Knowl Inf Syst
DOI 10.1007/s10115-013-0691-1

REGULAR PAPER

Fast PageRank approximation by adaptive sampling

Wenting Liu · Guangxia Li · James Cheng

Received: 18 January 2012 / Revised: 5 September 2013 / Accepted: 14 September 2013

Abstract PageRank is typically computed from the power of transition matrix in a Markov Chain model. It is therefore computationally expensive, and efficient approximation methods to accelerate the computation are necessary, especially when it comes to large graphs. In this paper, we propose two sampling algorithms for PageRank efficient approximation: Direct sampling and Adaptive sampling. Both methods sample the transition matrix and use the sample in PageRank computation. Direct sampling method samples the transition matrix once and uses the sample directly in PageRank computation, whereas adaptive sampling method samples the transition matrix multiple times with an adaptive sample rate which is adjusted iteratively as the computing procedure proceeds. This adaptive sample rate is designed for a good trade-off between accuracy and efficiency for PageRank approximation. We provide detailed theoretical analysis on the error bounds of both methods. We also compare them with several state-of-the-art PageRank approximation methods, including power extrapolation and inner–outer power iteration algorithm. Experimental results on several real-world datasets show that our methods can achieve significantly higher efficiency while attaining comparable accuracy than state-of-the-art methods.

Keywords PageRank · Adaptive Sampling · Power iteration

1 Introduction

PageRank, an important method for ranking nodes in a graph, has been extensively studied for years. It is also a link analysis method as it relies on information encoded in the edges

W. Liu (✉) · G. Li
School of Computer Engineering, Nanyang Technological University,
Singapore, Singapore
e-mail: wliu7@e.ntu.edu.sg

J. Cheng
Department of Computer Science and Engineering,
Chinese University of Hong Kong, Shatin, Hong Kong

of graph [5]. The computation of PageRank combines the link information of all nodes in the graph through the calculation of large length random walks in an attempt to determine a stationary target distribution of a Markov chain. The random walk mirrors the behavior of individual surfers, which can be represented by a transition matrix of a Markov chain model. PageRank is essentially the stationary distribution vector of a Markov chain whose transition matrix is a convex combination of the Web link graph and a certain rank-one matrix.

Surveys on PageRank computation can be found in Berkhin [3] and Langville and Meyer [16]. Existing methods are space- and time-consuming when applied to very large graphs. It is therefore imperative to seek efficient methods to accelerate the computation. As a highly efficient and a widely used technique, sampling can make the computation tractable for large-scale data which otherwise could not be processed by ordinary means. Since PageRank is computed from large length of random walks, we can estimate it iteratively after each random walk step by sampling. In this paper, we discuss the use of the non-uniform sampling method for low-rank matrix approximation during PageRank computation in large graphs. Our method estimates the state distribution after some steps of random walks by sampling the power of the transition matrix. During the computation process, we adjust the sampling rate adaptively to keep a good trade-off between accuracy and efficiency.

We propose two sampling algorithms for PageRank approximation: *Direct Sampling* and *Adaptive Sampling*. Both methods sample the transition matrix and use the sample in PageRank computation. Direct sampling samples the transition matrix once and uses the fixed sampled transition matrix in PageRank computation. Adaptive sampling samples the transition matrix multiple times with an adaptive sample rate which is adjusted iteratively as the computing procedure proceeds. Based on the theoretical analysis of error bounds and computational complexity, we derive a way to adjust the adaptive sample rate to make a good trade-off between accuracy and efficiency for PageRank approximation. Experimental results on several real-world datasets verify that the proposed algorithms can achieve significantly higher efficiency while attaining comparable accuracy compared with state-of-the-art PageRank approximation methods.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 proposes direct sampling and adaptive sampling for PageRank approximation. Section 4 gives the theoretical error analysis of two sampling methods and derives an adaptive sampling rate choosing scheme. Section 5 reports the experimental results. Section 6 concludes this paper.

2 Related work

2.1 PageRank computation methods

PageRank is defined as the limiting stationary distribution of the large length random walks in a Markov chain model. According to Langville and Meyer [16], for a graph with n nodes, we can represent the random walk by a $n \times n$ matrix A, whose element A_{ij} is the probability of moving from state i (page i) to state j (page j) in one step of random walk. Let d_i be the out-degree of page i. From page i, suppose a surfer has the probability c to choose any hyperlink pages (i.e., neighbors), and with probability $1 - c$ to choose any page randomly, where c is a damping factor (chosen as 0.85 in Google PageRank). It turns out that $A_{ij} = c/d_i$ if j is the neighbor of i; otherwise, $A_{ij} = (1 - c)/n$. The matrix A serves as the transition matrix of the Markov chain model, and PageRank is computed from the limiting stationary distribution of the transition matrix power, as follows

$$\pi = \lim_{k \to \infty} A^k v \tag{1}$$

where v is a personalized column vector that represents the initial importance of each page determined by the user. A common practice is to set v as the uniform vector, i.e., $v = \left[\frac{1}{n}\right]_{n\times 1}$. Other more general choices of v have been described in Brin et al. [6].

In practice, estimating the PageRank values involves the estimation of the transition matrix after a large number of random walk steps. For storage efficiency, we introduce a sparse hyperlink matrix P such that $P_{ij} = 1/d_i$ if j is the neighbor of i; otherwise, $P_{ij} = 0$. The transition matrix A is then given by $A = cP + (1-c)\left[\frac{1}{n}\right]_{n\times n}$. For a more general v, we can rewrite the transition matrix as $A = cP + (1-c)ve$, where e is $[1]_{1\times n}$. Since PageRank π is the limiting stationary distribution of the transition matrix power, thus, $A\pi = \pi$, that is, $(I - cP)\pi = (1-c)ve\pi$, where I is identity matrix. As π is a normalized vector with ith element representing the PageRank of page i, we have $e\pi = 1$, and then deduce that $\pi = (1-c)(I-cP)^{-1}v = (1-c)\left(\sum_{k=1}^{+\infty} c^k P^k\right) v$, which is the matrix inverse algorithm described in Langville and Meyer [16].

According to the definition in Eq. 1, PageRank can be computed iteratively by power iteration method. It starts by initializing a vector x with the personalized vector v and updates vector x iteratively by multiplying the matrix A and normalizing the result, i.e., $x \leftarrow \frac{Ax}{\|Ax\|}$. When convergence is reached, the vector x is considered as the PageRank result π. Algorithm 1 summarizes the power iteration method.

Algorithm 1: Power Iteration (PI)

```
Input: The $n \times n$ matrix $P$; initial vector $v$; error tolerance $\epsilon$
Output: Convergence eigenvector $x$
$x \leftarrow v$; $k \leftarrow 1$;
repeat
    $y \leftarrow cPx + (1-c)v$ (that is, $y = Ax = cPx + (1-c)v$);
    Normalize $y$;
    $x \leftarrow y$;
    $k \leftarrow k + 1$;
until $\|y - x\|_1 < \epsilon$;
Convergence iteration: $K = k$
```

The classical power iteration method and matrix inverse algorithm yield the exact PageRank solution, but suffer from the computation inefficiency problem when it comes to large-scale data. Existing efficient approximation methods speed up PageRank computation by two ways: some accelerate the convergence process of power iteration [10,11,13,14,18,20,23]; others adopt the matrix decomposition technique to reduce the matrix dimension in matrix inverse algorithms [2,12,17,19,22,24,25].

2.1.1 Accelerating power iteration

Power Iteration method suffers from the slow convergence speed, especially for large-scale data. There are many accelerating methods designed to speed up the convergence procedure of power method. Iterative methods based on the Arnoldi process have been proposed to accelerate the convergence of power method for PageRank computation [23]. The vector extrapolation method [14,20] periodically subtracts the estimates of non-principal eigenvectors in the current iteration. Based on the observation that large PageRank values converge more slowly than small ones, one can adaptively "lock" the converged elements of PageRank vector to exclude them from subsequent computations [13]. Haveliwala's efficient power method—power extrapolation [11] can be combined with these accelerating techniques. It is considered as an effective method that is applicable to large graphs. Algorithm 2 outlines power extrapolation with power iteration (PEPI) method.

