Fast computation of stationary joint probability distribution of sparse Markov chains

Weiyang Ding\textsuperscript{a,1}, Michael Ng\textsuperscript{a,2, 3}, Yimin Wei\textsuperscript{b,3}

\textsuperscript{a} Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong
\textsuperscript{b} School of Mathematical Sciences and Shanghai Key Laboratory of Contemporary Applied Mathematics, Fudan University, Shanghai, China

A R T I C L E   I N F O

Article history:
Received 29 December 2016
Received in revised form 26 October 2017
Accepted 27 October 2017
Available online 13 November 2017

Keywords:
Markov chains
Sparsity
Joint distribution
Optimization
Iterative methods

A B S T R A C T

In this paper, we study a fast algorithm for finding stationary joint probability distributions of sparse Markov chains or multilinear PageRank vectors which arise from data mining applications. In these applications, the main computational problem is to calculate and store solutions of many unknowns in joint probability distributions of sparse Markov chains. Our idea is to approximate large-scale solutions of such sparse Markov chains by two components: the sparsity component and the rank-one component. Here the non-zero locations in the sparsity component refer to important associations in the joint probability distribution and the rank-one component refers to a background value of the solution. We propose to determine solutions by formulating and solving sparse and rank-one optimization problems via closed form solutions. The convergence of the truncated power method is established. Numerical examples of multilinear PageRank vector calculation and second-order web-linkage analysis are presented to show the efficiency of the proposed method. It is shown that both computation and storage are significantly reduced by comparing with the traditional power method.

© 2017 IMACS. Published by Elsevier B.V. All rights reserved.

1. Introduction

One of the core and important research problems in information retrieval is to evaluate the importance or popularity of objects that can assist to handle many data mining tasks [2,8,10,11,16]. For instance, we employ ranking for the input query in a search engine, finding genes relevant to disease or extracting relevant communities in a social network, see [2,10,15,18,23,25]. For a detailed discussion, see the recent survey paper [8]. The PageRank algorithm [20] developed by Google is well-known and useful to determine the importance of nodes in a directed graph which is used to represent the World Wide Web and its hyperlinks among webpages. For other applications of Markov chains, we refer to [3,7,17,22,28]. Mathematically, it is interesting to note that the PageRank algorithm is used to compute a stationary probability distribution $x = [x_1, x_2, \ldots, x_n]^\top$ ( $^\top$ denotes the transpose of a vector) of a modified first-order Markov chain with $n$ states arising from a graph of $n$ nodes [11,20]:

\footnotesize
\begin{itemize}
\item \textsuperscript{1} This author’s work was partially supported by the Hong Kong Research Grants Council (Grant No. C1007-15G).
\item \textsuperscript{2} Research is supported in part by HKRGC GRF 12302715 and 12306616, and HKBU RC-ICRS/16-17/03.
\item \textsuperscript{3} This author is supported by the National Natural Science Foundation of China under grant 11771099 and International Cooperation Project of Shanghai Municipal Science and Technology Commission under grant 16510711200.
\end{itemize}

https://doi.org/10.1016/j.apnum.2017.10.008
0168-9274/© 2017 IMACS. Published by Elsevier B.V. All rights reserved.
\[ \mathbf{x} = \beta \mathbf{Px} + (1 - \beta) \mathbf{g}, \quad \text{or} \quad \mathbf{x}_i = \beta \sum_{j=1}^{n} p_{i,j} \mathbf{x}_j + (1 - \beta) g_i, \]

where \( \mathbf{P} = [p_{i,j}] \) is the given transition probability matrix (stochastic matrix) arising from the first-order Markov chain (all the entries \( p_{i,j} \) are non-negative and each column sum of \( \mathbf{P} \) is equal to one), \( \mathbf{g} = [g_1, g_2, \ldots, g_n]^\top \) is the given probability vector for the associated nodes [20], and \( \beta \) is a positive number in \((0, 1]\). When \( \beta = 1 \), Equation (1) is equivalent to solving a stationary probability distribution for a first-order Markov chain. When \( \mathbf{P} \) is irreducible, the solution \( \mathbf{x} \) is unique and its entries \( x_i \) are always positive [1]. For \( \beta < 1 \), it can be interpreted that a random walker iteratively visits from the current \( i \)-th node to the other nodes according to this probability vector: \( \beta [p_{i,1}, p_{i,2}, \ldots, p_{i,n}]^\top + (1 - \beta) [g_1, g_2, \ldots, g_n]^\top \). The probabilities are related to the transition probability matrix and the given probability vector for the associated nodes. The random walker has the steady state probabilities that will finally stay at different nodes. The corresponding solution \( \mathbf{x} \) is also unique, see [10,23].

1. Sparse second-order Markov chains

The main objective of this paper is to study efficient solution methods for the computation of stationary joint probabilities of sparse second-order Markov chains/multilinear PageRank:

\[ x_{i,j} = \beta \sum_{k=1}^{n} p_{i,j,k} x_{j,k} + (1 - \beta) g_{i,j}, \quad i, j = 1, 2, \ldots, n. \]

The quantity \( p_{i,j,k} \) represents the probability that the process will make a transition to the \( i \)-th node given that currently the process is in the \( j \)-th node and the process was in the \( k \)-th node previously. Also \( g_{i,j} \) is the given probability for the associated \( i \)-th and \( j \)-th nodes [9,20]. It is interesting to note in many practical applications that there may be many \( p_{i,j,k} \) equal to zero. For instance, when we study the citation ranking among researchers in a publication community, second-order transition probabilities can be constructed by using their successive citations [14,19]. As a researcher may not cite the papers written by other researchers in the whole community especially when the number of researchers \( n \) is very large, we expect many \( p_{i,j,k} \) would be zero. Here the \( i \)-th researcher cites the \( k \)-th researcher via the citation by the \( j \)-th researcher. For example, when we study successive links of webpages in World Wide Web, second-order transition probabilities can be constructed that the \( i \)-th webpage is linked to the \( k \)-th webpage via the \( j \)-th webpage. We expect most of these probabilities are zero especially when the number of webpages is very large. The technique for handling dangling states has been considered in [9,14,19], see Section 2.2. Moreover, there are many zeros in \( g_{i,j} \) in these applications. For example, we target for some specific query keywords, publications and researchers in communities [21], and we only set a few relevant keywords, publications and researchers in multilinear PageRank applications in which they refer the corresponding non-zeros in \( g_{i,j} \).

We note that any second-order Markov chain with \( n \) nodes, i.e., \( \beta = 1 \) in (2), can be reformulated into an \( n^2 \)-by-\( n^2 \) linear system. The block form of the equations can be described as follows:

\[
\begin{bmatrix}
    x_{1,j} \\
    x_{2,j} \\
    \vdots \\
    x_{n,j}
\end{bmatrix} =
\begin{bmatrix}
    p_{1,j,1} & p_{1,j,2} & \cdots & p_{1,j,n} \\
    p_{2,j,1} & p_{2,j,2} & \cdots & p_{2,j,n} \\
    \vdots & \vdots & \ddots & \vdots \\
    p_{n,j,1} & p_{n,j,2} & \cdots & p_{n,j,n}
\end{bmatrix}
\begin{bmatrix}
    x_{j,1} \\
    x_{j,2} \\
    \vdots \\
    x_{j,n}
\end{bmatrix},
\quad j = 1, 2, \ldots, n.
\]

or

\[ \mathbf{X}(\cdot,j) = \mathbf{P}_j \mathbf{X}(j,:)^\top, \quad j = 1, 2, \ldots, n. \]

We remark that the left hand side vector \( \mathbf{X}(\cdot,j) \) is different from \( \mathbf{X}(j,:)^\top \) and therefore the linear system is not separable in each block. By adopting Matlab notations, \( \mathbf{X}(\cdot,j) \) is the \( j \)-th column of \( \mathbf{X} \) and \( \mathbf{X}(j,:) \) is the \( j \)-th row of \( \mathbf{X} \). Also \( \mathbf{X}(j,:)^\top \) is the transpose of \( \mathbf{X}(j,:) \) and \( \mathbf{X}(j,:)^\top \) is a column vector. It is clear that the column summation of \( \mathbf{P}_j \) is equal to 1, i.e., \( \mathbf{P}_j \) is a stochastic matrix.

