Theory of Stochastic Processes
Vol. 29 (45), no. 2, 2025, pp. 127–138

AZ-EDDINE ZAKRAD AND ABDELAZIZ NASROALLAH

GENERALIZED PARTITIONING ALGORITHM FOR COMPUTING STEADY-STATE PROBABILITY VECTORS OF FINITE MARKOV CHAINS WITH COMPARISON TO THE CFTP ALGORITHM

Dedicated to all researchers in stochastic processes

In this paper we propose a generalization of the basic partitioning algorithm proposed by T.J. Sheskin for computing steady state probability vector of a finite Markov chain. This algorithm generates an exact solution for steady state probabilities for any finite, irreducible Markov chain. Theoretically there is no imposed limit on the size of the Markov chain for which steady state probabilities can be obtained, but in practice, the method will produce round off errors. we propose to generalize the partitioning technique to cover all possible variants of the Sheskin algorithm. Our proposal, besides being a mathematical curiosity, it gives answer to several possible modifications (variations) suggested, by the author of this algorithm. In the numerical part we compared our generalization with standard CFTP algorithm.

1. Introduction

Amongst the most used stochastic models to study the dynamics of random phenomena are the Markovian models. These models are generally appreciated by scientists because they are relatively simple and interpretable. In addition they use important mathematical tools both theoretically and practically. A major goal of a Markov model is to determine the steady state probabilities governing the stationary regime of the studied phenomena. For Markov chains that it is possible to compute the steady state solution, the standard algorithm is to solve a system of simultaneous linear equations. If this system is not sparse, iterative algorithms can be used to approximate the solution. One of the important algorithms that allow to perfectly simulate from the stationary law of a Markov chain, whose state space is finite, is the celebrate Coupling From The past Algorithm, called Standard CFTP, founded by Prop and Wilson in 1996 [16]. The CFTP algorithm has been adapted to the case of the simulation of a Markov chain whose state space is continuous by Murdoch and Green [14]. Their obtained algorithm is named Multi-Gamma-Coupler (MGC). For more details on CFTP, MGC, and their variants, the interested reader can consult [4, 14, 16]. In [21], Nasroallah and Zakrad combined the CFTP and MGC algorithms and produced an algorithm, called Mixed-CFTP, that can simulate the stationary distribution of a Markov chain whose state spaces $E = C \cup D$, is composed of a continuous part C and a finite part D such that $C \cap D = \emptyset$. The solution of simultaneous linear equations is not practical, however, for Markov chains that contain more than several hundred states. Such large Markov chains arise in applications such as performance evaluation of computer systems in operation research field Bolch et al. (2006), Dai (1996) and Mahévas and Rubino (2001). Exact steady state solutions to large Markov chains generally exploit special structure such as sparseness, lumpability

2010 *Mathematics Subject Classification.* 60J10 ; 68W15.
Key words and phrases. Markov chain; steady state probability; algorithm; CFTP algorithm; perfect simulation.
Corresponding author: Az-Eddine Zakrad.

Barr and Thomas (1977), or compression Sheskin (1976). Hillier and Boling (1967) have proposed approximate solutions, as well.

Our paper concerns a proposition of variants in the partitioning algorithm, initially proposed by Sheskin (1985). This algorithm generates an exact solution for steady state probabilities for any finite, irreducible Markov chain. Theoretically there is no imposed limit on the size of the Markov chain for which steady state probabilities can be obtained, but in practice, the method will produce round off errors.

The partitioning algorithm contains a backward pass for matrix reduction followed by a forward pass for vector enlargement. As explained in Sheskin (1985), the matrix reduction routine successively partitions, based on a result in Kemeny and Snell (1960), the transition probability matrix for the Markov chain. Each partition of a transition matrix produces four sub-matrices: a square matrix, a column vector, a single element, and a row vector. When combined, the four sub-matrices create a reduced matrix that also represents an irreducible Markov chain. The number of states in the reduced matrix is one less than the number in the original matrix.

Each reduced matrix is partitioned in the same manner as the original matrix, creating a sequence of successively smaller reduced matrices. This backward pass terminates when the last reduced matrix has two states. A forward pass begins with the last column vector from the final two-state reduced matrix and express the steady state of each state recursively as a function of the steady state of states computed before, until the last state. The steady state of the original Markov chain is obtained at the end by solving a normalizing equation.

In this paper, we propose to generalize the partitioning technique to cover all possible variants of the Sheskin algorithm. And in the numerical part, we compare our algorithm with the standard CFTP algorithm.

The paper is organized as follows: In section 2, we give a brief description of the standard CFTP algorithms. In section 3, we present a theorem, due to Kemeny and Snell (1960), that motivates the partitioning algorithm. In section 4, we propose our generalized partitioning algorithm that covers all possible variants. In section 5 we apply the algorithm to the case of a simple partition that we call 2×2-partition and for which we specify the computational cost. In section 6 We show how to use the general partition algorithm to solve systems in linear algebra. As basic numerical example, we use the 2×2-partition algorithm to compute the steady state probability of a Markov chain, and we compare with the standard CFTP algorithm in section 7. A conclusion is given in section 7, followed by a list of references.

2. Standard CFTP algorithm

The CFTP algorithm allows to simulate from the exact steady-state probability of a homogeneous and ergodic Markov chain $X = (X_t)_{t\in\mathbb{Z}}$ defined on a finite or bounded state space E with transition kernel $(K(x/y))_{x,y\in E}$, where $K(x/y) = I\!P(X_{t+1} = x/X_t = y)$. If E is finite (with size $|E|$), the standard CFTP runs $|E|$ copies of the studied Markov chain. If $|E| = +\infty$ and E is bounded, for standard CFTP we need the update function (the function governing the evolution of the Markov chain) to be monotonous. This allows to run only two copies (instead of $|E|$ copies) of the studied Markov chain (one from the maximum state and the other from the minimum state for a partial order in E).