Algorithm 2: Power Extrapolation with Power Iteration (PEPI)

```
Input: The $n \times n$ matrix $P$; initial vector $v$; error tolerance $\epsilon$; $d$
Output: Convergence eigenvector $x^{(K)}$
$x^{(0)} \leftarrow v$; $k \leftarrow 1$;
repeat
    $x^{(k)} \leftarrow cPx^{(k-1)} + (1-c)v$;
    Normalize $x^{(k)}$;
    if $k == d+2$ then Do power extrapolation: $x^{(k)} \leftarrow (x^{(k)} - c^d x^{(k-d)})(1-c^d)^{-1}$;
    $k \leftarrow k+1$;
until $\|x^{(k)} - x^{(k-1)}\|_1 < \epsilon$;
Convergence iteration: $K = k$
```

Inner–outer power iteration (IOPI) [10] introduces the inner–outer scheme: Once the inner iterations start converging quickly, switch back to power method. It is considered to be one of the state-of-the-art accelerating methods. Algorithm 3 summarizes IOPI method, where $power(\alpha y + (1-\alpha)v)$ denotes power iteration method with initial vector $\alpha y + (1-\alpha)v$.

2.1.2 Matrix decomposition techiniques

Decomposition techniques on the transition matrix are employed in iterative aggregation-disaggregation method [25]. The specific graph structure, such as block structure, hierarchical structure approach [24], or scale-free networks [2], have also been used to save the computation cost. These matrix decomposition techiniques can be easily adapted to parallel computation [12,17,22].

The accelerating power iteration methods and matrix decomposition techiniques can be also combined together to improve the computational efficiency. For example, Osborne and Wiggins [19] combined several existing techniques, which includes the power method, linear systems, iterative aggregation/disaggregation, and matrix re-orderings, to accelerate convergence.

Algorithm 3: Inner–Outer Power Iteration (IOPI)

```
][!b] Input: The $n \times n$ matrix $A$; initial vector $v$; error tolerance $\epsilon$, $\eta$; $\alpha$; $\beta$; $i_m = 1$
Output: Convergence eigenvector $x$
$x \leftarrow v$;
$y \leftarrow cPx + (1-c)v$ ;
while $\|\alpha y + (1-\alpha)v - x\|_1 < \epsilon$ do
    $f \leftarrow (\alpha - \beta)y + (1-\alpha)v$;
    $k = 1$;
    repeat
        $x \leftarrow f + \beta y$;
        $y \leftarrow cPx + (1-c)v$ ;
        $k \leftarrow k+1$;
    until $\|f + \beta y - x\|_1 < \eta$;
    Convergence iteration: $i = k$;
    if $i \leq i_m$ then $x = power(\alpha y + (1-\alpha)v)$; return;
end
$x \leftarrow \alpha y + (1-\alpha)v$;
```

3 Sampling methods for PageRank

In this section, we present two sampling methods for PageRank estimation. The proposed methods are based on the low-rank approximation of transition matrix. The feasibility of using low-rank matrix approximation for PageRank computation has been experimentally demonstrated in Benczur et al. [4]. Different from prior work, we adopt the element-wise sampling instead of vector-wise sampling for low-rank approximation of matrix. We also theoretically prove that the errors of PageRank approximation using this sampling method are up-bounded.

3.1 Low-rank approximation of matrix

Methods for low-rank approximation of matrix can be grouped into two categories [9,8]: vector-based fast Monte Carlo approach and element-wise sampling approach. Given a $n \times n$ matrix A, to obtain its sampled matrix $\tilde{A}$, the vector-based approach selects c row or column vectors from A according to certain probability and scales them appropriately. The sampled matrix $\tilde{A}$ resulting from this approach approximates the original matrix A with an up-bounded error [9,8]. The storage space is also reduced as the storage space of sampled matrix is c/n of the original one.

Compared with vector-based approach, the element-wise method designs selection probability with more flexibility. Specifically, it samples matrix A according to the following procedure

1. Design appropriate $\{p_{ij}\}$ to make $\sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} = 1$;
2. Sample $\tilde{A}_{ij}$ with probability p_{ij} independently, and set $\tilde{A}_{ij} = A_{ij}/p_{ij}$.

The element-wise sampling method can be implemented in a single pass of the matrix data [9,1]. It can be easily adapted to parallel computation and is more suitable for large-scale matrix.

We therefore adopt the element-wise sampling method to derive the sampled matrix $\tilde{A}$ from the original matrix A. We sample $\tilde{A}_{ij}$ with the probability p_{ij} and scale it as $\tilde{A}_{ij} = A_{ij}/p_{ij}$. The expectation of $\widetilde{A}_{ij}$ is

$$E(\widetilde{A}_{ij}) = \widetilde{A}_{ij} * p_{ij} = A_{ij} \tag{2}$$

It is desired that the sampled elements can preserve the original element distribution of A, i.e., the edge distribution of the input graph. Therefore, a non-uniform sampling instead of an uniform one is required. Denote by s the sample size, i.e., the number of nonzero elements of $\tilde{A}$, an appropriate p_{ij} can be designed as follows

$$p_{ij} = \begin{cases} \min\left\{1, \frac{sA_{ij}^2}{\|A\|_F^2}\right\} & \text{if } A_{ij} > \frac{\theta\|A\|_F}{\sqrt{s}} \\ \min\left\{1, \frac{\sqrt{s}A_{ij}}{\theta\|A\|_F}\right\} & \text{otherwise} \end{cases} \tag{3}$$

where θ is a user-defined parameter.

With the choice of p_{ij} defined above, the sampled matrix can be derived as

$$\tilde{A}_{ij} = \begin{cases} \max\left\{A_{ij}, \frac{\|A\|_F^2}{sA_{ij}}\right\} & \text{if } A_{ij} > \frac{\theta\|A\|_F}{\sqrt{s}} \\ \frac{\theta\|A\|_F}{\sqrt{s}} & \text{if } A_{ij} \le \frac{\theta\|A\|_F}{\sqrt{s}}, \text{ with probability } \frac{\sqrt{s}}{\theta\|A\|_F}A_{ij} \\ 0 & \text{if } A_{ij} \le \frac{\theta\|A\|_F}{\sqrt{s}}, \text{ with probability } 1 - \frac{\sqrt{s}}{\theta\|A\|_F}A_{ij} \end{cases} \tag{4}$$

It is obvious that $\theta\|A\|_F/\sqrt{s} < 1$ as $\theta\|A\|_F/\sqrt{s} < A_{ij}$ in some cases, and A_{ij} is always less than one. Thus, the corresponding parameter θ should be set as a small value, i.e., $\theta < 1$. For convenience, we use ϵ to denote the cutoff threshold, i.e., $\epsilon = \theta\|A\|_F/\sqrt{s}$. From Eq. (4), it can be seen that those important (high weight) elements (A_{ij} above ϵ) are sampled in an unbiased way, i.e., with high probability (w.h.p) without error. For those less important elements (A_{ij} below ϵ), some are shrunk to 0 to improve efficiency, others are assigned as ϵ as a compensation for accuracy.

Note that $|\widetilde{A}_{ij} - A_{ij}|$ is up-bounded by the small value ϵ. The upper bound is obtained only when $A_{ij} = 0$ and it is assigned as $\widetilde{A}_{ij} = \epsilon$. In fact, the upper bound occurs with very small probability. This means that the non-uniform sampling scheme can preserve the link structure of A with quite a small error w.h.p., especially for the important (high weight) elements. Since $A_{ij} = 1/\text{out-degree}(i)$, the out links of nodes with small out-degree are high likely be preserved during sampling. Therefore, a high proportion of web pages can be covered by using this non-uniform sampling scheme.

