition (here, G_1), then nodes of G_1 must be processed. These nodes are then inserted into the top- k ranking based on the computed distances. A similar process continues through the ordered set of ξ_{G_j} values for multi-way partitioned data.

8 Experimental evaluation

8.1 Experimental settings

Datasets. We use 8 real datasets with different scales, as illustrated below:

- **DBLP.** A collaboration (undirected) graph taken from DBLP bibliography.⁴ We extract a co-authorship subgraph from six top conferences in computer science (SIGMOD, VLDB, PODS, KDD, SIGIR, ICDE) during 2018–2020. If two authors (nodes) co-authored a paper, there is an edge between them.
- **Amazon.** A co-purchasing graph crawled from *Customers Who Bought This Item Also Bought* feature of Amazon⁵. Each node is a product, and edge $i \rightarrow j$ means that product j appears in the frequent co-purchasing list of i .
- **HEP-Citation.** A citation digraph from arXiv scholarly physics articles. In this graph, nodes represent papers, and there is a directed edge from paper u to paper v if paper u cites paper v .

⁴www.informatik.uni-trier.de/~ley/db/

⁵www.amazon.co.uk

Models	(Abbr.)	Description
RoleSim*	(RS*)	our proposed RoleSim* model in Algorithm 1.
Single-Src RS*	(SSRS*)	our proposed single-source RoleSim* model in Algorithm 2.
SimRank	(SR)	a pairwise similarity model proposed by Jeh and Widom [7].
MatchSim	(MS)	a model relying on the matched neighbors of node-pairs [17].
RoleSim	(RS)	a model that ensures the automorphic equivalence of nodes [9].
RoleSim++	(RS++)	an enhanced RoleSim that considers in- and out-neighbors [25].
CentSim	(CS)	a model that compares the centrality of node-pairs [14].

- P2P-Gnutella. A file sharing graph from Gnutella peer-to-peer network. In this graph, nodes represent hosts, and each edge denotes the connection from one host to another in the Gnutella network.
- Email-EuAll. A digraph constructed from emails of a research institute. Each node represents an email address, and there is a link from node u to v if at least one email is sent from u to v .
- Web-Google. A Google web digraph from SNAP⁶ network repository. In this digraph, nodes represent web pages, which are connected by directed edges that represent the hyperlinks from one web page to another.
- YouTube. A friendship (undirected) graph from YouTube video sharing website, which is an online social network. In this digraph, nodes denote users, and edges are the friendship relation between them.
- LiveJournal. A large friendship graph from LiveJournal community. This is an online social network, in which nodes are users, and each edge $i \rightarrow j$ is a recommendation of user j from user i .

All experiments are conducted on a PC with Intel Core i7-10510U 2.30GHz CPU and 16GB RAM, using Windows 10. Each experiment is repeated 5 times and the average is reported.

Compared Algorithms We implement the following algorithms in VC++:

Parameters We use the following parameters as default: (a) damping factor $\beta = 0.8$, (b) relative weight $\lambda = 0.7$, (c) total number of iterations $K = 5$.

Unsupervised Semantic Evaluation We design an unsupervised evaluation setting to quantify the effectiveness of the similarity measures. We use self-similarity as the ground truth and study the effect of sampling the immediate neighborhood of a query node on similarity scores in RoleSim* compared with SimRank and RoleSim.

Our evaluation is inspired by the problem of determining duplicate nodes in a network simply by examining their neighborhoods for similar patterns. In many applications, the underlying network contains duplicate entities with noisy, incomplete, and partially overlapping information, such as in a social network that has been scraped from multiple sources. Similarity of duplicate nodes is expected to be high. We consider duplicate entities as separate nodes, where each duplicate has some sampling of the total set of neighboring edges

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

available to the node. For example, in a co-purchasing product graph (AMZ), duplicates may exist when merging multiple e-commerce sources, or when identical products are sold by different sellers. This indicates that each of these duplicate products were frequently purchased along with certain other products as they share some common neighbors. Similarly, incorrectly spelled author names or multiple sources for a paper can lead to duplicates in co-authorship and citation networks (DBLP, CIT).

Consider a single query node q . In our experiments, we create a node q' and add it to the graph. We connect q' to some proportion (η) of the total number of neighbors of q , and hereby refer to q' as the “sampled clone”. The similarity scores of q to all other points in the graph are computed using SimRank, RoleSim, and RoleSim*. We evaluate how much the relative similarities are preserved when different measures are used. First, we vary η for q' with step size 0.25 (and ensuring no orphaned nodes) while varying $\lambda = 0.0, 0.3, 0.5, 0.7, 1.0$ for RoleSim* and compare the resulting similarity scores. In a similar experiment, we vary both η (for q) as well as η' (for q') each with step size 0.25, resulting in some overlap of neighborhoods as the values of η and η' grow towards 1. These results are aggregated over 20 queries on DBLP and AMZ graphs respectively, where query nodes are chosen as having high degree of neighbors.

8.2 Experimental results

Semantic Accuracy We first count the number of queries where the sampled clone q' appears in the top- k ($k = 1, 5, 10$) similar nodes to query q for RoleSim*. Intuitively, this studies how much structural information is gleaned about a query node. Figure 10a presents the number of such queries out of 20 on the undirected DBLP graph, considering top-5 similarity scores. Other top- k plots are omitted, but show that with increasing k for a given sampling proportion there are more such queries even at lower λ .

