Using Proximity Search to Estimate Authority Flow

Vagelis Hristidis
School of Computing and Information Sciences
Florida International University
vagelis@cis.fiu.edu

Yannis Papakonstantinou
Computer Science and Engineering Dept.
University of California, San Diego
yannis@cs.ucsd.edu

Ramakrishna Varadarajan
Department of Computer Sciences
University of Wisconsin-Madison
ramkris@cs.wisc.edu

***Abstract*—Authority flow and proximity search have been used extensively in measuring the association between entities in data graphs, ranging from the Web to relational and XML databases. These two ranking factors have been used and studied separately in the past. In addition to their semantic differences, a key advantage of proximity search is the existence of efficient execution algorithms. In contrast, due to the complexity of calculating the authority flow, current systems only use precomputed authority flows in runtime. This limitation prohibits authority flow to be used more effectively as a ranking factor. In this paper we present a comparative analysis of the two ranking factors. We present an efficient approximation of authority flow based on proximity search. We analytically estimate the approximation error and how this affects the ranking of the results of a query.**

I. Introduction

Authority flow and proximity search have been used extensively in measuring the association between entities in data graphs, ranging from the Web to relational and XML databases. These two ranking factors have been used and studied separately in the past. We present a comparative analysis of the two ranking factors. In particular, we show how the cheaper proximity search can be used to approximate the expensive authority flow. Authority flow is being used by various applications [4], [2] to measure the importance of a query result. The validity of this factor has been proven by the success of using PageRank in Google.

Due to the high complexity of calculating the authority flows, current systems only use a small precomputed set of authority flow values. For example, Google precomputes a single PageRank value for each page denoting its global importance. ObjectRank [2], [12] go a step further to compute for each page (database object in ObjectRank) an authority flow value for each topic or keyword respectively.

However, as we explain below, it is desirable to be able to on-the-fly compute the authority flow between arbitrary nodes at query-time. For example, suppose we want to know how much authority is flowing between two pages A and B. This may be useful if we locate a base page A for a topic and want to find all pages B with significant relationship to A (that is, high flow from A to B). As another example consider the query: "Who are the authors of UCSD who receive high authority from the papers published by FIU?". To answer this query we need to on-the-fly calculate the authority flows from the papers of FIU to the authors of UCSD.

Incorporating structure-based relevance feedback on authority flow queries [24] or structure-based query personalization –user-defined authority flow bounds on different edge types– also require on-the-fly authority flow computation.

Partly supported by NSF grant IIS-0811922 and DHS grant 2009-ST-062-000016.

Another application that will benefit from on-the-fly authority flow computation is keyword proximity search on data graphs – on the Web [23] or on a database [14], [13], [1], [3], [19]. The result of such queries is typically a tree of interconnected nodes (pages or database objects) that collectively contain all the query keywords. Intuitively, results with tighter relationships between their nodes (pages or database objects) should be ranked higher. This "tightness" of the relationships can be estimated by the authority flow between the nodes. Due to the prohibiting cost of the on-the-fly computation of these authority flows, all the above works estimate the "tightness" by the size of the result-tree, which can be misleading in many applications.

In this work we show how we can efficiently calculate an approximation of the authority flow between two nodes (we generalize for multiple nodes as well) given the paths (trees for more than two nodes) with length up to M connecting the nodes. The latter problem has been extensively studied in prior work [14], [13], [1], [3], [9], [17], [19] where efficient algorithms are presented to find all paths of a *data graph* (a graph of pages and hyperlinks for the web, or of data objects and their relationships for databases as we explain in Section III-A) connecting two nodes.

The intuition behind this approximation is that there is a correlation between the authority flow between two nodes and the number of paths connecting them. In particular, when there are many paths connecting nodes u, v then there is a good chance that there is high flow of authority between u, v, and inversely. Hence, we can approximate the authority flow between u, v by counting (in a weighted manner) the result-trees that contain them.

Another contribution of this paper is that through the definition of the approximation method we show the interplay between the proximity and the authority flow ranking factors. In previous work, the authority flow [4], [2] and the proximity (structure of result tree) factor [9], [3], [1] were considered fundamentally different ways of ranking the results of a (keyword) query.

This paper has the following contributions:

- We discuss and formalize the relationship between the popular authority flow and proximity search factors.
- We present an approximation method to estimate the authority flow between nodes on-the-fly. This method uses a proximity search algorithm (as mentioned above many such algorithms exist in prior work [14], [13], [1], [3], [9]) as a subroutine.