To analyze the error bound for estimating matrix A, we first note that previous studies [9,8] have shown that element-wise sampling does not yield a nice error bound for the Frobenius norm, but a nice error bound for the spectral norm (i.e., $\|\widetilde{A} - A\|_2$ is upper-bounded). We use the following theorem to derive the error bound for our sampling method. The proof can be found in the "Appendix."

Theorem 1 *Given a sparse $n \times n$ matrix A with N nonzero entries, let $\widetilde{A}$ be any random matrix whose entries are independent random variables such that $\forall i, j$, $E(\widetilde{A}_{ij}) = A_{ij}$, $Var(\widetilde{A}_{ij}) \leq \delta^2$, and $|\widetilde{A}_{ij} - A_{ij}| \leq \delta$, where δ is a positive constant. Then, $\|\widetilde{A} - A\|_2 \leq \eta\sqrt{N}\delta$ holds with high probability. The parameter η is a positive constant $\left(\eta \approx 2\sqrt{2n}/\sqrt{N}\right)$.*

Theorem 1 indicates that the error is bounded by $\eta\sqrt{N}\delta$. As η and δ are small constants, the error bound given by Theorem 1 is actually quite small in practice.

The following lemma will be used to derive the upper error bound for estimating the kth power of matrix A, i.e., A^k. The proof is shown in the "Appendix."

Lemma 2 *Given a sparse $n \times n$ matrix A with N nonzero entries, let $\widetilde{A}$ be the sampled matrix of A obtained from our sampling method, s be the number of nonzero entries of $\widetilde{A}$, and η be a small constant ($\eta \approx 2\sqrt{2n/N}$ as shown in Theorem 1). The upper bound of $\|\widetilde{A} - A\|_2$ is*

$$\|\widetilde{A} - A\|_2 \leq \eta\|A\|_F\sqrt{N/s}, \tag{5}$$

and the upper bound of $\|\widetilde{A}\|_2$ is

$$\|\widetilde{A}\|_2 \leq \|\widetilde{A}\|_F \leq \|A\|_F. \tag{6}$$

Both upper bounds hold w.h.p.

We implement the Non-Uniform Sampling scheme in a single pass of matrix data. It is outlined in Algorithm 4. The parameter θ is set to $(8\log(n))^2/\sqrt{n}$ as suggested in Achlioptas and McSherry [1]. An element A_{ij} is selected if and only if $sA_{ij}^2/r_{ij} \geq \|A\|_F^2$ and $sA_{ij}^2/\theta^2 r_{ij}^2 \geq \|A\|_F^2$, where r_{ij} is drawn uniformly and independently from [0, 1]. This is equivalent to $r_{ij} \leq \min\{sA_{ij}^2/\|A\|_F^2, \sqrt{s}A_{ij}/\theta\|A\|_F\}$, which is in fact the sampling probability p_{ij} of our Non-Uniform Sampling scheme.

```
Algorithm 4: Non_Uniform_Sampling($A, \alpha, \theta$)
  Input: Matrix $A$ with $N$ nonzero entries, $\alpha = \sqrt{N/s}$, $\theta$
  Output: $\tilde{A}$
  Initialize: An empty priority queue $Q$, Let $Z \leftarrow 0$, $s = N/\alpha^2$;
  for each $A_{ij}$ do
      set $Z \leftarrow Z + A_{ij}^2$;
      draw $r_{ij}$ uniformly and independently from [0, 1];
      insert $A_{ij}$ in $Q$ with key $k_{ij} = \max\{sA_{ij}^2/r_{ij}, sA_{ij}^2/\theta^2 r_{ij}^2\}$;
      remove from $Q$ all elements with key smaller than $Z$;
  end
  Set $\tilde{A}$ as the contents of $Q$;
```

3.2 Approximating PageRank by sampling methods

PageRank computation can be accelerated by sampling the transition matrix. In fact, vector-based sampling method has been applied to PageRank approximation [1]. As aforementioned, element-wise sampling is superior to vector-based sampling on several fronts. Since the proposed non-uniform sampling scheme is able to estimate a matrix while preserving its element's original distribution, we can use it to estimate the transition matrix during PageRank computation.

We thus propose two sampling methods to approximate PageRank based on the low-rank approximation of the transition matrix by the Non-Uniform Sampling scheme described in Algorithm 4. Specifically, to approximate PageRank $\pi = \lim_{k\to\infty} A^k v$, where $A = cP + (1-c)ve$, we use the low-rank approximation $\tilde{P}$ instead of P in the power iteration scheme (i.e., iterative matrix-vector multiplication).

The first method is the direct sampling in power iteration (DSPI). It samples matrix P once and uses the fixed sampled matrix $\tilde{P}$ during power iteration. The second method is the adaptive sampling in power iteration (ASPI). It adaptively adjusts the sampling rate to make a new sample of P in each iteration, and uses the adaptive sampled matrix in power iteration. As shown later, it is able to balance efficiency and accuracy of PageRank approximation in a better way. Algorithms 5 and 6 outline DSPI and ASPI, respectively.

```
Algorithm 5: Direct Sampling in Power Iteration (DSPI)
  Input: Given $n \times n$ matrix $P$; initial vector $v$; error tolerance $\epsilon$
  Output: Convergence eigenvector $x$
  $x \leftarrow v$;
  $\tilde{P}$ =Non_Uniform_Sampling($P, \alpha, \theta$);
  for $k = 1, \ldots$ Repeat do
      $y \leftarrow c\tilde{P}x + (1-c)v$ (approximating $y = Ax = cPx + (1-c)v$);
      Normalize $y$;
      Until $\|y - x\|_1 < \epsilon$, convergence iterations: $K = k$;
      $x \leftarrow y$;
  end
```

4 Theoretical analysis of sampling methods for PageRank

In this section, we give theoretical analysis on the PageRank approximation error of the two proposed sampling methods. Recall that the exact PageRank solution is derived by

Algorithm 6: Adaptive Sampling in Power Iteration (ASPI)

```
Input: Given $n \times n$ matrix $P$; initial vector $v$; error tolerance $\epsilon$; appropriate $\{\alpha_k\}_{k=1}^K$; small constant $\theta$
Output: Convergence normalized eigenvector $x$
Initialization:$x \leftarrow v$;
for $k = 1, \dots$ Repeat do
    $\tilde{P}_k$ =Non_Uniform_Sampling($P, \alpha_k, \theta$);
    $y \leftarrow c\tilde{P}_k x + (1-c)v$;
    Normalize $y$;
    Until $\|y - x\|_1 < \epsilon$, convergence iterations: $K = k$;
    $x \leftarrow y$;
end
```

$$\pi = (1-c)(I - cP)^{-1}v = (1-c)\left(\sum_{k=1}^{+\infty} c^k P^k\right)v,$$

and the approximated PageRank is derived by

$$\tilde{\pi} = (1-c)\left(\sum_{k=1}^{+\infty} c^k \widetilde{P^k}\right)v.$$

The PageRank approximation error is thus given by

$$\tilde{\pi} - \pi = (1-c)\left[\sum_{k=1}^{+\infty} c^k \left(\widetilde{P^k} - P^k\right)\right]v.$$

If the convergence iterations are comparable, the PageRank approximation error is determined by the difference between P^k and $\widetilde{P^k}$, that is, the matrix power approximation error from estimating P^k as $\widetilde{P^k}$. If the matrix power approximation error grows slower than c^{-k}, the PageRank approximation error can be bounded.

Therefore, to derive the error bound of using the proposed DSPI and ASPI method for PageRank approximation, we first need to analyze the error raised by approximating P^k. We thus derive two corresponding sampling methods for matrix power approximation: Direct sampling of P^k as shown in Algorithm 7, and adaptive sampling of P^k as shown in Algorithm 8.

