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

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

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

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

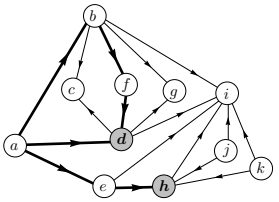
ABSTRACT

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

1. INTRODUCTION

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

Proceedings of the VLDB Endowment, Vol. 7, No. 1



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

Figure 1: Similarities on Citation Graph

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

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

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

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

IRWR: Incremental Random Walk with Restart

Weiren Yu[†‡], Xuemin Lin[♮†]
[†]The University of New South Wales, Australia [‡]NICTA, Australia
[♮]East China Normal University, China
{weirenyu, lxue}@cse.unsw.edu.au

ABSTRACT

Random Walk with Restart (RWR) has become an appealing measure of node proximities in emerging applications *e.g.,* recommender systems and automatic image captioning. In practice, a real graph is typically large, and is frequently updated with small changes. It is often cost-inhibitive to recompute proximities from scratch via *batch* algorithms when the graph is updated. This paper focuses on the incremental computations of RWR in a dynamic graph, whose edges often change over time. The prior attempt of RWR [1] deploys k-dash to find top-k highest proximity nodes for a given query, which involves a strategy to incrementally *estimate* upper proximity bounds. However, due to its aim to prune needless calculation, such an incremental strategy is *approximate*: in $O(1)$ time for each node. The main contribution of this paper is to devise an *exact* and fast incremental algorithm of RWR for edge updates. Our solution, IRWR, can incrementally compute any node proximity in $O(1)$ time for each edge update without loss of exactness. The empirical evaluations show the high efficiency and exactness of IRWR for computing proximities on dynamic networks against its batch counterparts.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Storage and Retrieval

Keywords

Random Walk with Restart; Proximity; Dynamic graph

1. INTRODUCTION

Measuring node proximities in graphs is a key task of web search. Due to various applications in recommender systems and social networks, many proximity metrics have come into play. For instance, Brin and Page [2] invented PageRank to determine the ranking of web pages. Jeh and Widom [3] proposed SimRank to assess node-to-node proximities. Random Walk with Restart (RWR) [4] is one of such useful proximity metrics for ranking nodes in order of relevance to a query node. In RWR, the proximity of node u *w.r.t.* query node q is defined as the limiting probability that a random surfer, starting from q, and then iteratively either moving to one of its out-neighbors with probability weighted by the edge weights, or restarting from q with probability c, will eventually arrive at node u. Recently, RWR has received increasing attention (*e.g.,* for collaborative filtering [1] and image labeling [5]) since it can fairly capture the global structure of graphs, and relations in interlinked networks [6].

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

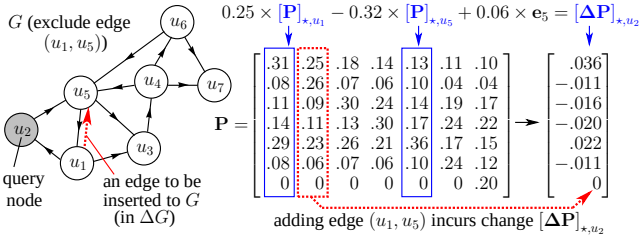


Figure 1: Computing RWR Incrementally

Prior RWR computing methods are based on static graphs, which is costly: Given a graph $G(V, E)$, and a query $q \in V$, k-dash [1] yields, in the worse case, $O(|V|^2)$ time and space, which, in practice, can be bounded by $O(|E| + |V|)$, to find top-k highest proximity nodes. B_LIN and NB_LIN [4] need $O(|V|^2)$ time and space for computing all node proximities. In general, real graphs are often constantly updated with small changes. This calls for the need for incremental algorithms to compute proximities. We state the problem below:

Problem (INCREMENTAL UPDATE FOR RWR)
Given a graph G, proximities $\mathbf{P}$ for G, changes ΔG to G, a query node q, and a restarting probability $c \in (0, 1)$.
Compute changes to the proximities *w.r.t.* q exactly.

Here, $\mathbf{P}$ is a proximity matrix whose entry $[\mathbf{P}]_{i,j}$ denotes the proximity of node i *w.r.t.* query j, and ΔG is comprised of a set of edges to be inserted into or deleted from G.

In contrast with the existing *batch* algorithms [1, 4] that recompute the updated proximities from scratch, our incremental algorithm can exploit the dynamic nature of graphs by pre-computing proximities only once on the entire graph via a batch algorithm, and then *incrementally* computing their changes in response to updates. The response time of RWR can be greatly improved by maximal use of previous computation, as shown in Example 1.

