Received: 22 July 2019 | Revised: 18 August 2020 | Accepted: 16 September 2020

DOI: 10.1002/nla.2340

RESEARCH ARTICLE

WILEY

A method for computing the Perron root for primitive matrices

Doulaye Dembélé

IGBMC, CNRS UMR7104 - INSERM U1258 - University of Strasbourg, Strasbourg, France

Correspondence
Doulaye Dembélé, Institut de Génétique et de Biologie Moléculaire et Cellulaire (IGBMC), CNRS UMR7104 - INSERM U1258 - University of Strasbourg, 1 rue Laurent Fries, 67400 Illkirch-Graffenstaden, Strasbourg, France.
Email: doulaye@igbmc.fr

Summary

Following the Perron theorem, the spectral radius of a primitive matrix is a simple eigenvalue. It is shown that for a primitive matrix A, there is a positive rank one matrix X such that $B = A \circ X$, where $\circ$ denotes the Hadamard product of matrices, and such that the row (column) sums of matrix B are the same and equal to the Perron root. An iterative algorithm is presented to obtain matrix B without an explicit knowledge of X. The convergence rate of this algorithm is similar to that of the power method but it uses less computational load. A byproduct of the proposed algorithm is a new method for calculating the first eigenvector.

KEYWORDS

Markov chain, Perron root, primitive matrix, stochastic matrix

1 | INTRODUCTION

Given a nonnegative matrix, the problem of computing the first eigenvalue and eigenvector is considered in this paper. For two matrices $A = (a_{ij})$ and $B = (b_{ij})$ with the same number of rows and columns, their Hadamard product is a matrix of elementwise products:

$$A \circ B = (a_{ij} b_{ij}).$$

For scalars α and β:

$$\alpha A \circ \beta B = \alpha\beta (A \circ B).$$

If A and B are rank one matrices, that is, $A = \mathbf{u}\mathbf{v}^T$ and $B = \mathbf{x}\mathbf{y}^T$ then

$$A \circ B = (\mathbf{u}\mathbf{v}^T) \circ (\mathbf{x}\mathbf{y}^T) = (\mathbf{u} \circ \mathbf{x})(\mathbf{v} \circ \mathbf{y})^T. \tag{1}$$

Many properties for Hadamard product are given in References 1 and chapter 5 of Reference 2.

If $A = (a_{ij}) \in \mathbb{R}^{n \times n}$, then A is called *positive* if $a_{ij} > 0$, and *nonnegative* if $a_{ij} \geq 0$. Perron[3] showed that the spectral radius of a positive matrix A is a simple eigenvalue (see p. 667 of Reference 4) that dominates all other eigenvalues in modulus. This eigenvalue, denoted $\rho(A)$, is called the *Perron root* and the associated normalized positive vector is called the *Perron vector*. Nonnegative matrices are frequently encountered in real life applications.[5,6] Frobenius[7] extended Perron's work on positive matrices to nonnegative matrices. The spectral radius of a nonnegative matrix A is positive if it is irreducible,

that is, $(I_n + A)^{n-1}$ is a positive matrix (see p. 534 of Reference 8 and p. 672 of Reference 4). The dominant eigenvalue of an irreducible matrix is unique if it is primitive (see p. 540 of Reference 8 and p. 674 of Reference 4). A nonnegative matrix is primitive if A^m is a positive matrix for some non-null m (see p. 540 of Reference 8 and p. 678 of Reference 4). Wielandt[9] showed that a nonnegative matrix A of order n is primitive if A^{n^2-2n+2} is a positive matrix (see p. 543 of Reference 8). To verify primitivity of a nonnegative matrix using Frobenius or Wielandt formula leads to huge calculations especially when n is high. It is shown in (see p. 544 of Reference 8) that only some power calculations of the matrix are necessary.

This paper is on the calculation of the Perron root. The power method is generally used to obtain the eigenvalue with the maximum modulus and associated eigenvector (see p. 545 of Reference 8, p. 330 of Reference 10, and p. 533 of Reference 4). The convergence rate of the power method depends on the ratio of the modulus of the second eigenvalue to the first (see p. 330 of Reference 10 and p. 533 of Reference 4). More iterations will be required when the modulus of the second highest eigenvalue is close to that of the first. Here, an iterative algorithm is proposed for calculating the Perron root for primitive matrices. This algorithm is based on successive improvement of bounds for the Perron root. There are many research works on localization of the Perron root for nonnegative matrices.[11-15] Frobenius carried out the following bounds (see p. 521 of Reference 8):

$$\min_{i=1,\dots,n}\{r_i(A)\} \le \rho(A) \le \max_{i=1,\dots,n}\{r_i(A)\} \tag{2}$$

$$\min_{j=1,\dots,n}\{c_j(A)\} \le \rho(A) \le \max_{j=1,\dots,n}\{c_j(A)\}, \tag{3}$$

where $r_i(A) = \sum_{j=1}^{n} a_{ij}$ and $c_j(A) = \sum_{i=1}^{n} a_{ij}$ are the row and column sums of A, respectively. In (2) and (3), equalities occur when $\rho(A)$ is equal to the row or column sums. The column sums of the matrix in (4) are both equal to 3.

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 3 & 0 & 3 \\ 0 & 2 & 0 \end{pmatrix}. \tag{4}$$

The matrix A in (4) is imprimitive and its eigenvalues are: 3, -3 and 0. This example shows that equality in (2) or (3) can occur for an imprimitive matrix.

Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ a vector with only positive values, $x_i > 0$, and $D_{\mathbf{x}}$ a diagonal matrix formed with $\mathbf{x}$. The matrix B defined by:

$$B = D_{\mathbf{x}}^{-1} A D_{\mathbf{x}}, \tag{5}$$

is diagonally similar to A,[16] and we have:

Lemma 1. *The matrices A and B in (5) have the same eigenvalues, and*

(a) if A is irreducible, then B is irreducible,
(b) if A is primitive, then B is primitive,

Proof. (a) if A is irreducible then $(I_n + A)^{n-1}$ is a positive matrix. From (5), we have:

$$I_n + B = I_n + D_{\mathbf{x}}^{-1} A D_{\mathbf{x}} = D_{\mathbf{x}}^{-1}(I_n + A) D_{\mathbf{x}}$$
$$(I_n + B)^{n-1} = D_{\mathbf{x}}^{-1}(I_n + A)^{n-1} D_{\mathbf{x}},$$

$D_{\mathbf{x}}$ has only positive values and $(I_n + A)^{n-1}$ is a positive matrix. The matrix $(I_n + B)^{n-1}$ is then positive that implies irreducibility of the matrix B.

(b) if A is a primitive matrix then there exists an integer m such that A^m is positive. Using (5) we have $B^m = D_{\mathbf{x}}^{-1} A^m D_{\mathbf{x}}$, that implies B^m is positive and the result follows. ∎

Using an improvement of bounds in (2) and (3) by Minc,[17] relation (5) and the uniqueness of eigenvalue with a maximum modulus for a primitive matrix, an iterative algorithm is proposed to obtaining the Perron root.

2 | METHODS

Lemma 2. *Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ a nonnegative matrix. If matrix A has a row (column) with only zero entries, then A cannot be a primitive matrix.*

Proof. See an exercise on irreducible matrices in (see p. 522 of Reference 8). ∎

For a primitive matrix, from the Lemma 2 and relations (2) and (3), we have the following two observations. The minimum value of the row (column) sums for a primitive matrix is greater than zero. The maximum value of the row (column) sums for a primitive matrix is greater than or equal to the Perron root.

Let us note $D_{\mathbf{r}}$ and $D_{\mathbf{c}}$ diagonal matrices formed with the row sums $\mathbf{r} = (r_1(A), r_2(A), \ldots, r_n(A))$ and the column sums $\mathbf{c} = (c_1(A), c_2(A), \ldots, c_n(A))$ of A, respectively. The Frobenius bounds (2) and (3) have been improved by Minc (see p. 27 of Reference 17):

$$\min_{i=1,\ldots,n}\{r_i(D_{\mathbf{r}}^{-1}AD_{\mathbf{r}})\} \le \rho(A) \le \max_{i=1,\ldots,n}\{r_i(D_{\mathbf{r}}^{-1}AD_{\mathbf{r}})\} \tag{6}$$

$$\min_{j=1,\ldots,n}\{c_j(D_{\mathbf{c}}^{-1}AD_{\mathbf{c}})\} \le \rho(A) \le \max_{j=1,\ldots,n}\{c_j(D_{\mathbf{c}}^{-1}AD_{\mathbf{c}})\}. \tag{7}$$

In (6) and (7), equalities hold when the row or column sums are the same and correspond to the Perron root. Using the row sums relation, (6) allows to write:

$$D_{\mathbf{r}}^{-1}AD_{\mathbf{r}} = \begin{pmatrix} a_{11} & \frac{r_2}{r_1}a_{12} & \frac{r_3}{r_1}a_{13} & \ldots & \frac{r_n}{r_1}a_{1n} \\ \frac{r_1}{r_2}a_{21} & a_{22} & \frac{r_3}{r_2}a_{23} & \ldots & \frac{r_n}{r_2}a_{2n} \\ \frac{r_1}{r_3}a_{31} & \frac{r_2}{r_3}a_{32} & a_{33} & \ldots & \frac{r_n}{r_3}a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{r_1}{r_n}a_{n1} & \frac{r_2}{r_n}a_{n2} & \frac{r_3}{r_n}a_{n3} & \ldots & a_{nn} \end{pmatrix} \tag{8}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{pmatrix} \circ \begin{pmatrix} 1 & \frac{r_2}{r_1} & \frac{r_3}{r_1} & \ldots & \frac{r_n}{r_1} \\ \frac{r_1}{r_2} & 1 & \frac{r_3}{r_2} & \ldots & \frac{r_n}{r_2} \\ \frac{r_1}{r_3} & \frac{r_2}{r_3} & 1 & \ldots & \frac{r_n}{r_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{r_1}{r_n} & \frac{r_2}{r_n} & \frac{r_3}{r_n} & \ldots & 1 \end{pmatrix}$$

$$= A \circ X, \tag{9}$$

where X is a positive matrix formed with:

$$x_{ij} = \frac{r_j(A)}{r_i(A)}; i,j = 1, 2 \ldots, n. \tag{10}$$

The unicity of the Perron root for a primitive matrix and (9) suggest that the components of the matrix X can be chosen to have the same row sums for $A \circ X$. For a second order nonnegative matrix ($n = 2$), we have:

$$A \circ X = \begin{pmatrix} a_{11} & a_{12}x \\ a_{21}/x & a_{22} \end{pmatrix}, \tag{11}$$

where $x = r_2(A)/r_1(A)$, $r_1(A) = a_{11} + a_{12}$ and $r_2(A) = a_{21} + a_{22}$.

If the row sums in (11) are the same and equal to S, an expression can be obtained for the parameter x:

$$x = \frac{S - a_{11}}{a_{12}}; \frac{1}{x} = \frac{S - a_{22}}{a_{21}}. \tag{12}$$

To have a value for x, a_{12} and a_{21} should be nonzero. Relation (12) allows to have a second-order equation which resolution leads to a value for S:

$$S^2 - (a_{11} - a_{22})S + a_{11}a_{22} - a_{12}a_{21} = 0. \tag{13}$$

The solution of (13) with the maximum modulus is:

$$S = (a_{11} + a_{22} + \sqrt{(a_{11} - a_{22})^2 + 4a_{12}a_{21}})/2. \tag{14}$$

A second-order nonnegative matrix A is primitive if a_{12}, and a_{21} are both nonzero, on the one hand. On the other hand, at least a_{11} or a_{22} should be nonzero. Hence, for a second-order primitive matrix, explicit expressions can be obtained for matrix X (parameter x) and the Perron root, (14). However, a direct search for components of the matrix X in (9) becomes difficult when $n > 2$.

