Research Article

Monte Carlo Based Personalized PageRank on Dynamic Networks

Zhang Junchao, 1,2 Chen Junjie, 1 Jiancheng Song, 2 and Rong-Xiang Zhao 3

1 College of Computer Science and Technology, Taiyuan University of Technology, Taiyuan 030024, China
2 Shanxi Key Laboratory of Coal Mining Equipment and Safety Control, Taiyuan 030024, China
3 Shanxi Taiyuan Tideflow Electronic Technology Co., Ltd., Taiyuan 030024, China

Correspondence should be addressed to Jiancheng Song; sjc6018@163.com

Received 24 June 2013; Accepted 25 August 2013

Academic Editor: Deguang Le

Copyright © 2013 Zhang Junchao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In large-scale networks, the structure of the underlying network changes frequently, and thus the power iteration method for Personalized PageRank computation cannot deal with this kind of dynamic network efficiently. In this paper, we design a Monte Carlo-based incremental method for Personalized PageRank computation. In a dynamic network, first, we do a random walk starting from each node and save the performed walks into a fingerprint database; second, we update the fingerprint database in a fixed time interval with our proposed update algorithm; finally, when a query is issued by a user, we estimate the Personalized PageRank vector by our proposed approximation algorithm. Experiments on real-world networks show that our method can handle multichanges of the underlying network at a time and is more efficient than related work, so it can be used in real incremental Personalized PageRank-based applications.

1. Introduction

Large-scale networks are ubiquitous in today’s world, such as World Wide Web, online social networks, and huge search and query-click logs regularly collected and processed by search engines. PageRank [1], which is the stability distribution of a random walk in a Markov chain, has emerged as an effective measure for ranking on large-scale networks. In addition, Personalized PageRank [2,3], which is the stability distribution of a random walk with restart in a Markov chain, has been proved to be an effective method of personalized ranking in many applications, such as link prediction [4–6], friend recommendation [7–9], and personalized searching [10, 11]. Given a network, there are two main methods for computing the PageRank or Personalized PageRank vector: one is power iteration applying the linear algebra proposed by Page et al. [1] and the other is the Monte Carlo approximation methods proposed by Litvak [12] and Fogaras and Rácz [13].

In practice, the structure of the underlying network is dynamic, such that Webpages join or leave frequently in World Wide Web and following relations appear or disappear frequently in an online social network. Recomputing PageRank or Personalized PageRank for every change, that appears or disappears in an edge, of the structure is impossible in large-scale networks, so incremental PageRank or Personalized PageRank algorithms are necessary for this situation.

1.1. Background. Given a graph \( G = (V, E) \), where \( V \) is the set of nodes, \( E \) is the set of edges, and the number of nodes and edges is \( n \) and \( m \), respectively; the adjacent matrix \( M \) of the graph \( G \) is

\[
m_{i,j} = \begin{cases} 
1 & (i, j) \in V, \\
0 & (i, j) \notin V.
\end{cases}
\]  

(1)

where \( i, j \in V \), and \( (i, j) \) are a directed edge from \( i \) to \( j \).

The transition possibility matrix \( P \) of the graph \( G \) is

\[
P_{i,j} = \frac{m_{i,j}}{\sum_{j=1}^{n} m_{i,j}},
\]  

(2)

where the weights on the outgoing edges of each node sum up to 1.
PageRank is the stability distribution of a random walk in a Markov chain. The main idea of random walk on a graph is that starting from a source node, at each step, jump to a random node with a probability $\varepsilon$ (usually called the teleport probability), and follow a random outgoing edge from the current with the probability $1 - \varepsilon$. If the number $k$ of random steps tends to be infinite, that is, $k \rightarrow +\infty$, then the probability that the random walk stops at each node is a constant, and, no matter how we choose the source node, the probability of stopping at each node does not change.

Personalized PageRank is the same as PageRank, except that the random walk jumps to the source node with a probability $\varepsilon$ at each step. Given a source (or called personalized) node $u$, the Personalized PageRank score of node $v$ is

$$\pi_u(v) = \varepsilon \delta_u(v) + (1 - \varepsilon) \sum_{i=1}^{n} p_{i,v} \cdot \pi_u(i),$$

where $\delta_u(v) = 1$ if and only if $u = v$, and $\delta_u(v) = 0$, otherwise.

### 1.2. Problem Statement

In this paper, we study the incremental computation of Personalized PageRank based on the Monte Carlo method; that is, given a graph, we do a random walk of fixed length starting from each node update the performed walks incrementally in fixed time interval, and approximate Personalized PageRank with the performed walks whenever needed.

Here, the core of the problem is designing an incremental update algorithm and a Personalized PageRank approximation algorithm.

### 1.3. Our Contribution

First, we design an approximation method for Personalized PageRank based on Monte Carlo method. We do a random walk of fixed length starting from each node, and save the performed walks in a fingerprint database.