4.1 Direct sampling for matrix power

In the DSPI method, the approximated PageRank is in fact $\tilde{\pi} = (1-c)\left(\sum_{k=1}^{+\infty} c^k \widetilde{P^k}\right)v$, where $\widetilde{P^k} = \tilde{P}^k$. The matrix power operation P^k plays an important role in the analysis of the PageRank estimation error by DSPI. Therefore, we first derive the corresponding direct sampling method to approximate matrix power P^k in Algorithm 7.

The error analysis on estimating P^k is shown in Theorem 3, with the proof shown in "Appendix."

Theorem 3 *Given a matrix P with N nonzero entries, let $\widetilde{P}$ be the sampled matrix of P using Algorithm 7 with parameter $\alpha = \sqrt{N/s}$, where s is the user-defined sample size (the number of nonzero entries of $\widetilde{P}$). The error bound for estimating P^k with $\widetilde{P}^k$ by direct sampling is*

$$\|\widetilde{P}^k - P^k\|_2 \leq k\eta\alpha\|P\|_F^k \tag{7}$$

where $\eta > 0$ is a small constant introduced in Lemma 2.

Algorithm 7: Direct Sampling of P^k

```
Input: The parameter $\alpha, \theta$ defined by user, and matrix $P$,
Output: $B_k$ as the estimation of $P^k$
Initialization: $B_0 \leftarrow I$, where $I$ is the identity matrix;
$\tilde{P}$ =Non_Uniform_Sampling($P, \alpha, \theta$);
for $i = 1 : k$ do
    $B_i \leftarrow B_{i-1}\tilde{P}$;
    Until convergence;
end
return the matrix $B_k$
```

Algorithm 8: Adaptive Sampling of P^k

```
Input: Design appropriate $\{\alpha_i\}_{i=1}^{k}$; small constant $\theta$ defined by user, and matrix $P$,
Output: $B_k$ as the estimation of $P^k$
Initialization: $B_0 \leftarrow I$, where $I$ is the identity matrix;
for $i = 1 : k$ do
    $\tilde{P}_i$ =Non_Uniform_Sampling($P, \alpha_i, \theta$);
    $B_i \leftarrow B_{i-1}\tilde{P}_i$;
    Until convergence;
end
return the matrix $B_k$
```

4.2 Adaptive sampling for matrix power

In the ASPI method, the approximated PageRank is obtained by iteratively approximating P^k via Algorithm 8. The error analysis on estimating P^k with the adaptive sampling is given by Theorem 4. The proof is shown in "Appendix."

Theorem 4 *Denote the estimation of P^k using Algorithm 8 by $B_k = \widetilde{P^k}$. Let B_i be the output matrix B at the ith iteration of Algorithm 8, that is, $B_i = B_{i-1}\widetilde{P_i}$, where $\widetilde{P_i}$ is the sampled matrix of P using α_i at the ith iteration. Then the total error of estimating P^k from adaptive sampling in Algorithm 8 is*

$$\|\widetilde{P^k} - P^k\|_2 \le \eta \|P\|_F^k \sum_{i=1}^{k} \alpha_i \tag{8}$$

If we set $\alpha_1 = a$ and adaptively choose $\alpha_k = a\alpha_{k-1}$. The total error bound of estimating A^k is

$$\|\widetilde{P^k} - P^k\|_2 \le \eta \|P\|_F^k \frac{a(1-a^k)}{1-a} \tag{9}$$

where a is an enhancing factor in Algorithm 8, and $\eta > 0$ is a small constant introduced in Lemma 2.

4.3 Adaptive chosen sample rate

Based on the above error analysis of matrix power approximation, this section gives an in-depth analysis about the accuracy and efficiency issues of sampling methods for approximating matrix power. By analyzing the computational complexity of the proposed direct sampling and adaptive sampling methods, we show how to choose the adaptive sampling rate to approximate PageRank efficiently and accurately.

From the derivation of error bound shown in "Appendix," we can see that the error of approximation matrix power P^k is accumulated iteratively. The error raised in former iterations influences the error of later iterations. By accumulating error bounds of each iteration iteratively, we obtain the upper error bound, which reflects the total error of approximating matrix power. We thus can analyze the error of matrix power approximation by the analysis of its upper error bound.

Since PageRank is computed by accumulating the normalized matrix-vector product iteratively, the error of our PageRank approximation is in fact determined by the relative error of approximating matrix power, i.e., $\|\widetilde{P^k} - P^k\|_2/\|P^k\|_2$. As $\|P^k\|_2$ is comparable with $\|P\|_F^k$ [15,7], according to Eqs. (7) and (8), the relative errors of approximating matrix power by direct sampling (using an unchanged parameter α) and adaptive sampling (using an adaptive parameter α_i) are determined by $k\alpha$ and $\sum_{i=1}^{k} \alpha_i$, respectively. This indicates that the relative error of approximating P^k increases linearly with increasing α or α_i.

Regarding the computational complexity, the non-uniform sampling which is outlined in Algorithm 4 requires only one pass of the matrix data. For a matrix with N nonzero entries, its time complexity is $\mathcal{O}(N)$ and the storage requirement for the sampled elements is $\mathcal{O}(s)$. As outlined in Algorithm 7, the direct sampling for approximating P^k samples P only once by using Algorithm 4, and performs matrix multiplication k times. Thus, it requires $\mathcal{O}\left(N + s^k\right)$ time and $\mathcal{O}(s)$ additional space. In terms of the adaptive sampling for approximating P^k (Algorithm 8), the sampling operation of P is performed k times by using Algorithm 4. Since each sampling is followed by a matrix multiplication, there are totally k times matrix multiplications. Thus, it requires $\mathcal{O}\left(kN + \Pi_{i=1}^{k} s_i\right)$ time and $\mathcal{O}\left(\sum_{i=1}^{k} s_i\right)$ additional space for the sampled elements. It can be seen that both the time and space complexity of direct sampling and adaptive sampling are related to s and s_i, respectively. As $s = N/\alpha^2$ and $s_i = N/\alpha_i^2$, the complexity decreases quadratically with increasing α or α_i.

From the analysis of accuracy and efficiency given above, we see that changing the parameter α_i adaptively results in a better trade-off between accuracy and efficiency. Recall that the error increases linearly with increasing α, while the time and space complexity decreases quadratically with increasing α. It turns out that adaptive increasing α_i can help to improve efficiency without losing much accuracy. As shown in Algorithms 6 and 8, we can make α_i increase iteratively by updating α_i as $\alpha_i = a\alpha_{i-1}$ with an enhancing factor $a > 1$.

4.4 Error analysis of PageRank approximation

We now analyze the PageRank approximation error of the proposed DSPI and ASPI algorithms. Since the approximated PageRank is calculated as $\tilde{\pi} = (1-c)\left(\sum_{k=1}^{+\infty} c^k \widetilde{P^k}\right) v$, both DSPI and ASPI inherit the error from approximating matrix power iteratively. The error of PageRank approximation by using DSPI and ASPI is given by

$$\|\tilde{\pi} - \pi\|_2 \leq (1-c) \sum_{k=1}^{K} c^k \|\widetilde{P^k} - P^k\|_2 \|v\|_2 \tag{10}$$

The following theorem gives the error bound of PageRank approximation by DSPI and ASPI. Its proof can be found in "Appendix."

