Stochastic bounds with a low rank decomposition

Ana Bušić [1] , Jean-Michel Fourneau [2], Mouad Ben Mamoun [3]

[1] Inria, DI ENS, 23 avenue d'Italie,

Paris, France,

[2] PRiSM, UVSQ and CNRS, UMR 8144,

45 avenue des Etats Unis, 78000 Versailles, France

3 Fac de Sciences, Univ. Mohammed V, Agdal,

Rabat, Maroc

July 10, 2014

Abstract

We investigate how we can bound a Discrete Time Markov Chain (DTMC) by a stochastic matrix with a low rank decomposition. In the first part of the paper we show the links with previous results for matrices with a decomposition of size 1 or 2. Then we show how the complexity of the analysis for steady-state and transient distributions can be simplified when we take into account the decomposition. Finally, we show how we can obtain a monotone stochastic upper bound with a low rank decomposition.

1 Introduction

The problem of approximating a matrix (not necessarily square) by another one which has a lower rank has received a considerable attention in the literature (see for instance [2, 11] and references therein). Indeed many algorithms can be derived with a much lower complexity when the rank is low and there exist many applications for this approximation (web search model [14], latent semantic indexing [7], image compression, again see the references already mentioned and [9] for Markov chains). The best approximation is known to be found from the Singular Value Decomposition (SVD) of the initial matrix. This is a well-known result by Eckart and Young [12].

Here we address a related problem. We want to compute a monotone upper bound of a stochastic matrix and we want this stochastic bound to have a low rank decomposition. We formally define a low rank decomposition of Markov chains in the next section but one must avoid the confusion with the rank

of the linear system defining the steady-state distribution. Indeed, for an irreducible Markov chain, it is well-known that this linear system has a rank equal to $N-1$, where N is the number of states.

We want to build such a bounding matrix with a low complexity algorithm and we also want to prove that analyzing the DTMC associated to the bound will be easier with ad-hoc algorithms which are based on the size of the decomposition. Note that the approach based on SVD is not considered in the literature on Markov chains. Indeed, computing the SVD of a square matrix of size N has a cubic complexity. Thus it is as difficult as the computation of steady-state distribution.

Furthermore we compute a monotone stochastic upper bound of the matrix rather than an approximation. With a monotone stochastic upper bound of the matrix, one can obtain stochastic bounds on the steady-state distribution or the transient distributions. This allows to derive guarantees in performance [17] or reliability [16] modeling or in stochastic model checking [5]. Such an approach is not feasible if we only have an approximation of the matrix.

This approach is motivated by two already published papers which have introduced two families of matrices which are really easy to analyze. Both the steady-state and the transient distributions have a closed form distribution based on some parameters. These parameters can be computed with a linear time algorithm in the size of the matrix. Furthermore, the distribution of the first passage time can also be solved with a simple numerical computation. These families are denoted as Class $\mathbf{C}$ DTMC [4, 6] and Class $\mathbf{C^G}$ DTMC [3]. It can be easily checked that matrices in both families have a decomposition of size one and a rank equal to one or two. Such a statement does not appear in the previous references on these families. In our opinion, this is the key property which explains why we have algorithms with linear complexity. Here we show that the size of the decomposition is related to the complexity of resolution of transient and stationary regimes and we generalize the approach to any DTMC with a low rank (or low size) decomposition as we establish a simple relation between the size of the decomposition and the rank of the matrix. We also give an algorithm to build a monotone stochastic upper bounding matrix with a low rank decomposition for an arbitrary DTMC.

The technical part of the paper is organized as follows. In section 2, we present the results for matrices of Class $\mathbf{C^G}$ and we formally introduce the concept of a low rank decomposition of a DTMC. Then in section 3, we give a short introduction to stochastic ordering to define the stochastic monotonicity and the stochastic comparison of matrices and DTMC. Section 4 is devoted to the analysis of steady state distribution of DTMC with a low rank decomposition while Section 5 contains the results about transient distributions and Section 6 is devoted to stochastic monotonicity. Finally in section 7, we explain how we can obtain a low rank stochastic bound of a DTMC. The results can be generalized to lower bound as well but this is omitted in this version of the paper. Instead, we present many small examples to explain the key ideas and the algorithms.