If the corresponding transition probability matrix in the above equation arises from a second-order Markov chain is irreducible, then the stationary joint probability distribution \( \mathbf{X} \) is unique. Similar to (1), we can deal with the case of modified higher-order Markov chains for multilinear PageRank applications. A detailed discussion can be found in [9]. The main computational problem is to calculate and store an \( n \)-by-\( n \) matrix \( \mathbf{X} \). According to (2), \( \mathbf{X} \) can be obtained by solving a matrix system where its size is \( n^2 \)-by-\( n^2 \). For example, when \( n = 10000 \), the matrix size would be of hundred million by hundred million, and the solution would have hundred million unknowns. Indeed, many large-scale applications can be found in web analysis, publication networks, social networks and bioinformatics, see for instance [8,14,15,19]. In these applications, \( n \) refers to the number of webpages, the number of authors, the number of users, and the number of genes respectively that can be very large.
Let us consider an example to motivate the proposed computational scheme. We randomly construct $p_{i,j,k}$ with $n = 100$, and the percentage of non-zeros of $p_{i,j,k}$ is $0.1\%$.\footnote{Here the technique of the dangling state is used [9,14,19], i.e., for each fixed $(j,k)$, when $p_{i,j,k} = 0$ for $1 \leq i \leq 100$, we set $p_{i,j,k} = 1/100$ for $1 \leq i \leq 100$.} We make use of the classical power method to find out the stationary joint probability distribution $X = [x_{i,j}]$ satisfying $x_{i,j} = \sum_{k=1}^{n} p_{i,j,k} x_{j,k}$. In Fig. 1, we see that there are some significantly large entries in each column. However, it is clear that $X$ is dense (is not sparse). Indeed, all the entries of $X$ are positive. We also observe the entries in the same column, except for those significantly large values, have about the same value (but dependent on the column position). In Fig. 2, we show the distribution of the values of entries of $X$. The significantly large entries have the value being greater than or equal to $0.6 \times 10^{-3}$. For other entries, their values are in between $0.1 \times 10^{-3}$ and $0.2 \times 10^{-3}$.

Based on these observations, we propose to study a sparse approximation of $x_{i,j}$ under the sparsity assumptions on $p_{i,j,k}$ and $g_{i,j}$. In the proposed algorithm, we only store large magnitude entries of such approximation, and therefore both the computation and storage can be significantly reduced. According to this sparse approximation scheme, in Section 2, we can design a power-like iterative algorithm with an $\ell_1$-optimization subproblem in each step. Moreover, a closed form solution to this $\ell_1$-optimization subproblem is proposed. The convergence of the iterative algorithm is also analyzed and studied in Section 3. In Section 4, numerical examples on synthetic and real data sets are presented to illustrate the usefulness of the proposed algorithm. Numerical results show that the performance of the proposed method in terms of computational time and storage is significantly better than that of power method. Finally, some concluding remarks are given in Section 5.
2. The proposed formulation

2.1. The related work

In [12], Li and Ng considered an approximation to a stationary joint probability distribution by using symmetric rank-one factorization, i.e., they solved the following tensor equations for second-order Markov chains:

\[ x_i = \sum_{j,k=1}^{n} p_{i,j,k} x_j x_k, \quad 1 \leq i \leq n. \]  

(4)

In this model, the joint probability distribution was given by \( x_i x_j \), which is different from \( x_{ij} \) that contains higher-order relations among \( n \) nodes for useful data analysis, see [13]. Li and Ng [12] also showed that when the corresponding transition probability tensor \( P = (p_{i,j,k}) \) is irreducible, there exists \( x \) with positive entries such that (4) is solvable. With some suitable conditions of \( P \), the entries \( x_i \) are unique. Under the same uniqueness condition, they studied a simple iterative algorithm for computing stationary joint probabilities, and showed its convergence. It is interesting to note that the theoretical results are related to Perron–Frobenius theorem for non-negative tensors, see [6,27,29] and the recent survey paper [5]. Similar to (1), Gleich, Lim, and Yu [9] considered the solution of a modified second-order Markov chain (multilinear PageRank) as follows:

\[ x_i = \beta \sum_{j,k=1}^{n} p_{i,j,k} x_j x_k + (1 - \beta) g_i, \quad 1 \leq i \leq n. \]  

(5)

Gleich, Lim, and Yu [9] showed that \( x_i \) are unique when \( \beta < 1/(m - 1) \). Moreover, they showed the convergence of the fixed-point iteration, a shifted fixed-point iteration, an inverse iteration, and a Newton iteration under the same assumption. Several numerical examples are presented to show the usefulness of their proposed algorithms.

Recently, Wu, Bian, and Zhang [26] proposed to formulate second-order Markov chains by using edge-to-edge transition probabilities, and developed Monte Carlo methods to compute stationary probability distributions. The edge-to-edge formulation is really competitive when this second-order Markov chain is determined by a graph and the underlying graph is extremely sparse. Nevertheless, many second-order Markov chains may not be induced by a graph. For Examples 4.1 and 4.2 in Section 4, our numerical results show that edge-to-edge solution vector densifies, and therefore, the edge-to-edge formulation may not be competitive in computational time and storage. Moreover, when the edge-to-edge matrix is only built based on the existence of edges in the given data, it is assumed that some two-step transition probabilities between two nodes are not considered, i.e., they are zero in probability calculation. The edge-to-edge formulation may miss some information and connection. For instance, there are two edges \( 1 \rightarrow 2 \) and \( 2 \rightarrow 3 \) in a graph. Even there is no direct edge between \( 1 \rightarrow 3 \), there is a connection between the two nodes \( 1 \) and \( 3 \). In contrast, the proposed method does not make this assumption, and study the sparsity nature of two-step transition probabilities to construct an efficient algorithm for computation of stationary joint probability distribution of second-order Markov chains. The proposed method might be able to be applied directly to higher order Markov chains, whereas the edge-to-edge formulation would not be straightforward.

2.2. The idea

Let us first describe the matrix system in (2) arising from second-order (modified) Markov chains. According to (3), we can write the resulting matrix system in (2) as follows:

\[ X(j,:) = \beta P_j X(j,:)^T + (1 - \beta) G(:,j), \quad j = 1, 2, \ldots, n, \]  

(6)

where \( G \) is non-negative and the summation of all its entries is equal to 1. Besides there are many entries \( p_{i,j,k} \) to be zeros in a sparse Markov chain, there may be many dangling nodes [9,20]. When \( p_{i,j,k} \) are equal to zero for all \( i = 1, 2, \ldots, n \), the state \( (j,k) \) is called the dangling state [9,20], which refers to no connection from the state \( (j,k) \). Similar to PageRank [9,14,19], \( p_{i,j,k} \) can be set to be \( 1/n \), i.e., it has an equal chance to visit for the \( n \) states \( (1,j), (2,j), \ldots, (n,j) \) from \( (j,k) \). By combining the above two considerations, each \( P_j \) can be written as the sum of two matrices: \( P_j = Q_j + N_j \), where \( Q_j \) contains non-zero entries of \( P_j \) and \( N_j \) contains the entries arising from the set of dangling states \( D \). There are two remarks: (i) The storage of \( P_j \) can be efficient as we only need to store non-zero entries of \( Q_j \) and the set of dangling states \( D \); (ii) the matrix-vector multiplication \( P_j z \) can be calculated efficiently as both \( Q_j z \) and \( N_j z \) can be computed inexpensively according to the sparse structure of \( Q_j \) and the uniform structure of \( N_j \). Nevertheless, the computational and the storage cost of the solution \( X \) can still be very expensive especially when \( n \) is large.