Next, we test the impact of sampling η and λ on ranking quality in RoleSim*. We plot the average ranking quality (normalized discounted cumulative gain (nDCG)), considering top-100 similar nodes of the sampled clone and comparing this to the baseline original query. We observe that the trend (with respect to η) seen in Figures 10b and 11b for $\lambda = 1$ resembles that for RoleSim, and the trend for $\lambda = 0.5$ is close to that for SimRank.

We further consider a fixed value of $\lambda = 0.7$ and confirm that the RoleSim* has higher ranking quality compared to SimRank and RoleSim, with respect to the average nDCG. Figure 10c with undirected DBLP graph shows that RoleSim* produces a more consistent nDCG even with small η . For the directed AMZ graph in Figure 11c too, RoleSim shows

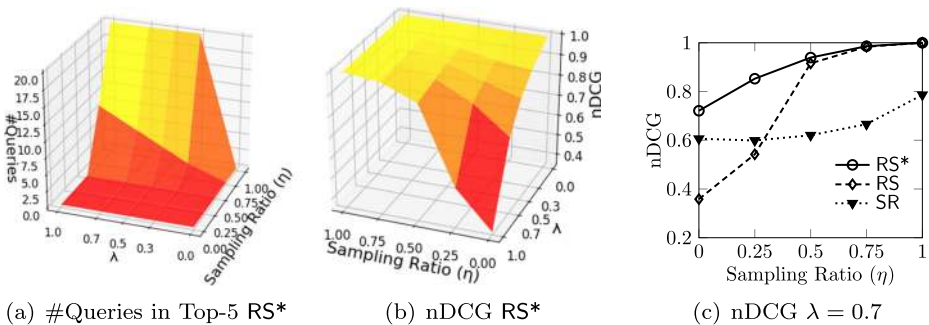


Figure 10 Effect of Sampling Ratio (η) & Weight (λ) on Ranking Quality (DBLP)

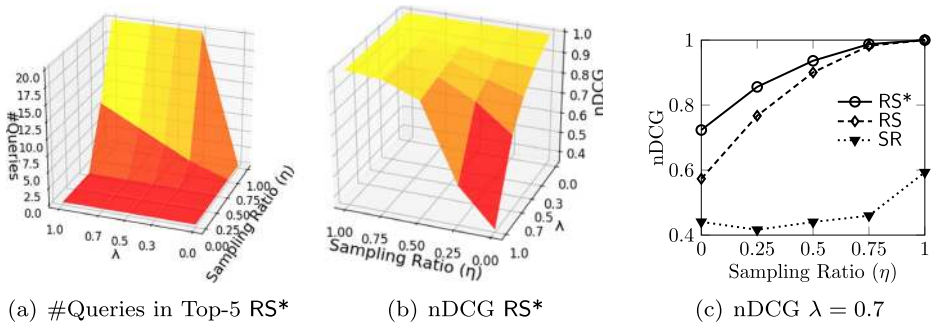


Figure 11 Effect of Sampling Ratio (η) & Weight (λ) on Ranking Quality (AMZ)

significant improvement at lower sampling, and the performance of SimRank is negatively affected throughout, while RoleSim* remains stable.

Finally, we also check the results obtained on varying the sampling of both q and q' together (sampling η and η' neighboring edges respectively). For the resulting top- k similarity lists, we count the number of queries (out of 100) for which clone q' is present in top-10 results for q (plots are omitted here). This provides an estimate for the number of duplicate entities that can be correctly identified. We note that RoleSim and RoleSim* are both heavily impacted when there is a large mismatch in the sampled neighborhood sizes. Specifically, the exact nature of the neighboring nodes themselves appears less important compared to the relative structure of connectivity patterns with the neighborhood. Despite random samples of neighborhoods, the results peak only when the neighborhood sizes are close to each other (*i.e.*, η and η' are equal).

Overall, nDCG scores of RoleSim* are superior to RoleSim, while SimRank performs poorly when sampling rates are low. These results together indicate that for the challenge of identifying duplicate entities, RoleSim* is best suited to correctly identify a match when presented with a noisy sample of edge connections from the duplicate node. In particular, taking only a small sample of edges from both the duplicate and the original nodes produces best matching results.

Table 2 Similarity rankings for “Philip S. Yu” on DBLP co-authorships data

#	RS*($\lambda = 0.6$)	RS*($\lambda = 0.8$)	RS	SR
1	Nitesh V. Chawla	Xia Hu	Xia Hu	Yuan Fang
2	Danai Koutra	Nitesh V. Chawla	Nitesh V. Chawla	Chenwei Zhang
3	Yanjie Fu	Yanjie Fu	Yanjie Fu	Nan Du
4	Jure Leskovec	Jure Leskovec	Huan Liu	Wei Fan
5	Haifeng Chen	Danai Koutra	Jure Leskovec	Lichao Sun
6	Xia Hu	Haifeng Chen	Haifeng Chen	Weiran Huang
7	Xing Xie	Xing Xie	Danai Koutra	Jianxin Ma
8	Xiangnan He	Xiangnan He	Xing Xie	Xinyue Liu
9	Di Niu	Di Niu	Xiangnan He	Binbin Hu
10	Jennifer G. Dy	Huan Liu	Fenglong Ma	Daixin Wang
...
28	Huan Liu	Dawei Yin	Di Wu	Ning Wu
...
89	Xiang Li	Han Zhu	Qinyong Wang	Huan Liu
...
350	Mao Yang	Houdong Hu	Xi (Stephen) Chen	Jure Leskovec