All the vectors are row vectors. The state space has size N. All the DTMC are time-homogenous.

2 Matrices with a low rank decomposition

We first recall class $\mathbf{C^G}$ matrices from [3], then propose a generalization.

2.1 Class $\mathbf{C^G}$ Matrices

Definition 1 *A stochastic matrix* $\mathbf{M}$ *belongs to class* $\mathbf{C^G}$ *if there are three vectors* v, r, *and* c *such that:*

$$\mathbf{M} = e^T v + r^T c,$$

where e *is a vector whose entries are all equal to* 1, v *is a stochastic vector (i.e.* $\sum_i v[i] = 1$, *and for all* i, $v[i] \geq 0$*),* c *is a vector whose entries sum to* 0, *and* e^T *is the transpose of* e.

Example 1 *Consider the following class* $\mathbf{C^G}$ *matrix:*

$$\begin{aligned} \mathbf{M} &= [1,1,1,1]^T[0.2, 0.1, 0.4, 0.3] \\ &+ [0,1,1,2]^T[0.1, -0.05, 0.05, -0.1]. \end{aligned}$$

M is equal to:

$$\mathbf{M} = \begin{bmatrix} 0.2 & 0.1 & 0.4 & 0.3 \\ 0.3 & 0.05 & 0.45 & 0.2 \\ 0.3 & 0.05 & 0.45 & 0.2 \\ 0.4 & 0.0 & 0.5 & 0.1 \end{bmatrix}$$

We first need to compute two quantities of interest which appear in the closed form solutions: $a = cr^T$ and $b = vr^T$.

Property 1 *[3] The stationary distribution of a class* $\mathbf{C^G}$ *DTMC is* $v + \mathbf{E}[r]c$ *where* $\mathbf{E}[r] = \frac{vr^T}{1-a}$ *if* $a = cr^T$ *is not equal to* 1. *The fact that* $\mathbf{E}[r]$ *is an expectation will appear more clearly in the next section.*

Property 2 *[3] Consider a class* $\mathbf{C^G}$ *DTMC defined by matrix* $\mathbf{M}$ *and initial distribution* $\pi^{(0)}$. *We have:*

$$\mathbf{M}^k = e^T v + [a^{k-1} r^T + (b \sum_{i=0}^{k-2} a^i) e^T] c, \tag{1}$$

and the transient distributions at instant k *is:*

$$\pi^{(k)} = v + \nu_k c$$

with the following recursion on ν_k:

$$\nu_k = b\sum_{i=0}^{k-2} a^i + (\pi^{(0)} r^T)a^{k-1}$$

The distribution of the time to absorption also has a closed form solution which can be numerically solved by a recursion [8].

Definition 2 *[4] A class* $\mathbf{C}$ *stochastic matrix* $\mathbf{M}$ *is defined by 2 vectors* v *and* c *such that:*

$$\mathbf{M} = [1, 1, 1, \ldots, 1]^T v + [0, 1, 2, \ldots, n-1]^T c,$$

where v *is a stochastic vector and* c *is a vector whose entries sum to* 0*.*

Property 3 *Class* $\mathbf{C}$ *is included into Class* $\mathbf{C^G}$*. This is trivial due to the definition of Class* $\mathbf{C}$ *matrices. Note that for Class* $\mathbf{C}$ *matrices, it is proved that* $a = cr^T$ *is not equal to* 1 *[4].*