Lemma 3. *Let $A = (a_{ij})$ a square matrix of order n, $\mathbf{y} = \alpha\mathbf{x}$ a vector where α is a nonzero scalar and $\mathbf{x}$ is an eigenvector associated with the eigenvalue λ of A. If all components of $\mathbf{x}$ are nonzero, the row sums of the matrix $D_{\mathbf{y}}^{-1}AD_{\mathbf{y}}$ are the same and equal to the eigenvalue λ of A.*

A similar result is obtained using A^T or the column sums.

Proof. Using the definition of the eigenvalue, the component i of $A\mathbf{x} = \lambda\mathbf{x}$ is:

$$\sum_{j=1}^{n} a_{ij}x_j = \lambda x_i. \tag{15}$$

The component i of the row sums of the matrix $D_{\mathbf{y}}^{-1}AD_{\mathbf{y}}$ is:

$$r_i = \frac{1}{x_i}\left(\sum_{j=1}^{n} a_{ij}x_j\right). \tag{16}$$

By replacing the right-hand expression of (15) in (16) the result follows. ∎

Compared to the Frobenius bounds (2) and (3), the bounds in (6) and (7) are based on a modification of the initial matrix. This process can be reiterated to further sharpen the bounds. From the unicity of the Perron root for a primitive matrix, a repetitive improvement of the Minc bounds will lead to equalities of the row (column) sums.

An essential result of this paper is the following.

Theorem 1. *An iterative algorithm based on a successive improvement of the Minc bounds is convergent for a primitive matrix.*

Proof. At iteration t, the row sum vectors associated with matrices $A^{(t)}$ and $A^{(t-1)}$ are $\mathbf{r}^{(t)}(A^{(t)})$ and $\mathbf{r}^{(t-1)}(A^{(t-1)})$, respectively. From the Minc theorem (see p. 27 of Reference 17), we have:

$$\min_i \{r_i^{(t)}(A^{(t)})\} \geq \min_i \{r_i^{(t-1)}(A^{(t-1)})\}. \tag{17}$$

$$\max_i \{r_i^{(t)}(A^{(t)})\} \leq \max_i \{r_i^{(t-1)}(A^{(t-1)})\}. \tag{18}$$

Let us define two decreasing sequences as follows:

$$\xi^{(t)} = \rho(A) - \min_i \{r_i^{(t)}(A^{(t)})\}. \tag{19}$$

$$\zeta^{(t)} = \max_i \{r_i^{(t)}(A^{(t)})\} - \rho(A). \tag{20}$$

From (18) and (20), we have

$$\begin{aligned}\zeta^{(t)} &= \max_i \{r_i^{(t)}(A)\} - \rho(A) \leq \max_i \{r_i^{(t-1)}(A)\} - \rho(A) = \zeta^{(t-1)}\\ \zeta^{(t)} &\leq c^{(t)}\zeta^{(t-1)},\end{aligned} \tag{21}$$

where $0 < c^{(t)} \leq 1$. Let $\zeta^{(0)}$ denotes the initial value obtained using (20). From relation (21) we have:

$$\zeta^{(t)} \leq \left(\prod_{i=1}^{t} c^{(i)}\right)\zeta^{(0)}. \tag{22}$$

Since $c^{(i)}$, $i = 1, 2, \ldots, t$, are positive numbers not all equal to 1, we have

$$\lim_{t\to\infty}\zeta^{(t)} = 0.$$

A similar reasoning using (17) and (19) leads to $\lim_{t\to\infty}\xi^{(t)} = 0$. Hence, when the number of iteration goes to infinity, relations (19) and (20) show that the row sums obtained with the algorithm converges to the Perron root. Referring to Lemma 3, it is like a vector with only ones is used to obtaining to Perron root. ∎

Relation (22) can be used to estimate the minimum number of iterations required by the algorithm before convergence when an error level α is set. Assuming that $E(c^{(i)}) = c$ is the mean of the $c^{(i)}$ coefficients, relation (22) becomes $\zeta^{(t)} = c^t\zeta^{(0)}$ and we have

$$c^t \leq \alpha \Rightarrow t \geq \frac{\log(\alpha)}{\log(c)}.$$

The main result of this paper is the following.

Theorem 2. *Let A be a primitive matrix of order n. There exists a positive rank one matrix X of the form*

$$X = \begin{pmatrix} 1 & x_2/x_1 & x_3/x_1 & \ldots & x_n/x_1\\ x_1/x_2 & 1 & x_3/x_2 & \ldots & x_n/x_2\\ x_1/x_3 & x_2/x_3 & 1 & \ldots & x_n/x_3\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ x_1/x_n & x_2/x_n & x_3/x_n & \ldots & 1 \end{pmatrix}, \tag{23}$$

such that the matrix $B = A \circ X$ is similar to A. In addition, the row (column) sums of B are the same and equal to the Perron root of A.

Proof of Theorem 2. See the exercise given below. ∎

2.1 | Exercise

By denoting $A^{(0)}$, $\mathbf{r}^{(0)}$ the initial matrix and its row sums, respectively:

- use the relation (6) to obtain the following one:

$$A^{(t)} = D^{-1}_{\mathbf{r}^{(t-1)}}A^{(t-1)}D_{\mathbf{r}^{(t-1)}} \text{ at iteration : } t = 1, 2, \ldots, \tag{24}$$

- use relation (24) and the notation in (9) to show that:

$$A^{(t)} = A \circ X = B, \tag{25}$$

where $X = X^{(0)} \circ X^{(1)} \circ \ldots \circ X^{(t-2)} \circ X^{(t-1)}$

- Let $\mathbf{x}$ and $\mathbf{y}$ be the first column and the first row, respectively, of the matrix in relation (23). Verify that:

$$X = \mathbf{x}\mathbf{y}^T. \tag{26}$$

  Deduce the rank of the matrix X.
- Show that the matrix B defined in (25) in similar to A.