Theorem 5 *The total error of estimating π by the DSPI method is proportional to*

$$\sum_{k=1}^{K} c^k \|\tilde{P}^k - P^k\|_2 \leq \sum_{k=1}^{K} c^k \|P\|_F^k k\alpha = \alpha \sum_{k=1}^{K} c^k \|P\|_F^k + \alpha \sum_{k=2}^{K} c^k \|P\|_F^k + \cdots + \alpha c^K \|P\|_F^K.$$

With certain choice of α_i for $1 \leq i \leq K$, the total error of estimating π by the ASPI method is proportional to

$$\sum_{k=1}^{K} c^k \|\widetilde{P^k} - P^k\|_2 \leq \sum_{k=1}^{K} c^k \|P\|_F^k \left(\sum_{i=1}^{k} \alpha_i\right)$$
$$= \alpha_1 \sum_{k=1}^{K} c^k \|P\|_F^k + \alpha_2 \sum_{k=2}^{K} c^k \|P\|_F^k + \cdots + \alpha_K c^K \|P\|_F^K.$$

For convenience, we denote $L(i) = \sum_{k=i}^{K} c^k \|P\|_F^k$. It is obvious that $L(i), i = 1, \ldots, K$ is a decreasing numerical number series. The total error of estimating π by the ASPI method can be rewritten as $\sum_{i=1}^{K} \alpha_i L(i)$, and the total error of estimating π by the DSPI method is $\sum_{i=1}^{K} \alpha L(i)$.

In ASPI, as i increases, $L(i)$ decreases, and α_i becomes less influential on the error part $\alpha_i L(i)$. That is, accuracy is more sensitive in former steps than in later steps. Thus, setting α_i a small value in former steps helps to reduce much error. On the other hand, according to the analysis in the previous section, big α_i can improve computational efficiency. Hence, we can use bigger α_i in later steps to improve efficiency without losing too much accuracy. Here we see that ASPI is able to maintain better trade-off between accuracy and efficiency than DSPI by setting adaptive increasing α_i. This has also been demonstrated in our experiments: compared with DSPI, ASPI achieves higher accuracy with comparable efficiency on small-scale datasets. On large-scale datasets, ASPI is more efficient and as accurate as DSPI.

5 Experimental results

We compare the proposed DSPI and ASPI with some state-of-the-art methods for PageRank computation. Specifically, our benchmarks include: power iteration (PI), power extrapolation with power iteration (PEPI) [11], and inner–outer power iteration (IOPI) [10]. All algorithms are implemented with Matlab and run on a machine with an Intel Xeon x86-64 2.6 GHz CPU and 32 GB RAM.

To evaluate the efficiency of an algorithm, we record its running time, memory usage, and number of iterations till convergence. To evaluate its accuracy, we use the following three metrics:

1. The total relative error, calculated by $\|\tilde{\pi} - \pi\|_1 / \|\pi\|_1$,
2. The difference between PageRank approximation result and exact result, measured by the L_1 norm;
3. The ranking similarity between PageRank approximation result and exact result, using the Spearman's rank correlation of top-n ranking nodes as measurement. It is defined as

$$r = 1 - \frac{6 \sum_{i=1}^{n} \left(\tilde{R}_i - R_i\right)^2}{n^3 - n} \tag{11}$$

where $\tilde{R}_i$ and R_i are the ranking of $\tilde{\pi}$ and π, respectively.

Table 1 Details of datasets

	# Node	# Edge
Wiki-Vote	8,298	103,689
Soc-Epinions	9,137	264,298
Slashdot-0902	82,168	870,161
Web-Stanford	281,904	2,312,497
Web-Google	916,428	5,105,039
Web-BerkStan	685,231	7,600,595

5.1 Datasets and parameter setting

The six datasets used in our experiment are obtained from the Stanford Network Analysis Platform (SNAP [21]). Three of them are relatively small datasets: *Wiki-Vote*, *Soc-Epinions*, and *Slashdot-0902*. And the other three are larger datasets: *Web-Stanford*, *Web-Google*, and *Web-BerkStan*. Specifically, Wiki-Vote is a Wikipedia who-votes-on-whom network. Soc-Epinions is who-trusts-whom network crawled from epinions.com. Slashdot-0902 is the social network obtained from slashdot.org on February 2009. In these three networks, nodes represent people and edges represent the interactions between people. Web-Stanford is the Web graph of Stanford.edu. Web-Google is a Web graph derived from Google. Web-BerkStan is a Web graph of Berkeley and Stanford. In the web graphs, nodes are pages and edges are hyperlinks. We list the number of nodes and edges of the six datasets in Table 1.

We set the convergence tolerance τ as 10^{-5}, damping factor c as 0.85, and maximum iteration as 100 for all algorithms. For all datasets, we set adaptive sampling's parameters as $\alpha_1 = \sqrt{2}$, enhancing factor $a = \sqrt{2}$, and $\alpha_{i+1} = a\alpha_i$ in $\{\alpha_i\}_{i=1}^{K}$. Direct sampling's parameter is set as $\alpha = (1/K)\sum_{i=1}^{K} \alpha_i$. For PEPI method, we set $d = 2$. For IOPI method, we set $\eta = 10^{-4}$, $\beta = 0.5$.

5.2 Performance evaluation on small-scale and large-scale datasets

Table 2 reports the experimental results on the three small-scale datasets. It can be seen that the proposed sampling methods (DSPI and ASPI) match other methods in memory consumption, but win out in terms of iteration number and running time. A relative small value of iteration number and running time suggests that the proposed sampling methods (DSPI and ASPI) converge more quickly and thus are more efficient than baselines. Speak to accuracy, PEPI is the best as it has the lowest "PageRank difference" which is measured by the L_1 norm of the difference between the result of approximation method and the exact one (PI). As first and second runner-ups, the proposed sampling methods (DSPI and ASPI) achieve comparable accuracy with PEPI. By comparing ASPI with DSPI, we find that with similar convergence iterations and memory usage, ASPI is more accurate and efficient (in terms of running time) than DSPI. This verifies the effect of the proposed adaptive sampling scheme.

Table 3 reports the experimental results on the large-scale datasets. It shows that PEPI outperforms other methods in terms of "PageRank difference." Compared with IOPI, the proposed methods (DSPI and ASPI) have much smaller "PageRank difference" values. In terms of efficiency, DSPI and ASPI save a lot of memory, running time, and converge more quickly than other methods. By comparing DSPI and ASPI, we observe that both of them can estimate PageRank with comparable accuracy, but ASPI is more efficient than DSPI, as the former has lower running time.

Table 2 Experimental results on small-scale datasets

Method	# Iteration to converge	Time (s)	Memory (MB)	PageRank difference (%)
(a) Wiki-Vote ($\theta^2 = 10^{-5}$)				
PI	10	0.0449	494.43	0
PEPI	13	0.0516	494.67	0.06
IOPI	18	0.0629	495.84	11.92
DSPI	7	0.0379	494.26	0.74
ASPI	7	0.0378	494.69	0.22
(b) Soc-Epinions ($\theta^2 = 10^{-5}$)				
PI	11	0.0895	503.29	0
PEPI	13	0.1047	503.53	0.002
IOPI	15	0.0910	505.14	1.03
DSPI	5	0.0567	503.12	0.28
ASPI	5	0.0539	503.05	0.05
(c) Slashdot-0902 ($\theta^2 = 10^{-6}$)				
PI	15	0.4047	550.47	0
PEPI	15	0.4155	550.70	0.05
IOPI	22	0.5405	552.69	4.55
DSPI	7	0.2161	542.31	1.19
ASPI	7	0.2056	542.98	1.13

Table 3 Experimental Results on Large-scale Datasets

Method	# Iteration to converge	Time (s)	Memory (MB)	PageRank difference (%)
(a) Web-Stanford ($\theta^2 = 10^{-6}$)				
PI	22	2.0322	857.15	0
PEPI	22	2.0325	857.43	0.01
IOPI	23	1.7564	872.60	0.17
DSPI	11	1.0050	832.26	0.03
ASPI	11	0.9950	834.52	0.04
(b) Web-Google ($\theta^2 = 10^{-6}$)				
PI	59	122.9678	3,542.51	0
PEPI	51	84.9109	3,485.23	0.0007
IOPI	65	62.4066	3,649.90	12.97
DSPI	39	47.8129	3,127.25	0.07
ASPI	30	25.7321	3,037.25	0.22
(c) Web-BerkStan ($\theta^2 = 10^{-7}$)				
PI	33	43.8221	3,404.72	0
PEPI	26	26.8577	3,286.95	0.01
IOPI	37	57.7538	3,378.95	1.18
DSPI	20	13.4836	3,261.17	0.04
ASPI	20	11.8073	3,223.70	0.05