2.2 Low rank decomposition

Definition 3 *We consider the set* $\mathcal{A}_k$ *of stochastic matrices* $\mathbf{M}$ *defined by:*

$$\mathbf{M} = e^T v + \sum_{i=1}^{k} r_i^T c_i,$$

where v *is a stochastic vector,* c_i *are vectors whose entries sum to* 0 *and* r_i *are vectors such that* $r_i[j] \leq 1$ *for all* i *and* j*. We say that a matrix in* $\mathcal{A}_k$ *has a decomposition of size* k*.*

Property 4 *Matrices in* $\mathcal{A}_k$ *have rank at most* $k+1$

Proof: Clear, as we add $k+1$ matrices of rank 1. □

Thus we call matrices in $\mathcal{A}_k$, matrices with a low rank decomposition when k is small. We show in the following that the complexity of the algorithms is related to k.

Remark 1 *By construction, matrices of class* $\mathbf{C^G}$ *are in* $\mathcal{A}_1$*.*

Remark 2 *By construction, matrices of* $\mathcal{A}_0$ *have all their rows equal. Therefore, after their first transition they are at steady-state, for any initial state. Indeed, matrices in* $\mathcal{A}_0$ *are equal to* $e^T v$ *and their steady-state distribution is* v*.*

Remark 3 *It must be clear that every row of* $\mathbf{M}$ *sums to* 1 *due to the constraints on the decomposition. However, these constraints do not imply that the entries of the matrix are not negative. Furthermore, such a decomposition is not unique. If* (v, r, c) *is a decomposition of size* 1*,* $(v, r/\alpha, \alpha c)$ *for any positive* α*, is also a valid decomposition of the same matrix. In the following, we only consider a valid decomposition of a stochastic matrix.*

Property 5 *[Simplification rules] For all x, y z row vectors with the same size, $xy^T z = zxy^T$ as xy^T is a scalar. Furthermore by construction $ve^T = 1$ and $c_i e^T = 0$ for all i.*

We assume in the following that $k << N$.

3 A brief introduction to stochastic comparison of DTMC

We first have to introduce the strong stochastic comparison of probability vectors. This will allow us to compare discrete random variables which can be defined through their probability mass functions, and also stochastic matrices whose rows are probability vectors. See also references [18] and [19] .

Definition 4 *Let p and q be two probability vectors of size N. Then*

$$p \preceq_{st} q \quad \textit{iff} \quad \sum_{j=k}^{N} p[j] \leq \sum_{j=k}^{N} q[j] \quad \forall k \in \{1, 2 \cdots N\}.$$

Definition 5 *Let X and Y be two discrete random variables taking values in a totally ordered space $\mathcal{S} = \{1, 2, \cdots, N\}$ with probability mass functions p and q respectively. Then*

$$X \preceq_{st} Y \quad \textit{iff} \quad p \preceq_{st} q.$$

The strong stochastic comparison is associated to the non-decreasing functions.

Proposition 1 *$X \preceq_{st} Y$, iff $\mathbf{E}[f(X)] \leq \mathbf{E}[f(Y)]$, for all non decreasing function f, provided that the expectations exist.*

Important performance and reliability indices such as average population, loss rates or tail probabilities, availability (with a suitable ordering of states) are non decreasing functions. Therefore, bounds on the distribution imply bounds on these performance indices as well. This property also holds for many formulas in stochastic model checking. As we deal with stochastic matrices, we use the following characterization that involves the matrix associated with strong stochastic comparison.

Proposition 2 *$p \preceq_{st} q$, if and only if $p\mathbf{K_{st}} \preceq_{el} q\mathbf{K_{st}}$, where $\preceq_{el}$ is the element-wise comparison and:*

$$\mathbf{K_{st}} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}.$$

The stochastic comparison of random variables has been generalized to the comparison of Markov chains. Both continuous time models and discrete time ones can be addressed (the conditions are slightly different) but we restrict ourselves here to DTMC.