Let $\phi : E\times]0,1[\longrightarrow E$, be the update function of X, defined, for example, by $\phi(y,u) = F_y^-(u)$, where for $(y,u) \in E\times]0,1[$, $F^-(u/y) = inf\{\ t\ ,\ F(t/y) \geq u\}$ is the generalized inverse of the probability distribution $K(./y)$ on E. Remark here that the update function form of a Markov chain is not necessarily unique.

For a partially ordered space $(E, \preceq)$, the update function ϕ is said to be monotonic if

$$\forall\, y_1, y_2 \in E \text{ such that } y_1 \preceq y_2\ , \text{ we have } \phi(y_1, w) \preceq \phi(y_2, w),\ \ \forall w \in]0,1[.$$

The aim of the CFTP algorithm is to find the coalescence point of all copies of trajectories of X, starting at time $-T$ in the past (each copy starts from a different state). By varying $T \in \mathbb{N}^*$ in an increasing sens, CFTP looks for a sate $x \in E$ such that

$$\Phi^0_{-T}(y, \mathbf{w}^{-1}_{-T}) := \phi(\phi(...\phi(y, w_{-T}), ..., w_{-2}), w_{-1}) = x,\ \ \forall y \in E,$$

where $\mathbf{w}^{-1}_{-T} = (w_{-T}, w_{-T+1}, ..., w_{-1})$ is a sequence of independent and identically distributed (i.i.d.) realizations of uniform randoms on $]0,1[$. In the sequel we denote $u \sim \mathcal{U}_{]0,1[}$ for uniform random realization u on $]0,1[$ and $v \perp w$ for two independent random realizations v and w.

Standard CFTP algorithm for finite E

Let $E = \{x_1, ..., x_N\}$ be the state space of the Markov chain X with update function ϕ. The main steps of the standard CFTP are the following:

Step1: Initialization : $t = 1$,

Step2: Generate $w_{-t} \sim \mathcal{U}_{]0,1[}$ and keep $\mathbf{w}^{-1}_{-t+1} = (w_{-t+1}, \ldots, w_{-1})$, where $w_s \perp w_{s'}$ for $s \neq s'$.

Step3: If $\exists x \in E$ such that $\Phi^0_{-t}(x_i, \mathbf{w}^{-1}_{-t}) = x$, $\forall i \in \{1, \cdots, N\}$, then return x and **Stop**.

else $t = t + 1$ and go to **Step2**.

Monotone CFTP algorithm for finite E

In the case where the state space E is bounded (with $\underline{y}$ and $\overline{y}$ its minimum and maximum states relatively to a partial order) and the update function ϕ is monotonic, the CFTP becomes Monotone CFTP (M-CFTP) by replacing **Step3** by the following **M-Step3**:

M-Step3:If $\exists x \in E$ such that $\Phi^0_{-t}(\underline{y}, \mathbf{w}^{-1}_{-t}) = \Phi^0_{-t}(\overline{y}, \mathbf{w}^{-1}_{-t}) = x$, then return x and **Stop**.

else $t = t + 1$ and go to **Step2**.

The following theorem insure that CFTP algorithm terminates with probability one and gives a realization from the exact steady-state of the studied Markov chain.

Theorem 2.1. [16]. *Let $X = (X_t)_{t\in\mathbb{Z}}$ be a homogeneous and ergodic Markov chain on a finite state space. Then*

(i) *the CFTP algorithm terminates with probability one.*

(ii) *the state turned by CFTP at time zero is a perfect realization of the steady-state of X.*

3. Partitioning method

Consider a finite, irreducible Markov chain with r states, and divide these states arbitrarily into two subsets. The first subset contains the first s states and the second one contains the last $r - s$ states. Now partition the transition matrix $P = (p_{i,j})_{1\leq i,j\leq r}$, of our Markov chain, into four sub-matrices called T, W, R, and Q, as shown below.

$$P = \left[\begin{array}{c|c} T & W \\ \hline R & Q \end{array}\right] \tag{1}$$

where T is $s \times s$ matrix and Q is $(r - s) \times (r - s)$. We compute a reduced matrix, P_s, which represents an irreducible, $s-$state Markov chain, according to the formula

$$P_s = T + W(I - Q)^{-1}R, \tag{2}$$

where I is the $(r - s) \times (r - s)$ identity matrix. The steady state probability vectors for P and P_s are related by the following theorem (see Kemeny and Snell (1960)).

Theorem 3.1. *Let* $\mathbf{a}_1^r = [a_1, a_2, \cdots, a_s, a_{s+1}, \cdots, a_r]$ *be the steady state probability vector for* P. *Then* $\mathbf{a}_1^s = \frac{1}{c_s}[a_1, a_2, \cdots, a_s]$ *is the steady state probability of* P_s, *where* $c_s = \sum_{i=1}^{s} a_i$.

Remark 3.1. The partitioning algorithm also applies, in an extended form, to a Markov process. If A represents the transition rate matrix (the generator) for an irreducible Markov process, and A is partitioned as in (1), then (2) is modified to compute a reduced transition rate matrix, A_s, by formula $A_s = T - WQ^{-1}R$.

The basic partitioning algorithm of Sheskin (1985) was given such that, at each iteration, Q is a scalar. The generalization we are going to propose lets Q to take all possible square matrix block (for example at stage t, $Q = A^{(t)}_{k_t,k_t}$). As for Sheskin algorithm, our proposal is composed in two phases (retrograde and forward).