The main idea of the proposed method is to approximate the stationary joint probability distribution matrix \( X \) of a sparse second-order Markov chain, i.e., the case when \( \beta = 1 \) in (6), into two parts

\[ X \approx S + 1_n u u^T, \]  

(7)

where \( S \) is expected to be a sparse matrix containing those significantly large entries, \( 1_n = [1, 1, \ldots, 1]^T \) is the all-one vector of size \( n \), and \( u = [u_1, u_2, \ldots, u_n]^T \) is a vector consisting of the background values \( u_j \) for the \( j \)-th column \( X(:, j) \). Thus each
column \(X(:, j)\) of the stationary joint probability distribution matrix is expressed as the summation of a background value \(u_j\) and some significant large entries \(S(:, j)\). The storage cost of a full matrix \(X\) is \(n^2\), which is huge when the size \(n\) is large. However, when the stationary joint probability distribution matrix can be written into the form (7), the storage requirement can be largely reduced to \(\text{nnz}(S) + n\), where \(\text{nnz}(\cdot)\) stands for the number of non-zero entries of a matrix.

On the other hand, when \(\beta = 1\), we note that there are several properties of the stationary joint probability distribution matrix \(X\):

\(\text{(C1)}\) The entries of \(X\) are non-negative, i.e., \(X \geq 0\);
\(\text{(C2)}\) The summation of all the entries of \(X\) is normalized, i.e., \(\sum(X) = 1\);
\(\text{(C3)}\) The column summations and the row summations of \(X\) are equal, i.e., \(\sum(X(:, j)) = \sum(X(j, :)\) for \(1 \leq j \leq n\).

We remark that if given any matrix \(X\) satisfies \(\text{(C1)}\–\text{(C3)}\), then we can think of it as being the stationary joint probability distribution of a second-order Markov chain defined by \(P_j = X(:, j) \cdot 1_n^\top / \sum(X(:, j))\) for \(j = 1, 2, \ldots, n\).

In this paper, we formulate an algorithm for finding a sparse approximation \(S + 1_nu^\top\) of \(X\) by imposing the last two conditions, i.e., \(\text{(C2)}\) and \(\text{(C3)}\). For the first condition in \(\text{(C1)}\), we have \(X \geq 0\) by \(S \geq 0\) and \(u \geq 0\), which can perform the computations more efficiently. Similarly, for the solution \(X\) of a modified second-order Markov chain, i.e., \(\beta \in (0, 1)\) in (6), we also have three parallel conditions:

\(\text{(C1)’}\) The entries of \(X\) are non-negative, i.e., \(X \geq 0\);
\(\text{(C2)’}\) The summation of all the entries of \(X\) is normalized, i.e., \(\sum(X) = 1\);
\(\text{(C3)’}\) \(\sum(X(:, j)) = \beta \cdot \sum(X(j, :)\) for \(1 \leq j \leq n\).

Moreover, the corresponding sparse approximation of the solution is \(\beta(S + 1_nu^\top) + (1 - \beta)G\) with \(\beta\) and \(G\) given, \(S \geq 0\) and \(u \geq 0\).

2.3. The truncated power method

We solve the problem iteratively via a fixed point operator \(\mathcal{D}\), i.e., \(Y_{k+1} = \mathcal{D}(Y_k)\). Here we expect that \(\lim_{k \to \infty} Y_k\) exists and the limit is a sparse approximation of \(X\). According to (6) and the sparse approximation of \(X\), each iteration loop contains two steps.

(i) The first step is to make use of power-type method and employ the sparsity in the computational procedure. The advantage of using the power-type method is that the corresponding matrix–vector multiplication can be performed very efficiently at each iteration. The sparsity calculation procedure can guarantee that the resulting matrix is sparse. More importantly, we only store the significantly large entries in the results. For simplicity, assuming \(Y_k\) is represented by \(Y_k = \beta(S_k + 1_nu_k^\top) + (1 - \beta)G\), the above set of operations can be rewritten into

\[Z = \mathcal{F}(\mathcal{D}(Y_k)),\]

where \(\mathcal{D}(\cdot)\) represents the operation

\[\mathcal{D}(Y_k)(i, :) = P_i Y_k(i, :)^\top, \quad 1 \leq i \leq n,\]

and \(\mathcal{F}(\cdot)\) represents the truncated operation related to sparse approximation.

(ii) The second step is to implement the operation of the output as shown in (6):

\[Y_{k+1} = \mathcal{F}(Z) := \beta(S_{k+1} + 1_nu_{k+1}^\top) + (1 - \beta)G.\]

Thus the operator \(\mathcal{D}(\cdot)\) is the composition of three operators \(\mathcal{D}(\cdot)\), \(\mathcal{F}(\cdot)\), and \(\mathcal{F}(\cdot)\).

In the first step, we choose an initial guess \(Y_0 = \beta(S_0 + 1_nu_0^\top) + (1 - \beta)G\) such that \(\sum(Y_0) = 1\). In the iterative scheme, we preserve the sparse approximation of the stationary joint probability distribution, i.e., \(Y_k\) has the form \(\beta(S_k + 1_nu_k^\top) + (1 - \beta)G\), where \(k\) is the iteration index. Here we propose the following sparse optimization problem for (8) to obtain \((S_{k+1}, u_{k+1})\) as the optimal point of

\[\min \frac{1}{2} \| (S + 1_nu^\top) - \mathcal{D}(\beta(S_k + 1_nu_k^\top) + (1 - \beta)G) \|^2_F + \alpha \|S\|_1,\]

s.t. \(S \geq 0, u \geq 0\),

\[\text{sum}(S(:, i)) + n \cdot u_i = \beta \cdot \text{sum}(S_k(i, :)^\top + u_k) + (1 - \beta) \cdot \text{sum}(G(:, i)), \quad i = 1, 2, \ldots, n,\]

where the constraint \(\text{sum}(S_{k+1} + 1_nu_{k+1}^\top) = 1\) is naturally satisfied because of \(\text{sum}(Y_0) = 1\) and the second constraint. Here \(\| \cdot \|_1\) refers to the sum of the magnitude of all the entries of a matrix and serves as an \(\ell_1\)-penalty term, which introduces the sparsity. The greater parameter \(\alpha\) we choose, the sparser matrix \(S\) we will obtain. It is easily understood that a fixed
point of this iterative scheme must satisfy the above three conditions (C1)’–(C3)’ so that it is a stationary joint probability distribution.

It is interesting to note that the above problem can be split into $n$ disjoint subproblems by partitioning $S$ into $n$ columns:

\[
\min \frac{1}{2} \left\| S(:,i) + 1_n \cdot u_i - P_i \cdot (\beta(S_k(:,i))^\top + u_k) + (1 - \beta)G(:,i) \right\|_2^2 + \alpha \left\| S(:,i) \right\|_1, \]

s.t. $S(:,i) \geq 0$, $u_i \geq 0$, 

\[
\sum S(:,i) + n \cdot u_i = \beta \cdot \sum (S_k(:,i))^\top + u_k + (1 - \beta) \cdot \sum (G(:,i)),
\]

for each $i = 1, 2, \ldots, n$. Furthermore, these subproblems can be unified into the same form as follows:

\[
\min \frac{1}{2} \left\| s + 1_n \cdot \mu - b \right\|_2^2 + \alpha \left\| s \right\|_1, 
\]

s.t. $s \geq 0$, $\mu \geq 0$,

\[
1_n^\top s + n \cdot \mu = 1_n^\top b. 
\]

That is, we solve the same optimization problem for each column. The computation of the vector $b$ can be regarded as a step in the power method. We will shortly see in the next section that this subproblem has a closed form solution.

### 3. The algorithm and its convergence analysis

#### 3.1. The convex optimization subproblem

In this subsection, we first study the optimization problem (11). The first statement about this problem is that the equality constraint $1_n^\top s + n \cdot \mu = 1_n^\top b$ can be removed.

**Lemma 3.1.** The optimal point of the optimization problem (11) is also the optimal point of the convex optimization problem

\[
\min \frac{1}{2} \left\| s + 1_n \cdot \mu - b \right\|_2^2 + \alpha \left\| s \right\|_1, 
\]

s.t. $s \geq 0$, $\mu \geq 0$, 

(11)

i.e., (11) without the equality constraint.