- An analytical model is presented to estimate the quality of this approximation as a function of the maximum length M of the paths generated by the proximity algorithm. Also, we analytically calculate the error in ordering that this approximation imposes.

The paper is organized as follows. Section II presents the basics of authority flow and proximity search that we need in this work. Section III presents our framework and a formal definition of authority flow. Section IV presents the proposed approximation method and Section V shows how the approximation error is calculated. Finally, we present related work in Section VI and conclude in Section VII.

II. Background

This section presents the required background on authority flow and proximity search in graphs. Notice that these graphs are typically data graphs (formally defined in Section III-A), which are graphs whose nodes correspond to data entities (web pages or database objects) and the edges to relationships (hyperlinks, primary-to-foreign key relationships, parent-child relationships, and so on). Also, queries in previous work have been plain keyword queries.

A. Background on Authority Flow

We describe next the essentials of PageRank and authority-based search, and the random surfer intuition. Let (V, E) be a graph, with a set of nodes $V = \{v_1, \ldots, v_n\}$ and a set of edges E. A surfer starts from a random node (web page) v_i of V and at each step, he/she follows a hyperlink with probability d or gets bored and jumps to a random node with probability $1-d$. The PageRank value of v_i is the probability $r(v_i)$ that at a given point in time, the surfer is at v_i. If we denote by $\mathbf{r}$ the vector $[r(v_1), \ldots, r_(v_i), \ldots, r(v_n)]^T$ then we have

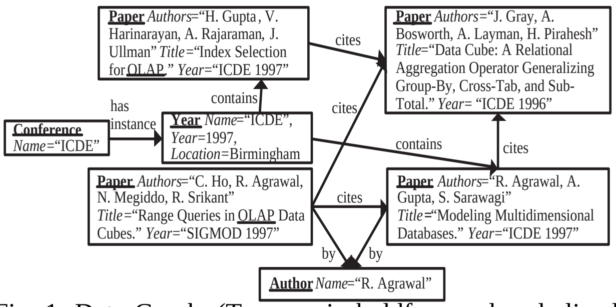
$$\mathbf{r} = d\mathbf{A}\mathbf{r} + \frac{(1-d)}{|V|}\mathbf{e} \qquad (1)$$

where $\mathbf{A}$ is a $n \times n$ matrix with $A_{ij} = \frac{1}{OutDeg(v_j)}$ if there is an edge $v_j \rightarrow v_i$ in E and 0 otherwise, where $OutDeg(v_j)$ is the outgoing degree of node v_j. Also, $\mathbf{e} = [1, \ldots, 1]^T$.

The above PageRank equation is typically precomputed before the queries arrive and provides a global, keyword-independent ranking of the pages. Instead of using the whole set of nodes V as the *base set*, i.e., the set of nodes where the surfer jumps when bored, one can use an arbitrary subset S of nodes, hence increasing the authority associated with the nodes of S and the ones most closely associated with them. In particular, we define a *base vector* $\mathbf{s} = [s_0, \ldots, s_i, \ldots, s_n]^T$ where s_i is 1 if $v_i \in S$ and 0 otherwise. The PageRank equation is then

$$\mathbf{r} = d\mathbf{A}\mathbf{r} + \frac{(1-d)}{|S|}\mathbf{s} \qquad (2)$$

The PageRank algorithm solves this fixpoint using a simple iterative method. The notion of the base set S was suggested in [4] as a way to do personalized rankings, by setting S to be the set of bookmarks of a user. ObjectRank [2] took it one step further and used the base set to estimate the relevance of a node to a keyword query – the base set consists of the nodes that contain the keyword.


Paper Authors="H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman" Title="Index Selection for OLAP." Year="ICDE 1997"
Paper Authors="J. Gray, A. Bosworth, A. Layman, H. Pirahesh" Title="Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total." Year= "ICDE 1996"
cites
contains
has instance
cites
Conference Name="ICDE"
Year Name="ICDE", Year=1997, Location=Birmingham
contains
cites
Paper Authors="C. Ho, R. Agrawal, N. Megiddo, R. Srikant" Title="Range Queries in OLAP Data Cubes." Year="SIGMOD 1997"
cites
Paper Authors="R. Agrawal, A. Gupta, S. Sarawagi" Title="Modeling Multidimensional Databases." Year="ICDE 1997"
by
by
Author Name="R. Agrawal"
Fig. 1: Data Graph. (Tags are in boldface and underlined.)

