ASAP: Towards Accurate, Stable and Accelerative Penetrating-Rank Estimation on Large Graphs

Xuefei Li[1], Weiren Yu[2,3], Bo Yang[3], and Jiajin Le[3]

[1] Fudan University, China
xuefei.li89@gmail.com
[2] University of New South Wales & NICTA, Australia
weirenyu@cse.unsw.edu.au
[3] Donghua University, China
{yangbo,lejiajin}@dhu.edu.cn

Abstract. Pervasive web applications increasingly require a measure of similarity among objects. Penetrating-Rank (P-Rank) has been one of the promising link-based similarity metrics as it provides a comprehensive way of jointly encoding both incoming and outgoing links into computation for emerging applications. In this paper, we investigate P-Rank efficiency problem that encompasses its accuracy, stability and computational time. (1) We provide an accuracy estimate for iteratively computing P-Rank. A symmetric problem is to find the iteration number K needed for achieving a given accuracy ϵ. (2) We also analyze the stability of P-Rank, by showing that small choices of the damping factors would make P-Rank more stable and well-conditioned. (3) For undirected graphs, we also explicitly characterize the P-Rank solution in terms of matrices. This results in a novel non-iterative algorithm, termed ASAP , for efficiently computing P-Rank, which improves the CPU time from $O(n^4)$ to $O\left(n^3\right)$. Using real and synthetic data, we empirically verify the effectiveness and efficiency of our approaches.

1 Introduction

The study of quantifying structural similarity between vertices in ubiquitous networks has attracted considerable attention over the past decade. Typical structural similarity metrics include Google PageRank, Hyperlink-Induced Topic Search (HITS), Cocitation, Bibliographic Coupling, SimRank and SimFusion (*e.g.,* [1,2,3,4,5,6]).

P-Rank (Penetrating-Rank) is a new similarity measure of this kind, which was initially proposed by Zhao *et al.* [7]. The similarity scores flowing from in-link neighbors of entities are penetrated through their out-link neighbors. Emerging real-world applications of P-Rank include biological networks, collaborative filtering, graph classification, web document ranking, and outlier detection (*e.g.,* [8,9,4,10,5,6]).

In contrast to other similarity measures, P-Rank has become one of the important metrics owing to the following two reasons. (i) P-Rank provides a comprehensive way to jointly explore both in- and out-link relationships with semantic completeness. In comparison, other similarity measures, say SimRank, have the "limited information problem" [5], in which only in-link relationships can be partially exploited. (ii) P-Rank transcends other similarity measures in its most general form. Other measures such as SimRank [5], Amsler [1] can be regarded as just special cases of P-Rank [7].

H. Wang et al. (Eds.): WAIM 2011, LNCS 6897, pp. 415–429, 2011.

Unfortunately, previous work on P-Rank suffers from the following limitations. (i) P-Rank solution is known to converge [7], but a precise accuracy estimation of P-Rank is not given. (ii) No prior work has well studied *the P-Rank condition number*, which indeed has played a paramount role in measuring how much the web graph can change in proportion to small changes in the P-Rank scoring results. (iii) The P-Rank time complexity in [7] retains *quartic* in the worst case for both digraphs and undirected graphs. To the best of our knowledge, there is no efficient P-Rank algorithm specially designed for *undirected graphs*. These practical needs call for an in-depth investigation of P-Rank and its efficiency.

Contributions. This paper studies the P-Rank problems regarding its *accuracy*, *stability* and *computational efficiency*. Our main results are summarized below.

1. We provide an accuracy estimation for P-Rank iteration (Section 3). We find that the number of iterations $K = \lceil \log \epsilon / \log (\lambda \cdot C_{\text{in}} + (1-\lambda) \cdot C_{\text{out}}) \rceil$ suffices to acquire a desired accuracy of ϵ, where λ is a weighting factor, and C_{in} and C_{out} are in- and out-link damping factors, respectively.
2. We introduce the notion of P-Rank *condition number* κ_∞ to investigate the stability issue of P-Rank (Section 4). We show that P-Rank is *well-conditioned* for small choices of the damping factors, by providing a tight *stability bound* for κ_∞.
3. We propose a novel *non-iterative* $O(n^3)$-time algorithm (ASAP) for efficiently computing similarities over undirected graphs (Section 5). We explicitly couch the P-Rank solution *w.r.t.* matrix products in connection with its eigen-problem.
4. We experimentally verify the efficiency of our methods on real and synthetic data (Section 6). We find that P-Rank exponentially converges *w.r.t.* the iteration number. The stability of P-Rank is determined by the damping factors and weighting factors. The ASAP algorithm outperforms baseline algorithms on undirected graphs.

2 Preliminaries

In this section, we first give a brief overview of the P-Rank model. We then present notations and formulations of the P-Rank similarity.

2.1 An Overview of P-Rank

The basic essence of the P-Rank similarity model is distilled from the standard concept of SimRank metric [5]. Concretely, the theme of P-Rank involves three facets below:

1. Two distinct objects are similar if they are referenced by similar objects. (in-link recursion)
2. Two distinct objects are similar if they reference similar objects. (out-link recursion)
3. Every object is maximally similar to itself. (a base case)

Based on these facets, P-Rank scores, as described in its name, flowing from the incoming neighbors of objects are able to penetrate the outgoing neighbors. The concise elegance of the P-Rank intuition makes it one of the useful and cutting-edge structural metrics in the link-based analysis of information networks.

2.2 Notations

symbol	designation	symbol	designation
$\mathcal{G}$	information network	$s(a,b)$	P-Rank score between vertices a and b
$\mathcal{I}(a)$	in-neighbors of vertex a	C_{in} / C_{out}	in-link / out-link damping factor
$\mathcal{O}(a)$	out-neighbors of vertex a	λ	weighting factor
n	number of vertices in $\mathcal{G}$	$\mathbf{A}$	adjacency matrix of $\mathcal{G}$
m	number of edges in $\mathcal{G}$	$\mathbf{S}$	P-Rank similarity matrix of $\mathcal{G}$
K	number of iterations	$\mathbf{I}$	identity matrix

2.3 Formulation of P-Rank Model

We first define the network graph, and then introduce the P-Rank formulation.

