Scalable Link-based Personalization for Ranking in Entity-Relationship Graphs

Vagelis Hristidis
Florida International University
Miami, FL
vagelis@cis.fiu.edu

Yao Wu
University of Maryland,
College Park
College Park, MD
yaowu@cs.umd.edu

Louiqa Raschid
University of Maryland,
College Park
College Park, MD
louiqa@umiacs.umd.edu

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 for the first time an approach to achieve efficient edge-based personization using a combination of precomputation and runtime algorithms.

In particular, we apply our method to the personalized authority flow bounds of ObjectRank, i.e., a 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. Both SchemaApprox and DataApprox are expensive so we develop efficient heuristic implementations. ScaleRank is an efficient linear programming solution to DataApprox. PickOne is a greedy heuristic for SchemaApprox. Extensive experiments on the DBLP data graph show that ScaleRank provides a fast and accurate personalized authority flow ranking.

Keywords

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] 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 [11, 12].

Figure 1 [6] shows the *authority transfer schema graph* for DBLP, a bibliographic database for computer science publications [13]. There are 8 edge types and each edge type is associated with a numeric value representing the personalized authority weight in WAV Θ; e.g., a weight of 0.2 for the edge type from Paper to Author. Details of ObjectRank are in Section 2. Consider another example of the biological Web. 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.

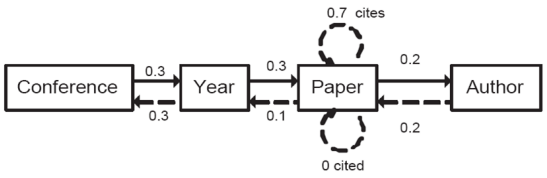

Figure 1: The DBLP authority transfer schema graph in ObjectRank ([6]).

Since users submit their queries and personalized WAV Θ 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 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.

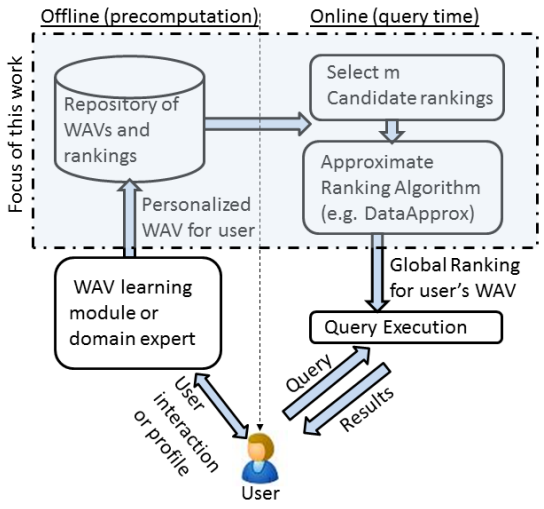

Figure 2: Incorporate personalized authority flow in user queries.

Figure 2 illustrates how a user interacts with the ranking system at precomputation and query times. During precomputation, 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 Θ is learned automatically by interacting with the user and exploiting user relevance feedback as explained in [14]. At query time, the system selects m candidate rankings from the repository, which are used to estimate the ranking for the requested Θ. 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. That is, our precomputation techniques facilitate the effective computation of Global (query-independent) ObjectRank [6] at runtime. Alternatively, we could use our approach to support a relatively small set of topic-sensitive ObjectRank rankings; Haveliwala [8] proposes building a set of topic-sensitive PageRank rankings for the Web. Supporting query-specific personalized ObjectRank for every keyword would be infeasible since a separate precomputation repository would be required for each keyword.

Example: For the schema graph of Figure 1, a user who believes that citations are the only important authority vote in a bibiographic 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 [14] 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 Θ^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 Θ^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 Θ^{comb}, to Θ^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 (δ), over all elements of the aggregate transition matrix of DataApprox and A_q, is minimized.

