FORank: Fast ObjectRank for Large Heterogeneous Graphs

Tomoki Sato*
University of Tsukuba
t.sato@kde.cs.tsukuba.ac.jp

Yuto Yamaguchi
Indeed Japan
yyamaguchi@indeed.com

Hiroaki Shiokawa*
University of Tsukuba
shiokawa@cs.tsukuba.ac.jp

Hiroyuki Kitagawa
University of Tsukuba
kitagawa@cs.tsukuba.ac.jp

ABSTRACT

ObjectRank is one of the popular graph mining methods that enables us to evaluate the importance of each vertex on heterogeneous graphs. However, it is computationally expensive to apply it to large graphs since ObjectRank needs to compute the importance of all vertices iteratively. In this work, we present a fast ObjectRank algorithm, *FORank*, that accurately approximates the keyword search results. *FORank* iteratively prunes vertices whose convergence score likely has less impact on the results during iterative computation. The experiments showed that *FORank* runs 7 times faster than ObjectRank computation with over 90% accuracy approximation.

CCS CONCEPTS

• Information systems → Graph-based database models;

KEYWORDS

ObjectRank; Graph data mining; Link structure analysis

ACM Reference Format:

Tomoki Sato, Hiroaki Shiokawa, Yuto Yamaguchi, and Hiroyuki Kitagawa. 2018. *FORank: Fast ObjectRank for Large Heterogeneous Graphs*. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3184558.3186950>

1 INTRODUCTION

ObjectRank [1] is a random walk-based graph analysis method that evaluates the importance of each vertex on the heterogeneous graphs where there are *multiple types* of vertices and edges. Although it is used in various applications, ObjectRank is computationally expensive since it requires iterative multiplications of matrix and vector to perform random walk. To address this problem, there is a line of works in the literature to accelerate ObjectRank computation [2]. However, these methods still require much computation time on large heterogeneous graphs since they need to run precomputations.

In this paper, we propose *FORank*, a fast ObjectRank algorithm that accurately approximates the keyword search results without

any precomputations. Our basic idea is that we exclude the vertices from the further computation if they do not have much impact on the ObjectRank scores of other vertices. In order to find which vertex to exclude, we theoretically derive the lower and upper bounds of the ObjectRank score. Our experiments showed that *FORank* runs 7 times faster than original ObjectRank computation while keeping the accuracy at over 90%.

2 PRELIMINARY: OBJECTRANK

ObjectRank first transforms records in the database into a heterogeneous graph. The graph is twofold: *Authority Transfer Schema Graph* and *Authority Transfer Data Graph* (*Schema Graph* and *Data Graph* for short). Schema Graph is a labeled directed graph that defines relationships of attributes in the database schema: a vertex and an edge represent an attribute and a relationship between attributes, respectively. Each edge takes a weight value from 0 to 1 that indicates the fraction of the importance among attributes.

Data Graph is a labeled directed graph that materializes all records in the database based on the Schema Graph. Each vertex indicates an attribute value in the database, and it has a set of keywords corresponding to the attribute. Edges in the Data Graph have a weight value that is calculated by dividing the weight defined in Schema Graph by the outgoing degree of the source vertex.

Given a keyword t and a Data Graph $G = (V, E)$, the ObjectRank score of each vertex $r_t(v)$ can be obtained by recursively applying the following equation until convergence:

$$\mathbf{r}_t = d\mathbf{A}\mathbf{r}_t + (1 - d)\mathbf{q}_t. \quad (1)$$

$\mathbf{A}$ is a $N \times N$ transition matrix of G , and $\mathbf{q}_t$ is the N -dimensional vector s.t. $q_t(v_i) = \frac{1}{|BS(t)|}$ for $v_i \in BS(t)$, otherwise $q_t(v_i) = 0$, where $|BS(t)|$ is the set of vertices that contains keyword t . As shown in Equation 1, ObjectRank consumes $O((N + M)T)$ times for each keyword, where N , M and T are the number of vertices, edges, and iteration steps until convergence. This incurs high computational cost for large heterogeneous graphs.

3 PROPOSED METHOD: FORANK

FORank reduces the number of vertices to be calculated by eliminating the ones with *small impact* on the final result. Thus, *FORank* identifies such vertices and prunes them iteratively from the further computations. In order to identify vertices to be pruned in the iterative computations, we theoretically derive upper bound $\bar{\mathbf{r}}_t^{(i)}$ and lower bound $\underline{\mathbf{r}}_t^{(i)}$ of the ObjectRank score $\mathbf{r}_t$. Since the upper bound $\bar{\mathbf{r}}_t^{(i)}$ always holds $\bar{\mathbf{r}}_t^{(i)} \geq \mathbf{r}_t$ (Lemma 3.3), we can prune vertex v

*Both authors contributed equally to the paper

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '18 Companion, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5640-4/18/04.

<https://doi.org/10.1145/3184558.3186950>

Algorithm 1 FORank

Input: Data Graph G ; keyword t ; threshold θ ; maximum # of iterations τ
Output: ObjectRank scores of all vertices excluding pruned ones

```

1:  $i \leftarrow 0, G_i \leftarrow G$ 
2: while  $i < \tau$  do
3:   for each  $v \in V_i$  do
4:     Get  $r_t^{(i)}(v)$ ,  $\underline{r}_t^{(i)}(v)$  and  $\bar{r}_t^{(i)}(v)$  by Equation (1), Definition 3.1 and 3.2, respectively.
5:   end for
6:   if all of ObjectRank scores of  $v \in G_i$  converges then
7:     stop algorithm
8:   end if
9:    $G_{i+1} \leftarrow \{v \in G_i : \bar{r}_t^{(i)}(v) \geq \theta\}, i \leftarrow i + 1$ 
10: end while

```

if its upper bound $\bar{r}_t^{(i)}(v)$ becomes smaller than the user-specified threshold θ at the i -th iteration, i.e. $\bar{r}_t^{(i)}(v) < \theta$.