Corollary 1. *The vector* $\mathbf{y}$ *allowing to obtain the matrix X in (26) is in the space spanned by the Perron vector of a primitive matrix A.*

Proof. Use (8), (9) and Lemma 3. ∎

Corollary 2. *For primitive matrices, the convergence rate of the proposed algorithm is similar to that of the power method and depends on the magnitude of the second highest eigenvalue.*

Proof. The matrix A can be write as the sum of rank one matrices using its eigenvalues and eigenvectors:

$$A = U\Lambda U^{-1} = U\Lambda V^T. \tag{27}$$

$$= \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots \lambda_n \mathbf{u}_n \mathbf{v}_n^T. \tag{28}$$

From (28), (26) and (1) we have:

$$A \circ X = \lambda_1 (\mathbf{u}_1 \circ \mathbf{x})(\mathbf{v}_1 \circ \mathbf{y})^T + \lambda_2 (\mathbf{u}_2 \circ \mathbf{x})(\mathbf{v}_2 \circ \mathbf{y})^T + \ldots \lambda_n \mathbf{u}_n \circ \mathbf{x})(\mathbf{v}_n \circ \mathbf{y})^T. \tag{29}$$

Using (27) and (28), an expression for power k of matrix A is

$$A^k = \lambda_1^k \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2^k \mathbf{u}_2 \mathbf{v}_2^T + \ldots \lambda_n^k \mathbf{u}_n \mathbf{v}_n^T.$$

This expression allows to have another one similar to (29). Then, the row sums at the first step of the algorithm using A^k are:

$$\begin{aligned}(A^k \circ X^{(0)})\mathbf{1} &= \lambda_1^k \delta_1 \mathbf{z}_1 + \lambda_2^k \delta_2 \mathbf{z}_2 + \ldots + \lambda_n^k \delta_n \mathbf{z}_n \\ &= \lambda_1^k \delta_1 \left(\mathbf{z}_1 + \frac{\delta_2}{\delta_1} \left(\frac{\lambda_2}{\lambda_1} \right)^k \mathbf{z}_2 + \ldots + \frac{\delta_n}{\delta_1} \left(\frac{\lambda_n}{\lambda_1} \right)^k \mathbf{z}_n \right),\end{aligned}$$

where $\delta_i = (\mathbf{v}_i \circ \mathbf{y}^{(0)})^T \mathbf{1}$, $\mathbf{z}_i = (\mathbf{u}_i \circ \mathbf{x}^{(0)})$, $i = 1, 2, \ldots, n$,

$\mathbf{x}^{(0)} = \left(1, \frac{r_1^{(0)}}{r_2^{(0)}}, \frac{r_1^{(0)}}{r_3^{(0)}}, \ldots, \frac{r_1^{(0)}}{r_n^{(0)}} \right)^T$ and $\mathbf{y}^{(0)} = \left(1, \frac{r_2^{(0)}}{r_1^{(0)}}, \frac{r_3^{(0)}}{r_1^{(0)}}, \ldots, \frac{r_n^{(0)}}{r_1^{(0)}} \right)^T$

From Corollary 1, vector $(A \circ X)\mathbf{1} \in span\{(A^k \circ X^{(0)})\mathbf{1}\}$ then,

$$dist(span(A \circ X)\mathbf{1}, span(\mathbf{z}_1)) = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right).$$

∎

2.2 | Algorithm and implementation

From (9) and (10), the components of matrix $A^{(t)}$ and its row sums are:

$$a_{ij}^{(t)} = \frac{r_j^{(t-1)}(A^{(t-1)})}{r_i^{(t-1)}(A^{(t-1)})} a_{ij}^{(t-1)}; i, j = 1, 2, \ldots, n. \tag{30}$$

$$r_i^{(t)}(A^{(t)}) = \sum_{j=1}^{n} a_{ij}^{(t)}; i = 1, 2, \ldots, n. \tag{31}$$

Relations (30) and (31) allow to obtain an algorithm for computing the matrix B in Theorem 2. A convergence test is based on the difference between the maximum and the minimum values of the row or column sums, that is, the range value.

$$error = \max_{i=1,\ldots,n} \{r_i^{(t)}(A^{(t)})\} - \min_{i=1,\ldots,n} \{r_i^{(t)}(A^{(t)})\} \tag{32}$$

$$error = \max_{j=1,\ldots,n} \{c_j^{(t)}(A^{(t)})\} - \min_{j=1,\ldots,n} \{c_j^{(t)}(A^{(t)})\}.$$

Another convergence test can be based on the examination of the minimum and maximum row sum values, that is, by using the decreasing sequences $\xi^{(t)}$ and $\zeta^{(t)}$ defined in Theorem 1.

2.2.1 | Algorithm A (using row sums): Perron root only

1. Initialization
   - set: $t \leftarrow 0$, $a_{ij}^{(t)} \leftarrow a_{ij}$, calculate the row sums using (31)
   - set stopping rules: eps (the acceptable error), maxIter (the maximum number of iterations), compute the initial error value using (32)
2. while (error > eps and $t <$ maxIter)
   - update matrix: (30)
   - calculate row sums: (31)
   - compute error: (32)
   - increase iteration number: $t \leftarrow t + 1$

Remark 1. As mentioned, Theorem 2 is also valid using column sums. In the implementation, one can compute the row and column sums and perform the next step using the sum where the initial error is the lowest.

Remark 2. From an iteration to the next, the error should decrease by an amount that depends on the convergence rate. Otherwise, we must stop the algorithm because the matrix does not seem to be primitive. This observation can be used as an indirect test for primitivity of a matrix.

Indeed, if the spectral radius is not simple (case of an irreducible imprimitive matrix) there may be at least two eigenvectors associated with eigenvalues having the same modulus.

Remark 3. With the proposed algorithm, the diagonal entries of A remain unchanged, only the off-diagonal components are modified. The Gerschgorin discs (see p. 388 of Reference 8) associated with a matrix allow to illustrate this.