Qualitative Case Study Table 2 compares the similarity ranking results from three algorithms (SR, RS and RS*) for retrieving top-10 most similar authors *w.r.t.* query “Philip S. Yu” on DBLP. From the results, we see that the top rankings of RS* are similar to RS, highlighting its capability to effectively capture automorphic equivalent neighboring information. For instance, “Jure Leskovec” is top-ranked in RS* list. This is reasonable because he and “Philip S. Yu” have similar roles - they are both Professors in Computer Science with close research expertise (*e.g.*, knowledge discovery, recommender systems, commonsense reasoning). However, the rankings of RS* are different from those of RS. For example, “Jure Leskovec” is ranked 350th by SR, but 4th by RS* and RS. This is because SimRank can only capture connected paths between two authors while ignoring their automorphic equivalent structure. “Jure Leskovec” has rare collaborations with “Philip S. Yu”, both direct and indirect, thus leading to a low SimRank score.

To evaluate RS* further, we choose two different values for $\lambda \in \{0.6, 0.8\}$ to show how RS* ranking results are perturbed *w.r.t.* λ . From the results, we notice that, when λ is varied from 0.6 to 0.8, nodes with small SR scores (*e.g.*, “Jure Leskovec”) exhibit a stable position in RS* ranking, whereas nodes having higher SR scores (*e.g.*, “Huan Liu”) have a substantial change. This conforms with our intuition because “Huan Liu”’s collaboration with “Philip S. Yu” is closer than “Jure Leskovec”’s, and RS* is able to capture both connectivity and automorphic equivalence of two authors using a balanced weight λ . Thus, compared with “Jure Leskovec”, “Huan Liu” who has higher SimRank value with “Philip S. Yu” is more sensitive to λ change, as expected.

Computational Time The second set of experiments is evaluating the computational time of seven algorithms (SSRS*, RS*, RS, MS, RS++, CS, SR) on various real-life datasets, including both medium graphs (*e.g.*, CIT, P2P, EML) and large graphs (*e.g.*, WEB, YOU, LJ).

Figure 12 compares the computational time of SSRS* with other competitors (*e.g.*, RS, MS, RS++, CS, SR) for single-source queries $\{s(*, q)\}$. On each dataset, we randomly select 100 nodes as queries, according to their PageRank values, to guarantee the selected queries cover a possible range of the most important and moderately important nodes. We take the average time for computing single-source $\{s(*, q)\}$ over all the queries. From the results, we observe that (1) on all datasets, SSRS* is consistently faster than the other algorithms, highlighting the effectiveness of our caching techniques that eliminates a significant number of unnecessary recomputations in DFS backtracking. In comparison, RS*, RS, MS, and RS++ must store all-pairs similarity values of the last iteration for iteratively computing the scores of the next iterations. (2) On large datasets (*e.g.*, WEB, YOU, LJ), SSRS* and CS scale well, whereas the other algorithms crash due to the explosion of the memory that is required for storing all-pairs similarity information for iterative computations.

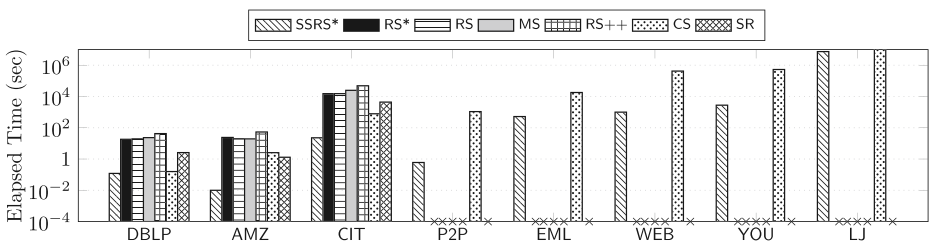


Figure 12 CPU Time Comparison on 8 Real Datasets

In contrast, SSRS* retrieves only a small portion of required pairwise similarity information per level on an as-needed basis. CS is also scalable on sizable graphs since it simply assesses pairwise similarities one by one through aggregating node centrality values, thus leading to low computation time. However, SSRS* is consistently 5–8 times faster than CS due to our unordered hashing techniques for minimising unnecessary recomputations. (3) On small datasets (*e.g.*, DBLP, AMZ, CIT) where all the algorithms survive, RS* has comparable computational time to RS and MS. This implies that RS* achieves high accuracy without sacrificing running speed. In addition, RS*, RS, and MS are faster than RS++. This is because RS++ needs to find two maximum bipartite matchings for both in- and out-neighboring pairs, as opposed to RS* that involves the computation of only one matching. SR is slightly faster than RS*. This is consistent with our analysis as SR simply takes the average of all similarities of the in-neighboring pairs without the need to find the maximum bipartite matching.

Figure 13a and b show the effect of iteration number k and threshold δ on the running time of RS* on DBLP and AMZ, respectively. For each dataset, we vary δ from 0 to 0.05. When $\delta = 0$, it reduces to RS* algorithm. From the results on both datasets, we discern that, for each fixed δ , the running time of threshold-based RS* increases as k grows. When δ becomes larger, the growth rate of RS* time tends to be sublinear. For example, when $\delta = 0.05$ on DBLP, only after $k = 5$ iterations, the increasing time of threshold-based RS* has leveled off. In contrast, when $\delta = 0.01$, the time becomes steady after $k = 8$ iterations. The reason is that a higher setting of threshold δ implies a larger number of pairs to be pruned per iteration, thus leading to the growth rate of the running time decreasing in an earlier stage during iterations.