Network Graph. A *network* is a labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}; l)$, in which (i) $\mathcal{V}$ is a finite set of vertices. A partition of $\mathcal{V}$ is formed by dividing $\mathcal{V}$ into N disjointed domain-specific parts $\mathcal{V}_i$ $(i = 1, \cdots, N)$ *s.t.* $\mathcal{V} = \bigcup_{i=1}^{N} \mathcal{V}_i$ and $\mathcal{V}_i \cap \mathcal{V}_j = \varnothing$ if $i \neq j$; (ii) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is an edge set, and each vertex-pair $(u, v) \in \mathcal{E}$ denotes an edge from vertex u to v; (iii) $\mathcal{A}$ is a finite alphabet set, with a labeling function $l : \mathcal{V} \to \mathcal{A}$.

Based on the network graph, P-Rank model can be formulated as follows:

P-Rank Similarity. Given a network $\mathcal{G}$, for any two distinct vertices $u, v \in \mathcal{V}$, we define a scoring function $s(u, v) \in [0, 1]$ as follows:

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

$$s(u,v)=\underbrace{\frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)|\,|\mathcal{I}(v)|}\sum_{i=1}^{|\mathcal{I}(u)|}\sum_{j=1}^{|\mathcal{I}(v)|} s\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right)}_{\text{in-link part}} + \underbrace{\frac{(1-\lambda)\cdot C_{\text{out}}}{|\mathcal{O}(u)|\,|\mathcal{O}(v)|}\sum_{i=1}^{|\mathcal{O}(u)|}\sum_{j=1}^{|\mathcal{O}(v)|} s\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right)}_{\text{out-link part}} \tag{2}$$

where (i) $\lambda \in [0, 1]$ balances the contributions of in- 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) $\mathcal{I}_i(u)$ and $\mathcal{O}_i(u)$ are the individual members 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.

It is worth noting that to avoid $s(u, v) = \infty$ when $|\mathcal{I}(\cdot)|$ or $|\mathcal{O}(\cdot)| = 0$, we define: (a) the in-link part of Eq.(2)$= 0$ if $\mathcal{I}(u) \cup \mathcal{I}(v) = \varnothing$; (b) the out-link part of Eq.(2)$= 0$ if $\mathcal{O}(u) \cup \mathcal{O}(v) = \varnothing$; (c) $s(u, v) = 0$, if $(\mathcal{I}(u) \cup \mathcal{I}(v)) \cap (\mathcal{O}(u) \cup \mathcal{O}(v)) = \varnothing$.

P-Rank Iterative Paradigm. To compute the P-Rank similarity $s(u, v)$ of Eq.(2), the conventional approach is to construct an iterative paradigm as follows:

$$s^{(0)}(u,v) = \begin{cases} 0, \text{ if } u \neq v; \\ 1, \text{ if } u = v. \end{cases} \tag{3}$$

For each $k = 0, 1, 2, \cdots$, the $(k+1)$-th iterate $s^{(k+1)}(\cdot,\cdot)$ can be computed as

$$\begin{aligned}
&s^{(k+1)}(u,u) = 1.\\
&s^{(k+1)}(u,v) = \frac{\lambda \cdot C_{\text{in}}}{|\mathcal{I}(u)||\mathcal{I}(v)|} \sum_{i=1}^{|\mathcal{I}(u)|}\sum_{j=1}^{|\mathcal{I}(v)|} s^{(k)}\left(\mathcal{I}_i(u), \mathcal{I}_j(v)\right)\\
&\qquad\qquad + \frac{(1-\lambda)\cdot C_{\text{out}}}{|\mathcal{O}(u)||\mathcal{O}(v)|} \sum_{i=1}^{|\mathcal{O}(u)|}\sum_{j=1}^{|\mathcal{O}(v)|} s^{(k)}\left(\mathcal{O}_i(u), \mathcal{O}_j(v)\right).\\
&s^{(k+1)}(u,v) = \text{the in-link part of Eq.(4), if } \mathcal{O}(u) \cup \mathcal{O}(v) = \varnothing,\\
&s^{(k+1)}(u,v) = \text{the out-link part of Eq.(4), if } \mathcal{I}(u) \cup \mathcal{I}(v) = \varnothing.
\end{aligned} \tag{4}$$

where $s^{(k)}(u,v)$ is the k-th iterative P-Rank score between vertices u and v. It has been proved in [7] that the exact similarity $s(u,v)$ is the supremum of $\{s^{(k)}(u,v)\}_{k=0}^{\infty}$, denoted by $\sup_k\{s^{(k)}(u,v)\}$, *i.e.,*

$$\lim_{k\to\infty} s^{(k)}(u,v) = \sup_{k\geq 0}\{s^{(k)}(u,v)\} = s(u,v) \quad (\forall u, v \in \mathcal{V}). \tag{5}$$

3 Accuracy Estimate on P-Rank Iteration

Based on Eq.(4), the iterative P-Rank similarity sequence $\{s^{(k)}(\cdot,\cdot)\}_{k=0}^{\infty}$ was known to converge monotonically [7]. However, the gap between the k-th iterative similarity $s^{(k)}(\cdot,\cdot)$ and the exact similarity $s(\cdot,\cdot)$ still remains unknown. Practically, quantifying this gap is very important in estimating the accuracy of iteratively computing P-Rank. For instance, given a tolerated error $\epsilon > 0$, one may naturally ask "What is the number of P-Rank iterations needed for achieving such an accuracy?".

This motivates us to study *the P-Rank accuracy estimate problem*. Given a network $\mathcal{G}$, for any iteration number $k = 1, 2, \cdots$, it is to find an upper bound ϵ_k of the gap between the k-th iterative similarity $s^{(k)}(\cdot,\cdot)$ and the exact $s(\cdot,\cdot)$, *s.t.* $|s^{(k)}(u,v) - s(u,v)| \leq \epsilon_k$ for any vertices u and 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

$$|s^{(k)}(u,v) - s(u,v)| \leq \epsilon_k. \quad (\forall k = 0, 1, \cdots,\ \forall u, v \in \mathcal{V}) \tag{6}$$

A detailed proof of Theorem 1 can be found in [11] due to space limitations.