Definition 6 *Let* $\{X_1(n), n \in \mathbb{N}\}$ *and* $\{X_2(n), n \in \mathbb{N}$ *two Discrete Time Markov Chains taking values in state space* $\mathcal{S}$. $\{X_1(n), n \in \mathbb{N}\}$ *is said to be stochastically smaller than* $\{X_2(n), n \in \mathbb{N}\}$ *(denoted as* $\{X_1(n)\} \preceq_{st} \{X_2(n)\}$*), iff*

$$X_1(0) \preceq_{st} X_2(0) \Longrightarrow X_1(n) \preceq_{st} X_2(n), \ \ \forall n > 0.$$

Thus the strong stochastic comparison implies the comparison of transient distributions. If the steady-state distributions exist, their comparison also holds as stated in the next property.

Property 6 (Comparison of steady-state) *[18] Let* π_1 *(resp.* π_2*) be the steady-state distribution of* $\{X_1(n), n \geq 0\}$ *(resp.* $\{X_2(n), n \geq 0\}$*). Then*

$$\{X_1(n)\} \preceq_{st} \{X_2(n)\} \Longrightarrow \pi_1 \preceq_{st} \pi_2.$$

We now give some sufficient conditions for stochastic comparison for DTMCs. These conditions are based on properties of the stochastic matrices associated with the chains: the stochastic comparison and the stochastic monotonicity [13] of stochastic matrices. In the following, we denote by $\mathbf{P}[i, *]$ the i-th row of stochastic matrix $\mathbf{P}$.

Definition 7 *Let* $\mathbf{P}$ *and* $\mathbf{Q}$ *be two stochastic matrices.* $\mathbf{Q}$ *is said to be a stochastic upper bounding matrix of* $\mathbf{P}$ ($\mathbf{P} \preceq_{st} \mathbf{Q}$) *if*

$$\forall i, \quad \mathbf{P}[i, *] \preceq_{st} \mathbf{Q}[i, *].$$

Definition 8 *Let* $\mathbf{P}$ *be a stochastic matrix.* $\mathbf{P}$ *is said to be stochastically monotone if for any probability vectors* p *and* q*,*

$$p \preceq_{st} q \implies p\,\mathbf{P} \preceq_{st} q\,\mathbf{P}.$$

A DTMC is said to be monotone if its probability transition matrix is stochastically monotone. In the case of finite state space with a total ordering, the following property provides a much easier characterization, which has also a nice algorithmic interpretation.

Property 7 $\mathbf{P}$ *is stochastically monotone if and only if* $\mathbf{P}[i, *] \preceq_{st} \mathbf{P}[i+1, *]$ *for all states* i *such that* $1 \leq i \leq N-1$.

It is now possible to give the first result on sufficient conditions for the comparison of DTMCs.

Theorem 1 *[18] Let* $\mathbf{P}$ *and* $\mathbf{Q}$ *be the probability transition matrices of two DTMCs (say* $\{X(n), n \in \mathbb{N}\}$ *and* $\{Y(n), n \in \mathbb{N}\}$*). If*

1. $X(0) \preceq_{st} Y(0)$,
2. *at least one of the stochastic matrices (*$\mathbf{P}$ *or* $\mathbf{Q}$*) is stochastically monotone,*
3. *the stochastic matrices are comparable, (i.e.* $\mathbf{P} \preceq_{st} \mathbf{Q}$*).*

then $X(n) \preceq_{st} Y(n)$ *for all* $n \in \mathbb{N}$*. Furthermore, if both chains are ergodic, then their steady-state distributions are also ordered:* $\pi_X \preceq_{st} \pi_Y$.

Assuming that matrix $\mathbf{P}$ is known and it is not stochastically monotone, we want to compute an upper bounding matrix $\mathbf{Q}$. It is quite easy to derive a set of linear inequalities which represent conditions 2 and 3 of Theorem 1:

$$\begin{cases} \sum_{k=j}^{N} \mathbf{Q}[1,k] & \geq & \sum_{k=j}^{N} \mathbf{P}[1,k] \quad \forall\ j, \\ \sum_{k=j}^{N} \mathbf{Q}[i+1,k] & \geq & \sum_{k=j}^{N} \mathbf{Q}[i,k] \quad \forall\ j, \quad \forall\ i = 1, \ldots, N-1\,. \end{cases} \tag{2}$$

Vincent's algorithm [1, 10] consists mainly in replacing inequalities by equalities and reordering them to obtain a constructive way to build the upper bounding matrix $\mathbf{Q}$.

