UniWalk: Unidirectional Random Walk Based Scalable SimRank Computation over Large Graph

Junshuai Song, Xiongcai Luo, Jun Gao, Chang Zhou, Hu Wei, and Jeffery Xu Yu

Abstract—SimRank is an important measure of vertex-pair similarity according to the structure of graphs. Although progress has been achieved, existing methods still face challenges to handle large graphs. Besides huge index construction and maintenance cost, existing methods may require considerable search space and time overheads in the online SimRank query. In this paper, we design a Monte Carlo based method, UniWalk, to enable the fast top-k SimRank computation over large undirected graphs. UniWalk directly locates the top-k similar vertices for any single source vertex u via R sampling paths originating from u, which avoids selecting candidate vertex set $\mathcal{C}$ and the following $O(|\mathcal{C}|R)$ bidirectional sampling paths. We also devise a path enumeration strategy to improve the SimRank precision by using path probabilities instead of path frequencies when sampling, a space-efficient method to reduce intermediate results, and a path-sharing strategy to lower the redundant path sampling cost for multiple source vertices. Furthermore, we extend UniWalk to existing distributed graph processing frameworks to improve its scalability. We conduct extensive experiments to illustrate that UniWalk has high scalability, and outperforms the state-of-the-art methods by orders of magnitude.

Index Terms—SimRank, Monte Carlo, random walk, distributed graph processing

1 Introduction

SimRank is an effective and domain-independent structural similarity measurement between two vertices in a graph [1], [2], [3], [4], [5]. Intuitively, SimRank follows the idea that *two objects are similar if they are related to similar objects* [2]. Given source vertices, the SimRank queries can yield the SimRank scores between the source vertices and all other vertices [4], or locate the top-k similar vertices for the source vertices [5], [6]. This work belongs to the latter category, since many applications require only very similar vertices [5], [6].

Although SimRank is well-studied, it is still prohibitively expensive due to its recursively dependency computation [3], [4], [5], especially in a large graph. Roughly speaking, existing approaches to SimRank can be categorized into straightforward and Monte Carlo methods approaches. Originally, SimRank is defined recursively, which can be computed iteratively until a fixed point is reached. That is, to compute all-pair SimRank scores, the straightforward method [2] needs $O(|V|^4K)$ time cost and $O(|V|^2)$ intermediate space in the worst case, where $|V|$ is the total number of vertices, and K is the number of iterations. The straightforward method obviously cannot process large graphs, even if some optimization strategies are adopted, like reusing partial sum of intermediate results [7] or considering top-k similar results [2].

Monte Carlo methods for SimRank, which yield results with lower overheads [8], have been received growing interests recently [4], [5], [6]. SimRank score between the vertex u and v is also interpreted as the expected meeting distance between u and v [2], which can be simulated by sufficient L-length bidirectional random walks (*BiWalk* for short) starting from u and v respectively. As only the sampling paths originated from u and v are needed in the single-pair SimRank computation, the cost in computing a single-pair SimRank score is then independent of the graph size (i.e., the number of vertices). In addition, the error bound of results computed by Monte Carlo methods exponentially declines with respect to the number of sampling paths[8]. These two factors make it possible to compute SimRank scores in a large graph via Monte Carlo methods.

Although Monte Carlo methods are promising, they still face challenges. Existing methods usually cache indices like fingerprint tree [8], one-way graph [5], etc, to improve the online query performance. These pre-computed results incur high indexing time and space overheads, as well as the maintenance cost in handling updates of the graph. In addition, the existing methods need huge search space using BiWalk in the online query phase. Take the top-k SimRank for a single source u as an example. A naive method requires $O(|V|R)$ sampling paths, in which R bidirectional random walks are performed between u and each vertex in V. It still costs high even if a distance based optimization strategy [6] is adopted, which selects partial "near" vertices into

- *J. Song, X. Luo, and J. Gao are with the Key Laboratory of High Confidence Software Technologies, Ministry of Education & School of EECS, Peking University, Haidian Qu, Beijing Shi 100080, China. E-mail: {songjunshuai, luoxiongcai, gaojun}@pku.edu.cn.*
- *C. Zhou and H. Wei are with the Alibaba Group, Hangzhou 311121, China. E-mail: {ericzhou.zc, kongwang}@alibaba-inc.com.*
- *J.X. Yu is with the Department of System Engineering, Chinese University of Hong Kong, Sha Tin, Hong Kong. E-mail: yu@se.cuhk.edu.hk.*

Manuscript received 19 May 2017; revised 16 Nov. 2017; accepted 24 Nov. 2017. Date of publication 4 Dec. 2017; date of current version 30 Mar. 2018. (Corresponding author: Jun Gao.)
Recommended for acceptance by Y. Xia.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2779126

candidates $\mathcal{C}$ first. Such a strategy still requires $O(|\mathcal{C}|)$ times BiWalk, each of which needs $O(R)$ sampling paths. Furthermore, it is not trivial to set a proper distance threshold. A threshold that results in fewer candidates can make the SimRank computation faster but may lose correct results.

Another issue lies in the fact that the existing methods result in the long response time or high memory consumption in distributed clusters [9], [10]. As the overheads of SimRank may exceed the computation capability of a single computer, distributed SimRank is inevitable on large graphs. At the same time, the synchronous overhead and network cost become major cost when the existing methods are extended to distributed processing frameworks.

