Reverse Top-k Search using Random Walk with Restart *

Adams Wei Yu[†§], Nikos Mamoulis[§], Hao Su[‡]
[†]School of Computer Science, Carnegie Mellon University
[§]Department of Computer Science, The University of Hong Kong
[‡]Computer Science Department, Stanford University
weiyu@cs.cmu.edu, nikos@cs.hku.hk, haosu@cs.stanford.edu

ABSTRACT

With the increasing popularity of social networks, large volumes of graph data are becoming available. Large graphs are also derived by structure extraction from relational, text, or scientific data (e.g., relational tuple networks, citation graphs, ontology networks, protein-protein interaction graphs). Node-to-node proximity is the key building block for many graph-based applications that search or analyze the data. Among various proximity measures, random walk with restart (RWR) is widely adopted because of its ability to consider the global structure of the whole network. Although RWR-based similarity search has been well studied before, there is no prior work on reverse top-k proximity search in graphs based on RWR. We discuss the applicability of this query and show that its direct evaluation using existing methods on RWR-based similarity search has very high computational and storage demands. To address this issue, we propose an indexing technique, paired with an on-line reverse top-k search algorithm. Our experiments show that our technique is efficient and has manageable storage requirements even when applied on very large graphs.

1. INTRODUCTION

Graph is a fundamental model for capturing the structure of data in a wide range of applications. Examples of real-life graphs include, social networks, the Web, transportation networks, citation graphs, ontology networks, and protein-protein interaction graphs. In most applications, a key concept is the node-to-node proximity, which captures the relevance between two nodes in a graph. A widely adopted proximity measure, due to its ability to capture the global structure of the graph, is *random walk with restart* (RWR). RWR proximity from node u to node v, is the probability for a random walk starting from u to reach v after infinite time; at any transition there is a chance $(0 < < 1)$ that the random walk restarts at u. Compared to other measures (like shortest path distance), a significant advantage of RWR is that it takes into account all the possible paths between two nodes. Other merits of RWR is that it can model the multi-faceted relationship between two nodes [13] and that RWR is *stable* to small changes in the graph [20]. RWR has been successfully applied in the search engine Google [21] to rank the importance of web pages. In addition, several other measures build upon RWR, including Personalized Pagerank [12], ObjectRank [5], Escape Probability [25], and PathSim [23].

Many search and analysis tasks rely on proximity computations. These include, citation analysis in bibliographical graphs [14], link prediction in social networks [19], graph clustering [2], and making recommendations [16]. The *top-k RWR proximity* query retrieves the k nodes with the highest proximity from a given query node q in a graph. This problem has been investigated previously and efficient solutions have been proposed for it (e.g., [11, 3, 10]).

In this paper, we study the *reverse top-k RWR proximity* query: given a node q, find all the nodes that have q in their top-k RWR proximity sets. Reverse top-k queries can be used for detection of *spam* nodes in a graph. Search engines, such as Google, aggregate the RWR proximities from all other nodes to one node in a single value, known as PageRank. Thus, the proximity from web page u to v can be interpreted as the PageRank contribution that u makes to v. When a node q is suspected to be a spam web page, one could run a reverse top-k search on q, and find out the pages which give one of their top-k contributions to q. If the answer set contains a large proportion of web pages already labeled as spam, then q is likely to be a spam too. As another application, consider an author in a co-authorship network who wishes to find the set of people that regard himself as the one of their most important direct or indirect collaborators. The reverse top-k result can be used for identifying the likelihood of successful collaborations in the future. The size of an author's reverse top-k list is also an indicator of his popularity in the community. Finally, in a product co-purchase graph, a reverse top-k query of a product q can identify which products influence the buying of q. One can leverage this information to promote q in future transactions.

To the best of our knowledge, there is no previous work on reverse top-k RWR-based search in large graphs. In addition, extending solutions for top-k RWR search to compute reverse top-k queries is not trivial. Specifically, while for a top-k search we only need to find the top-k proximity set of *a single* node q, the reverse top-k search must compute the top-k sets of *all* nodes in the graph and check whether q appears in each of them. Therefore, a reverse top-k query is substantially more expensive than top-k RWR search. Figure 1 illustrates a toy graph of 6 nodes and the entire proximity matrix $\boldsymbol{P}$ computed from it; the i-th column $\boldsymbol{p}_i$ of $\boldsymbol{P}$ contains the proximity values *from* node i *to* all nodes in the graph (e.g., the proximity from node 1 to node 3 is 0.12). In each column $\boldsymbol{p}_i$, the $k = 2$ largest entries are shaded; these indicate the results of a top-2 query from node i (e.g., the top-2 query from node 3 returns nodes 2 and 3). Observe that for any given node q, to answer a top-

*Supported by grant HKU 715413E from Hong Kong RGC. Work done while the first author was with HKU.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/. Obtain permission prior to any use beyond those covered by the license. Contact copyright holder by emailing info@vldb.org. Articles from this volume were invited to present their results at the 40th International Conference on Very Large Data Bases, September 1st - 5th 2014, Hangzhou, China.
Proceedings of the VLDB Endowment, Vol. 7, No. 5
Copyright 2014 VLDB Endowment 2150-8097/14/01.

Figure 1: Example graph and its proximity matrix

2 query, we only have to compute and access the values of a single column, whereas a reverse top-2 query for a node requires finding all the shaded entries in the 1st row (e.g., the reverse top-2 query for node 1 returns nodes 1, 2, and 5). To find whether an entry is shaded or not, we have to compute the entire matrix P and rank the values in each column. Computing the whole proximity matrix is both time and space-consuming, especially for large graphs.

