# Efficient Ranking on Entity Graphs with Personalized Relationships

Vagelis Hristidis, Yao Wu, and Louiqa Raschid

**Abstract**—Authority flow techniques like PageRank and ObjectRank can provide personalized ranking of typed entity-relationship graphs. There are two main ways to personalize authority flow ranking: Node-based personalization, where authority originates from a set of user-specific nodes; Edge-based personalization, where the importance of different edge types is user-specific. We propose the first approach to achieve efficient edge-based personalization using a combination of precomputation and runtime algorithms.

In particular, we apply our method to ObjectRank, where a personalized weight assignment vector (WAV) assigns different weights to each edge type or relationship type. Our approach includes a repository of rankings for various WAVs. We consider the following two classes of approximation: (a) SchemaApprox is formulated as a distance minimization problem at the schema level; (b) DataApprox is a distance minimization problem at the data graph level. SchemaApprox is not robust since it does not distinguish between important and trivial edge types based on the edge distribution in the data graph. In contrast, DataApprox has a provable error bound. Both SchemaApprox and DataApprox are expensive so we develop efficient heuristic implementations, ScaleRank and PickOne respectively. Extensive experiments on the DBLP data graph show that ScaleRank provides a fast and accurate personalized authority flow ranking.

**Index Terms**—object search, personalization, PageRank, approximation algorithms

## 1 Introduction

The success of PageRank [1] in ranking Web pages resulted in many flavors of authority flow-based ranking techniques for data in entity-relationship graphs [2], [3], [4], [5]. A key feature of ranking in entity-relationship graphs is that they provide intuitive personalization opportunities by adjusting the authority flow parameters associated with each edge type or relationship type. Authority originates from a query- or user-specific set of objects, and spreads via edges whose authority flow weights is determined by their edge (relationship) type. For instance, a paper-to-paper citation edge may have a higher authority flow weight than the paper-to-author edge in a bibliographic data graph.

Two fundamental approaches have been proposed to personalize authority flow ranking: (a) Node-based personalization: a personalized base set, i.e., the authority originates from a query- or user-specific set of objects; (b) Edge-based personalization: personalized weight assignment vector (WAV) which assigns a weight to each edge (relationship) type. We use ObjectRank [4], [6] as an exemplar of this latter class.

Both approaches are computationally expensive and do not support interactive response times for on-the-fly and scalable personalization. Authority flow techniques typically require dozens of iteration across the data graph. Previous work [2], [7], [8], [9], [10], [11], [12] has addressed the performance of the node-based personalization approach. There is no work to facilitate efficient computation of edge-based personalization. Our specific challenge is on-the-fly execution of authority flow fixpoint computation for a user-specific or query-specific weight assignment vector (WAV). While we use ObjectRank as an exemplar, our approach is applicable to other authority flow ranking techniques like [13], [14].

Given a keyword query, ObjectRank [4], [6] first computes the *base set* of nodes in the data graph that contain the query keywords. Then, authority flows from the base set to the whole data graph, until the authority scores on the nodes converge (Equation 3). The nodes with the top score are returned. The authority transfer edges of the data graph are represented by a transition matrix. More details are presented in Section 2.1.

Figure 1 [6] shows the *authority transfer schema graph* for DBLP, a bibliographic database for computer science publications [15]. There are 8 edge types and each edge type is associated with a numeric value representing the personalized authority weight in WAV $\Theta$; e.g., a weight of 0.2 for the edge type from Paper to Author. It has been argued that iteratively adjusting the WAVs on the DBLP graph is an effective query refinement mechanism [16].

As another example, consider the biological Web at the NIH [17]; we show a subset of its authority transfer schema graph in Figure 2. As explained in Varadarajan et al. [5], a biologist user may assign more importance (higher edge weight) to a protein-to-protein edge type, whereas another user may assign a higher importance to a paper-to-paper citation edge type.

Since users submit their queries and personalized WAV $\Theta$ on-the-fly, a key challenge is to compute personalized rankings online and to quickly provide answers to the user. Clearly, computing each personalized ranking at query


- *V. Hristidis is with the University of California, Riverside, CA 92521. E-mail: vagelis@cs.ucr.edu*
- *Y. Wu is with Microsoft, Redmond, WA, 98052.*
- *L. Raschid is with the University of Maryland, College Park, MD, 20742.*

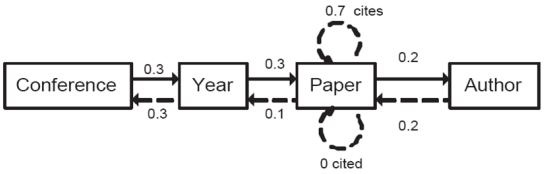


Fig. 1. The DBLP authority transfer schema graph in ObjectRank ([6]).

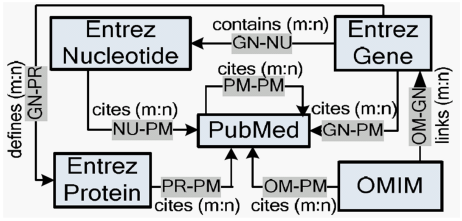


Fig. 2. Biomedical authority transfer schema graph.

time will not support online ranking. The other extreme of computing all possible personalized rankings a priori and storing them is infeasible. Our solution is a pragmatic hybrid solution. We will maintain a repository of pre-computed rankings. At query time, an *approximate personalized ranking* may be computed using some chosen set of pre-computed candidate rankings from the repository.

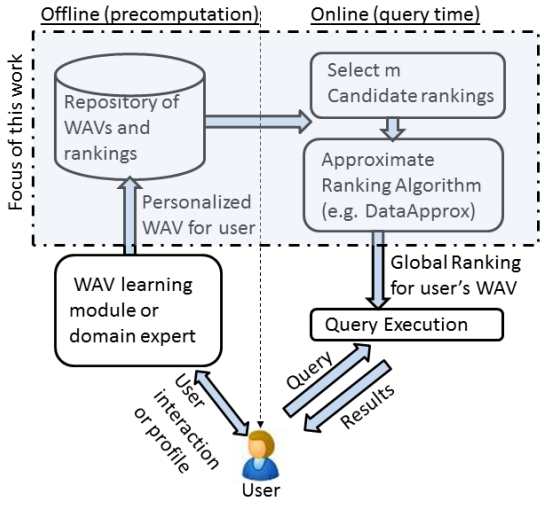


Fig. 3. Incorporate personalized authority flow in user queries.

Figure 3 illustrates how a user interacts with the ranking system at precomputation and query times. During pre-computation, the user-specific personalized WAV can be computed in two ways: (a) It can be set for each user by domain experts according to the user's profile [5], [6]. (b) WAV $\Theta$ is learned automatically by interacting with the user and exploiting user relevance feedback as explained in [16]. At query time, the system selects $m$ candidate rankings from the repository, which are used to estimate the ranking for the requested $\Theta$. This personalized ranking is input to the Query Execution module, which combines this ranking with other query-specific or query-independent factors and returns the query results.

Note that the authority flow ranking of the entities is just one of the factors used for ranking. In the same spirit, PageRank [1] was presented as a query-independent ranking of the pages, which is then combined with query-specific factors. We assume the same setting here.

**Example:** For the schema graph of Figure 1, a user who believes that citations are the only important authority vote in a bibliographic database, may have WAV $\Theta = (0.3, 0.3, 0.3, 0.1, 0.7, 0, 0.2, 0.2)$, which corresponds to the edge weights shown on the figure. Note that the paper citation edge has weight 0.7. Another user who believes that a good author almost always authors good papers, could have $\Theta = (0.3, 0.3, 0.3, 0.1, 0.3, 0, 0.4, 0.8)$, where the last coordinate corresponds to the Author-to-Paper edge. These WAVs are computed either automatically using [16] or by a domain expert.

The goal of this paper is to facilitate efficient ObjectRank execution for varying WAVs. Suppose that the repository of rankings contains rankings for WAVs: $\Theta_1 = (0.7, 0.3, 0.3, 0.1, 0.2, 0, 0.2, 0.2)$, $\Theta_2 = (0.3, 0.3, 0.3, 0.1, 0.7, 0.2, 0.2, 0.2)$, $\Theta_3 = (0.1, 0.3, 0.1, 0.1, 0.1, 0, 0.2, 0.3)$. Our algorithms select a subset with $m$ of these rankings and appropriately combine them to efficiently compute the ranking for the user WAV $\Theta^q$, without having to execute the expensive iterative ObjectRank algorithm. □

We consider the following challenges for our repository based approximation approach: (1) The best $m$ candidate rankings must be selected at query time. (2) The $m$ rankings must be appropriately combined in order to estimate the ranking for the given personalized WAV. (3) The approximate personalized ranking should be close to the ideal ranking so as to guarantee high quality. (4) The approximate ranking should be computed efficiently.

This paper makes the following contributions:

- Consider a user WAV $\Theta^q$, its transition matrix $A_q$ for ObjectRank computation, and the ideal ranking $R_q$. We consider the following two classes of approximation algorithms: (a) SchemaApprox is defined at the schema level and employs a least squares formulation to choose the $m$-best candidates so that the combined Euclidean distance of these $m$ candidates $\Theta^{comb}$, to $\Theta^q$, is minimized. (b) DataApprox is defined at the data level. DataApprox computes a weighted combination of $m$ candidate rankings; to do so it solves an optimization problem so that the maximum norm ($\delta$), over all elements of the aggregate transition matrix of DataApprox and $A_q$, is minimized.
- We prove that DataApprox has provable error bounds. We introduce the concept of an aggregate surfer and prove an authority flow linearity theorem for authority flow rankings.
- DataApprox and SchemaApprox are too expensive to facilitate interactive query response. ScaleRank and

PickOne are two heuristics. ScaleRank chooses $m$ candidates from the repository based on their Euclidean distance to $\Theta^q$ of the user; this is inspired by SchemaApprox and hence ScaleRank can also be viewed as a hybrid algorithm combining SchemaApprox and DataApprox. ScaleRank applies linear programming to solve the DataApprox optimization problem, and explores techniques to reduce the search so that it can be applied to large data graphs. PickOne is a greedy solution to SchemaApprox.

