Efficient Pairwise Penetrating-Rank Similarity Retrieval

WEIREN YU, [1]Nanjing University of Science and Technology, China [2]University of Warwick, UK
JULIE MCCANN, Imperial College, UK
CHENGYUAN ZHANG, Central South University, China

Many web applications demand a measure of similarity between two entities, such as collaborative filtering, web document ranking, linkage prediction, and anomaly detection. P-Rank (Penetrating-Rank) has been accepted as a promising graph-based similarity measure as it provides a comprehensive way of encoding both incoming and outgoing links into assessment. However, the existing method to compute P-Rank is iterative in nature and rather cost-inhibitive. Moreover, the accuracy estimate and stability issues for P-Rank computation have not been addressed. In this paper, we consider the optimization techniques for P-Rank search that encompasses its accuracy, stability and computational efficiency. (1) The accuracy estimation is provided for P-Rank iterations, with the aim to find out the number of iterations, k, required to guarantee a desired accuracy. (2) A rigorous bound on the condition number of P-Rank is obtained for stability analysis. Based on this bound, it can be shown that P-Rank is stable and well-conditioned when the damping factors are chosen to be suitably small. (3) Two matrix-based algorithms, applicable to digraphs and undirected graphs, are respectively devised for efficient P-Rank computation, which improves the computational time from $O(kn^3)$ to $O(vn^2 + v^6)$ for digraphs, and to $O(vn^2)$ for undirected graphs, where n is the number of vertices in the graph, and v ($\ll n$) is the target rank of the graph. Moreover, our proposed algorithms can significantly reduce the memory space of P-Rank computations from $O(n^2)$ to $O(vn + v^4)$ for digraphs, and to $O(vn)$ for undirected graphs, respectively. Finally, extensive experiments on real-world and synthetic datasets demonstrate the usefulness and efficiency of the proposed techniques for P-Rank similarity assessment on various networks.

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

Additional Key Words and Phrases: Similarity Search, Hyperlink Analysis, Web Document Ranking, Optimization

ACM Reference Format:
Weiren Yu, Julie McCann, and Chengyuan Zhang. 2019. Efficient Pairwise Penetrating-Rank Similarity Retrieval. *ACM Trans. Web* 37, 4, Article 111 (August 2019), 51 pages. https://doi.org/XXXX/XXXXXXXX

1 INTRODUCTION

The problem of quantifying similarity between entities based on network structure has witnessed growing interests over the last decades. There are various circumstances in which it would be useful to answer the questions such as "How similar are every two entities (vertices)?", and "Which

Authors' addresses: Weiren Yu, [1]Nanjing University of Science and Technology, 200 Xiaolingwei Street, Xuanwu District, Department of Computer Science and Technology, Nanjing, Jiangsu, 210094, China, [2]University of Warwick, Department of Computer Science, Coventry, CV4 7AL, UK, weiren.yu@warwick.ac.uk; Julie McCann, Imperial College, 180 Queen's Gate, Department of Computing, London, SW7 2RH, UK, j.mccann@imperial.ac.uk; Chengyuan Zhang, Central South University, 932 Lushan South Rd, Yuelu District, School of Computer Science and Engineering, Changsha, Hunan, 410083, China, cyzhang@csu.edu.cn.

1559-1131/2019/8-ART111 $15.00
https://doi.org/XXXX/XXXXXXXX

other entities (vertices) are most similar to a specific query (a given query vertex)?". Unlike textual similarity (*e.g.,* Jaccard similarity, Gram-based similarity) that may tokenize entities as the bag-of-words, structural similarity utilizes inherent hyperlink relationships to convey useful semantic information among entities. In recent years, there have emerged many appealing similarity measures that hinges purely on the structure topology of networks. Some renowned examples include Penetrating-Rank (P-Rank) [54], SimRank [14], Personalized PageRank (PPR) [32], Random Walk with Restart (RWR) [38], RoleSim [17], SimFusion [40], CoSimRank [52], and SimRank* [48]. These similarity models show great success in proliferative applications, *e.g.,* biological networks, recommendation systems, automatic image captioning, synonyms extraction, and outlier detection.

P-Rank is a promising similarity measure of this kind, which was proposed by Zhao *et al.* [54]. It encodes both incoming and outgoing links of entities into similarity assessment. The P-Rank similarities flowing from in-link neighbors of entities are penetrated through their out-link neighbors in a recursive fashion. In contrast to other similarity measures, P-Rank has stood out as an attractive one, due to the following advantages:

- **Semantic Completeness.** P-Rank provides a comprehensive way of jointly considering both in- and out-link relationships in a network with semantic completeness. In contrast, other similarity measures (*e.g.,* SimRank, and PPR) have the "limited information problem" [14], that is, only in-links are partially exploited, whereas out-links are totally ignored.
- **Adaptivity.** Similar to other link-based similarity measures, P-Rank can be combined with other domain-specific similarity measures (*e.g.,* [18]) to produce an overall measure, which is adaptive to any domains with entity-to-entity relationships.
- **Generality.** P-Rank formula has a general form that makes itself transcend other existing similarity measures [54]. As will be seen in Section 2, the P-Rank similarity $s^{(k)}(u,v)$ between two distinct nodes u and v at iteration k is defined iteratively in terms of the average similarity of (u,v)'s in- and out-neighbouring pairs at iteration $(k-1)$ as follows:

$$\begin{aligned} s^{(k)}(u,v) = {} & \lambda \cdot C_{\text{in}} \cdot \{\text{average similarity of } (u,v)\text{'s in-neighbour pairs at iteration } (k-1)\} \\ & +(1-\lambda) \cdot C_{\text{out}} \cdot \{\text{average similarity of } (u,v)\text{'s out-neighbour pairs at iteration } (k-1)\} \end{aligned} \tag{1}$$

  where λ is the weight factor balancing the importance of in- and out-links; and C_{in} and C_{out} are in- and out-link damping factors, respectively. The P-Rank model covers many well-known similarity measures (*e.g.,* SimRank [14] and Amsler [1]) as its special cases, when the parameters $(C_{\text{in}}, C_{\text{out}})$, λ, and k are chosen as follows:

	In-link	**Out-link**	**Both**
1-hop neighborhood	CoCitation $k=1, C_{\text{in}}=1, \lambda=1$	Coupling $k=1, C_{\text{out}}=1, \lambda=0$	Amsler $k=1, C_{\text{in}}=C_{\text{out}}=1, \lambda=1/2$
multi-hop neighborhood	SimRank $k=\infty, \lambda=1$	rvs-SimRank $k=\infty, \lambda=0$	P-Rank $k=\infty$

  For example, as illustrated in the table, when setting $k=\infty$ and $\lambda=0$ in Eq.(1), the P-Rank model reduces to Jeh and Widom's SimRank model [14] as follows:

$$s(u,v) = C_{\text{in}} \cdot \{\text{average similarity of } (u,v)\text{'s in-neighbour pairs}\}$$

  where only in-neighbour pairs of (u,v) recursively contribute to SimRank similarity $s(u,v)$.

Therefore, P-Rank has been recognized as an important and common similarity measure, which has a wide spectrum of real applications in fertile communities where other similarity measures (*e.g.,* Co-citation, SimRank, Amsler) are applicable, such as collaborative filtering, graph clustering, link prediction, and web document ranking (see [2, 15, 29, 50, 51, 55] and references therein).

Nevertheless, previous work on P-Rank leaves several challenging issues unaddressed.

Firstly, it is not straightforward to obtain a tight bound to better estimate the accuracy for P-Rank iterations. Although the convergence of P-Rank iterations has been proved in [54], it is still difficult to determine the total number of iterations needed for guaranteeing a given accuracy. To the best of our knowledge, there is only one work [29] that estimates the accuracy for SimRank iterations. That work provides an upper bound C^{k+1} for the difference between the k-th iterative SimRank and the exact one. However, if the SimRank upper bound C^{k+1} in [29] is directly applied to P-Rank accuracy estimation by simply taking the linear combination of the upper bounds for in- and out-link terms, the resulting bound $(\lambda \cdot {C_{\text{in}}}^{k+1} + (1-\lambda) \cdot {C_{\text{out}}}^{k+1})$ is less tight for P-Rank iterations, as will be explained in Section 3. Thus, it is imperative to derive a new upper bound, which is tight, to better estimate the accuracy of P-Rank iterations.

Secondly, no prior work has studied the stability of P-Rank. Indeed, P-Rank stability plays an important role in real applications as it 1) can gauge the sensitivity of similarity results to slight perturbations in the graph structure (*e.g.,* by adding or removing edges) and 2) implies whether large amounts of accumulated roundoff errors in the P-Rank computation may run the risk of producing nonsensical similarity results. To analyse the P-Rank stability, we provide a tight upper bound for P-Rank condition number $\kappa_\infty(\mathcal{G})$ based on the closed-form of P-Rank. As will be explained in Section 4.2, we define the P-Rank condition number $\kappa_\infty := \|\mathbf{M}\|_\infty \cdot \|\mathbf{M}^{-1}\|_\infty$ in terms of a large matrix $\mathbf{M}$ of size $n^2 \times n^2$ that contains the information of the entire graph structure, where n is the number of nodes in the graph $\mathcal{G}$. A complicated problem is how to efficiently compute $\|\mathbf{M}^{-1}\|_\infty$ in the closed-form of P-Rank since the traditional approach entails $O(n^6)$ time to inverse the large $n^2 \times n^2$ matrix $\mathbf{M}$ and $O(n^4)$ time to compute the ∞-norm of the large $n^2 \times n^2$ matrix $\mathbf{M}^{-1}$, which is prohibitively expensive. To address this issue, we propose a novel $O(1)$-time method to obtain a neat tight bound for the P-Rank condition number for P-Rank stability analysis.

Thirdly, it is a grand challenge to improve the computational complexity of P-Rank. The naive method computing P-Rank via the fixed-point iteration requires $O(kn^4)$ time for k iterations, which is inapplicable to large networks. For approximating P-Rank, Zhao *et al.* [54] have proposed the radius- or category- based pruning techniques to improve the estimation of P-Rank to $O(kd^2n^2)$ worst-case time, where d is the graph average degree. However, this method is inherently heuristic, and even worse, no theoretical guarantee is provided for the approximation error of the pruning results. More importantly, our analysis in Section 4 showed that large settings of (C_{in}, C_{out}) lead to good stability of P-Rank scores, but will make the iteration-based P-Rank solution much slower. The reason is that, for the iteration-based P-Rank, our accuracy estimate results in Section 3 proved that large settings of (C_{in}, C_{out}) will increase the number of iterations to attain a given accuracy. Thus, to achieve both accuracy and good stability, we aim to devise a novel non-iterative method to compute P-Rank that will not produce iterative errors relying on (C_{in}, C_{out}) settings. Fortunately, we have an observation that a large body of vertices in a real network usually share some similar neighborhood structures (*e.g.,* similar user preference in a recommender system). Thus, we have an opportunity to "merge" these similar vertices, and devise fast non-iterative algorithms to speed up P-Rank computation.

Contributions. The big picture of our work centers on the three aspects of P-Rank: (P1) accuracy estimate for P-Rank iterative model; (P2) stability analysis based on the closed-form of P-Rank; and (P3) accelerative techniques for efficient P-Rank computation based on our non-iterative model. Specifically, our main results are summarized below.

P1) We provide an accuracy estimation for P-Rank iteration-based model. (Section 3). We show that $k = \lceil \ln \epsilon / \ln (\lambda \cdot C_{\text{in}} + (1-\lambda) \cdot C_{\text{out}}) \rceil$ iterations suffice to guarantee a desired accuracy ϵ, which implies that, for P-Rank iteration-based model, large settings of (C_{in}, C_{out}) will reduce the number of iterations to attain a given accuracy. However, as will be seen in Section 4,

such settings of (C_{in}, C_{out}) will prevent P-Rank from achieving good stability. Thus, the results in this section motivate us to study another non-iterative model (shown in Section 5) that does not produce iterative errors relying on (C_{in}, C_{out}) for P-Rank computation.

P2) We introduce the notion of *P-Rank condition number* κ_∞ to analyze the stability of P-Rank (Section 4). We develop a new eigenvector-based approach to obtain a tight bound for κ_∞, and provide the conditions under which P-Rank is *stable*, that is, slight perturbations in the link structure will not cause large changes in the P-Rank similarity. We provide a real application to show how to use P-Rank condition number to set appropriate hyper-parameters for P-Rank which can improve the robustness and accuracy of the decentralized computing of P-Rank.

P3) We propose two novel matrix-based algorithms (DE P-Rank and UN P-Rank)[1] that can substantially speed up the computation of P-Rank from $O(kn^3)$ to $O(vn^2 + v^6)$ for digraphs, and to $O(vn^2)$ for undirected graphs (Section 5) with guaranteed accuracy, where v ($\ll n$) is the target rank of the graph. Besides, our proposed algorithms can significantly reduce the memory space of P-Rank computations from $O(n^2)$ to $O(vn + v^4)$ for digraphs, and to $O(vn)$ for undirected graphs, respectively. Our non-iterative algorithms in this section, unlike the iterative version in Section 3, does not produce iterative errors that hinge on (C_{in}, C_{out}) settings. Thus, for our non-iterative model, by setting small values of (C_{in}, C_{out}), we can achieve both high computational efficiency and good stability at the same time.

We empirically verify the efficiency of our methods on real and synthetic data (Section 6). The experimental results show that (1) P-Rank converges exponentially *w.r.t.* the iteration number; (2) the stability of P-Rank is sensitive to different choices of the damping factors and the weighted factor; (3) the proposed DE P-Rank and UN P-Rank outperform its competitors by up to one order of magnitude, and scale well over large networks.

Relations among (P1-P3). The relations among the three P-Rank problems (P1–P3) are as follows: In (P1), our accuracy estimate is based on the traditional iteration-based P-Rank equation, aiming at showing the limitation of this iteration-based approach, *i.e.,* it produces an iterative error that relies on (C_{in}, C_{out}) settings. Our quantitative results for (P1) indicate that, for achieving a given accuracy, large settings of (C_{in}, C_{out}) will reduce the number of iterations, and thus will speed up the computation of the P-Rank iteration-based approach. Unfortunately, our study for (P2) shows that, from the stability perspective, large settings of (C_{in}, C_{out}), which may increase the P-Rank conditional number, would make P-Rank results unstable. Thus, based on (P1) and (P2), we are aware that, for the iteration-based approach, it is difficult to find a good (C_{in}, C_{out}) for achieving both computational efficiency and stability at the same time. To resolve this issue, we next study (P3) and propose a non-iterative solution for P-Rank computation, which will never produce iterative errors that hinge on (C_{in}, C_{out}) settings. Based on (P2) and (P3), we notice that, for the non-iterative approach, high computational efficiency and good stability can be achieved at the same time for small settings of (C_{in}, C_{out}).

Our preliminary conference versions of P-Rank optimization methods were given in [24, 43]. In the current paper, we substantially extend our previous work [24, 43] by providing (i) the complete and rigorous mathematical proofs of all the lemmas and theorems in [24, 43]; (ii) theoretical proofs and examples to show that, for P-Rank iterative accuracy estimate, our error bound derived in Theorem 1 is always tighter than the straightforward extension of the SimRank bound in [22] to P-Rank; (iii) a detailed description of the closed-form P-Rank (Section 4.1); (iv) theoretical discussions to show how P-Rank conditional number is related to the robustness of P-Rank similarity (Section 4.2), along with some motivating examples to illustrate how hyper-parameters affects robustness

[1] DE P-Rank (*resp.* UN P-Rank) is named after its functionality of P-Rank search on DirEcted (*resp.* UNdirected) graphs.

Symbols	Definition	Symbols	Definition
$\mathcal{G}$	network	r	rank of graph adjacency matrix ($r \ll n$)
$\mathcal{I}(a)$	in-neighbors of vertex a	v	low rank of P-Rank approximation ($v \leq r$)
$\mathcal{O}(a)$	out-neighbors of vertex a	$s(a,b)$	P-Rank score between vertices a and b
n	number of vertices in $\mathcal{G}$	$C_{\text{in}}/C_{\text{out}}$	in-link/out-link damping factor
m	number of edges in $\mathcal{G}$	$\mathbf{A}$	adjacency matrix of $\mathcal{G}$
K	number of iterations	$\mathbf{S}$	P-Rank similarity matrix of $\mathcal{G}$
λ	weighting factor	$\mathbf{I}$	identity matrix

Table 1. Glossary of Symbols

of P-Rank on the real end-to-end application (*e.g.,* finding top-K most similar authors on DBLP); (v) more insights on how P-Rank stability analysis improves the effectiveness for decentralised P-Rank computing, and a detailed analysis on the trade-off between P-Rank iterative Accuracy and stability; (vi) two new observations to speed up P-Rank computations further over digraphs in Section 5.1; (vii) an improved version of the $O(rn^2)$-time UN P-Rank algorithm for P-Rank computation on undirected networks, which is a nontrivial extension of the $O(n^3)$-time ASAP algorithm in [24]; (viii) several additional experiments and some best-known competitors (*e.g.,* SemSim in EDBT 2019) in Sections 6.2.3 and 6.2.6, evaluating the DE P-Rank algorithm that focuses on computational time and accuracy; and (ix) a complete performance comparison of DE P-Rank and UN P-Rank with other baseline algorithms in Section 6.2.2; (x) more state-of-art references (since 2015) and a variety of approaches on graph neural networks (*e.g.,* DeepWalk, Node2vec, GraRep, and Graph Convolutional Networks) and update the related work (Section 7).

2 PRELIMINARIES

In accordance with [54], we assume that graphs studied in this paper have no multiple edges (corresponding to a 0-1 adjacency matrix). Table 1 lists the notations used throughout the paper.

The basic essence behind the P-Rank model [54] involves the following three facets:

1) Two entities are similar if they are referenced by similar entities. (*In-link Recursion*)
2) Two entities are similar if they reference similar entities. (*Out-link Recursion*)
3) Every entity is maximally similar to itself. (*Base Case*)

P-Rank Similarity. We revisit the formulation of P-Rank [54]. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of vertices, $\mathcal{V}$, and a set of edges, $\mathcal{E}$, the P-Rank model can be formulated as follows:

For every two distinct vertices u and v in $\mathcal{V}$, the similarity $s(u,v) \in [0,1]$ defined as

$$s(u,u) = 1; \tag{2}$$

$$s(u,v) = \overbrace{\frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|} \sum_{j=1}^{|\mathcal{I}(v)|} s\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right)}^{\text{in-link part}} + \underbrace{\frac{(1-\lambda) \cdot C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|} \sum_{j=1}^{|\mathcal{O}(v)|} s\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right)}_{\text{out-link part}}, \tag{3}$$

is called *the P-Rank similarity* between u and v, where (i) $\lambda \in [0,1]$ is a weight factor, balancing the importance between in-links and out-links; (ii) C_{in} and $C_{\text{out}} \in (0,1)$ are the damping factors for in- and out-link directions, respectively; (iii) $\mathcal{I}(u)$ and $\mathcal{O}(u)$ are the in- and out-neighbor set of vertex

u, respectively, with $\mathcal{I}_i(u)$ and $\mathcal{O}_i(u)$ being the i-th elements of $\mathcal{I}(u)$ and $\mathcal{O}(u)$, respectively; and (iv) $|\mathcal{I}(u)|$ and $|\mathcal{O}(u)|$ are the cardinalities of $\mathcal{I}(u)$ and $\mathcal{O}(u)$, respectively.

To avoid $s(u,v)=\infty$ in Eq.(3), we assume that

1) in-link part of Eq.(3)$=0$ if $\mathcal{I}(u)=\varnothing$ or $\mathcal{I}(v)=\varnothing$;
2) out-link part of Eq.(3)$=0$ if $\mathcal{O}(u)=\varnothing$ or $\mathcal{O}(v)=\varnothing$.

P-Rank Iterative Paradigm. The conventional method iteratively computes $s(u,v)$ as follows:

$$s^{(0)}(u,v)=\begin{cases}0, \text{ if } u\neq v;\\ 1, \text{ if } u=v.\end{cases} \tag{4}$$

For each iteration $k=1,2,\cdots$, the k-th iterative P-Rank similarity $s^{(k)}(u,v)$ is iteratively computed as

$$\begin{aligned}
&s^{(k)}(u,u)=1;\\
&s^{(k)}(u,v)=\frac{\lambda\cdot C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|}\sum_{i=1}^{|\mathcal{I}(u)|}\sum_{j=1}^{|\mathcal{I}(v)|}s^{(k-1)}\left(\mathcal{I}_i(u),\mathcal{I}_j(v)\right)\\
&\qquad\qquad+\frac{(1-\lambda)\cdot C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|}\sum_{i=1}^{|\mathcal{O}(u)|}\sum_{j=1}^{|\mathcal{O}(v)|}s^{(k-1)}\left(\mathcal{O}_i(u),\mathcal{O}_j(v)\right);\\
&s^{(k)}(u,v)=\text{Eq.(5)'s in-link part, if } \mathcal{O}(u)=\varnothing \text{ or } \mathcal{O}(v)=\varnothing;\\
&s^{(k)}(u,v)=\text{Eq.(5)'s out-link part, if } \mathcal{I}(u)=\varnothing \text{ or } \mathcal{I}(v)=\varnothing.
\end{aligned} \tag{5}$$

It was proved in [54] that the sequence $\{s^{(k)}(u,v)\}$ non-decreasingly converges to the exact similarity $s(u,v)$, *i.e.,*

$$\lim_{k\to\infty}s^{(k)}(u,v)=s(u,v)\quad(\forall u,v\in\mathcal{V}). \tag{6}$$

3 P-RANK ACCURACY ESTIMATE

Despite the convergence of the sequence $\{s^{(k)}(u,v)\}_{k=0}^{\infty}$, the gap between the k-th iterative similarity $s^{(k)}(u,v)$ and the exact one $s(u,v)$ still remains unknown. This motivates us to study *the P-Rank accuracy estimate problem*:

Given a network $\mathcal{G}$, for each iteration number $k=1,2,\cdots$, it is to find a tight bound ϵ_k for the difference between the k-th iterative similarity $s^{(k)}(u,v)$ and the exact one $s(u,v)$ for $\forall u,v\in\mathcal{G}$.

The main result of this section is the following.

Theorem 1. *The P-Rank accuracy estimate problem has a tight upper bound*

$$\epsilon_k=(\lambda C_{in}+(1-\lambda)C_{out})^{k+1}$$

such that $\forall k=0,1,\cdots,\ \ \forall u,v\in\mathcal{V}$,

$$|s(u,v)-s^{(k)}(u,v)|\leq\epsilon_k. \tag{7}$$

Proof. See Appendix A.1. □

Theorem 1 provides an a-priori estimate for the gap between the iterative and exact P-Rank. For each iteration, this gap merely hinges on λ, C_{in} and C_{out}. To be precise, for guaranteeing high accuracy at each iteration, it follows from

$$\epsilon_k=(\lambda(C_{\text{in}}-C_{\text{out}})+C_{\text{out}})^{k+1}$$

that smaller choices of C_{in} and C_{out} (i) with a smaller λ if $C_{\text{in}}>C_{\text{out}}$, or (ii) with a larger λ if $C_{\text{in}}<C_{\text{out}}$, will result in a smaller ϵ_k, and are thus more preferable.

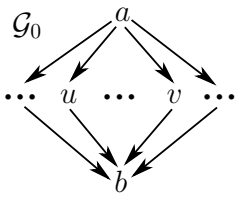


Fig. 1. "=" in Eq.(7) is attainable

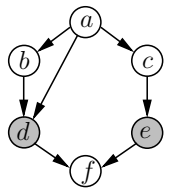


Fig. 2. Iterative Error of Computing $s(d, e)$

Example 1. *Setting $C_{in} = 0.6, C_{out} = 0.4, \lambda = 0.3, k = 5$ produces the following high accuracy for iterative P-Rank similarities computation:*

$$\epsilon_k = (0.3 \times 0.6 + (1 - 0.3) \times 0.4)^{5+1} = 0.0095. \qquad \square$$

It is worth noticing that the upper bound we obtained in Eq.(7) is tight for P-Rank iterations since the equality of this upper bound is attainable. Consider the network $\mathcal{G}_0$ in Figure 1. It is apparent that $s^{(0)}(u, v) = 0$. For $k = 1, 2, \cdots$, it can be easily obtained that $s^{(k)}(u, v) = \lambda C_{\text{in}} + (1 - \lambda)C_{\text{out}}$, which implies that $s(u, v) = \lambda C_{\text{in}} + (1 - \lambda)C_{\text{out}}$. Hence, in the case of $k = 0$,

$$|s(u, v) - s^{(k)}(u, v)| = (\lambda C_{\text{in}} + (1 - \lambda)C_{\text{out}})^{0+1},$$

which gives the precise upper bound in Eq.(7).

As a special case when $\lambda = 1$, Eq.(7) reduces to the SimRank accuracy estimate problem [29]. From this perspective, the P-Rank accuracy estimate problem is an extension of Proposition 1 in [29] by jointly considering both in- and out-links for similarity assessment.

However, it is important to note that, if the SimRank upper bound C^{k+1} in [29] is directly applied to P-Rank accuracy estimation by simply taking the linear combination of the upper bounds for in- and out-link terms, the resulting bound $(\lambda \cdot {C_{\text{in}}}^{k+1} + (1 - \lambda) \cdot {C_{\text{out}}}^{k+1})$ is always less tight than our upper bound $(\lambda \cdot C_{\text{in}} + (1 - \lambda) \cdot C_{\text{out}})^{k+1}$ for P-Rank iterations. The reason is as follows: For each iteration $k = 0, 1, 2, \cdots$, let $f(x) = x^k$ $(x \in (0, 1))$. Since $f(x)$ is a convex function, we have

$$\lambda \cdot f(C_{\text{in}}) + (1 - \lambda) \cdot f(C_{\text{out}}) \geq f(\lambda \cdot C_{\text{in}} + (1 - \lambda) \cdot C_{\text{out}})$$

which implies that the following inequality holds:

$$\lambda \cdot {C_{\text{in}}}^{k+1} + (1 - \lambda) \cdot {C_{\text{out}}}^{k+1} \geq (\lambda \cdot C_{\text{in}} + (1 - \lambda) \cdot C_{\text{out}})^{k+1}$$

Thus, our upper bound is tighter than the straightforward extension of the SimRank bound in [29].

Example 2. *Consider a graph $\mathcal{G}$ in Figure 2. Given $C_{in} = 0.8, C_{out} = 0.6, \lambda = 0.4$, the exact P-Rank similarity score between node d and e is $s(d, e) = 0.2023$. The following table compares, at each k step, the iterative error of P-Rank and the tightness of two upper bounds (Bound1 and Bound2) for P-Rank iterative accuracy estimate, where $Bound1 = \lambda \cdot {C_{in}}^{k+1} + (1-\lambda) \cdot {C_{out}}^{k+1}$ is the straightforward extension of the SimRank bound in [29], and $Bound2 = (\lambda \cdot C_{in} + (1 - \lambda) \cdot C_{out})^{k+1}$ is the tighter one we derived in Theorem 1.*

#-Iter k	*Iterative Error* $\lvert s(d,e) - s_k(d,e)\rvert$	*Bound1* $\lambda {C_{in}}^{k+1} + (1-\lambda){C_{out}}^{k+1}$	*Bound2* $(\lambda C_{in} + (1-\lambda)C_{out})^{k+1}$	*Bound Gap* $\lvert Bound1 - Bound2\rvert$
0	*0.2023*	*0.6800*	*0.6800*	*0*
1	*0.1802*	*0.4720*	*0.4624*	*0.0096*
2	*0.0372*	*0.3344*	*0.3144*	*0.0200*
3	*0.0084*	*0.2416*	*0.2138*	*0.0278*
4	*0.0020*	*0.1777*	*0.1454*	*0.0323*
5	*0.0005*	*0.1329*	*0.0989*	*0.0340*
6	*0.0001*	*0.1007*	*0.0672*	*0.0335*
7	*0.0001*	*0.0772*	*0.0457*	*0.0315*
8	*0.0000*	*0.0597*	*0.0311*	*0.0286*
9	*0.0000*	*0.0466*	*0.0211*	*0.0254*
10	...	...	...	...