Definition 3.1 (Lower bound). The lower bound $\underline{r}_t^{(i)}(v)$ of vertex v in the i -th iteration is defined as

$$\underline{r}_t^{(i)}(v) = (1 - d) \sum_{j=0}^i d^j p_t^{(j)}(v), \quad (2)$$

where $p_t^{(i)} = q_t$ if $i = 0$; otherwise, $p_t^{(i)} = A^i q_t$.

Definition 3.2 (Upper bound). The upper bound $\bar{r}_t^{(i)}(v)$ of vertex v in the i -th iteration is given as

$$\bar{r}_t^{(i)}(v) = \underline{r}_t^{(i)}(v) + d^{i+1} p_t^{(i)}(v) + (d^{i+1}/1 - d) \Delta_t^{(i)} \bar{A}(v), \quad (3)$$

where $\bar{A}(v) = \max\{A(v, u) : u \in V\}$; $\Delta_t^{(i)} = 1$ if $i = 0$, otherwise, $\Delta_t^{(i)} = \sum_{u \in V_0} \Delta_t^{(i)}(u)$, where $\Delta_t^{(i)}(u) = \max\{p_t^{(i)}(u) - p_t^{(i-1)}(u), 0\}$.

LEMMA 3.3 (LOWER AND UPPER ESTIMATION). In the i -th iteration, lower/upper bounds $\underline{r}_t^{(i)}(v)$, $\bar{r}_t^{(i)}(v)$ satisfy $\underline{r}_t^{(i)}(v) \leq r_t(v) \leq \bar{r}_t^{(i)}(v)$.

PROOF. Omitted due to the space limitation. $\square$

Based on Lemma 3.3, FORank can estimate the upper bound of the final score. FORank prunes vertices whose upper bounds are smaller than the user-specified threshold θ in each iteration. The pseudo-code of our proposal, FORank, is shown in Algorithm 1.

4 EXPERIMENTS

We compared three algorithms, FORank, ObjectRank [1] and the state-of-the-art algorithm, BinRank [2]. BinRank approximates ObjectRank by using *subgraph construction*. As a precomputation, BinRank runs the ObjectRank and keeps a subgraph by pruning vertices whose ObjectRank scores are smaller than the user-specified threshold for each clustered keywords. When query keyword t is given, BinRank re-computes the ObjectRank scores on the subgraph generated from the cluster that has keyword t . All experiments were conducted on a Linux server with 3.5GHz CPU and 128GB RAM. We used DBLP bibliographic dataset published by AMiner [3], which has 1,238,266 vertices and 5,149,294 edges.

Efficiency: Figure 1 indicates that FORank is much faster than ObjectRank and BinRank except for the significantly small threshold values. Specifically, FORank cuts the execution time by up to 86% from ObjectRank. ObjectRank iteratively computes all vertices and edges until convergence. On the other hand, FORank computes only the key vertices and edges by incrementally pruning low impact vertices during iterative computations. Figure 2 shows how many vertices remain in the subgraph G_i in the i -th iteration. As shown in Figure 2, FORank successfully reduces the number of vertices to be computed as the iteration number increases. Specifically, FORank removes from 98% to 99% of vertices in G_0 through only 6 to 11 iterations. Although FORank is also more efficient than the state-of-the-art algorithm BinRank when $\theta \geq 0.1$, its improvement is moderated compared with ObjectRank. However, BinRank requires the subgraph construction before querying a keyword. Figure 3 shows the precomputation time of BinRank varying the inner-parameter *maxBinSize*, which controls the size of subgraphs. Since BinRank consumes more than 10 hours under the all settings we examined, thus FORank is also faster than BinRank.

Accuracy: We evaluated the top- k approximation accuracy using $\text{Prec}(t, k) = |\text{CompSet}(t, k) \cap \text{ORSet}(t, k)|/k$, where $\text{CompSet}(t, k)$ is the set of top- k vertices for keyword t obtained by FORank or BinRank, and $\text{ORSet}(t, k)$ is those by ObjectRank. Figure 4 indicates that FORank showed better accuracy than BinRank: it achieved almost 90% accuracy compared with the results of ObjectRank.

5 SUMMARY

We propose FORank, a fast algorithm that approximates the keyword search results. FORank iteratively prunes vertices whose convergence score likely has less impact on the results during iterative computations using upper and lower bounds. Our experiments using real dataset showed that FORank can obtain results about 7 times faster than ObjectRank while keeping the accuracy at 90%.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number JP16H06650, JST ACT-I, and Interdisciplinary Computational Science Program in CCS, University of Tsukuba.

REFERENCES

- [1] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. 2004. ObjectRank: Authority-Based Keyword Search in Databases. In *Proc. VLDB*. 564–575.
- [2] Heasoo Hwang, Andrey Balmin, Berthold Reinwald, and Erik Nijkamp. 2009. BinRank: Scaling Dynamic Authority-Based Search Using Materialized SubGraphs. In *Proc. ICDE*. 66–77.
- [3] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnet-Miner: Extraction and Mining of Academic Social Networks. In *Proc. KDD*.