Theorem 1 provides an a-priori estimate for the gap between iterative and exact P-Rank similarity scores, which solely hinges on the weighing factor λ, the in- and out-link damping factors C_{in} and C_{out}, and the number k of iterations. To ensure more accurate P-Rank estimation results, it can be discerned from $\epsilon_k = (\lambda C_{\text{in}} + (1-\lambda)C_{\text{out}})^{k+1}$ $= (\lambda(C_{\text{in}} - C_{\text{out}}) + C_{\text{out}})^{k+1}$ that the 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 $C_{\text{in}} < C_{\text{out}}$, are more preferable.

Example 1. Setting $C_{\text{in}} = 0.6, C_{\text{out}} = 0.4, \lambda = 0.3, k = 5$ will produce the following hight accuracy :

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

Notice that the upper bound in Eq.(6) is attainable. Consider the network $\mathcal{G}_0$ depicted 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 $k = 0$, $|s(u,v) - s^{(k)}(u,v)| = (\lambda C_{\text{in}} + (1-\lambda)C_{\text{out}})^{0+1}$ gives the precise upper bound in Eq.(6).

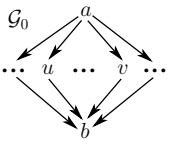


Fig. 1. "=" of Eq.(6) is attainable

Moreover, in the special case of $\lambda = 1$, Eq.(6) can be simplified to the SimRank accuracy estimate problem given in [8]. From this perspective, P-Rank accuracy estimate problem is the extension of Proposition 1 in [8] by jointly considering both in- and out-links of similarity computation.

Conversely, the exponential P-Rank convergence rate in Theorem 1 also implies that the number k of P-Rank iterations needed for attaining a desired accuracy ϵ amounts to

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

4 Stability Analysis of P-Rank Model

In this section, we investigate the important issues of P-Rank sensitivity and stability. We first reformulate the P-Rank formulae in matrix notations (Subsect. 4.1). Using this representation, we then give a rigorous bound of the P-Rank conditional number for its stability analysis (Subsect. 4.2).

4.1 P-Rank Matrix Representation

We start by introducing the following symbols used throughout this section.

For a network $\mathcal{G}$ with n vertices, we denote by (i) $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{n \times n}$ the *adjacency matrix* of $\mathcal{G}$ whose entry $a_{i,j}$ equals 1, if there exists an edge from vertex i to j; or 0, otherwise, (ii) $\mathbf{S} = (s_{i,j}) \in \mathbb{R}^{n \times n}$ the *P-Rank similarity matrix* whose entry $s_{i,j}$ is equal to P-Rank score $s(i,j)$, and (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 defined as follows:

$$q_{i,j} \triangleq \begin{cases} a_{j,i} / \sum_{j=1}^{n} a_{j,i}, & \text{if } \mathcal{I}(i) \neq \varnothing; \\ 0, & \text{if } \mathcal{I}(i) = \varnothing. \end{cases} \qquad p_{i,j} \triangleq \begin{cases} a_{i,j} / \sum_{j=1}^{n} a_{i,j}, & \text{if } \mathcal{O}(i) \neq \varnothing; \\ 0, & \text{if } \mathcal{O}(i) = \varnothing. \end{cases} \tag{7}$$

With the above notations, the P-Rank formulae (1) and (2) can be rewritten as[1]

$$\mathbf{S} = \lambda C_{\text{in}} \cdot \mathbf{Q} \cdot \mathbf{S} \cdot \mathbf{Q}^T + (1-\lambda) C_{\text{out}} \cdot \mathbf{P} \cdot \mathbf{S} \cdot \mathbf{P}^T + (1 - \lambda C_{\text{in}} - (1-\lambda) C_{\text{out}}) \cdot \mathbf{I}_n, \tag{8}$$

Also, dividing both sides of Eq.(8) by $(1 - \lambda C_{\text{in}} - (1-\lambda)C_{\text{out}})$ results in

$$\mathbf{S}' = \lambda C_{\text{in}} \cdot \mathbf{Q} \cdot \mathbf{S}' \cdot \mathbf{Q}^T + (1-\lambda) C_{\text{out}} \cdot \mathbf{P} \cdot \mathbf{S}' \cdot \mathbf{P}^T + \mathbf{I}_n, \text{ and} \tag{9}$$

[1] Although in this case the diagonal entries of $\mathbf{S}$ are not equal to 1, $\mathbf{S}$ still remains diagonally dominant, which ensures that "every vertex is maximally similar to itself". Li *et al.* [12] has showed that this revision for SimRank matrix representation is reasonable, which can be applied in the similar way to P-Rank.

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

Comparing Eq.(8) with Eq.(9), we see that the coefficient $(1 - \lambda C_{\text{in}} - (1-\lambda)C_{\text{out}})$ of $\mathbf{I}_n$ in Eq.(8) merely contributes an overall multiplicative factor to P-Rank similarity. Hence, setting the coefficient to 1 in Eq.(8) still preserves the *relative* magnitude of the P-Rank score though the diagonal entries of $\mathbf{S}$ in this scenario might not equal 1[2].

4.2 Conditional Number of P-Rank

Using the P-Rank matrix model of Eq.(9), we next theoretically analyze the stability of P-Rank, by determining its *conditional number* $\kappa_\infty(\cdot)$ in the infinity norm sense, which has important implications for the sensitivity of P-Rank computation. One complicated factor in P-Rank stability analyses is to bound the conditional number $\kappa_\infty(\cdot)$.

Before giving a formal definition of $\kappa_\infty(\cdot)$, we first introduce the following notations: (i) We denote by $\text{vec}(\mathbf{X}) \in \mathbb{R}^{n^2}$ 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, and (ii) the operator $\otimes$ the *Kronecker product* of two matrices.

Taking $\text{vec}(\cdot)$ on both sides of Eq.(9) and applying the Kronecker property of $(\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{AXB})$ [13], we have

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

Rearranging terms in the above equation produces

$$\underbrace{(\mathbf{I}_{n^2} - \lambda C_{\text{in}}(\mathbf{Q} \otimes \mathbf{Q}) - (1-\lambda)C_{\text{out}}(\mathbf{P} \otimes \mathbf{P}))}_{\triangleq \mathbf{M}} \cdot \underbrace{\text{vec}(\mathbf{S})}_{\triangleq \mathbf{s}} = \underbrace{\text{vec}(\mathbf{I}_n)}_{\triangleq \mathbf{b}}. \tag{10}$$

Notice that Eq.(10) is a *linear* system of n^2 equations in terms of $s_{i,j}$ in nature as it can be reduced in the form of $\mathbf{M} \cdot \mathbf{s} = \mathbf{b}$, as illustrated in Eq.(10).