Figure 14a and b show the influence of different threshold values (δ) and the number of iterations (k) on the computational time of SSRS* on EML and YOU, respectively. We vary δ from 0.01 to 0.1 for each dataset, and k from 3 to 9 and 6, respectively for EML and YOU. From the results, we notice that (1) for any fixed δ , the running time of threshold-based SSRS* increases more mildly than that of SSRS* as k grows. For instance, when $\delta = 0.05$ on EML, the threshold-based SSRS* is 2.9 (*resp.* 3.14) times faster than SSRS* at iteration $k = 7$ (*resp.* $k = 9$). This is because the similarity values decrease with the growing number of iterations. Thus, a larger number of node-pairs with smaller similarity values may appear more often when k is larger, thereby having a higher chance to be pruned, as

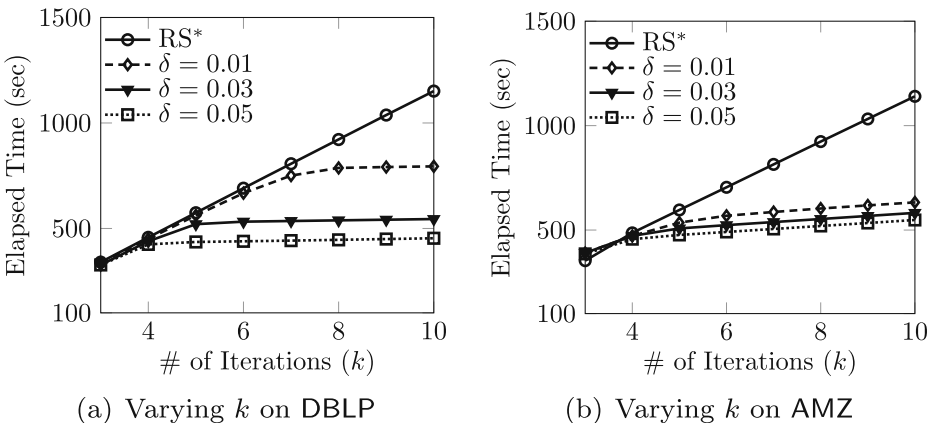


Figure 13 CPU Time Comparison for Different Threshold-Based RS*

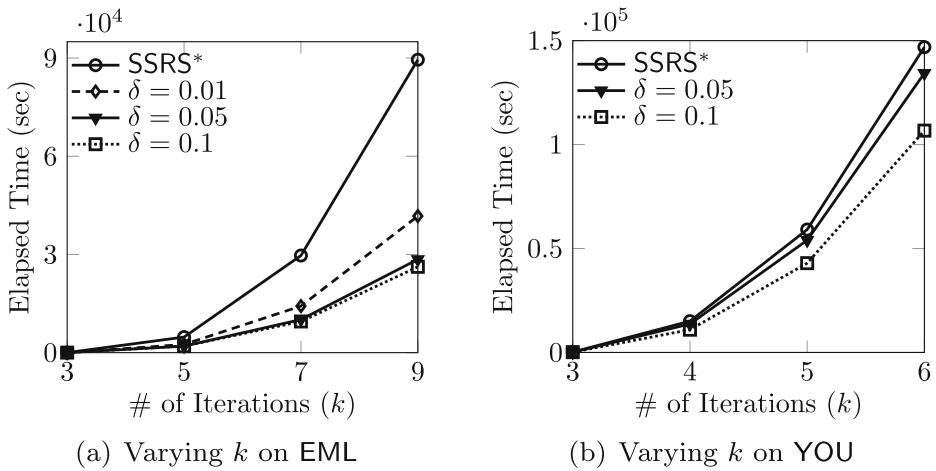


Figure 14 CPU Time Comparison for Different Threshold-Based SSRS*

expected. The similar trend holds on YOU. (2) At each iteration k , increasing the threshold δ will enable a moderate reduction in the running time of the threshold-based SSRS*. For example, for $k = 9$ on EML, when $\delta = 0.05$ (resp. $\delta = 0.1$), the threshold-based SSRS* is 3.14 (resp. 3.41) times faster than SSRS*. The reason is that higher settings of δ will result in a larger number of pairs to be pruned per iteration, which agrees well with our basic intuition.

Memory Usage We next compare the memory usage of SSRS* with other competitors on real datasets. The results are reported in Figure 15. It is discerned that SSRS* and CS are the only algorithms that scale well on all the datasets, including billion-edge graphs (e.g., YOU and LJ), as opposed to other algorithms that crash on any sizable graphs due to memory explosion. In addition, even on small graphs, where all the algorithms survive, the memory required by SSRS* is one order of magnitude smaller than others, except for the CS on DBLP, AMZ, CIT datasets. This is because RS*, RS, MS, RS++ need to memoise the entire similarity matrix of an iteration to compute similarities at the next iteration, thereby leading to quadratic memory. In comparison, SSRS* selects only the “important” pairs for memoization to eliminate unnecessary recomputations in DFS backtracking.