From the results, we notice that, for each iteration $k \geq 2$, our Bound2 is always tighter than Bound1, and the tightness of Bound2 is more pronounced when k is increasing. □

The exponential P-Rank convergence rate in Theorem 1 implies that the total iteration number K of iterations needed for attaining a desired accuracy ϵ is

$$K = \lceil \ln \epsilon / \ln (\lambda \cdot C_{\text{in}} + (1-\lambda) \cdot C_{\text{out}}) \rceil .$$

4 STABILITY ANALYSIS OF P-RANK MODEL

In this section, the stability issue of P-Rank measure will be investigated for analyzing the sensitivity of P-Rank similarities in response to the perturbation of the graph structure. In Subsection 4.1, we will provide a matrix form of the P-Rank solution to Eq.(3). This form, referred to as "*the closed-form of P-Rank*", can express the P-Rank similarity $s(u,v)$ of Eq.(3) in terms of a finite number of matrix operations that hinge *only* on the graph structure information, in contrast to the original P-Rank definition in Eq.(3) (*a.k.a.* "*the recursive form of P-Rank*") that represents $s(u,v)$ in terms of the P-Rank similarity itself. In Subsection 4.2, based on this closed-form of the P-Rank solution, we introduce the concept of "*P-Rank condition number*". This concept will play a key role in P-Rank stability analysis as it can be used to measure how much the P-Rank similarities can change for a small change in the graph structure. We will provide an example of "decentralised P-Rank similarity search" to demonstrate the practicality of the "P-Rank condition number" for stability analysis in real applications.

4.1 A Closed-form of P-Rank Solution

Let us recall the original definition of P-Rank in Eq.(3). For two distinct nodes u and v in a graph, the P-Rank similarity $s(u,v)$ is defined in terms of $s(*,*)$ itself as follows:

$$s(u,v) = \frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|} \sum_{j=1}^{|\mathcal{I}(v)|} s\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right) + \frac{(1-\lambda) \cdot C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|} \sum_{j=1}^{|\mathcal{O}(v)|} s\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right)$$

In the above equation, since the P-Rank similarity $s(*,*)$ appears on both sides, we call it "*the recursive form of P-Rank*". One limitation of this form, from the computational perspective, is that, to get the exact value of a single-pair similarity $s(u,v)$, one need prepare all its in- and out-neighboring similarity pairs $\{s(x,y)\}_{(x,y)\in\mathcal{I}(u)\times\mathcal{I}(v)}$ and $\{s(x,y)\}_{(x,y)\in\mathcal{O}(u)\times\mathcal{O}(v)}$, recursively. Consequently, a single-pair similarity $s(u,v)$ in the worst case may hinge on retrieving almost all pairs

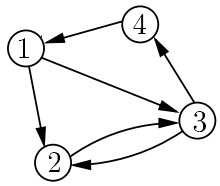


Fig. 3. A Web Graph G

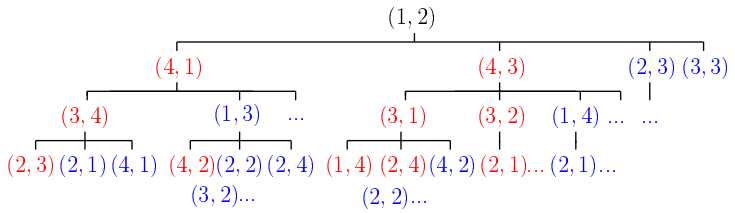


Fig. 4. Unrolling Recurrence of P-Rank Similarity $s(1,2)$ via Eq.(3)

of similarities $\{s(*,*)\}$ in a graph. Thus, this recursive form does not explicitly reflect how P-Rank similarities *purely* rely on the graph structure information.

Example 3. *Consider the graph $\mathcal{G}$ in Figure 3. To evaluate the P-Rank similarity $s(1,2)$ in $\mathcal{G}$, if we unroll the recurrence of Eq.(3) for three steps by repeatedly substituting $s(*,*)$ of the LHS into its RHS, it can be seen from Figure 4 that the single-pair similarity $s(1,2)$ relies on evaluating the similarities of almost all pairs of nodes in G except two singleton pairs $(1,1)$ and $(4,4)$. The tree structure in Figure 4 depicts the dependency (i.e., evaluation order) of node-pairs for P-Rank similarity assessment, where each pair in red (resp. blue) color denotes it is the in- (resp. out-) neighboring pair of the node-pair in its parent. For example, the parent $(1,2)$ in the tree has four children $(4,1),(4,3),(2,3),(3,3)$, meaning that, to evaluate P-Rank similarity $s(1,2)$ via Eq.(3), we need first evaluate similarities $s(4,1), s(4,3), s(2,3), s(3,3)$. Thus, the recursive form of Eq.(3) does not explicitly reflect how $s(1,2)$ is determined purely by the graph structure of $\mathcal{G}$.* □

Motivated by this, we next provide a new form of the P-Rank solution (*i.e.,* the closed-form) which characterises P-Rank similarity in terms of a finite number of matrix operations that hinge only on the graph structure. Unlike the recursive form whose $s(*,*)$ appear on both sides of Eq.(3), our closed-form can explicitly reflect how P-Rank scores purely depend on the graph structure, and guarantees that $s(*,*)$ will not appear on the right-hand side of the P-Rank expression. This closed-form will lay a foundation for our subsequent analysis of the stability of P-Rank similarities in response to the perturbation of the graph structure.

Notations. Before the closed-form of P-Rank is provided, we introduce some notations. For a network $\mathcal{G}$ with $n = |\mathcal{V}|$ vertices, we denote by

(i) $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{n\times n}$ the *adjacency matrix* of $\mathcal{G}$ whose entry $a_{i,j}$ is 1 if there exists an edge from vertex i to j, and 0 otherwise;

(ii) $\mathbf{S} = (s_{i,j}) \in \mathbb{R}^{n\times n}$ the *P-Rank similarity matrix* whose entry $s_{i,j}$ equals the P-Rank score $s(i,j)$ between vertices i and j;

(iii) $\mathbf{Q} = (q_{i,j}) \in \mathbb{R}^{n\times n}$ and $\mathbf{P} = (p_{i,j}) \in \mathbb{R}^{n\times n}$ the *one-step backward and forward transition probability matrix* of $\mathcal{G}$, respectively, whose entries are defined as follows:

$$q_{i,j} \triangleq \begin{cases} a_{j,i}/\sum_{j=1}^{n} a_{j,i}, & \text{if } \mathcal{I}(i) \neq \varnothing; \\ 0, & \text{if } \mathcal{I}(i) = \varnothing. \end{cases} \quad p_{i,j} \triangleq \begin{cases} a_{i,j}/\sum_{j=1}^{n} a_{i,j}, & \text{if } \mathcal{O}(i) \neq \varnothing; \\ 0, & \text{if } \mathcal{O}(i) = \varnothing. \end{cases} \tag{8}$$

Example 4. *Recall the graph $\mathcal{G}$ in Figure 3. Its adjacency matrix $\mathbf{A}$, backward transition matrix $\mathbf{Q}$, and forward transition matrix $\mathbf{P}$ are as follows:*

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{Q} = \textit{row-norm}(\mathbf{A}^T) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{P} = \textit{row-norm}(\mathbf{A}) = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \square$$

We also introduce two matrix operators [7, p.180]:

(i) $\text{vec}(\mathbf{X}) \in \mathbb{R}^{n^2 \times 1}$ is defined to be the *vectorization* of the matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ formed by stacking the columns of $\mathbf{X}$ into a single column vector.

(ii) $\mathbf{X} \otimes \mathbf{Y}$ is the *Kronecker product* of the matrices $\mathbf{X}$ and $\mathbf{Y}$.

Example 5. *Given two matrices* $\mathbf{X} = \left[\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right]$ *and* $\mathbf{Y} = \left[\begin{smallmatrix} 5 & 6 \\ 7 & 8 \end{smallmatrix}\right]$, *it follows that*

$$vec(\mathbf{X}) = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix} \in \mathbb{R}^{4\times 1}, \qquad \mathbf{X} \otimes \mathbf{Y} = \left[\begin{array}{c|c} 1 \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 2 \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ \hline 3 \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 4 \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{array}\right] = \begin{bmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad \square$$

With the above notations, the P-Rank formulae (2) and (3) can be rewritten as [2]

$$\mathbf{S} = \lambda C_{\text{in}} \cdot \mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T + (1-\lambda) C_{\text{out}} \cdot \mathbf{P} \cdot \mathbf{S} \cdot \mathbf{P}^T + (1 - \lambda C_{\text{in}} - (1-\lambda) C_{\text{out}}) \cdot \mathbf{I}_n. \tag{9}$$

Theorem 2. *Let* $\mathbf{S}$ *be the solution to Eq.*(9), *and* $\mathbf{S}'$ *the solution to the equation:*

$$\mathbf{S}' = \lambda C_{in} \cdot \mathbf{Q} \cdot \mathbf{S}' \cdot \mathbf{Q}^T + (1-\lambda) C_{out} \cdot \mathbf{P} \cdot \mathbf{S}' \cdot \mathbf{P}^T + \mathbf{I}_n. \tag{10}$$

Then, the relative similarity ranking by $\mathbf{S}$ *is exactly the same as that by* $\mathbf{S}'$. *More precisely,*

$$\mathbf{S} \equiv \xi \cdot \mathbf{S}' \quad \textit{where } \xi = 1 - \lambda C_{in} - (1-\lambda) C_{out}.$$

Proof. See Appendix A.2. □

Theorem 2 implies that we can simply employ Eq.(10) to evaluate P-Rank similarities. This is because, comparing Eq.(9) with Eq.(10), we see that the coefficient $(1 - \lambda C_{\text{in}} - (1-\lambda) C_{\text{out}})$ of $\mathbf{I}_n$ in Eq.(9) merely contributes an overall multiplicative factor to P-Rank similarity. Hence, replacing this coefficient in Eq.(9) with 1 still preserves the *relative* rankings of the P-Rank similarities though the diagonal entries of $\mathbf{S}$ in this scenario may not equal 1s.[3]

Example 6. *Consider the graph* $\mathcal{G}$ *in Figure 3. Given the damping factors* $C_{in} = 0.6$ *and* $C_{out} = 0.6$, *and the weight factor* $\lambda = 0.4$, *the P-Rank similarity rankings of all the distinct node-pairs in* $\mathcal{G}$ *obtained by* $\mathbf{S}$ *in Eq.*(9) *and* $\mathbf{S}'$ *in Eq.*(10) *are, respectively, as follows:*

Rank	*Node-Pairs*	$\mathbf{S}$
1	$(1,2)$ *and* $(2,1)$	*0.154*
2	$(2,4)$ *and* $(4,2)$	*0.137*
3	$(1,3)$ *and* $(3,1)$	*0.118*
4	$(2,3)$ *and* $(3,2)$	*0.096*
5	$(3,4)$ *and* $(4,3)$	*0.065*
6	$(1,4)$ *and* $(4,1)$	*0.064*

Rank	*Node-Pairs*	$\mathbf{S}'$
1	$(1,2)$ *and* $(2,1)$	*0.385*
2	$(2,4)$ *and* $(4,2)$	*0.344*
3	$(1,3)$ *and* $(3,1)$	*0.295*
4	$(2,3)$ *and* $(3,2)$	*0.239*
5	$(3,4)$ *and* $(4,3)$	*0.162*
6	$(1,4)$ *and* $(4,1)$	*0.161*

It can be noticed that the relative orders of $\mathbf{S}$ *and* $\mathbf{S}'$ *are exactly the same. Moreover, for each pair of nodes, the similarity value of* $\mathbf{S}$ *is the value of* $\mathbf{S}'$ *multiplied by a constant factor* $(1-\lambda C_{in}-(1-\lambda)C_{out}) = 1 - 0.4 \times 0.6 - (1-0.4) \times 0.6 = 0.4$, *i.e.,* $\mathbf{S} \equiv 0.4 \times \mathbf{S}'$. □

Closed-Form of P-Rank. To represent the closed-form of P-Rank similarity $\mathbf{S}$, we first introduce the following lemma, which can guarantee the existence of our closed-form.

Lemma 1. *The matrices* $\mathbf{Q} \otimes \mathbf{Q}$ *and* $\mathbf{P} \otimes \mathbf{P}$ *are both row sub-stochastic matrices.* [4]

[2] Although in this case the diagonal entries of $\mathbf{S}$ may not equal 1, $\mathbf{S}$ still remains diagonally dominant, which ensures that "every vertex is maximally similar to itself".

[3] In what follows, we shall base our techniques on the P-Rank matrix form of Eq.(10).

[4] *A row sub-stochastic matrix* is a non-negative matrix with each row sum being no greater than 1.

Fig. 5. Example of Theorem 3

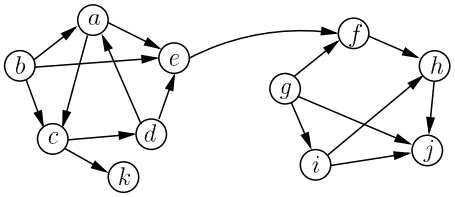


Fig. 6. A Fraction of the Social Network Crawled from Twitter

Proof. See Appendix A.3. □

Based on Lemma 1, we are now ready to obtain the closed-form solution of P-Rank.

Theorem 3. *Let* $\mathbf{M}$ *be an* $n^2 \times n^2$ *matrix defined by*

$$\mathbf{M} := \mathbf{I}_{n^2} - \lambda C_{in}(\mathbf{Q} \otimes \mathbf{Q}) - (1-\lambda)C_{out}(\mathbf{P} \otimes \mathbf{P}). \tag{11}$$

Then, the closed-form of P-Rank similarity $\mathbf{S}$ *can be represented as*

$$vec(\mathbf{S}) = (1 - \lambda C_{in} - (1-\lambda)C_{out}) \cdot \mathbf{M}^{-1} \cdot vec(\mathbf{I}_n). \tag{12}$$

Proof. See Appendix A.4 □

Theorem 3 provides the closed-form of P-Rank similarity, in which $\mathbf{S}$ does not appear in the RHS of Eq.(12) that relies only on the graph structure information in the matrix $\mathbf{M}$, as opposed to the recursive-form whose P-Rank similarity $\mathbf{S}$ appears on both sides of Eq.(9). The existence of $\mathbf{M}^{-1}$ is due to the fact that $\mathbf{M}$ is a diagonally dominant matrix, which is guaranteed by Lemma 1.

Example 7. *Consider the graph* $\mathcal{G}$ *in Figure 5. Given* $C_{in} = 0.6$, $C_{out} = 0.6$, *and* $\lambda = 0.4$, *the exact SimRank similarities can be obtained by Theorem 3 as follows:*

Since $\mathbf{Q} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right]$ *and* $\mathbf{P} = \left[\begin{smallmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \end{smallmatrix}\right]$, *the matrix* $\mathbf{M}$ *can be computed by Eq.*(11) *as*

$$\mathbf{M} = \mathbf{I}_4 - 0.4 \times 0.6 \times \left(\left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right] \otimes \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right]\right) - (1-0.4) \times 0.6 \times \left(\left[\begin{smallmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \end{smallmatrix}\right] \otimes \left[\begin{smallmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \end{smallmatrix}\right]\right)$$

Thus,

$$\mathbf{M} = \begin{bmatrix} 0.67 & -0.09 & -0.09 & -0.09 \\ -0.24 & 1 & 0 & 0 \\ -0.24 & 0 & 1 & 0 \\ -0.24 & 0 & 0 & 1 \end{bmatrix} \quad \Rightarrow \quad \mathbf{M}^{-1} = \begin{bmatrix} 1.652 & 0.149 & 0.149 & 0.149 \\ 0.397 & 1.036 & 0.036 & 0.036 \\ 0.397 & 0.036 & 1.036 & 0.036 \\ 0.397 & 0.036 & 0.036 & 1.036 \end{bmatrix}$$

Then, the P-Rank similarity can be derived from Eq.(12) *as follows:*

$$vec(\mathbf{S}) = (1 - 0.4 \times 0.6 - (1-0.4) \times 0.6) \cdot \mathbf{M}^{-1} \cdot vec(\mathbf{I}_2) = [0.720, 0.173, 0.173, 0.573]^T,$$

which implies that $\mathbf{S} = \left[\begin{smallmatrix} 0.720 & 0.173 \\ 0.173 & 0.573 \end{smallmatrix}\right]$. □

It is worth noticing that directly computing $\mathbf{S}$ from Eq.(12) involves the inverse of a large matrix $\mathbf{M}$ of size $n^2 \times n^2$, which is prohibitively expensive. However, the closed-form of P-Rank provides an *accurate non-iterative* method to represent the exact P-Rank similarity. It lays a solid foundation for us to (i) analyse the stability of P-Rank similarity $\mathbf{S}$ *w.r.t.* the perturbations of the graph structure information in matrix $\mathbf{M}$ (in Section 4.2), and (ii) devise efficient non-iterative algorithms to significantly speed up the computation of P-Rank (in Section 5).

4.2 Condition Number of P-Rank

Based on the closed-form of the P-Rank solution in Eq.(12), in this section we analyze the stability of P-Rank. We start with an intuitive example.

Example 8. *Consider the graph $\mathcal{G}$ in Figure 6, a fraction of the social network crawled from Twitter, where each node is a Twitter user, and each edge denotes that one user follows another. Given a query (user b), we want to rank all the users in $\mathcal{G}$ that are relevant to user b, by assessing P-Rank similarities $\{s(*, b)\}$ between each node and b.*

To evaluate the stability of P-Rank, we also construct a perturbed graph $\mathcal{G}^{(1)} := \mathcal{G} - \{c \to d\}$ (resp. $\mathcal{G}^{(2)} := \mathcal{G} - \{e \to f\}$) by randomly deleting an edge $c \to d$ (resp. $e \to f$) from $\mathcal{G}$. A deleted edge can be regarded as a missing "follow" relation via a web crawl, due to a mishap (e.g., IP address blocked, or URL connection error) when the crawl is running on Twitter. If P-Rank is stable, we would expect that the similarity ranks of $\{s(*, b)\}$ on the original $\mathcal{G}$ would have almost the same order as those on the perturbed $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$.

We fix weight factor $\lambda = 0.4$ and vary damping factors (C_{in}, C_{out}). The results for three different settings of (C_{in}, C_{out}) are shown in the following tables, respectively.

$(C_{in}, C_{out}) = (0.6, 0.5)$

Rank in $\mathcal{G}$	Node (User)	Rank in $\mathcal{G}^{(1)}$	Rank in $\mathcal{G}^{(2)}$
1	a	1	1
2	d	2	2
3	c	**4**	3
4	e	**3**	4
5	f	5	5
6	g	6	6
7	i	7	**8**
8	h	8	**7**
9	j	9	9
10	k	10	10
# of "flips"		2	2

$(C_{in}, C_{out}) = (0.8, 0.9)$

Rank in $\mathcal{G}$	Node (User)	Rank in $\mathcal{G}^{(1)}$	Rank in $\mathcal{G}^{(2)}$
1	d	**2**	**2**
2	a	**1**	**1**
3	c	**6**	3
4	e	**3**	4
5	f	5	5
6	g	**4**	6
7	i	7	**8**
8	h	8	**7**
9	j	9	9
10	k	10	10
# of "flips"		5	4

$(C_{in}, C_{out}) = (0.95, 0.95)$

Rank in $\mathcal{G}$	Node (User)	Rank in $\mathcal{G}^{(1)}$	Rank in $\mathcal{G}^{(2)}$
1	d	**2**	**2**
2	a	**1**	**1**
3	c	**6**	**3**
4	e	**3**	4
5	g	**4**	**6**
6	f	**5**	**5**
7	i	7	**8**
8	h	8	**7**
9	j	9	9
10	k	10	10
# of "flips"		6	6

Given (C_{in}, C_{out}), we can see the following: For each table, the first two columns report the rank of P-Rank similarities $\{s(x, b)\}_{x \in \mathcal{G}}$ on the original $\mathcal{G}$ (sorted in descending order) and the associated user x, whereas the last two columns report the similarity ranks on the perturbed graphs $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$, respectively. The last row counts the number of "flips" for the ranks on each perturbed graph w.r.t. the ranks on the original $\mathcal{G}$. For example, in the first table, the number of "flips" for the ranks on the perturbed $\mathcal{G}^{(1)}$ is 2 since two nodes c and e, which are ranked 3rd and 4th on the original $\mathcal{G}$, are ranked 4th and 3rd on the perturbed $\mathcal{G}^{(1)}$ (highlighted in bold).

We observe that, for small choice of (C_{in}, C_{out}), P-Rank's rankings exhibit a tiny flipping behavior, which makes P-Rank stable. When (C_{in}, C_{out}) become larger (e.g., (0.95,0.95)), the number of "flips" is increased (e.g., 6), which makes P-Rank less stable. Thus, different settings of damping factors will have an impact on the robustness of P-Rank. □

Example 8 indicates that (C_{in}, C_{out}) influences the stability of P-Rank. To investigate (i) how (C_{in}, C_{out}) influences the robustness of P-Rank and (ii) whether there are other factors that may affect the P-Rank stability, let us introduce the notion of *P-Rank condition number*.

Definition 1 (P-Rank condition number). *Given a graph $\mathcal{G}$, let $\mathbf{M}$ be an $n^2 \times n^2$ matrix defined by Eq.(11), i.e.,*

$$\mathbf{M} := \mathbf{I}_{n^2} - \lambda C_{in}(\mathbf{Q} \otimes \mathbf{Q}) - (1 - \lambda) C_{out}(\mathbf{P} \otimes \mathbf{P}).$$

Then, the quantity

$$\kappa_\infty(\mathcal{G}) \triangleq \|\mathbf{M}^{-1}\|_\infty \cdot \|\mathbf{M}\|_\infty \tag{13}$$

is called the P-Rank condition number of $\mathcal{G}$. Here, $\|\star\|_\infty$ denotes the ∞-norm that returns the maximum absolute row sum of the matrix.

To explain how the P-Rank condition number is related to the stability of P-Rank, let us revisit the closed-form of P-Rank solution in Eq.(12), which can be rewritten as

$$\mathbf{M} \cdot \mathrm{vec}(\mathbf{S}) = \xi \cdot \mathrm{vec}(\mathbf{I}_n) \quad \text{with} \quad \xi = 1 - \lambda C_{\text{in}} - (1-\lambda) C_{\text{out}}, \tag{14}$$

where the matrix $\mathbf{M}$ contains the graph structure information, and $\mathrm{vec}(\mathbf{S})$ contains all-pairs similarity values. Suppose there is a perturbation in the original graph, which leads to the changes $\Delta\mathbf{M}$ to old $\mathbf{M}$, and the similarity changes $\Delta\mathbf{S}$ to old $\mathbf{S}$. Then, in the new graph, it follows from Theorem 3 that

$$(\mathbf{M} + \Delta\mathbf{M}) \cdot \mathrm{vec}(\mathbf{S} + \Delta\mathbf{S}) = \xi \cdot \mathrm{vec}(\mathbf{I}_n). \tag{15}$$

Combining Eqs.(14) and (15) together produces

$$\mathrm{vec}(\Delta\mathbf{S}) = -\mathbf{M}^{-1} \cdot \Delta\mathbf{M} \cdot \mathrm{vec}(\mathbf{S} + \Delta\mathbf{S})$$

We take ∞-norm on both sides, which yields

$$\|\mathrm{vec}(\Delta\mathbf{S})\|_\infty \le \|\mathbf{M}^{-1}\|_\infty \cdot \|\Delta\mathbf{M}\|_\infty \cdot \|\mathrm{vec}(\mathbf{S} + \Delta\mathbf{S})\|_\infty$$

Rearrange the above terms and use the fact that $\|\mathrm{vec}(*)\|_\infty = \|*\|_{\max}$[5], which produces

$$\frac{\|\Delta\mathbf{S}\|_{\max}}{\|\mathbf{S} + \Delta\mathbf{S}\|_{\max}} \le \underbrace{\left(\|\mathbf{M}^{-1}\|_\infty \cdot \|\mathbf{M}\|_\infty\right)}_{\text{P-Rank condition number } \kappa_\infty} \cdot \frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty} \tag{16}$$

Note that the quantity $\frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty}$ is the relative change in the old $\mathbf{M}$ (which is induced by the perturbed graph structure), and the quantity $\frac{\|\Delta\mathbf{S}\|_{\max}}{\|\mathbf{S}+\Delta\mathbf{S}\|_{\max}}$ is the resulting relative change in the new P-Rank similarity matrix $(\mathbf{S} + \Delta\mathbf{S})$. The advantage of using relative changes is that they are dimensionless and will not be affected by overall scale factors.

It is important to notice that Eq.(16) indicates that the condition number $\kappa_\infty(\mathcal{G})$ is a relative error magnification factor. Changes in the RHS of Eq.(16) (related to perturbations in the graph structure) can cause changes $\kappa_\infty(\mathcal{G})$ times as large in the LHS of Eq.(16), *i.e.,* the P-Rank similarity values. Thus, the condition number $\kappa_\infty(\mathcal{G})$ defined by Eq.(14) can be utilized for analyzing P-Rank stability, *i.e.,* the sensitivity of P-Rank similarity in response to changes in the graph structure.

Intuitively, the P-Rank condition number can effectively measure how much the values of P-Rank similarities can change in response to a small perturbation in the graph structure. It can be noticed from Eq.(16) that

- When $\kappa_\infty(\mathcal{G})$ is small, a small change $\Delta\mathbf{M}$ in the link structure will not cause a large change $\Delta\mathbf{S}$ in the resulting P-Rank scores. In this case, P-Rank similarity rankings are stable.
- When $\kappa_\infty(\mathcal{G})$ is large, a small change $\Delta\mathbf{M}$ in the link structure may result in a large magnified error $\Delta\mathbf{S}$ in the resulting P-Rank scores, which makes P-Rank similarity rankings very sensitive to the perturbations of a graph and unstable.

Bounding P-Rank Condition Number. Despite the importance of $\kappa_\infty(\mathcal{G})$ in P-Rank stability analysis, the computation of $\kappa_\infty(\mathcal{G})$, if carried out naively by its definition Eq.(13), is rather expensive, as it involves the inverse of a very large matrix $\mathbf{M}$ of size $n^2 \times n^2$ first, and then computing its ∞-norm $\|\mathbf{M}^{-1}\|_\infty$, which is dominated by $O(n^6)$ time.

[5] $\|\mathbf{X}\|_{\max} = \max_{1 \le i,j \le n}\{x_{i,j}\}$ is a maximum element-wise matrix norm.

Motivated by this, we next propose an efficient lightweight method that can find the tight bound of $\kappa_\infty(\mathcal{G})$ in only $O(1)$ time, without the need to inverse the large matrix $\mathbf{M}$. Obviously, the lower bound of $\kappa_\infty(\mathcal{G})$ is always 1, which is due to the fact that

$$\kappa_\infty(\mathcal{G}) = \|\mathbf{M}^{-1}\|_\infty \cdot \|\mathbf{M}\|_\infty \geq \|\mathbf{M}^{-1} \cdot \mathbf{M}\|_\infty = \|\mathbf{I}_{n^2}\|_\infty = 1.$$

To find the upper bound of $\kappa_\infty(\mathcal{G})$, we now propose Lemmas 2 and 3 that can tightly bound $\|\mathbf{M}^{-1}\|_\infty$ and $\|\mathbf{M}\|_\infty$, respectively.

Lemma 2. *$\left\|\mathbf{M}^{-1}\right\|_\infty$ has the following upper bound:*

$$\left\|\mathbf{M}^{-1}\right\|_\infty \leq \frac{1}{1 - \lambda \cdot C_{in} - (1-\lambda) \cdot C_{out}}. \tag{17}$$

Proof. See Appendix A.5. □

Lemma 3. *$\|\mathbf{M}\|_\infty$ has the following upper bound:*

$$\|\mathbf{M}\|_\infty \leq 1 + \lambda \cdot C_{in} + (1-\lambda) \cdot C_{out}. \tag{18}$$

Proof. See Appendix A.6. □

Combining Lemmas 2 and 3, the following result is immediate.

Theorem 4. *The P-Rank condition number has the following tight bound:*

$$\kappa_\infty(\mathcal{G}) \leq \frac{1 + \lambda \cdot C_{in} + (1-\lambda) \cdot C_{out}}{1 - \lambda \cdot C_{in} - (1-\lambda) \cdot C_{out}}. \tag{19}$$

Theorem 4 gives a tight upper bound of $\kappa_\infty(\mathcal{G})$, which has several important applications:

(1) It can evaluate the robustness of P-Rank similarity in response to the perturbations in the graph structure (such as (noisy) uncertain graphs, or evolving graphs that involve frequent minor node/edge updates).
(2) It can measure the accuracy of P-Rank similarity rankings invoked by the roundoff errors in P-Rank iterative computations.
(3) It can help us set appropriate parameters (*e.g.,* damping factors (C_{in}, C_{out}) to improve the effectiveness of decentralised P-Rank computing on large scale graphs by removing a few number of inter-edges across different partitions, as will be demonstrated shortly.