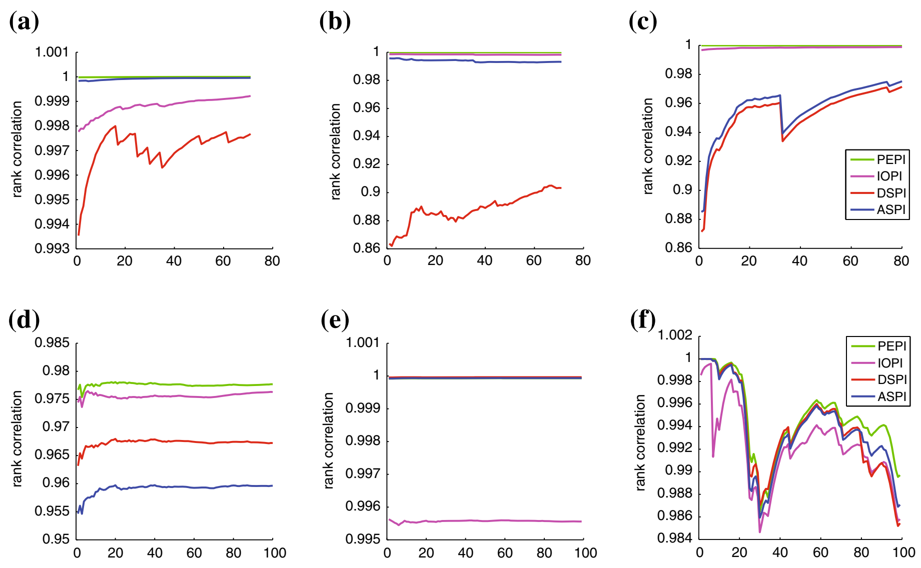


Fig. 1 Rank correlation of top-k nodes on all datasets. **a** Wiki-Vote (k:100-8000). **b** Soc-Epinions (k:100-8000). **c** Slashdot-0902 (k:1000-80000). **d** Web-Stanford (k:2000-20000). **e** Web-Google (k:10000-800000). **f** Web-BerkStan (k:10000-600000)

Figure 1 depicts the rank correlation of top-k nodes with the exact PageRank result (PI) as reference. From Fig. 1, we see that all methods can approximate PageRank with Spearman's rank correlation greater than 0.95 in most of the cases.

6 Conclusion

In this paper, we proposed two sampling algorithms, namely the direct sampling and the adaptive sampling, for estimating the power operation of a transition matrix, which serves as the essential part of PageRank computation. Integrating them with power iteration scheme gives two PageRank approximation algorithms: DSPI and ASPI. We gave an in-depth theoretical analysis on the error bound and computational complexity of the proposed algorithms. Based on the analysis, we derived a way to adjust the sampling rate adaptively to keep a good trade-off between accuracy and efficiency. Experimental results on six real-world datasets show that the proposed methods are able to achieve significantly higher computational efficiency and comparable accuracy compared with several state-of-the-art PageRank approximation methods. Since the proposed methods use the power iteration scheme, existing techniques that accelerate the convergence procedure of power iteration, such as power extrapolation [11] and inner–outer power iteration [10], can be easily incorporated to further improve the computational efficiency.

7 Appendix

The proof of Theorem 1 is as follows.

Proof From Theorem 3.1 in Achlioptas and McSherry [1], when $E(\widetilde{A}_{ij}) = A_{ij},\ Var(\widetilde{A}_{ij}) \leq \delta^2$, and $|\widetilde{A}_{ij} - A_{ij}| \leq \delta K$, where $K = \left(\frac{\log(1+\epsilon)}{\log(2n)}\right)^2 \times \sqrt{2n}$, for any fixed $\epsilon > 0$. When we choose $K > 1$, then for any $\omega > 0$ and $2n > 152$, $\|\widetilde{A} - A\|_2 \leq 2(1+\epsilon+\omega)\delta\sqrt{2n}$ holds w.h.p.

As $|\widetilde{A}_{ij} - A_{ij}| \leq \delta \leq \delta K$ (choosing $K > 1$), then $\|\widetilde{A} - A\|_2 \leq 2(1+\epsilon+\omega)\delta\sqrt{2n}$ holds w.h.p. for any $\omega > 0$ and $n > 76$.

Especially, for sparse transition matrix A, the number of nonzero entries $N = dn$, where d is the average degree and $d \leq 50$ for most sparse real datasets; thus, $\sqrt{2n}/\sqrt{N}$ is a constant. Let $\eta = 2(1+\epsilon+\omega)\sqrt{2n}/\sqrt{N}$, then $\|\widetilde{A} - A\|_2 \leq \eta\sqrt{N}\delta$ holds w.h.p. □

The proof of Lemma 2 is as follows.

Proof We can compute $E(\widetilde{A}^2_{ij})$ by Eq. 4 as follows

$$E(\widetilde{A}^2_{ij}) = \widetilde{A}^2_{ij} * p_{ij} = \frac{A^2_{ij}}{p_{ij}} \leq \begin{cases} \frac{\|A\|_F^2}{s} & \text{if } A_{ij} > \frac{\theta\|A\|_F}{\sqrt{s}} \\ A_{ij}\frac{\theta\|A\|_F}{\sqrt{s}} \leq \frac{\theta^2\|A\|_F^2}{s} < \frac{\|A\|_F^2}{s} & \text{otherwise} \end{cases}$$

It is obvious that

$$E\left(\widetilde{A}^2_{ij}\right) \leq \frac{\|A\|_F^2}{s} \tag{12}$$

The upper bound for the variance of $\widetilde{A}_{ij}$ is given as

$$\begin{aligned} Var(\widetilde{A}_{ij}) &= E\left[(\widetilde{A}_{ij} - A_{ij})^2\right] \\ &= E(\widetilde{A}^2_{ij}) - A^2_{ij} \leq E(\widetilde{A}^2_{ij}) \leq \frac{\|A\|_F^2}{s} \end{aligned} \tag{13}$$

Let $\delta = \|A\|_F/\sqrt{s}$. Then, $|\widetilde{A}_{ij} - A_{ij}| \leq \delta\theta < \delta$ and $Var(\widetilde{A}_{ij}) \leq \delta^2$. According to Eq. 2 and Theorem 1, we have

$$\|\widetilde{A} - A\|_2 \leq \sqrt{N}\delta\eta = \eta\|A\|_F\sqrt{N/s},$$

holds w.h.p., where η is a small constant.

We prove Eq. 6 as follows. According to Eq. 12, we can derive the upper bound of $\|\widetilde{A}\|_F$ as follows.

As $E\left(\|\widetilde{A}\|_F\right) \leq \sqrt{E\left(\|\widetilde{A}\|_F^2\right)} \leq \sqrt{\sum_{ij} E\left(\widetilde{A}^2_{ij}\right)} \leq \sqrt{s\frac{\|A\|_F^2}{s}} = \|A\|_F$. According to Chernoff bound, $Pr\left[\|\widetilde{A}\|_F \leq \|A\|_F\right] \geq 1 - \exp(-\Omega(\|A\|_F))$. Thus,

$$\|\widetilde{A}\|_F \leq \|A\|_F$$

holds w.h.p.

Since $\|\widetilde{A}\|_2 \leq \|\widetilde{A}\|_F$, we have

$$\|\widetilde{A}\|_2 \leq \|\widetilde{A}\|_F \leq \|A\|_F.$$

□

The proof of Theorem 3 is as follows.