- We conduct extensive experiments to evaluate the execution time and the quality for ScaleRank and PickOne, i.e., how close the approximate ranking is to the ideal ranking $R_q$. The experiments are conducted on the complete DBLP dataset. We use the well known Spearman's Footrule distance [18] as a proxy for the quality of the solution. ScaleRank outperforms PickOne in quality while achieving fast response times.

The paper is organized as follows. Section 2 reviews background and related work. Section 3 defines approximations SchemaApprox and DataApprox and proves the linearity theorem for authority flow weights. Section 4 anayzes the errors of the two approaches. Section 5 describes the system architecture and Section 6 presents the ScaleRank algorithm and its analysis. Experimental results are described in Section 7, and we conclude in Section 8.

## 2 Background and Related Work

### 2.1 Authority Flow Ranking: The ObjectRank Algorithm

Given a keyword query, ObjectRank [4], [6] first computes the *base set* of nodes in the data graph that contain the query keywords. Then, authority flows from the base set to the whole data graph, until the authority scores on the nodes converge (Equation 3). The nodes with the top score are returned. As mentioned in Section 1, we focus on the WAV personalization and hence we assume the base set is the whole graph (referred as Global ObjectRank in [6]). That is, when we say ObjectRank, we usually mean Global ObjectRank.

ObjectRank personalizes ranking in Entity-Relationship graphs; it models nodes as entity types and groups edges by their edge type or semantic type. Authority flow is personalized by a *weight assignment vector* (WAV) $\Theta$ for the semantic edge types. In Figure 1, there are 8 edge types.

The transition matrix $A_{OR}$ of ObjectRank depends on the authority transfer $\Theta$ specified on the *schema graph*; however, $A_{OR}$ is defined at the level of the data graph. To demonstrate the relationship of the ObjectRank transition matrix and the PageRank transition matrix, without loss of generality we assume that the objects of the same type are grouped together. Consider an authority transfer schema graph with $t$ entity types. The *weight assignment vector* (WAV) $\Theta = \{\alpha_{1,1}, \alpha_{1,2}, ..., \alpha_{1,t}, \alpha_{2,1}, \alpha_{2,2}, ..., \alpha_{2,t}, ..., \alpha_{t,1}, \alpha_{t,2}, ..., \alpha_{t,t}\}$ represents the authority transfer weights. $A_{OR}$ contains $t \times t$ submatrices. Each submatrix entry of the transition matrix $A_{OR}$ is multiplied by the authority transfer weight for the corresponding semantic edge type. $A_{OR}$ can be expressed as follows:

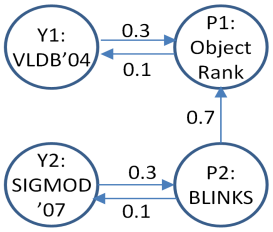


Fig. 4. Authority Flow Instance Graph

$$A_{OR} = \begin{pmatrix} \alpha_{1,1}A_{1,1} & \alpha_{1,2}A_{1,2} & \cdots & \alpha_{1,t}A_{1,t} \\ \alpha_{2,1}A_{2,1} & \alpha_{2,2}A_{2,2} & \cdots & \alpha_{2,t}A_{2,t} \\ \vdots & \vdots & & \vdots \\ \alpha_{t,1}A_{t,1} & \alpha_{t,2}A_{t,2} & \cdots & \alpha_{t,t}A_{t,t} \end{pmatrix} \quad (1)$$

The submatrix $A_{p,q}$ contains authority transfer probabilities from objects of type $p$ to objects of type $q$. Let $e^T(v_i, v_j)$ be the semantic type of edge $(v_i, v_j)$ in the data graph. Let $\alpha(e^T(v_i, v_j))$ denote the weight assignment for $e^T(v_i, v_j)$. $OutDeg(v_i, e^T(v_i, v_j))$ is the number of outgoing edges from page $v_i$, of type $e^T(v_i, v_j)$. The submatrix $A_{p,q}$ is defined as follows:

$$A_{p,q}[i,j] = \begin{cases} \frac{1}{OutDeg(v_i, e^T(v_i,v_j))} & \text{if } (v_i,v_j) \text{ exists} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

**Example:** Figure 4 shows an instance of the authority transfer schema graph in Figure 1, where we show that the BLINKS SIGMOD 2007 paper cites the ObjectRank VLDB 2004 paper. Note that the weights would change if the rest of the graph is added, e.g., the 0.7 flow from BLINKS to ObjectRank would be divided by the number of citations of BLINKS. If we order the 4 nodes as Y1,Y2,P1,P2, then:

$$A_{OR} = \begin{pmatrix} 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \\ 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0.7 & 0 \end{pmatrix}$$

□

Let $A_{OR}^T$ denote the transpose of $A_{OR}$. The ObjectRank vector $R_{OR}$ is recursively defined as follows in Equation 3:

$$R_{OR} = \epsilon A_{OR}^T \cdot R_{OR} + (1-\epsilon)P \quad (3)$$

where $P$ is a vector that specifies the nodes of the graph that are the authority sources. In ObjectRank, $P$ specifies the nodes that contain the query keywords (all nodes for Global ObjectRank).

In addition to ObjectRank, which is an extension of the PageRank authority flow ranking method, HITS [19] or its extensions [20] may also be used for authority flow-based personalization.

### 2.2 Approximation methods

The node-based personalization problem as well as a corresponding linearity theorem has been studied in [2], [7], [8], [9], [10]. The approach in [8] is based on a linearity theorem that is used to combine multiple personalized PageRank vectors. Let vector $v_P$ be the personalization vector that replaces $P$; then the personalized PageRank equation is as follows:

$$R_P = \epsilon A^T \cdot R_P + (1-\epsilon)v_P \quad (4)$$

$R_P$ is the personalized PageRank vector (PPV) for $v_P$.

*Theorem 1:* **(The Linearity Theorem) [8], [10]** For any personalization vectors $v_{P1}$ and $v_{P2}$, if $R_{P1}$ and $R_{P2}$ are the two corresponding PPVs, then for any constants $\beta_1$, $\beta_2 \geq 0$ such that $\beta_1 + \beta_2 = 1$,

$$\beta_1 R_{P1} + \beta_2 R_{P2} = \epsilon A^T \cdot (\beta_1 R_{P1} + \beta_2 R_{P2}) + (1-\epsilon)(\beta_1 v_{P1} + \beta_2 v_{P2})$$

Based on Theorem 1, [10] proposed a technique that encodes personalized ranking vectors as partial vectors. They also presented efficient dynamic programming algorithms. In [7], an algorithm that simulates random walks is used to pre-compute an *index database* of personalized PageRank vectors (fingerprints). [2], [9] consider using a personalized base set on entity-relationship graphs. They do not consider personalized weight assignments a la ObjectRank. HubRank [2] employs query log statistics to select a small fraction of nodes, and computes and stores fingerprints for these nodes. A small subgraph is identified at query time to form approximate personalized PageRank vectors. BinRank [9] stores subgraphs such that any keyword query can be answered by performing ObjectRank on one subgraph. lgOR [5] is an extension of ObjectRank that is tailored to a layered graph; this is a restricted graph where there can only be edges between nodes in adjacent layers. A graph-sampling technique is applied to approximate the lgOR scores. The problem is reduced to estimating a subgraph for the result graph, such that with high confidence the relative error of computing lgOR on the subgraph is small. The sampling technique of lgOR was only applied to a layered graph and cannot be applied directly to approximate ObjectRank on a general graph. PopRank [13] applied the idea of authority flow rankings to Web objects. A simple simulated annealing algorithm is presented to learn a good query-independent weight assignment. Given a ranking, [16] learns the WAV for every user and query using relevance feedback. A preliminary version of this work, with no error bounds, ScaleRank optimizations or detailed experimentation, was published as a workshop paper [21].

### 2.3 Non-Authority Flow Personalization

We summarize alternative approaches to personalization. Trust and similarity have been used to compute personalized rankings [22]. These concepts are captured from explicit user input and implicit user behavioral patterns to describe the user's taste and preference. [23] consider the Web community of a specific user in personalization. Past interactions of the user with the search engine are used to improve future search results. For each Web community, its neighborhoods including the documents linked to, or from, documents in the community are determined. The query answers are ordered to reflect the number of times these community neighborhoods have been visited. [24], [25], [26], [27] proposed adapting search results based on users' history, navigational history, browsing history, or query history. There are multiple data mining techniques that can be applied to extract usage patterns from Web logs [24], [25].

Albeit the various personalization works, there is no technique to approximate edge-based personalization for authority-flow ranking.

## 3 Approximation Using a Repository

### 3.1 The Problem Definition

Consider user WAV $\Theta^q$, the ObjectRank transition matrix $A_q$, and ranking $R_q$. Our problem can be described informally as follows: Given a set of $M$ candidate rankings in a repository, choose the $m$ best candidates using some metric and appropriately combine their rankings, so that it provides an approximate ranking of the highest quality, compared to $R_q$. The objective for the concrete problem needs to satisfy the following requirements: 1) We should choose an appropriate distance metric to choose the candidate rankings. 2) The distance should be easy to compute. 3) The distance metric should be correlated to the approximation quality (e.g., Spearman's Footrule). We expect that a stronger correlation will improve the choice of candidate rankings and the approximation quality. We formalize the problem as follows:

**Problem statement**:
Let $\mathcal{S} = \{(\Theta_1, R_1), (\Theta_2, R_2), \cdots, (\Theta_M, R_M)\}$ be the ranking repository of $M$ precomputed ranking vectors and their corresponding weight assignment vectors. Let $\Theta^q$ be the user weight assignment vector. The goal is to approximate the authority flow ranking for $\Theta^q$ efficiently, with the maximal quality, by utilizing the precomputed ranking vectors in the repository.

### 3.2 Distances to Choose Candidate Rankings

Consider $q$, WAV $\Theta^q$, transition matrix $A_q$, and ranking $R_q$. Further consider any candidate $\Theta^{cand}$, $A_{cand}$ and $R_{cand}$ from the repository that may be used to approximate $R_q$.

For SchemaApprox, defined at the schema level, we consider a schema level metric to choose candidates; SchemaApprox should minimize the distance of this metric between $\Theta^{cand}$ and $\Theta^q$. We considered the Euclidean distance $\boldsymbol{\pi} = ||\Theta^{cand} - \Theta^q||_2$ and the first norm $\boldsymbol{\pi_1} = ||\Theta^{cand} - \Theta^q||$.