Based on Eq.(10), we now give a formal definition of *P-Rank conditional number*.

Definition 1 (P-Rank conditional number). *For a network $\mathcal{G}$, let $\mathbf{Q}$ and $\mathbf{P}$ be the one-step backward and forward transition probability matrix of $\mathcal{G}$, respectively, defined by Eq.*(7)*, and let*

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

The P-Rank conditional number of $\mathcal{G}$, denoted by $\kappa_\infty(\mathcal{G})$, is defined as

$$\kappa_\infty(\mathcal{G}) \triangleq \|\mathbf{M}\|_\infty \cdot \|\mathbf{M}^{-1}\|_\infty, \tag{12}$$

where $\|\cdot\|_\infty$ is the ∞-norm that returns the maximum absolute row sum of the matrix.

The conditional number is introduced for being applied to P-Rank stability analysis.

Theorem 2. *Given the network $\mathcal{G}$, for any weighting factor $\lambda \in [0,1]$ and in- and out-link damping factors $C_{in}, C_{out} \in (0,1)$, the P-Rank problem has the following tight bound of conditional number*

$$\kappa_\infty(\mathcal{G}) \leq \frac{1 + \lambda \cdot C_{in} + (1-\lambda) \cdot C_{out}}{1 - \lambda \cdot C_{in} - (1-\lambda) \cdot C_{out}}. \tag{13}$$

[2] In what follows, we shall base our techniques on the P-Rank matrix form of Eq.(9).

For the interest of space, please refer to [11] for a detailed proof of Theorem 2.

Theorem 2 provides a tight upper bound of the P-Rank conditional number $\kappa_\infty(\mathcal{G})$. Intuitively, the value of $\kappa_\infty(\mathcal{G})$ measures how stable the P-Rank similarity score is to the changes either in the link structure of the network $\mathcal{G}$ (by inserting or deleting vertices or edges), or in the damping and weighting factors of $C_{\text{in}}, C_{\text{out}}$ and λ.

To get a feel for how $\kappa_\infty(\mathcal{G})$ affects P-Rank sensitivity, we denote by $\Delta\mathbf{A}$ the updates in $\mathcal{G}$ to the adjacency matrix $\mathbf{A}$, $\Delta\mathbf{M}$ (*resp.* $\Delta\mathbf{s}$) the changes (caused by $\Delta\mathbf{A}$) to $\mathbf{M}$ (*resp.* $\mathbf{s}$) defined in Eq.(10). As Eq.(10) is a linear equation, it is known that

$$\frac{\|\Delta\mathbf{s}\|_\infty}{\|\mathbf{s}\|_\infty} \leq \kappa_\infty(\mathcal{G}) \cdot \frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty} \leq \frac{1+\lambda\cdot C_{\text{in}}+(1-\lambda)\cdot C_{\text{out}}}{1-\lambda\cdot C_{\text{in}}-(1-\lambda)\cdot C_{\text{out}}} \cdot \frac{\|\Delta\mathbf{M}\|_\infty}{\|\mathbf{M}\|_\infty}. \tag{14}$$

This tells that the small $\kappa_\infty(\mathcal{G})$ (*i.e.,* small choices for C_{in} and C_{out}) would make P-Rank stable, implying that a small change $\Delta\mathbf{M}$ in the link structure to $\mathbf{M}$ may not cause a large change $\Delta\mathbf{s}$ in P-Rank scores. Conversely, the large value of $\kappa_\infty(\mathcal{G})$ would make P-Rank *ill-conditioned*.

To see how the weighting factor λ affects $\kappa_\infty(\mathcal{G})$, we compute the partial derivatives of the bound for $\kappa_\infty(\mathcal{G})$ *w.r.t.* λ to get

$$\frac{\partial}{\partial\lambda}\left(\frac{1+\lambda\cdot C_{\text{in}}+(1-\lambda)\cdot C_{\text{out}}}{1-\lambda\cdot C_{\text{in}}-(1-\lambda)\cdot C_{\text{out}}}\right) = \frac{2\,(C_{\text{in}}-C_{\text{out}})}{(1-\lambda\cdot C_{\text{in}}-(1-\lambda)\cdot C_{\text{out}})^2}, \tag{15}$$

which implies that when $C_{\text{in}} > C_{\text{out}}$ (*resp.* $C_{\text{in}} < C_{\text{out}}$), for the increasing 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.

5 An Efficient Algorithm for P-Rank Estimating on Undirected Graphs

In this section, we study the P-Rank optimization problem over undirected networks.

The key idea in our optimization is to maximally use the adjacency matrix $\mathbf{A}$ to characterize the P-Rank similarity $\mathbf{S}$ as a power series in a function form like $\mathbf{S} = \sum_{k=0}^{+\infty} f(\mathbf{A}^k)$. The rationale behind this is that $\mathbf{A} = \mathbf{A}^T$ for undirected networks, implying that there exists a diagonal matrix $\mathbf{D}$ such that $\mathbf{Q} = \mathbf{P} = \mathbf{D}\cdot\mathbf{A}$ in Eq.(8). Hence, computing $\mathbf{S}$ boils down to solving $\sum_{k=0}^{+\infty} f(\mathbf{A}^k)$. For computing $f(\mathbf{A}^k)$, it is far less costly to first diagonalize $\mathbf{A}$ into $\mathbf{\Lambda}$ via eigen-decomposition than to compute $\mathbf{A}^k$ in a brute-force way. Then calculating $f(\mathbf{A}^k)$ reduces to computing the function on each eigenvalue for $\mathbf{A}$.

More specifically, the main result of this section is the following.

Theorem 3. *For undirected networks, the P-Rank similarity problem of Eq.*(8) *can be solvable in* $O(n^3)$ *worst-case time.*

It is worth noting that SimRank over undirected graphs is a special case of the more general P-Rank when $\lambda = 1$ and $C_{\text{in}} = C_{\text{out}}$. In contrast to the $O(n^3 + Kn^2)$-time of

Algorithm 1. ASAP $(\mathcal{G}, \lambda, C_{\text{in}}, C_{\text{out}})$

Input : a labeled undirected network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}; l)$, the weighting factor λ, and in- and out-link damping factors C_{in} and C_{out}.