EXAMPLE 1. *Figure 1 depicts a graph G, taken from [1]. Given the query u_2, the old proximities $\mathbf{P}$ for G, and $c = 0.2$, we want to compute new proximities w.r.t. u_2 when there is an edge (u_1, u_5) inserted into G, denoted by ΔG. The existing methods, k-dash and B_LIN, have to recompute the new proximities in $G \cup \Delta G$ from scratch, without using the previously computed proximities in G, which is costly.*

Fast Incremental SimRank on Link-Evolving Graphs

Weiren Yu [†,♭], Xuemin Lin [†], Wenjie Zhang [†]

[†] *The University of New South Wales, Sydney, Australia* [♭] *NICTA, Australia*
{weirenyu, lxue, zhangw}@cse.unsw.edu.au

***Abstract*—SimRank is an arresting measure of node-pair similarity based on hyperlinks. It iteratively follows the philosophy that 2 nodes are similar if they are referenced by similar nodes. Real graphs are often large, and links constantly evolve with small changes over time. This paper considers fast incremental computations of SimRank on link-evolving graphs. The prior approach [1] to this issue factorizes the graph via the singular value decomposition (SVD) first, and then incrementally maintains this factorization for link updates at the expense of exactness. Consequently, all node-pair similarities are estimated in $O(r^4n^2)$ time on a graph of n nodes, where r is the target rank of the low-rank approximation, which is not negligibly small in practice.**

In this paper, we propose a novel fast incremental paradigm. (1) We characterize the SimRank update matrix $\Delta\mathbf{S}$, in response to every link update, via a rank-one Sylvester matrix equation. By virtue of this, we devise a fast incremental algorithm computing similarities of n^2 node-pairs in $O(Kn^2)$ time for K iterations. (2) We also propose an effective pruning technique capturing the "affected areas" of $\Delta\mathbf{S}$ to skip unnecessary computations, without loss of exactness. This can further accelerate the incremental SimRank computation to $O(K(nd+|\mathsf{AFF}|))$ time, where d is the average in-degree of the old graph, and $|\mathsf{AFF}|$ $(\leq n^2)$ is the size of "affected areas" in $\Delta\mathbf{S}$, and in practice, $|\mathsf{AFF}| \ll n^2$. Our empirical evaluations verify that our algorithm (a) outperforms the best known link-update algorithm [1], and (b) runs much faster than its batch counterpart when link updates are small.

I. Introduction

With many recent eye-catching advances of the Internet, link analysis has become a common and important tool for web data management. Due to the proliferative applications (*e.g.,* link prediction, recommender systems, citation analysis), a surge of link-based similarity measures have come into play. For instance, Brin and Page [2] proposed a very successful relevance metric called Google PageRank to rank web pages. Jeh and Widom [3] introduced a novel measure, SimRank, to assess structural similarity between nodes based on hyperlinks. Sun *et al.* [4] invented PathSim to quantify node proximity for heterogeneous graphs. In these emerging similarity measures, SimRank has stood out as an arresting one over the last decade, due to its succinct and iterative philosophy that "two nodes are similar if they are referenced by similar nodes", coupled with the base case that "every node is maximally similar to itself". This recursion not only allows SimRank to capture the global structure of the graph, but also equips SimRank with appealing mathematical insight that has inspired research in recent years. For example, Fogaras and Rácz [5] interpreted SimRank as the first meeting time of the coalescing path-pair random walks. Li *et al.* [1] harnessed an elegant matrix equation to depict the closed form of SimRank.

Nevertheless, the batch computation of SimRank is costly: $O(Kd'n^2)$ time for all node-pairs [6], where K is the number of iterations, and $d' \leq d$ (d is the average graph in-degree). In general, real graphs are often large, with links constantly evolving with minor changes. This is particularly evident in *e.g.,* co-citation networks, web graphs, and social networks. As a statistical example [7], there are 5%–10% links updated every week in a web graph. It is rather expensive to reassess similarities for all pairs of nodes from scratch when the graph is updated. Fortunately, we observe that when link updates are small, the affected areas for SimRank updates are often small as well. With this comes the need for incremental algorithms computing changes to SimRank in response to link updates, to skip unnecessary recomputations. Hence, we investigate the following problem in this paper.

Problem (Incremental SimRank Computation)
Given a graph G, the similarities $\mathbf{S}$ for G, the link changes ΔG [1] to G, and the damping factor $C \in (0,1)$.
Compute the changes $\Delta\mathbf{S}$ to the similarities $\mathbf{S}$.