Intuitively, to show how the upper bound of $\kappa_\infty(\mathcal{G})$ is associated with the P-Rank stability, let us apply Theorem 4 to Eq.(16):

$$\frac{\|\Delta\mathbf{S}\|_{\max}}{\|\mathbf{S} + \Delta\mathbf{S}\|_{\max}} \leq \frac{1 + \lambda \cdot C_{\text{in}} + (1-\lambda) \cdot C_{\text{out}}}{1 - \lambda \cdot C_{\text{in}} - (1-\lambda) \cdot C_{\text{out}}} \cdot \frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty}, \tag{20}$$

To guarantee stable P-Rank ranking results, we expect the RHS of Eq.(20) to be small so that relative similarity changes in $\mathbf{S}$ in the LHS of Eq.(20) can be bounded by small values.

Effects of C_{in} and C_{out}. It can be noticed that the quantity

$$\left(\frac{1 + \lambda \cdot C_{\text{in}} + (1-\lambda) \cdot C_{\text{out}}}{1 - \lambda \cdot C_{\text{in}} - (1-\lambda) \cdot C_{\text{out}}}\right) = \frac{2}{1 - \lambda \cdot C_{\text{in}} - (1-\lambda) \cdot C_{\text{out}}} - 1$$

decreases when $(C_{\text{in}}, C_{\text{out}})$ increases. Thus, small settings of C_{in} and C_{out} will make P-Rank stable. Conversely, increasing the values of C_{in} and C_{out} makes the RHS of Eq.(20) larger, which may cause P-Rank unstable. These results are well consistent with Example 8.

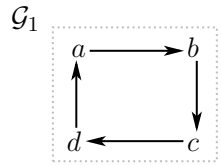


Fig. 7. The equality of Eq.(19) is attainable for $\mathcal{G}_1$

Effects of λ. The P-Rank condition number $\kappa_\infty(\mathcal{G})$ can also vary with the weight factor λ. To see this, let us compute the partial derivatives *w.r.t.* λ in Eq.(19):

$$\frac{\partial}{\partial\lambda}\left(\frac{1+\lambda\cdot C_{\text{in}}+(1-\lambda)\cdot C_{\text{out}}}{1-\lambda\cdot C_{\text{in}}-(1-\lambda)\cdot C_{\text{out}}}\right)=\frac{2\,(C_{\text{in}}-C_{\text{out}})}{(1-\lambda\cdot C_{\text{in}}-(1-\lambda)\cdot C_{\text{out}})^2}. \tag{21}$$

This implies that when $C_{\text{in}} > C_{\text{out}}$ (*resp.* $C_{\text{in}} < C_{\text{out}}$), for the increased λ, a small change in $\mathcal{G}$ may result in a large (*resp.* small) change in P-Rank, which makes P-Rank an *ill-conditioned* (*resp. a well-conditioned*) problem; when $C_{\text{in}} = C_{\text{out}}$, the value of $\kappa_\infty(\mathcal{G})$ is independent of λ.

It is worth noting that the upper bound of $\kappa_\infty(\mathcal{G})$ in Eq.(19) is attainable if and only if each vertex in network $\mathcal{G}$ has at least one in-degree and one out-degree because in this case each row sum and each column sum of $\mathbf{A}$ are *strictly* greater than 0, which ensures that $\mathbf{Q}$ and $\mathbf{P}$ are exactly row stochastic matrices[6] rather than sub-stochastic ones, and hence $\|\mathbf{Q}\otimes\mathbf{Q}\|_\infty = \|\mathbf{P}\otimes\mathbf{P}\|_\infty = 1$.

Example 9. *Consider a directed cycle network of length 4, depicted as $\mathcal{G}_1$ in Figure 7, in which each vertex has one in-link and one out-link. Setting $\lambda = 0.5$, $C_{in} = 0.8$, and $C_{out} = 0.6$, one can verify that the equality of Eq.*(19) *is attained for $\mathcal{G}_1$ as follows.*

On one hand, since $\mathbf{A} = \mathbf{Q} = \mathbf{P}$ for $\mathcal{G}_1$, $\mathbf{M}$ and $\mathbf{M}^{-1}$ can be solved naively from Eq.(11)*, which follows that*

$$\kappa_\infty(\mathcal{G}) = \|\mathbf{M}\|_\infty\cdot\|\mathbf{M}^{-1}\|_\infty = 1.7\times\frac{1}{0.3}=\frac{17}{3};$$

on the other, computing the right-hand side of (19) *produces*

$$\frac{1+\lambda\cdot C_{in}+(1-\lambda)\cdot C_{out}}{1-\lambda\cdot C_{in}-(1-\lambda)\cdot C_{out}}=\frac{1+0.5\times0.8+(1-0.5)\times0.6}{1-0.5\times0.8-(1-0.5)\times0.6}=\frac{17}{3}.$$

Both results are exactly the same, and hence the equality of Eq.(19) *holds.*

Application. (Decentralised Computing on Large Scale Graphs) An important application of P-Rank stability analysis is to improve the effectiveness for decentralised computing on large scale graphs. Many real graphs have a block-wise/community structure. Given a large graph $\mathcal{G}$, to efficiently evaluate P-Rank similarities on $\mathcal{G}$, the basic idea is to

(1) decompose $\mathcal{G}$ into several small components $\{\mathcal{G}_i\}_{i=1}^N$ via a graph partitioning method (*e.g.*, METIS[7]) such that the edges across different components are minimized;
(2) on each small component $\mathcal{G}_i$, evaluate the block P-Rank similarity matrix $\mathbf{S}_i$;
(3) merge all the block P-Rank similarities $\mathbf{S}_i$ on each $\mathcal{G}_i$ into one block diagonal matrix $\hat{\mathbf{S}} = diag(\mathbf{S}_1,\cdots,\mathbf{S}_N)$, which gives an approximation of the true similarity $\mathbf{S}$ on $\mathcal{G}$.

The main advantage of this approach is that Step (2) can be processed in parallel on each processor independently. However, how well the approximation $\hat{\mathbf{S}}$ fits the true similarity $\mathbf{S}$ depends on the P-Rank conditional number $\kappa_\infty(\mathcal{G})$ on $\mathcal{G}$. When $\kappa_\infty(\mathcal{G})$ is small, $\hat{\mathbf{S}}$ is a good approximation of $\mathbf{S}$.

[6] A matrix having row sums equal to 1 is called *a row stochastic matrix*.
[7] http://glaros.dtc.umn.edu/gkhome/metis/

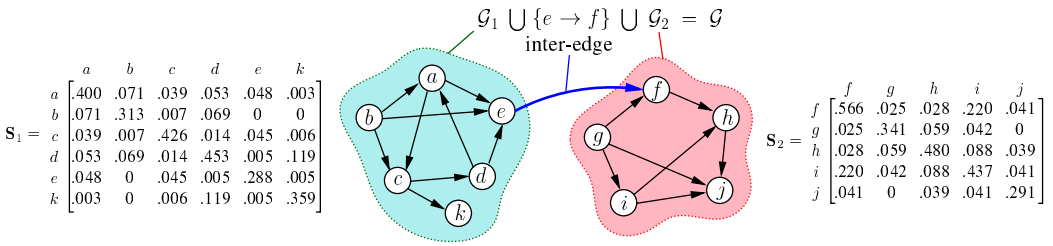


Fig. 8. Decentralised P-Rank Computing on $\mathcal{G}$

When $\kappa_\infty(\mathcal{G})$ is large, $\hat{\mathbf{S}}$ cannot approximate $\mathbf{S}$ well. The reason is that, based on our stability analysis, we can regard the union of all components $\hat{\mathcal{G}} := \bigcup_{i=1}^{N} \mathcal{G}_i$ as the original graph $\mathcal{G}$ perturbed by Δ that are all the inter-edges across different components:

$$\hat{\mathcal{G}} := \textstyle\bigcup_{i=1}^{N} \mathcal{G}_i = \mathcal{G} - \Delta$$

One can readily verify that $\hat{\mathbf{S}}$ is the exact P-Rank similarity on $\hat{\mathcal{G}}$. If P-Rank is stable, then $\hat{\mathcal{G}} \approx \mathcal{G}$ implies that $\hat{\mathbf{S}} \approx \mathbf{S}$. Otherwise, $\hat{\mathbf{S}}$, produced by the parallel algorithm, is not a good approximation of the true similarity $\mathbf{S}$. Thus, small settings of C_{in} and C_{out}, which will lead to small $\kappa_\infty(\mathcal{G})$, will make the parallel P-Rank algorithm produce accurate results. □

Example 10. *Given a graph $\mathcal{G}$ in Figure 6. We evaluate the P-Rank similarities on $\mathcal{G}$ in a decentralised way, as illustrated in Figure 8. We first utilize METIS to decompose $\mathcal{G}$ into two components $\mathcal{G}_1$ and $\mathcal{G}_2$, plus an inter-edge $\{e \to f\}$ across them. Then, the block P-Rank similarity matrix $\mathbf{S}_1$ on $\mathcal{G}_1$ and $\mathbf{S}_2$ on $\mathcal{G}_2$ can be evaluated in parallel independently.*

To make the result $\hat{\mathbf{S}} := \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix}$ a good approximation of the true P-Rank similarity $\mathbf{S}$ on $\mathcal{G}$, we need set the damping factors (C_{in}, C_{out}) small when evaluating $\mathbf{S}_1$ and $\mathbf{S}_2$. The following table illustrates how (C_{in}, C_{out}) affects the approximate error $\|\hat{\mathbf{S}} - \mathbf{S}\|_2$ on $\mathcal{G}$.

Damping Factors	(C_{in}, C_{out})	*(0.3,0.2)*	*(0.4, 0.3)*	*(0.5, 0.4)*	*(0.6, 0.5)*	*(0.7, 0.8)*
Average Approximate Error	$\frac{1}{n^2}\|\hat{\mathbf{S}} - \mathbf{S}\|_2$	*0.0010*	*0.0015*	*0.0019*	*0.0022*	*0.0027*
P-Rank Conditional Number	$\kappa_\infty(\mathcal{G})$	*1.625*	*2.006*	*2.500*	*3.159*	*5.743*
Upper Bound of $\kappa_\infty(\mathcal{G})$	$\frac{1+\lambda\cdot C_{in}+(1-\lambda)\cdot C_{out}}{1-\lambda\cdot C_{in}-(1-\lambda)\cdot C_{out}}$	*1.632*	*2.030*	*2.571*	*3.348*	*7.333*

We discern that, when (C_{in}, C_{out}) is small, $\|\hat{\mathbf{S}} - \mathbf{S}\|_2$ on $\mathcal{G}$ is also small. This is because small (C_{in}, C_{out}) leads to small P-Rank conditional number $\kappa_\infty(\mathcal{G})$ (i.e., P-Rank is stable), thus resulting in small approximate error $\|\hat{\mathbf{S}} - \mathbf{S}\|_2$ when $\mathcal{G}$ is perturbed to $\hat{\mathcal{G}}$. □

A Tradeoff between Iterative Accuracy and Stability. Based on the P-Rank stability analysis in Eq.(20), we notice that small choices of $(C_{\text{in}}, C_{\text{out}})$ make P-Rank stable. However, as shown in Theorem 1 (see Eq.(7) in Section 3), our error analysis on the P-Rank iterative method reveals that

$$|s(u,v) - s^{(k)}(u,v)| \le (\lambda C_{\text{in}} + (1-\lambda)C_{\text{out}})^{k+1}.$$

Since the right-hand side $(\lambda C_{\text{in}} + (1-\lambda)C_{\text{out}})^{k+1}$ is a decreasing function *w.r.t.* $(C_{\text{in}}, C_{\text{out}})$, it implies that, to guarantee the same accuracy, small choices of $(C_{\text{in}}, C_{\text{out}})$ requires more P-Rank iterations, thus leading to more computational time. Hence, if we use iterative methods for P-Rank similarity computation, there is a tradeoff between the P-Rank iterative accuracy and stability.

To resolve this problem, we continue to select small values of $(C_{\text{in}}, C_{\text{out}})$ to achieve good P-Rank stability, but devise a different (non-iterative) method for P-Rank computation that does not

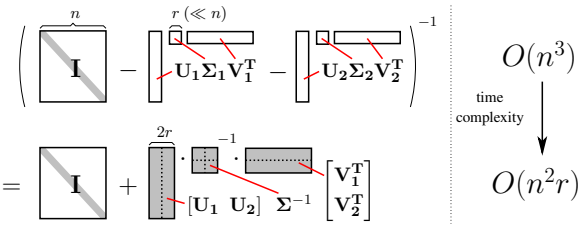


Fig. 9. Low-rank update of matrix inversion

produce any iterative errors relying on $(C_{\text{in}}, C_{\text{out}})$. In the next section, we propose efficient matrix-based algorithms to speed up P-Rank computations without yielding iterative errors. For these new matrix-based algorithms, small choices of $(C_{\text{in}}, C_{\text{out}})$ will not sacrifice speedup while making P-Rank stable.

5 P-RANK OPTIMIZATION TECHNIQUES

In this section, optimization techniques for accelerating the P-Rank computation are provided. (i) For directed networks, we propose novel techniques that can significantly reduce the computational time of P-Rank from $O(Kn^4)$ to $O(nr^2 + r^4)$ and the memory from $O(n^2)$ to $O(nr + r^4)$ with guaranteed accuracy, where $r(\leq n)$ is the target rank of low-rank approximation, in general $r \ll n$ (Subsection 5.1). As a by-product, we show that our P-Rank optimization approach, as a special case, can also substantially improve the prior SimRank computational time of Li *et al.* [23] from $O(r^4n^2 + nr^2)$ to $O(nr^2 + r^4)$, and the memory space from $O(n^2r^2)$ to $O(nr + r^4)$. (ii) For undirected networks, we also design a novel eigen-decomposition method that can improve the computation of P-Rank further to $O(nr^2)$ time and $O(nr)$ memory space (Subsection 5.2).

5.1 Optimizing P-Rank on Directed Networks

5.1.1 Exact P-Rank Similarity Optimization. In this subsection, our accelerative approach focuses on optimizing the *exact* P-Rank computation.

Warm-Up. Before detailing our techniques, we first introduce the following matrix inversion identity, which will be useful to our subsequent P-Rank optimization.

Lemma 4. *Let* $\mathbf{I}_n$ *be an* $n \times n$ *identity matrix,* $\mathbf{U}_i$ *and* $\mathbf{V}_i$ *be* $n \times r$ *matrices, and* $\mathbf{C}_i$ *be* $r \times r$ *matrices* $(i = 1, 2)$. *Then the following* matrix inversion identity *holds.*

$$\left(\mathbf{I}_n - \mathbf{U}_1\Sigma_1\mathbf{V}_1^T - \mathbf{U}_2\Sigma_2\mathbf{V}_2^T\right)^{-1} = \mathbf{I}_n + \left(\mathbf{U}_1\ \mathbf{U}_2\right)\Sigma^{-1}\begin{pmatrix}\mathbf{V}_1^T\\ \mathbf{V}_2^T\end{pmatrix} \tag{22}$$

$$\text{with } \Sigma = \left(\begin{array}{c|c} {\Sigma_1}^{-1} - \mathbf{V}_1^T\mathbf{U}_1 & -\mathbf{V}_1^T\mathbf{U}_2 \\ \hline -\mathbf{V}_2^T\mathbf{U}_1 & {\Sigma_2}^{-1} - \mathbf{V}_2^T\mathbf{U}_2 \end{array}\right)$$

Proof. See Appendix A.7. □

The advantage of Lemma 4 is that it can efficiently convert the inversion of an $n \times n$ matrix into the inversion of a small $r \times r$ matrix $(r \ll n)$, thus greatly improving the computational efficiency. Figure 9 pictorially visualizes the main idea of Lemma 4, which indicates that, when one wishes to inverse the $n \times n$ matrix $(\mathbf{I}_n - \mathbf{U}_1\Sigma_1\mathbf{V}_1^T - \mathbf{U}_2\Sigma_2\mathbf{V}_2^T)^{-1}$, it is only necessary to inverse the small $r \times r$ matrix Σ and compute $\mathbf{I}_n + \left(\mathbf{U}_1\ \mathbf{U}_2\right)\Sigma^{-1}(\mathbf{V}_1\ \mathbf{V}_2)^T$, which requires only $O(n^2r + r^2n + r^3)$ time in total, as opposed to the $O\left(n^3\right)$ time of the conventional matrix inversion.

Basic Idea for Characterising S. In light of Lemma 4, we next propose our techniques for optimizing P-Rank computation. Recall the closed form of the P-Rank equation that we derived in

Section 4.1:

$$\mathrm{vec}\,(\mathbf{S}) = (\mathbf{I}_{n^2} - \lambda C_{\text{in}}\,(\mathbf{Q}\otimes\mathbf{Q}) - (1-\lambda)\,C_{\text{out}}\,(\mathbf{P}\otimes\mathbf{P}))^{-1}\cdot \mathrm{vec}\,(\mathbf{I}_n) \tag{23}$$

As long as the matrices $\lambda C_{\text{in}}\,(\mathbf{Q}\otimes\mathbf{Q})$ and $(1-\lambda)\,C_{\text{out}}\,(\mathbf{P}\otimes\mathbf{P})$ can be decomposed as $\mathbf{U}_1\Sigma_1\mathbf{V}_1^T$ and $\mathbf{U}_2\Sigma_2\mathbf{V}_2^T$, respectively, Lemma 4 can be used to speed up the computation of $\mathbf{S}$. Fortunately, we notice that, when $\mathbf{Q}$ and $\mathbf{P}$ are respectively decomposed as

$$\mathbf{Q} = \mathbf{U}_\mathbf{Q}\Sigma_\mathbf{Q}\mathbf{V}_\mathbf{Q}^T \quad\text{and}\quad \mathbf{P} = \mathbf{U}_\mathbf{P}\Sigma_\mathbf{P}\mathbf{V}_\mathbf{P}^T,$$

$\mathbf{Q}\otimes\mathbf{Q}$ and $\mathbf{P}\otimes\mathbf{P}$ in Eq.(23) takes the following form:

$$\mathbf{Q}\otimes\mathbf{Q} = \tilde{\mathbf{U}}_\mathbf{Q}\tilde{\Sigma}_\mathbf{Q}\tilde{\mathbf{V}}_\mathbf{Q}^T \quad\text{and}\quad \mathbf{P}\otimes\mathbf{P} = \tilde{\mathbf{U}}_\mathbf{P}\tilde{\Sigma}_\mathbf{P}\tilde{\mathbf{V}}_\mathbf{P}^T, \quad\text{where}$$

$$\begin{aligned}
&\tilde{\mathbf{U}}_\mathbf{Q} = \mathbf{U}_\mathbf{Q}\otimes\mathbf{U}_\mathbf{Q}, && \tilde{\Sigma}_\mathbf{Q} = \Sigma_\mathbf{Q}\otimes\Sigma_\mathbf{Q}, && \tilde{\mathbf{V}}_\mathbf{Q} = \mathbf{V}_\mathbf{Q}\otimes\mathbf{V}_\mathbf{Q},\\
&\tilde{\mathbf{U}}_\mathbf{P} = \mathbf{U}_\mathbf{P}\otimes\mathbf{U}_\mathbf{P}, && \tilde{\Sigma}_\mathbf{P} = \Sigma_\mathbf{P}\otimes\Sigma_\mathbf{P}, && \tilde{\mathbf{V}}_\mathbf{P} = \mathbf{V}_\mathbf{P}\otimes\mathbf{V}_\mathbf{P}.
\end{aligned}$$

Substituting these into Eq.(23) and applying Lemma 4 produce

$$\begin{aligned}
\mathrm{vec}\,(\mathbf{S}) &= \underbrace{(\mathbf{I}_{n^2} - \lambda C_{\text{in}}\tilde{\mathbf{U}}_\mathbf{Q}\tilde{\Sigma}_\mathbf{Q}\tilde{\mathbf{V}}_\mathbf{Q}^T - (1-\lambda)\,C_{\text{out}}\tilde{\mathbf{U}}_\mathbf{P}\tilde{\Sigma}_\mathbf{P}\tilde{\mathbf{V}}_\mathbf{P}^T)^{-1}}_{=\{\text{using Lemma 4}\}}\cdot\mathrm{vec}\,(\mathbf{I}_n)\\
&= \mathrm{vec}\,(\mathbf{I}_n) + \left(\tilde{\mathbf{U}}_\mathbf{Q}\ \tilde{\mathbf{U}}_\mathbf{P}\right)\mathbf{\Lambda}\left(\tilde{\mathbf{V}}_\mathbf{Q}\ \tilde{\mathbf{V}}_\mathbf{P}\right)^T\mathrm{vec}\,(\mathbf{I}_n),
\end{aligned} \tag{24}$$

where

$$\mathbf{\Lambda} = \left(\begin{array}{c|c} \frac{1}{\lambda C_{\text{in}}}\tilde{\Sigma}_\mathbf{Q}^{-1} - \tilde{\mathbf{V}}_\mathbf{Q}^T\tilde{\mathbf{U}}_\mathbf{Q} & -\tilde{\mathbf{V}}_\mathbf{Q}^T\tilde{\mathbf{U}}_\mathbf{P} \\ \hline -\tilde{\mathbf{V}}_\mathbf{P}^T\tilde{\mathbf{U}}_\mathbf{Q} & \frac{1}{(1-\lambda)C_{\text{out}}}\tilde{\Sigma}_\mathbf{P}^{-1} - \tilde{\mathbf{V}}_\mathbf{P}^T\tilde{\mathbf{U}}_\mathbf{P} \end{array}\right)^{-1}. \tag{25}$$

Directly using Eqs.(24) and (25) to compute $\mathbf{S}$ requires $O(r^4n^2)$ time and $O(r^2n^2)$ memory. This complexity is dominated by computing $\tilde{\mathbf{V}}_\mathbf{Q}^T\tilde{\mathbf{U}}_\mathbf{Q}$, $\tilde{\mathbf{V}}_\mathbf{Q}^T\tilde{\mathbf{U}}_\mathbf{P}$, $\tilde{\mathbf{V}}_\mathbf{P}^T\tilde{\mathbf{U}}_\mathbf{Q}$, $\tilde{\mathbf{V}}_\mathbf{P}^T\tilde{\mathbf{U}}_\mathbf{P}$ in Eq.(25). For example, when we compute $\tilde{\mathbf{V}}_\mathbf{Q}^T\tilde{\mathbf{U}}_\mathbf{Q} := (\mathbf{V}_\mathbf{Q}\otimes\mathbf{V}_\mathbf{Q})^T(\mathbf{U}_\mathbf{Q}\otimes\mathbf{U}_\mathbf{Q})$, the sizes of $\mathbf{V}_\mathbf{Q}$ and $\mathbf{U}_\mathbf{Q}$ are both $n\times r$, which implies that the sizes of $(\mathbf{V}_\mathbf{Q}\otimes\mathbf{V}_\mathbf{Q})$ and $(\mathbf{U}_\mathbf{Q}\otimes\mathbf{U}_\mathbf{Q})$ are both $n^2\times r^2$. Consequently, multiplying $(\mathbf{V}_\mathbf{Q}\otimes\mathbf{V}_\mathbf{Q})^T$ and $(\mathbf{U}_\mathbf{Q}\otimes\mathbf{U}_\mathbf{Q})$ requires $O(r^4n^2)$ time and $O(r^2n^2)$ memory.

Speeding Up P-Rank Computation. There are great opportunities to significantly reduce the computational cost. Our optimization techniques are based on the following two observations:

Observation 1. *$(\mathbf{V}\otimes\mathbf{V})^T(\mathbf{U}\otimes\mathbf{U})$ can be efficiently computed as follows:*

$$(\mathbf{V}\otimes\mathbf{V})^T(\mathbf{U}\otimes\mathbf{U}) = \mathbf{\Theta}\otimes\mathbf{\Theta} \qquad \textit{with} \quad \mathbf{\Theta} = \mathbf{V}^T\mathbf{U}. \tag{26}$$

This can substantially reduce the computational time from $O(r^4n^2)$ to $O(r^2n + r^4)$, and the memory usage from $O(r^2n^2)$ to $O(rn + r^4)$, where $r \ll n$.

Proof. See Appendix A.8. □

It is important to note that, in Eq.(26), computing $\mathbf{\Theta} = \mathbf{V}^T\mathbf{U}$ just requires $O(r^2n)$ time and $O(rn)$ memory space, and the size of the resulting $\mathbf{\Theta}$ is $r\times r$. Thus, $\mathbf{\Theta}\otimes\mathbf{\Theta}$ entails $O(r^2\times r^2)$ time and $O(r^4)$ memory space. In addition, the matrix $\mathbf{\Theta}$ in Eq.(26) needs computing only once, and can be subsequently reused to compute $\mathbf{\Theta}\otimes\mathbf{\Theta}$. Hence, the total cost of computing Eq.(26) just requires $O(r^2n+r^4)$ time and $O(rn+r^4)$ memory space, which is a significant improvement over the existing method via Eq.(25) which requires $O(r^4n^2)$ time and $O(r^2n^2)$ memory space to compute $\tilde{\mathbf{V}}_\mathbf{Q}^T\tilde{\mathbf{U}}_\mathbf{Q}$.

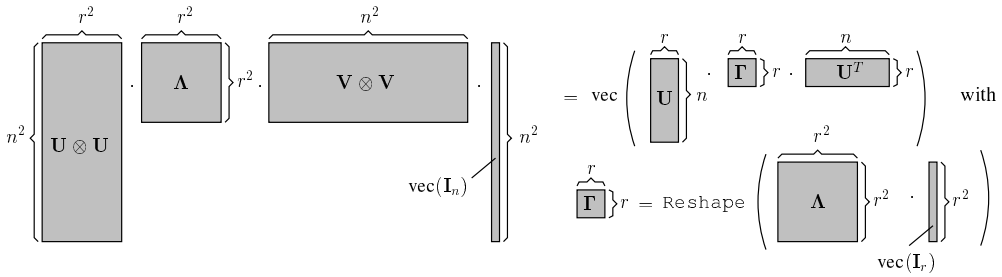


Fig. 10. Comparison of Li *et al.*'s method (in LHS) and our method (in RHS) to compute Eq.(27)

Example 11. *Let* $\mathbf{V}=[1,2,3]^T$ *and* $\mathbf{U}=[2,1,0]^T$. $(\mathbf{V}\otimes\mathbf{V})^T(\mathbf{U}\otimes\mathbf{U})$*, if carried out directly, involves taking two Kroneckor products and storing the intermediate results with two large* 9×1 *matrices:*

$$(\mathbf{V}\otimes\mathbf{V})^T=[1,2,3,2,4,6,3,6,9],\quad \mathbf{U}\otimes\mathbf{U}=[4,2,0,2,1,0,0,0,0]^T \quad\Rightarrow\quad (\mathbf{V}\otimes\mathbf{V})^T(\mathbf{U}\otimes\mathbf{U})=16$$

In contrast, our method only needs to compute $\mathbf{V}^T\mathbf{U}$*, whose result* $\mathbf{\Theta}$ *is a small* 1×1 *scalar. Due to small size of* $\mathbf{\Theta}$*, the cost of computing* $\mathbf{\Theta}\otimes\mathbf{\Theta}$ *is significantly reduced, i.e.,*

$$\mathbf{\Theta}:=\mathbf{V}^T\mathbf{U}=[1,2,3]\cdot[2,1,0]^T=[4]\quad\Rightarrow\quad\mathbf{\Theta}\otimes\mathbf{\Theta}=[4]\otimes[4]=16 \qquad \square$$

Another observation to speed up the computation of Eq.(24) is that there is no need to compute the tensor product in $(\tilde{\mathbf{V}}_\mathbf{Q}|\tilde{\mathbf{V}}_\mathbf{P})^T:=\left(\mathbf{V}_\mathbf{Q}\otimes\mathbf{V}_\mathbf{Q}|\mathbf{V}_\mathbf{P}\otimes\mathbf{V}_\mathbf{P}\right)^T$. The reason is as follows.