Output: similarity matrix $\mathbf{S} = (s_{i,j})_{n\times n}$ with $s_{i,j}$ denoting P-Rank score between vertices i and j.

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

2 compute the diagonal matrix $\mathbf{D} = diag(d_{1,1}, d_{2,2}, \cdots, d_{n,n})$ with its entry $d_{i,i} = (\sum_{j=1}^{n} a_{i,j})^{-1}$, if $\sum_{j=1}^{n} a_{i,j} \neq 0$; and $d_{i,i} = 0$, otherwise;

3 compute the auxiliary matrix $\mathbf{T} = \mathbf{D}^{1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{1/2}$

4 decompose $\mathbf{T}$ into the diagonal matrix $\mathbf{\Lambda} = diag(\Lambda_{1,1}, \Lambda_{2,2}, \cdots, \Lambda_{n,n})$ and the orthogonal $\mathbf{U}$ via QR factorization *s.t.* $\mathbf{T} = \mathbf{U} \cdot \mathbf{\Lambda} \cdot \mathbf{U}^T$;

5 compute the auxiliary matrix $\mathbf{\Gamma} = (\Gamma_{i,j})_{n\times n} = \mathbf{U}^T \cdot \mathbf{D}^{-1} \cdot \mathbf{U}$ and $\mathbf{V} = \mathbf{D}^{1/2} \cdot \mathbf{U}$ and the constant $C = \lambda C_{\text{in}} + (1-\lambda) C_{\text{out}}$;

6 compute the matrix $\mathbf{\Psi} = (\psi_{i,j})_{n\times n}$ whose entry $\psi_{i,j} = \Gamma_{i,j}/(1 - C \cdot \Lambda_{i,i} \cdot \Lambda_{j,j})$;

7 compute the P-Rank similarity matrix $\mathbf{S} = (1-C) \cdot \mathbf{V} \cdot \mathbf{\Psi} \cdot \mathbf{V}^T$;

8 **return** $\mathbf{S}$;

SimRank optimization over undirected graphs in our early work [14], this work further optimizes P-Rank time in $O(n^3)$ with no need for iteration.

To show Theorem 3, we first characterize P-Rank elegantly on undirected graphs.

Theorem 4. *For the undirected network $\mathcal{G}$ with n vertices, let (i) $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{n\times n}$ be the adjacency matrix of $\mathcal{G}$, (ii) $\mathbf{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix*[3]*.*

$$\mathbf{D} = diag((\sum_{j=1}^{n} a_{1,j})^{-1}, \cdots, (\sum_{j=1}^{n} a_{n,j})^{-1}), \tag{16}$$

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

Then the P-Rank similarity matrix $\mathbf{S}'$ of Eq.(9) *can be explicitly computed as*[4]*.*

$$\mathbf{S}' = \mathbf{D}^{1/2}\mathbf{U} \cdot \mathbf{\Psi} \cdot \mathbf{U}^T\mathbf{D}^{1/2}, \tag{17}$$

where

$$\mathbf{\Psi} = (\Psi_{i,j})_{n\times n} = \left(\frac{[\mathbf{U}^T\mathbf{D}^{-1}\mathbf{U}]_{i,j}}{1 - (\lambda \cdot C_{in} + (1-\lambda) \cdot C_{out})\, \Lambda_{i,i}\Lambda_{j,j}} \right)_{n\times n}, \text{ and} \tag{18}$$

$[\mathbf{U}^T\mathbf{D}^{-1}\mathbf{U}]_{i,j}$ *denotes the (i,j)-entry of the matrix $\mathbf{U}^T\mathbf{D}^{-1}\mathbf{U}$.*

Due to space limitations, a detailed proof of this theorem can be found in [11].

We next prove Theorem 3 by providing an algorithm for P-Rank computation over undirected networks.

[3] We define $\frac{1}{0} \triangleq 0$, thereby avoiding division by zero when the column/row sum of $\mathbf{A}$ equals 0.

[4] $\mathbf{D}^{1/2}$ is a diagonal matrix whose diagonal entries are the principal square root of those of $\mathbf{D}$.

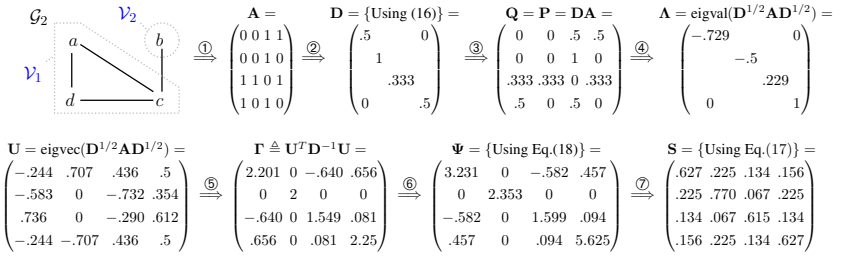


Fig. 2. An example of computing P-Rank over undirected network $\mathcal{G}_2$ via non-iterative method

Algorithm. The algorithm, referred to as ASAP , is shown in Algorithm 1. It takes as input a labeled undirected network $\mathcal{G}$, a weighting factor λ, and in- and out-link damping factors C_{in} and C_{out}; and it returns the P-Rank similarity matrix $\mathbf{S}=(s_{i,j})_{n\times n}$ of all vertex-pairs in $\mathcal{G}$.

The algorithm ASAP works as follows. It first initializes the adjacency matrix $\mathbf{A}$ of the network $\mathcal{G}$ (line 1). Using $\mathbf{A}$, it then compute the auxiliary diagonal matrix $\mathbf{D}$ whose (i,i)-entry equals the reciprocal of the i-th column sum of $\mathbf{A}$, if this reciprocal exists; or 0, otherwise (line 1). ASAP then uses QR eigen-decomposition [13] to factorize $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ as $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, in which all columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$, and all diagonal entries of $\mathbf{\Lambda}$ are the corresponding eigenvalues of $\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ (lines 1-1). Utilizing $\mathbf{U}$ and $\mathbf{\Lambda}$, it calculates $\mathbf{\Psi}$ (lines 1-1) to obtain the similarity matrix $\mathbf{S}$ (lines 1-1), which can be justified by Eqs.(17) and (18).