- We propose two heuristics, ScaleRank and PickOne, as DataApprox and SchemaApprox respectively are too expensive to facilitate interactive query response. ScaleRank chooses m candidates from the repository based on their Euclidean distance to Θ^q of the user; this is inspired by SchemaApprox and hence ScaleRank can also be viewed as a hybrid algorithm combining SchemaApprox and DataApprox, as discussed in Section 4. It applies linear programming to solve the DataApprox optimization problem. 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 [15] 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 approximate algorithms SchemaApprox and DataApprox. Section 4 describes the repository architecture and Section 5 presents ScaleRank. Experimental results are described in Section 6.

2. AUTHORITY FLOW RANKING: THE OBJECTRANK ALGORITHM

Given a keyword query, ObjectRank [6, 4] 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) Θ 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 Θ 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:

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

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:

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

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).

3. APPROXIMATION USING A REPOSITORY

3.1 The Problem Definition

Consider user WAV Θ^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 Θ^q be the user weight assignment vector. The goal is to approximate the authority flow ranking for Θ^q efficiently, with the maximal quality, by utilizing the precomputed ranking vectors in the repository.

3.2 Solution Approaches

SchemaApprox: Approximation at the schema level

Let Θ^{comb} be a linear combination of m weight assignment vectors (WAVs) selected from the repository $\mathcal{S}$. The goal is to find β values used to combine the m WAVs, such that the Euclidean distance π between the linear combination Θ^{comb} and the user WAV Θ^q is minimized. Let $\pi = ||\Theta^{comb} - \Theta^q||_2$. SchemaApprox is defined as follows:

$$\begin{array}{ll} \text{minimize} & \pi = ||\Theta^{comb} - \Theta^q||_2 \\ \text{subject to:} & \\ & \Theta^{comb} = \sum_{l=1}^m \beta_l \Theta_l \\ & \sum_{l=1}^m \beta_l = 1 \\ & 0 \le \beta_l \qquad for\ all\ 1 \le l \le m \end{array} \quad (4)$$

SchemaApprox can be solved using an approach to solve the Least Squares Problem [16].

DataApprox: Approximation at the data graph level

Before presenting DataApprox, we present a necessary theorem without proof due to space constraints:

THEOREM 1. **(Authority Transfer Weights Linearity Theorem)** *Let $R_1, \ldots, R_m$ be m ranking vectors for weight assignment vectors $\Theta_1, \ldots, \Theta_m$, with transition matrices $A_1, \ldots, A_m$. Let $\beta_1, \ldots, \beta_m$ be non-negative constants such that $\beta_1 + \cdots + \beta_m = 1$. For a random walk with transition matrix $A_{agg}(\mathcal{S})$, 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 ranking vector $R = \sum_{i=1\ldots m} \beta_i \cdot R_i$.*

From the linearity theorem we infer that if we select m rankings and compute appropriate β'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:

$$\begin{array}{lll} \text{minimize} & \delta & \\ \text{subject to:} & & \\ |A_{agg}(\mathcal{S})[i,j] - A_q[i,j]| & \le & \delta, for\ all\ entry\ (i,j) \\ \sum_{l=1}^m \beta_l = 1 & & \\ 0 \le \beta_l & & for\ all\ 1 \le l \le m \end{array} \quad (5)$$

Note that $m = M$ if we do not perform any selection of m candidates before defining the above optimization

4. SCALERANK ARCHITECTURE

In this section we discuss the architecture of the ScaleRank system which is an approximation of DataApprox; the algorithm is in Section 5. Figure 3 shows the architecture of ScaleRank. The input is a personalized WAV Θ^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

Θ^{cand}, and its ranking vector R_{cand}, are stored. Given Θ^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 $\sum_{i=1\ldots m} \beta_i \cdot R_i$ of R_q. Finally a top K algorithm is used to produce the top K objects.