Observation 2. *Let* $\mathbf{X}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ *be a rank-r SVD decomposition of an* $n\times n$ *matrix* $\mathbf{X}$*, where* $\mathbf{U}$ *and* $\mathbf{V}$ *are* $n\times r$ *matrices, and* $\mathbf{\Sigma}$ *is an* $r\times r$ *matrix. For any* $r^2\times r^2$ *matrix* $\mathbf{\Lambda}$*, the following identity holds:*

$$(\mathbf{U}\otimes\mathbf{U})\mathbf{\Lambda}(\mathbf{V}\otimes\mathbf{V})^T vec(\mathbf{I}_n)=vec(\mathbf{U}\cdot\mathsf{Reshape}(\mathbf{\Lambda}vec(\mathbf{I}_r))\cdot\mathbf{U}^T) \tag{27}$$

where $\mathsf{Reshape}(\mathbf{x})$ *returns a* $r\times r$ *matrix whose entries are taken from the* $r^2\times 1$ *vector* $\mathbf{x}$*.*

Proof. See Appendix A.9. $\square$

By virtue of Observation 2, the computational time of Eq.(27) can be substantially improved from $O(2r^2n^2+r^4)$ to $O(rn^2+r^4)$, and the memory space from $O(r^2n^2)$ to $O(rn+r^4)$. Figure 10 pictorially visualizes the main idea underpinning our approach. It can be discerned that the traditional method to compute Eq.(27) (in the LHS) requires $O(2r^2n^2+r^4)$ time and $O(r^2n^2)$ memory space, consisting of three phases:

(i) $O(r^2n^2)$ time and $O(r^2n^2)$ memory to compute $\mathbf{x}\leftarrow(\mathbf{V}\otimes\mathbf{V})\mathrm{vec}(\mathbf{I}_n)$,
(ii) $O(r^4)$ time and $O(r^4)$ memory to compute $\mathbf{y}\leftarrow\mathbf{\Lambda}\mathbf{x}$, where $\mathbf{y}$ is of size $r^2\times 1$, and
(iii) $O(r^2n^2)$ time and $O(r^2n^2)$ memory to compute $(\mathbf{U}\otimes\mathbf{U})\mathbf{y}$.

In contrast, our method to compute Eq.(27) (in the RHS) includes:

(i) $O(r^4)$ time and $O(r^4)$ memory to compute $\mathbf{x}\leftarrow\mathbf{\Lambda}\mathrm{vec}(\mathbf{I}_r)$, where $\mathbf{x}$ is of size $r^2\times 1$;
(ii) $O(rn^2)$ time and $O(rn)$ memory to compute $\mathbf{Y}\leftarrow\mathbf{U}\cdot\mathsf{Reshape}(\mathbf{x})\cdot\mathbf{U}^T$.
Note that, although the resulting matrix $\mathbf{Y}$ is of size $n\times n$, we do not need $O(n^2)$ memory. Instead, we can compute $\mathbf{Y}$ column by column online, which requires only $O(rn)$ memory to store $\mathbf{U}$ and $\mathsf{Reshape}(\mathbf{x})$.

Thus, the total computational cost of our method entails $O(rn^2+r^4)$ time and $O(rn+r^4)$ memory, much superior to the traditional one.

Example 12. *Given* $\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ *and* $\mathbf{\Lambda} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 4 & 5 & 6 \\ 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}$. *Let* $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ *be a rank-2 SVD decomposition of* $\mathbf{X}$, *where* $\mathbf{U} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix}^T$, $\mathbf{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$, $\mathbf{V} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$.

The naive method to compute $(\mathbf{U} \otimes \mathbf{U})\mathbf{\Lambda}(\mathbf{V} \otimes \mathbf{V})^T vec(\mathbf{I}_3)$ *is rather expensive as it involves the computation of two large Kroneckor products as follows:*

$$(\mathbf{U} \otimes \mathbf{U})\mathbf{\Lambda}(\mathbf{V} \otimes \mathbf{V})^T vec(\mathbf{I}_3) = \underbrace{\begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T}_{\mathbf{U}\otimes\mathbf{U}} \underbrace{\begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 4 & 5 & 6 \\ 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}}_{\mathbf{\Lambda}} \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{(\mathbf{V}\otimes\mathbf{V})^T} vec(\mathbf{I}_3)$$

$$= [\tfrac{3}{2}, \tfrac{9}{\sqrt{2}}, \tfrac{3}{2}, \tfrac{1}{\sqrt{2}}, 4, \tfrac{1}{\sqrt{2}}, \tfrac{3}{2}, \tfrac{9}{\sqrt{2}}, \tfrac{3}{2}]^T.$$

In contrast, our method first evaluates $\mathbf{\Theta} := \mathtt{Reshape}(\mathbf{\Lambda} vec(\mathbf{I}_2)) = \mathtt{Reshape}([3, 9, 1, 4]^T) = \begin{bmatrix} 3 & 1 \\ 9 & 4 \end{bmatrix}$, *and then computes*

$$vec(\mathbf{U} \cdot \mathbf{\Theta} \cdot \mathbf{U}^T) = vec\left(\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix}^T \begin{bmatrix} 3 & 1 \\ 9 & 4 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix} \right) = [\tfrac{3}{2}, \tfrac{9}{\sqrt{2}}, \tfrac{3}{2}, \tfrac{1}{\sqrt{2}}, 4, \tfrac{1}{\sqrt{2}}, \tfrac{3}{2}, \tfrac{9}{\sqrt{2}}, \tfrac{3}{2}]^T$$

which is much faster and more memory-efficient than the naive approach. □

By virtue of the above two observations, we can significantly speed up the computation of P-Rank similarity $\mathbf{S}$ in Eqs.(24) and (25), based on the following theorem.

Theorem 5. *The P-Rank matrix* $\mathbf{S}$ *in Eq.*(24) *can be efficiently computed as*

$$\mathbf{S} = \mathbf{I}_n + \mathbf{U}_\mathbf{Q} \cdot \mathtt{Reshape}\Big((\mathbf{\Lambda}_{1,1} + \mathbf{\Lambda}_{1,2}) vec(\mathbf{I}_r)\Big) \cdot {\mathbf{U}_\mathbf{Q}}^T + \mathbf{U}_\mathbf{P} \cdot \mathtt{Reshape}\Big((\mathbf{\Lambda}_{2,1} + \mathbf{\Lambda}_{2,2}) vec(\mathbf{I}_r)\Big) \cdot {\mathbf{U}_\mathbf{P}}^T,$$

where

$$\mathbf{\Lambda} = \left(\begin{array}{c|c} \mathbf{\Lambda}_{1,1} & \mathbf{\Lambda}_{1,2} \\ \hline \mathbf{\Lambda}_{2,1} & \mathbf{\Lambda}_{2,2} \end{array}\right) = \left(\begin{array}{c|c} \frac{1}{\lambda C_{in}}(\mathbf{\Sigma}_\mathbf{Q}^{-1} \otimes \mathbf{\Sigma}_\mathbf{Q}^{-1}) - \mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}} \otimes \mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}} & -\mathbf{\Theta}_{\mathbf{Q},\mathbf{P}} \otimes \mathbf{\Theta}_{\mathbf{Q},\mathbf{P}} \\ \hline -\mathbf{\Theta}_{\mathbf{P},\mathbf{Q}} \otimes \mathbf{\Theta}_{\mathbf{P},\mathbf{Q}} & \frac{1}{(1-\lambda)C_{out}}(\mathbf{\Sigma}_\mathbf{P}^{-1} \otimes \mathbf{\Sigma}_\mathbf{P}^{-1}) - \mathbf{\Theta}_{\mathbf{P},\mathbf{P}} \otimes \mathbf{\Theta}_{\mathbf{P},\mathbf{P}} \end{array}\right)^{-1}$$

and $\mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}} = \mathbf{V}_\mathbf{Q}^T \mathbf{U}_\mathbf{Q}$, $\mathbf{\Theta}_{\mathbf{Q},\mathbf{P}} = \mathbf{V}_\mathbf{Q}^T \mathbf{U}_\mathbf{P}$, $\mathbf{\Theta}_{\mathbf{P},\mathbf{Q}} = \mathbf{V}_\mathbf{P}^T \mathbf{U}_\mathbf{Q}$, $\mathbf{\Theta}_{\mathbf{P},\mathbf{P}} = \mathbf{V}_\mathbf{P}^T \mathbf{U}_\mathbf{P}$.

Proof. See Appendix A.10. □

Theorem 5 provides an efficient method of computing P-Rank similarities, as pictorially depicted in Figure 11. Two illustrative examples will be provided shortly in Figure 13 after our algorithm is introduced.

Complexity. Regarding computational complexity, we have the following theorem.

Theorem 6. *The exact P-Rank similarity matrix* $\mathbf{S}$ *in Eq.*(24) *is solvable in only* $O(rn^2 + r^6)$ *time and* $O(rn + r^4)$ *memory space, without any loss of accuracy.*

Proof. See Appendix A.11. □

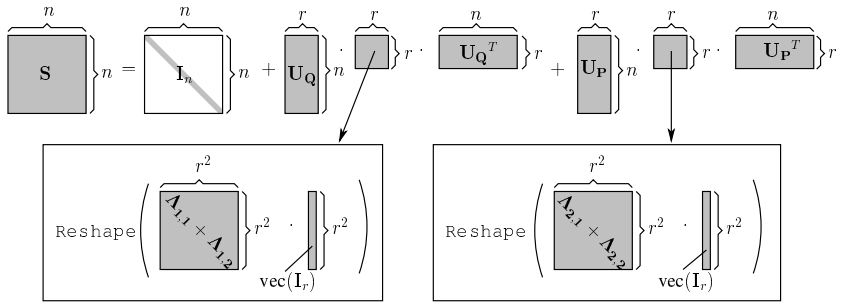


Fig. 11. Efficiently Computing P-Rank matrix S via Theorem 5

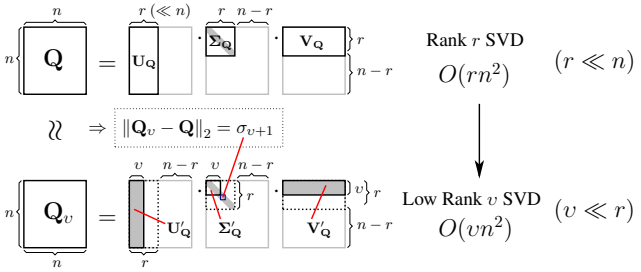


Fig. 12. Low rank υ approximation truncating the smallest $r - \upsilon$ singular values of $\mathbf{Q}$

5.1.2 Low Rank-υ ($\upsilon \leq r$) Approximation of P-Rank with Guaranteed Accuracy. In the previous subsection, we have considered the efficient *exact* P-Rank computing method by using the rank-r decomposition, where r is the rank of the adjacency matrix, *i.e.,* rank($\mathbf{Q}$ or $\mathbf{P}$). However, for many practical applications, users are more interested in *approximate* P-Rank scores if a little compromise in accuracy can substantially speed up the computation of P-Rank further.

Inspired by this, we next study the low rank-υ approximation of P-Rank (where $\upsilon \leq r$) with guaranteed accuracy. Specifically, we show the following main result in this subsection.

Theorem 7. *Let r be the rank of the graph adjacency matrix. Given a target rank υ ($\leq r$) for the SVD approximation of $\mathbf{Q}$ (or $\mathbf{P}$), the P-Rank similarities can be estimated in $O\left(\upsilon n^2 + \upsilon^6\right)$ time and $O\left(\upsilon n + \upsilon^4\right)$ memory space with the guaranteed error*

$$\epsilon_\upsilon \leq \frac{\lambda C_{in}\sigma_1\sigma_{\upsilon+1} + (1-\lambda)\, C_{out}\bar{\sigma}_1\bar{\sigma}_{\upsilon+1}}{1 - \lambda C_{in} - (1-\lambda)\, C_{out}}\sqrt{n},$$

where σ_i and $\bar{\sigma}_i$ $(i = 1, \upsilon+1)$ are the i-th largest singular values of $\mathbf{Q}$ and $\mathbf{P}$ respectively.

(The detailed proof will be provided after some discussions.)

The a-posteriori error ϵ_υ in Theorem 7 is often acceptable in practice due to the small values taken by the $(\upsilon+1)$-th largest singular values $\sigma_{\upsilon+1}$ and $\bar{\sigma}_{\upsilon+1}$. Figure 12 depicts the low-rank υ ($\leq r$) decomposition procedure that truncates the smallest $r - \upsilon$ almost zero singular values of the adjacency matrix. Indeed, the truncated dimensions (1) contain less important graph information for computing P-Rank similarities, but (2) require considerable amounts of computational costs. Thus, truncating such dimensions will enable a huge speedup of P-Rank computation, yet sacrifice only a little accuracy, as will be demonstrated by our experiments in Section 6.

Example 13. *On a Wiki graph with 1.2M vertices, by setting* $C_{\text{in}} = C_{\text{out}} = 0.8$ *and* $\lambda = 0.5$ *(as suggested in [54]), we obtain a high accuracy of*

$$\epsilon_\upsilon \leq \frac{0.5 \times 0.8 \times 1.12 \times 10^{-7} + 0.5 \times 0.8 \times 1.08 \times 10^{-7}}{1 - 0.5 \times 0.8 - 0.5 \times 0.8} \times \sqrt{1.2M} = 0.000482,$$

where $\sigma_1 = 1.12$ *and* $\bar{\sigma}_1 = 1.08$, *with* $\max(\sigma_{\upsilon+1}, \bar{\sigma}_{\upsilon+1}) \leq 10^{-7}$ *being truncated.*

The choice of the low rank υ ($\leq r$) has a user-controlled effect over the approximation error, which is a speed-accuracy trade-off. As a special case when $\upsilon = r$ ($\ll n$), we notice that $\sigma_{r+1} = \bar{\sigma}_{r+1} = 0$, and thereby $\epsilon_r = 0$. From this perspective, the approximate P-Rank in Theorem 7 is an extension of our exact approach in Theorem 6.

We next prove Theorem 7 by providing an algorithm for the low rank-υ P-Rank approximation.

Algorithm. The algorithm, referred to as DE P-Rank, is depicted in Algorithm 1. It accepts as inputs a directed graph $\mathcal{G}$, a weighted factor λ, two damping factors C_{in} and C_{out}, and a target rank υ (an optional parameter). If υ is omitted, the default value of υ is the rank r of adjacency matrix. DE P-Rank returns the approximate P-Rank similarity matrix $\mathbf{S}$ with an accuracy ϵ_υ if $\upsilon \leq r$; otherwise it returns the exact similarity $\mathbf{S}$ with $\epsilon_\upsilon = 0$.

Before describing the algorithm, we first introduce the following notations.

(i) RowNorm ($\mathbf{A}$) returns a row-stochastic matrix by normalizing each nonzero row of $\mathbf{A}$;
(ii) Rank ($\mathbf{A}$) returns the rank of $\mathbf{A}$;
(iii) RSVD ($\mathbf{Q}, \upsilon$) returns the low rank-υ SVD decomposition of $\mathbf{Q}$, *i.e.,* $\mathbf{Q}_\upsilon = \mathbf{U}_\mathbf{Q}\Sigma_\mathbf{Q}\mathbf{V}_\mathbf{Q}^T$ that minimizes $\|\mathbf{Q}_\upsilon - \mathbf{Q}\|_2 = \sigma_{\upsilon+1}$, where $\mathbf{U}_\mathbf{Q}$ and $\mathbf{V}_\mathbf{Q}$ are $n \times \upsilon$ column-orthogonal matrices, and $\Sigma_\mathbf{Q}$ is a $\upsilon \times \upsilon$ diagonal matrix.

Algorithm DE P-Rank works as follows. First, it initializes $\mathbf{Q}$ and $\mathbf{P}$ by normalizing each nonzero row of $\mathbf{A}$ and $\mathbf{A}^T$, respectively (lines 1–2). When the optional argument υ is not supplied, DE P-Rank also assigns a default rank Rank ($\mathbf{A}$) to υ (line 3). Then, DE P-Rank employs RSVD () to decompose the large $\mathbf{Q}$ (*resp.* $\mathbf{P}$) into small matrices, *i.e.,* $\mathbf{U}_\mathbf{Q}\Sigma_\mathbf{Q}\mathbf{V}_\mathbf{Q}^T$ (*resp.* $\mathbf{U}_\mathbf{P}\Sigma_\mathbf{P}\mathbf{V}_\mathbf{P}^T$). This results in a compact representation of a graph (line 4). In the case when $\upsilon <$ Rank ($\mathbf{A}$), RSVD () provides the best rank-υ SVD approximation of $\mathbf{Q}$ (*resp.* $\mathbf{P}$) in the least square error sense; otherwise, it yields the exact factorizations for $\mathbf{Q}$ and $\mathbf{P}$. Leveraging $\mathbf{U}_\mathbf{Q}, \Sigma_\mathbf{Q}, \mathbf{V}_\mathbf{Q}^T, \mathbf{U}_\mathbf{P}, \Sigma_\mathbf{P}, \mathbf{V}_\mathbf{P}^T$, DE P-Rank then computes P-Rank similarity matrix $\mathbf{S}$ (lines 5-8) via Theorem 5, and accuracy ϵ_υ (lines 9-10) via Theorem 7.

Example 14 (Heterogenous Graph). *Figure 13(a) depicts how DE P-Rank calculates similarity scores in a bipartite shopping graph* $\mathcal{G}_2$. $\mathcal{G}_2$ *is a heterogenous graph, whose vertices consist of two types of objects: (a) persons* $\{(A), (B)\}$, *and (b) an item set* $\{sugar, egg, flour\}$ *they purchased. We use the following parameters:* $C_{in} = 0.4$, $C_{out} = 0.6$, $\lambda = 0.5$.

DE P-Rank first computes $\mathbf{Q}$ *and* $\mathbf{P}$ *from the adjacency matrix* $\mathbf{A}$ *of* $\mathcal{G}_2$. *It then decomposes* $\mathbf{Q}$ *and* $\mathbf{P}$ *into* $\mathbf{U}_\mathbf{Q}\Sigma_\mathbf{Q}\mathbf{V}_\mathbf{Q}^\mathbf{T}$ *and* $\mathbf{U}_\mathbf{P}\Sigma_\mathbf{P}\mathbf{V}_\mathbf{P}^\mathbf{T}$, *respectively. Using these factorized small matrices, DE P-Rank calculates the block matrix* $\left(\begin{smallmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{smallmatrix}\right)$, *and returns* $\mathbf{S}$ *as the final P-Rank similarity matrix with no approximation errors, due to the default target rank* $\upsilon = \mathsf{Rank}(\mathbf{A}) = 1$ *it adopts.*

Example 15 (Homogeneous Graph). *DE P-Rank can also be applied to homogeneous domains, e.g., the web graph* $\mathcal{G}_3$ *in Figure 13(b). Each vertex in* $\mathcal{G}_3$ *corresponds to a web page, and each edge to a hyperlink from one page to another. Similarly, by setting* $C_{in} = 0.4$, $C_{out} = 0.6$, $\lambda = 0.5$, *a detailed process is given in Figure 13(b) which illustrates how DE P-Rank works on* $\mathcal{G}_3$. *Note that there are no approximation errors since we choose the default target rank* $\upsilon = \mathsf{Rank}(\mathbf{A}) = 2$.

To complete the proof of Theorem 7, we show that (1) DE P-Rank is bounded by $O\left(\upsilon n^2 + \upsilon^6\right)$ time and $O\left(\upsilon n + \upsilon^4\right)$ memory; and (2) it guarantees the accuracy bound stated in Theorem 7.

ALGORITHM 1: DE P-Rank $(\mathcal{G}, \lambda, C_{\text{in}}, C_{\text{out}}, \upsilon)$

Input : $\mathcal{G}$: a directed graph,
λ : weight factor,
$C_{\text{in}}, C_{\text{out}}$: in-/out-link damping factors,
υ: target rank of P-Rank approximation.

Output : S: P-Rank similarity matrix,
ϵ_υ: approximation error.

1 initialize the adjacency matrix $\mathbf{A}$ of $\mathcal{G}$.

2 compute the transition matrices $\mathbf{Q}$ and $\mathbf{P}$ in $\mathcal{G}$:
$\mathbf{Q} \leftarrow \mathtt{RowNorm}\left(\mathbf{A}^T\right)$, $\mathbf{P} \leftarrow \mathtt{RowNorm}\left(\mathbf{A}\right)$.

3 **if** υ *is empty* **then** $\upsilon \leftarrow \mathsf{Rank}\,(\mathbf{A})$

4 do low rank SVD approximation for $\mathbf{Q}$ and $\mathbf{P}$:
$\left[\mathbf{U_Q}, \Sigma_\mathbf{Q}, \mathbf{V_Q}; \sigma_1, \sigma_{\upsilon+1}\right] \leftarrow \mathsf{RSVD}\,(\mathbf{Q}, \upsilon)$,
$[\mathbf{U_P}, \Sigma_\mathbf{P}, \mathbf{V_P}; \bar{\sigma}_1, \bar{\sigma}_{\upsilon+1}] \leftarrow \mathsf{RSVD}\,(\mathbf{P}, \upsilon)$.

5 compute the auxiliary matrices :
$\mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}} \leftarrow \mathbf{V}_\mathbf{Q}^T\mathbf{U}_\mathbf{Q}$, $\mathbf{\Theta}_{\mathbf{Q},\mathbf{P}} \leftarrow \mathbf{V}_\mathbf{Q}^T\mathbf{U}_\mathbf{P}$,
$\mathbf{\Theta}_{\mathbf{P},\mathbf{Q}} \leftarrow \mathbf{V}_\mathbf{P}^T\mathbf{U}_\mathbf{Q}$, $\mathbf{\Theta}_{\mathbf{P},\mathbf{P}} \leftarrow \mathbf{V}_\mathbf{P}^T\mathbf{U}_\mathbf{P}$.

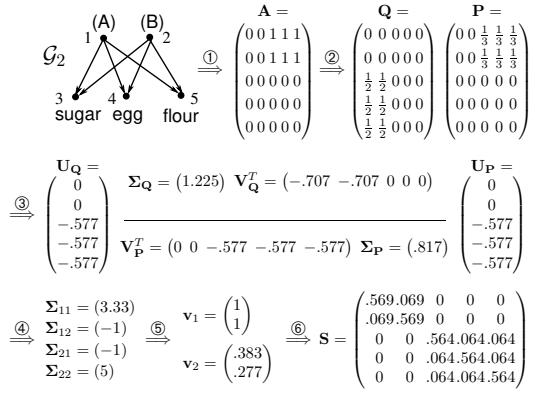
6 compute the matrix Σ :
$\Sigma_{11} \leftarrow \frac{1}{\lambda C_{\text{in}}}\Sigma_\mathbf{Q}^{-1} \otimes \Sigma_\mathbf{Q}^{-1} - \mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}} \otimes \mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}}$,
$\Sigma_{12} \leftarrow \mathbf{\Theta}_{\mathbf{Q},\mathbf{P}} \otimes \mathbf{\Theta}_{\mathbf{Q},\mathbf{P}}$,
$\Sigma_{22} \leftarrow \frac{1}{(1-\lambda)C_{\text{out}}}\Sigma_\mathbf{P}^{-1} \otimes \Sigma_\mathbf{P}^{-1} - \mathbf{\Theta}_{\mathbf{P},\mathbf{P}} \otimes \mathbf{\Theta}_{\mathbf{P},\mathbf{P}}$,
$\Sigma_{21} \leftarrow -\mathbf{\Theta}_{\mathbf{P},\mathbf{Q}} \otimes \mathbf{\Theta}_{\mathbf{P},\mathbf{Q}}$.

7 compute the matrix $\mathbf{\Lambda}$:
$\mathbf{\Lambda} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix} \leftarrow \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}^{-1}$

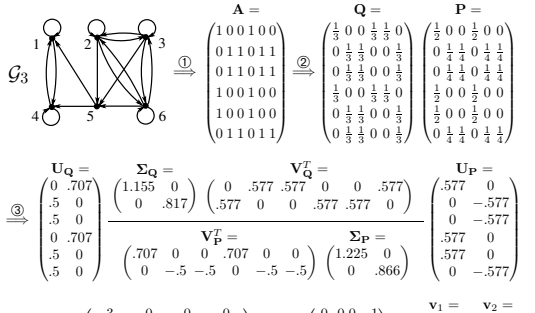
8 compute the P-Rank similarity matrix $\mathbf{S}$:
$\mathbf{\Gamma_Q} \leftarrow \mathtt{Reshape}\big((\mathbf{\Lambda_{1,1}} + \mathbf{\Lambda_{1,2}})\mathrm{vec}(\mathbf{I}_r)\big)$,
$\mathbf{\Gamma_P} \leftarrow \mathtt{Reshape}\big((\mathbf{\Lambda_{2,1}} + \mathbf{\Lambda_{2,2}})\mathrm{vec}(\mathbf{I}_r)\big)$,
$\beta \leftarrow (1 - \lambda C_{\text{in}} - (1-\lambda)C_{\text{out}})$,
$\mathbf{X_Q} \leftarrow \mathbf{U_Q}\mathbf{\Gamma_Q}$, $\mathbf{X_P} \leftarrow \mathbf{U_P}\mathbf{\Gamma_P}$,
$[\mathbf{S}]_{*,i} \leftarrow \beta([\mathbf{I}_n]_{*,i} + \mathbf{X_Q}[\mathbf{U_Q}]_{i,*}^T + \mathbf{X_P}[\mathbf{U_P}]_{i,*}^T)$. $\forall i$

9 **if** $\upsilon <$ Rank $(\mathbf{A})$ **then**
estimate accuracy
$\epsilon_\upsilon \leftarrow \frac{\lambda C_{\text{in}}\sigma_1\sigma_{\upsilon+1} + (1-\lambda)C_{\text{out}}\bar{\sigma}_1\bar{\sigma}_{\upsilon+1}}{1-\lambda C_{\text{in}} - (1-\lambda)C_{\text{out}}}\sqrt{n}$.
else
$\epsilon_\upsilon \leftarrow 0$.

10 **return** S and ϵ_υ.



(a) Heterogenous Shopping Graph $\mathcal{G}_2$



(b) Homogeneous Scientific Paper Network $\mathcal{G}_3$

Fig. 13. How DE P-Rank computes similarity

Correctness & Complexity. The correctness of DE P-Rank can be verified by Theorem 5. The computational cost DE P-Rank is bounded by $O\left(\upsilon n^2 + \upsilon^6\right)$ time and $O\left(\upsilon n + \upsilon^4\right)$ memory, which consists of three phases:

(i) For pre-processing (lines 1-4), (*a*) it takes $O\left(n^2\right)$ time and $O(m)$ memory to compute $\mathbf{Q}$ and $\mathbf{P}$ by normalizing $\mathbf{A}$ (lines 1 and 2); (*b*) calculating Rank (*) takes $O\left(n^2\right)$ time (line 3); (*c*) RSVD (*) is computed in $O\left(\upsilon n^2 + \upsilon^2 n\right)$ time and $O(\upsilon n)$ memory (line 4). In particular, the diagonal matrices $\Sigma_\mathbf{Q}$ and $\Sigma_\mathbf{P}$ can be stored in two υ-dimensional vectors with the entries σ_i and

$\bar{\sigma}_i$ $(i = 1, \cdots, v)$ sorted in descending order, respectively. Thus, the total cost of this phase is bounded by $O\left(vn^2 + v^2n\right)$ time and $O(vn)$ memory.

(ii) For the similarity computation phase (lines 5-8), we analyze the time complexity as follows: (*a*) Computing the 4 auxiliary matrices $\mathbf{\Theta}_{\mathrm{Q,Q}}, \mathbf{\Theta}_{\mathrm{P,Q}}, \mathbf{\Theta}_{\mathrm{Q,P}}, \mathbf{\Theta}_{\mathrm{P,P}}$ requires $O\left(v^2n\right)$ time and $O(vn)$ memory (line 5); (*b*) Computing all submatrices $\mathbf{\Sigma}_{i,j}$ $(i, j = 1, 2)$ of the matrix $\mathbf{\Sigma}$ entails $O(v^4)$ time and $O\left(v^4\right)$ memory (line 6); (*c*) The matrix $\mathbf{\Lambda}$ can be computed in $O(v^6)$ time and $O\left(v^4\right)$ memory (line 7); (*d*) $\mathbf{\Gamma}_{\mathrm{Q}}$ and $\mathbf{\Gamma}_{\mathrm{P}}$ can be obtained in $O(v^4)$ time and $O(v^4)$ memory, and $\mathbf{S}$ can be computed from $\mathbf{\Gamma}_{\mathrm{Q}}$ and $\mathbf{\Gamma}_{\mathrm{Q}}$ in $O(vn^2)$ time and $O(vn)$ memory column by column (line 8).