In contrast with the work on batch SimRank computation, the study on incremental SimRank for link updates is limited. Indeed, due to the recursive nature of SimRank, it is hard to identify "affected areas" for incrementally updating SimRank. To the best of our knowledge, there is only one work [1] by Li *et al.* who gave an interesting method for finding the SimRank changes in response to link updates. Their idea is to factorize the backward transition matrix $\mathbf{Q}$ [2] of the original graph into $\mathbf{U}\cdot\mathbf{\Sigma}\cdot\mathbf{V}^T$ [3] via the singular value decomposition (SVD) first, and then incrementally estimate the updated matrices of $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}^T$ for link changes at the expense of exactness. As a result, updating the similarities of all node-pairs entails $O(r^4n^2)$ time without guaranteed accuracy, where r $(\leq n)$ is the target rank of the low-rank approximation[4], which is not always negligibly small in practice, as illustrated in the following example.

Example 1. *Fig. 1 is a citation graph G (a fraction of* DBLP*) where each edge depicts a reference from one paper to another. Assume G is updated by adding an edge (i,j), denoted by ΔG (see the dash arrow). Using the damping factor $C = 0.8$ [5], we want to compute SimRank scores in the new graph $G \cup \Delta G$. The existing method by Li et al. (see Algorithm 3 in [1]) first decomposes the old $\mathbf{Q} = \mathbf{U}\cdot\mathbf{\Sigma}\cdot\mathbf{V}^T$ as a precomputation step, then, when edge (i,j) is added, it incrementally updates the old $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^T$, and utilizes their updated versions to obtain*

[1] ΔG consists of a sequence of edges to be inserted/deleted.

[2] In the notation of [1], the backward transition matrix $\mathbf{Q}$ is denoted as $\tilde{\mathbf{W}}$, which is the row-normalized transpose of the adjacency matrix.

[3] We use $\mathbf{X}^T$ (instead of $\tilde{\mathbf{X}}$ in [1]) to denote the transpose of matrix $\mathbf{X}$.

[4] According to [1], using our notations, $r \leq \text{rank}(\mathbf{\Sigma} + \mathbf{U}^T \cdot \Delta\mathbf{Q} \cdot \mathbf{V})$, where $\Delta\mathbf{Q}$ is the changes to $\mathbf{Q}$ for link updates.

[5] According to [3], the damping factor C is empirically set around 0.6–0.8, which indicates the rate of decay as similarity flows across edges.

Towards Efficient SimRank Computation on Large Networks

Weiren Yu [†1], Xuemin Lin [†2], Wenjie Zhang [†3]

[†] *The University of New South Wales & NICTA , Australia*

[1,2,3] {weirenyu, lxue, zhangw}@cse.unsw.edu.au

***Abstract*—SimRank has been a powerful model for assessing the similarity of pairs of vertices in a graph. It is based on the concept that two vertices are similar if they are referenced by similar vertices. Due to its self-referentiality, fast SimRank computation on large graphs poses significant challenges. The state-of-the-art work [16] exploits partial sums memorization for computing SimRank in $O(Kmn)$ time on a graph with n vertices and m edges, where K is the number of iterations. Partial sums memorizing can reduce repeated calculations by caching part of similarity summations for later reuse. However, we observe that computations among different partial sums may have duplicate redundancy. Besides, for a desired accuracy ϵ, the existing SimRank model requires $K = \lceil \log_C \epsilon \rceil$ iterations [16], where C is a damping factor. Nevertheless, such a geometric rate of convergence is slow in practice if a high accuracy is desirable.**

In this paper, we address these gaps. (1) We propose an adaptive clustering strategy to eliminate partial sums redundancy (*i.e.,* duplicate computations occurring in partial sums), and devise an efficient algorithm for speeding up the computation of SimRank to $O(Kd'n^2)$ time, where d' is typically much smaller than the average in-degree of a graph. (2) We also present a new notion of SimRank that is based on a differential equation and can be represented as an exponential sum of transition matrices, as opposed to the geometric sum of the conventional counterpart. This leads to a further speedup in the convergence rate of SimRank iterations. (3) Using real and synthetic data, we empirically verify that our approach of partial sums sharing outperforms the best known algorithm by up to one order of magnitude, and that our revised notion of SimRank further achieves a 5X speedup on large graphs while also fairly preserving the relative order of original SimRank scores.