Let us consider an example:

$$A = \begin{pmatrix} 3 & \sqrt{3} \\ \sqrt{3} & 1 \end{pmatrix}; A^{(1)} = \begin{pmatrix} 3 & 1 \\ 3 & 1 \end{pmatrix}.$$

The off-diagonal components of A are modified in such a way all discs cross the same highest point on the x-axis (4 in this example). To show this, let us write:

$$A = D_A + P,$$

where D_A is a diagonal matrix formed with the diagonal elements of A, and P is matrix A where the diagonal elements are set to zero. From (25), we have:

$$B = D_A + P \circ X.$$

Hence, the row sums of matrix B are given by:

$$r_i(B) = r_i(D_A) + r_i(P \circ X) = a_{ii} + \mathbf{p}_{i.}^T \mathbf{x}_i,$$

where $\mathbf{p}_{i.}$ and $\mathbf{x}_{i.}$ are vectors formed with row i of matrices P and X, respectively.

Remark 4. Compared to the power method, there is no initial vector to set. The results obtained using the power method are the Perron root and the associated eigenvalue. Only the Perron root is obtained using this algorithm. However, since the row (column) sums of the matrix B in (25) are the same, a vector $\mathbf{1}$ formed with only ones is an eigenvector of B (B^T).

$$B\mathbf{1} = \rho(A)\mathbf{1} = (A \circ X)\mathbf{1}.$$

Remark 5. At each iteration, the total numbers of multiplications and additions of the matrix-vector multiplication by the power method are equal to the total number of operations for the proposed algorithm. Hence, using the power method, the additional arithmetic operations used for calculating the eigenvector and the eigenvalue are extra computational load compared to the proposed algorithm.

2.2.2 | Algorithm B (using row sums): Perron root and vector

Instead of the Algorithm A (2.1.1), another one can consist in searching for a vector $\mathbf{y}$ similar to the Perron vector. For this purpose, the matrix B is initially equal to A and the vector $\mathbf{y}$ is set to $\mathbf{1}$, then, the row sums of B are calculated. At iteration t, a vector $\mathbf{y}^{(t)}$ is formed using row sums and the matrix B is updated. At the convergence, we should have $\mathbf{y}^{(t)} \approx \mathbf{1}$.

1. Initialization
   - set: $t \leftarrow 0$, $B \leftarrow A$, $\mathbf{y} \leftarrow \mathbf{1}$ and compute row sums $r_i^{(0)}$ of B,
   - calculate initial error: $\max(r_i^{(0)}) - \min(r_i^{(0)})$.
   - set stopping rules: *eps* and *maxIter* (see Algorithm A).
2. while (error > eps and $t <$ maxIter)
   - form $\mathbf{y}^{(t)}$ using row sums $r_i^{(t)}$
   - update $\mathbf{y}$: $\mathbf{y} \circ \mathbf{y}^{(t)}$
   - form $\mathbf{x}\,(1/\mathbf{y})$ and update matrix B: $A \circ (\mathbf{x} * \mathbf{y}^T)$
   - compute error: $\max |\mathbf{y}^{(t)} - \mathbf{1}|$
   - increase iteration number: $t \leftarrow t + 1$
   - calculate row sums $r_i^{(t)}$ of B

The codes A0.1 and A0.2 in the Appendix are the R implementation of algorithm B. An input matrix should be square, nonnegative and each row sum should be greater than zero. For code A0.1, the convergence test is based on an examination of the vector $\mathbf{y}$. For the code A0.2, the convergence test is based on an examination of the minimum and the maximum row sum values of the decreasing sequences $\xi^{(t)}$ and $\zeta^{(t)}$ defined in Theorem 1. The convergence always occurs with the code A0.2 while this is not the case when using the code A0.1 for an imprimitive matrix.

2.2.3 | Application to row-stochastic matrices

Let us consider Markov chains modeling a dynamic system with finite discrete n states. At each time t, this system is in one state. When the system is in state i, the move to state j or to stay in state i is controlled by the probability $p_{ij} \geq 0$, $\sum_{j=1}^{n} p_{ij} = 1$. The probabilities are organized in a transition matrix P which is nonnegative. Interestingly, the power k of matrix P is also a transition matrix which element p_{ij} corresponds to the probability to move from state i to state j at time $t + k$. The Markov chains are used in: (a) biological sequences analysis,[18,19] (b) internet traffic or search engines,[20-22] (c) … The state occupied by the Markov chain process at time $t + k$ depends on the nature of transition matrix P: reducible/irreducible, imprimitive/primitive. In some applications, the observed transition matrix is modified to be primitive.[20] For this

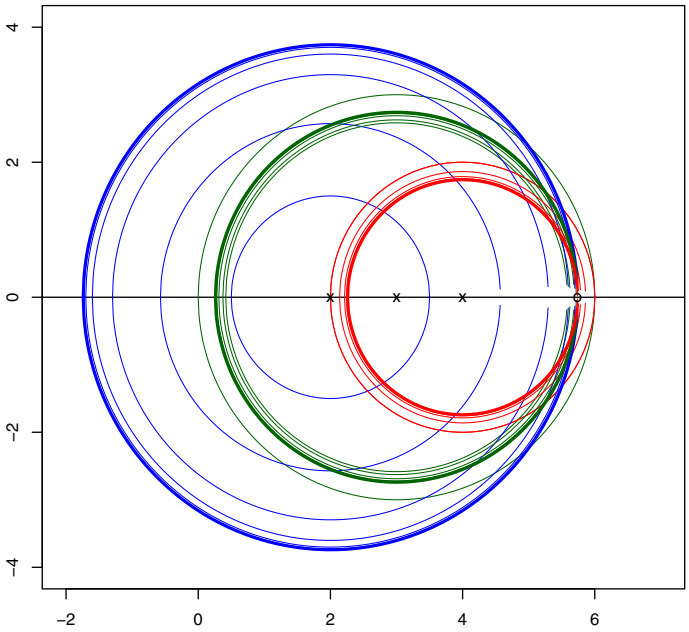

FIGURE 1 Gerschgorin's discs: thin plot lines for A and $A^{(t)}$ before convergence, bold plot lines for $A^{(t)}$ at convergence

particular case, the power of the modified matrix $\hat{P}$ converges to a matrix formed with the same vector $\mathbf{u}$ verifying: $\mathbf{u}^T\hat{P} = \mathbf{u}^T$. The vector $\mathbf{u}$ is the left-hand vector of the matrix $\hat{P}$. In the search engines based on the Markov chains, the components of the vector $\mathbf{u}$ allow to hierarchize the information. Applying Algorithm B (2.1.2) to the transposed of the row-stochastic (modified transition) matrix lead to a vector $\mathbf{y}$ which components are then normalized in a way that their sum is 1. This normalized vector corresponds to $\mathbf{u}$.