(iii) The last phase, error estimation, is in $O(1)$ time and $O(1)$ memory (line 9).

Taking (i)–(iii) together, DE P-Rank is bounded by $O\left(vn^2 + v^6\right)$ time and $O\left(vn + v^4\right)$ memory in total.

Error Bound. We show that, when $v < \mathsf{Rank}(\mathbf{A})$, DE P-Rank gives the approximation error:

$$\epsilon_v \leq \frac{\lambda C_{\mathrm{in}}\sigma_1\sigma_{v+1} + (1-\lambda)\, C_{\mathrm{out}}\bar{\sigma}_1\bar{\sigma}_{v+1}}{1 - \lambda C_{\mathrm{in}} - (1-\lambda)\, C_{\mathrm{out}}}\sqrt{n}. \tag{28}$$

To prove this error bound, let us first define

$$\mathbf{M}_v := \mathbf{I}_{n^2} - \lambda C_{\mathrm{in}}\left(\mathbf{Q}_v \otimes \mathbf{Q}_v\right) - (1-\lambda)\, C_{\mathrm{out}}\left(\mathbf{P}_v \otimes \mathbf{P}_v\right), \tag{29}$$

where $\mathbf{Q}_v$ and $\mathbf{P}_v$ are the rank-v SVD approximation of $\mathbf{Q}$ and $\mathbf{P}$, respectively (see Figure 12). We first show the following lemma.

Lemma 5. *Let* $\mathbf{M}$ *and* $\mathbf{M}_v$ *be the matrices defined by Eqs.*(11) *and* (29)*. Then the following inequality holds:*

$$\|\mathbf{M}_v - \mathbf{M}\|_\infty \leq \sqrt{n}\|\mathbf{M}_v - \mathbf{M}\|_2.$$

where σ_i *and* $\bar{\sigma}_i$ $(i = 1, v+1)$ *are the* i*-th largest singular values of* $\mathbf{Q}$ *and* $\mathbf{P}$*, respectively.*

Proof. See Appendix A.12. □

To find the bound of $\|\mathbf{M}_v - \mathbf{M}\|_2$, the following lemma is needed.

Lemma 6. *The following inequality holds:*

$$\|\mathbf{M}_v - \mathbf{M}\|_2 \leq \lambda C_{in}\sigma_1\sigma_{v+1} + (1-\lambda)\, C_{out}\bar{\sigma}_1\bar{\sigma}_{v+1}, \tag{30}$$

where σ_i *and* $\bar{\sigma}_i$ $(i = 1, v+1)$ *are the* i*-th largest singular values of* $\mathbf{Q}$ *and* $\mathbf{P}$*, respectively.*

Proof. See Appendix A.13. □

Capitalizing on Lemmas 5 and 6, we can obtain Eq.(28).

5.1.3 Tailoring DE P-Rank to SimRank Optimization. In this subsection, we show that our P-Rank optimization approach proposed in the above two subsections, as a special case, can also substantially improve the SimRank computation of Li *et al.*'s SVD-based method [23].

In matrix notations, Li *et al.*'s SimRank model [23] is formulated as follows:

$$\mathbf{S} = C_{\mathrm{in}} \cdot \mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T + (1 - C_{\mathrm{in}}) \cdot \mathbf{I}_n. \tag{31}$$

The existing Li *et al.*'s SVD-based method [23] requires $O(r^4n^2)$ time and $O(r^2n^2)$ memory space to compute SimRank, which is dominated by the expensive Kroneckor product of the SVD decomposed matrices. We observe that, when $\lambda = 1$, the P-Rank model of Eq.(9) reduces to Li *et al.*'s SimRank model. Thus, our P-Rank optimization approach proposed in the above two subsections also applies to SimRank.

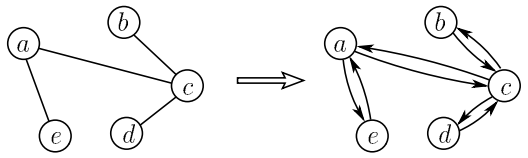


Fig. 14. Turning an Undirected Graph into a Directed Graph

Specifically, when tailoring DE P-Rank to SimRank by choosing $\lambda = 1$, we can get the following efficient SimRank algorithm, referred to as DE SR, which can significantly speed up Li *et al.*'s SimRank computation, as illustrated in Algorithm 2.

Correctness & Complexity. The correctness of DE SR is guaranteed by that of DE P-Rank. It can be shown that DE SR yields exactly the same result of Li *et al.*'s method, but is much faster.

Regarding computational cost, we have the following theorem:

Theorem 8. *The total computational cost of DE SR can be bounded by $O(n^2r + r^6)$ time and $O(rn + r^4)$ memory space.*

Proof. See Appendix A.14. □

ALGORITHM 2: DE SR $(\mathcal{G}, C_{\text{in}}, \upsilon)$

Input : $\mathcal{G}$: a directed graph, C_{in}: damping factor, υ: target rank of SVD.
Output: S: SimRank similarity matrix.

1 initialize the adjacency matrix $\mathbf{A}$ of $\mathcal{G}$.
2 compute the transition matrix $\mathbf{Q} \leftarrow \texttt{RowNorm}\left(\mathbf{A}^T\right)$.
3 **if** υ *is empty* **then**
  $\upsilon \leftarrow \text{Rank}(\mathbf{A})$
4 decompose the matrix $\mathbf{Q}$ via rank-υ SVD:
  $[\mathbf{U}, \Sigma, \mathbf{V}] \leftarrow \texttt{RSVD}(\mathbf{Q}, \upsilon)$.
5 memoize the intermediate matrices:
  $\text{inv}\Sigma \leftarrow \Sigma^{-1}, \qquad \mathbf{\Theta} \leftarrow \mathbf{V}^T\mathbf{U}, \qquad \mathbf{\Lambda} \leftarrow (\text{inv}\Sigma \otimes \text{inv}\Sigma - C_{\text{in}}(\mathbf{\Theta} \otimes \mathbf{\Theta}))^{-1}$
6 reshape the resulting $\upsilon^2 \times 1$ vector $(\mathbf{\Lambda} vec(\mathbf{I}_\upsilon))$ into the $\upsilon \times \upsilon$ matrix $\mathbf{\Gamma}$:
  $\mathbf{\Gamma} \leftarrow \texttt{Reshape}(\mathbf{\Lambda} vec(\mathbf{I}_\upsilon))$
7 compute the SimRank similarity matrix $\mathbf{S}$:
  memoize the intermediate matrix: $\mathbf{X} \leftarrow \mathbf{U} \cdot \mathbf{\Gamma}$
  return $[\mathrm{S}]_{*,i} \leftarrow (1 - C_{\text{in}}) \cdot ([\mathbf{I}_n]_{*,i} + C_{\text{in}} \cdot \mathbf{X} \cdot [\mathrm{U}]_{i,*}^T) \qquad \forall i = 1, 2, \cdots, n$

In contrast to the $O(r^4n^2 + r^6)$ time and $O(r^2n^2)$ memory of Li *et al.*'s method, our algorithm DE SR can efficiently reduce the computational cost to $O(rn^2 + r^6)$ time and $O(rn + r^4)$ memory space while guaranteeing the same accuracy of Li *et al.*'s algorithm.

5.2 Computing P-Rank on Undirected Graphs

Our P-Rank optimization techniques in Subsection 5.1 are designed for directed graphs. It is worth mentioning that P-Rank, designed to model in-links and out-links, is also suitable for undirected graphs. This is because we can always turn an undirected graph into a directed graph by replacing each undirected edge with two directed edges pointing in the opposite directions, as illustrated in Figure 14. Thus, all our P-Rank optimizations for directed graphs in Subsection 5.1 can be applied

to undirected graphs as well. However, for undirected graphs, there are more efficient techniques for computing P-Rank, as will be shown in this subsection.

It is worth mentioning that all our methods in this subsection for computing undirected P-Rank can also be applied to computing undirected SimRank with only a slight modification. This is because, for undirected P-Rank, $\mathcal{I}(x) = \mathcal{O}(x) = \mathcal{N}(x)$ for every node $x \in \mathcal{V}$, where $\mathcal{N}(x)$ denotes the neighbor set of node x in an undirected graph. Thus, on undirected graphs, the P-Rank definition in Eq.(3) becomes

$$s(u,v) = \big(\lambda \cdot C_{\text{in}} + (1-\lambda)\cdot C_{\text{out}}\big) \cdot \frac{\sum_{(i,j)\in\mathcal{N}(u)\times\mathcal{N}(v)} s(i,j)}{|\mathcal{N}(u)||\mathcal{N}(v)|}$$

In contrast, the SimRank definition on undirected graphs is

$$s(u,v) = C_{\text{in}} \cdot \frac{\sum_{(i,j)\in\mathcal{N}(u)\times\mathcal{N}(v)} s(i,j)}{|\mathcal{N}(u)||\mathcal{N}(v)|}$$

Hence, for undirected graphs, the difference between P-Rank and SimRank scores is up to a constant multiplicative factor.

The key idea underpinning our optimization is to diagonalize the adjacency matrix $\mathbf{A}$ into $\mathbf{\Lambda}$ and utilize $\mathbf{\Lambda}$ to compute the P-Rank similarity matrix $\mathbf{S}$ from its power series representation: $\mathbf{S} = \sum_{k=0}^{+\infty} f(\mathbf{\Lambda}^k)$. Due to undirected graphs, $\mathbf{A}$ is symmetric, and thereby diagonalizable to $\mathbf{\Lambda}$. Since the power of diagonal matrix $\mathbf{\Lambda}$ is much easier to compute than that of $\mathbf{A}$, the efficiency of P-Rank computation on undirected graphs can be substantially improved further.

The challenging problem of our method is the characterization of P-Rank similarity $\mathbf{S}$ in terms of the eigenvectors of an undirected graph. To address this issue, we show the following theorem.

Theorem 9. *Given an undirected graph, let $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{n\times n}$ be its adjacency matrix, and $\mathbf{D}$ be a diagonal matrix defined by* [8]

$$\mathbf{D} := diag\Big((\textstyle\sum_{j=1}^{n} a_{1,j})^{-1}, \cdots, (\sum_{j=1}^{n} a_{n,j})^{-1}\Big) \in \mathbb{R}^{n\times n}. \tag{32}$$

The P-Rank similarity matrix $\mathbf{S}$ can be represented as [9]

$$\mathbf{S} = (1 - \lambda\cdot C_{in} - (1-\lambda)\cdot C_{out}) \cdot \mathbf{D}^{1/2}\mathbf{U}\cdot\mathbf{\Psi}\cdot\mathbf{U}^T\mathbf{D}^{1/2}, \tag{33}$$

where $\mathbf{\Psi} = \big(\Psi_{i,j}\big)_{r\times r}$ whose (i,j)-entry is expressible as

$$\Psi_{i,j} = \frac{[\mathbf{U}^T\mathbf{D}^{-1}\mathbf{U}]_{i,j}}{1 - (\lambda\cdot C_{in} + (1-\lambda)\cdot C_{out})\,\Lambda_{i,i}\Lambda_{j,j}}. \tag{34}$$

Here, $\mathbf{U}$ and $\mathbf{\Lambda}$ are the eigen-decomposition of $\mathbf{T} := \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$ is an orthogonal matrix with its columns being all eigenvectors of $\mathbf{T}$, and $\mathbf{\Lambda} = (\Lambda_{i,j}) \in \mathbb{R}^{r\times r}$ is an diagonal matrix with its diagonal entries being all eigenvalues of $\mathbf{T}$.

Proof. See Appendix A.15. □

Figure 15 visualizes the efficient computation of P-Rank on undirected graphs by Theorem 9. In light of this, we next present an algorithm to compute P-Rank on undirected graphs.

Algorithm. The algorithm, referred to as UN P-Rank, is shown in Algorithm 3. It takes as input an undirected graph $\mathcal{G}$, a weighting factor λ, and in- and out-link damping factors C_{in} and C_{out}, and outputs the exact P-Rank similarity matrix $\mathbf{S}$ in $\mathcal{G}$.

[8] We define $\frac{1}{0} \triangleq 0$, thereby avoiding division by zero when the column/row sum of $\mathbf{A}$ equals 0.
[9] $\mathbf{D}^{1/2}$ is a diagonal matrix whose diagonal entries are the principal square root of those of $\mathbf{D}$.

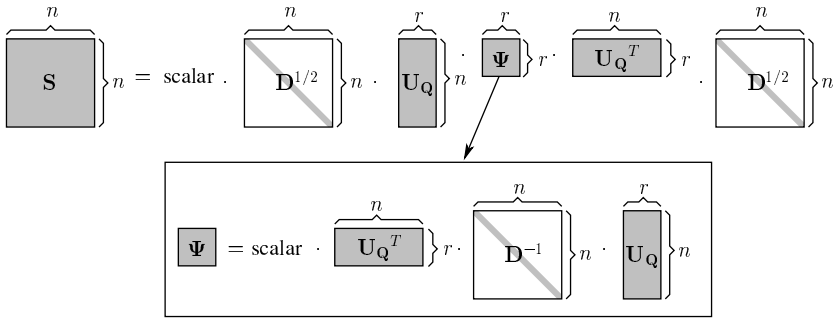


Fig. 15. Efficiently Computing P-Rank Similarity S on Undirected Graphs by Theorem 9

ALGORITHM 3: UN P-Rank $(\mathcal{G}, C_{\text{in}}, C_{\text{out}}, \lambda)$

Input : $\mathcal{G}$: an undirected graph, $C_{\text{in}}/C_{\text{out}}$: in- and out-link damping factors, λ: weight factor.
Output: S: P-Rank similarity matrix.

1 initialize the adjacency matrix $\mathbf{A}$ of $\mathcal{G}$.
2 compute the diagonal matrix $\mathbf{D} = diag(d_{1,1}, \cdots, d_{n,n})$:
for $i \leftarrow 1, 2, \cdots, n$ **do**
  initialize $deg(i) \leftarrow \sum_{j=1}^{n} a_{i,j}$
  if $deg(i) \neq 0$ **then**
    $d_{i,i} \leftarrow 1/deg(i)$
  else
    $d_{i,i} \leftarrow 0$
3 compute the auxiliary matrix $\mathbf{T} \leftarrow \mathbf{D}^{1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{1/2}$.
4 decompose $\mathbf{T} := \mathbf{U} \cdot \mathbf{\Lambda} \cdot \mathbf{U}^T$, where $\mathbf{\Lambda} = diag(\Lambda_{1,1}, \cdots, \Lambda_{r,r})$ is diagonal and $\mathbf{U}$ is orthogonal.
5 compute two auxiliary matrices and one scalar
  $\mathbf{\Gamma} = (\Gamma_{i,j})_{r\times r} \leftarrow \mathbf{U}^T \cdot \mathbf{D}^{-1} \cdot \mathbf{U}$, $\mathbf{V} \leftarrow \mathbf{D}^{1/2} \cdot \mathbf{U}$, $C \leftarrow \lambda C_{\text{in}} + (1-\lambda) C_{\text{out}}$.
6 compute the matrix $\mathbf{\Psi} = (\psi_{i,j})_{r\times r}$ whose entry $\psi_{i,j} \leftarrow \Gamma_{i,j}/(1 - C \cdot \Lambda_{i,i} \cdot \Lambda_{j,j})$.
7 **return** $[\mathrm{S}]_{*,i} \leftarrow (1-C) \cdot \mathbf{V} \cdot \mathbf{\Psi} \cdot [\mathbf{V}]_{i,*}^T \quad \forall i = 1, 2, \cdots, n$.

Algorithm UN P-Rank works as follows. It first initializes the adjacency matrix $\mathbf{A}$ (line 1). Utilizing $\mathbf{A}$, it then computes the auxiliary diagonal matrix $\mathbf{D}$ whose (i,i)-entry is the reciprocal of the i-th column sum of $\mathbf{A}$ if this reciprocal exists, and 0 otherwise (line 2). UN P-Rank then uses eigenvalue decomposition (EVD) [7] to factorize $\mathbf{T} := \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ as $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where all columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{T}$, and all diagonal entries of $\mathbf{\Lambda}$ are the eigenvalues of $\mathbf{T}$ (lines 3-4). Using $\mathbf{U}$ and $\mathbf{\Lambda}$, it computes $\mathbf{\Psi}$ (lines 5-6) to obtain the similarity matrix S (line 7), as justified by Eqs.(33) and (34).

Example 16. *Consider an undirected friendship graph $\mathcal{G}_4$. Each vertex corresponds to a person, and there is an edge between two people whenever they are friends. The detailed process of computing* S *is depicted in Figure 16 step by step. UN P-Rank returns* S *as the final similarities, which is the exact solution to Eq.*(9).

Complexity. The computational complexity of UN P-Rank is shown in the following theorem.

Theorem 10. *For an undirected graph, its P-Rank similarity matrix* S *is solvable in $O(rn^2)$ time and $O(rn)$ memory, without any loss of exactness.*

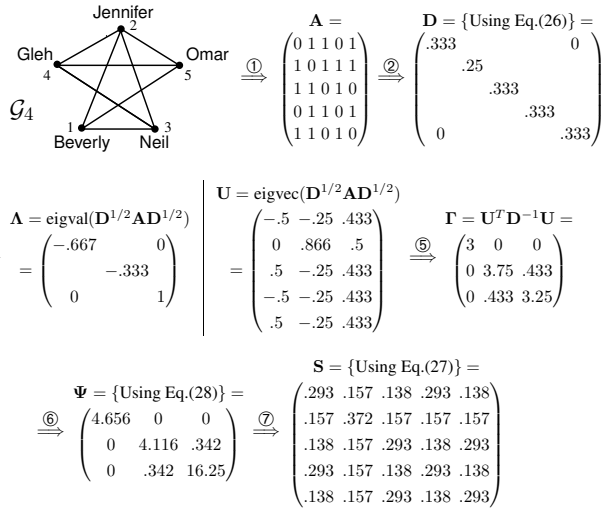


Fig. 16. How UN P-Rank works on the undirected graph $\mathcal{G}_4$

Proof. See Appendix A.16. □

Compared with the $O(rn^2 + r^6)$ time and $O(rn + r^4)$ memory on directed graphs, Theorem 10 significantly reduces the computation of P-Rank on undirected graphs. Moreover, Theorem 10 also contains the optimization of SimRank on undirected graphs as a special case. Indeed, when $\lambda = 1$ and $C_{\text{in}} = C_{\text{out}}$, Theorem 10 reduces to SimRank optimization, which suggests that SimRank scores on undirected graphs can be accurately computed in $O(rn^2)$ time and $O(rn)$ memory. This result, as opposed to the $O(n^3 + kn^2)$ time of the existing work [44], substantially improves the efficiency of SimRank on undirected graphs.

6 EXPERIMENTAL EVALUATION

In this section, a comprehensive empirical study of our P-Rank methods is presented. By using real and synthetic data, four sets of experiments are conducted to evaluate the accuracy, stability, computational efficiency in terms of running time and memory space of our proposed approaches.

6.1 Experimental Setting

(1) Real-life Data. We used various real-world datasets with different sizes:

	Datasets		m	n	Description
small	DBLP	98-99	5,929	1,525	DBLP 10-year (from 1998 to 2007) author-paper information, and the selected papers are published on 6 major conferences (ICDE, VLDB, SIGMOD, WWW, SIGIR and KDD)
		98-01	13,441	3,208	
		98-03	24,762	5,307	
		98-05	39,399	7,984	
		98-07	54,844	10,682	
medium	p2p-Gnutella		147,892	62,586	Gnutella peer to peer network
	email-EuAll		420,045	265,214	Email network from a EU research institution
large	web-Stanford		2,312,497	281,903	Web graph of *Stanford.edu*
	Amazon		3,200,440	400,727	Amazon product co-purchasing network

- DBLP. The DBLP datasets were taken from the DBLP computer science bibliography.[10] We extracted the 10-year (from 1998 to 2007) author-paper information and picked up papers published by 6 major conferences in database and information management ("ICDE", "VLDB", "SIGMOD", "WWW", "SIGIR" and "KDD"). Every two years made a time step. For each time step, we built a co-authorship network incrementally from the one of previous time step. For every DBLP network, a node represents an author. We chose the relationship that there is an edge between authors if one author wrote a paper with another.
- p2p-Gnutella. The p2p-Gnutella network is a snapshot of the Gnutella peer-to-peer file sharing network collected by SNAP.[11] Every node denotes a host in the Gnutella network topology, and each edge represents a connection between two Gnutella hosts.
- email-EuAll. The email-EuAll network is an email graph generated from a large European research institution. Given a set of email messages, each node denotes an email address. We created an edge from node i to j, if i sent at least one message to j.
- web-Stanford. The web-Stanford network is a Stanford web graph taken from Stanford University (*stanford.edu*). In this network, nodes represent web pages from Stanford University, and directed edges represent hyperlinks from one web page to another. The dataset was collected by SNAP in 2002.
- Amazon. The Amazon network is a product co-purchasing graph collected by crawling the Amazon website. It is based on *Customers Who Bought This Item Also Bought* feature of the Amazon website. In this network, nodes denote Amazon products. If a product i is frequently co-purchased with product j, the graph contains a directed edge from i to j.

(2) Synthetic Data. We used a C++ boost generator to produce graphs, which was controlled by two parameters: the number of vertices (n), and the number of edges (m). We generated 5 synthetic networks (RAND) by varying n from 100,000 to 1,000,000 with edges randomly chosen. All the synthetic networks follow the densification power law and linkage generation models.

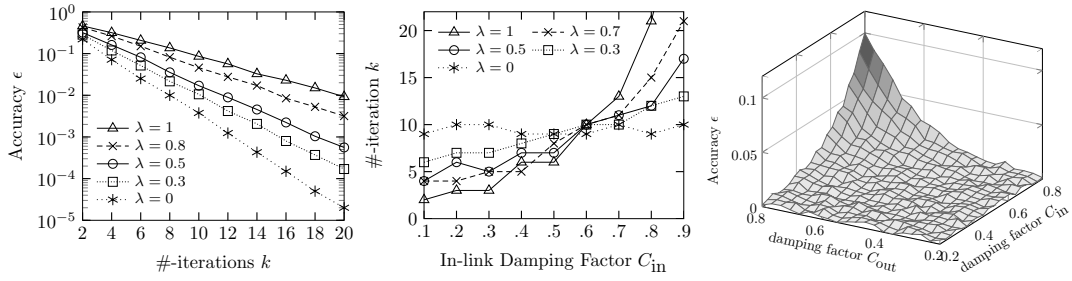
(3) Compared Algorithms. We have implemented the following algorithms in Visual C++ 10: (a) DE P-Rank and UN P-Rank, our proposed P-Rank algorithms over directed and undirected graphs, respectively; (b) Iter, the linearized iterative SimRank algorithm [20] modified to P-Rank computation plus the radius-based pruning methods of [54]; (c) Memo, the memoization-based algorithm [29] applied to P-Rank computation; (d) AUG, SimRank optimized algorithm [44] over undirected graphs.

(4) Parameter Settings. For fairness of comparison, the following parameters were used as default values (unless otherwise specified).

Notations	Description	Default Values
λ	weighting factor	0.5
C_{in}	in-link damping factor	0.8
C_{out}	out-link damping factor	0.6
υ	low approximation rank	$25\% \times \mathsf{Rank}(\mathcal{G})$
ϵ	desired accuracy	0.001

All the experiments were run on a machine with an Intel Core i7-4700MQ CPU @ 2.40GHz (8CPUs) and 32GB RAM, using Windows 7 Professional 64-bit. Each experiment was repeated 5 times and the average is reported.

[10] www.informatik.uni-trier.de/~ley/db/
[11] https://snap.stanford.edu/data/index.html



(a) p2p-Gnutella (b) email-EuAll (c) DBLP

Fig. 17. P-Rank Accuracy

6.2 Experimental Results

6.2.1 Accuracy. In the first set of experiments, we evaluate the impacts of the weighting factor λ and damping factors C_{in} and C_{out} on the accuracy of P-Rank iterations on all the real datasets. Due to similar trends, we pick up one representative dataset to report the result. The accuracy is measured by the absolute difference between the iterative and exact P-Rank similarity scores.[12]

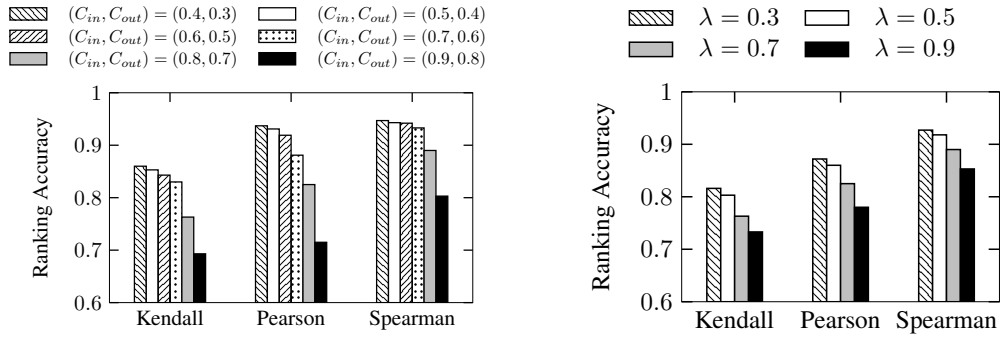
Varying λ and k. We first evaluate the effect of the number of iterations (k) on the accuracy of P-Rank iterations. Given the damping factors C_{in} and C_{out}, we vary the weighting factor λ from 0 to 1. For each pair of vertices and each fixed λ, we vary the number of iterations k from 2 to 20. The results are reported in Figure 17(a). Note that the logarithmic scale is chosen across the y-axis, to provide an illustrative look for the asymptotic rate of P-Rank convergence.

Due to the similar tendencies among different datasets, Figure 17(a) only depicts the results on one representative dataset (p2p-Gnutella). It can be noticed that, for each fixed λ, the downward lines for P-Rank iterations reveal an exponential accuracy as k increases, which is well consistent with Theorem 1. We also observe that, when λ increases from 0 to 1, the slope of the lines in Figure 17(a) will increase accordingly. This validates our theoretical results in Theorem 1 that increasing the weighting factor will decrease the convergence rate of P-Rank iteration, as expected.

Effect of damping factors on k, given ϵ. For guaranteeing the fixed accuracy ϵ, Figure 17(b) shows the impacts of damping factors *w.r.t.* the required number of P-Rank iterations (k). Due to the similar trends among various datasets, we only report the results on one representative dataset (email-EuAll). Fixing $\epsilon = 0.001$ and $C_{\text{out}} = 0.6$, we vary C_{in} from 0.1 to 0.9. (For space constraints, a similar result of varying C_{out} is omitted.) It can be noticed that when $\lambda = 0$, the curve in Figure 17(b) approaches a horizontal line. This is because P-Rank in this case boils down to the reversed SimRank model with no consideration of in-links, which makes C_{in} insensitive to the final P-Rank score. In the case of $0 < \lambda \leq 1$, the iteration number k shows a general increasing tendency as C_{in} grows. This tells that small choices of damping factors may reduce the number of iterations required for attaining a fixed accuracy, and hence, improves the efficiency of P-Rank.

Varying damping factors *w.r.t.* ϵ. Figure 17(c) evaluates the impact of both C_{in} and C_{out} *w.r.t.* the accuracy of P-Rank. Due to similar results on different datasets, we illustrate only the results on DBLP (1998-2007) dataset. We fix $k = 10$ and $\lambda = 0.5$, and vary C_{in} and C_{out} in x-axis and y-axis, respectively. It can be discerned that there is a 3D shaded surface from the average of P-Rank

[12]To select the P-Rank "exact" solution $s(\cdot,\cdot)$, we used *the Cauchy's criterion for convergence* and regarded the 100-th iterative $s^{(100)}(\cdot,\cdot)$ score as the "exact" one *s.t.* $|s^{(100)}(\cdot,\cdot) - s^{(101)}(\cdot,\cdot)| \ll 1 \times 10^{-10}$.