B. Background on Proximity Search

The proximity problem on a graph is similar to the Group Steiner Tree problem [21], which is known to be NP-hard. Prior work has followed various ways to overcome the NP-hard complexity of the problem. Goldman et al. [9] uses precomputation to minimize the runtime cost. BANKS [3] uses an algorithm to approximate the Group Steiner Tree problem following forward edges, whereas [17] also uses backward edges. References [14], [13], [1] assume the existence of a schema and exploit the schema properties to achieve efficient execution. Finally, Kimelfeld and Sagiv [19] present an algorithm with polynomial delay which repeatedly calls a traditional Steiner Tree approximation algorithm.

III. Definitions and Framework

This section presents the data model (Section III-A), and the authority flow problem (Section III-B) from the point of view of proximity search in order to understand the interplay between them.

A. Data Model

As mentioned above, the data can be the link-structure of the web, a structured database, or any other source that can be modeled as a graph. In particular, we view a database as a labeled graph, which is a model that easily captures the web, relational and XML databases. The *data graph* $D(V, E)$ is a labeled undirected graph where every node v (typically corresponding to a page for the web [4], to an element in XML [2], [14] and to a tuple in relational systems [2], [1], [13]) has a label $\lambda(v)$ and a *nodeid*. $\lambda(v)$ consists of the tag $\tau(v)$ and an optional value $\nu(v)$. For example, in Figure 1, the bottom node v has $\tau(v)$ ="Author", $\nu(v)$ ="Name="R. Agrawal"" and $\lambda(v)$ =<"Author", "Name", "R. Agrawal">.

From the data graph D, we can create the *authority transfer graph* $D^A(V, E^A)$ (Figure 2), if we know the amount of authority flow (*authority transfer rate*) that each direction of each edge can carry. In general, every edge $u - v$ in D^A has two authority transfer rates $a(u \rightarrow v)$ and $a(u \leftarrow v)$. The authority transfer data graph is a directed graph, where edges with zero authority transfer rate are ommited. For simplicity we will assume that there are no parallel edges.

B. Definition of Authority Flow

In this section we formally define authority flow from a perspective other than the random surfer (Section II-A). In

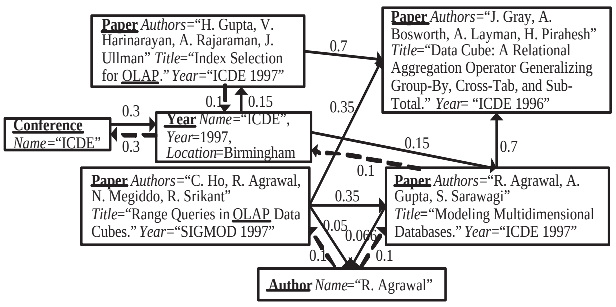

Paper Authors="H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman" Title="Index Selection for OLAP." Year="ICDE 1997"
Paper Authors="J. Gray, A. Bosworth, A. Layman, H. Pirahesh" Title="Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total." Year= "ICDE 1996"
Conference Name="ICDE"
Year Name="ICDE", Year=1997, Location=Birmingham
Paper Authors="C. Ho, R. Agrawal, N. Megiddo, R. Srikant" Title="Range Queries in OLAP Data Cubes." Year="SIGMOD 1997"
Paper Authors="R. Agrawal, A. Gupta, S. Sarawagi" Title="Modeling Multidimensional Databases." Year="ICDE 1997"
Author Name="R. Agrawal"
0.7
0.1
0.15
0.3
0.3
0.35
0.15
0.7
0.1
0.35
0.05
0.066
0.1
0.1

Fig. 2: Authority Transfer Data Graph.

particular, we define authority flow between nodes in terms of the paths connecting the nodes. We initially focus on pairwise authority flows, that is, we approximate the authority flow between a base set comprised of a single node u and a target node v. In Section IV-A, we extend for base sets with multiple nodes.

To formulate the problem, the basic quantity we use is the *probability of traversal* $p(u \to v)$ of an edge $u \to v$, which is defined as the probability that a random surfer currently at u will be at v in the next time unit following the edge $u \to v$. The probability of traversal $p([v_1, v_2, \cdots, v_n])$ of a path $l = (v_1 \to v_2 \to \cdots \to v_n)$ is the probability that a random surfer currently at v_1 will reach v_n in $n-1$ time units following one by one the edges of l. That is, $p([v_1, v_2, \cdots, v_n]) = p(v_1 \to v_2) \cdot p([v_2, \cdots, v_n])$ where $p(v_i \to v_j) = \frac{d}{OutDeg(v_j)}$ for regular edges. Note that we also discuss below, for virtual edges appearing due to the damping factor, it is $p(v_i \to v_j) = 1 - d$.