I. Introduction

Identifying similar objects based on hyperlink structure is a fundamental operation for many web mining tasks. Examples include web page ranking [3], hypertext classification (KNN) [13], graph clustering (K-means) [5], and collaborative filtering [11], [17]. In the last decade, with the overwhelming number of objects on the Web, there is a growing need to be able to automatically and efficiently assess the similarity of these objects on large graphs. Indeed, the Web has huge dimensions and continues to grow rapidly— more than 5% of new objects are created per week [4]. As a result, similarity assessment on web objects is apt to become obsolete very shortly. In light of this, it is imperative for similarity assessment to get a fast computational speed on large graphs.

Amid the existing similarity metrics, SimRank [11] has emerged as a powerful tool for assessing structural similarities between objects. Similar to the well-known PageRank [3], SimRank scores depend merely on the link structure of the Web, independent of the textual content of objects. The main difference between the two models is the scoring mechanism. PageRank assigns an authority weight for each object, whereas SimRank assigns a similarity score between two objects. SimRank was first proposed by Jeh and Widom [11], and has gained enormous popularity for its success in many areas such as bibliometrics [14], top-K search [13], and recommender systems [1]. This reveals the importance of SimRank as an effective measure. The intuition underlying SimRank is a subtle recursion that "two vertices are similar if their incoming neighbors are similar", together with the base case that "every vertex is maximally similar to itself" [11]. Due to this self-referential concept, conventional algorithms for computing SimRank have an iterative nature. The sheer size of the Web has presented striking challenges to fast SimRank computing. The best known algorithm proposed by Lizorkin *et al.* [16] (hereafter referred to as psum-SR) requires $O(Kmn)$ time ($O(Kn^3)$ in the worst case) for K iterations, where n and m denote the number of vertices and edges, respectively.

The beauty of psum-SR algorithm [16] resides in the following three observations. (1) *Essential nodes selection* may eliminate the computation of a fraction of node pairs with a-priori zero scores. (2) *Partial sums memorizing* can effectively reduce repeated calculations of the similarity among different node pairs by caching part of similarity summations for later reuse. (3) *A threshold setting* on the similarity enables a further reduction in the number of node pairs to be computed. Particularly, the second observation of *partial sums memorizing* plays a paramount role in greatly speeding up the computation of SimRank from the naive $O(Kd^2n^2)$ [11] to $O(Kdn^2)$, [1] where d is the number of average in-degrees in a graph.

Before shedding light on the limitations of psum-SR [16], let us first revisit the central idea of partial sums memorizing, as illustrated in the following example.

Example 1. Consider a paper citation network $\mathcal{G}$ in Fig. 1a, where each vertex represents a paper, and an edge a citation. For any vertex a, we denote by $\mathcal{I}(a)$ the set of in-neighbors of a. Individual element in $\mathcal{I}(a)$ is denoted as $\mathcal{I}_i(a)$. Let $s(a,b)$ be the SimRank similarity between vertices a and b. In what follows, we want to compute $s(a,b)$ and $s(a,d)$ in $\mathcal{G}$.

Before partial sums memorizing is introduced, a naive way is to sum up the similarities of all in-neighbors $(\mathcal{I}_i(a), \mathcal{I}_j(b))$ of (a,b) for computing $s(a,b)$, and to sum up the similarities

[1]The degree sum formula $n \cdot d = m$ implies that $O(Kmn)$ time in [16] is equivalent to $O(Kdn^2)$.

SimFusion+: Extending SimFusion Towards Efficient Estimation on Large and Dynamic Networks

Weiren Yu[†‡], Xuemin Lin[†♯], Wenjie Zhang[†], Ying Zhang[†], Jiajin Le[♮]
[†]The University of New South Wales, Australia [♯]East China Normal University, China
[‡]NICTA, Australia [♮]Donghua University, China
{weirenyu, lxue, zhangw, yingz}@cse.unsw.edu.au lejiajin@dhu.edu.cn

ABSTRACT

SimFusion has become a captivating measure of similarity between objects in a web graph. It is iteratively distilled from the notion that "the similarity between two objects is reinforced by the similarity of their related objects". The existing SimFusion model usually exploits the *Unified Relationship Matrix* (URM) to represent latent relationships among heterogeneous data, and adopts an iterative paradigm for SimFusion computation. However, due to the row normalization of URM, the traditional SimFusion model may produce the trivial solution; worse still, the iterative computation of SimFusion may not ensure the global convergence of the solution. This paper studies the revision of this model, providing a full treatment from complexity to algorithms. (1) We propose SimFusion+ based on a notion of the *Unified Adjacency Matrix* (UAM), a modification of the URM, to prevent the trivial solution and the divergence issue of SimFusion. (2) We show that for any vertex-pair, SimFusion+ can be performed in $O(1)$ time and $O(n)$ space with an $O(km)$-time precomputation done only once, as opposed to the $O(kn^3)$ time and $O(n^2)$ space of its traditional counterpart, where n, m, and k denote the number of vertices, edges, and iterations respectively. (3) We also devise an incremental algorithm for further improving the computation of SimFusion+ when networks are dynamically updated, with performance guarantees for similarity estimation. We experimentally verify that these algorithms scale well, and the revised notion of SimFusion is able to converge to a non-trivial solution, and allows us to identify more sensible structure information in large real-world networks.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Storage and Retrieval; G.2.2 [**Graph Theory**]: Discrete Mathematics