3 | RESULTS AND CONCLUSIONS

All calculations were performed on the same computer (a laptop equipped with i7-66600U processor, 16 GB of RAM, under Microsoft Windows 10) and R version 3.6.2. The error level was arbitrarily set to 1.0E-8.

3.1 | Results 1

The first example is:

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 0.5 & 3 & 2 \\ 1 & 2 & 4 \end{pmatrix}.$$

The range values for the row and the column sums are 4 and 2.5, respectively. The algorithm is performed using the column sums. Figure 1 presents the Gerschgorin discs for all iterations. The Perron root for this example is 5.739952, the proposed algorithm and the power method require 17 and 19 iterations, respectively. Figure 1 shows that the major modifications of the off-diagonal elements of the matrix A are done during the first five iterations.

For all of the tests performed, the algorithm proposed and the power method have close number of iterations. Worse results, in term of the number of iterations, were obtained using a tridiagonal matrix. Let $T(n; c, a, b)$ a tridiagonal matrix of order n, where a is the value for the diagonal components, b is the value for the upper diagonal components and c is the value for the under diagonal components. An explicit expression relating eigenvalues of matrix T is available:[23]

$$\lambda_k = a + 2\sqrt{bc}\cos\frac{k\pi}{n+1}.$$

TABLE 1 Illustrative example with four genes and their GO terms

$gene_1$	GO:000000a, GO:000000b
$gene_2$	GO:000000c, GO:000000d, GO:000000e, GO:000000f
$gene_3$	GO:000000c, GO:000000a, GO:000000b
$gene_4$	GO:000000g, GO:000000h, GO:000000a

The ratio λ_2/λ_1 for T is near 1 when n is high. For $n=50$, $a=3$, $b=2$ and $c=1$ the first two eigenvalues of matrix T are: 5.823063 and 5.806989. The proposed algorithm took 5,174 (using the R code given in Appendix) iterations to calculate the Perron root. The power method did better by using 5,159 iterations.

3.2 | Results 2

High throughput biological technologies allow to analyze the functions of genes at the genome level. Genes associated with a given disease can be revealed using these technologies.[24-27] When a set of interesting genes is identified, researcher generally like to have further knowledge for some or all these genes. For this purpose, additional experiments are performed. In addition, data mining analyses are often done to access to biological information already available in databases. The gene ontology (GO) database provides curated and structured knowledge for genes functions and for many species.[28,29] The GO information is organized under three main categories: biological process, molecular function and cellular component. Identifiers (ID) are linked to annotations which are supported by experimental data from research papers. From a set of selected genes, the GO database is usually used to performing enrichment analysis. That consists in searching for the GO terms which are significantly associated with the genes in the set.[30] Here, we propose to hierarchize the GO identifiers associated with genes of a subset, hence providing supplementary information. Consider an academic example of four genes and their fictitious GO IDs.

For this example, there are eight unique identifiers: GO:0-0a, GO:0-0b, GO:0-0c, GO:0-0d, GO:0-0e, GO:0-0f, GO:0-0g, GO:0-0h. We consider a graph were IDs are the nodes and genes allow to make the edges. For each gene, a matrix of order equal to the number of unique identifiers is create. In this matrix, all identifiers (nodes) associated with a gene allow to form a graph. Each node of this graph is linked to the other nodes and to itself. A value 1 is assigned to each link. Summing the matrices from all genes allow to obtain a (symmetric) score matrix which is finally standardized for leading to a column stochastic matrix. Matrices P and A in (33) are the score and standardized matrices of the example in Table 1.

$$P=\begin{pmatrix} 3 & 2 & 1 & 0 & 0 & 0 & 1 & 1\\ 2 & 2 & 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 2 & 1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1\\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}; A=\begin{pmatrix} 3/8 & 2/5 & 1/7 & 0 & 0 & 0 & 1/3 & 1/3\\ 2/8 & 2/5 & 1/7 & 0 & 0 & 0 & 0 & 0\\ 1/8 & 1/5 & 2/7 & 1/4 & 1/4 & 1/4 & 0 & 0\\ 0 & 0 & 1/7 & 1/4 & 1/4 & 1/4 & 0 & 0\\ 0 & 0 & 1/7 & 1/4 & 1/4 & 1/4 & 0 & 0\\ 0 & 0 & 1/7 & 1/4 & 1/4 & 1/4 & 0 & 0\\ 1/8 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3\\ 1/8 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 \end{pmatrix}. \quad (33)$$

The normalized eigenvector of the matrix A in (33) is: (0.21053, 0.13158, 0.18421, 0.10526, 0.10526, 0.10526, 0.07895, 0.07895). The term GO:000000a has the highest value followed by GO:000000c, then GO:000000b. That matches our intuition regarding data in Table 1.