4. General partitioning method

At first we consider P as a matrix P_t composed in t^2 blocks: $P_t = (A^{(t)}_{k_i,k_j})_{1\le i,j\le t}$, where

$$A^{(t)}_{k_i,k_j} = \begin{bmatrix} p^{(t)}_{k_1^{i-1}+1,k_1^{j-1}+1} & p^{(t)}_{k_1^{i-1}+1,k_1^{j-1}+2} & \cdots & p^{(t)}_{k_1^{i-1}+1,k_1^{j-1}+k_j} \\ p^{(t)}_{k_1^{i-1}+2,k_1^{j-1}+1} & p^{(t)}_{k_1^{i-1}+2,k_1^{j-1}+2} & \cdots & p^{(t)}_{k_1^{i-1}+2,k_1^{j-1}+k_j} \\ \cdots & \cdots & \cdots & \cdots \\ p^{(t)}_{k_1^{i-1}+k_i,k_1^{j-1}+1} & p^{(t)}_{k_1^{i-1}+k_i,k_1^{j-1}+2} & \cdots & p^{(t)}_{k_1^{i-1}+k_i,k_1^{j-1}+k_j} \end{bmatrix},$$

with $k_i^j = k_i + k_{i+1} + \cdots + k_j, (i \le j)$ and $k_i^0 = 0$.

The retrograde phase begins with the dynamic partition

$$P_t = \left[\begin{array}{c|c} T_t & W_t \\ \hline R_t & A^{(t)}_{k_t,k_t} \end{array}\right], \tag{3}$$

where $W_t = (A^{(t)}_{k_1,k_t}, A^{(t)}_{k_2,k_t}, \cdots, A^{(t)}_{k_{t-1},k_t})'$, with the power " $'$ " is the transposition operator, and $R_t = (A^{(t)}_{k_t,k_1}, A^{(t)}_{k_t,k_2}, \cdots, A^{(t)}_{k_t,k_{t-1}})$.

The reduced matrix P_{t-1} is then computed using $P_{t-1} = T_t + W_t[I_{k_t} - A^{(t)}_{k_t,k_t}]^{-1}R_t$, where I_{k_t} is the $k_t \times k_t$ identity matrix.

If P_{t-1} is the final desired $(k_1 \times k_1)$ matrix, we stop the retrograde phase, else we restat the dynamic partition 3 with matrix P_{t-1} (instead of P_t) whose dimension is reduced by k_t with respect to P_t. We continue the retrograde reduction phase until the finale reduced $(k_1 \times k_1)$ matrix P_1. We compute the stationary distribution $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_{k_1})$ of P_1, and we use it in the forward enlargement phase to calculate recursively the stationary probabilities, when adding progressively successive blocks, by multiplying by constant $L_2, L_3, \ldots, L_t$.

4.1. **Determination of constant** $L_m, m = 2, \ldots, t$**.** At first, we compute $(\mathbf{a}_1^{k_1})' := (a_1, \ldots, a_{k_1})$ such that $\mathbf{a}_1^{k_1} = (\mathbf{a}_1^{k_1})'P_1$, where $P_1 = A^{(1)}_{k_1,k_1}$. The relation between $\mathbf{a}_1^{k_1}$ and λ is

$$\mathbf{a}_1^{k_1} = \bar{a}\lambda, \quad \text{where} \quad \bar{a} = \sum_{i=1}^{k_1} a_i \tag{4}$$

At the second step, we compute $(\mathbf{a}_1^{k_1^2})' := \left(a_1, \ldots, a_{k_1}, a_{k_1+1}, \ldots, a_{k_1^2}\right)$ such that $\mathbf{a}_1^{k_1^2} = (\mathbf{a}_1^{k_1^2})' P_2$, where

$$P_2 = \left[\begin{array}{c|c} A^{(2)}_{k_1,k_1} & A^{(2)}_{k_1,k_2} \\ \hline A^{(2)}_{k_2,k_1} & A^{(2)}_{k_2,k_2} \end{array}\right], \tag{5}$$

So

$$\mathbf{a}^{k_1^2}_{k_1+1} = (\mathbf{a}_1^{k_1})' A^{(2)}_{k_1,k_2} + (\mathbf{a}^{k_1^2}_{k_1+1})' A^{(2)}_{k_2,k_2}$$

and consequently

$$\mathbf{a}^{k_1^2}_{k_1+1} = (\mathbf{a}_1^{k_1})' A^{(2)}_{k_1,k_2} \left[I_{k_2} - A^{(2)}_{k_2,k_2}\right]^{-1} = (\mathbf{a}_1^{k_1})' L_2,$$

where $L_2 = A^{(2)}_{k_1,k_2} \left[I_{k_2} - A^{(2)}_{k_2,k_2}\right]^{-1}$.

At the step t, we compute $(\mathbf{a}_1^{k_1^t})' := \left((\mathbf{a}_1^{k_1})', (\mathbf{a}_{k_1+1}^{k_2})', \ldots, (\mathbf{a}_{k_1^{t-1}+1}^{k_1^t})'\right)$ such that $\mathbf{a}_1^{k_1^t} = (\mathbf{a}_1^{k_1^t})' P_t$. We obtain

$$\mathbf{a}^{k_1^t}_{k_1^{t-1}+1} = \sum_{j=0}^{t-1} (\mathbf{a}^{k_1^{j+1}}_{k_1^j+1})' A^{(t)}_{k_{j+1},k_t}$$

which imply

$$\mathbf{a}^{k_1^{(t)}}_{k_1^{t-1}+1} = (\mathbf{a}_1^{k_1})' \left(A^{(t)}_{k_1,k_t} + \sum_{j=2}^{t-1} L_j A^{(t)}_{k_j,k_t}\right) \left[I_{k_t} - A^{(t)}_{k_t,k_t}\right]^{-1},$$

where

$$L_m = \left(A^{(m)}_{k_1,k_m} + \sum_{j=2}^{m-1} L_j A^{(m)}_{k_j,k_m}\right) \left[I_{k_m} - A^{(m)}_{k_m,k_m}\right]^{-1}, \quad \text{for} \quad m = 3, 4, \ldots, t$$

4.2. **Deduction of the global steady state probability.** Summing the following equations

$$\mathbf{a}^{k_1^m}_{k_1^{m-1}+1} = (\mathbf{a}_1^{k_1})' L_m, \quad m = 2, 3, \ldots, t,$$

we obtain

$$\sum_{m=2}^{t} \sum_{j=1}^{k_m} a_{k_1^{m-1}+j} = a_1 L^1 + a_2 L^2 + \ldots + a_{k_1} L^t, \tag{6}$$

where $L^i = L_2^i + L_3^i + \ldots + L_t^i$ and $L_j^i = L_j^{i,1} + L_j^{i,2} + \ldots, + L_j^{i,k_j}$, with $L_j^{i,k}$ is the (i,k)-entry of the matrix L_j.