Proof According to Lemma 2, we have $\|\widetilde{P} - P\|_2 \leq \eta\alpha\|P\|_F$, where $\alpha = \sqrt{\frac{N}{s}}$, $\eta > 0$ is a small constant; and $\|\widetilde{P}\|_2 \leq \|P\|_F$.

Let $R_k = \widetilde{P}^k - P^k$, where $k \in \{1, \ldots, K\}$, then,

$$\begin{aligned}\|R_k\|_2 = \|\widetilde{P}^k - P^k\|_2 &= \|\widetilde{P}\widetilde{P}^{k-1} - PP^{k-1}\|_2\\ &= \|\widetilde{P}(\widetilde{P}^{k-1} - P^{k-1} + P^{k-1}) - PP^{k-1}\|_2\\ &= \|\widetilde{P}R_{k-1} + \widetilde{P}P^{k-1} - PP^{k-1}\|_2\\ &\leq \|\widetilde{P}R_{k-1}\|_2 + \|(\widetilde{P} - P)P^{k-1}\|_2\\ &\leq \|\widetilde{P}\|_2\|R_{k-1}\|_2 + \|\widetilde{P} - P\|_2\|P\|_2^{k-1}\\ &\leq \|P\|_F\|\|R_{k-1}\|_2 + \eta\alpha\|P\|_F^k \end{aligned} \tag{14}$$

From the above steps, we can see that $\|R_k\|_2$ can be induced from $\|R_{k-1}\|_2$. Next, we induce

$$\|R_k\|_2 \leq k\eta\alpha\|P\|_F^k \tag{15}$$

via mathematical induction as follows.

Basis: Initially,

$$\|R_1\|_2 = \|\widetilde{P} - P\|_2 \leq \eta\alpha\|P\|_F. \tag{16}$$

When $k = 2$, according to Eqs. 14 and 16,

$$\|R_2\|_2 \leq \|P\|_F\|\eta\alpha\|P\|_F + \eta\alpha\|P\|_F^2 = 2\eta\alpha\|P\|_F^2.$$

Thus, Eq. 15 holds for the first step.

Inductive step: Assume that Eq. 15 holds for the $(k-1)$th step,

$$\|R_{k-1}\|_2 \leq (k-1)\eta\alpha\|P\|_F^{k-1}. \tag{17}$$

Then, we deduce the kth step according to Eqs. 14 and 17 as follows.

$$\|R_k\|_2 \leq \|P\|_F(k-1)\eta\alpha\|P\|_F^{k-1} + \eta\alpha\|P\|_F^k = k\eta\alpha\|P\|_F^k$$

that is, Eq. 15 also holds for the kth step.

Since both the basis and the inductive step have been performed, by mathematical induction, the statement Eq. 15 holds for all natural k. As such, we have

$$\|\widetilde{P}^k - P^k\|_2 \leq k\eta\alpha\|P\|_F^k, \tag{18}$$

where $\alpha = \sqrt{\frac{N}{s}}$ and $\eta > 0$ is a small constant. □

The proof of Theorem 4 is as follows.

Proof Let B_i be the input matrix B at the ith iteration of Algorithm 8, where $1 \leq i \leq k$. Then, $B_i = B_{i-1}\widetilde{P}_i$, where $\widetilde{P}_i$ is the sampled matrix of A with $\alpha_i = \sqrt{\frac{N}{s}}$, where N is the number nonzero entries of P, and s is the sample size of $\widetilde{P}_i$. According to Lemma 2, $\|\widetilde{P}_i - P\|_2 \leq \eta\alpha_i\|P\|_F$, we have

$$\begin{aligned}\|B_i - B_{i-1}P\|_2 &= \|B_{i-1}\widetilde{P}_i - B_{i-1}P\|_2\\ &\leq \|B_{i-1}\|_2\|\widetilde{P}_i - P\|_2\\ &\leq \|B_{i-1}\|_F\eta\alpha_i\|P\|_F. \end{aligned} \tag{19}$$

From Lemma 2, $\|\widetilde{P}_i\|_F \leq \|P\|_F$, thus,

$$\|B_i\|_F = \|B_{i-1}\widetilde{P}_i\|_F \leq \|B_{i-1}\|_F\|\widetilde{P}_i\|_F \leq \|B_{i-1}\|_F\|P\|_F. \tag{20}$$

Since $\|B_1\|_2 = \|P\|_2 \leq \|P\|_F$ and from Eq. 20, we have

$$\|B_i\|_F \leq \|P\|_F^i. \tag{21}$$

Since the estimation of P^k from Algorithm 8 is $\widetilde{P^k} = B_k = B_{k-1}\widetilde{P}_k$, the total error for estimating P^k in Algorithm 8 is given by $\|B_k - P^k\|_2$. Note that $B_0 = I$ is the identity matrix. According to Eqs. 19 and 21, we have

$$\begin{aligned}
\|\widetilde{P^k} - P^k\|_2 &= \|B_k - P^k\|_2 \\
&= \|(B_k - B_{k-1}P) + (B_{k-1}P - B_{k-2}P^2) + \cdots + (B_{k-i}P^i - B_{k-i-1}P^{i+1}) \\
&\quad + \cdots + (B_1P^{k-1} - P^k)\|_2 \\
&= \left\|\sum_{i=0}^{k-1}(B_{k-i} - B_{k-i-1}P)P^i\right\|_2 \\
&\leq \sum_{i=0}^{k-1}\|(B_{k-i} - B_{k-i-1}P)P^i\|_2 \\
&\leq \sum_{i=0}^{k-1}\|B_{k-i-1}\|_F\eta\alpha_{k-i}\|P\|_F\|P^i\|_2 \\
&\leq \sum_{i=0}^{k-1}\|P\|_F^{k-i-1}\eta\alpha_{k-i}\|P\|_F\|P\|_F^i \\
&= \eta\|P\|_F^k\sum_{i=0}^{k-1}\alpha_{k-i} = \eta\|P\|_F^k\sum_{i=1}^{k}\alpha_i
\end{aligned} \tag{22}$$

If we adaptively choose $\alpha_i = a\alpha_{i-1}$ and $\alpha_1 = a$, then $\alpha_i = a^i$, then we obtain the error bound as follows:

$$\|\widetilde{P^k} - P^k\|_2 \leq \eta\|P\|_F^k\sum_{i=1}^{k}a^i = \eta\|P\|_F^k\frac{a(1-a^k)}{1-a}. \tag{23}$$

□

The proof of Theorem 5 is as follows.

Proof The total error of estimating π is represented by

$$\|\tilde{\pi} - \pi\|_2 \leq (1-c)\left[\sum_{k=1}^{K}c^k\|\widetilde{P^k} - P^k\|_2\right]\|v\|_2 \tag{24}$$

where c is a constant, and v is constant vector.