Algorithm 1 Vincent's Algorithm for an Upper Bound

Require: $\mathbf{P}$.
Ensure: monotone upper bound $\mathbf{Q}$.

```
for i = 1 to N do
    Q[1,i] = P[1,i]
end for
for j = 2 to N do
    for i = N downto 1 do
        Q[j,i] = Max(sum_{k=i}^{N} P[j,k], sum_{k=i}^{N} Q[j-1,k]) - sum_{k=i+1}^{N} Q[j,k]
    end for
end for
```

Unfortunately the matrix obtained by Vincent's algorithm is in general as difficult as the original matrix to analyze. Therefore many algorithms have been proposed (see [10] for a survey) to provide bounds which are easy to compute. Here we use a property of Vincent's algorithm: it computes the maximum for the strong stochastic ordering of the probability distributions associated with the rows of $\mathbf{P}$. This computation is part of the derivation of an upper stochastic bound with a size k decomposition.

Remark 4 *Every stochastic matrix has a monotone upper bound with a size* 0 *decomposition. Indeed, we have* $\mathbf{P} \preceq_{st} e^T max$*, where max is the maximum with the stochastic ordering of the rows of* P.

4 Solving the stationary regime for a DTMC with a low rank decomposition

Let us give the first result which justify to study a DTMC with a low rank decomposition. We define matrix $\mathbf{A}$ by:

$$\mathbf{A}[i,l] = c_l r_i^T.$$

This $k \times k$ matrix plays a role similar to scalar a for chains of class $\mathbf{C^G}$. Similarly, let V be the vector whose entry i is vr_i^T.

Property 8 *If matrix* $(\mathbf{I}-\mathbf{A})$ *is not singular, then the stationary distribution of an ergodic DTMC* $\mathbf{M}$ *in* $\mathcal{A}_k$ *is given as follows:*

$$\pi = v + \sum_{i=1}^{k} \mathbf{E}[r_i] c_i, \tag{3}$$

where the values of $\mathbf{E}[r_i]$*, which represent the expectations of* r_i *on the steady-state distribution* π*, can be solved from a linear system of size* k*:*

$$\mathbf{E}[r_i] = \sum_j r_i[j]v[j] + \sum_{l=1}^{k} \mathbf{E}[r_l] \sum_j r_i[j]c_l[j].$$

Note that we compute the expectations $\mathbf{E}[r_i]$ *before we compute the steady-state distribution. Let* E *be the vector of expectations. As matrix* $(\mathbf{I}-\mathbf{A})$ *is not singular by assumption, we have:*

$$E = V(\mathbf{I}-\mathbf{A})^{-1}.$$

Proof: By definition, as the Markov chain is ergodic, π exists and is the solution of $\pi = \pi\mathbf{M}$. After substitution, we get:

$$\pi = \pi e^T v + \sum_{i=1}^{k} \pi r_i^T c_i.$$

But $\pi e^T = 1$ and $\pi r_i^T = \mathbf{E}[r_i]$. After substitution, we get Equation 3. It remains to compute the values of $\mathbf{E}[r_i]$ for all i. By definition we have:

$$\mathbf{E}[r_i] = \sum_j r_i[j]\pi[j] = \sum_j r_i[j] \left(v[j] + \sum_{l=1}^{k} \mathbf{E}[r_l]c_l[j] \right).$$

Thus:

$$\begin{aligned} \mathbf{E}[r_i] &= \textstyle\sum_j r_i[j]v[j] + \sum_{l=1}^{k} \mathbf{E}[r_l] \sum_j r_i[j]c_l[j] \\ &= vr_i^T + \textstyle\sum_{l=1}^{k} \mathbf{E}[r_l]c_l r_i^T. \end{aligned}$$

This is a non homogenous linear system of size k. The equation can be written in a more abstract form as:

$$E = V + E\mathbf{A}.$$

Therefore as matrix $(\mathbf{I} - \mathbf{A})$ is not singular by assumption, we have:

$$E = V(\mathbf{I} - \mathbf{A})^{-1}.$$

□

The property allows to derive an algorithm with a much lower complexity than N^3.