Now from equation (6), we obtain

$$1 - \sum_{i=1}^{k_1} a_i = a_1 L^1 + a_2 L^2 + \ldots + a_{k_1} L^t \tag{7}$$

Using (4) and (7), we obtain

$$a_i = \frac{\lambda_i}{1 + \sum_{j=1}^{k_1} \lambda_j L^j}, \quad i = 1, 2, \ldots, k_1.$$

In order to digest the complicated notations used in the general partition method, we propose the following simple partition method (2×2-Partition method) using 2×2 matrices.

5. 2×2-Partition method

In this section we present the case where $k_s = 2$ for each iteration s. For simplicity and without loss of the generality, we assume that r is even. If r is odd, the backward steps end with a 3×3 matrix for which the steady state must be computed.

5.1. **Backward steps. Step 0.** initialization: $n = r$,

Step 1. set $P_n = P$ (i.e. $p_{i,j}^{(n)} = p_{i,j}$)

$$P_n = \left[\begin{array}{c|c|c|c} A_{1,1}^{(n)} & A_{1,2}^{(n)} & \cdots & A_{1,n/2}^{(n)} \\ \hline A_{2,1}^{(n)} & A_{2,2}^{(n)} & \cdots & A_{2,n/2}^{(n)} \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline A_{n/2,1}^{(n)} & A_{n/2,2}^{(n)} & \cdots & A_{n/2,n/2}^{(n)} \end{array}\right],$$

where $A_{i,j}^{(n)}$ is given by

$$A_{i,j}^{(n)} = \left[\begin{array}{cc} p_{2i-1,2j-1}^{(n)} & p_{2i-1,2j}^{(n)} \\ p_{2i,2j-1}^{(n)} & p_{2i,2j}^{(n)} \end{array}\right], \qquad 1 \leq i, j \leq n/2$$

Step 2. Partition P_n such that

$$P_n = \left[\begin{array}{c|c} T_n & W_n \\ \hline R_n & A_{n/2,n/2}^{(n)} \end{array}\right], \tag{8}$$

where

$$W_n = (A_{1,n/2}^{(n)}, A_{2,n/2}^{(n)}, \cdots, A_{n/2-1,n/2}^{(n)})',$$

and

$$R_n = (A_{n/2,1}^{(n)}, A_{n/2,2}^{(n)}, \cdots, A_{n/2,n/2-1}^{(n)}).$$

Step 3. Compute the reduced matrix $P_{n-2} = T_n + W_n[I - A_{n/2,n/2}^{(n)}]^{-1} R_n$

Step 4. preparation for the next iteration: set $n \longleftarrow n - 2$.

If $n = 2$ then stop the backward pass,

Else go to **Step 1.**

5.2. **Forward steps.** When the retrograde reduction phase ends, the finale reduced matrix 2×2 is $P_2 = A_{1,1}^{(2)}$. This phase is followed by a vectorial forward enlargement phase that calculates recursively the stationary probabilities of successive enlargement blocks. To do this, we start with the stationary vector (a_1, a_2) of P_2 that we compute using the steady state $\lambda = (\lambda_1, 1 - \lambda_1)$ as follows

$$a_1 = \frac{\lambda_1}{1 - \lambda_1} a_2 \tag{9}$$

Using the fact that (a_1, a_2, a_3, a_4) is a stationary vector of P_4, we obtain:
$(a_3, a_4) = (a_1, a_2)A_{1,2}^{(4)} + (a_3, a_4)A_{2,2}^{(4)} = (a_1, a_2)K_2$, where $K_2 = A_{1,2}^{(4)} \left[I_2 - A_{2,2}^{(4)}\right]^{-1}$.
Using the fact that $(a_1, a_2, a_3, a_4, a_5, a_6)$ is a stationary vector of P_6, we obtain:
$(a_5, a_6) = (a_1, a_2)A_{1,3}^{(6)} + (a_3, a_4)A_{2,3}^{(6)} + (a_5, a_6)A_{3,3}^{(6)} = (a_1, a_2)K_3$,
where $K_3 = \left(A_{1,3}^{(6)} + K_2 A_{2,3}^{(6)}\right) \left[I_2 - A_{3,3}^{6}\right]^{-1}$.
The procedure is carried out until $(a_{n-1}, a_n) = (a_1, a_2)K_{n/2}$, where

$$K_{n/2} = \left(A_{1,n/2}^{(n)} + \sum_{i=2}^{n/2-1} K_i A_{i,n/2}^{(n)}\right) \left[I_{n/2} - A_{n/2,n/2}^{(n)}\right]^{-1}, \quad n = 6, 8, \ldots, r.$$

Summing the equations obtained in the above procedure, we obtain

$$\sum_{i=2}^{n/2}(a_{2i-1},a_{2i})=(a_1,a_2)K,\quad \text{where}\quad K=\sum_{i=2}^{n/2}K_i$$