Effect of Threshold δ on RS* Accuracy Figure 16a and b show the influence of threshold δ on RS* accuracy over real datasets (DBLP and AMZ). The accuracy is evaluated using three

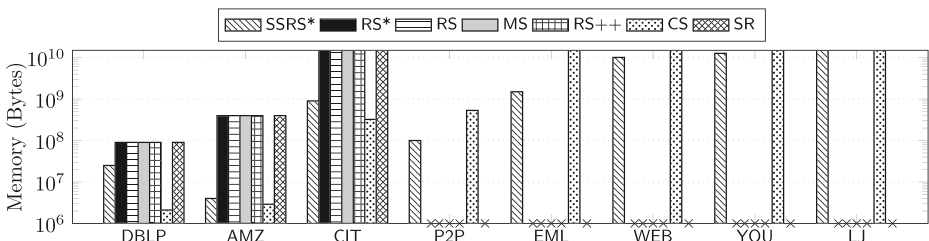


Figure 15 Memory space comparison on real datasets

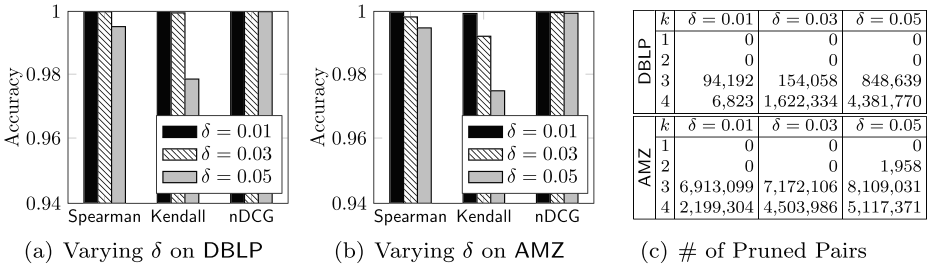


Figure 16 Accuracy comparison for different threshold-based RS*

ranking measures (Spearman, Kendall, nDCG). We randomly sample 100 queries from each dataset, and vary threshold δ from 0.01 to 0.05. For each δ , we compute single-source threshold-based RS* similarities $\{s_k^\delta(*, q)\}$ w.r.t. each query q . Choosing non-threshold based RS* similarities $\{s_k(*, q)\}$ as the baseline, we evaluate the average value of Spearman, Kendall, and nDCG, respectively, for each threshold based RS* over 100 queries on each dataset. We notice that, on each dataset, all the threshold based RS* consistently achieve > 98% accuracy by each ranking measure. For top-100 results on both datasets, the similarity rankings attain > 99% nDCGs on average. These imply that the accuracy compromised by the threshold based RS* is negligibly small for fast speed. Moreover, when δ increases from 0.01 to 0.05, the accuracy decreases slightly for each ranking measure because large threshold may prune a large number of node-pairs per iteration. This agrees well with the pruning table in Figure 16c, where large δ implies more pairs are eliminated at each iteration.

Figure 17a and b illustrate the influence of threshold δ on SSRS* accuracy over EML and YOU datasets. The accuracy evaluated using Spearman, Kendall and nDCG ranking measures on 100 randomly selected queries from each dataset. To compare the effect of threshold value on accuracy we select three $\delta = 0.01$, $\delta = 0.05$ and $\delta = 0.1$ values. For each δ value, we compute SSRS* similarities $\{s_k^\delta(*, q)\}$ w.r.t. each query q by choosing non-threshold-based SSRS* similarities $\{s_k(*, q)\}$ as the baseline. We evaluate the average value of Spearman, Kendall, and nDCG, respectively, for each threshold-based SSRS* over 100 queries and on each dataset. We notice that, (1) on each dataset, threshold-based SSRS* with $\delta = 0.01$ and $\delta = 0.05$ achieve > 99% accuracy by Spearman and Kendall ranking measures, while for $\delta = 0.1$ the ranking accuracy slightly decreases to > 96%, this is predictable due to the high number of pruned pairs that caused by higher δ value, which is consistent with the pruning table in Figure 16c. (2) For top-100 results on both datasets, the similarity ranking results reach > 99% nDCGs on average. These results point out that the

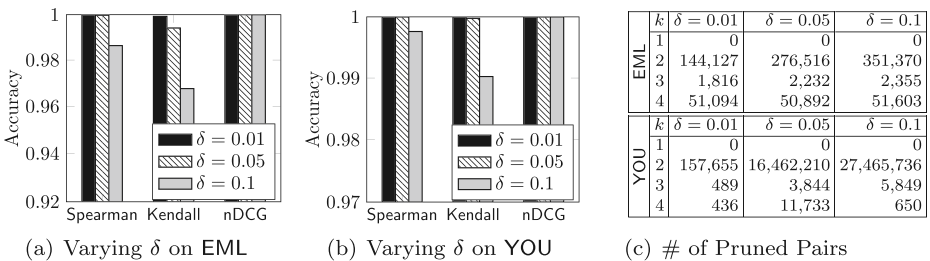


Figure 17 Accuracy comparison for different threshold-based SSRS*

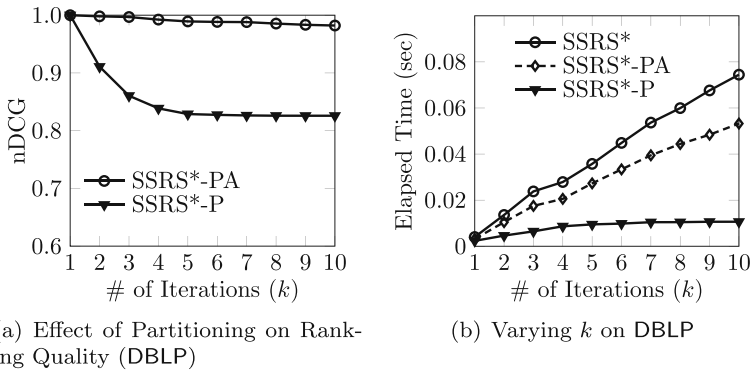


Figure 18 Accuracy and CPU time comparison for partitioned SSRS* (DBLP)