We now give a running example to show how ASAP computes the P-Rank similarity matrix $\mathbf{S}$ for a given network.

Example 2. Consider a labeled undirected network $\mathcal{G}_2$ with 4 vertices $\mathcal{V}=\mathcal{V}_1\cup\mathcal{V}_2=\{a,c,d\}\cup\{b\}$ and 4 edges $\mathcal{E}=\{(a,c),(a,d),(c,d),(b,c)\}$. Figure 2 depicts the detailed process of computing $\mathbf{S}$ step by step with no need for any iteration. ASAP returns $\mathbf{S}$ as the P-Rank matrix, which is exactly the solution to the P-Rank formula of Eq.(8).

To complete the proof of Theorem 3, we next show that algorithm ASAP has cubic-time complexity bound in the number of vertices.

Complexity. (i) In lines 1-1, computing the diagonal $\mathbf{D}$ and $\mathbf{T}=\mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ takes $O(m)$ and $O(n^2)$ time, respectively, with n and m being the number of vertices and edges in $\mathcal{G}$ respectively. (ii) In line 1, QR factorization of $\mathbf{T}$ into the orthogonal $\mathbf{U}$ and the diagonal $\mathbf{\Lambda}$ requires $O(n^3)$ worse-case time. (iii) In lines 1-1, computing the auxiliary matrices $\mathbf{\Gamma}$, $\mathbf{V}$, $\mathbf{\Psi}$ and the similarity matrix $\mathbf{S}$ yields $O(n^3), O(n^2), O(n^2)$ and $O(n^3)$, respectively, which can be bounded by $O(n^3)$. Combining (i), (ii) and (iii), the total time of ASAP is in $O(n^3)$.

6 Experimental Evaluation

We next present an experimental study of our P-Rank estimating methods. Using real-life and synthetic data, we conducted two sets of experiments to evaluate the accuracy, stability, computational efficiency of our approaches for similarity estimation *v.s.* (a)

the conventional pruning P-Rank iterative paradigm [7] and (b) the memoization-based P-Rank algorithm [8,7].

Experimental setting. We used real-life data and synthetic data.

(1) Real-life data. The real-life data was taken from DBLP [5]. We extracted the 10-year (from 1998 to 2007) author-paper information from the entire DBLP dataset. We picked up papers published on 6 major conferences ("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. We chose the relationship that there is an edge between authors if one author wrote a paper with another. The sizes of these DBLP networks are as follows:

DBLP Data	1998-1999	1998-2001	1998-2003	1998-2005	1998-2007
n	1,525	3,208	5,307	7,984	10,682
m	5,929	13,441	24,762	39,399	54,844

(2) Synthetic data. We also used a C++ boost generator to produce graphs, controlled by two parameters: the number n of vertices, and the number m of edges. We then produced a set of 5 networks (undirected RAND data) by increasing the vertex size n from 100K to 1M with edges randomly chosen.

(3) Algorithms. We have implemented the following algorithms in C++: (a) our algorithm ASAP ; (b) the conventional P-Rank iterative algorithm Iter [7] with the radius-based pruning technique; (c) the memoization-based algorithm Memo [8] applied on P-Rank; (d) SimRank optimized algorithm AUG [14] over undirected graphs.

The experiments were run on a machine with a Pentium(R) Dual-Core (2.0GHz) CPU and 4GB RAM, using Windows Vista. Each experiment was repeated 5 times and the average is reported here.

For ease and fairness of comparison, the following parameters were used as default values (unless otherwise specified). We set the in-link damping factor $C_{\text{in}} = 0.8$, the out-link damping factor $C_{\text{out}} = 0.6$, the weighting factor $\lambda = 0.5$, the total iteration number $k = 10$, the desired accuracy $\epsilon = 0.001$.

Experimental Results. We now present our findings.

Exp-1: Accuracy. In the first set of experiments, we evaluated the accuracy of P-Rank iteration in Eq.(6) using synthetic and real data. We also investigated the impact of in- and out-link damping factors on P-Rank accuracy, using synthetic data.

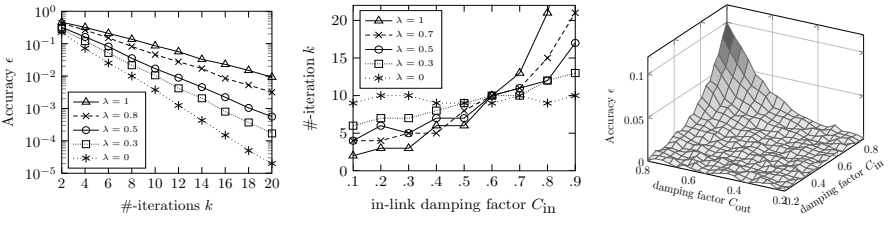
We considered various weighting factors λ from 0 to 1 for 5 RAND networks. For each RAND data with vertex size ranged from 0.2M to 1M, fixing the damping factors $C_{\text{in}} = 0.8$ and $C_{\text{out}} = 0.6$, we varied the number k of P-Rank iterations in Eq.(4) ranged from 2 to 20 for each vertex-pair. Due to space limitations, Figure 3(a) only reports the results over the RAND network (with 1M vertices), which visualizes the P-Rank accuracy *w.r.t.* the number of iterations performed. Here, the accuracy is measured

[5] http://www.informatik.uni-trier.de/~ley/db/

by the absolute difference between the iterative P-Rank and the exact solution[6]. Note that the logarithmic scale is chosen across the y-axis in Figure 3(a) to provide a more illustrative look for the asymptotic rate of P-Rank convergence. For each fixed λ, the downward lines for P-Rank iterations reveal an exponential accuracy as k increases, as expected in Theorem 1. We also observe that the larger λ may dramatically increase the slope of a line, which tells that increasing the weighting factor decreases the rate of convergence for P-Rank iteration.