Second, we propose an incremental algorithm, which updates the performed walks in fixed time interval, while the structure of underlying graph changes.

Third, we propose an approximation algorithm for computing Personalized PageRank vector based on the performed walks.

Finally, we do some experiments on real world graphs, and the experiments show that our proposed method is more efficient than related researches in estimating Personalized PageRank on dynamic networks.

### 2. Related Work

**Matrix Computation.** Matrix computation, which applies power iteration of linear algebra to compute the PageRank vector [1], is the most popular method for PageRank computation. In order to accelerate the convergence of power iteration, Borgs et al. [14] propose a sublinear time algorithm for PageRank computations. Because the graphs in real applications are usually sparse, del Corso et al. [15] and Fujiwara et al. [16] all apply sparse matrices to speed up the PageRank computation. Though these algorithms can accelerate the Personalized PageRank computation, they cannot be used in dynamic networks.

**Monte Carlo Method.** The Monte Carlo approximation methods [12,13] are efficient for estimating Personalized PageRank on large-scale graphs. The basic idea is that starting from the source node, do a number $R$ of random walks (called fingerprints); at each step, stop the random walk with a probability $\varepsilon$, and follow a random outgoing edge from the current node with the probability $1 - \varepsilon$. The lengths of the $R$ random walks from each node are subject to the geometrical distribution $\text{Geom}(\varepsilon)$. Then, use the frequencies of visit to different nodes in the performed walks to approximate the Personalized PageRank vector. The lengths of the performed walks conform to geometrical distribution, and thus there should be some long walks, so Sarma et al. [17] propose to do all random walks with length $1/\varepsilon$. In addition, Bahmani et al. [18] study the Personalized PageRank problem on MapReduce [19], and propose an optimal algorithm for constructing single random walk on MapReduce. In dynamic networks, Bahmani et al. [20] propose to store the performed walks in database and update them when an edge joins or leaves the underlying network. They consider that there is only one joining or leaving edge at a time, and update the fingerprints whenever the underlying network changes. However, in this paper, we study the same problem as [20], but we assume that there should be multichanges (multijoining and leaving edges) in a network.

**Personalized PageRank Application.** Personalized PageRank and other random walk with restart-based measures are the most well-known graph computation problems, and they have proved to be very effective in a variety of applications, such as link prediction [5, 21] and friend recommendation [22, 23], Webpage retrieval [1–3, 24–27], and trust-based recommendation systems [28–31] in large-scale networks. A similar work is [32], which uses dynamic GPS data to recommend an individual with interesting locations or travel sequences matching her travel preferences. But, the difference is that although they use the dynamic dataset, they do not mention how they handle this dynamic dataset incrementally. However, in this paper, we focus on the incremental update of recommendations, and the results can be used in any Personalized PageRank-based applications.

### 3. Monte Carlo-Based Incremental Personalized PageRank Computation

In this section, we describe our proposed system for incremental Personalized PageRank computation and illustrate the system in detail.

#### 3.1. System Overall Design

Our system answers users’ Personalized PageRank queries or recommend Personalized PageRank results to the specific users, and the architecture of the system is in Figure 1. First, we initialize the fingerprints and save them in a fingerprint database, and then, in a fixed time interval, we update the fingerprint database. These two
phases are done automatically by the system. When a query is issued by a personalized user, the searching engine retrieves related fingerprints from the fingerprint database, estimates the Personalized PageRank vector, and then returns nodes with highest score to the user. Some of the main phases of the system work as follows.

(i) Initializing Fingerprint Database. Given a graph $G = (V, E)$, we first do a random walk (fingerprint) with fixed length $\lambda$ starting from each node in the graph, and save all the performed walks in the fingerprint database.

(ii) Updating Fingerprint Database. The structure of the graph changes frequently, and it is impossible to update the fingerprint database with every change, so we update the fingerprint database with a fixed time interval $t$.

(iii) Computing Personalized PageRank. When a query is issued from a user, the searching engine generates SQL queries according to the specified user, retrieves related results from the fingerprint database, computes the Personalized PageRank vector, and returns a top-$k$ list to the user.

3.2. Initializing Fingerprint Database. Given a graph $G = (V, E)$, for each node $u \in V$, do a random walk $w_u$ of length $\lambda$ starting from the node $u$, and the details are given in Algorithm 1.

After doing the performed random walks, we save the performed walks in the fingerprint database.

3.3. Updating Fingerprint Database. In dynamic networks, Bahmani et al. [20] propose a model to update the fingerprint database whenever there is a joining edge or leaving edge. However, the structure of the underlying graph changes frequently, so it is impossible to update the fingerprint database for every change. In this paper, we propose an update algorithm that handles multichanges while updating the fingerprint database.