For DataApprox, defined at the level of the data graph, we consider an objective function that will minimize the distance between $A_{cand}$ and $A_q$. Let matrix $A_{diff} = A_{cand} - A_q$. The difference between $A_{cand}$ and $A_q$ can be naturally represented by the matrix norms of $A_{diff}$. The definition for entry-wise matrix norm is an extension to

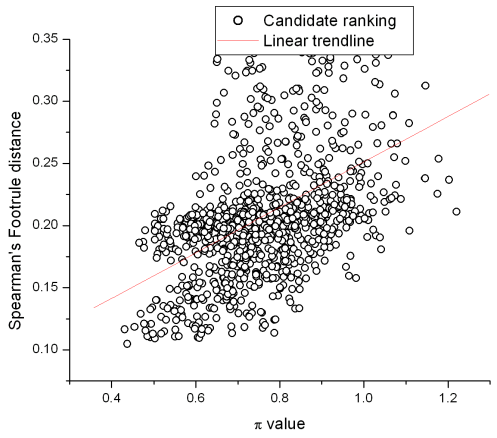


Fig. 5. The correlation between $\pi$ and Spearman's Footrule distance (SchemaApprox).

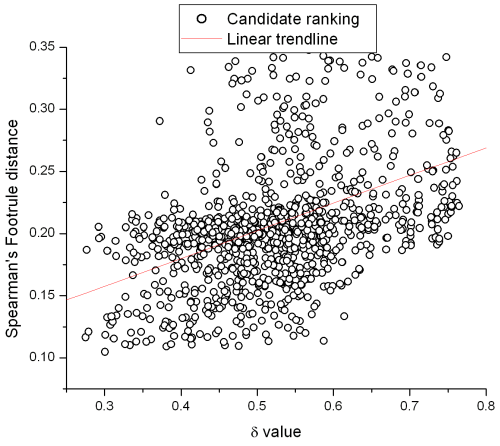


Fig. 6. The correlation between $\delta$ and Spearman's Footrule distance (DataApprox).

the definition for vector norm [28]. Using the p-norm for vectors, we have the following:

$$\|A_{diff}\|_p = \left( \sum_{i=1}^{m} \sum_{j=1}^{n} |A_{diff}[i,j]|^p \right)^{1/p}$$

When $p = 2$, this is the Frobenius norm, and when $p = \infty$ this is the maximum norm. The alternatives for the distance to be considered for DataApprox are as follows:

- The maximum norm for $A_{diff}$, $\boldsymbol{\delta} = max_{\{i,j\}}\{|A_{diff}[i,j]|\}$.
- The 1-norm for $A_{diff}$, $\boldsymbol{\sigma} = \sum_{i,j} |A_{diff}[i,j]|$.
- The Frobenius norm for $A_{diff}$, $\boldsymbol{\phi} = \sqrt{\sum_{i,j} A_{diff}[i,j]^2}$.

It is widely accepted that an order based ranking is the best measure of quality. We therefore consider the well known Spearman's Footrule Distance [29] between the approximate ranking $R$ and the ideal ranking $R_q$ as a proxy for the quality of our approximate solutions. Unfortunately, minimizing an order based ranking is theoretically very difficult; thus, we are unable to apply an order based ranking when solving SchemaApprox or DataApprox.

To empirically understand the correlation behavior between the distance metrics that are minimized by SchemaApprox or DataApprox, and the Spearman's Footrule distance, we generate a ranking repository with 1000 random authority flow rankings for the DBLP dataset. We consider a test set of 20 user WAVs $\Theta^q$, where non-zero edge weights are used only for Paper-Paper and Paper-Author edges, i.e., authority flows only on these edge types. The purpose of this experiment is to see which distance measures seem to correlate more strongly with the ranking distance; these measures are then used to guide our optimization challenge. Note that in our evaluation, reported in Section 7, we considered several test WAVs where many (all) edge types may be assigned non-zero values.

We compute the distances $\pi_1$ and $\pi$ used for SchemaApprox, as well as the 3 norms $\delta$, $\sigma$ and $\phi$ used for DataApprox, over the 20 user WAVs; we report on the average over the 20 user WAVs. We also compute the average Spearman's Footrule Distance over the 20 user WAVs. Figure 5 is a scatterplot of the distance $\pi$ against the Spearman's Footrule distance. Figure 6 is the corresponding plot for $\delta$. Each point in the plot is a candidate ranking in the repository. The lines in the figures are the linear trendlines. The correlation coefficients are 0.4191 ($\delta$), 0.3459 ($\sigma$), 0.4036 ($\phi$), 0.45968 ($\pi$) and 0.40815 ($\pi_1$). We conclude that the matrix norm $\delta$ seems to be the best metric for DataApprox and the Euclidean distance $\pi$ is best for SchemaApprox.

Note that the correlation for $\pi$ appears higher than that for $\delta$ for this test set of WAVs; these WAVs only have a non-zero edge weight for the Paper-Paper and Paper-Author edges. This no longer holds for the richer test WAVs of Section 7, where all edge weights may have non-zero values. A related discussion of the pros and cons of solving SchemaApprox or DataApprox is (somewhat) independent of the relative correlation of $\pi$ or $\delta$; this discussion is presented in Section 3.3.

### 3.3 Solution Approaches

**SchemaApprox: Approximation at the schema level**

Recall that SchemaApprox uses the Euclidean distance $\pi$. Let $\Theta^{comb}$ be a linear combination of $m$ weight assignment vectors (WAVs) selected from the repository $\mathcal{S}$. The goal is to find $\beta$ values used to combine the $m$ WAVs, such that the Euclidean distance $\pi$ between the linear combination $\Theta^{comb}$ and the user WAV $\Theta^q$ is minimized.

SchemaApprox is defined as follows:

$$\begin{array}{ll} \text{minimize} & \pi = ||\Theta^{comb} - \Theta^{q}||_2 \\ \text{subject to:} & \\ & \Theta^{comb} = \sum_{l=1}^{m} \beta_l \Theta_l \\ & \sum_{l=1}^{m} \beta_l = 1 \\ & 0 \leq \beta_l \qquad\qquad for\ all\ 1 \leq l \leq m \end{array} \tag{5}$$

**DataApprox: Approximation at the data graph level**

First, we introduce the linearity theorem, which is proved below. Given two weight assignment vectors, there exists a random walk that is defined by combining their independent random walks; the ranking vector is a linear combination of the two independent ranking vectors.

*Theorem 2:* **(Authority Transfer Weights Linearity Theorem)** Let $R_1$ and $R_2$ be two ranking vectors for weight assignment vectors $\Theta_1$ and $\Theta_2$ respectively. Let $A_1$ and $A_2$ be the corresponding transition matrices. Let $\beta_1$, $\beta_2$ be constants such that $\beta_1, \beta_2 \geq 0$ and $\beta_1 + \beta_2 = 1$. For a random walk with transition matrix $A$, where $A[i,j] = \frac{\beta_1 A_1[i,j] R_1[i] + \beta_2 A_2[i,j] R_2[i]}{\beta_1 R_1[i] + \beta_2 R_2[i]}$, the ranking vector is $R = \beta_1 R_1 + \beta_2 R_2$.

To be more general, Theorem 2 is extended to $m$ surfers as follows: We are given a set of $m$ weight assignment vectors and their corresponding ranking vectors $\mathcal{S} = \{(\Theta_1, R_1), (\Theta_2, R_2), \cdots, (\Theta_m, R_m)\}$. Let $A_l$ be the transition matrix for $\Theta_l$. We define the behavior of an aggregate surfer with transition matrix $A_{agg}(\mathcal{S})$ as follows:

$$A_{agg}(\mathcal{S})[i,j] = \frac{\sum_{l=1}^{m} \beta_l A_l[i,j] R_l[i]}{\sum_{l=1}^{m} \beta_l R_l[i]} \tag{6}$$

From Theorem 2 and particularly from its generalization (Equation 6) we infer that if we select $m$ rankings and compute appropriate $\beta$'s to combine them such that $A_{agg}(\mathcal{S})[i,j] = \frac{\sum_{l=1}^{m} A_l[i,j] R_l[i] \beta_l}{\sum_{l=1}^{m} R_l[i] \beta_l}$ is close to $A_q$, then $\sum_{i=1\ldots m} \beta_i \cdot R_i$ is a good approximation of $R_q$. We define the DataApprox optimization problem as follows:

$$\begin{array}{lll} \text{minimize} & \delta & \\ \text{subject to:} & & \\ |A_{agg}(\mathcal{S})[i,j] - A_q[i,j]| & \leq & \delta,\ for\ all\ (i,j) \\ \sum_{l=1}^{m} \beta_l = 1 & & \\ 0 \leq \beta_l & & for\ all\ 1 \leq l \leq m \end{array} \tag{7}$$

### 3.4 Proof and Intuitive View of Theorem 2

*Proof:* We first show that $A$ is row stochastic given that $A_1$ and $A_2$ are row stochastic. We know that $\sum_{j=1}^{n} A_1[i,j] = 1$ and $\sum_{j=1}^{n} A_2[i,j] = 1$. For any row $i$, we have:

$$\begin{array}{rcl} \sum_{j=1}^{n} A[i,j] & = & \frac{\beta_1 R_1[i] \sum_{j=1}^{n} A_1[i,j] + \beta_2 R_2[i] \sum_{j=1}^{n} A_2[i,j]}{\beta_1 R_1[i] + \beta_2 R_2[i]} \\ & = & \frac{\beta_1 R_1[i] + \beta_2 R_2[i]}{\beta_1 R_1[i] + \beta_2 R_2[i]} \\ & = & 1 \end{array}$$

If we complement the random walk with jumps from dangling pages then the Markov Chain we defined is irreducible and aperiodic. The ranking scores converge to a unique vector $R$. Next we show that $R$ converges to $\beta_1 R_1 + \beta_2 R_2$.