Real microarray data available on the Gene Expression Omnibus website,[31] under the accession number GSE51649 were used. In this dataset, wild-type and knockout mouse samples were compared in order to select genes involved in a neuronal disease, Fragile X Syndrome.[32] Only ontologies regarding the biological process (BP) were used. Selected GO

TABLE 2 Top 15 GO terms

	5% error selection		all data	
	799 order matrix		5,612 order matrix	
	357 iterations		107 iterations	
	GO term	**Weight**	**GO term**	**Weight**
1	GO:0043065 (184)	0.01701	GO:0045449 (1802)	0.01578
2	GO:0030335 (85)	0.01632	GO:0006350 (1753)	0.01485
3	GO:0007399 (303)	0.01590	GO:0007165 (1979)	0.01434
4	GO:0031175 (91)	0.01560	GO:0008150 (11170)	0.01263
5	GO:0001934 (99)	0.01524	GO:0006810 (1761)	0.01144
6	GO:0008284 (286)	0.01503	GO:0007275 (948)	0.00995
7	GO:0030324 (120)	0.01485	GO:0006355 (808)	0.00936
8	GO:0051384 (57)	0.01485	GO:0008152 (1978)	0.00838
9	GO:0045663 (39)	0.01473	GO:0045944 (415)	0.00732
10	GO:0010552 (133)	0.01459	GO:0008284 (286)	0.00653
11	GO:0010976 (60)	0.01422	GO:0030154 (579)	0.00650
12	GO:0006935 (114)	0.01415	GO:0007186 (1640)	0.00639
13	GO:0000902 (116)	0.01402	GO:0006811 (612)	0.00546
14	GO:0001654 (61)	0.01402	GO:0006915 (511)	0.00504
15	GO:0002053 (67)	0.01402	GO:0006468 (588)	0.00472

terms (GO subset/slim[28]) are available for 22,849 genes of this dataset. The genes having only the category name term (GO:0008150 / biological process) were delete and it remains 15,515 genes. These genes have 5,612 unique GO IDs. The fold change rank ordering method,[27] was used to select a set of genes showing a significant difference of expression between the wild-type samples and the knockout samples. Using an error level equal to 5%, 446 genes were selected. The number of unique GO biological process IDs associated with these genes is 799. The score matrix and its standardized form were computed, as described above, for the set of 446 genes for all the 15,515 genes. The normalized eigenvector was computed for the two stochastic matrices. Table 2 shows the top-15 GO biological process IDs obtained. The column *weight* in Table 2 is used for the components of the Perron vector. Algorithm B was used for these results and the number of iterations required is given. For each ID in this Table 2, we indicate in the parenthesis the number of times it is observed in the list of 15,515 genes. The category name term (GO:0008150) is present in nearly 72% of all the 15,515 genes. The top-15 GO terms of the selected genes seems to be specific to the study, since there are less present in the dataset compared to the top-15 of the entire dataset. Only one GO term (GO:0008284) is shared by both top-15 lists.

It is the first time, to authors' knowledge, the GO terms are used in this way. The results in Table 2 are encouraging and may be useful in a system biology study. This preliminary work will be extended to all GO categories and for many others high throughput biological data.

3.3 | Conclusions

Except the cases where the modulus of the second eigenvalue is near to that of the first, the algorithm proposed converges after few iterations, especially when the first eigenvalue is largely dominant. The biological data example shows that the method proposed can be used for high dimensional problems.

With a convergence rate similar to that of the classic power method, the proposed algorithm for computing the Perron root is computationally less demanding. But, it applies to only primitive matrices. However, it can be used as a low-cost primitivity test compared to the matrix power calculations involved in the Frobenius and Wielandt tests.

ACKNOWLEDGEMENTS

This work was supported by funds from CNRS, INSERM and University of Strasbourg. Author is grateful to the referee and the Editor for the valuable comments and suggestions.

ORCID

Doulaye Dembélé https://orcid.org/0000-0003-3879-6940

REFERENCES

1. Styan GPH. Hadamard products and multivariate statistical analysis. Linear Algebra Appl. 1973;6:217–240.
2. Horn RR, Johnson CR. Topics in matrix analysis. Cambridge, MA: Cambridge University Press, 1991.
3. Perron O. Zur theorie der matrices. Math Ann. 1907;64(2):248–263.
4. Meyer C. Matrix analysis and applied linear algebra. Philadelphia, PA: SIAM, 2000.
5. Brualdi RA, Ryser HJ. Combinatorial matrix theory. Encyclopedia of mathematics and its applications. Vol 39. Cambridge, MA: Cambridge University Press, 1991.
6. Berman A, Plemmons RJ. Nonnegative matrices in the mathematical sciences. Philadelphia, PA: SIAM, 1994.
7. Frobenius FG. Ueber Matrizen aus nicht negativen Elementen. Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften. 1912;26:456–477.
8. Horn RR, Johnson CR. Matrix analysis. 2nd ed. Cambridge, MA: Cambridge University Press, 2019.
9. Wielandt H. Unzerlegbare, nicht negative Matrizen. Math Z. 1950;52(1):642–648.
10. Golub GH, Loan CFV. Matrix computations. 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.
11. Kolotilina LY. Lower bounds for the perron root of a nonnegative matrix. Linear Algebra Appl. 1993;180:133–151.
12. Liu SL. Bounds for the greatest characteristic root of a nonnegative matrix. Linear Algebra Appl. 1996;239:151–160.
13. Duan X, Zhou B. Sharp bounds on the spectral radius of a nonnegative matrix. Linear Algebra Appl. 2013;439:2961–2970.
14. Xing R, Zhou B. Sharp bounds on the spectral radius of a nonnegative matrices. Linear Algebra Appl. 2014;449:194–209.
15. Liao P. Bounds for the perron root of nonnegative matrices and spectral radius of iteration matrices. Linear Algebra Appl. 2017;530:253–265.
16. Elsner L, Johnson CR, Dias da Silva J. The perron root of a weighted geometric mean of nonnegative matrices. Linear Multilinear Algebra. 1988;24(1):1–13.
17. Minc H. Nonnegative matrices. New York, NY: Wiley, 1988.
18. Durbin R, Eddy SR, Krogh A, Mitchison G. Biological sequences analysis. Cambridge, MA: Cambridge University Press, 2002.
19. Ewens WJ, Grant GR. Statistical methods in bioinformatics: An introduction. New-York, NY: Springer-Verlag, 2002.
20. Langville AN, Meyer CD. A survey of eigenvector methods for web information retrival. SIAM Rev. 2005;47(1):135–161.
21. Wu G, Wei Y. A Power-Arnoldi algorithm for computing PageRank. Numer Linear Algebra Appl. 2007;14:521–546.
22. Wen C, Huand TZ, Shen ZL. A note on the two-step matrix splitting iteration for computing PageRank. J Comput Appl Math. 2017;315:87–97.
23. Noschese S, Pasquini L, Reichel L. Tridiagonal Toeplitz matrices: Properties and novel applications. Numer Linear Algebra Appl. 2013;20(2):302–326.
24. Schulze A, Downward J. Navigating gene expression using microarrays - a technology review. Nat Cell Biol. 2001;3:E190–E195.
25. Pop M, Salzberg SL. Bioinformatics challenges of new sequencing technology. Trends Genet. 2007;24(3):142–149.
26. van Dijk EL, Auger H, Jaszczyszyn Y, Thermes C. Ten years of next-generation sequencing technology. Trends Genet. 2014;30(9):418–426.
27. Dembélé D. Analysis of high-throughput biological data using their rank values. Stat Methods Med Res. 2019;28(8):2276–2291.
28. The Gene Ontology Consortium. The gene ontology resource: 20 years and still GOing strong. Nucleic Acids Res. 2019;47:D330–D338.
29. Thomas PD, Hill DP, Mi H, et al. Gene ontology causal activity modeling (GO-CAM) moves beyond GO annotations to structured descriptions of biological functions and systems. Nat Genet. 2019;51:1429–1433.
30. Munoz-Torres M, Carbon S. Get GO! retrieving GO data using AmiGO, QuickGO, API, files, and tools. In: Dessimoz C, Skunca N, editors. The gene ontology handbook. Methods in molecular biology. Volume 1446. New York, NY: Humana Press, 2017; p. 149–160.
31. Barrett T, Wilhite SE, Ledoux P, et al. NCBI GEO: Archive for functional genomics data sets-update. Nucl Acids Res. 2013;41(D1):D991–D995.
32. Tabet R, Moutine E, Becker JA, et al. Fragile X mental retardation protein (FMRP) controls diacylglycerol kinase activity in neurons. Proc Nat Acad Sci. 2016;113(26):E3619–E3628.