**Proof.** Denote the objective function

\[
f(s, \mu) := \frac{1}{2} \left\| s + 1_n \cdot \mu - b \right\|_2^2 + \alpha \left\| s \right\|_1.
\]

Since we consider the minimization problem over the non-negative region, the objective function is differential at all the points in this region with

\[
\nabla_{s_i} f = s_i + \mu - b_i + \alpha \quad (i = 1, 2, \ldots, n) \quad \text{and} \quad \nabla_{\mu_i} f = 1_n^\top s + n \cdot \mu - 1_n^\top b.
\]

Let $(s^\star, \mu^\star)$ be a minimizer of this convex optimization problem (12). According to the convex optimization theory (see [4]), the optimal condition can be expressed as

\[
\begin{cases}
    s^\star \geq 0, \quad &\nabla_{s_i} f(s^\star, \mu^\star) \geq 0, \\
    \mu^\star \geq 0, \quad &\nabla_{\mu_i} f(s^\star, \mu^\star) \geq 0, \\
    s^\star_i \cdot \nabla_{s_i} f(s^\star, \mu^\star) = 0, \quad &i = 1, 2, \ldots, n, \\
    \mu^\star \cdot \nabla_{\mu_i} f(s^\star, \mu^\star) = 0.
\end{cases}
\]

From the optimal condition, we can see that if $\mu^\star$ is positive, then the gradient about $\mu$ at the optimal point $\nabla_{\mu_i} f(s^\star, \mu^\star) = 0$, which is exactly the equality constraint in (11). Next we show that the optimal $\mu^\star$ is always non-zero for all non-zero non-negative vector $b$.

The optimal condition can be translated into the following four situations.

(S1) If $s^\star_i > 0$, then $s^\star_i + \mu^\star = b_i - \alpha$, which implies $b_i > \mu^\star + \alpha$.

(S2) If $s^\star_i = 0$, then $b_i \leq \mu^\star + \alpha$.

(S3) If $\mu^\star > 0$, then $1_n^\top s^\star + n \cdot \mu^\star = 1_n^\top b$.

(S4) If $\mu^\star = 0$, then $1_n^\top s^\star + n \cdot \mu^\star \geq 1_n^\top b$.

Assume that $\mu^\star = 0$. When $s^\star$ is all-zero, we know from the situation (S4) that $1_n^\top b \leq 0$. Since $b$ is non-negative, thus it is also a vector of all-zero. Therefore, for any non-zero non-negative $b$, there must exist some positive $s^\star_i$ such that the sum of all the entries of $s^\star + 1_n \cdot \mu^\star$ is

\[
1_n^\top s^\star + n \cdot \mu^\star = \sum_{s^\star_i > 0} s^\star_i = \sum_{s^\star_i > 0} (b_i - \alpha) < 1_n^\top b.
\]
But it is a contradiction to (S4). It implies that the optimal background value $\mu^*$ must be positive for a non-zero non-negative vector $\mathbf{b}$, then the constraint $\mathbf{1}_n \mathbf{s} + n \cdot \mu^* = \mathbf{1}_n \mathbf{b}$ is naturally satisfied. Even when $\mathbf{b}$ is all-zero, this equality constraint is still satisfied. Hence, we solve the convex minimization problem over the non-negative region (12) instead of the original problem (11) with an additional equality constraint. \[ \square \]

Denote two index sets $\Lambda_0 = \{i : s_i^* = 0\} \quad \text{and} \quad \Lambda_1 = \{i : s_i^* > 0\}.$

We have some further descriptions about these index sets corresponding to an optimal point of (12).

**Lemma 3.2.** Let $(\mathbf{s}^*, \mu^*)$ be a minimizer of the convex optimization problem (12). Then

1. $\Lambda_0$ is always non-empty, and
2. $b_{i_0} < b_{i_1}$ for all $i_0 \in \Lambda_0, i_1 \in \Lambda_1.$

**Proof.** If all $s_i^*$ $(i = 1, 2, \ldots, n)$ are non-zero, then from (S1) in the proof of the previous lemma, we have

$$\mathbf{1}_n \mathbf{s}^* + n \cdot \mu^* = \sum_{i=1}^n (b_i - \alpha) < \mathbf{1}_n \mathbf{b},$$

which is a contradiction to (S3). So there must be some entry of $\mathbf{s}^*$ being zero, i.e., $\Lambda_0$ is non-empty. Moreover, we can easily conclude that $b_{i_0} < b_{i_1}$ for all $i_0 \in \Lambda_0$ and $i_1 \in \Lambda_1$ by comparing (S1) and (S2). \[ \square \]

Without loss of generality, we assume that $b_1 \geq b_2 \geq \ldots \geq b_n$, the first $d$ entries of $\mathbf{s}^*$ are non-zero, and the other entries are zero. That is, $\Lambda_1 = \{1, 2, \ldots, d\}$ and $\Lambda_0 = \{d+1, d+2, \ldots, n\}$. Here, $d$ might be zero, i.e., all the entries of $\mathbf{s}^*$ are zeros. But $d$ must be smaller than $n$ since $\Lambda_0$ is always non-empty. When the number $d$ of non-zero entries of $\mathbf{s}^*$ is known, denote

$$\mathbf{s}^* = [s_1^*, s_2^*, \ldots, s_d^*]^\top \quad \text{and} \quad \mathbf{b} = [b_1, b_2, \ldots, b_d]^\top.$$  

Then (S1) is rewritten into a linear system of equations

$$\begin{bmatrix} \mathbf{1}_d \\ \mathbf{1}_n \end{bmatrix} \begin{bmatrix} \mathbf{s}^* \\ \mu^* \end{bmatrix} = \begin{bmatrix} \mathbf{b} - \mathbf{1}_d \cdot \alpha \end{bmatrix}. $$

It is easily verified that the coefficient matrix is non-singular for each $d < n$. Moreover, the solution of this linear system has an explicit form

$$\begin{align*}
\begin{cases}
    s_i^* = b_i - \alpha - \mu^*, & i = 1, 2, \ldots, d, \\
    \mu^* = \frac{1}{n-d} \sum_{j=d+1}^n b_j + \frac{d}{n-d} \cdot \alpha.
\end{cases}
\end{align*} \tag{13}$$

Suppose $d$ is known, we have obtained an explicit solution of the optimization problem (12). Furthermore, the minimization problem (12) has a finite number of global minimizers because the candidates of $d$ are $0, 1, \ldots, n-1$. Therefore, the problem (12) has a unique global minimizer since it is a convex optimization problem.

The remaining question is how to fix the number of non-zeros in $\mathbf{s}^*$, i.e., $d$. By combining (S1), (S2) and the explicit solution in (13), we have

$$\begin{align*}
\begin{cases}
    b_i > \frac{1}{n-d} \sum_{j=d+1}^n b_j + \frac{n}{n-d} \cdot \alpha & \text{for } i = 1, 2, \ldots, d, \\
    b_i \leq \frac{1}{n-d} \sum_{j=d+1}^n b_j + \frac{n}{n-d} \cdot \alpha & \text{for } i = d+1, d+2, \ldots, n.
\end{cases}
\end{align*}$$

Because of the order $b_1 \geq b_2 \geq \ldots \geq b_n$, the above criteria is equivalent to

$$\sum_{j=d+1}^n b_j + n\alpha \geq (n-d)b_{d+1}.$$ \tag{14}

It is straightforward to verify that
\( (n-d')b_{d'+1} > \sum_{j=d'+1}^{n} b_j + n\alpha \) for each \( d' < d \), and

\( (n-d')b_{d'} \leq \sum_{j=d'+1}^{n} b_j + n\alpha \) for each \( d' > d \).

Therefore, there is a unique number \( d \in [0, 1, \ldots, n-1] \) satisfying the criteria (14) for an arbitrary non-negative vector \( b \). Notice that \( d \) depends on both \( b \) and \( \alpha \).