This implies that

$$1-a_1-a_2=a_1K_{11}+a_2K_{21}+a_1K_{12}+a_2K_{22},$$

where K_{ij} is the $(i,j)^{th}$ entry of the matrix K. Now using (9), we obtain

$$a_1=\frac{1}{(1+K_{11}+K_{12})+\frac{1-\lambda_1}{\lambda_1}(1+K_{12}+K_{22})}\quad\text{and}\quad a_2=\frac{1-\lambda_1}{\lambda_1}a_1 \tag{10}$$

Starting with P_2, the retrograde phase can be executed according to the following successive steps:

Step 1: Compute $\lambda_1=\frac{p^{(2)}_{2,1}}{1-p^{(2)}_{1,1}+p^{(2)}_{2,1}}$ and $K_2=A^{(4)}_{1,2}\left[I_2-A^{(4)}_{2,2}\right]^{-1}$

Step 2: Compute $K_m=\left(A^{(2m)}_{1,m}+\sum_{i=2}^{m-1}K_iA^{(2m)}_{i,m}\right)\left[I_2-A^{(2m)}_{m,m}\right]^{-1}$ for $m=3,4,\ldots,r/2$

Step 3: Compute $K=\sum_{i=2}^{r/2}K_i$ and (a_1,a_2) using (10)

Step 4: Compute $a_3,a_4,\ldots,a_r$ using equations $(a_{2i-1},a_{2i})=(a_1,a_2)K_i,\ \ i=2,4,\ldots,r/2$.

5.3. **Computation cost.** Assuming that the four basic operations ($+,-,*$, and $\div$) have the same calculation cost, a comparison of the costs of the different partitions gives the advantage to 1×1 (the standard case) and 2×2 partitions. For a $n\times n$ transition matrix, the cost of these two partitions is $O(n^3)$ while it is higher for the other partitions. This is predictable since for large partitions we have larger matrices to reverse.

6. Solving system in linear algebra

The general partition method can be applied to solve linear algebra systems. The reduced matrices, used in the method, were appeared for the first time in numerical optimization methods (see Cottle (1974)), Schur (1917) and Zhang (1999)) and they were called *Schur complements*.

Let $\pi A=b$, where A is a $n\times n$ invertible matrix and b and π are vectors in $\mathbb{R}^n$. Given A and b, the unkown vector π can be calculated by the partitioning technique as follows: we partition A in s^2 blocks and b in s blocks:

$$A=A_s=(A^s_{k_i,k_j})_{1\le i,j\le s}$$

where $A^s_{k_i,k_i}$ are square matrices and $A^d_s=(A^s_{k_i,k_j})_{d\le i,j\le s}$ are assumed invertible for $d=1,...,s$. If we denote $b^{s,t}=(b^{k_1}_s,b^{k_2}_s,...,b^{k_t}_s)$ with $b^{k_i}_s=(b^s_{k_1^{i-1}+1},...,b^s_{k_1^i})=(b_{k_1^{i-1}+1},...,b_{k_1^i})$ and $b=b^{s,s}$ $(b^s\equiv b^{s,s})$,

we have

$$(\pi^1,\pi^2,...,\pi^s)A_s=b^s\qquad \pi^i=(\pi_{k_1^{i-1}+1},...,\pi_{k_1^i})\quad\forall i=1,...,s$$

$$(\pi^1,\pi^2,...,\pi^s)\begin{pmatrix}T_s & W_s\\ R_s & A^s_{k_s,k_s}\end{pmatrix}=b^s$$

We compute the matrix $A_{s-1} = T_s - W_s(A^s_{k_s,k_s})^{-1}R_s$ and the reduced vector $b^{s-1} = b^{s-1,s-1} = b^{s,t-1} - b^{k_s}_s(A^s_{k_s,k_s})^{-1}$. So

$$(\pi^1, \pi^2, ..., \pi^{s-1})A_{s-1} = b^{s-1}$$

Iteratively, we compute A_{s-1} and b^{s-1} from A_s and b^s until $s = 1$.

At this step, we have $\pi^1 A_1 = b_1^{k_1} \Rightarrow \pi^1 = b_1^{k_1} A_1^{-1}$ and for $i = 2, ..., s$ we get $\pi^i = K_i + \pi^1 L_i$, where $K_2 = b_2^{k_2}(A^2_{k_2,k_2})^{-1}$, $L_2 = A^2_{k_1,k_2}(A^2_{k_2,k_2})^{-1}$, and for $i > 2$, $K_i = \left(b_i^{k_i} - \sum_{j=2}^{i-1} K_j A^i_{k_j,k_i}\right)(A^i_{k_i,k_i})^{-1}$ and $L_i = \left(-A^i_{k_1,k_i} - \sum_{j=2}^{i-1} L_j A^i_{k_j,k_i}\right)(A^i_{k_i,k_i})^{-1}$

The procedure can be summarized in the following algorithm:

(1) Compute $\pi^1 = b^1 A_1^{-1}$, $K_2 = b^{k_2}(A^2_{k_2,k_2})^{-1}$ and $L_2 = A^2_{k_1,k_2}(A^2_{k_2,k_2})^{-1}$

(2) For $3 \le i \le s$, compute

$$K_i = \left(b^{k_i} - \sum_{j=2}^{i-1} K_j A^i_{k_j,k_i}\right)(A^i_{k_i,k_i})^{-1} \text{ and } L_i = \left(-A^i_{k_1,k_i} - \sum_{j=2}^{i-1} L_j A^i_{k_j,k_i}\right)(A^i_{k_i,k_i})^{-1}$$

(3) Compute $\pi^i = K_i + \pi^1 L_i$ for $2 \le i \le s$

7. Numerical example

In this example we have considered a finite Markov chain (of 10 states) whose transition matrix is P. And we used our "2 partition" algorithm to explicitly determine **a** the stationary vector of our Markov chain, then we used the standard CFTP algorithm on the same chain to determine $\widehat{\mathbf{a}}$ an estimate of the stationary vector for many sample sizes. The approximations obtained are compared to **a**.