Algorithm 2 Computation of the steady-state distribution for a DTMC with a size k decomposition

Require: $\mathbf{M}$.
Ensure: steady-state distribution π

1: Compute vector V: $V(i) = vr_i^T$.
2: Compute matrix $\mathbf{A}$: $\mathbf{A}(i,l) = c_l r_i^T$.
3: Compute vector $E = V(\mathbf{I} - \mathbf{A})^{-1}$.
4: Compute π from Eq. 3 .

Property 9 *The complexity for computing the steady state distribution is* $O(k^2N)$.

Proof: The first step requires k products of vector of size N while Step 2) needs k^2 products of vectors with the same size. The numerical solution of the linear system at Step 3) could be done with any elimination algorithm. They typically are cubic in time (i.e. $O(k^3)$ here). Finally, the last step has a complexity of $O(kN)$ as we have to compute the N entries of vector π and all of them require k additions and multiplications. As we assume that $k << N$, the most important step for time complexity is Step 2.

□

Example 2 *Consider matrix* $\mathbf{M}$ *whose decomposition of size* 2 *is:*

$$\begin{array}{rl} & [1,1,1,1,1,1,1,1]^T \\ & [0.1,0.2,0.1,0.1,0.1,0,0.2,0.2] \\ + & [0,0.01,0.01,0.1,0.1,0.2,0.5,0.5]^T \\ & [-0.05,-0.05,-0.1,0.2,0,0,0,0] \\ + & [0,0.5,1,0,1,0,1,0]^T \\ & [0,0,0,0,-0.1,0.1,-0.1,0.1]. \end{array}$$

Matrix $\mathbf{A}$ *is readily computed:* $\begin{bmatrix} 0.0185 & -0.125 \\ -0.01 & -0.2 \end{bmatrix}$, *and* V *is* $[0.223, 0.5]$. *Therefore* $E = [0.2312031, 0.3925830]$

and the steady state distribution of the matrix is:

$$v + 0.2312031 * c1 + 0.392583 * c2.$$

One finally get that this distribution is:

$$[0.0884, 0.1884, 0.0769, 0.1462, 0.0608, 0.0393, 0.1607, 0.2393].$$

One can also compute matrix $\mathbf{M}$ *(see below) from the decomposition and check that the former vector is the steady-state distribution:*

$$\begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.1 & 0.1 & 0. & 0.2 & 0.2 \\ 0.0995 & 0.1995 & 0.099 & 0.102 & 0.05 & 0.05 & 0.15 & 0.25 \\ 0.0995 & 0.1995 & 0.099 & 0.102 & 0. & 0.1 & 0.1 & 0.3 \\ 0.095 & 0.195 & 0.09 & 0.12 & 0.1 & 0. & 0.2 & 0.2 \\ 0.095 & 0.195 & 0.09 & 0.12 & 0. & 0.1 & 0.1 & 0.3 \\ 0.09 & 0.19 & 0.08 & 0.14 & 0.1 & 0. & 0.2 & 0.2 \\ 0.075 & 0.175 & 0.05 & 0.2 & 0. & 0.1 & 0.1 & 0.3 \\ 0.075 & 0.175 & 0.05 & 0.2 & 0.1 & 0. & 0.2 & 0.2 \end{bmatrix}.$$

5 Computing the transient distribution for a DTMC with a low rank decomposition

Let us now consider the transient distribution. Let $\pi^{(0)}_{\mathbf{M}}$ be the initial distribution.