To go from path traversal probability to authority flow, we define the probability of travel in n steps as follows.

Definition 1: The n-step travel probability $P_n(u, v)$ from node u to node v is the probability that a random surfer currently at u will be at v after n time units. Furthermore, the up-to-n-steps travel probability $P_n^+(u, v)$ is defined as

$$P_n^+(u, v) = \frac{1}{n} \sum_{i=1 \cdots n} P_i(u, v) \tag{3}$$

Notice that we divide by n to ensure that the value is up to 1, since $P_n^+(u, v)$ is a probability. Note that if D_A was a DAG then this normalization would not be needed because the sum is always up to 1. However, due to the damping factor, which introduces virtual backedges, the graph is not a DAG. The intuition behind summing all $P_i(u, v)$'s is to get an average over all possible time instances from t to $t + n \cdot t_{unit}$, since the authority flow is the probability of being at a node at a random time instance.

The authority flow $Flow(u, v)$ from node u to node v is

$$Flow(u,v) = \begin{cases} lim_{n\to\infty} P_n^+(u,v) & \text{, if authority transfer graph } D^A \text{ has a cycle} \\ P_s^+(u,v) & \text{, where } s \text{ is length of longest path in } D^A \text{ if } D^A \text{ is a DAG} \end{cases} \tag{4}$$

As mentioned above, the graph is not a DAG, so we use the first branch of Equation 4.

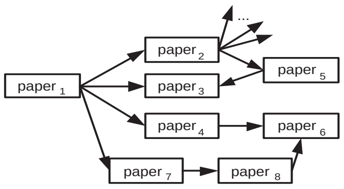


Fig. 3: Example showing the relation between number of result-trees and authority flow.

IV. Approximation method

We approximate the authority flow $Flow(u, v)$ between nodes u, v with the paths of length up to M (M is the maximum number of edges of the paths). These paths are calculated using proximity search algorithms as described in Section II-B. The contribution of each path $l : u \rightsquigarrow v$ is the probability of traversal $p(l)$ as defined above. For example in Figure 3, the path $l_1 : paper_1 \to paper_2 \to paper_5$ represents less authority transfer from $paper_1$ to $paper_5$ than $l_2 : paper_1 \to paper_4 \to paper_6$ from $paper_1$ to $paper_6$. Also, $l_2 : paper_1 \to paper_3 \leftarrow paper_5$ carries no authority from $paper_1$ to $paper_5$, because authority only flows along the direction of the edges (authority only flows to cited papers and not to citing as it is explained in the ObjectRank work [2]). Note that in case of other edge types, authority could potentially flow in both directions as shown in Figure 2.

If $B_M(u, v)$ is the set of paths with size up to M in D_A going from u to v, the authority flow calculated by the approximation method is $\sum_{l \in B_M(u,v)} p(l)$. To be consistent with the normalization method of Section III-B and ensure that the flow value is up to 1, we divide this quantity by M. Hence

$$Flow_M(u, v) = \frac{1}{M} \sum_{l \in B_M(u,v)} p(l) = P_M^+(u, v) \tag{5}$$

This approximation suffers from the *size error*, which is due to the fact that all works [9], [13], [3] returning result-trees limit their maximum size to M, but authority may flow along longer paths. In the example above, if $M = 2$, the path $l_4 : paper_1 \to paper_7 \to paper_8 \to paper_6$ is ignored since it has size greater than 2. The relative size error E_S is defined by the following formula

$$E_S = \frac{Flow(u, v) - Flow_M(u, v)}{Flow(u, v)} \tag{6}$$

Notice that it is always $E_s \geq 0$ because $Flow(u, v)$ includes $Flow_M(u, v)$. Ignoring the constant factors for simplicity, we get

$$E_S = \frac{lim_{n\to\infty} P_{M+1,n}^+(u, v)}{Flow(u, v)} \tag{7}$$

where $P_{m,n}^+(u, v) = \frac{1}{n-m} \sum_{i=m \cdots n} P_i(u, v)$. Section V estimates the value of E_S.

Finally, we must address the issue of whether we should materialize the virtual edges implied by the damping factor d. In particular, a virtual edge $u \to v$ should be added going from each node u to every node v of the base set. To be accurate, we should do this before applying the proximity search algorithms. However, to reduce the complexity of the proximity search algorithms, we can omit this step, since d

Name	#Nodes	#Edges	Size(Mb)
DS7	699,199	3,533,756	2,189
DBLP	876,110	4,166,626	3,950

TABLE I: Datasets

is typically close to 1 ($d = 0.85$ in PageRank [4]), and the virtual edges have authority transfer rate of $1 - d$.