(a) Varying (C_{in}, C_{out}) (b) Varying λ

Fig. 18. Effects of Hyper-Parameters on P-Rank Stability

#	Most Similar Authors *w.r.t.* Dr. Dengfeng Gao	(C_{in}, C_{out}) (0.5, 0.4)	(0.7, 0.6)	(0.9, 0.8)
1	Thomas Phan	1	1	1
2	Rafae Bhatti	2	**3**	2
3	Masahiro Ohkawa	3	2	3
4	Masayuki Numao	4	4	4
5	Christine Robson	5	5	5
6	Yuji Watanabe	6	6	6
7	Hirofumi Matsuzawa	7	7	7
8	Merrie Brucks	8	8	8
9	L. Edwin McKenzie	9	9	9
10	Giedrius Slivinskas	10	10	10
11	Stardas Pakalnis	11	11	11
12	Shohei Hido	12	12	12
13	Harunobu Kubo	13	13	13
14	Yasuhiko Morimoto	**34**	14	14
15	Takeshi Tokuyama	14	15	15
16	Vishal S. Batra	15	**40**	16
17	Mahadevan Subramanian	16	16	17
18	Hisashi Kashima	**19**	17	18
19	Takeshi Fukuda	17	18	19
20	Khaled K. Al-Taha	18	19	**60**
21	Tsuyoshi Id	20	20	20
22	Simonas Saltenis	21	22	**28**
23	Rimantas Benetis	22	23	**29**
24	Gytis Karciauskas	23	24	**30**
25	Inderpal Narang	24	**21**	21
26	Laurynas Biveinis	25	25	**31**
27	Wen-Syan Li	**30**	26	22
28	Shinichi Morishita	26	27	23
29	Kyoji Hirata	27	**33**	24
30	Yoshinori Hara	28	**34**	25

(a) Varying (C_{in}, C_{out})

#	Most Similar Authors *w.r.t.* Prof. Xiaofei He	λ 0.5	0.7	0.9
1	Wanli Min	1	1	1
2	Kun Zhou	2	2	2
3	Shaozhi Ye	3	3	3
4	Lawrence J. Henschen	4	4	4
5	Daniel C. Fain	5	5	5
6	Wei Vivian Zhang	6	6	6
7	Fei Wu	7	7	**8**
8	Yuanzhi Zhang	8	8	**9**
9	Deng Cai	9	9	7
10	Cheong Youn	10	10	10
11	Wiley Greiner	**15**	**12**	11
12	Shengnan Cong	11	11	**21**
13	David A. Padua	12	13	12
14	Andrew Y. Wu	13	14	13
15	Michael Garland	14	15	14
16	Benjamin Rey	16	16	15
17	Yuguo Chen	17	**22**	16
18	Yunxiao Ma	**20**	17	**23**
19	Qing Yu	**24**	**24**	18
20	Xianghong Jasmine Zhou	18	19	19
21	Haifeng Liu	19	20	20
22	Behzad Shahraray	21	21	**17**
23	Avideh Zakhor	22	**23**	**50**
24	Noboru Babaguchi	23	**18**	**24**
25	Ji-Rong Wen	**54**	**53**	22
26	Nebojsa Stefanovic	25	**31**	**29**
27	Yifan Li	26	25	**32**
28	Yiwen Yin	27	26	26
29	Zaiqing Nie	**32**	27	27
30	Jean Hou	28	28	**41**

(b) Varying λ

Fig. 19. Qualitative Case Studies for P-Rank Stability

similarity accuracy for all vertex-pairs on z-axis. The P-Rank similarity residuals become huge only when C_{in} and C_{out} are both increasing to 1; and the iterative P-Rank results become accurate when C_{in} and $C_{\text{out}} < 0.6$. This validates that small choices of damping factors are suggested in P-Rank iterations in order to improve the accuracy of P-Rank similarities.

6.2.2 Stability. The second set of experiment is the evaluation of P-Rank stability. We show how the hyper-parameters (*e.g.,* weight factor λ and damping factors $(C_{\text{in}}, C_{\text{out}})$ affects the P-Rank stability, *i.e.,* how the similarity ranking accuracy on finding similarity objects changes in response to graph perturbations.

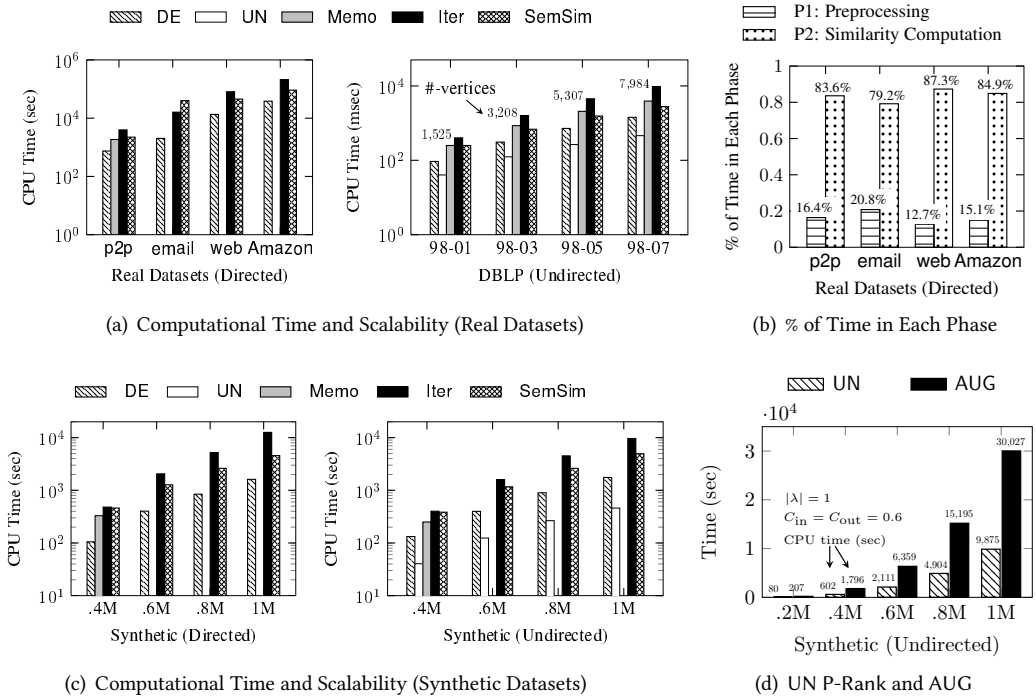
For each real graph $\mathcal{G}$, we randomly choose 100 nodes as a set of queries $\mathcal{Q}$ and randomly remove 10% edges to construct a perturbed graph $\tilde{\mathcal{G}}$. For each query $q \in \mathcal{Q}$, we assess the P-Rank similarities $\{s(\star, q)\}$ between all nodes and query q on the original $\mathcal{G}$ and perturbed $\tilde{\mathcal{G}}$, respectively, and then evaluate the correlations of the similarity ranks before and after graph perturbations, by using the following three metrics (Kendall's τ, Spearman's ρ, Pearson's γ):

- *Kendall's τ* is defined as
$$\tau = \frac{2}{N(N-1)} \sum_{\{i,j\} \in P} \bar{K}_{i,j}(\tau_1, \tau_2),$$
with $\bar{K}_{i,j}(\tau_1, \tau_2) = 1$ if i and j are in the same order in τ_1 and τ_2, and otherwise 0. Here, τ_1 and τ_2 are the rankings of elements in two lists, P is the set of unordered pairs in τ_1 and τ_2, and N is the number of elements in a ranking list.
- *Pearson's γ* is defined by
$$\gamma = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}}$$
where $\{x_i\}_{i=1}^{N}$ and $\{y_i\}_{i=1}^{N}$ are two similarity ranks, and $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$.
- *Spearman's ρ* is given by
$$\rho = 1 - \frac{6}{N(N^2-1)} \sum_{i=1}^{N} d_i^2,$$
where d_i is the ranking difference between the i-th elements in two lists.

For each measure, we take the average value for the rank correlations over all the queries in $\mathcal{Q}$.

Effects of Damping Factors C_{in} and C_{out} on Ranking Stability. We first fix the weight factor $\lambda = 0.4$, and vary the damping factors C_{in} and C_{out} to evaluate the stability of P-Rank on each dataset. Due to similar trends on different datasets, we only show the results on one dataset (DBLP). Figure 18(a) depicts how the ranking accuracy on finding similar authors changes in response to graph perturbations when we vary C_{in} and C_{out} from 0.3 to 0.9. From the results, we notice that, for the same perturbations on DBLP, when C_{in} and C_{out} are smaller, the ranking accuracy measured by Kendall's τ, Spearman's ρ, Pearson's γ are all higher, meaning that small choices of damping factors will make P-Rank similarity more stable to the graph perturbations. When $(C_{\text{in}}, C_{\text{out}})$ is increased from $(0.4, 0.3)$ to $(0.9, 0.8)$, there is a sharp decrease in the ranking accuracy, which decreases from 0.86 to 0.69 by Kendall's τ, from 0.95 to 0.80 by Spearman's ρ, and from 0.94 to 0.71 by Pearson's γ. This is because larger $(C_{\text{in}}, C_{\text{out}})$ induces a large P-Rank condition number which can make P-Rank less stable in response to graph perturbations. This result is well consistent with our theoretical analysis in Theorem 7.

Effects of Weight Factor λ on Ranking Stability. We next fix the damping factors $(C_{\text{in}}, C_{\text{out}}) = (0.8, 0.6)$, and vary the weight factor λ from 0.3 to 0.9 to see how this affects the stability of P-Rank. The results are reported in Figure 18(b). We can see that, when λ grows from 0.3 to 0.9, the ranking accuracy in response to the same graph perturbations decreases from 0.81 to 0.73 by Kendall's τ, from 0.92 to 0.85 by Spearman's ρ, and from 0.87 to 0.78 by Pearson's γ. This implies that, for the same graph perturbations, small choices of λ will make P-Rank similarity rankings less stable. The



(a) Computational Time and Scalability (Real Datasets)

(b) % of Time in Each Phase

(c) Computational Time and Scalability (Synthetic Datasets)

(d) UN P-Rank and AUG

Fig. 20. P-Rank Computational Time Efficiency

reason is that small λ will make the upper bound of the P-Rank condition number small, which guarantees that perturbations in graph structure will not be largely magnified to perturbations in similarity ranking results. This agrees well with our stability analysis of P-Rank in Subsection 4.2.

Two Qualitative Case Studies on Ranking Stability. We provide two qualitative case studies to illustrate the impact of hyper-parameters on the ranking stability of P-Rank for finding top-30 most similar authors *w.r.t.* a given query author. We issue our query on the original DBLP and the perturbed DBLP with 10% edges randomly removed, respectively, and compare their perturbations in ranking results. In the first case study, we issue the query "Dr. Dengfeng Gao", and analyse the stability for finding its top-30 most similar authors. We fix weight factor $\lambda = 0.4$ and vary $(C_{\text{in}}, C_{\text{out}})$ from $(0.5, 0.4)$ to $(0.9, 0.8)$. The results are shown in Figure 19(a), where the first two columns are the ranks and the corresponding similar authors *w.r.t.* "Dr. Dengfeng Gao" on the original DBLP, and the last three columns are the ranks under different settings of damping factors $(C_{\text{in}}, C_{\text{out}})$. We see that when $(C_{\text{in}}, C_{\text{out}})$ is small, the number of "flips" in ranks (underlined bold font) is small. When $(C_{\text{in}}, C_{\text{out}})$ increases to $(0.9, 0.8)$, there are a mass of "flips" in ranks. This implies that the large settings of damping factors $(C_{\text{in}}, C_{\text{out}})$ will make the ranking results less stable, as expected.

Similarly, for the second case study, we issue the query "Prof. Xiaofei He". Fixing the damping factors $(C_{\text{in}}, C_{\text{out}}) = (0.8, 0.6)$, we vary the weight factor λ from 0.5 to 0.9. Figure 19(b) depicts the ranking perturbations *w.r.t.* the perturbations of DBLP graph structure for finding similar authors under different settings of the weight factor λ. We notice that, when λ grows from 0.5 to 0.9, the number of ranking "flips" increases from 5 to 10, which indicates that P-Rank is unstable for large λ. This result is consistent with our stability analysis in Subsection 4.2.

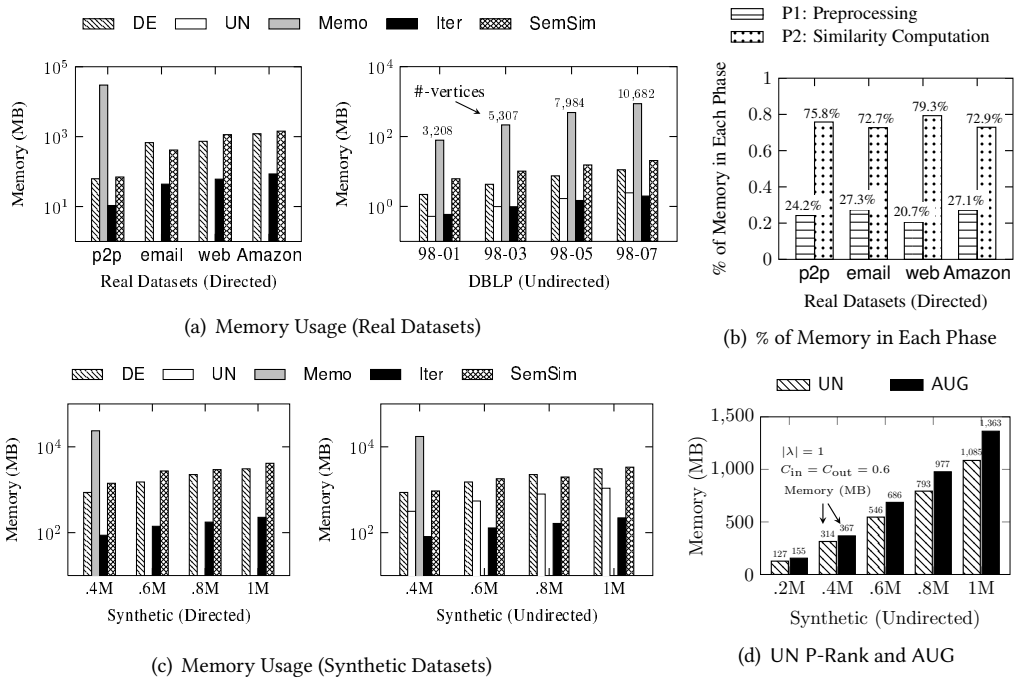
6.2.3 Computational Time. The third set of experiment is the evaluation of the computational time of DE P-Rank and UN P-Rank and their scalability over various real-life and synthetic datasets, including both directed and undirected networks.

Time and Scalability on Real Datasets. Figure 20(a) compares the computational time of DE P-Rank and UN P-Rank with those of Memo and Iter on a variety of real directed graphs (p2p-Gnutella, email-EuAll, web-Stanford, Amazon) and five undirected DBLP networks. The logarithmic scale is used on y-axis. The number of iterations for Iter and Memo is set to 10. Note that different time unit is chosen across the vertical axis in the two subplots of Figure 20(a) to provide a clear look for each bar shape.

From the results, we can discern the following: (1) On all real-life datasets, DE P-Rank is consistently much faster than other competitors. Fox example, on email-EuAll, the computational time of DE P-Rank (2,025s) is 7.96x faster than Iter (16,110s), whereas Memo crashes due to memory explosion. This is because DE P-Rank takes advantage of the low-rank structure of real networks, and eliminates significant amounts of unnecessary computations in its tensor products. Thus, by using DE P-Rank algorithm, the final P-Rank similarity matrix can be represented in a low-rank decomposition form, as opposed to the Memo that need to memorize all pairs iterative similarity scores of the previous iteration to compute those in the current iteration. (2) When the size of datasets grows larger, DE P-Rank and Iter, unlike Memo, can scale well at large scale, but Iter runs 5-10x slower than DE P-Rank The reason is that Iter is based on the SimRank-like linearized model to optimize the memory efficiency, which will increase the number of iterations to avoid memorizing intermediate results. In contrast, DE P-Rank is non-iterative by nature, and its scalability is guaranteed by exploiting the low-rank structure of the P-Rank similarity matrix. (3) The computational time of all the algorithms on DBLP increases with the increasing number of nodes. On all the DBLP datasets, UN P-Rank performed the fastest, and DE P-Rank the second. This is because UN P-Rank can fully utilize the undirected graph structure of DBLP (that is, the symmetry of its adjacency matrix) to speed up the computation of P-Rank similarities further, which is consistent with our theoretical time complexity bounds derived from Theorems 7 and 10. (4) On all the datasets, the computational time of the state-of-the-art SemSim is 1.7x–2.3x faster than Iter . This is because SemSim leverages a pruning technique based on the Monte-Carlo framework, in which only a fraction of important paths have been sampled as opposed to Iter that iteratively exploits the entire graph structure. However, DE P-Rank and UN P-Rank consistently outperform SemSim by 3–7.6 times on all the datasets. The reason is that the low-rank method sof DE P-Rank and UN P-Rank can efficiently aggregate the retrieval of nodes having similar neighboring structures, thus eliminating repeated path traversal for similarity computation. In contrast, SemSim at each time samples pairs of paths for assessing just a single-pair node similarity. A number of duplicate similarity computations among different pairs of nodes cannot be merged and eliminated.

% of Time in Each Phase. On every real-world dataset, we next evaluate the percentage of the computational time for DE P-Rank in each phase (that is, the pre-processing phase, and the similarity computation phase). The results are reported in Figure 20(b). It can be noticed that, on each dataset, the percentage of time in the pre-processing phase is around 16.4%–20.8%, whereas the percentage of the time in the similarity computation phase is around 79.9%–87.3%. These indicate that the similarity computation phase is consistently far more time-consuming than the pre-processing phase, which is consistent with our complexity analysis in Subsection 5.1.

Time and Scalability on Synthetic Datasets. By varying the number of vertices from 200K to 1M, we next compare the computational time of DE P-Rank and UN P-Rank with those of Memo and Iter on synthetic datasets, both directed and undirected, respectively. The results are



(a) Memory Usage (Real Datasets)

(b) % of Memory in Each Phase

(c) Memory Usage (Synthetic Datasets)

(d) UN P-Rank and AUG

Fig. 21. P-Rank Memory Space Usage

depicted in 20(c), in which y-axis is in the logarithmic scale. It can be seen that (1) in all cases when Memo does not fail, the computational time of DE P-Rank and UN P-Rank consistently outperforms those of Memo and Iter by 5–8 times, highlighting the effectiveness of our proposed non-iterative model for P-Rank similarity assessment. (2) On undirected synthetic datasets, UN P-Rank is always more than one order of magnitude faster than Iter and about 5 times faster than DE P-Rank, respectively. Similar to the results on real-world datasets, with the increasing size of the synthetic networks, Memo fails due to memory crash, but DE P-Rank and UN P-Rank can scale well on large networks. The reason is that, for Iter and Memo, there are a large number of repeated iterations to reach a fixed-point of P-Rank similarity scores, which impedes their scalability for similarity computation over large networks.

Time of UN P-Rank and AUG on Synthetic Data. To compare the computational time of UN P-Rank and AUG, we next compute SimRank similarities over undirected synthetic datasets. For fairness of comparison, we set $\lambda = 1$ for UN P-Rank. Thus, UN P-Rank reduces to SimRank, a special case of P-Rank with no consideration of out-links. By varying the number of vertices from 200K to 1M over synthetic datasets, we report the results in Figure 22(a). It can be seen that the computational time of UN P-Rank is approximately 3 times faster than that of AUG. This is because, after eigen-decomposition, AUG requires extra iterations to be performed in its eigen-subspace, and thus takes a significant amount of time to perform a number of iterations. In contrast, UN P-Rank can compute similarities in terms of eigenvectors with no need for any additional iterations.

6.2.4 Memory Space. The fourth set of experiment is the evaluation of the memory space of DE P-Rank and UN P-Rank on a variety of real-life and synthetic datasets.

Memory Usage on Real Datasets. Figure 21(a) evaluates the memory consumption of DE P-Rank and UN P-Rank on directed real datasets (p2p-Gnutella, email-EuAll, web-Stanford, and Amazon) and undirected DBLP datasets. It can be observed that (1) when the size of these real datasets grows, the memory space of each algorithm increases. This is well consistent with our space complexity analysis in Section 5. (2) In all cases when Memo does not crash, the memory of DE P-Rank is consistently smaller than that of Memo. For example, on p2p-Gnutelladataset, the memory of DE is almost 4.5 orders of magnitude smaller than Memo. This is because Memo requires quadratic storage to maintain all pairs of similarity scores. In contrast, DE P-Rank represents the similarity matrix into the low-rank form, which is the product of three small matrices, without the need to store the entire dense similarity matrix into memory. In addition, for DE P-Rank and UN P-Rank, the computation of the similarity matrix can be performed column-wisely, unlike Memo that requires to prepare all pairs of similarities of the previous iteration. (3) On each dataset, Iter requires less memory space than DE P-Rank, but the computational time of Iter is much more expensive than that of DE P-Rank. This is because Iter is based on a linearized iterative matrix equation for similarity computation, which does not need more memory to perform SVD decomposition and tensor products. However, since Iter is not based on low-rank matrix decomposition, it cannot take advantage of the low-rank structure of the real networks, and thereby entails high computational time. (4) On each DBLP dataset, the memory of UN P-Rank is smaller than that of DE P-Rank, and have comparable space with Iter. This space improvement is due to its non-iterative model that does not store the intermediate results of several tensor products. (5) The memory consumption of SemSim consistently retains the same order of magnitude as that of DE P-Rank on all the datasets. On large datasets, the memory of SemSim is almost slightly larger than DE P-Rank, whereas on small DBLP datasets, the extra space of SemSim relative to DE P-Rank is more pronounced, which is due to the storage of the random walk index built by SemSim.

% of Memory in Each Phase on Real Datasets. Figure 21(b) illustrates the percentage of memory usage in each phase of DE P-Rank over four real datasets, respectively. From the results, we can see that the percentage of memory consumption in the pre-processing phase is stable around 20.7%–27.3%, whereas its percentage in the similarity computation phase consistently increases to 72.2%–79.3%. This tells us that the similarity computation phase requires much more storage, which is mainly due to the memorization of the tensor products for computing the auxiliary matrix $\mathbf{\Lambda}$. This result agrees well with our space analysis in Subsection 5.1.

Memory Usage on Synthetic Datasets. We compare the memory usage of DE P-Rank and UN P-Rank with that of Memo and Iter on both directed and undirected synthetic datasets. For each type of datasets, we vary the number of nodes from 200K to 1M. Figure 21(c) shows the results on their memory usage. We can discern the following. (1) With the increasing number of nodes, the memory usage of DE P-Rank and UN P-Rank increases. This is because the growing size of the synthetic networks will lead to the increasing size of the decomposed matrices for P-Rank similarity computation, which conforms with our theoretical space complexity analysis in Section 5. (2) When the number of nodes is small, Memo does not fail but its memory consumption is one order of magnitude larger than DE P-Rank. This considerable increase is due to the memorization of all pairs of iterative similarities. In comparison, DE P-Rank and UN P-Rank adopt novel non-iterative models that compute all pairs of similarities column by column. (3) The memory space required by Iter is less than that of DE P-Rank and UN P-Rank. This is because Iter utilizes a linearized model to iteratively compute similarities, which does not need to store the intermediate results of tensor products for the low-rank factorization. (4) On undirected networks, UN P-Rank consistently yields less memory usage than DE P-Rank. The reason is that, unlike DE P-Rank on directed

graphs, UN P-Rank takes advantage of the symmetry of the adjacency matrix for undirected graphs without the need to store the auxiliary matrices Σ and $\mathbf{\Lambda}$.

Memory of UN P-Rank and AUG on Synthetic Datasets. As one special case when $\lambda = 1$, we now compare the memory space of UN P-Rank and AUG over undirected synthetic datasets for SimRank similarities computation. We vary the number of nodes from 200K to 1M and test their memory efficiency. The results are depicted in Figure 21(d). We can notice that, when the size of graphs grows, the memory consumptions for both UN P-Rank and AUG increase. However, the increment of AUG is faster than that of UN P-Rank. In addition, in all cases, the memory space of AUG is consistently smaller than that of UN P-Rank. The reason is that AUG requires additional memory to store iterative low-rank similarity scores in the compact space after eigen-decomposition, whereas UN P-Rank can directly obtain the resulting similarities with no need to iteratively compute similarity low-rank scores, thus saving additional memory space.

6.2.5 Accuracy. We next compare the accuracy of DE P-Rank and UN P-Rank with those of Memo and Iter on real-world datasets. Due to the similar tendencies, we only report the results on DBLP.

To measure the accuracy of P-Rank similarity scores, the following three metrics are adopted: Kendall's τ, Spearman's ρ, and NDCG. The definitions of first two measures (Kendall's τ and Spearman's ρ) are given in Subsection 6.2.2. *NDCG at position p w.r.t. query q* is given by

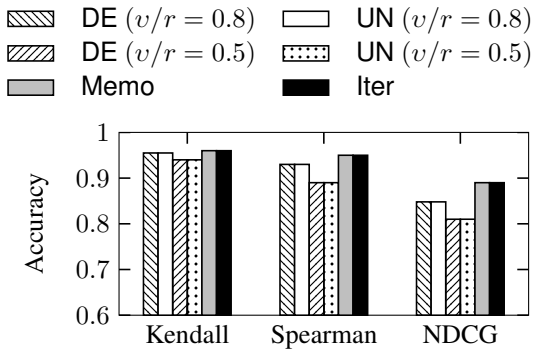
$$\mathrm{NDCG}_p(q) = \frac{1}{\mathrm{IDCG}_p(q)} \sum_{i=1}^{p} \frac{2^{s(i,q)} - 1}{\log_2(1+i)},$$

where $s(i,q)$ is the similarity score between nodes i and q, and $\mathrm{IDCG}_p(q)$ is a normalized factor ensuring the "true" NDCG ordering to be 1.

To produce the ground truth on real DBLP dataset, more than 20 researchers are hired from database and data mining areas to validate the "true" relevance of each retrieved co-authorship. They can refer to the "coauthor paths" in Microsoft Academic Search[13] to visualize the "degrees of separation" between two collaborators. The results are rendered by a majority vote of feedbacks.

Quantitative Results on Real Datasets. We randomly sample 100 queries from the DBLP dataset. These queries cover a board range of authors in DBLP, from newcomers to distinguished scholars. For each query q, we use the above three measures to evaluate the accuracy of four algorithms, respectively, by assessing their P-Rank similarities $s(q,v)$ for all nodes $v \in \mathcal{V}$. For algorithms DE P-Rank and UN P-Rank, we also vary the target approximation rank v from $0.5r$ to $0.8r$. The results are depicted in Figure 22(a). It can be discerned that (1) for every measure, the accuracies of all the algorithms are above 0.8. In particular, Memo and Iter have the same accuracy, and they are only slightly higher than DE P-Rank and UN P-Rank. For example, for Kendall's τ measure, the accuracies of Memo and Iter (0.95) are slightly higher than those of DE P-Rank and UN P-Rank (0.93 when $v = 0.8r$). This indicates that our low rank $v = 0.8r$ approximation will not sacrifice much accuracy while enabling a significant speedup in the similarity computation. (2) When the low rank v decreases from $0.8r$ to $0.5r$, the accuracies of DE P-Rank and UN P-Rank drop just a little, but are still acceptable (> 0.8) in practice. Thus, for some real applications (*e.g.,* top K search) when small errors are tolerated, we can choose a moderate low rank v to achieve a high computational speedup. This result is consistent with our theoretical bound in Subsection 5.1. (3) For each accuracy measure, given the low rank v, the accuracies of DE P-Rank and UN P-Rank are exactly the same. This is because DBLP is an undirected graph, and thus the resulting similarity scores of DE P-Rank are the same as those of UN P-Rank.