Keywords

Similarity Computation, SimFusion, Web Ranking Algorithm

1. INTRODUCTION

The conundrum of measuring similarity between objects based on hyperlinks in a graph has fueled a growing interest in the fields of information retrieval. This problem is also known as "link-based analysis" or "structural similarity search", and it has been extensively studied by different communities with a proliferation of emerging applications. Examples include collaborative filtering, hyper-text classification, graph clustering and proximity query processing. Recently, while the scale of the Web has dramatically increased our need to produce large graphs, the study of efficiently computing object similarity on such large graphs becomes a desideratum. In practice, an effective similarity measure should not only correlate well with human intuition but also scale well for large amounts of data.

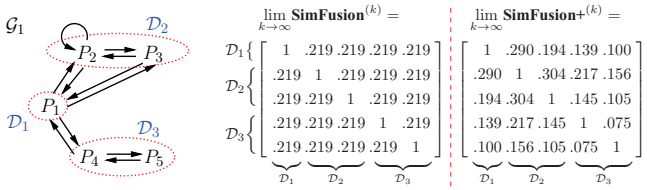


Figure 1: Trivial SimFusion on Heterogeneous Domain

Among the existing metrics, SimFusion [1] can be regarded as one of the attractive ones on account of the following reasons. (i) Similar to PageRank [2] and SimRank [3], SimFusion is based on hyperlinks and follows the reinforcement assumption that "the similarity between objects is reinforced by the similarity of their related objects", which is fairly intuitive and conforms to our basic understandings. (ii) Unlike other measures (*e.g.,* PageRank and SimRank) that explore the linkage patterns merely from a single data space [2–4], SimFusion has the extra benefits of incorporating both inter- and intra-relationships from multiple data spaces in a unified manner to measure the similarity of heterogeneous data objects. (iii) SimFusion offers more intuitive and flexible ways of assigning weighting factors to each data space that reflects their relative importance, as opposed to the PageRank and SimRank measures that need to determine a damping factor. (iv) SimFusion provides a general-purpose framework for measuring structural similarity in a recursive fashion; other well-known measures, such as Co-Citation [5] and Coupling [6] are just special cases of SimFusion.

However, existing work on SimFusion has the following problems. Firstly, although the basic intuition behind the SimFusion model is appealing, it seems inappropriate to use the *Unified Relationship Matrix* (URM) to represent the relationships of heterogeneous objects. The main problem is that, according to the definition of URM $\mathbf{L}$ in [1], the sum of each row of $\mathbf{L}$ is always equal to 1. Since the product of $\mathbf{L}$ and the matrix $\mathbf{1}$ whose entries are all ones is equal to the matrix $\mathbf{1}$ of all ones, there always exists a trivial solution $\mathbf{S} = \mathbf{1}$ to the original SimFusion formula $\mathbf{S} = \mathbf{L} \cdot \mathbf{S} \cdot \mathbf{L}^T$ [1], as illustrated in Example 1. The same phenomena of yielding such a trivial solution may occur in our experimental results in Section 6. To address this issue, we shall revise the original SimFusion model.

EXAMPLE 1 (TRIVIAL SOLUTION). *Figure 1 depicts a graph $\mathcal{G}_1$ partly extracted from Cornell CS Department. Each vertex P_i*

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

World Wide Web (2012) 15:327–353
DOI 10.1007/s11280-010-0100-6

A space and time efficient algorithm for SimRank computation

Weiren Yu · Wenjie Zhang · Xuemin Lin · Qing Zhang · Jiajin Le

Received: 27 August 2010 / Revised: 19 November 2010 /
Accepted: 19 November 2010 / Published online: 7 December 2010