threshold-based SSRS* provides high computation speed with an insignificant decrease in accuracy.

Effect of Partitioning on SSRS* We present an illustrative partitioning scheme (using the triangle inequality property) that may be applied for approximate SSRS* computation. The accuracy (nDCG) and efficiency over real datasets (DBLP and AMZ) are evaluated by randomly sampling 100 queries from each dataset. With varying number of iterations k , we compute single-source RS* similarities $\{s_k(*, q)\}$ w.r.t. each query q . Choosing SSRS* similarities $\{s_k(*, q)\}$ as the baseline, we evaluate the average value of nDCG for both SSRS*-P and SSRS*-PA over 100 queries on each dataset. For both approaches, METIS [11] is used to generate 2-way partitioning of the graph using the vertex separator method. Similarity scores are pre-computed from these vertex separator nodes to all nodes, and these cached values are retrieved during single-source RS* computations. SSRS*-P denotes an approach where only the pruned subgraph is considered for similarity computation, while SSRS*-PA denotes an approach where access to neighboring nodes in other partitions is allowed during the similarity computation. Figures 18a and 19a indicate that, on these datasets, SSRS*-P achieves close to 85% accuracy in terms of nDCG for top-100 results. SSRS*-PA is more accurate, however it incurs much higher computational time due to more edge connections being taken into consideration. As seen from Figures 18b and 19b, a partitioning approach

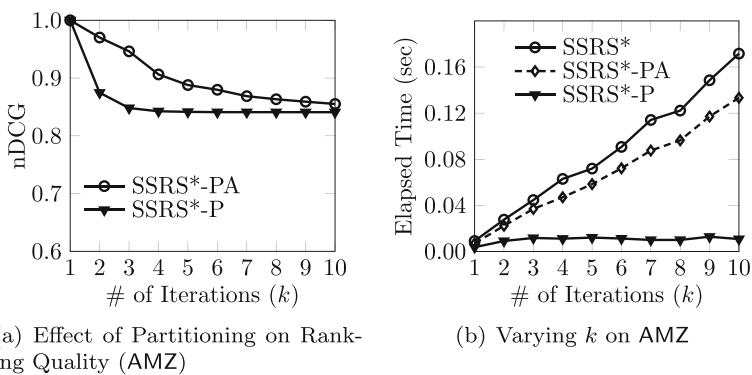
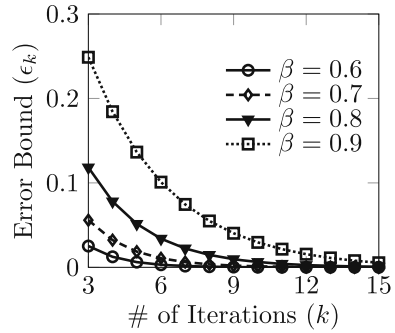


Figure 19 Accuracy and CPU time comparison for partitioned SSRS* (AMZ)

Figure 20 Effect of (k, β) on ϵ_k



like SSRS*-P may offer a more scalable computation of approximate similarity scores even for large number of iterations.

Iterative Error. Finally, we evaluate the effects of number of iterations k on the iterative error of RS*. The error is measured by difference ϵ_k between k -th iterative score $s_k(*, *)$ and exact one $s(*, *)$. We only report the results for a pair of nodes on DBLP since the trends for other pairs and on other datasets are similar. For any pair of nodes on DBLP, we fix damping factor β , and vary k from 1 to 15.

Figure 20 depicts how k -th iterative error ϵ_k changes with k . It is discerned that, for any given damping factor β , ϵ_k exponentially decreases to 0 as k grows. The larger damping factor β will cause a shift outward in the accuracy curve, thereby exhibiting the slower convergence rate of RoleSim* iterations. In addition, at each iteration k , it is noticed that small settings of damping factor β will lead to small iterative error of RoleSim*. These agree well with our theoretical bound $k = \lceil \log_{\beta} \epsilon_k \rceil$ in Theorem 4 for RoleSim* accuracy analysis.

The actual and the estimated error bound value for $\beta = 0.6$ and $\beta = 0.8$ per iteration are illustrated in Table 3, which shows that, for each iteration, the computed actual error bounds are completely compatible with the theoretical estimated error bounds.