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



(a) Average Accuracy over 100 queries on DBLP

	DE / UN ($v/r = 0.5$)	DE / UN ($v/r = 0.8$)	Memo / Iter
1	Shivnath Babu	Shivnath Babu	Shivnath Babu
2	Yingwei Cui	Yingwei Cui	Yingwei Cui
3	Jun Yang	Chris Olston	Chris Olston
4	Chris Olston	Jun Yang	Jun Yang
5	David J. DeWitt	Arvind Arasu	Rajeev Motwani
6	Arvind Arasu	Rajeev Motwani	Arvind Arasu
7	Rajeev Motwani	Anish Das Sarma	Utkarsh Srivastava
8	Utkarsh Srivastava	Alon Y. Halevy	David J. DeWitt
9	Glen Jeh	Omar Benjelloun	Omar Benjelloun
10	Alon Y. Halevy	David J. DeWitt	Alon Y. Halevy

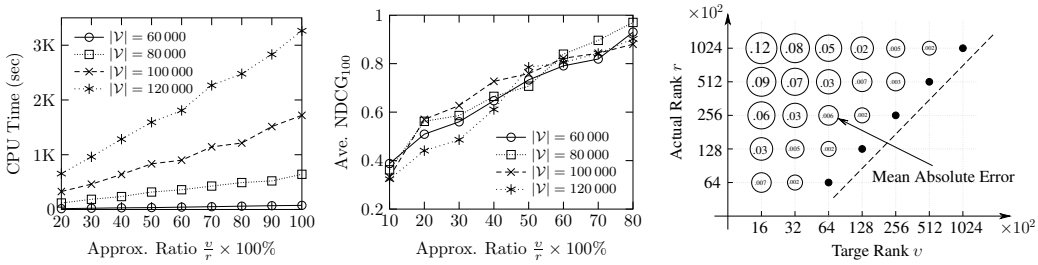
(b) Top-10 Similar Authors of "Jennifer Widom" on DBLP

Fig. 22. P-Rank Accuracy

A Qualitative Case Study. To demonstrate the effectiveness of our proposed methods, we next provide a real case study on DBLP (98–07) that retrieves the top K most similar authors for a given query q. For algorithms DE P-Rank and UN P-Rank, we vary the target approximation rank v from $0.5r$ to $0.8r$. We randomly sample 100 queries. Due to the space constraints, Figure 22(b) depicts the top-10 similarity ranking results *w.r.t.* the query "Jennifer Widom" according to the P-Rank similarity scores returned by DE P-Rank, UN P-Rank, Memo and Iter, respectively. These people were frequent co-authors of the 6 major conference papers with "Jennifer Widom" from 1998 to 2007. It can be noticed that the ranking results for different algorithms on DBLP (98-07) follow our common sense pretty well. When v increases from $0.5r$ to $0.8r$, the similarity ranking results returned by DE P-Rank and UN P-Rank almost preserve the relative order of those by Memo and Iter. Hence, DE P-Rank and UN P-Rank can be effectively used for P-Rank top-K nearest neighbor search on real-world networks.

6.2.6 Effect of Target Low Rank v. To evaluate the efficiency of DE P-Rank algorithm further, we next investigate the impact of the target low rank v on its similarity computational time and accuracy by using synthetic datasets.

Speed-Accuracy Trade-off. We generate four networks with the number of nodes n ranging from 60,000 to 120,000. For each network, we vary the target low rank v from $20\% \times r$ to $100\% \times r$, with r being the rank of its adjacency matrix. The results are depicted in Figures 23(a) and 23(b), in



(a) Target Rank v *w.r.t.* Time (b) Target Rank v *w.r.t.* NDCG (c) Target Rank v *w.r.t.* MAE

Fig. 23. Effect of Target Rank v for DE P-Rank

which the target low rank v is a speed-accuracy trade-off. We can discern the following. When v get increasingly close to r (*i.e.,* the approximation ratio $\frac{v}{r}$ approaches 1), high accuracy ($NDCG_{30}$) can be expected (Figure 23(b)), but more computational time needs to be consumed (Figure 23(a)). This tells that adding the target low rank v can produce smaller errors for similarity computation, but will sacrifice its computational time. As an extreme case of $v = r$, DE P-Rank will produce exact P-Rank similarity scores.

Target Rank v *w.r.t.* Accuracy. We next vary the target rank v on five synthetic datasets, and evaluate the impact of v on the accuracy of DE P-Rank. To measure the accuracy, we adopt the *Mean Absolute Error* (MAE) defined as follows:

$$\text{MAE} = \frac{1}{n^2} \sum_{u,v \in \mathcal{V}} |s(u,v) - \hat{s}(u,v)|,$$

where $s(u,v)$ is the exact similarity from DE P-Rank ($v = r$), and $\hat{s}(u,v)$ its rank-v approximate similarity ($v < r$). Fixing the number of nodes (n) and the number of edges (m), we generate five networks with the rank r growing geometrically from 6,400 to 100,000. Figure 23(c) illustrates the accuracy (MAE) of DE P-Rank when we vary v from 1,600 to r for each network, in which x-axis represents the target approximation rank we use, and y-axis denotes the rank r of the adjacency matrix for a given network. The number enclosed in a circle ∘ corresponds to the mean absolute error obtained by the rank v approximation of DE P-Rank, and the size of the circles is proportional to the mean absolute error. It can be observed that (1) in all cases, the size of circle shrinks when v approaches r. This implies that the large choice of v will reduce the error of the rank v approximation for DE P-Rank, and thereby can achieve high accuracy. (2) The size of the circles in the diagonal direction (↗) remains almost the same. This tells us that, for the rank v approximation, the accuracy of DE P-Rank mainly hinges on the approximation ratio $\frac{v}{r}$.

7 RELATED WORK

There has been a surge of studies on link-based analysis (*e.g.,* [3, 14, 19, 26, 32, 35, 40, 43, 45, 55]) over the last decade. Google PageRank is a link analysis model that has become popular since the well-known result of Larry Page [32] for ranking web pages. Unlike P-Rank that provides a *node-pair* similarity measure, PageRank is a *node* ranking algorithm that is independent of user queries. Since then, a large body of work has been proposed for measuring node-to-node similarities that are based on the topology of a network. Some of these attractive similarity models include P-Rank [54] and its variations (*e.g.,* SimRank [14], SimRank++ [2], RoleSim [17], MatchSim [27]),

Personalized PageRank (PPR) [10], Random Walk with Restart (RWR) [38], C-Rank [41], SimFusion [40], and CoSimRank [52], to name a few.

The P-Rank model was first proposed by Zhao *et al.* [54] who noticed the limited information problem of Jeh and Widom's SimRank model [14], that is, SimRank similarity scores consider only in-link relationships of entities whereas out-link relationships are totally neglected. However, the P-Rank model refines SimRank by jointly considering both in- and out-links of entity pairs. The conventional algorithm for P-Rank similarity computation is based on a fixed-point iteration, which requires $O(kn^4)$ time and $O(n^2)$ memory for k iterations. To optimize the computational efficiency, Zhao *et al.* [54] have proposed a radius- or category-based pruning technique that can reduce the computation of P-Rank to $O(kd^2n^2)$ time. However, the memory requirement is still bounded by $O(n^2)$ space. In addition, this approach is heuristic in nature, and thereby the accuracy of the pruning result is not guaranteed.

Recent years have witnessed an upsurge of interest in the optimization of SimRank similarity computation that can be modified to P-Rank as well. The existing interesting algorithms include iterative amortization-based methods [20, 29], low rank matrix-based methods [6, 23, 44], parallel computing methods [11, 25], local partial-pairs computing methods [5, 12, 30, 36, 49], uncertain schemes [56], and dynamical incremental methods [23, 34, 46]. Among these methods, two pieces of work [6, 20] are worth mentioning as they are the state of the art for iterative and non-iterative SimRank models, respectively.

Kusumoto *et al.* [20] have proposed a linearized recursive formula for SimRank search problem. This algorithm is a very scalable on large networks with billions of edges. Its main idea can be directly extended to P-Rank search problem, which requires only $O(m)$ memory, but its computational cost is expensive, yielding $O(k^2nm)$ time for evaluating the similarities of all (n^2) pairs of nodes. In comparison, our algorithm is non-iterative, which can return exact similarity scores. Regarding the computational efficiency, our method can be 5-10x faster while sacrificing a little memory. However, the additional memory requirement does not affect the scalability of our algorithm over large networks.

Fujiwara *et al.* [6] proposed the following non-iterative model for SimRank computation

$$\frac{1}{C}(\mathbf{W}^T)^{-1}\mathbf{S} - \mathbf{S}\mathbf{W} = \frac{1-C}{C}(\mathbf{W}^T)^{-1}$$

where $\mathbf{W}$ is the transpose of the column-normalized adjacency matrix, $\mathbf{S}$ is the SimRank similarity matrix, and C is the damping factor. This model can be readily generalized to P-Rank computation. However, according to the above matrix equation, the inverse of the normalized adjacency matrix $(\mathbf{W}^T)$ must exist. This indicates that this non-iterative model is well applicable to real-life networks whose rank of the adjacency matrix is full (that is, $\text{rank}(\mathbf{W}^T) = n$).

In the case of $\text{rank}(\mathbf{W}^T) < n$, Li *et al.* [23] proposed a non-iterative matrix-based scheme for SimRank computation, which requires $O(v^4n^2)$ time and $O(v^2n^2)$ memory yet with no theoretically guaranteed error bound for directed graphs. These techniques, if directly generalized to P-Rank computation, will produce the same computational complexity, which is rather expensive. In contrast, our methods first provide optimization techniques that can significantly speed up the computation of Li *et al.*'s SimRank algorithm to $O(vn^2 + v^6)$ time and $O(vn + v^4)$ memory, with guaranteed accuracy. Moreover, we extend our proposed optimization techniques to P-Rank computation further, and also derive a more efficient algorithm over undirected graphs that entails $O(vn^2)$ time and $O(vn)$ memory. We are also the first to analyze the stability of P-Rank measure.

Lizorkin *et al.* [29] leveraged a novel function for partial sums memorization, which can significantly speed up the computation of SimRank to $O(kn^3)$ time for k iterations. Later, one piece of work [46] made a further attempt towards this direction, by proposing fine-grained memorization

algorithms. All these approaches can be extended to P-Rank computation, but they require $O(n^2)$ memory to store all pairs of similarities at the previous iteration. Most recently, to avoid $O(n^2)$ memory, some efficient iterative methods [20, 49] have been proposed. However, they differ from this work in that those algorithms are iterative in nature, whereas our method is non-iterative and can compute similarity matrix column by column accurately.

An accuracy estimation for SimRank iterations was addressed in [29]. However, directly applying the SimRank bound [29] to P-Rank iterations is less tight. This is because, as shown in Section 3, the upper bound we derived in Theorem 1 is always smaller (tighter) than the simple linear combination of the SimRank bound [29] for P-Rank accuracy estimation. We have also provided an illustrative example to show the difference (tightness) between these two bounds in Example 2.

For undirected networks, a time-efficient algorithm AUG [44] was proposed for SimRank computation, which requires $O(n^3 + kn^2)$ time and $O(n^2)$ memory. In contrast, this work significantly reduces the computational bound of [44] to $O(rn^2)$ time and $O(rn)$ memory without any loss of exactness.

In recent years, there has also been a host of work on efficient pairwise similarity search. Tian and Xiao [37] propose an excellent index structure, called SLING, to efficiently retrieve SimRank single-source similarities, which guarantees the worst-case error in each SimRank score returned. Most recently, Milo *et al.* [42] present SemSim, a variant of SimRank, that exploits semantic-aware random walk while preserving its scalable computation based on the probabilistic framework of SLING. To the best of our knowledge, this is the state-of-the-art algorithm for efficiently retrieving a single pair SimRank score, which utilises importance sampling techniques along with effective pruning rules and maintains a small error rate. However, when SemSim is extended to evaluate more pairs of nodes, unlike our methods that efficiently aggregate the retrieval of nodes that have a similar neighbouring structure, there may exist repeated traversal of paths for SemSim to evaluate all pairs of similarities.

Hamedani and Kim [9] pointed out a SimRank-like pairwise normalization problem (PNP) also exists in P-Rank, where the P-Rank score of a pair of nodes referenced commonly by a large number of nodes tends to be lower than that of another pair of nodes commonly referenced by a small number of nodes. They modified the normalization factors of P-Rank to resolve this problem. Yu and Julie [49] proposed an efficient scheme of partial-pairs SimRank search for similarity join on sizable graphs. Zhang *et al.* [53] provided extensive experimental studies on many existing SimRank algorithms in a unified environment. Their results demonstrate that, despite recent research efforts, there are still much space of improvement for the computational cost and accuracy of existing similarity search algorithms.

Regarding dynamical similarity search, Shao *et al.* [34] and Jiang *et al.* [16] proposed TSF and READS indexing techniques, respectively, to efficiently handle top-k SimRank search over dynamic graphs. Liu *et al.* [28] proposed ProbeSim, an index-free approach for dynamic single-source and top-k SimRank queries with accuracy guarantees. Yu *et al.* [47] proposed a dynamical algorithm for incrementally assessing all-pairs SimRank scores on time-varying networks. Wang *et al.* [39] designed an efficient incremental single-source SimRank search over dynamic graphs. However, it seems difficult to directly extend these incremental SimRank techniques to dynamic P-Rank computation since the changes to the P-Rank similarity score in response to the graph perturbation cannot be simply captured by resampling the incoming paths, given the fact that outgoing edges interleaved with incoming edges will have a recursive impact on the changes to the resulting P-Rank similarity score.

More recently, there are a variety of approaches on graph neural networks that seek to learn representations that encode structural information about the graph, such as DeepWalk [33], node2vec [8], GraRep [4], and Graph Convolutional Networks (GCN) [21]. Perozzi *et al.* [33] proposed a

pioneering approach, DeepWalk, for learning latent representations of nodes in a network. It is the pioneering algorithm proposing node embedding learned in an unsupervised manner, which highly resembles word embedding in terms of the training process. DeepWalk performs random walks on nodes in a graph to generate node sequences, based on which, it then runs Skip-gram to learn the embedding of each node. However, DeepWalk is not suitable for dynamic graphs since, when a new node comes in, it has to re-train the model.

Node2vec, proposed by Grover and Leskovec [8], is an excellent extension of DeepWalk with the difference in random walks. As opposed to DeepWalk using first-order uniform random walks which gives us no control over the explored neighbourhoods, Node2vec defines an unnormalized transition probability tensor with hyperparameters p and q, and then normalizes it to be the transition probability of a second-order random walk.

Cao *et al.* [4] proposed GraRep, a novel factorization-based model, for learning vertex representations of weighted graphs, which suggests using SVD on a log-transformed DeepWalk transition probability matrix of different orders. GraRep generalizes LINE to incorporate information from network neighbourhoods beyond 2-hops.

Graph convolutional networks (GCNs) [21] present an end-to-end approach to structured learning, in contrast to Node2vec and DeepWalk producing summaries that are later analyzed with a machine learning technique. GCN methods seek to generalize traditional convolutional networks to the variable, unordered structures.

In comparison, our work differs from the aforementioned neural methods in that our goal is to analyse the stability and optimize the computational efficiency of a specific P-Rank similarity model by employing advanced linear algebra tools (which provides deterministic provable guaranteed accuracy) rather than using graph neural networks to learn graph representations (which may induce a new different similarity model). In our future work, we will incorporate these graph embedding methods to pairwise similarity measures, aiming at enhancing the semantics and efficiency of the graph-based similarity models further.

8 CONCLUSIONS

In this study, we have studied the problems of P-Rank computation on large graphs. We have proposed an accuracy estimate for the P-Rank iteration, by finding out the exact number of iterations needed to attain a given accuracy. We have obtained a tight bound for the P-Rank condition number to analyze the stability of P-Rank. We have also devised efficient algorithms to evaluate all P-Rank similarities in $O(vn^2 + v^6)$ time and $O(vn + v^4)$ space for digraphs, and $O(vn^2)$ time and $O(vn)$ space for undirected graphs. Empirical results on both real and synthetic datasets demonstrate the high efficiency of our methods for P-Rank computation in terms of computational time, memory usage, and search quality.

9 ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China (61972203 and 61702560), Natural Science Foundation of Jiangsu Province (BK20190442), and Natural Science Foundation of Hunan Province (2018JJ3691).

REFERENCES

[1] R. Amsler. Application of citation-based automatic classification. Technical report, The University of Texas at Austin, Linguistics Research Center, December 1972.

[2] I. Antonellis, H. Garcia-Molina, and C.-C. Chang. SimRank++: Query rewriting through link analysis of the click graph. *PVLDB*, 1(1), 2008.

[3] Y. Cai, M. Zhang, C. H. Q. Ding, and S. Chakravarthy. Closed form solution of similarity algorithms. In *SIGIR*, pages 709–710, 2010.

[4] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900, 2015.

[5] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, pages 641–650, 2005.

[6] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, and M. Onizuka. Efficient search algorithm for SimRank. In *ICDE*, pages 589–600, 2013.

[7] G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.).* John Hopkins University Press, 1996.

[8] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016.

[9] M. R. Hamedani and S. Kim. Pairwise normalization in SimRank variants: Problem, solution, and evaluation. In *SAC*, pages 534–541, 2019.

[10] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

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

[12] J. He, H. Liu, J. X. Yu, P. Li, W. He, and X. Du. Assessing single-pair similarity over graphs by aggregating first-meeting probabilities. *Inf. Syst.*, 42:107–122, 2014.

[13] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1990.

[14] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[15] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279, 2003.

[16] M. Jiang, A. W. Fu, R. C. Wong, and K. Wang. READS: A random walk approach for efficient and accurate dynamic simrank. *PVLDB*, 10(9):937–948, 2017.

[17] R. Jin, V. E. Lee, and L. Li. Scalable and axiomatic ranking of network role similarity. *TKDD*, 8(1):3:1–3:37, 2014.

[18] Y. Kanza, E. Kravi, E. Safra, and Y. Sagiv. Location-based distance measures for geosocial similarity. *TWEB*, 11(3):17:1–17:32, 2017.

[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[20] M. Kusumoto, T. Maehara, and K. Kawarabayashi. Scalable similarity search for SimRank. In *SIGMOD*, pages 325–336, 2014.

[21] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[22] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg. Signed networks in social media. In *CHI*, pages 1361–1370, 2010.

[23] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of SimRank for static and dynamic information networks. In *EDBT*, 2010.

[24] X. Li, W. Yu, B. Yang, and J. Le. ASAP: Towards accurate, stable and accelerative Penetrating-Rank estimation on large graphs. In *WAIM*, pages 415–429, 2011.

[25] Z. Li, Y. Fang, Q. Liu, J. Cheng, R. Cheng, and J. C. S. Lui. Walking in the cloud: Parallel SimRank at scale. *PVLDB*, 9(1):24–35, 2015.

[26] Y. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. L. Tseng. Detecting splogs via temporal dynamics using self-similarity analysis. *TWEB*, 2(1):4:1–4:35, 2008.

[27] Z. Lin, M. R. Lyu, and I. King. Matchsim: a novel similarity measure based on maximum neighborhood matching. *Knowl. Inf. Syst.*, 32(1):141–166, 2012.

[28] Y. Liu, B. Zheng, X. He, Z. Wei, X. Xiao, K. Zheng, and J. Lu. ProbeSim: Scalable single-source and top-k simrank computations on dynamic graphs. *PVLDB*, 11(1):14–26, 2017.

[29] D. Lizorkin, P. Velikhov, M. N. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for SimRank computation. *VLDB J.*, 19(1), 2010.

[30] T. Maehara, M. Kusumoto, and K. Kawarabayashi. Scalable simrank join algorithm. In *ICDE*, pages 603–614, 2015.

[31] C. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM: Society for Industrial and Applied Mathematics, Feb. 2001.

[32] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, November 1999.

[33] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014.

[34] Y. Shao, B. Cui, L. Chen, M. Liu, and X. Xie. An efficient similarity search framework for SimRank over large dynamic graphs. *PVLDB*, 8(8):838–849, 2015.

[35] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Am. Soc. Inf. Sci.*, 24(4):265–269, 1973.

[36] W. Tao, M. Yu, and G. Li. Efficient top-k SimRank-based similarity join. *PVLDB*, 8(3):317–328, 2014.

[37] B. Tian and X. Xiao. SLING: A near-optimal index structure for SimRank. In *SIGMOD*, pages 1859–1874, 2016.
[38] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.
[39] Y. Wang, X. Lian, and L. Chen. Efficient simrank tracking in dynamic graphs. In *ICDE*, pages 545–556, 2018.
[40] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. SimFusion: Measuring similarity using unified relationship matrix. In *SIGIR*, 2005.
[41] S. Yoon, S. Kim, and S. Park. C-Rank: A link-based similarity measure for scientific literature databases. *Inf. Sci.*, 326:25–40, 2016.
[42] B. Youngmann, T. Milo, and A. Somech. Boosting SimRank with semantics. In *EDBT*, pages 37–48, 2019.
[43] W. Yu, J. Le, X. Lin, and W. Zhang. On the efficiency of estimating Penetrating Rank on large graphs. In *SSDBM*, pages 231–249, 2012.
[44] W. Yu, X. Lin, and J. Le. Taming computational complexity: Efficient and parallel SimRank optimizations on undirected graphs. In *WAIM*, 2010.
[45] W. Yu, X. Lin, W. Zhang, L. Chang, and J. Pei. More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks. *PVLDB*, 7(1):13–24, 2013.
[46] W. Yu, X. Lin, W. Zhang, and J. A. McCann. Fast all-pairs SimRank assessment on large graphs and bipartite domains. *IEEE Trans. Knowl. Data Eng.*, 27(7):1810–1823, 2015.
[47] W. Yu, X. Lin, W. Zhang, and J. A. McCann. Dynamical SimRank search on time-varying networks. *VLDB J.*, 27(1):79–104, 2018.
[48] W. Yu, X. Lin, W. Zhang, J. Pei, and J. A. McCann. SimRank*: Effective and scalable pairwise similarity search based on graph topology. *VLDB J.*, 28(3):401–426, 2019.
[49] W. Yu and J. A. McCann. Efficient partial-pairs SimRank search for large networks. *PVLDB*, 8(5):569–580, 2015.
[50] W. Yu and J. A. McCann. High quality graph-based similarity search. In *ACM SIGIR*, pages 83–92, 2015.
[51] W. Yu and J. A. McCann. Random walk with restart over dynamic graphs. In *ICDM*, pages 589–598, 2016.
[52] W. Yu and F. Wang. Fast exact CoSimRank search on evolving and static graphs. In *WWW*, pages 599–608, 2018.
[53] Z. Zhang, Y. Shao, B. Cui, and C. Zhang. An experimental evaluation of SimRank-based similarity search algorithms. *PVLDB*, 10(5):601–612, 2017.
[54] P. Zhao, J. Han, and Y. Sun. P-Rank: A comprehensive structural similarity measure over information networks. In *CIKM*, 2009.
[55] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1), 2009.
[56] R. Zhu, Z. Zou, and J. Li. Simrank computation on uncertain graphs. In *ICDE*, pages 565–576, 2016.

A PROOFS OF THEOREMS & LEMMAS

A.1 Proof of Theorem 1

Proof. (i) For $u = v$, Eq.(7) obviously holds since

$$s(u,u) - s^{(k)}(u,u) = 1 - 1 = 0. \quad (\forall k \geq 0,\ \forall u \in \mathcal{V})$$

(ii) For $u \neq v$, we use induction on k to prove Eq.(7) as follows:

Inductive Basis. We show that Eq.(7) holds for $k = 0$. Using Eq.(3) and $s^{(0)}(u,v) = 0$, we have

$$\begin{aligned}
&s(u,v) - s^{(0)}(u,v) = s(u,v) \\
&= \frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|}\sum_{j=1}^{|\mathcal{I}(v)|} \underbrace{s\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right)}_{\leq 1} + \frac{(1-\lambda)\cdot C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|}\sum_{j=1}^{|\mathcal{O}(v)|} \underbrace{s\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right)}_{\leq 1} \\
&\leq \frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|}\sum_{j=1}^{|\mathcal{I}(v)|} 1 + \frac{(1-\lambda)\cdot C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|}\sum_{j=1}^{|\mathcal{O}(v)|} 1 \\
&= \lambda \cdot C_{\text{in}} + (1-\lambda)\cdot C_{\text{out}}.
\end{aligned}$$

Inductive Step. Assume that Eq.(7) holds for a fixed k as the inductive hypothesis. We need to prove that Eq.(7) holds for $k+1$ as well. Combining Eqs.(3) and (5) yields

$$s(u,v) - s^{(k+1)}(u,v)$$

$$
\begin{aligned}
&= \frac{\lambda C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|} \sum_{j=1}^{|\mathcal{I}(v)|} \overbrace{\left(s\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right) - s^{(k)}\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right)\right)}^{\text{using inductive hypothesis } \leq (\lambda C_{\text{in}} + (1-\lambda) C_{\text{out}})^{k+1}} \\
&+ \frac{(1-\lambda)\, C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|} \sum_{j=1}^{|\mathcal{O}(v)|} \underbrace{\left(s\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right) - s^{(k)}\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right)\right)}_{\text{using inductive hypothesis } \leq (\lambda C_{\text{in}} + (1-\lambda) C_{\text{out}})^{k+1}} \\
&\leq \frac{\lambda C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|} \sum_{j=1}^{|\mathcal{I}(v)|} (\lambda C_{\text{in}} + (1-\lambda)\, C_{\text{out}})^{k+1} \\
&\quad + \frac{(1-\lambda)\, C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|} \sum_{j=1}^{|\mathcal{O}(v)|} (\lambda C_{\text{in}} + (1-\lambda)\, C_{\text{out}})^{k+1} \\
&= \lambda C_{\text{in}} (\lambda C_{\text{in}} + (1-\lambda)\, C_{\text{out}})^{k+1} + (1-\lambda)\, C_{\text{out}} (\lambda C_{\text{in}} + (1-\lambda)\, C_{\text{out}})^{k+1} \\
&= (\lambda C_{\text{in}} + (1-\lambda)\, C_{\text{out}})^{k+2}.
\end{aligned}
$$

This completes the induction. □

A.2 Proof of Theorem 2

Proof. Dividing both sides of Eq.(9) by $(1 - \lambda C_{\text{in}} - (1-\lambda) C_{\text{out}})$ results in

$$\mathbf{S}' = \lambda C_{\text{in}} \cdot \mathbf{Q} \cdot \mathbf{S}' \cdot \mathbf{Q}^T + (1-\lambda) C_{\text{out}} \cdot \mathbf{P} \cdot \mathbf{S}' \cdot \mathbf{P}^T + \mathbf{I}_n,$$

where

$$\mathbf{S} = (1 - \lambda C_{\text{in}} - (1-\lambda) C_{\text{out}}) \cdot \mathbf{S}'.$$

□

A.3 Proof of Lemma 1

Proof. It follows from Eq.(8) that for each row $i = 1, \cdots, n$, the sum of each row in $\mathbf{Q}$ and $\mathbf{P}$ is no greater than 1. Then, for each row i' of $\mathbf{Q} \otimes \mathbf{Q}$, we have

$$\sum_{k=1}^{n} (q_{i',k} \sum_{j=1}^{n} q_{i,j}) \leq \sum_{k=1}^{n} (q_{i',k} \times 1) \leq 1,$$

which indicates that $\mathbf{Q} \otimes \mathbf{Q}$ is a row sub-stochastic matrix.

A similar proof holds for $\mathbf{P} \otimes \mathbf{P}$. □

A.4 Proof of Theorem 3

Proof. Let us take vec$(*)$ on both sides of Eq.(10), and then apply the Kronecker property vec$(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A}) \cdot$ vec$(\mathbf{X})$ [7, p.180], which produces

$$\text{vec}(\mathbf{S}') = (1-\lambda) C_{\text{out}} (\mathbf{P} \otimes \mathbf{P}) \cdot \text{vec}(\mathbf{S}') + \lambda C_{\text{in}} (\mathbf{Q} \otimes \mathbf{Q}) \cdot \text{vec}(\mathbf{S}') + \text{vec}(\mathbf{I}_n).$$

Rearranging the terms in the above equation yields

$$\mathbf{M} \cdot \text{vec}(\mathbf{S}') = \text{vec}(\mathbf{I}_n) \;\text{ with }\; \mathbf{M} = \mathbf{I}_{n^2} - \lambda C_{\text{in}} (\mathbf{Q} \otimes \mathbf{Q}) - (1-\lambda) C_{\text{out}} (\mathbf{P} \otimes \mathbf{P}),$$

which is a linear matrix equation.

By Lemma 1, $\mathbf{Q}\otimes\mathbf{Q}$ and $\mathbf{P}\otimes\mathbf{P}$ are sub-stochastic matrices. Thus, it can be readily proved that $\mathbf{M}$ is a diagonally dominant matrix, which implies that $\mathbf{M}$ is invertible. Hence, pre-multiplying $\mathbf{M}^{-1}$ on both sides of $\mathbf{M}\cdot\text{vec}(\mathbf{S}') = \text{vec}(\mathbf{I}_n)$ and, by Theorem 2, we have

$$\text{vec}(\mathbf{S}) = (1-\lambda C_{\text{in}} - (1-\lambda)C_{\text{out}})\cdot\mathbf{M}^{-1}\cdot\text{vec}(\mathbf{I}_n).$$

□

A.5 Proof of Lemma 2

Proof. Let $\mathbb{1}_{n^2}$ be a vector of length n^2 with entries of all 1s, and $\mathbf{e}_i$ a unit vector of length n^2 with a 1 in the i-th entry and 0s in all others.