Abstract SimRank has become an important similarity measure to rank web documents based on a graph model on hyperlinks. The existing approaches for conducting SimRank computation adopt an iteration paradigm. The most efficient deterministic technique yields $O(n^3)$ worst-case time per iteration with the space requirement $O(n^2)$, where n is the number of nodes (web documents). In this paper, we propose novel optimization techniques such that each iteration takes $O(\min\{n \cdot m, n^r\})$ time and $O(n+m)$ space, where m is the number of edges in a web-graph model and $r \leq \log_2 7$. In addition, we extend the similarity transition matrix to prevent random surfers getting stuck, and devise a pruning technique to eliminate impractical similarities for each iteration. Moreover, we also develop a reordering technique combined with an over-relaxation method, not only speeding up the convergence rate of the existing techniques, but achieving I/O efficiency as well. We conduct

The work was supported by ARC Grants DP0987557 and DP0881035 and Google Research Award. We also appreciate the general support from NICTA.

W. Yu (✉) · W. Zhang · X. Lin
School of Computer Science & Engineering, University of New South Wales,
Sydney, NSW 2052, Australia
e-mail: weirenyu@cse.unsw.edu.au, ywr0708@hotmail.com

W. Zhang
e-mail: zhangw@cse.unsw.edu.au

X. Lin
e-mail: lxue@cse.unsw.edu.au

Q. Zhang
E-Health Research Center, Australia CSIRO ICT Center,
Herston, Queensland 4029, Australia
e-mail: qing.zhang@csiro.au

J. Le
School of Computer Science & Technology, Donghua University, Shanghai, China
e-mail: lejiajin@dhu.edu.cn

On the Efficiency of Estimating Penetrating Rank on Large Graphs

Weiren Yu[1], Jiajin Le[2], Xuemin Lin[1], and Wenjie Zhang[1]

[1] University of New South Wales & NICTA, Australia
{weirenyu,lxue,zhangw}@cse.unsw.edu.au
[2] Donghua University, China
lejiajin@dhu.edu.cn

Abstract. P-Rank (Penetrating Rank) has been suggested as a useful measure of structural similarity that takes account of both incoming and outgoing edges in ubiquitous networks. Existing work often utilizes memoization to compute P-Rank similarity in an iterative fashion, which requires *cubic* time in the worst case. Besides, previous methods mainly focus on the deterministic computation of P-Rank, but lack the probabilistic framework that scales well for large graphs. In this paper, we propose two efficient algorithms for computing P-Rank on large graphs. The first observation is that a large body of objects in a real graph usually share similar neighborhood structures. By merging such objects with an explicit low-rank factorization, we devise a *deterministic* algorithm to compute P-Rank in *quadratic* time. The second observation is that by converting the iterative form of P-Rank into a matrix power series form, we can leverage the random sampling approach to *probabilistically* compute P-Rank in *linear* time with provable accuracy guarantees. The empirical results on both real and synthetic datasets show that our approaches achieve high time efficiency with controlled error and outperform the baseline algorithms by at least one order of magnitude.

1 Introduction

Structural similarity search that ranks objects based on graph hyperlinks is a major tool in the fields of data mining. This problem is also known as link-based analysis, and it has become popularized in a plethora of applications, such as nearest neighbor search [26], graph clustering [27], and collaborative filtering [9]. For example, Figure 1 depicts a recommender system, in which person (A) and (B) purchase itemsets {egg, pancake, sugar} and {egg, pancake, flour}, respectively. We want to identify similar users and similar items.

Existing link-based approaches usually take advantage of graph structures to measure similarity between vertices. Each object (*e.g.,* person, or item) can be regarded as a vertex, and a hyperlink (*e.g.,* purchase relationship) as a directed edge in a graph. Then a scoring rule is defined to compute similarity between vertices. Consider the well-known SimRank scoring rule [18] "two vertices are similar if they are referenced (have incoming edges) from similar vertices" in Figure 1. We can see that the items sugar and egg are similar as they are purchased by the same person (A). In spite of its worldwide popularity [1, 4, 6, 18, 24, 27], SimRank has the "limited information problem" — it only takes incoming edges into account while ignoring outgoing links [26]. For instance, person (A) and (B) have the SimRank score zero as they have no incoming edges. This

A. Ailamaki and S. Bowers (Eds.): SSDBM 2012, LNCS 7338, pp. 231–249, 2012.

2010 12th International Asia-Pacific Web Conference

A Space and Time Efficient Algorithm for SimRank Computation

Weiren Yu
Dept. of Computer Science & Technology
Donghua University
Shanghai, China
ywr0708@mail.dhu.edu.cn

Xuemin Lin
School of Computer Science & Engineering
University of New South Wales
NSW, Australia
lxue@cse.unsw.edu.au