$$\begin{array}{rcl} & & \beta_1 R_1 + \beta_2 R_2 \\ & = & \beta_1(\epsilon A_1^T R_1 + (1-\epsilon)P) + \beta_2(\epsilon A_2^T R_2 + (1-\epsilon)P) \\ & = & \epsilon(\beta_1 A_1^T R_1 + \beta_2 A_2^T R_2) + (\beta_1 + \beta_2)(1-\epsilon)P \\ & = & \epsilon(\beta_1 A_1^T R_1 + \beta_2 A_2^T R_2) + (1-\epsilon)P \end{array}$$

Let $v_1 = \beta_1 A_1^T R_1 + \beta_2 A_2^T R_2$ and $v_2 = A^T(\beta_1 R_1 + \beta_2 R_2)$. Next we show that $v_1 = v_2$. For the $j$-th entry $v_1[j]$ in vector $v_1$, we have:

$$\begin{array}{rcl} & & v_1[j] \\ & = & \beta_1(\sum_{i=1}^{n} A_1[i,j] R_1[i]) + \beta_2(\sum_{i=1}^{n} A_2[i,j] R_2[i]) \\ & = & \sum_{i=1}^{n} \beta_1 A_1[i,j] R_1[i] + \beta_2 A_2[i,j] R_2[i] \\ & = & \sum_{i=1}^{n} \frac{\beta_1 A_1[i,j] R_1[i] + \beta_2 A_2[i,j] R_2[i]}{\beta_1 R_1[i] + \beta_2 R_2[i]} (\beta_1 R_1[i] + \beta_2 R_2[i]) \\ & = & \sum_{i=1}^{n} A[i,j](\beta_1 R_1[i] + \beta_2 R_2[i]) \\ & = & v_2[j] \end{array}$$

Since $\beta_1 A_1^T R_1 + \beta_2 A_2^T R_2 = A^T(\beta_1 R_1 + \beta_2 R_2)$, we have:

$$\beta_1 R_1 + \beta_2 R_2 = \epsilon A^T(\beta_1 R_1 + \beta_2 R_2) + (1-\epsilon)P$$

This concludes the proof that $R = \beta_1 R_1 + \beta_2 R_2$. □

An intuitive view of Theorem 2 (Authority Transfer Weights Linearity Theorem) is that if we know the behavior of two individual random surfers, that is, the way they "surf" ($A_1$, $A_2$) and the expected probability of being at each page ($R_1$, $R_2$), then for a new random surfer, who "surfs" in a way that combines the two surfers, specifically $\frac{\beta_1 A_1[i,j] R_1[i] + \beta_2 A_2[i,j] R_2[i]}{\beta_1 R_1[i] + \beta_2 R_2[i]}$ where $\beta_1$ and $\beta_2$ are constants, then the expected probability of the new surfer being at each page (R) is a linear combination of $R_1$ and $R_2$: $R = \beta_1 R_1 + \beta_2 R_2$. We call this random surfer an *aggregate surfer*.

### 3.5 Complexity of SchemaApprox

SchemaApprox, which is defined in Equation 5, can be solved using an approach to solve the Least Squares Problem [30]. A linear system of equations is *overdetermined* if there are more equations than unknowns, which often requires an approximate solution; it is as follows:

$$\sum_{j=1}^{n} X_{ij} \beta_j = y_i, (i = 1, 2, \cdots, m)$$

There are $m$ linear equations, $n$ unknowns ($\beta_1, \beta_2, \cdots, \beta_n$), and $m > n$. Let $X$ be the $m \times n$ matrix representing all coefficients $X_{ij}$ and $\beta$ be the vector of all $\beta_j$. The system can be written as $X\beta = y$.

Since an overdetermined system usually has no solution, the goal is to find the $\beta$ vector that is the closest to satisfying all the equations. *Least squares* is one of the commonly used approximation criterion. The intuition is

to minimize the distance between $X\beta$ and $y$. The least squares problem formulation is as follows:

$$argmin_\beta \sum_{i=1}^{m} |y_i - \sum_{j=1}^{n} X_{ij}\beta_j|^2 = argmin_\beta ||y - X\beta||_2$$

The least squares problem can be solved using the Singular Value Decomposition (SVD) of a matrix [31]. SVD is an important factorization of matrix. The SVD of an $m \times n$ matrix $X$ can be computed in $O(mn^2)$ using a 2-stage algorithm [30].

Given that the SchemaApprox problem can be mapped to the least squares problem, the only known approach to solve SchemaApprox has cubic complexity.

The complexity of DataApprox is discussed in Section 6.

# 4 Error Analysis for SchemaApprox and DataApprox

## 4.1 SchemaApprox

The properties that describe the behavior of SchemaApprox are unknown, i.e., there is no linearity theorem that describes the behavior of a random walk based on $\Theta^{comb}$, with respect to the random walk based on $\Theta^q$. The reason, intuitively, is that SchemaApprox operates on the schema graph instead of the data graph, and hence it ignores much of the information of the data graph. In particular, since SchemaApprox minimizes $\pi$, based on $\Theta^q$ defined at the schema level, it is insensitive to the properties of the actual *data graph*, in particular, the edge distribution for the different edge types.

In the rest of this section, we demonstrate a linearity theorem and error bound for DataApprox that is dependent on $\delta$; recall that $\delta$ is a maximum bound on the distance between any element of the approximate transition matrix $A_{DA}$ chosen by DataApprox, and $A_{OR}$, the ideal transition matrix. Since SchemaApprox does not utilize the edge distribution of the data graph, we know that it will not be associated with a similar maximum bound on the distance between any element of the approximate transition matrix $A_{SA}$ chosen by SchemaApprox, and $A_{OR}$, the ideal transition matrix. Further, our experiments will show that the data graph has a significant impact on the quality of the approximation.

To summarize, SchemaApprox can be reduced to a problem with cubic complexity. The approximate solution, represented by the transition matrix $A_{SA}$, cannot provide an error bound on the quality of SchemaApprox. For all of these reasons, we do not attempt to solve SchemaApprox, but instead rely on a heuristic as will be discussed in Section 7.

## 4.2 DataApprox

In this section, we provide a bound on the error corresponding to the ranking vector produced by DataApprox when it combines $m$ candidate rankings. We use the $L_1$ distance, a score-based distance, to determine the error; this distance is often used for error analysis. However, we note that the order of rankings and metrics such as P@K is more relevant to the user. Therefore, we report on the Spearman's Footrule distance [32] in our evaluation in Section 7.

The intuition behind the bound is as follows: Using the Authority Weights Linearity Theorem, the DataApprox ranking is described as a linear combination of candidate rankings. Simultaneously, the ranking for the aggregate surfer can be computed by applying traditional iterative approaches to reach convergence [1], [6], [33]. Thus, we can apply the iterative approach to obtain a bound.

Let $R_{DA}$ and $R_{OR}$ be the ranking vectors from DataApprox (approximate) and ObjectRank (exact) respectively, each with length $n$, where $n$ is number of nodes in the graph. Let $|E|$ be the number of edges in the graph. We will derive the $L_1$ distance between $R_{DA}$ and $R_{OR}$, $\| R_{DA} - R_{OR} \|_1$. Let $R^r_{DA}$ and $R^r_{OR}$ be the ranking vectors after the $r$-th iteration from DataApprox and ObjectRank. We first derive the error bound for the base case when $r = 1$.

*Lemma 4.1:* After the first iteration, the DataApprox ranking vector $R^1_{DA}$ satisfies the following:

$$\| R^1_{DA} - R^1_{OR} \|_1 \leq \epsilon|E|\delta$$

*Proof:*

$$\begin{array}{lcl} & & \| R^1_{DA} - R^1_{OR} \|_1 \\ (i) & = & \sum_{j=1}^{n} |R^1_{DA}[j] - R^1_{OR}[j]| \\ (ii) & = & \sum_{j=1}^{n} |\epsilon \sum_{i=1}^{n} A_{DA}[i,j] \cdot 1 + (1-\epsilon)d(j) \\ & & -\epsilon \sum_{i=1}^{n} A_{OR}[i,j] \cdot 1 - (1-\epsilon)d(j)| \\ (iii) & = & \epsilon \sum_{j=1}^{n} \sum_{i=1}^{n} |A_{DA}[i,j] - A_{OR}[i,j]| \\ (iv) & \leq & \epsilon|E|\delta \end{array}$$

We first expand the $L_1$ distance by the definition. In Equation $ii$, we calculate $R^1_{DA}[j]$ and $R^1_{OR}[j]$ assuming initial scores for DataApprox and ObjectRank are all 1s. $d(j)$ represents the probability to jump to page $v_i$. In ObjectRank, $d(j) = 1/|S|$ if page $v_j$ contains the keyword, where $|S|$ is the number of pages that contain the keyword; otherwise, $d(j) = 0$. We replace $|A_{DA}[i,j] - A_{OR}[i,j]|$ by $\delta$ in Inequality $iv$, which is specified by Equation 7. Since the number of non-zero entries in the graph is the number of edges $|E|$, we concludes the proof in Inequality $v$. □

Next we develop the error bound for the $L_1$ distance $\| R^r_{DA} - R^r_{OR} \|_1$ after $r$ iterations.