To overcome the limitations of existing methods, we have proposed a UniWalk approach [11] to top-k SimRank computation. Compared with the existing BiWalk, UniWalk has the following advantages: it does not require indexing or candidate selection phase; it needs much fewer sampling paths in the online query phase to locate the top-k similar vertices for a single source vertex; it enables high parallelism and easy computation sharing, and supports the single-source, multiple-sources, and all-pair SimRank computation adaptively.

In this paper, we make following substantial extensions. The original UniWalk, like other Monte-Carlo methods, may have high variance of SimRank scores when the sampling paths are limited, and such an imprecision may be enlarged by the rectified factor used in the UniWalk. In order to address such an issue, we introduce a path enumeration method in Section 4.1, which computes SimRank using path probabilities instead of path frequencies when sampling. Such a strategy can achieve higher precision, and reduce redundant path sampling cost, while maintaining the efficiency and scalability of the original method. We also conduct experiments to compare UniWalk with other competitors in terms of precision over large graphs. In summary, our main contributions are as below:

1) We propose a novel unidirectional random walk, UniWalk, to compute the top-k SimRank for given source vertices S. With a rectified factor, UniWalk can simulate the original BiWalk in computing SimRank scores. The time cost of UniWalk (excluding sorting cost for the final top-k results) is $O(|S|RL^2)$, where R is the number of $2L$-length sampling paths. Usually, R and L do not vary much. Thus, the cost of UniWalk is approximately linear to the number of source vertices. (Section 3)
2) We further introduce three optimization strategies, including a path enumeration strategy to achieve more precise results and avoid redundant sampling cost, an M-times candidates strategy to relax the space requirement, and a path-sharing strategy for multiple sources to lower path sampling cost. We also devise a distributed UniWalk method, and illustrate that UniWalk can easily leverage existing distributed graph processing frameworks to improve the scalability. (Section 4)
3) We conduct extensive experiments to evaluate UniWalk in terms of precision, efficiency and scalability on 8 real-life graphs and a sequence of Random and Powerlaw graphs. The experiments show that UniWalk can achieve high precision, scalability, and outperform the existing methods by orders of magnitude. (Section 5)

The remainder of this paper is organized as follows. We first review preliminaries in Section 2. Then we describe UniWalk in Section 3, and devise its optimizations and distributed extensions in Section 4. Section 5 reports experimental results in term of the precision, efficiency and scalability. We review related works in Section 6, and conclude the paper in Section 7.

2 Preliminaries

In this section, we first review graph, SimRank, and the basic Monte Carlo estimation for SimRank. We then describe distributed graph processing frameworks. Finally, we formulate the problem.

2.1 Graph and SimRank

This paper studies SimRank computation on undirected graphs. Formally, a graph is denoted by $G = (V, E)$, where V is a vertex set, and E is an edge set. For each vertex u, $u.id$ represents its identifier, and $N(u)$ contains all neighbor vertices of u. Correspondingly, $|N(u)|$ is the degree of u. A path $p = v_0 \rightsquigarrow v_l$ is a sequence of vertex $\langle v_0, \ldots, v_l \rangle$, where $(v_i, v_{i+1}) \in E$ for $0 \leq i < l$. p can be represented by $v_0 \leftrightsquigarrow v_l$ in undirected graphs. The length $len(p)$ of p is the total number of edges in a path. The vertex at the index i in a path p can be referred by $p[i]$. For simplicity, $p.start$ is the start vertex, and $p.end$ is the end vertex of p.

SimRank [2], proposed by Glen Jeh and Jennifer Widom in 2002, is an important and effective similarity measurement between vertices. SimRank can be computed iteratively with Equation (1). Here, $s(u, v)$ denotes the SimRank score between vertices u and v, C is a decay factor, $I(u)$ is the set of u's incoming neighbors, and $|I(u)|$ means the cardinality of $I(u)$. The formula says that $s(u, v)$ is an averaged similarities of all combinations of their incoming neighbors. $|I(u)|$ ($|I(v)|$) in the formula above can be replaced by $|N(u)|$ ($|N(v)|$), respectively, in undirected graphs

$$s(u,v) = \begin{cases} 1 & u = v \\ \frac{C}{|I(u)||I(v)|} \sum_{i \in I(u), j \in I(v)} s(i,j) & otherwise. \end{cases} \quad (1)$$

Fig. 1 shows an example bipartite graph about users and items. SimRank captures the similarity of two vertices in a more reasonable way than other measurements [1], [12]. For example, both similarities $s(Jack, James)$ and $s(Jack, Kate)$ are 0 with the co-citation measurement [12], which counts common neighbors for two vertices. However, *Jack* is more similar to *James* than to *Kate*, as *Computer* is more similar to *Health monitor* than to *Fish oil*. SimRank can capture the similarities when the number of iterations is larger than 1.

SimRank $s(u, v)$ can also be interpreted as the expected meeting distance of two walkers, which start from u and v respectively, and move step by step randomly and simultaneously [2]. Take Fig. 1 as an example. Two paths $1 \rightarrow 2 \rightarrow 3$ and $3 \leftarrow 6 \leftarrow 5$ can be used in computing the SimRank score $s(1, 5)$ between vertex 1 and 5. By enumerating all possible situations, the SimRank score between two vertices u and v can be computed by the expected meeting distance in

$$s(u,v) = \sum_{p: u \rightsquigarrow t \leftsquigarrow v} prob(p) C^{len(p)/2}. \quad (2)$$