Here we give a numerical computation of the steady state probability of our Markov chain, using the 2 partition.

$$P = P_{10} = \begin{pmatrix} 0.10 & 0.25 & 0.15 & 0.10 & 0.00 & 0.00 & 0.10 & 0.05 & 0.10 & 0.15 \\ 0.15 & 0.10 & 0.20 & 0.10 & 0.10 & 0.00 & 0.10 & 0.01 & 0.19 & 0.05 \\ 0.00 & 0.20 & 0.10 & 0.20 & 0.19 & 0.10 & 0.00 & 0.10 & 0.10 & 0.01 \\ 0.00 & 0.10 & 0.20 & 0.10 & 0.20 & 0.10 & 0.10 & 0.05 & 0.07 & 0.08 \\ 0.00 & 0.10 & 0.11 & 0.20 & 0.10 & 0.20 & 0.05 & 0.04 & 0.09 & 0.11 \\ 0.00 & 0.10 & 0.10 & 0.20 & 0.20 & 0.10 & 0.10 & 0.00 & 0.05 & 0.15 \\ 0.09 & 0.10 & 0.10 & 0.00 & 0.10 & 0.01 & 0.20 & 0.10 & 0.20 & 0.10 \\ 0.20 & 0.10 & 0.15 & 0.15 & 0.00 & 0.10 & 0.10 & 0.00 & 0.10 & 0.10 \\ 0.10 & 0.20 & 0.05 & 0.04 & 0.09 & 0.11 & 0.00 & 10 & 0.11 & 0.20 \\ 0.10 & 0.10 & 0.01 & 0.00 & 0.20 & 0.10 & 0.20 & 19 & 0.05 & 0.05 \end{pmatrix}$$

$P_8 = T_{10} + R_{10}(I_2 - A^{(10)}_{5,5})^{-1}W_{10}$

$$P_8 = \begin{pmatrix} 0.130640 & 0.292908 & 0.157971 & 0.104907 & 0.047786 & 0.031867 & 0.136744 & 0.097175 \\ 0.181777 & 0.153680 & 0.211939 & 0.108761 & 0.139461 & 0.033968 & 0.119749 & 0.050664 \\ 0.014889 & 0.226320 & 0.106061 & 0.204572 & 0.207205 & 0.116032 & 0.006918 & 0.118002 \\ 0.018636 & 0.127074 & 0.205239 & 0.103375 & 0.227989 & 0.119479 & 0.120395 & 0.077813 \\ 0.024764 & 0.135655 & 0.116833 & 0.204357 & 0.137546 & 0.225853 & 0.077744 & 0.077248 \\ 0.023758 & 0.130341 & 0.105009 & 0.202633 & 0.240275 & 0.124417 & 0.134351 & 0.039216 \\ 0.128779 & 0.162118 & 0.113214 & 0.009336 & 0.151885 & 0.051113 & 0.230880 & 0.152675 \\ 0.225015 & 0.136984 & 0.157289 & 0.154788 & 0.036864 & 0.126212 & 0.126092 & 0.036756 \end{pmatrix}$$

$P_6 = T_8 + R_8(I_2 - A_{4,4}^{(8)})^{-1}W_8$

$$P_6 = \begin{pmatrix} 0.186150 & 0.343405 & 0.201401 & 0.127280 & 0.082974 & 0.058789 \\ 0.221355 & 0.191898 & 0.243517 & 0.122617 & 0.168009 & 0.052604 \\ 0.047364 & 0.248589 & 0.129454 & 0.224546 & 0.216430 & 0.133617 \\ 0.065477 & 0.170183 & 0.242025 & 0.121783 & 0.258461 & 0.142072 \\ 0.062094 & 0.168200 & 0.145648 & 0.220742 & 0.159001 & 0.244315 \\ 0.063538 & 0.170146 & 0.137134 & 0.215241 & 0.271144 & 0.142797 \end{pmatrix}$$

$P_4 = T_6 + R_6(I_2 - A_{3,3}^{(6)})^{-1}W_6$

$$P_4 \begin{pmatrix} 0.201174 & 0.383893 & 0.235375 & 0.179558 \\ 0.244645 & 0.254731 & 0.296597 & 0.204027 \\ 0.084436 & 0.348511 & 0.213405 & 0.353648 \\ 0.107870 & 0.284468 & 0.338141 & 0.269521 \end{pmatrix}$$

$P_2 = T_4 + R_4(I_2 - A_{2,2}^{(4)})^{-1}W_4$

$$P_2 \begin{pmatrix} 0.297565 & 0.702435 \\ 0.360566 & 0.639434 \end{pmatrix}$$

Vector enlargement routine:

$K_2 = A_{1,2}^{(4)}[I_2 - A_{2,2}^{(4)}]^{-1} = \begin{pmatrix} 0.511314 & 0.493353 \\ 0.627786 & 0.583238 \end{pmatrix}$

$K_3 = \{A_{1,3}^{(6)} + \sum_{i=2}^{2} K_i A_{i,3}^{(6)}\}(I_2 - A_{3,3}^{(6)})^{-1} = \begin{pmatrix} 0.502184 & 0.373181 \\ 0.686126 & 0.451444 \end{pmatrix}$

$K_4 = \{A_{1,4}^{(8)} + \sum_{i=2}^{3} K_i A_{i,4}^{(8)}\}(I_2 - A_{4,4}^{(8)})^{-1} = \begin{pmatrix} 0.429156 & 0.326864 \\ 0.453631 & 0.321925 \end{pmatrix}$

$K_5 = \{A_{1,5}^{(10)} + \sum_{i=2}^{4} K_i A_{i,5}^{(10)}\}(I_2 - A_{5,5}^{(10)})^{-1} = \begin{pmatrix} 0.441302 & 0.494380 \\ 0.588856 & 0.464689 \end{pmatrix}$