How to cite this article: Dembélé D. A method for computing the Perron root for primitive matrices. *Numer Linear Algebra Appl.* 2020;e2340. https://doi.org/10.1002/nla.2340

APPENDIX . A R CODE USING ROW SUMS

Algorithm B1

```
## This function computes iteratively the Perron root
## and the eigenvector of matrix A using row sums
#
# A: nonnegative square matrix (all row sums are > 0)
# tol: error level used (stopping criterion)
# maxIter: maximum number of iterations (stooping criterion)
#
# Returned
# B: matrix which has the same row sums (B = A o X)
# pfr: minimum and maximum the row sums, that defines to the Perron root
# y: vector (leading to have the eigenvector and matrix X)
# iter: number of iterations performed
calcPRb <- function(A, tol=1.0e-8, maxIter=50) {
 n <- nrow(A); m <- ncol(A)
 ri <- apply(A, 1, sum)
 ko <- (sum(A<0) || (min(ri)==0))
 if ((n != m) || (ko)) {
 stop("calcPRb(): for nonnegative primitive matrices")
 }
 err.t <- max(ri) - min(ri)
 err <- err.y <- err.t
 y <- rep(1,n)
 iter <- 0; B <- A
 while ((min(err.y, err.t) > tol) && (iter < maxIter)) {
    yt <- ri/ri[1]; y <- y*yt
    err.y <- max(abs(yt-1))
    B <- A * ((1/y)
    iter <- iter + 1
    ri <- apply(B, 1, sum)
    err.t <- max(ri) - min(ri); err <- c(err, err.t)
 }
 pfr <- c(min(ri), max(ri))
 list(B=B, pfr=pfr, y=y, iter=iter, err=err)
}
```

Algorithm B2

```
## This function computes iteratively the Perron root
## and the eigenvector of matrix A using row sums
#
# A: nonnegative square matrix (all row sums are > 0)
# tol: error level used (stopping criterion)
# maxIter: maximum number of iterations (stooping criterion)
#
# Returned
# B: matrix which has the same row sums (B = A o X)
# pfr: minimum and maximum the row sums, that defines to the Perron root
# y: vector (leading to have the eigenvector and matrix X)
# iter: number of iterations performed
```

```
# rmin: sequences with minimum row sum values for iterations
# rmax: sequences with maximum row sum values for iterations
calcPRc <- function(A, tol=1.0e-8, maxIter=50) {
 n <- nrow(A); m <- ncol(A)
 ri <- apply(A, 1, sum)
 ko <- (sum(A<0) || (min(ri)==0))
 if ((n != m) || (ko)) {
 stop("calcPRc(): for nonnegative primitive matrices")
 }
 y <- rep(1,n)
 iter <- 1; B <- A
 rmin <- c(); erMin <- rmin[iter] <- min(ri)
 rmax <- c(); erMax <- rmax[iter] <- max(ri)
 erIter <- ((erMin > tol) || (erMax > tol))
 while (erIter \&\& (iter < maxIter)) {
    yt <- ri/ri[1]; y <- y*yt
    B <- A * ((1/y) \* t(y))
    ri <- apply(B, 1, sum)
    iter <- iter + 1
    rmin[iter] <- ri.min <- min(ri)
    rmax[iter] <- ri.max <- max(ri)
    erMin <- rmin[iter] - rmin[iter-1]
    erMax <- rmax[iter-1] - rmax[iter]
    erIter <- ((erMin > tol) || (erMax > tol))
 }
 pfr <- c(ri.min, ri.max)
 list(B=B, pfr=pfr, y=y, iter=iter-1, rmin=rmin, rmax=rmax)
}
```