Since $\mathbf{Q}\otimes\mathbf{Q}$ and $\mathbf{P}\otimes\mathbf{P}$ are row sub-stochastic matrices, it follows from Eq.(11) that $\forall i = 1, 2, \cdots, n^2$,

$$\begin{aligned}&\|\mathbf{I}_{n^2} - \mathbf{M} + [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T\|_\infty\\ =\ &\|\lambda C_{\text{in}}(\mathbf{Q}\otimes\mathbf{Q}) + (1-\lambda)C_{\text{out}}(\mathbf{P}\otimes\mathbf{P}) + [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T\|_\infty \le 1.\end{aligned}$$

Hence, $\mathbf{I}_{n^2} - \mathbf{M} + [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T$ is a row sub-stochastic matrix. Due to the *spectral radius*[14] property $\rho(\star) \le \|\star\|_\infty$, it follows that

$$\rho(\mathbf{I}_{n^2} - \mathbf{M} + [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T) \le 1.$$

Notice that $\mathbf{I}_{n^2} - \mathbf{M} + [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T$ is nonnegative. According to *the eigen-pair property for the nonnegative matrix* [15], there exists some row-vector $\mathbf{x}_i^T$ with $\|\mathbf{x}_i^T\|_\infty = 1$ such that $\forall i = 1, 2, \cdots, n^2$,

$$\mathbf{x}_i^T\cdot(\mathbf{I}_{n^2} - \mathbf{M} + [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T) \le \mathbf{x}_i^T.$$

Rearranging the terms in the above inequality produces

$$\mathbf{x}_i^T\cdot\mathbf{M} \ge [1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}}]\cdot\mathbf{x}_i^T\cdot\mathbb{1}_{n^2}\cdot\mathbf{e}_i^T. \tag{35}$$

Note that $\|\mathbf{x}_i^T\|_\infty = 1$, which implies that $\mathbf{x}_i^T\cdot\mathbb{1}_{n^2} = 1$. Post-multiplying by $\mathbf{M}^{-1}$ on both sides of Eq.(35) produces $\forall i = 1, 2\cdots, n^2$,

$$\mathbf{e}_i^T\cdot\mathbf{M}^{-1} \le 1/(1-\lambda\cdot C_{\text{in}} - (1-\lambda)\cdot C_{\text{out}})\cdot\mathbf{x}_i^T.$$

Also, notice that $\|\mathbf{M}^{-1}\|_\infty = \max_{1\le i\le n^2}\|\mathbf{e}_i^T\cdot\mathbf{M}^{-1}\|_\infty$ and $\|\mathbf{x}_i^T\|_\infty = 1$. Taking ∞-norm on both sides of the above inequality yields Eq.(17). □

A.6 Proof of Lemma 3

Proof. By definition, the diagonal (i,i)-entry of $\mathbf{Q}\otimes\mathbf{Q}$ equals $q_{i',i'}\times q_{i'',i''}$, where

$$i' = \lceil i/n\rceil \text{ and } i'' = [(i-1) \bmod n] + 1.$$

Then, taking ∞-norm on both sides of Eq.(11) yields

$$\begin{aligned}\|\mathbf{M}\|_\infty &= \overbrace{\|\mathbf{I}_{n^2}\|_\infty}^{=1} + \lambda\cdot C_{\text{in}}\cdot\overbrace{\|\mathbf{Q}\otimes\mathbf{Q}\|_\infty}^{\le 1} + (1-\lambda)\cdot C_{\text{out}}\cdot\overbrace{\|\mathbf{P}\otimes\mathbf{P}\|_\infty}^{\le 1}\\ &\le 1 + \lambda\cdot C_{\text{in}} + (1-\lambda)\cdot C_{\text{out}}.\end{aligned} \tag{36}$$

□

[14] The *spectral radius* of $\mathbf{X}$, denoted by $\rho(\mathbf{X})$, is the maximum of the absolute values of the eigenvalues of $\mathbf{X}$.

[15] According to [31, p.670], for a nonnegative matrix $\mathbf{A}\in\mathbb{R}^{n\times n}$, there exists a vector $\mathbf{x}\in\{\mathbf{z}|\mathbf{z}\ge\mathbf{0}$ with $\mathbf{z}\ne\mathbf{0}\}$ such that $\mathbf{A}\mathbf{x} = \rho(\mathbf{A})\mathbf{x}$.

A.7 Proof of Lemma 4

Proof. Let $\mathbf{U} = (\mathbf{U}_1\ \mathbf{U}_2)$, $\mathbf{V} = (\mathbf{V}_1\ \mathbf{V}_2)$ and $\Sigma = \begin{pmatrix} \Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2 \end{pmatrix}$. Substituting on both sides of the *Woodbury formula* $\left(\mathbf{I}_n - \mathbf{U}\mathbf{C}\mathbf{V}^T\right)^{-1} = \mathbf{I}_n + \mathbf{U}\left(\Sigma^{-1} - \mathbf{V}^T\mathbf{U}\right)^{-1}\mathbf{V}^T$ [7] gives

$$\text{LHS} = \left(\mathbf{I}_n - \begin{pmatrix}\mathbf{U}_1\ \mathbf{U}_2\end{pmatrix}\begin{pmatrix}\Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2\end{pmatrix}\begin{pmatrix}\mathbf{V}_1^T \\ \mathbf{V}_2^T\end{pmatrix}\right)^{-1} = \left(\mathbf{I}_n - \mathbf{U}_1\Sigma_1\mathbf{V}_1^T - \mathbf{U}_2\Sigma_2\mathbf{V}_2^T\right)^{-1},$$

$$\text{RHS} = \mathbf{I}_n + \begin{pmatrix}\mathbf{U}_1\ \mathbf{U}_2\end{pmatrix}\left(\begin{pmatrix}\Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2\end{pmatrix}^{-1} - \begin{pmatrix}\mathbf{V}_1^T \\ \mathbf{V}_2^T\end{pmatrix}\begin{pmatrix}\mathbf{U}_1\ \mathbf{U}_2\end{pmatrix}\right)^{-1}\begin{pmatrix}\mathbf{V}_1^T \\ \mathbf{V}_2^T\end{pmatrix}$$

$$= \mathbf{I}_n + \begin{pmatrix}\mathbf{U}_1\ \mathbf{U}_2\end{pmatrix}\begin{pmatrix}{\Sigma_1}^{-1} - \mathbf{V}_1^T\mathbf{U}_1 & -\mathbf{V}_1^T\mathbf{U}_2 \\ -\mathbf{V}_2^T\mathbf{U}_1 & {\Sigma_2}^{-1} - \mathbf{V}_2^T\mathbf{U}_2\end{pmatrix}^{-1}\begin{pmatrix}\mathbf{V}_1^T \\ \mathbf{V}_2^T\end{pmatrix}.$$

Since LHS = RHS, this yields the desired results. □

A.8 Proof of Observation 1

Proof. The correctness of Eq.(26) is based on the two properties of Kroneckor product:

(i) Transpositions are distributive over the Kronecker product, *i.e.,*

$$(\mathbf{V} \otimes \mathbf{V})^T = \mathbf{V}^T \otimes \mathbf{V}^T$$

(ii) Mixed product property, *i.e.,* if $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are matrices of such size that one can form the matrix products $\mathbf{AC}$ and $\mathbf{BD}$, then

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}) \ \text{ with } \ \mathbf{A} = \mathbf{B} = \mathbf{V}^T \ \text{ and } \ \mathbf{C} = \mathbf{D} = \mathbf{U}.$$

Combining the above two Kroneckor properties, Eq.(26) follows immediately. □

A.9 Proof of Observation 2

Proof. Since $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ is the SVD decomposition, it follows that $\mathbf{I}_r = \mathbf{V}^T\mathbf{V}$. By taking vec$(*)$ on both sides and applying the Kroneckor property, we have

$$\text{vec}(\mathbf{I}_r) = \text{vec}(\mathbf{V}^T\mathbf{V}) = (\mathbf{V}^T \otimes \mathbf{V}^T)\text{vec}(\mathbf{I}_n) = (\mathbf{V} \otimes \mathbf{V})^T\text{vec}(\mathbf{I}_n)$$

Hence,

$$(\mathbf{U} \otimes \mathbf{U})\Lambda(\mathbf{V} \otimes \mathbf{V})^T\text{vec}(\mathbf{I}_n) = (\mathbf{U} \otimes \mathbf{U})\Big(\Lambda\text{vec}(\mathbf{I}_r)\Big)$$

Also, by applying the Kronecker property that

$$\text{vec}(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X}) \ \text{ with } \ vec(\mathbf{X}) := \Lambda\text{vec}(\mathbf{I}_r), \ \ \mathbf{B} := \mathbf{U}^T, \ \ \mathbf{A} := \mathbf{U},$$

we can obtain the final result. □

A.10 Proof of Theorem 5

Proof. The P-Rank matrix representation in Eq.(24) implies that

$$\text{vec}(\mathbf{S} - \mathbf{I}_n) = \begin{pmatrix}\mathbf{U}_\mathbf{Q} \otimes \mathbf{U}_\mathbf{Q}\ \mathbf{U}_\mathbf{P} \otimes \mathbf{U}_\mathbf{P}\end{pmatrix}\begin{pmatrix}\Lambda_{11}\ \Lambda_{12} \\ \Lambda_{21}\ \Lambda_{22}\end{pmatrix}\begin{pmatrix}(\mathbf{V}_\mathbf{Q} \otimes \mathbf{V}_\mathbf{Q})^T \\ (\mathbf{V}_\mathbf{P} \otimes \mathbf{V}_\mathbf{P})^T\end{pmatrix}\text{vec}(\mathbf{I}_n). \tag{37}$$

Since

$$\begin{pmatrix}(\mathbf{V}_\mathbf{Q} \otimes \mathbf{V}_\mathbf{Q})^T \\ (\mathbf{V}_\mathbf{P} \otimes \mathbf{V}_\mathbf{P})^T\end{pmatrix}\text{vec}(\mathbf{I}_n) = \begin{pmatrix}({\mathbf{V}_\mathbf{Q}}^T \otimes {\mathbf{V}_\mathbf{Q}}^T)\text{vec}(\mathbf{I}_r) \\ ({\mathbf{V}_\mathbf{P}}^T \otimes {\mathbf{V}_\mathbf{P}}^T)\text{vec}(\mathbf{I}_r)\end{pmatrix} = \begin{pmatrix}\text{vec}(\mathbf{I}_n) \\ \text{vec}(\mathbf{I}_n)\end{pmatrix},$$

it follows from Eq.(37) that

$$\text{vec}(\mathbf{S} - \mathbf{I}_n) = \begin{pmatrix}\mathbf{U}_\mathbf{Q} \otimes \mathbf{U}_\mathbf{Q}\ \mathbf{U}_\mathbf{P} \otimes \mathbf{U}_\mathbf{P}\end{pmatrix}\begin{pmatrix}\Lambda_{11}\ \Lambda_{12} \\ \Lambda_{21}\ \Lambda_{22}\end{pmatrix}\begin{pmatrix}\text{vec}(\mathbf{I}_r) \\ \text{vec}(\mathbf{I}_r)\end{pmatrix}$$

$$
\begin{aligned}
&= (\mathbf{U_Q} \otimes \mathbf{U_Q})\left((\mathbf{\Lambda}_{11}+\mathbf{\Lambda}_{12})\mathrm{vec}(\mathbf{I}_r)\right) + (\mathbf{U_P} \otimes \mathbf{U_P})\left((\mathbf{\Lambda}_{21}+\mathbf{\Lambda}_{22})\mathrm{vec}(\mathbf{I}_r)\right)\\
&= \mathrm{vec}\left(\mathbf{U_Q} \cdot \mathtt{Reshape}\left((\mathbf{\Lambda}_{11}+\mathbf{\Lambda}_{12})\mathrm{vec}(\mathbf{I}_r)\right) \cdot \mathbf{U_Q}^T\right)\\
&\quad + \mathrm{vec}\left(\mathbf{U_P} \cdot \mathtt{Reshape}\left((\mathbf{\Lambda}_{21}+\mathbf{\Lambda}_{22})\mathrm{vec}(\mathbf{I}_r)\right) \cdot \mathbf{U_P}^T\right).
\end{aligned}
$$

Finally, removing vec(∗) on both sides produces the final results.

In addition, the expression of $\mathbf{\Lambda}$ follows immediately by applying Observation 1 to Eq.(24). □

A.11 Proof of Theorem 6

Proof. The total computational time of P-Rank mainly consists of the following three phases:

(i) $O(r^2n)$ time and $O(rn)$ memory to compute four $r \times r$ matrices: $\mathbf{\Theta}_{\mathbf{Q},\mathbf{Q}}, \mathbf{\Theta}_{\mathbf{Q},\mathbf{P}}, \mathbf{\Theta}_{\mathbf{P},\mathbf{Q}}, \mathbf{\Theta}_{\mathbf{P},\mathbf{P}}$,
(ii) $O(r^6)$ time and $O(r^4)$ memory to compute four $r^2 \times r^2$ blocks $(\mathbf{\Lambda}_{1,1}, \mathbf{\Lambda}_{1,2}, \mathbf{\Lambda}_{2,1}, \mathbf{\Lambda}_{2,2})$ of $\mathbf{\Lambda}$, where the entire matrix $\mathbf{\Lambda}$ is of size $2r^2 \times 2r^2$, and
(iii) $O(rn^2)$ time and $O(rn)$ memory to compute $\mathbf{U_Q} \cdot \mathtt{Reshape}\big((\mathbf{\Lambda}_{1,1} + \mathbf{\Lambda}_{1,2})\mathrm{vec}(\mathbf{I}_r)\big) \cdot \mathbf{U_Q}^T$ and $\mathbf{U_P} \cdot \mathtt{Reshape}\big((\mathbf{\Lambda}_{2,1} + \mathbf{\Lambda}_{2,2})\mathrm{vec}(\mathbf{I}_r)\big) \cdot \mathbf{U_P}^T$ column by column online.

Hence, the total computational cost of computing $\mathbf{S}$ is bounded by $O(rn^2 + r^6)$ time and $O(rn + r^4)$ memory space. □

A.12 Proof of Lemma 5

Proof. Recall that the exact P-Rank similarity Eq.(12) can be rewritten as

$$\mathbf{M} \cdot \mathrm{vec}\,(\mathbf{S}) = (1 - \lambda C_{\mathrm{in}} - (1-\lambda)C_{\mathrm{out}}) \cdot \mathrm{vec}\,(\mathbf{I}_n)\,.$$

In contrast, the rank-υ approximation of P-Rank equation satisfies

$$\mathbf{M}_\upsilon \cdot \mathrm{vec}(\hat{\mathbf{S}}_\upsilon) = (1 - \lambda C_{\mathrm{in}} - (1-\lambda)C_{\mathrm{out}}) \cdot \mathrm{vec}(\mathbf{I}_n).$$

To estimate the error ϵ_υ, we combine the above two equations to yield

$$\mathbf{M} \cdot (\mathrm{vec}(\mathbf{S}) - \mathrm{vec}(\hat{\mathbf{S}}_\upsilon)) = (\mathbf{M}_\upsilon - \mathbf{M}) \cdot \mathrm{vec}(\hat{\mathbf{S}}_\upsilon).$$

Let

$$\epsilon_\upsilon := \frac{\|\mathbf{S} - \hat{\mathbf{S}}_\upsilon\|_{\max}}{\|\hat{\mathbf{S}}_\upsilon\|_{\max}} = \frac{\|\mathrm{vec}(\mathbf{S} - \hat{\mathbf{S}}_\upsilon)\|_\infty}{\|\mathrm{vec}(\hat{\mathbf{S}}_\upsilon)\|_\infty}.$$

Since $\mathbf{M}$ is invertible (as proved in Subsection 4.1), pre-multiplying by $\mathbf{M}^{-1}$ and taking ∞-norm on both sides of the above equation produces

$$\epsilon_\upsilon \le \|\mathbf{M}^{-1}\|_\infty \cdot \|\mathbf{M}_\upsilon - \mathbf{M}\|_\infty.$$

According to Lemma 2, we have

$$\|\mathbf{M}^{-1}\|_\infty \le 1/(1 - \lambda C_{\mathrm{in}} - (1-\lambda)\,C_{\mathrm{out}}). \tag{38}$$

By the equivalence of norms, it follows that

$$\|\mathbf{M}_\upsilon - \mathbf{M}\|_\infty \le \sqrt{n}\|\mathbf{M}_\upsilon - \mathbf{M}\|_2.$$

□

A.13 Proof of Lemma 6

Proof. Subtracting Eq.(29) from Eq.(11) and taking 2-norms of both sides yield

$$\|\mathbf{M}_v - \mathbf{M}\|_2 \leq \lambda \cdot C_{\text{in}} \|\mathbf{Q} \otimes \mathbf{Q} - \mathbf{Q}_v \otimes \mathbf{Q}_v\|_2 + (1-\lambda) \cdot C_{\text{out}} \|\mathbf{P} \otimes \mathbf{P} - \mathbf{P}_v \otimes \mathbf{P}_v\|_2.$$

To find an upper bound for $\|\mathbf{Q} \otimes \mathbf{Q} - \mathbf{Q}_v \otimes \mathbf{Q}_v\|_2$, let $\mathbf{Q} = \mathbf{U}_\mathbf{Q} \Sigma_\mathbf{Q} \mathbf{V}_\mathbf{Q}^T$ be a truncated SVD with

$$\Sigma_\mathbf{Q} = diag\left(\sigma_1, \sigma_2, \cdots, \sigma_r\right),$$

and $\mathbf{Q}_v = \mathbf{U}_\mathbf{Q} \Sigma_{\mathbf{Q}_v} \mathbf{V}_\mathbf{Q}^T$ be a truncated SVD with

$$\Sigma_{\mathbf{Q}_v} = diag(\sigma_1, \cdots, \sigma_v, \underbrace{0, \cdots, 0}_{r-v}).$$

Then it follows that

$$\begin{aligned}
&\Sigma_\mathbf{Q} \otimes \Sigma_\mathbf{Q} - \Sigma_{\mathbf{Q}_v} \otimes \Sigma_{\mathbf{Q}_v} \\
&= diag\,(\overbrace{0, \cdots, 0}^{v}, \sigma_1\sigma_{v+1}, \cdots, \sigma_1\sigma_r, \\
&\quad \cdots, \\
&\quad \underbrace{0, \cdots, 0}_{v}, \sigma_v\sigma_{v+1}, \cdots, \sigma_v\sigma_r, \\
&\quad \sigma_{v+1}\sigma_1, \sigma_{v+1}\sigma_2, \cdots, \sigma_{v+1}\sigma_r, \\
&\quad \cdots, \\
&\quad \sigma_r\sigma_1, \sigma_r\sigma_2, \cdots, \sigma_r\sigma_r).
\end{aligned}$$

Since $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ are the nonzero singular values of $\mathbf{Q}$, arranged in non-increasing order, we know by SVD property that

$$\left\|\Sigma_\mathbf{Q} \otimes \Sigma_\mathbf{Q} - \Sigma_{\mathbf{Q}_v} \otimes \Sigma_{\mathbf{Q}_v}\right\|_2 = \sigma_1\sigma_{v+1}. \tag{39}$$

From the truncated SVDs of $\mathbf{Q}$ and $\mathbf{Q}_v$, it follows that

$$\mathbf{Q} \otimes \mathbf{Q} - \mathbf{Q}_v \otimes \mathbf{Q}_v = \left(\mathbf{U}_\mathbf{Q} \otimes \mathbf{U}_\mathbf{Q}\right)\left(\Sigma_\mathbf{Q} \otimes \Sigma_\mathbf{Q} - \Sigma_{\mathbf{Q}_v} \otimes \Sigma_{\mathbf{Q}_v}\right)\left(\mathbf{V}_\mathbf{Q} \otimes \mathbf{V}_\mathbf{Q}\right)^T.$$

Due to the property that the Kronecker product of two orthogonal matrices is orthogonal [13], it follows that $\mathbf{U}_\mathbf{Q} \otimes \mathbf{U}_\mathbf{Q}$ and $\mathbf{V}_\mathbf{Q} \otimes \mathbf{V}_\mathbf{Q}$ are orthogonal. Hence, by SVD and Eq.(39) we have

$$\|\mathbf{Q} \otimes \mathbf{Q} - \mathbf{Q}_v \otimes \mathbf{Q}_v\|_2 = \left\|\Sigma_\mathbf{Q} \otimes \Sigma_\mathbf{Q} - \Sigma_{\mathbf{Q}_v} \otimes \Sigma_{\mathbf{Q}_v}\right\|_2 = \sigma_1\sigma_{v+1}. \tag{40}$$

Similarly, we can obtain

$$\|\mathbf{P} \otimes \mathbf{P} - \mathbf{P}_v \otimes \mathbf{P}_v\|_2 = \bar{\sigma}_1\bar{\sigma}_{v+1} \tag{41}$$

with $\bar{\sigma}_i \quad (i = 1, v+1)$ being the i-th largest singular values of $\mathbf{P}$. Substituting Eqs.(40) and (41) into Eq.(30) yields

$$\begin{aligned}
\|\mathbf{M}_v - \mathbf{M}\|_2 &\leq \lambda \cdot C_{\text{in}} \cdot \overbrace{\|\mathbf{Q} \otimes \mathbf{Q} - \mathbf{Q}_v \otimes \mathbf{Q}_v\|_2}^{=\sigma_1\sigma_{v+1}} + (1-\lambda) \cdot C_{\text{out}} \cdot \overbrace{\|\mathbf{P} \otimes \mathbf{P} - \mathbf{P}_v \otimes \mathbf{P}_v\|_2}^{=\bar{\sigma}_1\bar{\sigma}_{v+1}} \\
&= \lambda C_{\text{in}}\sigma_1\sigma_{v+1} + (1-\lambda) C_{\text{out}}\bar{\sigma}_1\bar{\sigma}_{v+1},
\end{aligned}$$

which completes the proof. □

A.14 Proof of Theorem 8

Proof. The total complexity of DE SR is dominated by the following two steps:

(i) $O(r^6)$ time and $O(r^4)$ memory to compute $\mathbf{\Lambda}$ (in Line 5);
(ii) $O(n^2 r)$ time and $O(rn)$ memory to compute $(\mathbf{U} \cdot \mathbf{\Gamma} \cdot \mathbf{U}^T)$ column by column (in Line 7).

□

A.15 Proof of Theorem 9

Proof. (i) We first present the power series form of $\mathbf{S}$. Due to the undirectedness of a graph, its adjacency matrix is symmetric. Therefore, it follows that

$$\mathbf{Q} = \mathbf{P} = \mathbf{D} \cdot \mathbf{A} \quad \text{with } \mathbf{D} \text{ defined by Eq.(32).} \tag{42}$$

Let $C \triangleq \lambda \cdot C_{\text{in}} + (1-\lambda) \cdot C_{\text{out}}$. Substituting Eq.(42) into P-Rank definition Eq.(10) yields

$$\mathbf{S} = (1-C)\Big(C(\mathbf{DA})\mathbf{S}(\mathbf{DA})^T + \mathbf{I}_n\Big). \tag{43}$$

The recursion of $\mathbf{S}$ in Eq.(43) leads itself to have the following power series form:

$$\mathbf{S} = (1-C)\sum_{k=0}^{+\infty} C^k (\mathbf{DA})^k ((\mathbf{DA})^k)^T. \tag{44}$$

(ii) We next compute $(\mathbf{DA})^k$ in Eq.(44). Observing that $\mathbf{D} \cdot \mathbf{A} = \mathbf{D}^{1/2} \cdot (\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}) \cdot \mathbf{D}^{-1/2}$, we can obtain

$$\begin{aligned}(\mathbf{D} \cdot \mathbf{A})^k &= \mathbf{D}^{1/2}(\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2})\overbrace{\mathbf{D}^{-1/2} \cdot \mathbf{D}^{1/2}}^{=\mathbf{I}}(\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}) \\ &\quad \underbrace{\mathbf{D}^{-1/2} \cdot \mathbf{D}^{1/2}}_{=\mathbf{I}}(\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2})\mathbf{D}^{-1/2} \cdots \\ &= \mathbf{D}^{1/2} \cdot (\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2})^k \cdot \mathbf{D}^{-1/2}. \end{aligned} \tag{45}$$

Due to the symmetry of $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$, it can be decomposed into $\mathbf{U} \cdot \mathbf{\Lambda} \cdot \mathbf{U}^T$ via eigen-decomposition. Therefore, it follows from $\mathbf{U}^T \cdot \mathbf{U} = \mathbf{I}$ that

$$(\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2})^k = (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^T. \tag{46}$$

Substituting Eq.(46) back into Eq.(45) produces

$$(\mathbf{D} \cdot \mathbf{A})^k = \mathbf{D}^{1/2} \cdot \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^T \cdot \mathbf{D}^{-1/2}. \tag{47}$$

(iii) We now express P-Rank similarity matrix $\mathbf{S}$ in terms of $\mathbf{\Lambda}$. Let $\mathbf{\Gamma} = (\Gamma_{i,j})_{r\times r} \triangleq \mathbf{U}^T\mathbf{D}^{-1}\mathbf{U}$. By applying Eq.(47) to Eq.(44), we have

$$\begin{aligned}\tfrac{1}{1-C}\mathbf{S} &= \sum_{k=0}^{+\infty} C^k \cdot \mathbf{D}^{\frac{1}{2}}\mathbf{U} \cdot \mathbf{\Lambda}^k \cdot \mathbf{U}^T\mathbf{D}^{-\frac{1}{2}} \cdot \left(\mathbf{D}^{\frac{1}{2}}\mathbf{U} \cdot \mathbf{\Lambda}^k \cdot \mathbf{U}^T\mathbf{D}^{-\frac{1}{2}}\right)^T \\ &= \mathbf{D}^{\frac{1}{2}}\mathbf{U} \cdot \left(\sum_{k=0}^{+\infty} C^k \cdot \mathbf{\Lambda}^k \cdot \mathbf{\Gamma} \cdot \mathbf{\Lambda}^k\right) \cdot \mathbf{U}^T\mathbf{D}^{\frac{1}{2}} \\ &= \mathbf{D}^{1/2}\mathbf{U} \cdot \mathbf{\Psi} \cdot \mathbf{U}^T\mathbf{D}^{1/2}, \end{aligned}$$

where

$$\mathbf{\Psi} = \sum_{k=0}^{+\infty} C^k \cdot \begin{pmatrix} \left(\Lambda_{1,1}\Lambda_{1,1}\right)^k \Gamma_{1,1} & \cdots & \left(\Lambda_{1,1}\Lambda_{r,r}\right)^k \Gamma_{1,r} \\ \vdots & \ddots & \vdots \\ \left(\Lambda_{r,r}\Lambda_{1,1}\right)^k \Gamma_{r,1} & \cdots & \left(\Lambda_{r,r}\Lambda_{r,r}\right)^k \Gamma_{r,r} \end{pmatrix} = \begin{pmatrix} \frac{\Gamma_{1,1}}{1-C\Lambda_{1,1}\Lambda_{1,1}} & \cdots & \frac{\Gamma_{1,r}}{1-C\Lambda_{1,1}\Lambda_{r,r}} \\ \vdots & \ddots & \vdots \\ \frac{\Gamma_{r,1}}{1-C\Lambda_{r,r}\Lambda_{1,1}} & \cdots & \frac{\Gamma_{r,r}}{1-C\Lambda_{r,r}\Lambda_{r,r}} \end{pmatrix}.$$

□

A.16 Proof of Theorem 10

For Algorithm 3, the total computational cost of UN P-Rank consists of three phases:

(i) In lines 2-3, computing the diagonal $\mathbf{D}$ and $\mathbf{T} = \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ needs $O(m), O(n^2)$ time, and $O(m), O(m)$ memory, respectively.

(ii) In line 4, the EVD of $\mathbf{T}$ into the orthogonal $\mathbf{U}$ and the diagonal $\mathbf{\Lambda}$ requires $O(rn^2)$ time and $O(rn)$ memory.

(iii) In lines 5-7, computing the auxiliary $\mathbf{\Gamma}, \mathbf{V}, \mathbf{\Psi}$ and similarity $\mathbf{S}$ yields $O(r^3), O(rn), O(r^2), O(r^2n + rn^2)$ time, and $O(rn), O(rn), O(r^2), O(nr)$ memory space, respectively. They can be bounded further by $O(rn^2)$ time and $O(rn)$ memory.

Combining (i), (ii) and (iii), the total time of UN P-Rank is in $O(rn^2)$ time and $O(rn)$ memory.