A. Extend to a Base Set with Multiple Nodes

So far we have explained how to approximate the authority flow from a single-node base set to a node v. In this section we generalize to base sets with multiple nodes. To do so, we use the linearity theorem from Jeh and Widom [16], which states that if a base set B consists of two sets B_1 and B_2, such that $B = B_1 \cup B_2$, then the PageRank score $r^B(v)$ of a node v with respect to B is $r^B(u) = r^{B_1}(v) + r^{B_2}(v)$, where by $r^S(v)$ we denote the authority flow of node v with respect to a base set S.

Suppose that the base set B consists of nodes $u_1, \cdots, u_s$. Then from the linearity theorem, we have

$$Flow(B, v) = \sum_{i=1}^{s} Flow(u_i, v) \tag{8}$$

Hence, to approximate the authority flow of a node v with respect to a base set B, we need to approximate the flow to v from each node $u \in B$. Fortunately, proximity search works [9], [14], [13], [1], [3] adopt under the same assumption. That is, they find the shortest paths between sets of nodes and not between a pair of nodes.

V. Evaluation of Approximation Method

This section analytically estimates the error imposed by applying the approximation method of Section IV. In particular, we first (Section V-A) estimate the relative error E_S of the authority flow value calculated using the approximation method, and then (Section V-B) we present how this error affects the ordering of the results of a query that uses authority flow as the ranking factor. Finally, we present an experimental analysis (Section V-C) of performance and quality of results.

Datasets: We use two real datasets (Table I). DBLP[1] is a bibliographical dataset with papers, conference, authors and year of publication. DS7 is a biological dataset with PubMed[2] publications, Entrez genes, nucleotides, proteins and OMIM objects created following an experimental protocol that starts from annotated gene records in public Web accessible sources, and follows hyperlinks, to reach publications in PubMed.

A. Estimation of E_S

As explained in Section II-A, the authority flow is calculated using Equation 1. This equation can be calculated (as in [4], [2]) using the power method, which is an iterative method which calculates the primary eigenvalue of a matrix $\mathbf{A}$. To estimate E_S we make the following observation: Taking into account the connecting paths of size up to M is equivalent to executing the power method for M iterations. Hence, E_S is equal to the error of stopping the power method after M iterations. Note that in order to maintain $E_s \geq 0$ for all M values, it is important to initialize the page rank vector $\mathbf{r}$ same as the base vector $\mathbf{s}$.

The convergence of the power method is linear, reducing the error of the eigenvector (i.e., the Euclidean distance) by about $|\lambda_2|/|\lambda_1|$ in each iteration [15], where λ_1,λ_2 are the first and second eigenvalues. Hence, the relative error of the eigenvector after M iterations is $E_S = O((|\lambda_2|/|\lambda_1|)^M)$.

Furthermore, Haveliwala and Kamvar [11] have proven (Theorem 1) that $\lambda_2 = d$, where d is the damping factor, for the web graph.

Theorem 1: For any matrix $A = [cP + (1-d)E]^T$, where P is an $n \times n$ row-stochastic matrix, E is a nonnegative $n \times n$ rank-one row-stochastic matrix, and $0 \leq d \leq 1$, the second eigenvalue of A has modulus $|\lambda_2| \leq d$. Furthermore, if P has at least two irreducible closed subsets, the second eigenvalue $\lambda_2 = d$.

Hence, since λ_1 has been chosen to be 1 by convention [11], the error of the eigenvector in each iteration is reduced by d. Notice that the conditions of Theorem 1 hold for the web graph and most database graphs in practice. Hence, we reach

$$E_S = O(d^M). \tag{9}$$

B. Error in Ordering

The absolute values of the authority flows are typically used [4], [2] to create an ordering of the nodes with respect to their authority flow from a base set B, which for simplicity we assume that it contains a single node u. Hence, it is important to analyze how applying the approximation method affects this ordering. In particular, this section estimates the probability of the error in ordering of two nodes v, v' of D^A with respect to their authority flow from node u.

Probability Density Function of E_S. To do so, we need to know the distribution function of the error E_S, which has mean value d^M as we explained in Section V-A. There is no analytical work on calculating the probability density function (PDF) of E_S, hence we adopted an experimental approach. In particular, we executed the power method for the DBLP dataset (see ObjectRank [2] for details on how to create the authority flow graph for the DBLP dataset) for multiple base sets (each base set corresponds to a keyword as in [2]) and multiple numbers of iterations.