In a time interval $t$, there will be $k$ changes in the structure of the underlying network. These $k$ changes include joining edges and leaving edges in any order. We can apply the algorithm proposed in [20] $k$ times, but this will consume lots of computation resources. In this paper, we propose an update algorithm, which could handle multichanges at a time, and the details of the algorithm are in Algorithm 2.

Given a series of changes $k = \{+,-,-,+,-,\ldots,+\}$, where $+$ means a joining edge, and $-$ means a leaving edge we can reorder them into $k' = \{\ldots,-,-,+,\ldots,+\}$, where $k = k_1 + k_2$.

For the $k_1$ joining edges, we search in the fingerprint database for the fingerprints which contain one of these leaving edges, and the results are presented with $L$. For each $l \in L$, if it contains one leaving edge $e$ from its beginning, then we delete $e$ and its following edges but keep the start node of $e$.

For the $k_2$ joining edges, we collect all start nodes in $S$, search in the fingerprint database for the fingerprints which contain any node $s \in S$, and the results are presented with $J$. For each $j \in J$, if it contains one node $s$ from its beginning, then, with a probability $1/(d+1)$, we delete the following nodes of $s$.

Finally, for both $L$ and $J$, the lengths of fingerprints are smaller than $\lambda$, so we go on walking randomly on $G' = G - k_1 \cup k_2$ until their lengths reach $\lambda$.

---

Algorithm 1: Initialization ($G, \lambda$).
3.4. Computing Personalized PageRank. In this subsection, we propose an approximation method for estimating Personalized PageRank vector based on the performed walks, and the details are in Algorithm 3.

For each fingerprint $w \in W$, we find the subfingerprints which start from the personalized node $u$, and have the length of $1/\varepsilon$. In order to make these subfingerprints independent of each other, we must make sure that these subfingerprints do not overlap. After finding out those subfingerprints, we can estimate the Personalized PageRank vector with visit frequencies of the nodes.

4. Experiments

In this section, we do some experiments to validate the efficiency of our proposed method. We use the method proposed by the Bahmani et al. [20] as baseline algorithm, and compare it with our method in execution time.

| Dataset  | $n$    | $m$    | $n/m$ |
|----------|--------|--------|-------|
| Dianping | 204074 | 926720 | 4.5   |
| Gowalla  | 196591 | 1900654| 9.7   |

4.1. Dataset. To validate the performance of both algorithms, we use two real-world datasets of networks, Dianping [33] and Gowalla [34]. The details of the datasets are in Table 1.

4.2. Experimental Setup. Our experimental environment is a personal computer with a 4-core processor and 4GB memory. All algorithms are implemented by Java on the Ubuntu operation system. Our toolkit for dealing with graph is Jung [35], and the database for storing fingerprints is MySQL.
Input: $W$, the set of fingerprints, $u$, a personalized node for Personalized PageRank, and $\varepsilon$, the teleport probability.

Output: $\pi_u$, the Personalized PageRank vector for $u$.

1. Let $result = \{}$, $\pi_u(v) = 0$ for $1 \leq v \leq n$;
2. For each $w \in W$ do
   3. Scan $w$, find all sub-fingerprints of length $1/\varepsilon$ starting from $u$, and add these sub-fingerprints to $result$;
   4. End for;
3. For each $w \in result$ do
   4. For each node $v \in w$ do
      5. $\pi_u(v) = \pi_u(v) + 1$;
   6. End for;
   7. End for;
4. For $v = 1$ to $n$ do
   8. $\pi_u(v) = \frac{\pi_u(v)}{\sum_{i=1}^{n} \pi_u(v)}$;
   9. End for;
10. Return $\pi_u$;

**Algorithm 3: Computing PPR ($W, u, \varepsilon$).**

### 4.3. Experimental Results

For each graph $G$, we choose 90% of the dataset as already known graph $A$, and the other 10% as $B$. When an edge joins $A$, we randomly choose an edge $e$ from $B$, delete it from $B$, and add it to $A$. When an edge leaves $A$, we randomly choose an edge $e$ from $A$, delete it from $A$, and add it to $B$. The probabilities of joining and leaving are equal. We build an extra thread that changes the structure of the underlying graph once per ten seconds.

First, we compare the update time between our method with the method proposed by Bahmani et al. We update the fingerprint database in a fixed time interval from 1 to 10 second, and the results for two datasets are in Figures 2 and 3. From these two figures we can see that our update algorithm executes only once, but the Bahmani’s algorithm need to update the fingerprint database multitudes that is proportional to the time interval. Moreover, we find that the time used in updating the fingerprint database contains a nearly constant time retrieving the related fingerprints.