Using the same RAND , we fixed the desired accuracy $\epsilon = 0.001$ and varied damping factors by increasing C_{in} and C_{out} from 0.1 to 0.9. Fixing $C_{\text{out}} = 0.6$, the result of varying C_{in} is reported in Figure 3(b) (for space constraints, a similar result of varying C_{out} is omitted), in which the x-axis represents the value of in-link damping factor, and y-axis gives the number of iterations needed for attaining the given accuracy ϵ. When $\lambda = 0$, the curve in Figure (b) visually approaches a horizontal line. This is because in this case, P-Rank boils down to an iterative form of the reversed SimRank with no in-link similarity considered, which makes C_{in} insensitive to the final P-Rank score. When $0 < \lambda \leq 1$, k shows a general increased tendency as C_{in} is growing. This tells us that small choices of damping factors may reduce the number of iterations required for a fixed accuracy, and hence, improves the efficiency of P-Rank, as expected.

To evaluate the impact of both C_{in} and C_{out} *w.r.t.* the accuracy, we used the real DBLP data. We only report the result on DBLP 1998-2007 data in Figure 3(c), which shows a 3D shaded surface from the average of accuracy value for all vertex-pairs on z-axis when we fixed $k = 10$ and $\lambda = 0.5$, and varied C_{in} and C_{out} in x-axis and y-axis, respectively. It can be seen that the residual becomes huge only when C_{in} and C_{out} are both increasing to 1; and the iterative P-Rank is accurate when C_{in} and $C_{\text{out}} < 0.6$. This explains why small choices of damping factors are suggested in P-Rank iteration.



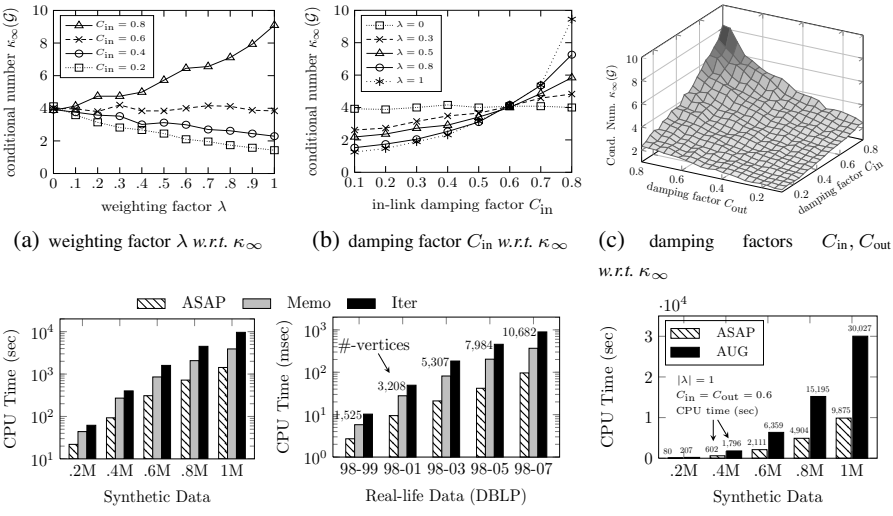
(a) #-iteration k *w.r.t.* accuracy ϵ (b) damping factor C_{in} *w.r.t.* k (c) damping factors $C_{\text{in}}, C_{\text{out}}$ *w.r.t.* ϵ

Fig. 3. Experimental Results on P-Rank Accuracy

Exp-2: Stability. We evaluated P-Rank stability using synthetic data and DBLP .

Fixing the value of out-link damping factor $C_{\text{out}} = 0.6$, we varied the values of C_{in} from 0.2 to 0.8. The result over RAND 1M data is reported in Figure 4(a), in which x-axis indicates various weighting factors λ with their sizes ranged from 0 to 1. Accordingly, varying λ from 0 and 1, Figure 4(b) visualizes the impact of in-link damping factor C_{in} on P-Rank stability when $C_{\text{out}} = 0.6$ is fixed.

[6] To select the P-Rank "exact" solution $s(\cdot,\cdot)$, we used *the Cauchy's criterion for convergence* and regarded the 100th 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) weighting factor λ *w.r.t.* κ_∞ (b) damping factor C_{in} *w.r.t.* κ_∞ (c) damping factors $C_{\text{in}}, C_{\text{out}}$ *w.r.t.* κ_∞

(d) #-vertices n *w.r.t.* CPU time over synthetic and real-life data (e) ASAP *v.s.* AUG on synthetic data

Fig. 4. Experimental Results on P-Rank Stability and Time Efficiency

The results in Figures 4(a) and 4(b) both show that increasing weighting factor λ induces a large P-Rank conditional number when $C_{\text{in}} > 0.6$. Notice that for different C_{in}, there is one common point $(\lambda, \kappa_\infty) = (0, 4)$ of intersection of all curves in Figure 4(a); correspondingly, in the extreme case of $\lambda = 0$, the curve in Figure 4(b) approaches to a horizontal line. These indicate that varying C_{in} when $\lambda = 0$ has no effect on the stability κ_∞ of P-Rank, for in this case only the contribution of out-links is considered for computing P-Rank similarity. When $C_{\text{in}} < 0.6$, however, $\kappa_\infty(\mathcal{G})$ is decreased as λ grows. This tells that small quantities of weighting factor and damping factors yield small P-Rank conditional numbers, and thus make the P-Rank well-conditioned, as expected in Theorem 2.

For real-life datasets, Figure 4(c) comprehensively depicts in 3D view the impacts of both C_{in} and C_{out} on P-Rank stability over DBLP 1M data, in which x- and y-axis represent in- and out-link damping factors respectively, and z-axis stands for the P-Rank conditional number. The result demonstrates that P-Rank is comparatively stable when both C_{in} and C_{out} are small (less than 0.6). However, when C_{in} and C_{out} are approaching to 1, P-Rank is ill-conditioned and not very useful since small perturbations in similarity computation may cause P-Rank scores drastically altered, which carries the risk of producing nonsensical similarity results. In light of this, small choices of damping factors are preferable.