We propose a reverse top-k query evaluation framework which alleviates this issue. In a nutshell, our approach computes (at a pre-processing step) from the graph G (having $|V|$ nodes) a graph index, which is based on $K \times |V|$ matrix, containing in each column v the K largest approximate proximity values from v to any other nodes in G. K is application-dependent and represents the highest value of k in a practical query. At each column v of the index, the approximate values are lower bounds of the K largest proximity values from v to all other nodes, computed after adapting and partially executing Berkhin's Bookmark Coloring Algorithm (BCA) [7]. Given the graph index and a reverse top-k query q ($k \leq K$), we prove that the exact proximities from any node v to query q can be efficiently computed by applying the power method. By comparing these with the corresponding lower bounds taken from the k-th row of the graph index, we are able to determine which nodes (i.e., columns of P) are certainly not in the reverse top-k result of q. For some of the remaining nodes, we may also be able to determine that they are certainly in the reverse top-k result, based on derived upper bounds for the k-th largest proximity value from them. Finally, for any candidate that remains, we progressively refine its approximate proximities, until based on its lower or upper bound we can determine if it should be in the result. The proximities refined during query processing can be updated into the graph index, making its values progressively more accurate for future queries.

Our contributions can be summarized as follows:

- We study for the first time reverse top-k proximity queries based on RWR in large graphs.
- We propose a dynamically refined, space-efficient index structure, which supports reverse top-k query evaluation. The index is paired with an efficient online query algorithm, which prunes a large number of nodes that are definitely in or not in the reverse top-k result and minimizes the required refinement for the remaining candidates.
- A side contribution of our online algorithm is a proof that we can apply the power method for computing the exact proximities from all nodes to a given node q. This result can serve as a module of any applications that need to compute RWR proximities to a given node.
- We conduct an experimental study demonstrating the efficiency of our framework, as well as the effectiveness of the reverse top-k RWR query in real graph applications.

The remainder of this paper is organized as follows. Section 2 provides a definition for the RWR-based proximity vector which captures the proximities from a given node to all other nodes and reviews methods for computing it. The reverse top-k RWR proximity search problem is formalized in Section 3 and the baseline brute force solution is discussed. In Section 4, we present our solution which is experimentally evaluated in Section 5. In Section 6, we briefly discuss previous work related to reverse top-k RWR proximity search. Finally, Section 7 concludes the paper.

General	
A	Transition probability matrix
P	Proximity matrix
p_u	Proximity vector from node u to other nodes
e_u	Unit vector having $e_u(u) = 1$, and $e_u(v) = 0 \ \forall v \neq u$
$p_u^{k_{max}}$	The k-th largest entry of p_u
BCA starting from node u	
p_u^t (r_u^t)	Retained (residue) ink distribution at iteration t
s_u^t (w_u^t)	Ink accumulated at hubs (non-hubs) at iteration t
$\hat{p}_u$ ($\hat{p}_u^t$)	Descending ranked list of p_u (p_u^t)
lb_u^t (ub_u^t)	Lower (upper) bound of $\hat{p}_u^t(k)$

Table 1: Main notations

2. PRELIMINARIES

In this section, we first provide definitions for the RWR proximity matrix of a graph and the proximity vectors of nodes in it. Then, we review the Bookmark Coloring Algorithm (BCA) [7] for computing the RWR proximity from a given node to all other nodes, based on which our offline index is built. For a matrix M, $m_{:j}$ or $m_{:j}$ denotes its j-th column, $m_{i:}$ denotes its i-th row, and $m_{i;j}$ denotes the element of its i-th row and j-th column. For a vector v, $v(i)$ denotes its i-th entry and $v(1:i)$ denotes its first i entries. Table 1 summarizes the main symbols used in the paper.

2.1 Definitions

Let $G = (V; E)$ be a directed graph, with a set $V = \{1; 2; \ldots; n\}$ of vertices and a set $E \subseteq V \times V$ of edges. Let $n = |V|$; $m = |E|$, $A = [a_1; a_2; \ldots; a_n] \in R^{n \times n}$ be the column-stochastic transition probability matrix, and $OD(i)$ be the out-degree of node i. We assume that $a_{i;j} = \frac{1}{OD(j)}$ if edge $j \rightarrow i$ exists and $a_{i;j} = 0$, otherwise[1]. In other words, the RWR transition probability from node j to any of its out-neighbors i only depends on the out-degree of j (i.e., all out-neighbors are equally likely to be visited). For a given node u, the RWR proximity values from it to other nodes is the solution of the following linear system w.r.t. p_u:

$$p_u = (1 - \alpha)Ap_u + \alpha e_u; \quad (1)$$

where $p_u \in R^n$ is the proximity vector of node u, with $p_u(v)$ denoting the proximity from u to v; $e_u \in R^n$ is a unit vector having $e_u(u) = 1$ and all other values set to 0, and $\alpha \in [0; 1]$ denotes the restart probability in RWR (typically, $\alpha = 0.15$). The analytical solution is

$$p_u = P e_u; \quad (2)$$

where $P = \alpha(I - (1 - \alpha)A)^{-1} = [p_1; p_2; \ldots; p_n]$ is called the proximity matrix. In fact, P can also be used to compute the Pagerank (pr) and any Personalized Pagerank ($ppr(\cdot)$) vector as follows:

$$pr = \frac{1}{n} P e; \quad ppr_v = P v; \quad (3)$$

[1] In the case where dangling nodes with no outgoing edges exist, we can simply delete them, or add a sink node which links to itself and is pointed by each dangling node.