Table 3 Actual & Estimated Error

#- Iter. (k)	$\beta = 0.6$		$\beta = 0.8$	
	Actual Error (ϵ_k)	Estimated Bound (β^{k+1})	Actual Error (ϵ_k)	Estimated Bound (β^{k+1})
1	0.1029	0.3600	0.2717	0.6400
2	0.0509	0.2160	0.1793	0.5120
3	0.0252	0.1296	0.1183	0.4096
4	0.0125	0.0778	0.0780	0.3277
5	0.0062	0.0467	0.0515	0.2621
6	0.0031	0.0280	0.0339	0.2097
7	0.0015	0.0168	0.0224	0.1678
8	0.0008	0.0101	0.0147	0.1342
9	0.0004	0.0060	0.0097	0.1074
10	0.0002	0.0036	0.0064	0.0859

9 Conclusions

We propose RoleSim*, a novel similarity model that guarantees automorphic equivalence while considering neighboring similarity information beyond automorphically equivalent sets, thereby achieving better performance than both SimRank and RoleSim. We prove the existence and uniqueness of the RoleSim* solution, show that iteratively computing RoleSim* is bounded, and induce a RoleSim* distance obeying sum-transitivity of similarity scores. We also propose a threshold-based RoleSim* model to prune a number of pairs with tiny similarity values, which enables a further speedup with guaranteed accuracy. Moreover, we propose an efficient single-source RoleSim* algorithm that scales well on large graphs with billions of edges. Taking advantage of the “triangular inequality” property of RoleSim*, we also introduce a partitioning-based strategy to scale RoleSim* on large graphs. Finally, we evaluate our model on different real datasets to demonstrate its superior ranking quality, fast speed, and high scalability against state-of-the-art competitors.

Acknowledgements A preliminary version of this work has been published in [33]. We summarise the main changes to [33] as follows: 1) For techniques and methods, we add three new sections on top of [33]: Section 4 (threshold-based RoleSim*), Section 5 (scaling single-source RoleSim* search on large graphs), and Section 6 (top-K efficient RoleSim* search using triangle inequality and partitioning). 2) Experiments (Section 8.2). We conduct additional experiments to demonstrate the high efficiency of the threshold-based RoleSim* and high scalability of single-source RoleSim* search on more sizable datasets. 3) Related Work (Section 2). We also add new related work that has appeared recently to make the paper more complete. This work was supported by the National Natural Science Foundation of China (NSFC 61972203), and Natural Science Foundation of Jiangsu Province (BK20190442). Aparajita Haldar is supported by a Feuer International Scholarship in Artificial Intelligence.

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/>.

References


1. Antonellis, I., Garcia-Molina, H., Chang, C.-C.: SimRank++: Query rewriting through link analysis of the click graph. *PVLDB*, 1(1) (2008)
2. Bijsterbosch, J., Volgenant, A.: Solving the rectangular assignment problem and applications. *Ann. Oper. Res.* **181**(1), 443–462 (2010)
3. Chen, H., Giles, C.L.: ASCOS++: an asymmetric similarity measure for weighted networks to address the problem of simrank. *ACM Trans. Knowl. Discov. Data* **10**(2), 15:1–15:26 (2015)
4. Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Onizuka, M.: Efficient search algorithm for SimRank. In: *ICDE*, pp. 589–600 (2013)
5. He, G., Feng, H., Li, C., Chen, H.: Parallel SimRank computation on large graphs with iterative aggregation. In: *KDD* (2010)
6. He, J., Liu, H., Yu, J.X., Li, P., He, W., Du, X.: Assessing single-pair similarity over graphs by aggregating first-meeting probabilities. *Inf. Syst.* **42**, 107–122 (2014)
7. Jeh, G., Widom, J.: SimRank: A measure of structural-context similarity. In: *KDD*, pp. 538–543 (2002)
8. Jiang, M., Fu, A.W., Wong, R.C., Wang, K.: READS: A random walk approach for efficient and accurate dynamic simrank. *PVLDB* **10**(9), 937–948 (2017)