**Theorem 3.3.** Let \( b \) be a non-negative vector satisfying \( b_1 \geq b_2 \geq \cdots \geq b_n \). The convex optimization problem

\[
\min \frac{1}{2}\|s + 1_n \cdot \mu - b\|_2^2 + \alpha\|s\|_1, \\
\text{s.t. } s \geq 0, \mu \geq 0
\]

has a unique minimizer \((s^*, \mu^*)\) with the explicit form

\[
s_i^* = \begin{cases} 
  b_i - \alpha - \mu^*, & 1 \leq i \leq d, \\
  0, & d < i \leq n,
\end{cases}
\]

and \( \mu^* = \frac{1}{n-d} \sum_{j=d+1}^{n} b_j + \frac{d}{n-d} \cdot \alpha \),

where \( d \) is the unique integer satisfying

\[
(n-d)b_d > \sum_{j=d+1}^{n} b_j + n\alpha \geq (n-d)b_{d+1}.
\]

An obvious drawback of this method is that we need to sort the entries of \( b \). Nevertheless, only those \( b_i > \alpha \) rather than all the entries are required to sort, since \( s_i^* \) must be zero for an index \( i \) such that \( b_i \leq \alpha \) according to (S1) in the optimal condition. Therefore, we summarize our algorithm for solving the subproblem (11) in our truncated power method as Algorithm 1.

**Algorithm 1** The thresholding method for solving the subproblem (11).

**Input:** The non-negative vector \( b \) and the parameter \( \alpha \);

**Output:** The sparse vector \( s \) (the non-zero components) and the background value \( \mu \);

1: Sort the components of \( b \) into \( b_1 \geq \cdots \geq b_i \geq \cdots \geq b_n \);

2: Find \( d \in [0, 1, \ldots, r] \) such that \( (n-d)b_d > \sum_{j=d+1}^{n} b_j + n\alpha \geq (n-d)b_{d+1} \) \( (b_n := +\infty) \);

3: \( \mu = \frac{1}{n-d} \sum_{j=d+1}^{n} b_j + \frac{k}{n-d} \cdot \alpha \);

4: \( s_i = \begin{cases} 
  b_i - \alpha - \mu, & j = 1, 2, \ldots, d, \\
  0, & j = d+1, d+2, \ldots, n.
\end{cases} 
\)

5: return \( s, \mu \);

The computational complexity of Algorithm 1 is given as follows. The first line of the algorithm can be done by two steps: (i) judging whether \( b_i > \alpha \) or \( b_i \leq \alpha \) for all \( i = 1, 2, \ldots, n \), (ii) sort the \( r \) components which are greater than \( \alpha \). The computational complexities of these two steps are \( O(n) \) and \( O(r \log r) \), respectively. Since \( r \) is generally much smaller than the size \( n \) for the sparse Markov chains, thus the computational complexity of the first line is about \( O(n) \). Moreover, the following three lines in the algorithm cost \( O(r), O(n-d) \), and \( O(d) \) operations, respectively. To sum up, the computational complexity of Algorithm 1 is \( O(n) \) which is a linear complexity.

**3.2. The convergence of the truncated power method**

In this subsection, we show the convergence of the proposed truncated power method with sparsity, i.e., Algorithm 2.

We begin with the subproblems. Given a non-negative vector \( b \in \mathbb{R}_+^n \), we denote the unique optimal point of the minimization problem (12) as \((s^*, \mu^*)\). Then an operator on the non-negative region to itself \( \mathcal{T} : \mathbb{R}_+^n \rightarrow \mathbb{R}_+^n \) can be defined by

\[
\mathcal{T}(b) = s^* + 1 \cdot \mu^*.
\]

Furthermore, we know from the above discussion that \( \|\mathcal{T}(b)\|_1 = \|b\|_1 \) and

\[
\mathcal{T}(b)_j = \begin{cases} 
  b_j - \alpha, & \text{if } b_j > \alpha + \mu^*, \\
  \mu^*, & \text{if } b_j \leq \alpha + \mu^*.
\end{cases}
\]
When the optimal background $\mu^*$ is zero, the operator $T$ is exactly the soft thresholding operator. And the following theorem tells that the operator $T$ is non-expansive.

**Theorem 3.4.** The operator $T : \mathbb{R}^n_+ \rightarrow \mathbb{R}^n_+$ is non-expansive, i.e.,

$$\|T(b) - T(\tilde{b})\|_1 \leq \|b - \tilde{b}\|_1.$$  

**Proof.** Let $(s, \mu)$ and $(\tilde{s}, \tilde{\mu})$ be the optimal points of the problem (12) corresponding to $b$ and $\tilde{b}$, respectively. Denote the following four index sets

$$\Omega_1 = \{i : s_i > 0, \tilde{s}_i > 0\}, \quad \Omega_2 = \{i : s_i > 0, \tilde{s}_i = 0\},$$

$$\Omega_3 = \{i : s_i = 0, \tilde{s}_i > 0\}, \quad \Omega_4 = \{i : s_i = 0, \tilde{s}_i = 0\},$$

and their numbers of elements as $c_1$, $c_2$, $c_3$, $c_4$, respectively. Note that $c_1 + c_2$ and $c_1 + c_3$ are the numbers of non-zero entries of $s$ and $\tilde{s}$, respectively. Then we have

$$T(b)_i - T(\tilde{b})_i = \begin{cases} 
b_i - \tilde{b}_i, & i \in \Omega_1, \\
b_i - \tilde{b}_i - \mu - \tilde{\mu}, & i \in \Omega_2, \\
\mu - \tilde{\mu}, & i \in \Omega_3, \\
\mu - \tilde{\mu}, & i \in \Omega_4.
\end{cases}$$

Without loss of generality, we assume that $\mu \geq \tilde{\mu}$. For those indices $i \in \Omega_1$, it is obvious that

$$\sum_{i \in \Omega_1} |T(b)_i - T(\tilde{b})_i| = \sum_{i \in \Omega_1} |b_i - \tilde{b}_i|. \quad (15)$$

When the index $i \in \Omega_2$, we know that $b_i > \mu + \alpha$ from the definition of $T$, thus $T(b)_i - T(\tilde{b})_i > \mu - \tilde{\mu} > 0$. Recall that $\mu \geq \tilde{\mu}$, then we have

$$\sum_{i \in \Omega_2} |T(b)_i - T(\tilde{b})_i| = \sum_{i \in \Omega_2} (b_i - \alpha - \tilde{\mu}) + \sum_{i \in \Omega_4} (\mu - \tilde{\mu})$$

$$= \sum_{i \in \Omega_2} b_i + c_4\mu - (c_2 + c_4)\tilde{\mu} - c_2\alpha$$

$$= \sum_{i \in \Omega_2} b_i + (c_3 + c_4)\mu - (c_2 + c_4)\tilde{\mu} - c_3\mu - c_2\alpha. \quad (16)$$

Further notice from Theorem 3.3 and the definitions of $\Omega_3, \Omega_4$ that

$$\mu = \frac{1}{c_3 + c_4} \sum_{i \in \Omega_3 \cup \Omega_4} b_i + \frac{c_1 + c_2}{c_3 + c_4},$$

which can be rewritten into

$$(c_3 + c_4)\mu = \sum_{i \in \Omega_3 \cup \Omega_4} b_i + (c_1 + c_2)\alpha.$$  

Similarly, we also have

$$(c_2 + c_4)\tilde{\mu} = \sum_{i \in \Omega_3 \cup \Omega_4} \tilde{b}_i + (c_1 + c_3)\alpha.$$  

Substitute the above two equations into (16), we obtain that
\[
\sum_{i \in \Omega_2 \cup \Omega_4} |I(b)_i - I(b')]_i = \sum_{i \in \Omega_2 \cup \Omega_4} (b_i - b_i') + \sum_{i \in \Omega_3} (b_i - \mu - \alpha).
\]  

(17)

When the index \(i \in \Omega_3\), we have \(b_i \leq \mu + \alpha\). Hence,

\[
\sum_{i \in \Omega_3} |I(b)_i - I(b')]_i = \sum_{i \in \Omega_3} |\mu + \alpha - b_i| \\
\leq \sum_{i \in \Omega_3} (|\mu + \alpha - b_i| + |b_i - b_i'|) = \sum_{i \in \Omega_3} (\mu + \alpha - b_i) + \sum_{i \in \Omega_3} b_i - \tilde{b}_i |.
\]

(18)

Combining (15), (17), and (18), we finally obtain that

\[
\left\| \tilde{I}(b) - \tilde{I}(b') \right\|_1 = \sum_{i \in \Omega_2 \cup \Omega_3 \cup \Omega_4} \left| \tilde{I}(b)_i - \tilde{I}(b')_i \right| \leq \sum_{i \in \Omega_2 \cup \Omega_3 \cup \Omega_4} \left| b_i - \tilde{b}_i \right| = \|b - \tilde{b}\|_1.
\]

\[\square\]