*Lemma 4.2:* After an arbitrary positive integer $r > 1$ iterations, the DataApprox ranking vector $R^r_{DA}$ satisfies:

$$\| R^r_{DA} - R^r_{OR} \|_1 \leq \epsilon \| R^{r-1}_{DA} - R^{r-1}_{OR} \|_1 + \epsilon\delta n$$

*Proof:*

$$
\begin{array}{lcl}
 & & \| R^r_{DA} - R^r_{OR} \|_1 \\
(i) & = & \sum_{j=1}^{n} |R^r_{DA}[j] - R^r_{OR}[j]| \\
(ii) & = & \sum_{j=1}^{n} |\epsilon \sum_{i=1}^{n} A_{DA}[i,j] \cdot R^{r-1}_{DA}[i] + (1-\epsilon)d(j) \\
 & & -\epsilon \sum_{i=1}^{n} A_{OR}[i,j] \cdot R^{r-1}_{OR}[i] - (1-\epsilon)d(j)| \\
(iii) & = & \epsilon \sum_{j=1}^{n} \sum_{i=1}^{n} |A_{DA}[i,j] \cdot R^{r-1}_{DA}[i] \\
 & & -A_{OR}[i,j] \cdot R^{r-1}_{OR}[i]| \\
(iv) & \leq & \epsilon \sum_{j=1}^{n} \sum_{i=1}^{n} |A_{DA}[i,j] R^{r-1}_{DA}[i] - \\
 & & A_{DA}[i,j] R^{r-1}_{OR}[i]| + \epsilon \sum_{j=1}^{n} \sum_{i=1}^{n} \delta R^{r-1}_{OR}[i] \\
(v) & \leq & \epsilon \sum_{j=1}^{n} \sum_{i=1}^{n} A_{DA}[i,j] |R^{r-1}_{DA}[i] - R^{r-1}_{OR}[i]| \\
 & & +\epsilon\delta \sum_{j=1}^{n} \sum_{i=1}^{n} R^{r-1}_{OR}[i] \\
(vi) & \leq & \epsilon \sum_{i=1}^{n} |R^{r-1}_{DA}[i] - R^{r-1}_{OR}[i]| \sum_{j=1}^{n} A_{DA}[i,j] \\
 & & +\epsilon\delta \sum_{j=1}^{n} \sum_{i=1}^{n} R^{r-1}_{OR}[i] \\
(vii) & \leq & \epsilon \sum_{i=1}^{n} |R^{r-1}_{DA}[i] - R^{r-1}_{OR}[i]| + \epsilon\delta \sum_{j=1}^{n} 1 \\
(viii) & \leq & \epsilon \| R^{r-1}_{DA} - R^{r-1}_{OR} \|_1 + \epsilon\delta n
\end{array}
$$

The idea of the first three equations in the proof is identical to the proof for Lemma 4.1, except that the scores from previous iteration are $R^{r-1}_{DA}$ and $R^{r-1}_{OR}$ instead of all 1s in Equation $ii$. Inequality $iv$ is derived based on the constraint that $|A_{DA}[i,j] - A_{OR}[i,j]| \leq \delta$. Reorganizing the terms leads to Inequality $v$ and $(vi)$. Because the transition matrix $A_{DA}$ is column stochastic, $\sum_{j=1}^{n} A_{DA}[i,j] = 1$. Considering $\sum_{i=1}^{n} R^{r-1}_{OR}[i] = 1$, the inequality is reduced to $vii$. The definition of $L_1$ distance concludes proof. □

We are now ready to prove the main theorem:

*Theorem 3:* **(DataApprox Error Bound)**

$$\| R_{DA} - R_{OR} \|_1 \leq \delta \frac{\epsilon}{1-\epsilon} n$$

*Proof:*

$$
\begin{array}{ll}
 & \| R^r_{DA} - R^r_{OR} \|_1 \\
\leq & \epsilon \| R^{r-1}_{DA} - R^{r-1}_{OR} \|_1 + \epsilon\delta n \\
\leq & \epsilon(\epsilon \| R^{r-2}_{DA} - R^{r-2}_{OR} \|_1 + \epsilon\delta n) + \epsilon\delta n \\
\leq & \epsilon^2 \| R^{r-2}_{DA} - R^{r-2}_{OR} \|_1 + (\epsilon^2 + \epsilon)\delta n \\
\leq & \epsilon^r |E| \delta + (\epsilon^r + \epsilon^{r-1} + \cdots + \epsilon)\delta n
\end{array}
$$

When the number of iterations $r$ becomes infinity, this gives

$$\| R_{DA} - R_{OR} \|_1 \leq \delta \frac{\epsilon}{1-\epsilon} n$$

□

Previous works [34] have studied the problem of PageRank perturbation stability assuming few edges are changed. In particular, [34] showed that PageRank is not rank-stable, that is, even small changes on the graph may lead to big changes on the ranking. In our case, modifying $\Theta^q$ may potentially change all edges in the graph. This gives a bigger importance to the bound of the above theorem, given the relatively weak correlation of the graph matrix and the ranking. Further, in Section 7 we show that this theoretical bound translates to close ranking approximation in practice.

The theorem shows that when $\delta$ is very small, DataApprox gives accurate ranking. Another interesting observation is that the error bound increases with $\epsilon$, whereas in the case of PageRank perturbation, it has been shown [35] that the error decreases with $\epsilon$. The reason is that in the case of perturbation, for large $\epsilon$, the scores of high-score nodes are influenced by thousands of paths and hence removing a few edges does not make a difference. On the other hand, in our problem, larger $\epsilon$ means that the authority transfer weights are multiplied by a larger constant and hence small differences in authority bounds translate to large differences in weights and hence to larger errors.

## 5 ScaleRank Architecture

In this section we discuss the architecture of the ScaleRank system which is an approximation of DataApprox; the algorithm is in Section 6. Figure 7 shows the architecture of ScaleRank. The input is a personalized WAV $\Theta^q$; the output are the top $K$ objects based on the personalized authority score. ScaleRank maintains a repository of $M$ candidate rankings. For each candidate ranking, its WAV $\Theta^{cand}$, and its ranking vector $R_{cand}$, are stored. Given $\Theta^q$, the Candidate Ranking Selector selects $m$ candidate rankings from the $M$ in the repository. We place a bound on $m$ candidates since $m$ can impact the running time as will be seen. ScaleRank then finds an efficient solution to DataApprox and determines $\beta_1, \ldots, \beta_m$, the best way to combine these $m$ rankings to compute the approximation $\sum_{i=1}^{m} \beta_i \cdot R_i$ of $R_q$. Finally a top $K$ algorithm is used to produce the top $K$ objects.

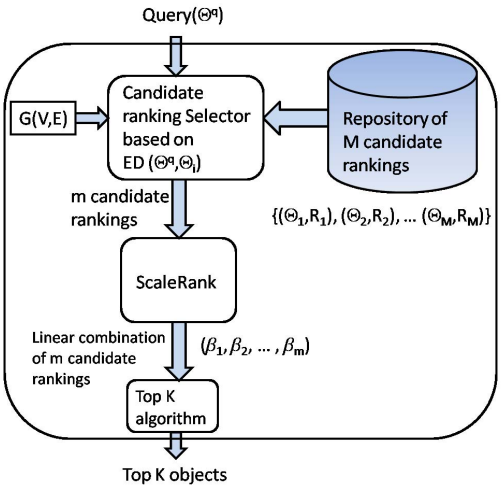


Fig. 7. The system architecture

**Materializing candidate rankings in the repository**: The set of rankings in the repository affects the quality of our approximation. Ideally, we would pre-compute rankings for each user's WAV. This is not feasible since the number of users may be huge and users may keep changing their WAVs.

A natural way to materialize candidate rankings is to generate a grid to represent all possible weight assignments in $\Theta$ for a given granularity, e.g., for each edge type in the semantic graph, we can select $W$ distinct values that are uniformly spread over some desired range. One drawback is that we would have to generate a very large number of candidate rankings in order to provide a uniform coverage of all the points in the grid. A small value of $W$ may not

provide uniform coverage of the grid of values of $\Theta$ and may produce poor candidate rankings.

To overcome this limitation, we generate $M$ (e.g. 1000) candidate rankings by randomly generating the values of $\Theta$; each candidate can be considered to correspond to a randomly selected point of the grid. We found that this random method provides good uniform coverage of the grid.

For each candidate ranking $i$ in the repository, its weight assignment vector $\Theta_i$ and its ranking vector $R_i$ are materialized. The repository can be represented by a set $\{(\Theta_1, R_1), (\Theta_2, R_2), \cdots, (\Theta_M, R_M)\}$.

**The candidate rankings selector**:
Among the $M$ candidate rankings in the repository, we choose the $m$ best candidate rankings. The intuition is that we use the best candidates in the repository to produce the combined ranking vector, to avoid the prohibitive cost of combining all candidates in the repository. Therefore, we use the Euclidean distance $||\Theta^q - \Theta_i||_2$ between $\Theta^q$ and each candidate $\Theta_i$. As we discuss below, this gives ScaleRank some of the benefits of SchemaApprox, which picks candidates based on the $\Theta$'s. By referring to a subset of relevant ranking candidates, ScaleRank is capable to process much larger datasets compared to DataApprox.

**The ScaleRank algorithm**:
Given the $m$ best candidates, ScaleRank finds an efficient solution to DataApprox. Note that the $m$ best candidates are selected using their WAV only (which is very fast), that is, using SchemaApprox semantics. Hence, ScaleRank can be also viewed as a hybrid algorithm that uses SchemaApprox ($\pi$) distance as a first filter and then solves DataApprox in the next stage. We emphasize however, that ScaleRank approximates DataApprox in the second stage and it is not an approximation for SchemaApprox. ScaleRank is discussed in detail in the next section.

**Creating a Merged Top K Ranking**:
The problem of combining multiple ranking vectors (sorted lists) to output the Top $K$ objects is well studied and there are numerous efficient algorithms [36], [37], [38]. For example, the well known TA algorithm [38] deals with monotone functions to aggregate the ranking scores. The weighted sum of the ScaleRank algorithm is a monotone function. Hence, we use TA to produce the Top K objects.

# 6 The ScaleRank Algorithm

## 6.1 The algorithm

The DataApprox problem is to solve Equation 7 where $A_{agg}(\mathcal{S})[i,j] = \frac{\sum_{l=1}^{m} A_l[i,j] R_l[i] \beta_l}{\sum_{l=1}^{m} R_l[i] \beta_l}$. The first constraint of Equation 7 sets an upper bound for the difference between two matrices. The intuition behind ScaleRank (Figure 8) is that the optimization problem of Equation 7 can be solved by solving a series of feasibility problems without addressing the objective function, that is, one can choose a $\delta$ and check if the constraints hold. Since $\delta$ is the absolute value of the difference between two entries of the two transition matrices, it is in the range $[0, 1]$. Therefore, we can use binary search to find the minimum $\delta$ with upper bound $u = 1$ and lower bound $l = 0$. The search continues until $|u - l| < \tau$, where $\tau$ is the user defined accuracy requirement. Given the candidate rankings $\mathcal{S}$, the data graph $G$, the query weight assignment $\Theta^q$, and accuracy requirement $\tau$ for $\delta$, we describe the $ScaleRank$ algorithm as follows:

**Algorithm** $ScaleRank(\mathcal{S}, G, \Theta^q, \tau)$
1. $u = 1$, $l = 0$
2. $min\_\delta = u$
3. **while** $(u - l \geq \tau)$ **do**
4. $\quad \delta = (u + l)/2$
5. $\quad$ **if** $(Feasibility(\mathcal{S}, G, \Theta^q, \delta))$
6. $\quad\quad min\_\delta = \delta$
7. $\quad\quad u = \delta$
8. $\quad$ **else**
9. $\quad\quad l = \delta$
10. **return** $min\_\delta$

Fig. 8. The outline of the ScaleRank algorithm.