$K = K_2 + K_3 + K_4 + K_5 = \begin{pmatrix} 1.883956 & 1.687778 \\ 2.356399 & 1.821296 \end{pmatrix}$

$\lambda_1 = \dfrac{p_{2,1}^{(2)}}{1 - p_{1,1}^{(2)} + p_{2,1}^{(2)}} = 0.339196$

$(a_1, a_2) = \left(\dfrac{1}{(1 + K_{11} + K_{12}) + \frac{\lambda_1}{1-\lambda_1}(1 + K_{21} + K_{22})}; \frac{1-\lambda_1}{\lambda_1} a_1 \right)$

$= (0.068219, 0.132901)$

$(a_3, a_4) = (a_1, a_2)K_2 = (0.118315, 0.111169)$

$(a_5, a_6) = (a_1, a_2)K_3 = (0.125445, 0.085455)$

$(a_7, a_8) = (a_1, a_2)K_4 = (0.089565, 0.065083)$

$(a_9, a_{10}) = (a_1, a_2)K_5 = (0.108365, 0.095484)$

So the exact steady state probability $\mathbf{a}$ is

$\mathbf{a} = (0.068219, 0.132901, 0.118315, 0.111169, 0.125445, 0.085455, 0.089565, 0.065083, 0.108365, 0.095484)$.

In the following figures, there are $(\widehat{a}_i)_{i=1,\ldots,10}$ the estimates of the components of the $\mathbf{a}$ stationary vector, which obtained by the "standard CFTP" algorithm, for many sample sizes ($n = 10^4$, 5.10^4, 10^5, 5.10^5, 10^6, 5.10^6, 10^7, 5.10^7, 10^8) and $(a_i)_{i=1,\ldots,10}$ which obtained by the algorithm of "2×2 partition". And again the figures explain the difference between the two results.

We clearly notice that the estimates are getting closer and closer to each other, when we grow up sample sizes.

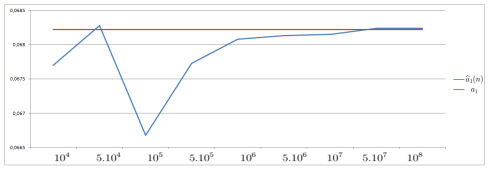

FIGURE 1. a_2 and $\widehat{a}_2(n)$

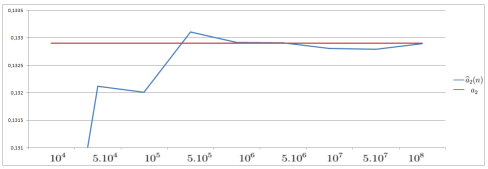

FIGURE 2. a_3 and $\widehat{a}_3(n)$

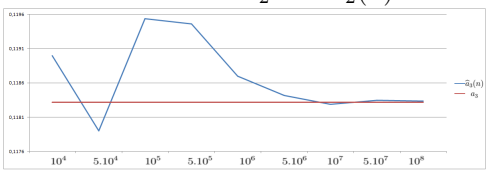

FIGURE 3. a_4 and $\widehat{a}_4(n)$

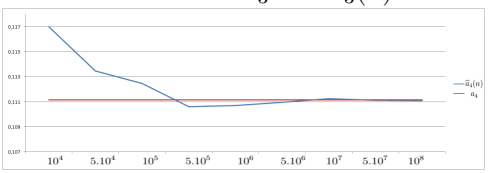

FIGURE 4. a_5 and $\widehat{a}_5(n)$

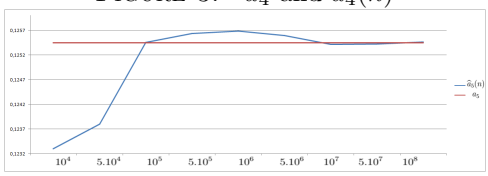

FIGURE 5. a_6 and $\widehat{a}_6(n)$

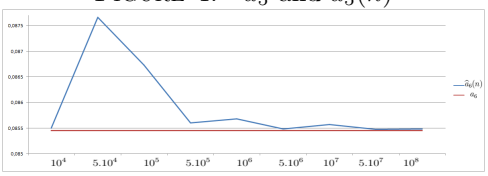

FIGURE 6. a_7 and $\widehat{a}_7(n)$

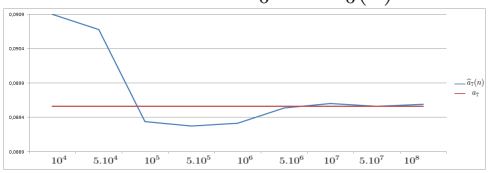

FIGURE 7. a_8 and $\widehat{a}_8(n)$

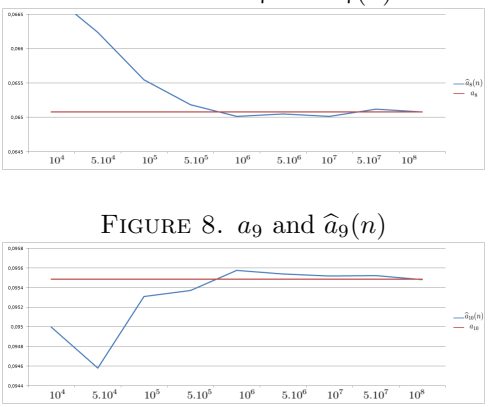

FIGURE 8. a_9 and $\widehat{a}_9(n)$

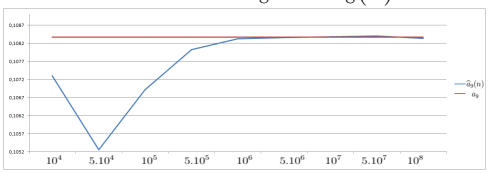

FIGURE 9. a_{10} and $\widehat{a}_{10}(n)$

FIGURE 10. a_{11} and $\widehat{a}_{11}(n)$

8. Conclusion

Following the same approach proposed by Sheskin in his paper (Sheskin, T. J. (1985)), which consist of working with a partition that reduces the dimension of the matrix by a single dimension at each stage, we proposed a general partition method to solve Markov chain steady state probability. Among all proposed partitions, the 2×2-Partition method is the only one that computes with the Sheskin partition with a same cost $O(n^3)$ for a $n \times n$ matrix. The general partitioning method is well adapted for solving system of linear equations. A numerical example of computing the exact steady state probability of a Markov chain by the 2×2-Partition method was given to show the smooth running of the proposed algorithm, then we compared these results by the estimates of the CFTP algorithm that allow to perfectly simulate from the stationary law of a Markov chain, whose state space is finite.