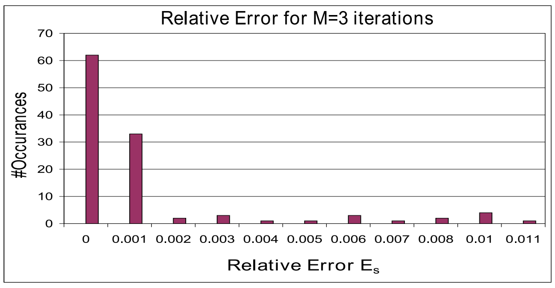
For each number of iterations we recorded the error E_S for multiple nodes (the top-20 results for the base set) and multiple base sets. By analyzing the results, we observed that the PDF of E_S is similar to the exponential distribution. For example, Figure 4a shows the distribution of E_S values in the DBLP dataset for the top-20 nodes for 12 different base-sets, for $M = 3$ iterations. Figure 4b shows similar distribution in the DS7 dataset for $M = 3$ iterations. Note that in order to guarantee positive E_S values for each test node for every path length M, we initialize the ObjectRank vector with the base vector.

Hence, since the mean value of the exponential distribution $f(x) = s \cdot e^{-s \cdot x}$ is $1/s$, we approximate the PDF of E_S by

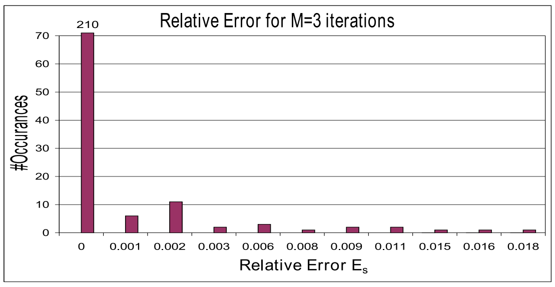
$$f(x) = d^{-M} e^{-d^{-M} x} \tag{10}$$

[1]http://www.informatik.uni-trier.de/l̃ey/db/

[2]http://www.ncbi.nlm.nih.gov/pubmed/



(a) DBLP Dataset



(b) DS7 Dataset

Fig. 4: Distribution of E_S

Ordering error. Given the PDF of E_S, we can calculate the probability of creating the correct ordering for v, v' as follows. For simplicity in notation, we set $r(v) = Flow(u, v)$, $r(v') = Flow(u, v')$, $a(v) = Flow_M(u, v)$, $a(v') = Flow_M(u, v')$. That is, $r(.)$ are the actual flows and $a(.)$ are the approximations after M iterations. We will calculate the probability $P\left(a(v) > a(v')\right)$, that is, the probability that the approximate method ranks v higher than v'. Then we show that this probability goes to 1 (0) if $r(v) > r(v')$ ($r(v) < r(v')$) as M increases.

Theorem 2: $P\left(a(v) > a(v')\right) = 1 - \frac{e^{-(1-\frac{r(v')}{r(v)})d^{-M}}}{1+\frac{r(v')}{r(v)}d^M}$

Proof: Given Equation 6 and the simplified notation above, it is

$$a(v) = r(v) - E_S(v)r(v)$$
$$a(v') = r(v') - E_S(v')r(v')$$

Hence,

$$P\left(a(v) > a(v')\right) = P\left(E_S(v) < \frac{r(v) - r(v') + r(v')E_S(v')}{r(v)}\right)$$

$$= \int_0^\infty \left(f(y) \int_0^{\frac{r(v)-r(v')+r(v')y}{r(v)}} f(z)dz \right) dy$$

Using Equation 10 and the fact that $\int_0^\infty s \cdot e^{-sx}dx = 1$ we get

$$P\left(a(v) > a(v')\right) = 1 - \frac{e^{-(1-\frac{r(v')}{r(v)})d^{-M}}}{1 + \frac{r(v')}{r(v)}d^M}$$

■

$P\big(a(v) > a(v')\big)$ is a probability and hence has to take values from 0 to 1. However it is possible that it becomes negative for some values $r(v), r(v')$ satisfying $r(v) < r(v')$. If such a case arises, we set the probability to 0. Also, notice that

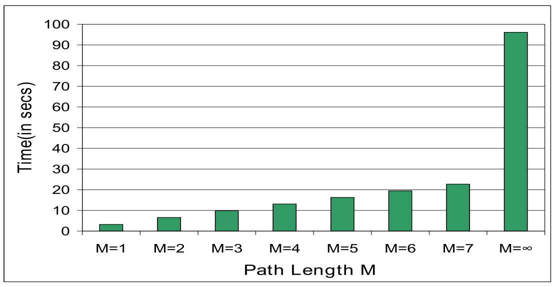
$$lim_{M\to\infty} P\big(a(v) > a(v') \mid r(v) > r(v')\big) = 1 \qquad (11)$$
$$lim_{M\to\infty} P\big(a(v) > a(v') \mid r(v) < r(v')\big) = 0 \qquad (12)$$

since $0 < d < 1$.