According to the analysis in Section 3.1, we let \((S^*, u^*)\) be the optimal point of the minimization problem

\[
\begin{align*}
& \min \frac{1}{2} \| S + I_u u^T - B \|_F^2 + \alpha \| S \|_1, \\
& \text{s.t. } S \geq 0, \ u \geq 0, \\
& \quad \sum(S(:, i)) + n \cdot u_i = \sum(B(:, i)), \ i = 1, 2, \ldots, n.
\end{align*}
\]

This problem can be split into \(n\) disjoint subproblems, thus \(S^*(:, i) + 1 \cdot u^*_i = \tilde{I}(B(:, i))\). Hence, we still use the notation \(\tilde{I}\) to represent the operator which maps the non-negative matrix \(B\) to another non-negative matrix \(S^* + 1_u(u^*)^T\), and it follows straightforwardly that \(\tilde{I} : \mathbb{R}^{n \times n}_+ \to \mathbb{R}^{n \times n}_+\) is also non-expansive. Then our proposed truncated power method can be rewritten into

\[
Y_{k+1} = \beta \tilde{I} (\tilde{I}(Y_k)) + (1 - \beta) G =: \tilde{I}(Y_k), \quad k = 0, 1, 2, \cdots, 
\]

(19)

see (8) and (9). Recall that the definition of the operator \(\tilde{I}\) is

\[
\tilde{I}(Y)(; i) = P_i \cdot Y(i, :)^T, \quad i = 1, 2, \ldots, n.
\]

Since all the column sums of each \(P_i\) equal 1, i.e., the operator 1-norm of each \(P_i\) is exactly 1, we have

\[
\left\| \tilde{I}(Y) \right\|_1 = \sum_{i = 1}^n \left\| \tilde{I}(Y)(; i) \right\|_1 = \sum_{i = 1}^n \| P_i \cdot Y(i, :)^T \|_1 \leq \sum_{i = 1}^n \| Y(i, :) \|_1 = \| Y \|_1,
\]

and the equality holds when \(Y\) is an all-one matrix. Thus the operator norm of the linear operator \(\tilde{I}\) exactly equals 1, i.e., \(\| \tilde{I} \|_1 = 1\), then for two arbitrary non-negative matrices \(Y\) and \(\bar{Y}\), it holds

\[
\left\| \tilde{I}(Y) - \tilde{I}(\bar{Y}) \right\|_1 = \beta \left\| \tilde{I}(\tilde{I}(Y)) - \tilde{I}(\tilde{I}(\bar{Y})) \right\|_1 \\
\leq \beta \left\| I(Y) - I(\bar{Y}) \right\|_1 \leq \beta \left\| Y - \bar{Y} \right\|_1.
\]

That is, the operator \(\tilde{I} : \mathbb{R}^{n \times n}_+ \to \mathbb{R}^{n \times n}_+\) is also non-expansive since \(\beta \in (0, 1]\), and furthermore it is a contraction mapping with the Lipschitz constant \(\beta\) when \(\beta \in (0, 1)\). Hence, by the Banach fixed point theorem, we know that when \(\beta < 1\) our algorithm \(Y_{k+1} = \tilde{I}(Y_k)\) converges to the unique fixed point in \(\mathbb{R}^{n \times n}_+\) with an arbitrary initial point \(Y_0 \in \mathbb{R}^{n \times n}_+\).

**Theorem 3.5.** When \(\beta \in (0, 1)\), the truncated power method (Algorithm 2) converges to a unique point \((S, u)\) for any initial point \((S_0, u_0)\), and

\[
\left\| (S + 1_u u^T) - (S_k + 1_u u^T_k) \right\|_1 \leq \beta^k \cdot \left\| (S + 1_u u^T) - (S_0 + 1_u u^T_0) \right\|_1.
\]

Furthermore, we shall store all the intermediate results in this compact form rather than only the final one. Otherwise, it is still too expensive and unwise to deal with the huge storage \(n^2\) in each step. The classical power method updates the next status by acting the transition probability matrix on the present status \(\tilde{I}(X_k)\). So all the matrices in the intermediate steps are probably full. Simple truncations or thresholds can be applied to introduce sparsity in each steps. However, this strategy may give a meaningless solution, i.e., the final result may not be a feasible stationary joint probability distribution. Therefore, we propose a truncated power method based on the sparse-add-background model, which constrains the intermediate results to be a probability distribution with the form \(S + 1_u u^T\) so that the iterations converge to a stationary joint probability distribution with the same form.
4. Numerical examples

In this section, we test the performance of the proposed truncated power method in Section 3, and compare it with the classical power method. The testing algorithms are conducted on a desktop machine with 4 GB RAM (3.24 GB usable) and 32 Intel® Core™ i5-2520M CPU of 2.50 GHz. The stopping criterion of the proposed truncated power method and the classical power method is that the relative error of the successive iterates \( \|X_k - X_{k-1}\|_1 \) is less than \( 1 \times 10^{-12} \).

**Example 4.1.** In the first experiment, we consider a sparse second-order Markov chain whose transition probability tensor is of size \( 4000 \times 4000 \times 4000 \) and of density 0.001%. Here the density refers to the number of non-zero entries over \( 4000^3 \) (the total number of entries). We would like to check the effect of \( \alpha \) (the parameter for controlling the sparsity of solution) on the convergence of the algorithm and the computed solution. We test for different values of \( \alpha \) and set to be 0.5/n^2, 1.5/n^2, 2/n^2, 2.5/n^2, 3/n^2. Here 1/n^2 can be considered to be a joint probability distribution of all the entries with equal probability. It is clear when \( \alpha \) is large, the sparsity of computed solution is large (i.e., more zero entries).

In Fig. 3, the results show that when \( \alpha \) increases, the number of iterations required for convergence is less and the sparsity of computed solution also increases. As for the comparison, we compare the computed solution \( X_p \) by the proposed method with the solution \( X_p \) by the power method which can be regarded as the true joint probability distribution of the given Markov chain. Our experiment results show that when \( \alpha \) increases (0.5/n^2, 1.0/n^2, 1.5/n^2, 2.0/n^2, 2.5/n^2 and 3.0/n^2) the error \( \|X_c - X_p\|_F \) also increases (2.561e−05, 4.886e−05, 4.954e−05, 5.027e−05, 5.028e−05, 5.029e−05) but the increase is not affected significantly.

**Example 4.2.** In the second experiment, we randomly construct some sparse third-order stochastic tensors \( P \) of size \( n \times n \times n \) (\( n = 2000, 4000, 6000, 8000, 10000 \)). Their densities are set to be 0.001%, 0.0005% or 0.0001%. The prior probability matrix \( G \) is set to be the same sparsity of \( P \). Both the classical power method and the proposed truncated power method are employed to solve the stationary joint probability distribution \( X \) from \( X = \beta P(X) + (1 - \beta) G \) for \( \beta = 1.00, 0.99, 0.98, 0.95 \).

We further utilize a server with 1536 GB RAM and 4 x Intel® Xeon® E7-4890 v2 CPU of 2.8 ~ 3.4 GHz to calculate the power method solution, and computed the distance between our computed solutions and the power method solution, where the Frobenius norm is used.

The computational results (the number of iterations required for convergence, the computational time required, and the distance between the computed solution and the power method solution) are summarized and displayed in Tables 1–3. Since the desktop computer has only 4 GB RAM, it is not sufficient to store the solution matrix by the power method. For those cases, we state in the table “out of memory”. In order to compute the error of the proposed algorithm, we utilize a server with 1536 GB RAM and 4 x Intel® Xeon® E7-4890 v2 CPU of 2.8 ~ 3.4 GHz to calculate the power method solution, and computed the distance between our computed solutions and the power method solution, where the Frobenius norm is used. From the tables, we conclude that the truncated power method can determine a stationary probability distribution from a sparse second-order Markov chain very effectively. When the number of states is large, there is not enough memory to store joint probability distributions for a classical power method. However, the proposed method can find solutions very efficiently. It is interesting to note for some examples that the proposed method even converges faster than the classical power method. For different testing values of \( \alpha, \beta \), and density numbers, the number of iterations required for convergence increases very slightly when the size \( n \) increases. When \( \alpha \) is large, the proposed method requires less number of iterations since there are more zero entries in the solution.