According to error analysis of direct sampling of P^k from Theorem 3,

$$\|\widetilde{P}^k - P^k\|_2 \leq k\eta\alpha\|P\|_F^k,$$

where $\eta > 0$ is small constant; thus, the total error of estimating π by the direct sampling method is proportional to

$$\sum_{k=1}^{K} c^k \| \widetilde{P}^k - P^k \|_2 \le \sum_{k=1}^{K} c^k \| P \|_F^k k\alpha$$

And since

$$\begin{aligned}\sum_{k=1}^{K} c^k \| P \|_F^k k\alpha &= \alpha \left[c\| P \|_F + 2c^2 \| P \|_F^2 + \cdots + kc^k \| P \|_F^k + \cdots + Kc^K \| P \|_F^K \right] \\ &= \alpha \left[\left(c\| P \|_F + \cdots + c^k \| P \|_F^k + \cdots + c^K \| P \|_F^K \right) \right. \\ &\quad \left. + \left(c^2 \| P \|_F^2 + \cdots + c^k \| P \|_F^k + \cdots + c^K \| P \|_F^K \right) + \cdots + c^K \| P \|_F^K \right] \\ &= \alpha \sum_{k=1}^{K} c^k \| P \|_F^k + \alpha \sum_{k=2}^{K} c^k \| P \|_F^k + \cdots + \alpha c^K \| P \|_F^K\end{aligned}$$

we have

$$\begin{aligned}\sum_{k=1}^{K} c^k \| \tilde{P}^k - P^k \|_2 &\le \sum_{k=1}^{K} c^k \| P \|_F^k k\alpha \\ &= \alpha \sum_{k=1}^{K} c^k \| P \|_F^k + \alpha \sum_{k=2}^{K} c^k \| P \|_F^k + \cdots + \alpha c^K \| P \|_F^K. \qquad (25)\end{aligned}$$

With certain choice of α_i for $1 \le i \le K$, according to error analysis of adaptive sampling of P^k in Theorem 4,

$$\| \widetilde{P^k} - P^k \|_2 \le \eta \| P \|_F^k \sum_{i=1}^{k} \alpha_i,$$

where $\eta > 0$ is small constant; thus, the total error of estimating π by the adaptive sampling method is proportional to

$$\begin{aligned}\sum_{k=1}^{K} c^k \| \widetilde{P^k} - P^k \|_2 &\le \sum_{k=1}^{K} c^k \| P \|_F^k \left(\sum_{i=1}^{k} \alpha_i \right) \\ &= c\| P \|_F \alpha_1 + c^2 \| P \|_F^2 (\alpha_1 + \alpha_2) \\ &\quad + \cdots + c^k \| P \|_F^k (\alpha_1 + \cdots + \alpha_k) + \cdots + c^K \| P \|_F^K (\alpha_1 + \cdots + \alpha_K) \\ &= \alpha_1 \sum_{k=1}^{K} c^k \| P \|_F^k + \alpha_2 \sum_{k=2}^{K} c^k \| P \|_F^k + \cdots + \alpha_K c^K \| P \|_F^K \qquad (26)\end{aligned}$$

□

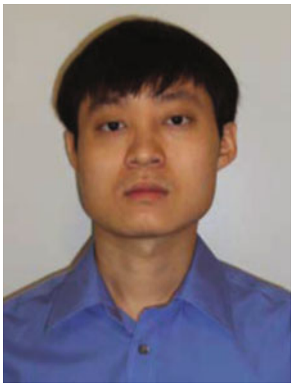
References

1. Achlioptas D, McSherry F (2007) Fast computation of low-rank matrix approximations. J. ACM 54(2):9
2. Avrachenkov K, Lebedev D (2006) Pagerank of scale-free growing networks. Internet Math 3(2):207–232
3. Berkhin P (2005) Survey: a survey on pagerank computing. Internet Math 2(1):73–120
4. Benczur A, Csalogány K, Sarlós T (2005) On the feasibility of low-rank approximation for personalized pagerank. In: Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, May 2005, pp 972–973

5. Borodin J, Roberts G, Tsaparas P (2005) Link analysis ranking: algorithms, theory, and experiments. ACM Trans Internet Technol 5: pp 231–297
6. Brin RMS, Page L, Winograd T (1999) The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999
7. Candès EJ, Plan Y (2010) Tight oracle bounds for low-rank matrix recovery from a minimal number of random measurements. CoRR. abs/1001.0339, 2010
8. Drineas P, Kannan R (2001) Fast Monte-Carlo algorithms for approximate matrix multiplication. In: 42nd annual symposium on foundations of computer science, Las Vegas, Nevada, USA, October 2001, pp 452–459
9. Drineas P, Kannan R, Mahoney MW (2006) Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication. SIAM J Sci Comput 36:132–157
10. Gleich DF, Gray AP, Greif C, Lau T (2010) An inner–outer iteration for PageRank. SIAM J Sci Comput 32(1):349–371
11. Haveliwala T, Kamvar S, Klein D, Manning C, Golub G (2003) Computing PageRank using power extrapolation. Stanford University Technical Report, July 2003
12. He G, Feng H, Li C, Chen H (2010) Parallel SimRank computation on large graphs with iterative aggregation. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, Washington, DC, USA, July 2010, pp 543–552
13. Kamvar S, Haveliwala T, Golub G (2003) Adaptive methods for the computation of pagerank. Technical Report 2003-26, Stanford InfoLab, April 2003
14. Kamvar S, Haveliwala T, Manning C, Golub G (2003) Extrapolation methods for accelerating pagerank computations. In: Proceedings of the twelfth international world wide web conference, Budapest, Hungary, May 2003, pp 261–270
15. Kwong MK, Zettl A (1991) Norm inequalities for the powers of a matrix. Am Math Mon 98(6):533–538
16. Langville AN, Meyer CD (2003) Survey: deeper inside pagerank. Internet Math 1(3):335–380
17. Lee CP, Golub GH, Zenios SA (2007) A two-stage algorithm for computing pagerank and multistage generalizations. Internet Math 4 (4):299–327
18. McSherry F (2005) A uniform approach to accelerated pagerank computation. In: Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, May 2005, pp 575–582
19. Osborne JRS, Wiggins E (2009) On accelerating the pagerank computation. Internet Math 6(2):157–172
20. Sidi A (2008) Methods for acceleration of convergence (extrapolation) of vector sequences. In: Wah BW (ed) Wiley encyclopedia of Computer Science and Engineering. Wiley, New York
21. SNAP (2007). Stanford Network Analysis Platform Standard Large Network Dataset Collection, Jure Leskovec. http://snap.stanford.edu/data/index.html
22. Wicks J, Greenwald AR (2007) More efficient parallel computation of pagerank. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, Amsterdam, The Netherlands, July 2007, pp 861–862
23. Wu G, Wei Y (2010) Arnoldi versus GMRES for computing pagerank: a theoretical contribution to google's pagerank problem. ACM Trans Inf Syst 28(3):11:1–11:28
24. Xue GR, Yang Q, Zeng HJ, Yu Y, Chen Z (2005) Exploiting the hierarchical structure for link analysis. In: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval, Salvador, Brazil, August 2005, pp 186–193
25. Zhu Y, Ye S, Li X (2005) Distributed pagerank computation based on iterative aggregation-disaggregation methods. In: ACM fourteenth conference on information and knowledge management (CIKM), Bremen, Germany, November 2005, pp 578–585

Author Biographies

Wenting Liu received her Bachelor degree in computational mathematics from Wuhan University, P.R. China, in 2006 and her Master degree in software engineering from Beihang University, Beijing, P.R. China, in 2011. She is currently a Ph.D. candidate in the school of computer engineering at Nanyang Technological University, Singapore. Her current research interests include data mining, statistical machine learning with applications in bioinformatics.



Guangxia Li received his bachelor's degree in computer science from Northwestern Polytechnical University, Xi'an, P.R. China, in 2006. He is currently a Ph.D. candidate in the school of computer engineering at Nanyang Technological University, Singapore. His research interests are information retrieval, data mining, and statistical machine learning. He work currently focuses on statistical and graph models and their applications to real-world problems in text and web mining, sentiment analysis, and social network analysis.

Dr. James Cheng is currently an Assistant Professor in the Department of Computer Science and Engineering at the Chinese University of Hong Kong (CUHK). Dr. Cheng received his Ph.D., M.Phil., and B.Eng. (First Class Honors) degrees in Computer Science from the Hong Kong University of Science and Technology in August 2008, August 2004, and August 2003, respectively. Before he joined CUHK, he was an Assistant Professor in the School of Computer Engineering at Nanyang Technological University, Singapore, from May 2009 to Dec. 2012. Dr. Cheng received the Hong Kong Young Scientist Award in 2010. He is a professional member of the ACM and the IEEE.