Jiajin Le
Dept. of Computer Science & Technology
Donghua University
Shanghai, China
lejiajin@dhu.edu.cn

***Abstract*—SimRank has been proposed to rank web documents based on a graph model on hyperlinks. The existing techniques for conducting SimRank computation adopt an iteration computation paradigm. The most efficient technique has the time complexity $O\left(n^3\right)$ with the space requirement $O\left(n^2\right)$ in the worst case for each iteration where n is the number of nodes (web documents). In this paper, we propose novel optimization techniques such that each iteration takes the time $O\left(\min\left\{n\cdot m, n^r\right\}\right)$ and requires space $O\left(n+m\right)$ where m is the number of edges in a web-graph model and $r \leq \log_2 7$. We also show that our algorithm accelerates the convergence rate of the existing techniques. Moreover, our algorithm not only reduces the time and space complexity of the existing techniques but is also I/O efficient. We conduct extensive experiments on both synthetic and real data sets to demonstrate the efficiency and effectiveness of our iteration techniques.**

***Keywords*-Graph Similarity; SimRank; Link-based Analysis; Optimal Algorithms;**

I. Introduction

Recently, the complex hyperlink-based similarity search has attracted considerable attention in the field of Information Retrieval. One of the promising measures is the SimRank similarity with applications to search engine ranking and document corpora clustering. SimRank is a recursive refinement of co-citation measure that computes similarity by common neighbours alone [1]. The intuitive model for SimRank measure is based on random walk over a web-graph like Google PageRank [2]. The SimRank similarity between two pages is defined recursively as the average similarity between their neighbours, along with the base case that a page is maximally similar to itself. Unlike many other domain-specific measures that require human-built hierarchies, SimRank can be used in any domain in combination with traditional textual similarity to produce an overall similarity measure [3], [4].

Motivations: For the efficient SimRank computation, it is desirable to have optimization techniques that improve the time and space complexity of the SimRank algorithm. The idea of approximating SimRank scores has been studied in [3] based on *Monte Carlo method*. This computation model is inherently stochastic. However, with respect to the *non-probabilistic* SimRank iterative computation, little work has been done to establish a theoretical foundation of the optimization. The straightforward iterative SimRank computation has the time complexity $O\left(Kn^4\right)$ in the worst case and requires the space $O\left(n^2\right)$ [1]. The bottleneck mainly lies in high computational complexity. To the best of our knowledge, there is only one research work in [5] concerning *deterministic* methods for SimRank optimization. In that work SimRank computation takes the time $O\left(n^3\right)$ per iteration in the worst case with the space requirement $O\left(n^2\right)$, which has yet been regarded as the most efficient technique in *non-probabilistic* SimRank iteration.

Contributions: In this paper, we investigate the optimal algorithms that can further improve the efficiency of SimRank computation. We provide theoretical guarantee for our methods and present experimental results. The main contributions of this paper are summarized below:

- We introduce a matrix representation and storage schemes for SimRank model to reduce space requirement from $O\left(n^2\right)$ to $O\left(m+n\right)$ with time complexity from $O\left(n^3\right)$ to $O\left(\min\left\{n\cdot m, n^r\right\}\right)$ in the worst case, where $r \leq \log_2 7$.
- We develop optimization techniques for minimizing the matrix bandwidth for SimRank computation, which may improve the I/O efficiency of SimRank iteration.
- We show a successive over-relaxation method for SimRank computation to significantly accelerate the convergence rate of the existing technique.

Organizations: The rest of the paper is organized as follows. In the next section, the problem definition for SimRank is formally introduced. In Sect. III, a solution framework for SimRank optimization techniques is established. In Sect. IV, three optimization techniques for SimRank computation are suggested; the time and space complexity of the proposed algorithm is analyzed. In Sect. V, the experimental results are reported on the efficiency of our methods over synthetic and real-life data sets. The related work appears in Sect. VI and Sect. VII concludes the paper.

II. Preliminaries

In this section, the formal definition of SimRank is given and some notations are presented. The material in this section recalls Jeh's previous work [1].

A. Problem Definition

Given a directed graph $\mathcal{G} = (V, E)$, where each node in V represents a web page and a directed edge $\langle a, b\rangle$ in E corresponds to a hyperlink from page a to b, we can derive *a node-pair graph* $\mathcal{G}^2 \triangleq \left(V^2, E^2\right)$, where

(i) $\forall\,(a,b) \in V^2 \;\; if \;\; a, b \in V$;
(ii) $\forall\,\langle (a_1,b_1),(a_2,b_2)\rangle \in E^2 \;\; if \;\; \langle a_1,a_2\rangle, \langle b_1,b_2\rangle \in E$.