The algorithm $ScaleRank$ finds the minimum $\delta$ such that the optimization problem in Equation 7 is feasible, and stores the $\beta$ vector which produces $min\_\delta$ in $Feasibility$ algorithm. The *while* loop is usually executed for around 10 times if we choose accuracy requirement $\tau = 0.001$. The Feasibility procedure in Line 5 of algorithm solves the Linear Programming problem of Equation 7 without the objective function, that is, for a given $\delta$.

## 6.2 Reduce the complexity of the feasibility problem

ScaleRank repeatedly solves a linear programming (LP) problem. The Simplex algorithm can typically provide solutions to the LP problem efficiently *in practice*. There are $|E|$ non-zero matrix entries, where $|E|$ is the number of edges in the graph. Recall that $m$ is the number of candidate rankings and $\tau$ be the accuracy requirement for the LP problem. The binary search to satisfy accuracy requirement takes at most $\lceil log_2 \frac{1}{\tau} \rceil$ iterations. Next, we consider several techniques to reduce the number of constraints of the LP.

### 6.2.1 One constraint per semantic type

The first constraint of Equation 7 can be rewritten as follows with the help of Equation 6:

$$|\sum_{l=1}^{m} \beta_l R_l[i](A_l[i,j] - A_q[i,j])| - \delta \sum_{l=1}^{m} \beta_l R_l[i] \leq 0 \ for\ all\ (i,j) \quad (8)$$

where in the first constraint of Equation 7, we replace $A_{agg}(S)[i,j]$ by Equation 6.

The constraint of Equation 8 addresses the transition entry differences between the new random walk and existing random walks. For an important page $v_i$, if $v_i$ leads to thousands of pages, does this imply thousands of

constraints? We will show we can reduce the number of constraints dramatically. To do this, we reformulate this constraint.

We look into the authority flow ranking definition. Let $e^T(v_i, v_j)$ be the semantic type of edge $(v_i, v_j)$. $\alpha(e^T(v_i, v_j))$ denotes the weight assignment for $e^T(v_i, v_j)$ in the new query, and $\alpha_l(e^T(v_i, v_j))$ denotes the weight assignment for $e^T(v_i, v_j)$ in candidate weight assignment $l$. $OutDeg(v_i, e^T(v_i, v_j))$ is the number of outgoing edges from page $v_i$, of type $e^T(v_i, v_j)$. According to the authority flow ranking definition, $A_q[i,j] = \frac{\alpha(e^T(v_i,v_j))}{OutDeg(v_i,e^T(v_i,v_j))}$. Equation 8 becomes as follows:

$$|\sum_{l=1}^{m} \beta_l R_l[i] \frac{\alpha_l(e^T(v_i, v_j)) - \alpha(e^T(v_i, v_j))}{OutDeg(v_i, e^T(v_i, v_j))}| - \delta \sum_{l=1}^{m} \beta_l R_l[i] \leq 0 \quad (9)$$

From Equation 9, it is clear that for outgoing edges from page $v_i$, if they belong to the same semantic type, the same constraint holds. Therefore, for outgoing edges, the number of constraints can be reduced to the number of semantic types departing from page $v_i$.

#### 6.2.2 Restricting to U constraints

PageRank-style ranking scores typically conform to a power law distribution with the Top K pages having very high scores [39]. Table 1 lists the sum of the normalized ranking scores of the Top K pages; K ranges from 20 to 2000, for a data graph with 1707898 nodes.

| Top K | 20 | 100 | 500 | 1000 | 1500 | 2000 |
|---|---|---|---|---|---|---|
| Sum | 0.1302 | 0.6349 | 0.9732 | 0.9858 | 0.9888 | 0.9905 |

TABLE 1
The sum of scores of the Top K pages.

Equation 9 describes the constraints for a node $v_i$ for each of its outgoing edge types $e^T(v_i, v_j)$.

If page $v_i$ is assigned high ranking score in existing ranking $R_h$, then the similarity of the weight assignment $\Theta_h$ to the query weight assignment $\Theta^q$ has more impact. In our LP formulation, we consider Top K pages in all selected candidate rankings. This way, we do not underestimate the impact of important pages from any selected candidate ranking. Let $U$ be the cardinality of the union of the Top $K$ objects from the $m$ candidates. The number of constraints would be $U$ in the LP formulation. Other ways to define $U$ are also possible.

#### 6.2.3 The range for $\delta$

For each edge type $T$ going from an object of type $p$ to an object of type $q$ (i.e., $T = e^T(v_i, v_j)$ if $v_i, v_j$ are of type $p, q$ respectively), it is trivial that Equation 9 is satisfied if

$$\delta \geq u_T = max_{l \in 1..m}\{|\alpha_l(T) - \alpha(T)|\} \quad (10)$$

given that $OutDeg(v_i, e^T(v_i, v_j)) \geq 1$.

Generalizing on all edge types $T \in 1..t$, we know that all instances of Equation 9 are trivially satisfied if we pick

$$\delta \geq u = max_{T \in 1..t} u_T \quad (11)$$

Hence, to save time, instead of searching for the smallest value for $\delta$ in $[0, 1)$, we only search for the smallest delta in $[0, u]$.

## 7 Experimental Evaluation

### 7.1 Experiment Description

#### 7.1.1 The dataset

Bibliographic databases (DBLP or CiteSeer) are frequently used to evaluate authority flow ranking [6], [2], [16]. We use the DBLP dataset (June 2008) to build a data graph that conforms to the schema graph of Figure 1. We crawled CiteSeerX [40] to get additional citation links. This dataset contains 8 edge types, 1707898 objects and 7704633 links.

We expect our results on DBLP should hold over a wide range of datasets since the DBLP graph possesses the typical power law edge distribution of many real-world graphs [41]. While our experiments used a WAV vector with at most 8 values, we think that this too is reasonable. While some graphs, e.g., the semantic Web, may have many edge types, it is unlikely that users would provide a personalization vector covering more than a few edge types.

#### 7.1.2 Factors affecting the quality of the algorithms

Next, we discuss the impact of the dataset and WAV characteristics. These observations will help us understand the experimental results. When an entity type has fewer outgoing semantic edge types, then their weights are typically higher in $\Theta$. ObjectRank favors such edge types. Similarly, the edge distribution at the data graph level has an impact. Consider an object where the outdegree for one edge type is significantly smaller than for another. Suppose their edge weights in $\Theta$ are similar. Then ObjectRank favors links where the outdegree is small. The $\beta$ vector of DataApprox, $(A_{agg}(\mathcal{S})[i,j] = \frac{\sum_{l=1}^{m} A_l[i,j] R_l[i] \beta_l}{\sum_{l=1}^{m} R_l[i] \beta_l}$ (Equation 6)) is dominated by links with such smaller outdegree.

Finally, $skew$ in $\Theta^q$ has an impact. If the weights are similar, then authority flow is uniform. However, when there is variance among weights, then authority flow is skewed. We observe in our experiments that the quality of the approximation can degrade with increasing $skew$ in $\Theta^q$.

#### 7.1.3 Evaluation metrics

For a given authority flow weight assignment $\Theta^q$, we compute the exact Global ObjectRank ranking vector and the ScaleRank approximate vector. While the L1 distance was used to obtain an error bound for ScaleRank, users are less interested in the actual scores for results and are more interested in the rank order. In the information retrieval (IR) literature, metrics such as precision, precision at R

(PR), mean average precision (MAP), recall and the F-measure are typically applied to user evaluation of ranked lists [42]; they work well when there is manually annotated ground truth. An alternate metric that has been widely used to evaluate ranking algorithms including PageRank and personalized PageRank is the normalized Spearman's Footrule Distance [29] between two vectors. This metric is appropriate in cases such as ours where there is no manually annotated ground truth and the ranked lists produced by two algorithms are being compared. Since there are many tied pages with the same score, we use an extension for ranking with ties [43]. We report on the Spearman's Distance averaged over up to 20 user WAVs and the complete result (or Top K results).

Note that there are no other works that tackle the problem of efficient personalization for varying WAVs and hence we do not compare to previous approaches, except the original ObjectRank execution [6].

#### 7.1.4 Baseline algorithm PickOne

We compare ScaleRank against a baseline algorithm PickOne. Recall that the Euclidean distance $\pi$ had the strongest correlation with the Spearman's Footrule distance, for all candidate rankings; this was the motivation to define SchemaApprox. The PickOne algorithm mimics SchemaApprox. PickOne calculates the Euclidean distance $||\Theta^q, \Theta^{cand}||_2$ and chooses the candidate with the minimum Euclidean distance. Note that PickOne satisfies the constraint of Theorem 3.

We do not include results for a heuristic implementation of SchemaApprox that involves combining multiple ranking for the reasons explained in Section 4.1.

We implemented ScaleRank, PickOne and ObjectRank in Java. Our experiments were run on a Solaris machine with two 2.8 GHz dual-core processors and 12 GB RAM.

#### 7.1.5 Ranking repository

Recall that we discussed strategies to generate candidate rankings in Section 5. We generate 1000 candidate rankings by randomly generating the values of $\Theta$; each candidate can be considered to correspond to a randomly selected point of the grid. The randomized method provides good uniform coverage of the grid.

We measured the storage requirement for the Top 1000 ($K = 1000$) objects for 1000 ($M = 1000$) rankings to be 30 MB. If we want to reduce the space, the ScaleRank heuristic can use a smaller $K$ as addressed in Section 6.2.2. We experiment with $K = 50$ (see Figure 9) and above.