Exp-3: Time Efficiency over Undirected Networks. In the third set of experiments, we used synthetic RAND and the real DBLP data to evaluate the benefits of the time-efficient algorithm ASAP . Figure 4(d) compares the performance of ASAP with those of Memo and Iter on both datasets. We use the logarithmic scale on the CPU time (y-axis). The iteration number for Iter and Memo is set to 10. Note that different time unit is chosen across the vertical axis in the two plots of Fig.4(d) to provide a clear

look for each bar shape. (i) Varying the number of vertices from 200K to 1M, the result on RAND indicates that ASAP outperformed Memo and Iter ; the computational time of ASAP has almost one order of magnitude faster than Iter , *i.e.,* computing P-Rank similarity from the explicit characterization of its solution is efficient. In most cases, there are a considerable amount of repeated iterations for Iter and Memo to reach a fixed-point of P-Rank scores, and these impede their time efficiency in P-Rank computation. (ii) The result on DBLP also shows the running time with respect to the number of nodes for P-Rank estimation when the sizes of DBLP data increased from 1.5K to 10K. In all cases, ASAP performed the best, by taking advantage of its non-iterative paradigm.

To compare the performances of ASAP and AUG , we applied them to compute SimRank similarities over synthetic RAND data, by setting $\lambda = 1$ for ASAP (a special case of P-Rank without out-links consideration). Figure 4(e) reports the result over synthetic RAND data. It can be seen that ASAP runs approx. 3 times faster than AUG though the CPU time of the ASAP and AUG are of the same order of magnitude. The reason is that after eigen-decomposition, AUG still requires extra iterations to be performed in the small eigen-subspace, which takes a significant amount of time, whereas ASAP can straightforwardly compute similarities in terms of eigenvectors with no need for iterations.

7 Related Work

There has been a surge of studies (*e.g.,* [2,15,3,9,4,5,16,6]) on link-based analysis over information networks in recent years. PageRank became popular since the famous result of Page *et al.* [6] was used by the Google search engine for ranking web sites based on link structures. Since then, a host of new ranking algorithms for web pages have been developed. The famous results include the HITS algorithm [2] proposed by Jon *et al.* (now used `www.ask.com`), SimRank [17,10,5], SimFusion [15,4] and P-Rank algorithm [7].

SimRank [5] is a recursive structural similarity measure based on the intuition that "*two objects are similar if they are related to similar objects* ", which extends the Bibliometric Coupling and Co-citation [3,1] beyond the local neighborhood so that the information of the entire network can be exploited globally. A naive iterative approach was initially proposed in [5] to compute the SimRank score with a pruning mechanism, which is in quartic worst-case time. Several optimization problems were investigated for SimRank estimation, including the pair-wise iterative methods [5,8], matrix-based approaches [8,12,17,14] and probabilistic algorithms [18].

More recently, Zhao *et al.* [7] presented a new P-Rank model when noticing the limited information problem of SimRank —the similarity scores are only determined by their in-link relationships. P-Rank measure refines SimRank by jointly considering both in- and out-links of entity pairs. The conventional algorithm for computing P-Rank is based on iterative techniques, which still requires quartic time complexity. To optimize its computational time, a similar memoization approach in [8] can be applied to P-Rank, which reduces its time to be cubic in the worst case. In comparison, our work focuses on the problems of P-Rank accuracy, stability and computational time over undirected graphs.

Closer to this work are [14,8]. A time-efficient algorithm AUG for SimRank computation was proposed on undirected graphs in [14], which is in $O(n^3 + kn^2)$ time. In contrast, we further improve [14] (i) by providing a non-iterative $O(n^3)$-time algorithm that explicitly characterizes the similarity solution, and (ii) by extending SimRank to the general P-Rank measure. An accuracy estimation for SimRank was addressed in [8], it differs from this work in that our focus is on P-Rank estimation, in which the accuracy depends on both in- and out-link damping factors rather than the in-link damping factor alone.

8 Conclusions

In this study, several P-Rank problems were investigated. First, 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. Second, we have introduced the notion of P-Rank conditional number based on the matrix representation of P-Rank. A tight bound of P-Rank conditional number has been provided to show how the weighting factor and the damping factors affect the stability of P-Rank. Finally, we have also devised an $O(n^3)$-time algorithm to deal with the P-Rank optimization problem over undirected networks. Our empirical results have verified the accuracy, stability and computational efficiency of our approaches.

References

1. Amsler, R.: Application of citation-based automatic classification. Technical Report, The University of Texas at Austin, Linguistics Research Center (December 1972)
2. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46(5), 604–632 (1999)
3. Small, H.: 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)
4. Xi, W., Fox, E.A., Fan, W., Zhang, B., Chen, Z., Yan, J., Zhuang, D.: Simfusion: measuring similarity using unified relationship matrix. In: SIGIR (2005)
5. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: KDD, pp. 538–543 (2002)
6. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab (November 1999)
7. Zhao, P., Han, J., Sun, Y.: P-rank: a comprehensive structural similarity measure over information networks. In: CIKM 2009: Proceeding of the 18th ACM Conference on Information and Knowledge Management (2009)
8. Lizorkin, D., Velikhov, P., Grinev, M.N., Turdakov, D.: Accuracy estimate and optimization techniques for simrank computation. VLDB J. 19(1) (2010)
9. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB 2(1) (2009)
10. Antonellis, I., Garcia-Molina, H., Chang, C.C.: Simrank++: query rewriting through link analysis of the click graph. PVLDB 1(1) (2008)
11. Yu, W.: Efficiently computing p-rank similiarty on large networks. Technical report, The University of New South Wales (2011), `http://www.cse.unsw.edu.au/~weirenyu/yu-tr-waim2011.pdf`

12. Li, C., Han, J., He, G., Jin, X., Sun, Y., Yu, Y., Wu, T.: Fast computation of simrank for static and dynamic information networks. In: EDBT (2010)
13. Golub, G.H., Loan, C.F.V.: Matrix computations, 3rd edn. John Hopkins University Press, Baltimore (1996)
14. Yu, W., Lin, X., Le, J.: Taming computational complexity: Efficient and parallel simrank optimizations on undirected graphs. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) WAIM 2010. LNCS, vol. 6184, pp. 280–296. Springer, Heidelberg (2010)
15. Cai, Y., Zhang, M., Ding, C.H.Q., Chakravarthy, S.: Closed form solution of similarity algorithms. In: SIGIR, pp. 709–710 (2010)
16. Yu, W., Lin, X., Le, J.: A space and time efficient algorithm for simrank computation. In: APWeb, pp. 164–170 (2010)
17. He, G., Feng, H., Li, C., Chen, H.: Parallel simrank computation on large graphs with iterative aggregation. In: KDD (2010)
18. Fogaras, D., Rácz, B.: Scaling link-based similarity search. In: WWW (2005)