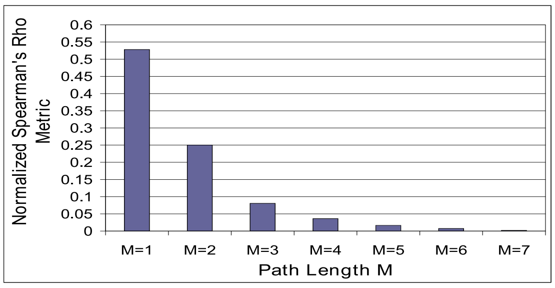
C. Performance vs. Results' Quality

The experiments were evaluated on a Windows Server 2003 machine with Intel Xeon 2.40 GHz processor and 2GB of RAM. All algorithms were implemented in Java (JDK version 1.6.0_10). Oracle DBMS (version 10g Enterprise Edition Release 10.2.0.1.0) was used to store the database and JDBC was used to connect to the database system. We use the global ObjectRank scores to initialize the nodes in the data graph as done in [12]. Figures 5a and 6a show the performance for increasing values of path length, M for DBLP and DS7 datasets respectively. To provide a baseline, we compare our execution time and results' quality with the exact solution - the original ObjectRank algorithm executed over the data graph. This is equivalent to setting M to ∞.

Figures 5b and 6b show the quality of the top-1000 results for increasing values of M with respect to the exact top-1000 results using the top-k Spearman's rho metric [6]. Notice that as the path length M increases, the performance degrades and the quality improves (lower value of Spearman's rho metric) since the number of paths considered for authority flow increases. There is clearly a trade-off; for lower M we have lower delay but also lower quality. Notice that in both datasets, for M=3, we achieve a good tradeoff of quality and performance (higher quality for a relatively shorter delay time). We note that, on average, for M=29.1 (in DBLP) and M=8.5 (in DS7) we get exact results, but this doesn't mean that there is no path longer than that, since the graph will contain cycles (especially because of bidirectional edges that transfer authority flow is both directions in ObjectRank [12]). The optimal path length M is computed by progressively increasing M until the authority scores converge. The higher exact M in DBLP, which also leads to higher execution times, is due to its higher connectivity.

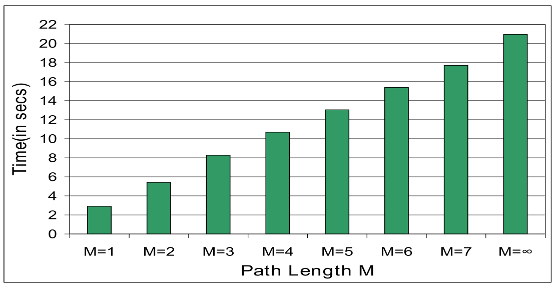


(a) Performance with varying Path Length M

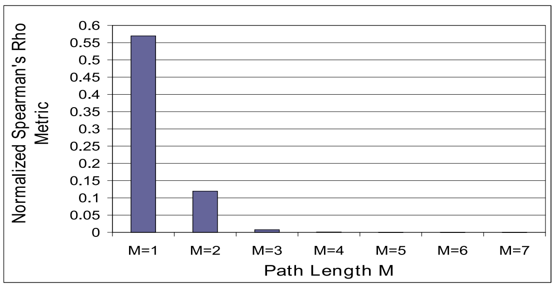


(b) Results' Quality with varying Path Length M

Fig. 5: Performance and Quality Experiments on DBLP



(a) Performance with varying Path Length M



(b) Results' Quality with varying Path Length M

Fig. 6: Performance and Quality Experiments on DS7

VI. Related Work

Link-based analysis: The notion of importance has been defined in the context of the Web using PageRank [4], where a global score is assigned to each Web page. Recently, the idea of PageRank has been applied to structured databases [2]. Faloutsos et al. [7] view the authority flow problem as the maximization of the electric current between the nodes, where each edge of the data graph is represented by an electric resistor.