#### 7.1.6 User and Repository WAVs

In Table 2, we show user WAVs and candidate WAVs from the repository. The second column shows the values for $\Theta$ for 7 edge types; they are ((conference,year),(year,conference), (year,paper), (paper,year), (paper,paper), (paper,author), (author,paper)). The is-cited-by edge (paper,paper) has 0.0 weight.

| | weight assignment vector |
|---|---|
| user WAV 1 | $(1, 0.5, 0.5, 0.33, 0.33, 0.33, 0.1)$ |
| user WAV 2 | $(0.03, 0.4, 0.5, 0.05, 0.02, 0.01, 0.7)$ |
| candidate 1 | $(0.99, 0.18, 0.75, 0.35, 0.30, 0.23, 0.26)$ |
| candidate 2 | $(0.28, 0.18, 0.75, 0.37, 0.22, 0.36, 0.088)$ |

TABLE 2
A sample of user and candidate WAVs.

#### 7.1.7 ScaleRank parameter selection

While there are many parameters, the significant ones are as follows: (1) $M$: the number of rankings in the repository; (2) $U$: the number of constraints in the LP; (3) $\tau$: a factor to control the bound on $\delta$. Our experiments show that the quality of ScaleRank improves with higher values of $M$ and $K$ (and hence of $U$). Increasing $M$ increases the preprocessing time and repository space, whereas increasing $K$ increases execution time. The value of $\tau$ is less critical as is observed in the running time presented in Section 7.6. We found $\tau$ of $0.01$ to be a good choice.

### 7.2 The impact of the Top K on ScaleRank

For this experiment we consider the $m = 10$ best candidate rankings chosen by the Candidate Ranking Selector. We report on the ScaleRank quality (Spearman's Footrule distance) as we vary $K$, which determines the number of constraints $U$. Recall from Section 6.2.2 that we consider only $U$ constraints in the feasibility problem corresponding to the union of the Top $K$ objects with greatest scores in the $m$ candidate rankings. We report on the average over 20 user WAVs. Figure 9 reports on the ScaleRank distance of the whole result lists when $K$ is varied from 50 to 500; the corresponding number of constraints $U$ ranges from 500 to 5000 for $m = 10$. The left vertical axis shows the scale for the Spearman's Footrule distance. The ScaleRank distance values are represented by blue crosses, while the distance values for PickOne are the red squares (horizontal line). The quality of ScaleRank is much higher than PickOne. ScaleRank improves on the distance value (approximately $30\%$ – $40\%$) for this range of increasing $K$. Interestingly, increasing $U$ leads to higher delta (weaker theoretical bound according to Theorem 2) but better error in practice, because more nodes (constraints) are considered. PickOne is indifferent to the value of $K$.

To understand ScaleRank improvement, the right vertical axis shows the scale of the $\delta$ values and the triangles report on the $\delta$ values for ScaleRank. As $U$ ($K$) increases, ScaleRank solves for more constraints in the LP and thus it has a more complete view of the candidate rankings. Thus, when $U$ is larger, the value of $\delta$ typically increases since the maximum norm $\delta$ is computed over a larger set of elements in the transition matrix. However, ScaleRank is able to benefit from the additional constraints and is able to produce a better approximation.

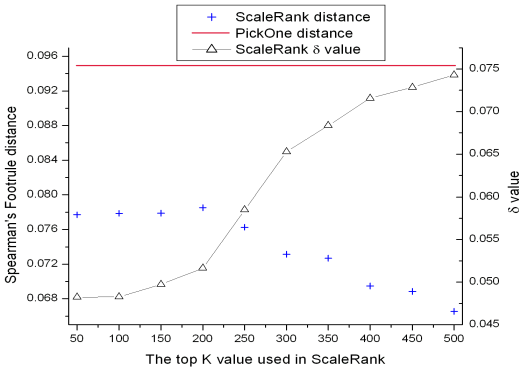


Fig. 9. The average Spearman's Footrule distance when the value of $K$ is varied.

### 7.3 The Accuracy of the Top K (Top 100) Ranking Results

As is well accepted in the information retrieval literature, metrics such as precision and recall, PR, or mean average precision (MAP) [42] are effective in determining the accuracy and goodness of a retrieval system. In the ranking literature, in the absence of human annotated ground truth, the approach that has been taken is to report on the ranking of the Top K results, e.g., the Top 100 results. The intuition is that there is more interest in the accuracy of the Top K results rather than results that are not in the Top K and therefore of much less interest to the user.

| Algorithm | Number of user WAVs | Average Distance |
|---|---|---|
| ScaleRank | 5 | 0.049 |
| ScaleRank | 10 | 0.079 |
| ScaleRank | 15 | 0.120 |
| PickOne | 5 | 0.084 |
| PickOne | 10 | 0.179 |
| PickOne | 15 | 0.303 |

TABLE 3
The Spearman's Footrule Distance for the Top 100 Ranking Results for ScaleRank and PickOne

Table 3 reports on the Spearman's Footrule distance for the Top 100 results of ScaleRank and PickOne. We report for a value of $K$ = 1000 for ScaleRank; PickOne is indifferent to $K$. We report on the average distance over 5, 10, and 15 user WAVs for ScaleRank and for PickOne; we sort the user WAVs by the result accuracy to observe any trends. As can be observed, the range of values of the Spearman's Footrule distance for the Top 100 results are much higher when compared with the distance over the entire ranking of the results as reported in Figure 9; this is to be expected since the Spearman's distance is normalized over the cardinality of the results.

What is of note is that ScaleRank provides an accurate ranking of the Top 100 results for a majority of the user WAVs; the average distance is as low as 0.049 for the 5 best user WAVs and increases to 0.079 for 10 user WAVs. The average is 0.120 over all 15 user WAVs. PickOne is a greedy heuristic and this is reflected in its rapidly degrading performance. The average distance is 0.084 for the 5 best user WAVs (compared to 0.049 for ScaleRank). The distance increases rapidly to 0.179 averaged over 10 user WAVs and to 0.303 over all 15 user WAVs.

These results confirm that a greedy heuristic may provide an inaccurate ranking of the Top 100 results and justifies the need for a complex hybrid solution such as ScaleRank.

### 7.4 The impact of M on ScaleRank

Figure 10 reports on the behavior of ScaleRank and PickOne when we increase the number $M$ of rankings in the repository. We consider 10 different ranking repositories, whose size varies from $M = 100$ to 1000. The $\delta$ values for ScaleRank are the triangles, the ScaleRank distance values are the blue crosses, and the distance for PickOne are the red squares. When the size of the repository increases, the value of $\delta$ for ScaleRank decreases. To explain, the algorithm has more candidates to select $m$, and this leads to the smaller feasible $\delta$.

We observe that ScaleRank consistently outperforms PickOne. As $M$ increases, the ScaleRank distance shows a decreasing trend. PickOne however does not show a decreasing trend. Recall that PickOne is a greedy heuristic. With larger $M$, PickOne may pick a new ranking with smaller WAV distance. However, from Figure 5, we have observed that the WAV distance $\pi$ is not always correlated with the Spearman's Footrule distance. Thus, PickOne can get confused by the larger $M$. In contrast, the ScaleRank optimization problem typically always benefits from larger $M$.

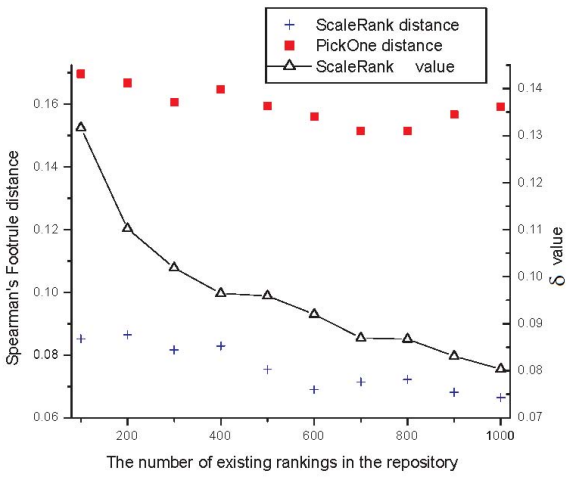


Fig. 10. The average Spearman's Footrule distance for varying $M$.

### 7.5 Skewed Weight assignment vector

In this experiment, we show that ScaleRank is much more robust than PickOne, for different experiment settings. This is because ScaleRank is sensitive to properties of the data graph but PickOne is insensitive. We consider user WAVs

where the authority weights are skewed. For instance, for $\Theta = \{0.05, 0.02, 0.03, 0.4, 0.4, 0.15, 0.2\}$, the sum of outgoing weights for *paper* is close to 1: $0.4 + 0.4 + 0.15$; this corresponds to edge types *(paper,year)*, *(paper,paper)*, *(paper,author)* respectively. In contrast, for the other node types, the sum of outgoing weights is $<< 1$. The average distance for ScaleRank varies from 0.08 to 0.10 while the average distance for PickOne varies from 0.18 to 0.20. For some user WAVs, the PickOne distance was as high as 0.3. Typically, the quality of ScaleRank is twice as good as PickOne.

### 7.6 Scalerank runtime

We report on the runtime of ScaleRank and compare it to the exact ObjectRank algorithm. Recall that the average runtime for ScaleRank is approximately 2 seconds whereas the exact the ObjectRank computation takes over 5 minutes for some parameter settings.

The ScaleRank algorithm (Figure 8) employs an LP subroutine in each iteration of the while loop in lines 3–9. The number of iterations is a function of the choice of the accuracy requirement $\tau$. In experiments, we choose $\tau = 0.001$ which leads to good approximations. For all cases, ScaleRank is executed for up to 10 iterations.

The ObjectRank algorithm is computed through iterations until convergence. In practice, it takes around 25 iterations on our data graph. In each iteration, the ObjectRank examines all the edges in the graph.

Our analysis shows that the ScaleRank is an efficient algorithm, since we typically choose top K less than 500 which leads to good approximation. For ObjectRank, however, there can be millions of links.

ScaleRank calls an LP solver during the binary search to find the smallest value of $\delta$. We used an open source LP solver *glpk* [44] and its Java interface [45]. It is reported that the commercial LP solver CPLEX is 10 to 100 times faster than *glpk* [46] for a range of problems. Thus, despite the slower execution of *glpk*, the execution times reported in Figure 11 reflect that ScaleRank is efficient and can be performed online at query time.