On a node-pair graph $\mathcal{G}^2$, we formally define a similarity function measured by SimRank score.

Definition 1 (SimRank similarity): Let $s : V^2 \to [0,1] \subset \mathbb{R}$ be a real-valued function on $\mathcal{G}^2$ defined by

$$s(a,b) = \begin{cases} 1, & a = b; \\ \frac{c}{|I(a)||I(b)|} \sum\limits_{j=1}^{|I(b)|} \sum\limits_{i=1}^{|I(a)|} s\left(I_i(a), I_j(b)\right), & I(a), I(b) \neq \varnothing; \\ 0, & otherwise. \end{cases} \tag{1}$$

where $c \in (0,1)$ is a constant decay factor , $I(a)$ denotes all in-neighbours of node a , $|I(a)|$ is the cardinality of $I(a)$, an individual member of $I(a)$ is referred to as $I_i(a) \quad (1 \leq i \leq |I(a)|)$, then $s(a,b)$ is called *SimRank similarity score between node a and b.*

The underlying intuition behind SimRank definition is that "*two pages are similar if they are referenced by similar pages*". Figure 1 visualizes the propagation of SimRank similarity in $\mathcal{G}^2$ from node to node, which corresponds to the propagation from pair to pair in $\mathcal{G}$, starting with the singleton node $\{4,4\}$. Since a unique solution to the SimRank recursive equation (1) is reached by iteration to a

978-0-7695-4012-2/10 $26.00 © 2010 IEEE
DOI 10.1109/APWeb.2010.42

Taming Computational Complexity: Efficient and Parallel SimRank Optimizations on Undirected Graphs

Weiren Yu[1,2], Xuemin Lin[2], and Jiajin Le[1]

[1] Donghua University, Shanghai 201620, China
ywr0708@hotmail.com, lejiajin@dhu.edu.cn
[2] University of New South Wales, NSW 2052, Australia
lxue@cse.unsw.edu.au

Abstract. SimRank has been considered as one of the promising link-based ranking algorithms to evaluate similarities of web documents in many modern search engines. In this paper, we investigate the optimization problem of SimRank similarity computation on undirected web graphs. We first present a novel algorithm to estimate the SimRank between vertices in $O\left(n^3 + K \cdot n^2\right)$ time, where n is the number of vertices, and K is the number of iterations. In comparison, the most efficient implementation of SimRank algorithm in [1] takes $O\left(K \cdot n^3\right)$ time in the worst case. To efficiently handle large-scale computations, we also propose a parallel implementation of the SimRank algorithm on multiple processors. The experimental evaluations on both synthetic and real-life data sets demonstrate the better computational time and parallel efficiency of our proposed techniques.

1 Introduction

SimRank is a useful and important similarity measure exploiting the relationships between vertices (web documents) on web graphs. It has been widely studied in the literature [2,3,4,1,5,6,7]. As a multi-step generalization of co-citation [8,9], SimRank similarity is based on the recursive concept that *two different vertices are similar if their neighbors are similar*. SimRank has broad applications involving "find-similar-document" query, search engine optimization, graph clustering, etc.

Existing techniques for SimRank computation can be distinguished into two categories: (i) *probabilistic* method [3,4] that estimates SimRank by the expected value $s\left(a,b\right) = \mathbb{E}\left(c^{\tau(a,b)}\right)$, where $\tau_{(a,b)}$ is a random variable denoting the first meeting time for vertices a and b, and $c \in (0,1)$ is a decay factor; (ii) *deterministic* method [1,5,6,7] that computes SimRank iteratively for finding a fixed point of the SimRank function $s\left(a,b\right)$. Since the latter approach produces better accuracy with high time complexity compared to the probabilistic method, there has been a growing interest in SimRank deterministic optimization over the recent years. To the best of our knowledge, there are several interesting pieces of work which can efficiently reduce the time complexity of SimRank *deterministic* computation. One work [1] is mainly based on *a partial sums function* reducing the computational time from $O\left(Kn^4\right)$ to $O\left(Kn^3\right)$ in the worst case. Another work [7] oriented on *matrix representations* takes $O\left(K \min\left(mn, n^r\right)\right)$ time, where m is the number of edges, and $r \in (2, \log_2 7]$ is a positive constant. Li et al.

L. Chen et al. (Eds.): WAIM 2010, LNCS 6184, pp. 280–296, 2010.

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.