Performance: A set of works [10], [5], [16], [18] have tackled the problem of improving the performance of the original PageRank algorithm. [10], [5] present algorithms to improve the calculation of a global PageRank. Jeh and Widom [16] present a method to efficiently calculate the PageRank values for multiple base sets, by precomputing a set of *partial vectors* which are used in runtime to calculate the PageRanks. Tong et al. [22] exploit structural properties of real graphs to efficiently perform random walk over large graphs and employ precomputation to improve performance. A key motivation of our work is to on-the-fly compute the authority flow for personalized search (as in [24]), where different user-specific authority rates are assigned to the edges of the graph for each query. This makes pre-computation infeasible. Fogaras et al. [8] present an approximate and scalable method that achieves full personalization using fingerprint paths (random walks) where the length of the walk is of geometric distribution of damping factor, d. In contrast, in authority flow, the length of the random walk is of uniform distribution.

Wu and Raschid [25] present approximation methods for ranking a subgraph assuming that the rest of the data graph is unchanged. Koren et al. [20] present a new measure of proximity in graphs, called cycle free effective conductance (CFEC), which refines the ideas of Faloutsos et al. [7]. CFEC measure is the probability that a random surfer will go from a node u to a node y following a simple path (no cycles) between them. Their approximation algorithm computes CFEC using the k-shortest simple paths. In contrast, we adopt the standard authority flow semantics, used by PageRank [4], ObjectRank [2] and other works, which consider all paths. We do not claim that the one semantics is better than the other. A key contribution of our work is that we provide a theoretical approximation bound analysis, whereas [20] does not.

VII. Conclusions

We presented a method to efficiently approximate the authority flow between a base set B and a node v. This method assumes no prior knowledge of B or v and hence it is suitable for on-the-fly authority flow computation. Our work allows authority flow to be used in new real-time applications, like measuring the quality of a result-tree in a proximity search system, or answering complex on-the-fly queries. We also analytically prove the error of our approximation method and the error it imposes in the ordering of query results.

References

[1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System For Keyword-Based Search Over Relational Databases. *ICDE*, 2002.
[2] A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-Based Keyword Search in Databases. *VLDB*, 2004.
[3] G. Bhalotia, C. Nakhey, A. Hulgeri, S. Chakrabarti, and S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. *ICDE*, 2002.
[4] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *WWW*, 1998.
[5] Y. Chen, Q. Gan, and T. Suel. I/O-efficient techniques for computing PageRank. *CIKM*, 2002.
[6] R. Fagin, R. Kumar, and D. Shivakumar. Comparing top k lists. *SODA*, 2003.
[7] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. *KDD*, 2004.
[8] D. Fogaras, B. Racz, K. Csalogany, and T. Sarlos. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics Vol. 2, No. 3: 333-358*, 2005.
[9] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity Search in Databases. *VLDB*, 1998.
[10] T. Haveliwala. Efficient computation of PageRank. *Technical report, Stanford University (http://www.stanford.edu/ taherh/papers/efficient-pr.pdf)*, 1999.
[11] T. Haveliwala and S. Kamvar. The Second Eigenvalue of the Google Matrix. *Stanford University Technical Report*, 2003.
[12] V. Hristidis, H. Hwang, and Y. Papakonstantinou. Authority-based keyword search in databases. *ACM TODS*, 2008.
[13] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. *VLDB*, 2002.
[14] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. *ICDE*, 2003.
[15] J. H.Wilkinson. The Algebraic Eigenvalue Problem. *Oxford University Press*, 1965.
[16] G. Jeh and J. Widom. Scaling Personalized Web Search. *WWW*, 2003.
[17] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. *VLDB*, 2005.
[18] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Extrapolation Methods for Accelerating PageRank Computations. *WWW*, 2003.
[19] B. Kimelfeld and Y. Sagiv. Finding and Approximating Top-k Answers in Keyword Proximity Search. *PODS*, 2006.
[20] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. *KDD*, 2006.
[21] G. Reich and P. Widmayer. Beyond Steiner's Problem: A VLSI Oriented Generalization. *WG*, 1989.
[22] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. *ICDM*, 2006.
[23] R. Varadarajan, V. Hristidis, and T. Li. Beyond Single-Page Web Search Results. *IEEE TKDE*, 2008.
[24] R. Varadarajan, V. Hristidis, and L. Raschid. Explaining and reformulating authority flow queries. *ICDE*, 2008.
[25] Y. Wu and L. Raschid. Approxrank: Estimating rank for a subgraph. *ICDE*, 2009.