We also employ the case that the sparsity is 0.001% and \( \beta = 1 \) to show that the edge-to-edge formulation may not be suitable for these second-order Markov chains. Because of the treatment of dangling nodes in such sparse second-order Markov chains, there is at least one nonzero entry in each column \( P(:,j,k) \). Thus all the edges in the associated graph will
Nevertheless, the model is truncated for computational reason. Table 4 contains the comparison of the truncated power method with different $\alpha$ and the classical power method. Here the sparsities of $P$ and $G$ are $0.001\%$.

Table 4
The comparison of the truncated power method with different $\alpha$ and the classical power method. Here the sparsities of $P$ and $G$ are $0.005\%$.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Size</th>
<th>Power method</th>
<th>$\alpha = 1/n^2$</th>
<th>$\alpha = 2/n^2$</th>
<th>$\alpha = 3/n^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter.</td>
<td>Time (s)</td>
<td>Iter.</td>
<td>Time (s)</td>
<td>Error</td>
<td>Iter.</td>
</tr>
<tr>
<td>1.00</td>
<td>2000</td>
<td>12</td>
<td>2.38</td>
<td>8</td>
<td>3.31</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>9.90</td>
<td>9</td>
<td>15.39</td>
<td>4.89e−5</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>Out of memory</td>
<td>11</td>
<td>64.81</td>
<td>3.94e−5</td>
</tr>
<tr>
<td></td>
<td>8000</td>
<td>Out of memory</td>
<td>12</td>
<td>181.18</td>
<td>3.89e−5</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>Out of memory</td>
<td>13</td>
<td>473.52</td>
<td>2.99e−5</td>
</tr>
<tr>
<td>0.99</td>
<td>2000</td>
<td>12</td>
<td>2.39</td>
<td>9</td>
<td>4.02</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>9.76</td>
<td>11</td>
<td>22.83</td>
<td>2.05e−4</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>Out of memory</td>
<td>11</td>
<td>85.60</td>
<td>1.94e−4</td>
</tr>
<tr>
<td></td>
<td>8000</td>
<td>Out of memory</td>
<td>12</td>
<td>258.65</td>
<td>1.77e−4</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>Out of memory</td>
<td>13</td>
<td>686.12</td>
<td>1.64e−4</td>
</tr>
<tr>
<td>0.98</td>
<td>2000</td>
<td>13</td>
<td>2.62</td>
<td>10</td>
<td>4.50</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>10.51</td>
<td>11</td>
<td>22.95</td>
<td>5.78e−4</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>Out of memory</td>
<td>12</td>
<td>97.12</td>
<td>3.74e−4</td>
</tr>
<tr>
<td></td>
<td>8000</td>
<td>Out of memory</td>
<td>13</td>
<td>300.15</td>
<td>3.52e−4</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>Out of memory</td>
<td>13</td>
<td>771.52</td>
<td>3.21e−4</td>
</tr>
<tr>
<td>0.95</td>
<td>2000</td>
<td>13</td>
<td>2.56</td>
<td>10</td>
<td>4.47</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>10.54</td>
<td>11</td>
<td>23.09</td>
<td>1.29e−3</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>Out of memory</td>
<td>12</td>
<td>97.62</td>
<td>8.71e−4</td>
</tr>
<tr>
<td></td>
<td>8000</td>
<td>Out of memory</td>
<td>13</td>
<td>313.06</td>
<td>8.90e−4</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>Out of memory</td>
<td>14</td>
<td>886.65</td>
<td>8.14e−4</td>
</tr>
</tbody>
</table>

be involved. This implies that the number of edges is exactly $n^2$, and the size of the resulting solution is still $n$-by-$n$. The computational results are given in Table 4. When $n$ is large, the storage is still too large for computation. For $n = 2000, 4000$, the computational time of the node-to-node power method is about the same as that of the edge-to-edge power method. Nevertheless, the construction of edge-to-edge formulated matrix takes a lot of time.

Example 4.3. In this experiment, we test a real data set and demonstrate the usefulness of the proposed algorithm. The data set comes from 100,000 webpages from .GOV Web collection in 2002 TREC [24]. There are $n = 99988$ webpages and $m = 39255$ anchor terms. An $n \times n \times m$ tensor $A$ is formed by recording linkages among webpages, i.e.,

$$A(i, j, s) = \begin{cases} 1, & \text{if webpage } i \text{ links to webpage } j \text{ via anchor term } s, \\ 0, & \text{otherwise}. \end{cases}$$

W. Ding et al. / Applied Numerical Mathematics 125 (2018) 68–85
Table 3
The comparison of the truncated power method with different $\alpha$ and the classical power method. Here the sparsities of $\mathcal{P}$ and $\mathcal{G}$ are 0.0001%.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Size</th>
<th>Power method</th>
<th>$\alpha = 1/n^2$</th>
<th>$\alpha = 2/n^2$</th>
<th>$\alpha = 3/n^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iter.</td>
<td>Time (s)</td>
<td>Error</td>
<td>Iter.</td>
</tr>
<tr>
<td>1.00</td>
<td>2000</td>
<td>9</td>
<td>1.82</td>
<td>5</td>
<td>1.97</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>6</td>
<td>6.66</td>
<td>5</td>
<td>6.31</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>9</td>
<td>18.06</td>
<td>6</td>
<td>18.47</td>
</tr>
<tr>
<td>0.99</td>
<td>2000</td>
<td>10</td>
<td>1.95</td>
<td>6</td>
<td>2.37</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>10</td>
<td>7.38</td>
<td>7</td>
<td>7.80</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>10</td>
<td>20.31</td>
<td>7</td>
<td>22.63</td>
</tr>
<tr>
<td>0.98</td>
<td>2000</td>
<td>10</td>
<td>1.97</td>
<td>6</td>
<td>2.36</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>10</td>
<td>7.36</td>
<td>7</td>
<td>9.05</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>11</td>
<td>21.78</td>
<td>8</td>
<td>25.96</td>
</tr>
<tr>
<td>0.95</td>
<td>2000</td>
<td>10</td>
<td>1.98</td>
<td>6</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>10</td>
<td>7.38</td>
<td>7</td>
<td>9.02</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>11</td>
<td>21.68</td>
<td>8</td>
<td>25.61</td>
</tr>
<tr>
<td>0.90</td>
<td>2000</td>
<td>10</td>
<td>2.03</td>
<td>12</td>
<td>463.32 + 2.37</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>13</td>
<td>9.09</td>
<td>13</td>
<td>14134.86 + 10.73</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>10</td>
<td>16.84</td>
<td>6</td>
<td>11.42</td>
</tr>
</tbody>
</table>

Table 4
The sparsity of randomly generated $\mathcal{P}$ is 0.001%. The running time of the edge-to-edge formulation consists of the time for constructing the sparse $n^2$-by-$n^2$ matrix and the time for iterations.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Size</th>
<th>Power method</th>
<th>$\alpha = 1/n^2$</th>
<th>$\alpha = 2/n^2$</th>
<th>$\alpha = 3/n^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iter.</td>
<td>Time (s)</td>
<td>Iter.</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1.00</td>
<td>2000</td>
<td>12</td>
<td>2.03</td>
<td>12</td>
<td>463.32 + 2.37</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>13</td>
<td>9.09</td>
<td>13</td>
<td>14134.86 + 10.73</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>10</td>
<td>16.84</td>
<td>6</td>
<td>11.42</td>
</tr>
</tbody>
</table>

The sparsity of $\mathcal{A}$ is $1.2207 \times e^{-7}\%$. We construct a second-order Markov chain to model linkage among webpages via the same anchor term, i.e.,

$\text{webpage } k \xrightarrow{\text{via anchor term } s} \text{webpage } j \xrightarrow{\text{via anchor term } s} \text{webpage } i$.