References

1. D. R. Bar and M. U. Thomas, *An Eigenvector Condition for Markov Chain Lumpability*, Opns. Res. **25** (1977), 1028-1031. https://doi.org/10.1287/opre.25.6.1028
2. R. Besenczi, N. Bátfai, P. Jeszenszky, R. Major, F. Monori, and M. Ispány, *Large-scale simulation of traffic flow using Markov model*, PLoS ONE 16(2): e0246062 (2021). https://doi.org/10.1371/journal.pone.0246062
3. G. Bolch, S. Greiner, H. de Meer and K.S. Trivedi, *Queueing Networks and Markov Chain Modelling and Performance Evaluation with Computer Science Applications*, John Wiley & Sons, 2006. https://doi.org/10.1002/0471791571
4. Y. A. Cai, *Non-Monotone CFTP Perfect Simulation Method*, Statistica Sinica **15** (2005), 927-943.
5. R. W. Cottle, *Manifestations of the schur complement*, Linear Algebra and its Applications **8** (1974), no 3, 189-211. https://doi.org/10.1016/0024-3795(74)90066-4
6. L. Dai, *Sensitivity analysis of stationary performance measures for Markov chains*, Math. Comput. Modelling **23** (1996), no. 11-12, 143-160. https://doi.org/10.1016/0895-7177(96)00069-6
7. Dianne P. O'Leary and Yuan-Jye Jason Wu, *A Block-GTH algorithm for finding the stationary vector of a Markov chain*, SIAM Journal on Matrix Analysis and Applications **17** (1996), no. 3, 470-488. https://doi.org/10.1137/s0895479894262534
8. H. Fakhouri and A. Nasroallah, *On the simulation of Markov chain steady-state distribution using CFTP algorithm*, Monte Carlo Methods Appl. **15** (2009), no. 2, 91-105. https://doi.org/10.1515/MCMA.2009.005
9. P. A. Gagniuc, *Markov Chains: From Theory to Implementation and Experimentation*, USA, NJ: John Wiley & Sons, 2017, pp. 1-235. https://doi.org/10.1002/9781119387596
10. F. S. Hillier and R. W. Boling, *Finite Queues in Series with Exponential or Erlang Service Times — A Numerical Approach*, Opns. Res. **15** (1967), no. 2, 286-303. https://doi.org/10.1287/opre.15.2.286
11. J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, Van Nostrand, Princeton, N.J, 1960.
12. S. Mahévas and G. Rubino, *Bound computation of dependability and performance measures*, IEEE Transactions on computers **50(5)** (2001), 399-413. https://doi.org/10.1109/12.926156
13. J. Møller, *Perfect simulation of conditionally specified models*, Journal of the Royal Statistical Society Series B: Statistical Methodology **61** (1999), no. 1, 251-264. https://doi.org/10.1111/1467-9868.00175
14. D. J. Murdoch and P. J. Green, *Exact sampling from a continuous state space*, Scand. J. Statist. **25(3)** (1998), 483-502. https://doi.org/10.1111/1467-9469.00116
15. Murdoch, D. J., and J. S. Rosenthal, *Efficient Use of Exact Samples*, Statistics and Computing **10** (2000), 237-243. https://doi.org/10.1023/A:1008991527785
16. J. G. Propp and D. B. Wilson, *Exact sampling with coupled Markov chains and applications to statistical mechanics*, Random Structures and Algorithms **9** (1996), 223-252. https://doi.org/10.1002/(SICI)1098-2418(199608/09)9:1/2<223::AID-RSA14>3.3.CO;2-R
17. J. G. Propp, and Wilson, D. B. *How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph*, Journal of Algorithms **27** (1998), no. 2, 170-217. https://doi.org/10.1006/jagm.1997.0917
18. I. Schur, *Uber potenzreiben, die im innern des einheitskteises beschrankt sind*, Journal fur die reine und angewandte Mathematik, 1917. https://doi.org/10.1515/crll.1917.147.205
19. T. J. Sheskin, *Allocation of Interstage Storage Along an Automatic Production Line*, AIIE Trans. **8** (1976), 146-152.
20. T. J. Sheskin, *A Markov Chain Partitioning Algorithm for Computing Steady State Probabilities*, Operations Research **33** (1985), no. 1, 228-235. https://doi.org/10.1287/opre.33.1.228
21. A.-E. Zakrad and A. Nasroallah, *Perfect simulation of steady-state Markov chain on mixed state space*, Communications in Statistics - Theory and Methods **51** (2022), no. 6, 1569-1587. https://doi.org/10.1080/03610926.2021.1924783
22. F. Z. Zhang, *Matrix Theory: Basic Results and Techniques*, Springer-Verlag, New York, 1999, pp. 36-39.

Department of Mathematics, Semlalia Faculty of Sciences, Cadi Ayyad University, Morocco
E-mail address: zakrad92@gmail.com

Department of Mathematics, Semlalia Faculty of Sciences, Cadi Ayyad University, Morocco
E-mail address: nasroallah@uca.ac.ma