9. Jin, R., Lee, V.E., Hong, H.: Axiomatic ranking of network role similarity. In: Apté, C., Ghosh, J., Smyth, P. (eds.) Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21–24, 2011, pp. 922–930. ACM (2011)
10. Jin, R., Lee, V.E., Li, L.: Scalable and axiomatic ranking of network role similarity. *TKDD* **8**(1), 3:1–3:37 (2014)
11. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Scientif. Comput.* **20**(1), 359–392 (1998)
12. Kusumoto, M., Maehara, T., Kawarabayashi, K.: Scalable Similarity Search for SimRank. In: SIGMOD, pp. 325–336 (2014)
13. 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: EDBT (2010)
14. Li, L., Qian, L., Lee, V.E., Leng, M., Chen, M., Chen, X.: Fast and accurate computation of role similarity via vertex centrality. In: Li, J., Sun, Y. (eds.) WAI, volume 9098 of Lecture Notes in Computer Science, pp. 123–134. Springer (2015)
15. Li, P., Liu, H., Yu, J.X., He, J., Du, X.: Fast single-pair simrank computation. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA, pp. 571–582. SIAM (2010)
16. Li, Z., Fang, Y., Liu, Q., Cheng, J., Cheng, R., Lui, J.C.S.: Walking in the cloud: Parallel SimRank at scale. *PVLDB* **9**(1), 24–35 (2015)
17. Lin, Y., Sundaram, H., Chi, Y., Tatemura, J., Tseng, B.L.: Detecting splogs via temporal dynamics using self-similarity analysis. *TWEB* **2**(1), 4:1–4:35 (2008)
18. Lin, Z., Lyu, M.R., King, I.: Matchsim: a novel similarity measure based on maximum neighborhood matching. *Knowl. Inf. Syst.* **32**(1), 141–166 (2012)
19. 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. *PVLDB* **11**(1), 14–26 (2017)
20. Lizorkin, D., Velikhov, P., Grinev, M.N., Turdakov, D.: Accuracy estimate and optimization techniques for SimRank computation. *VLDB J.* **19**(1) (2010)
21. Lu, J., Gong, Z., Yang, Y.: A matrix sampling approach for efficient SimRank computation. *Inf. Sci.* **556**, 1–26 (2021)
22. Maehara, T., Kusumoto, M., Kawarabayashi, K.: Scalable Simrank Join Algorithm. In: ICDE, pp. 603–614 (2015)
23. Rothe, S., Schütze, H.: CoSimRank: A Flexible & Efficient Graph-Theoretic Similarity Measure. In: ACL, pages 1392–1402. The Association for Computer Linguistics (2014)
24. Shao, Y., Cui, B., Chen, L., Liu, M., Xie, X.: An efficient similarity search framework for SimRank over large dynamic graphs. *PVLDB* **8**(8), 838–849 (2015)
25. Shao, Y., Liu, J., Shi, S., Zhang, Y., Cui, B.: Fast de-anonymization of social networks with structural information. *Data Sci. Eng.* **4**(1), 76–92 (2019)
26. Tian, B., Xiao, X.: SLING: A Near-Optimal Index Structure for SimRank. In: SIGMOD, pp. 1859–1874 (2016)
27. Wang, H., Wei, Z., Yuan, Y., Du, X., Wen, J.: Exact single-source SimRank computation on large graphs. In: Maier, D., Pottinger, R., Doan, A., Tan, W., Alawini, A., Ngo, H.Q. (eds.) 653–663. ACM (2020)
28. Wang, Y., Lian, X., Chen, L., 545–556: Efficient Simrank Tracking in Dynamic Graphs. In: ICDE (2018)
29. Wei, Z., He, X., Xiao, X., Wang, S., Liu, Y., Du, X., Wen, J.: PRSim: Sublinear time simrank computation on large power-law graphs. In: Boncz, P.A., Manegold, S., Ailamaki, A., Deshpande, A., Kraska, T. (eds.) SIGMOD, pp. 1042–1059. ACM (2019)
30. Xi, W., Fox, E.A., Fan, W., Zhang, B., Chen, Z., Yan, J., Zhuang, D.: Simfusion: Measuring Similarity Using Unified Relationship Matrix. In: SIGIR (2005)
31. Yoon, S., Kim, S., Park, S.: C-rank: A link-based similarity measure for scientific literature databases. *Inf. Sci.* **326**, 25–40 (2016)
32. Youngmann, B., Milo, T., Somech, A.: Boosting SimRank with Semantics. In: EDBT, pp. 37–48 (2019)
33. Yu, W., Iranmanesh, S., Haldar, A., Zhang, M., Ferhatosmanoglu, H.: An Axiomatic Role Similarity Measure Based on Graph Topology. In: Software Foundations for Data Interoperability and Large Scale Graph Data Analytics, pp. 33–48. Springer International Publishing (2020)
34. Yu, W., Lin, X., Zhang, W.: Towards Efficient SimRank Computation on Large Networks. In: ICDE, pp. 601–612 (2013)
35. Yu, W., Lin, X., Zhang, W., McCann, J.A.: Fast all-pairs SimRank assessment on large graphs and bipartite domains. *IEEE Trans. Knowl. Data Eng.* **27**(7), 1810–1823 (2015)
36. Yu, W., Lin, X., Zhang, W., McCann, J.A.: Dynamical SimRank search on time-varying networks. *VLDB J.* **27**(1), 79–104 (2018)

37. Yu, W., Lin, X., Zhang, W., Pei, J., McCann, J.A.: SimRank*: Effective and scalable pairwise similarity search based on graph topology. *VLDB J.* **28**(3), 401–426 (2019)
38. Yu, W., Lin, X., Zhang, W., Zhang, Y., Le, J.: Simfusion+: Extending SimFusion Towards Efficient Estimation on Large and Dynamic Networks. In: *SIGIR*, pp. 365–374 (2012)
39. Yu, W., McCann, J.A.: Efficient partial-pairs SimRank search for large networks. *PVLDB* **8**(5), 569–580 (2015)
40. Yu, W., Wang, F.: Fast Exact CoSimRank Search on Evolving and Static Graphs. In: *WWW*, pp. 599–608 (2018)
41. Zhao, P., Han, J., Sun, Y.: P-Rank: A Comprehensive Structural Similarity Measure over Information Networks. In: *CIKM* (2009)
42. Zhu, R., Zou, Z., Li, J.: Simrank Computation on Uncertain Graphs. In: *ICDE*, pp. 565–576 (2016)

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

Affiliations

Weiren Yu^{1,2}  · Sima Iranmanesh² · Aparajita Haldar² · Maoyin Zhang¹ · Hakan Ferhatosmanoglu²

Sima Iranmanesh
sima.iranmanesh@warwick.ac.uk

Aparajita Haldar
aparajita.haldar@warwick.ac.uk

Maoyin Zhang
maoyinzhang@hotmail.com

Hakan Ferhatosmanoglu
h.ferhatosmanoglu@warwick.ac.uk

¹ Nanjing University of Science and Technology, Jiangsu, China

² University of Warwick, Coventry, CV4 7AL, UK