For the ObjectRank implementation, we set the damping factor $\epsilon$ to be 0.85, which is the standard value use in PageRank and ObjectRank, and the convergence of the algorithms is identified when the absolute value of the $L_1$ norm is less than 0.1. Typically this takes $25-26$ iterations to converge. The average runtime for ObjectRank on 20 user WAVs is 338 seconds; this is not shown in Figure 11 due to its different scale. Note that this runtime does not include the preprocessing time required for the graph to be loaded into memory.

Figure 11 reports on the initialization time, i.e., the time to pick the $m$ best rankings from the repository; this is the white bar. The blue bar is the execution time, i.e., the time to call the LP solver multiple times. ScaleRank typically calls the solver $9-10$ times to satisfy the accuracy requirement of $\tau = 0.001$. As $K$ increases, the number of constraints $K$ in the LP also increases.

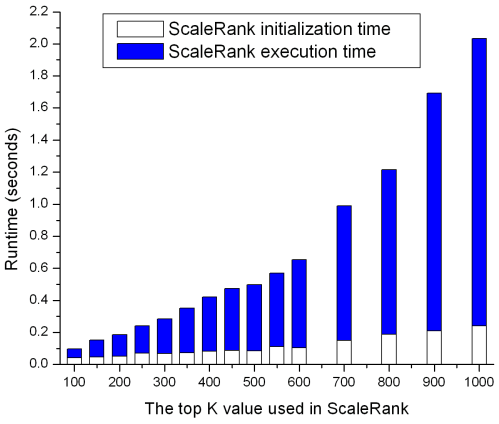


Fig. 11. Average runtime of ScaleRank for varying values of Top $K$

We conducted further experiments to determine the behavior of ScaleRank as we vary other parameters in the repository. We do not provide detailed figures due to lack of space. As we vary the number $M$ of candidate rankings in the repository, for Top $K$ = 500, the initialization time varied from 0.10 seconds to 0.35 seconds, while the execution time varied from 0.50 to 0.80 seconds. The total time was always below 1.0 seconds. Note that as $M$ increases, the runtime for ScaleRank does not always increase. As $M$ increases, Scalerank has more choices and can choose a better set of $m$ best candidates. This usually reduces the execution time.

To summarize, despite the use of a comparatively slow LP solver *glpk*, ScaleRank execution performance is very fast and makes it an option for runtime personalization.

## 8 Conclusions

We addressed the challenge of approximating personalized authority flow ranking. We defined two problems SchemaApprox and DataApprox that use a repository of rankings. We proved an Authority Transfer Weights linearity theorem for the aggregate surfer; this also describes the behavior of DataApprox and provides a bound on its approximation accuracy. We developed a heuristic solution ScaleRank for the DataApprox problem. Extensive experiments show that ScaleRank is efficient and has good quality.

## 9 Acknowledgments

This research was partially supported by NSF Awards IIS-0960963, IIS-0811922, IIS-0952347, HRD-0833093, CNS-1126619, and a Google Research Award.

## References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998.

[2] S. Chakrabarti, "Dynamic personalized PageRank in entity-relation graphs," in *WWW '07*. New York, NY, USA: ACM, 2007, pp. 571–580.

[3] F. Geerts, H. Mannila, and E. Terzi, "Relational link-based ranking," in *VLDB*, 2004, pp. 552–563.

[4] V. Hristidis, H. Hwang, and Y. Papakonstantinou, "Authority-based keyword search in databases," *ACM Trans. Database Syst.*, vol. 33, no. 1, pp. 1–40, 2008.

[5] R. Varadarajan, V. Hristidis, L. Raschid, M.-E. Vidal, L. Ibá nez, and H. Rodríguez-Drumond, "Flexible and efficient querying and ranking on hyperlinked data sources," in *EDBT '09*, 2009, pp. 553–564.

[6] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Objectrank: Authority-based keyword search in databases." in *VLDB*, 2004, pp. 564–575.

[7] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós, "Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments," *Internet Mathematics*, vol. 2, no. 3, 2005.

[8] T. H. Haveliwala, "Topic-sensitive PageRank," in *WWW '02*.

[9] H. Hwang, A. Balmin, B. Reinwald, and E. Nijkamp, "BinRank: Scaling dynamic authority-based search using materialized subgraphs," in *ICDE '09*, 2009, pp. 66–77.

[10] G. Jeh and J. Widom, "Scaling personalized web search," in *WWW '03*. New York, NY, USA: ACM, 2003, pp. 271–279.

[11] B. Bahmani, A. Chowdhury, and A. Goel, "Fast incremental and personalized PageRank," *Proc. VLDB Endow.*, vol. 4, pp. 173–184, December 2010.

[12] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized PageRank on MapReduce," *ACM SIGMOD*, 2011.

[13] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma, "Object-level ranking: bringing order to web objects," in *WWW '05*.

[14] S. Chakrabarti and A. Agarwal, "Learning parameters in entity relationship graphs from ranking preferences," in *PKDD*, ser. Lecture Notes in Computer Science, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., vol. 4213. Springer, 2006, pp. 91–102.

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

[16] R. Varadarajan, V. Hristidis, and L. Raschid, "Explaining and reformulating authority flow queries," in *ICDE '08*.

[17] *http://www.ncbi.nlm.nih.gov/*.

[18] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," in *SODA '03*, 2003, pp. 28–36.

[19] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[20] A. Borodin, G. Roberts, J. Rosenthal, and P. Tsaparas, "Link analysis ranking: algorithms, theory, and experiments," *ACM Transactions on Internet Technology*, vol. 5, no. 1, pp. 231–297, 2005.

[21] V. Hristidis, L. Raschid, and Y. Wu, "Scalable link-based personalization for ranking in Entity-Relationship graphs," *SIGMOD International Workshop on the Web and Databases (WebDB)*, 2011.

[22] L. Srour, A. I. Kayssi, and A. Chehab, "Personalized web page ranking using trust and similarity," in *AICCSA*, 2007, pp. 454–457.

[23] A. Kritikopoulos and M. Sideri, "The compass filter: Search engine result personalization using web communities." in *ITWP*, ser. Lecture Notes in Computer Science, pp. 229–240.

[24] R. Cooley, B. Mobasher, and J. Srivastava, "Data preparation for mining world wide web browsing patterns," *Knowledge and Inf. Systems*, vol. 1, pp. 5–32, 1999.

[25] M. Spiliopoulou and L. C. Faulstich, "WUM: A tool for Web utilization analysis," *Lecture Notes in Computer Science*, vol. 1590, pp. 184–203, 1999.

[26] K. Sugiyama, K. Hatano, and M. Yoshikawa, "Adaptive web search based on user profile constructed without any effort from users," in *WWW '04*, 2004, pp. 675–684.

[27] J. Teevan, S. T. Dumais, and E. Horvitz, "Personalizing search via automated analysis of interests and activities," in *SIGIR*, 2005.

[28] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, October 1996.

[29] M. Kendall, "Rank correlation methods," *Griffin*, 1962.

[30] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Society for Industrial Mathematics, 1995.

[31] C. Meyer, *Matrix Analysis and Applied Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, February 2001.

[32] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *WWW '01*.

[33] Y. Wu and L. Raschid, "Approxrank: Estimating rank for a subgraph," in *ICDE '09*, 2009, pp. 54–65.

[34] R. Lempel and S. Moran, "Rank-stability and rank-similarity of link-based web ranking algorithms in authority-connected graphs," *Inf. Retr.*, vol. 8, no. 2, pp. 245–264, 2005.

[35] S. Chien, C. Dwork, R. Kumar, D. R. Simon, and D. Sivakumar, "Link evolution: Analysis and algorithms," *Internet Mathematics*, vol. 1, no. 3, 2003.

[36] K. C.-C. Chang and S. won Hwang, "Minimal probing: supporting expensive predicates for top-k queries," in *SIGMOD '02*.

[37] R. Fagin, "Combining fuzzy information from multiple systems (extended abstract)," in *PODS '96*, 1996, pp. 216–226.

[38] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *PODS '01*.

[39] G. Pandurangan, P. Raghavan, and E. Upfal, "Using PageRank to characterize web structure," in *Computing and Combinatorics*, ser. Lecture Notes in Computer Science, O. Ibarra and L. Zhang, Eds., 2002, vol. 2387, pp. 1–4.

[40] *http://citeseerx.ist.psu.edu/*.

[41] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Phys. Rev. E*, vol. 64, no. 4, p. 046135, Sep 2001.

[42] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[43] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, "Comparing and aggregating rankings with ties," in *PODS '04*.

[44] *http://www.gnu.org/software/glpk/*.

[45] *http://bjoern.dapnet.de/glpk/index.htm*.

[46] *http://www.go.dlr.de/pdinfo_dv/glpk/GLPK_FAQ.txt*.


**Vagelis Hristidis** is an Associate Professor of Computer Science at UC Riverside. He received his PhD in Computer Science from UC San Diego. His key areas of expertise are Databases, Information Retrieval, and particularly the intersection of these two areas. His work in these areas has received more than 2,500 citations according to Google Scholar. His key achievements also include the NSF CAREER award, a Google Research Award, an IBM Award, and a Kauffmann Entrepreneurship Award.

**Yao Wu** received her PhD in Computer Science from the University of Maryland, College Park. Her research interests are information management, including link analysis ranking and optimization techniques. She is currently with Microsoft at Redmond.

**Louiqa Raschid** has made significant contributions towards solving the challenges of data management, data integration, and performance for multiple non-traditional domains. Her multi-disciplinary research spans the fields of computer science to business analytics, with a strong link to important data science applications including humanitarian disaster relief applications, discovery in life science annotations, human behavior modeling within social streams, and the next generation of financial cyberinfrastructure to manage financial ecosystems and supply chains. She has published approximately 150 papers in the leading conferences and journals in databases, scientific computing, Web data management, bioinformatics and AI. Her research has received multiple awards including over 25 grants from the NSF and DARPA. She has organized working groups on information mediation and biological data management for the NIH and DARPA. She is leading an effort sponsored by the NSF and the Office of Financial Research (Department of the Treasury) on the next generation financial cyberinfrastructure. She has advised and mentored over 30 Ph.D. or post-doctoral researchers including many women and minority students and has played a key role in the Sahana FOSS project for disaster information management. She has been a Distinguished Scientist of the ACM since 2008.