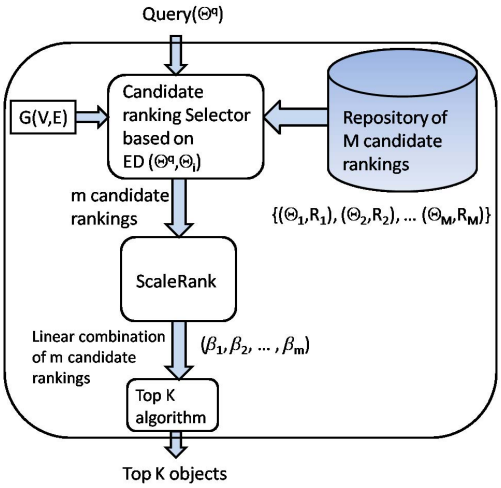

Figure 3: The system architecture.

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 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 Θ and may produce poor candidate rankings.

To overcome this limitation, we generate M (e.g. 1000) candidate rankings by randomly generating the values of Θ; 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 Θ_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 ranking selector: Among the M candidate rankings in the repository, we choose the m best candidate rankings. The intuition is that the best candidate should be chosen using the Euclidean distance $||\Theta^q, \Theta_i||_2$ between Θ^q and each candidate Θ_i. Using the above distance gives ScaleRank some of the benefits of SchemaApprox, which is evaluated on the Θ.

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 (π) 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:
We use the TA algorithm [17] to combine multiple ranking vectors and produce the Top K objects.

5. THE SCALERANK ALGORITHM

The DataApprox problem is to solve Equation 5 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 5 sets an upper bound for the difference between two matrices. The intuition behind ScaleRank is that the optimization problem of Equation 5 can be solved by solving a series of feasibility problems without addressing the objective function, that is, one can choose a δ and check if the constraints hold. Since δ 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 δ with upper bound $u = 1$ and lower bound $l = 0$. The search continues until $|u - l| < \tau$, where τ is the user defined accuracy requirement. Given the candidate rankings $\mathcal{S}$, the data graph G, the query weight assignment Θ^q, and accuracy requirement τ for δ, 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_δ

Figure 4: The outline of the ScaleRank algorithm.

The algorithm $ScaleRank$ finds the minimum δ such that the optimization problem in Equation 5 is feasible, and stores the β vector which produces min_δ in $Feasibility$ algorithm. The *while* loop is usually executed for around 10 times if we choose accuracy requirement $\tau = 0.1$. The Feasibility procedure in Line 5 of algorithm solves the Linear Programming problem of Equation 5 without the objective function, that is, for a given δ.

6. EXPERIMENTAL EVALUATION

6.1 Experiment Description

6.1.1 The dataset

Bibliographic databases (DBLP or CiteSeer) are frequently used to evaluate authority flow ranking [6, 2, 14]. We use the DBLP dataset (June 2008) to build a data graph that conforms to the schema graph of Figure 1. We crawled CiteSeerX [18] to get additional citation links. This dataset contains 8 edge types, 1707898 objects and 7704633 links.

We performed experiments on synthetic graphs but do not report on the results due to lack of space. 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 [19]. 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.

6.1.2 *Evaluation metrics*

For a given authority flow weight assignment Θ^q, we compute the exact Global ObjectRank ranking vector and the ScaleRank approximate vector. While the L_1 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 [20]; 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 [21] 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 a many tied pages with the same score, we use an extension for ranking with ties [22]. 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].

6.1.3 *Baseline algorithm PickOne*

We compare ScaleRank against a baseline algorithm: PickOne. The PickOne algorithm mimics SchemaApprox. PickOne calculates the Euclidean distance $||\Theta^q, \Theta^{cand}||_2$ and chooses the candidate with the minimum Euclidean distance.

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.

6.1.4 *Ranking repository*

Recall that we discussed strategies to generate candidate rankings in Section 4. We generate 1000 candidate rankings by randomly generating the values of Θ; 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. We experiment with $K = 50$ and above.