Second, we compare the execution time of Personalized PageRank approximation between the proposed method and Bahmani’s method. Both of the algorithms retrieve fingerprints from the fingerprint database first and then estimate the Personalized PageRank vector with the fingerprints. The Bahmani’s algorithm retrieves related fingerprints starting from the personalized node, and there are millions of fingerprints, so the searching process takes a lot of time. Our algorithm returns all fingerprints, and it needs only one pass of scanning the whole fingerprints to compute the Personalized PageRank vector. We randomly choose 10 personalized nodes and compare the average approximation time of Personalized PageRank, and the details are in Figure 4.

Third, we compare the overall performance between the two methods. The update time and Personalized PageRank approximation time of our method are all shorter than the Bahmani’s method, and thus it is obvious that the overall time used by our method should be shorter. Here, using the multithreading technology of Java, we compare the throughput of the two methods, and the results are in Figure 5. In this experiment, one thread is used to generate changes of the structure of underlying graph, one thread is used to update fingerprints, and other threads are generated dynamically to compute the Personalized PageRank vector. We use the number of finished Personalized PageRank vectors per minute as the throughput. From the figure we can see that our proposed method is more efficient than the Bahmani’s method in throughput.

Finally, we compare the accuracy of the two algorithms. To measure the quality of the results, for a node $u$ and a value $k$, we define $Z^k_u$ to be the set of nodes with the $k$ largest Personalized PageRank scores for $u$. In most Personalized PageRank-based applications, only nodes in $Z^k_u$
are of interest. For example, in recommending top-$k$ friends in online social networks or returning top-$k$ documents for a query, Personalized PageRank usually acts as one of the most important features. In this experiment, we assume Personalized PageRank as standard and compare how much approximation methods approximate it by the following equation:

$$\text{Err}(u, k) = \frac{\sum_{v \in Z_k^u} |\pi_u(v) - \tilde{\pi}_u(v)|}{\sum_{v \in Z_k^u} \pi_u(v)},$$

(4)

where $\tilde{\pi}_u(v)$ is the approximate Personalized PageRank by either Bahmani's or our algorithm. For the ground truth of $\pi_u(v)$, we use 100 iterations of power iteration. For testing the efficiency of the algorithms for different nodes, we divide nodes in a graph into 4 buckets, where bucket $i$ ($1 \leq i \leq 4$) contains nodes with degree in $[10^{i-1}, 10^i)$. There is no node with degree larger than 10000 in our datasets, so bucket 4 equals to $[1000, +\infty)$. We randomly choose 10 nodes from each bucket and average the error defined by (4).

We return top 10 and 100 results for each node and observe how approximation error changes along with the degree of nodes. Figures 6 and 7 illustrate the top-10 recommender results on both datasets, and Figures 8 and 9 illustrate the top-100 recommender results on both datasets. From these four figures we can see that, for both top 10 and top 100, our algorithm is more accurate in approximating Personalized PageRank than Bahmani’s algorithm.
5. Conclusion

Personalized PageRank is very important for complex networks, and it has been proved to be effective in many real applications, such as link prediction, entity recommendation, and social searching. However, in these applications, the underlying networks are usually dynamic, and traditional power iteration method cannot deal with them efficiently.

In this paper, we design a Monte Carlo-based incremental method for Personalized PageRank approximation in dynamic networks. We propose an efficient update algorithm to handle changes of the structure of the underlying network, and an efficient Personalized PageRank approximation algorithm with the performed walks. Experiments on real-world networks show that our method can handle multichanges of the underlying network structure at one time and is more efficient than related work, so can be used in real incremental Personalized PageRank-based applications.

Acknowledgments

This work was supported in part by the National Significant Science and Technology Special Project of China and the National Natural Science Foundation of China.

References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: bringing order to the web,” Tech. Rep., Stanford InfoLab, 1999.
[2] T. H. Haveliwala, “Topic-sensitive PageRank,” in Proceedings of the 11th International Conference on World Wide Web (WWW ’02), pp. 517–526, May 2002.
[3] T. Haveliwala, S. Kamvar, and G. Jeh, “An analytical comparison of approaches to personalizing PageRank,” Tech. Rep., Stanford, 2003.
[4] W. Liu and L. Lü, “Link prediction based on local random walk,” Europhysics Letters, vol. 89, no. 5, Article ID 58007, 2010.
[5] D. Liben-Nowell and J. Kleinberg, “The link-prediction problem for social networks,” Journal of the American Society for Information Science and Technology, vol. 58, no. 7, pp. 1019–1031, 2007.
[6] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu, “Scalable proximity estimation and link prediction in online social networks,” in Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, pp. 322–335, 2009.
[7] F. Jiang and Z. Wang, “PageRank-based collaborative filtering recommendation,” in Information Computing and Applications, pp. 597–604, Springer, 2010.
[8] J. H. Su, B. W. Wang, and V. S. Tseng, “Effective ranking and recommendation on web page retrieval by integrating association mining and PageRank,” in Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence.