To derive a two-step transition probability, we introduce a random variable $X$ taking values of webpage, and a random variable $Y$ taking values of anchor terms. Then we have the following calculation:

\[
\Pr(X_t = i \mid X_{t-1} = j, X_{t-2} = k) = \sum_{s=1}^{m} \Pr(Y = s, X_t = i \mid X_{t-1} = j, X_{t-2} = k)
\]

\[
= \sum_{s=1}^{m} \Pr(Y = s, X_t = i, X_{t-1} = j) \times \Pr(X_{t-1} = j, X_{t-2} = k)
\]

\[
= \sum_{s=1}^{m} \frac{\Pr(X_t = i, X_{t-2} = k \mid Y = s, X_{t-1} = j) \times \Pr(Y = s, X_{t-1} = j)}{\Pr(X_{t-1} = j, X_{t-2} = k)}
\]

\[
= \sum_{s=1}^{m} \frac{\Pr(X_t = i \mid Y = s, X_{t-1} = j) \times \Pr(X_{t-2} = k, Y = s, X_{t-1} = j)}{\Pr(X_{t-1} = j, X_{t-2} = k)}
\]
Fig. 4. The sparse component $S$ of the stationary joint probability distribution $X$.

The transition probabilities of a second-order Markov chain are given by

$$P(i, j, k) = \sum_{s=1}^{m} \left( \frac{A(j, i, s)}{\sum_{i=1}^{m} A(j, i, s)} \right) \cdot \left( \frac{A(k, j, s)}{\sum_{s=1}^{m} A(k, j, s)} \right).$$

(20)

It is clear that $\sum_{i=1}^{n} P(i, j, k) = 1$ for all $j$ and $k$.

In this example, we would like to illustrate the proposed truncated power method can provide the consistent ranking results of webpages. It is clear that the classical power method cannot be applied in the current desktop computing platform as the size of solution is 99988-by-99988. Using the classical power method would require about $10^5 \times 10^5 \times 8$ bytes $\approx 75$ GB of RAM to store the solution matrix, whereas our algorithm is able to run on a machine with fewer than 4 GB of RAM. We test $\beta = 1$, and there is no prior information on the selected webpages. We employ $\alpha = 1/n^2, 2/n^2, 3/n^2$, and rank the webpages according to the score of stationary joint probability distribution. In Fig. 4, the sparse component $S$ of stationary joint probability distribution is visualized. In Fig. 5, the background component $u$ of stationary joint probability distribution is displayed. We also apply a server with 1536 GB RAM and 4 x Intel® Xeon® E7-4890 v2 CPU of 2.8~3.4 GHz to calculate the power method solution for comparison.

The ranking score of the $j$-th webpage is given by the multilinear Pagerank vector, i.e., it can be calculated by $u_j + \sum_{i=1}^{n} s_{i,j}$ from the joint probability distribution. The top-ten webpages are listed in Table 5. For different values of $\alpha$, we obtain consistent ranking results from the method. The first two webpage numbers are the same, and there are nine webpages appearing in the top-10 lists by the power method and the truncated power method with different values of $\alpha$. 
Fig. 5. The background part \( u \) of the stationary joint probability distribution \( X \).

The webpage numbered “36701” in the power method solution is used, and does not appear in the other three lists. The webpage numbered “29493” is replaced in the lists of the truncated power method.

Next we set the prior information in ranking webpages. We would like to search and rank webpages relevant to the \( j_0 \)-th webpage. In this case, we choose \( \beta = 0.95 \) and set
Table 5
The top-ten webpage numbers. (\(\alpha = 0\) stands for the classical power method.)

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9063</td>
<td>18281</td>
<td>35984</td>
<td>56276</td>
<td>28454</td>
<td>26369</td>
<td>41219</td>
<td>54330</td>
<td>36701</td>
<td>40593</td>
</tr>
<tr>
<td>(1/n^2)</td>
<td>9063</td>
<td>18281</td>
<td>28454</td>
<td>41219</td>
<td>35984</td>
<td>56276</td>
<td>40593</td>
<td>29493</td>
<td>26369</td>
<td>54330</td>
</tr>
<tr>
<td>(2/n^2)</td>
<td>9063</td>
<td>18281</td>
<td>28454</td>
<td>41219</td>
<td>56276</td>
<td>35984</td>
<td>40593</td>
<td>29493</td>
<td>26369</td>
<td>54330</td>
</tr>
<tr>
<td>(3/n^2)</td>
<td>9063</td>
<td>18281</td>
<td>28454</td>
<td>41219</td>
<td>56276</td>
<td>35984</td>
<td>40593</td>
<td>29493</td>
<td>26369</td>
<td>54330</td>
</tr>
</tbody>
</table>

Table 6
The top-ten webpages relevant to the webpage number \(j_0 = 28454\). (\(\alpha = 0\) stands for the classical power method.)

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28454</td>
<td>26369</td>
<td>9063</td>
<td>18281</td>
<td>54330</td>
<td>41219</td>
<td>56276</td>
<td>35984</td>
<td>36701</td>
<td>40593</td>
</tr>
<tr>
<td>(1/n^2)</td>
<td>28454</td>
<td>26369</td>
<td>9063</td>
<td>18281</td>
<td>41219</td>
<td>54330</td>
<td>56276</td>
<td>35984</td>
<td>40593</td>
<td>29493</td>
</tr>
<tr>
<td>(2/n^2)</td>
<td>28454</td>
<td>26369</td>
<td>9063</td>
<td>18281</td>
<td>41219</td>
<td>54330</td>
<td>56276</td>
<td>35984</td>
<td>40593</td>
<td>29493</td>
</tr>
<tr>
<td>(3/n^2)</td>
<td>28454</td>
<td>26369</td>
<td>9063</td>
<td>18281</td>
<td>41219</td>
<td>54330</td>
<td>35984</td>
<td>56276</td>
<td>40593</td>
<td>29493</td>
</tr>
</tbody>
</table>

Fig. 6. The sparse component \(S\) of the stationary joint probability distribution \(X\).

\[ g_{i,j} = \begin{cases} 
1/n, & \text{if } j = j_0, \\
0, & \text{otherwise}. 
\end{cases} \]

The modified second-order Markov chain \(X = \beta P(X) + (1 - \beta)G\) can be applied to find the webpages which are mostly related to the \(j_0\)-th webpage. In the test, we consider the truncated power method with \(\alpha = 1/n^2, 2/n^2, 3/n^2\), and also use the server mentioned above to compute the power method solution for comparison. The top-ten webpages for the power method and the truncated power method with different \(\alpha\) are listed in Table 6 according to their ranking scores. The sparse
component $S$ is visualized in Fig. 6, and the background component $u$ is displayed in Fig. 7. Again we obtain consistent ranking results from the method for different values of $\alpha$. The first four webpage numbers are the same, and there are nine webpages appearing in the top-10 lists by the power method and the truncated power method with different values of $\alpha$. The webpage numbered “36701” in the power method solution is used, and does not appear in the other three lists. The webpage numbered “29493” is replaced in the lists for the truncated power method.
5. Concluding remarks

As a summary, we have proposed and studied an efficient algorithm for finding stationary joint probability distribution of sparse second-order Markov chains. The proposed iterative method can store and compute large-scale stationary joint probabilities very efficiently. Our idea is to approximate such stationary probabilities of sparse Markov chains by a sparse component vector and a rank-one background vector. The convergence of the proposed iterative method is established. Numerical examples are also presented to demonstrate the effectiveness of the proposed method compared with the classical power method. A real data set for ranking webpages is used to illustrate the proposed method can provide a good approximation for ranking webpages for large-scale data mining applications.

On the other hand, Wu et al. [26] proposed to formulate second-order Markov chains by using edge-to-edge transition probabilities. We can employ their approach to compute stationary probability distributions of sparse second-order Markov chains. The main idea is to construct the corresponding edge-to-edge formulated matrix according to the sparsity structure and dangling nodes structure (see Section 2.2). Since the number of edges is about $n^2$ where $n$ is the number of nodes, future research work includes how to design an approximation scheme for computing these stationary probability distributions.

In general, it is interesting to study how to find stationary joint probability distribution of sparse high-order Markov chains by using our proposed technique.

Acknowledgements

We would like to thank the editor, Professor Zdzislaw Jackiewicz, and the referees for their detailed and helpful comments.

References