6.2 The Accuracy of the Top K Ranking Results

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 1: The Spearman's Footrule Distance for the Top 100 Ranking Results for ScaleRank and PickOne.

Table 1 reports on the Spearman's Footrule distance for the Top 100 results of ScaleRank and PickOne. 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.

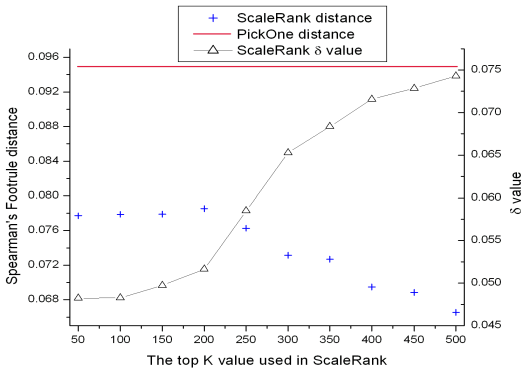

Figure 5: The average Spearman's Footrule distance when the value of K is varied.

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.

Figure 5 shows the Spearman's Footrule for the whole list of results, if we execute ScaleRank for various values of K. PickOne is constant since it does not consider K.

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

6.3 The impact of M on ScaleRank

Figure 6 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 δ 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 δ for ScaleRank decreases. To explain, the algorithm has more candidates to select m, and this leads to the smaller feasible δ.

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.

6.4 ScaleRank runtime

We report on the runtime of ScaleRank and compare it to the exact ObjectRank algorithm. 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 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 δ. We used an open source LP solver *glpk* [23]

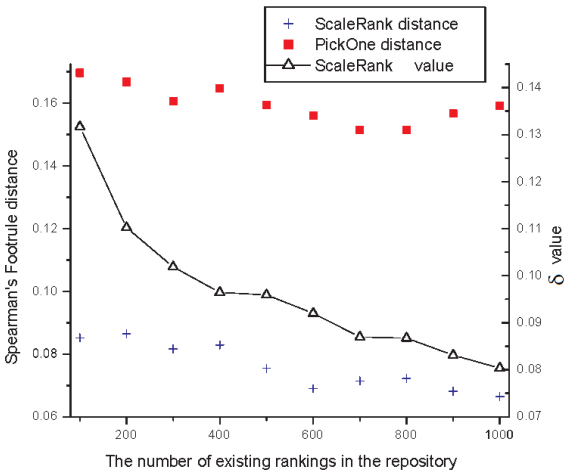

Figure 6: The average Spearman's Footrule distance for varying M.

and its Java interface [24].

The average runtime for ObjectRank on 20 user WAVs is 338 seconds; this is not shown in Figure 7 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 7 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.

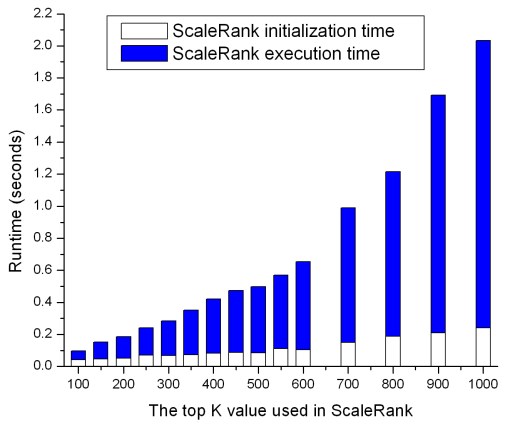

Figure 7: Average runtime of ScaleRank for varying values of Top K.

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.

7. 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 developed a heuristic solution ScaleRank for the DataApprox problem. Extensive experiments show that ScaleRank is efficient and has good quality.

8. ACKNOWLEDGMENTS

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

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

[12] S. Chakrabarti and A. Agarwal, "Learning parameters in entity relationship graphs from ranking preferences," in *PKDD*, 2006, pp. 91–102.

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

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

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

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

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

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

[19] 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.

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

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

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

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

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