Property 10 *The transient distribution at discrete time* n *of a DTMC* $\mathbf{M}$ *in* $\mathcal{A}_k$ *is given* $\pi^{(n)} = \pi^{(0)}\mathbf{M}^n$ *where* $\mathbf{M}^n$ *is given by:*

$$\mathbf{M}^n = e^T v + \sum_{i=1}^{k} (s_i^n)^T c_i, \tag{4}$$

where the s_i^n *are given by the following recursion:*

$$\begin{bmatrix} s_i^1 & = & r_i \\ s_i^{n+1} & = & v(s_i^n)^T e + \sum_j c_j (s_i^n)^T r_j \end{bmatrix}$$

Proof: by induction on the time step n. The equation is clearly true by construction for $n = 1$.

Let us now assume that it it true for n and compute $\mathbf{M}^{n+1}$.

$$\begin{aligned}\mathbf{M}^{n+1} &= \mathbf{M}\mathbf{M}^n \\ &= (e^T v + \sum_{j=1}^k r_j^T c_j)(e^T v + \sum_{i=1}^k (s_i^n)^T c_i) \\ &= e^T v e^T v + \sum_{i=1}^k r_i^T c_i e^T v + e^T v \sum_{i=1}^k (s_i^n)^T c_i \\ &\quad + \sum_{j=1}^k r_j^T c_j \sum_{i=1}^k (s_i^n)^T c_i.\end{aligned}$$

Remember that $ve^T = 1$ and $c_i e^T = 0$ for all i. After substitution we get:

$$\mathbf{M}^{n+1} = e^T v + \sum_{i=1}^k e^T v (s_i^n)^T c_i + \sum_{i=1}^k \sum_{j=1}^k r_j^T c_j (s_i^n)^T c_i.$$

As $v(s_i^n)^T$ and $c_j(s_i^n)^T$ $c_j(s_i^n)^T$ are scalars, we can exchange the terms in the products to obtain:

$$\mathbf{M}^{n+1} = e^T v + \sum_{i=1}^k v(s_i^n)^T e^T c_i + \sum_{i=1}^k \sum_{j=1}^k c_j (s_i^n)^T r_j^T c_i.$$

After identification, we get the recursion with

$$(s_i^{n+1})^T = v(s_i^n)^T e^T + \sum_j c_j (s_i^n)^T r_j^T.$$

After a final transposition we get the recursion equation. □

The recursion equation can be solved after introducing some notation. Let $\gamma_i^n = v(s_i^n)^T$ and $\beta_{i,j}^n = c_j(s_i^n)^T$ We define vectors $G^{(n)}$ and matrices $\mathbf{B}^{(n)}$ as follows:

$$G^{(n)}[i] = v(s_i^n)^T = \gamma_i^n$$

and,

$$\mathbf{B}^{(n)}[i,j] = c_j(s_i^n)^T = \beta_{i,j}^n$$

We also use matrix $\mathbf{A}$ and vector V previously defined. Remember that $\mathbf{A}[i,l] = c_l r_i^T$ and $V[i] = v r_i^T$.

Property 11 *The recursion*

$$\left[\begin{array}{lcl} s_i^1 & = & r_i \\ s_i^{n+1} & = & \gamma_i^n e + \sum_j \beta_{i,j}^n r_j, \end{array}\right.$$

where γ_i^n and $\beta_{i,j}^n$ are the entries of the following vectors and matrices:

$$G^{(n)} = \sum_{m=0}^{n-1} \mathbf{A}^m V,$$

and

$$\mathbf{B}^{(n)} = \mathbf{A}^n.$$

Proof:

Remember the recursion equation:

$$(s_i^{n+1})^T = v(s_i^n)^T e^T + \sum_j c_j (s_i^n)^T r_j^T.$$

Multiply by vector v:

$$v(s_i^{n+1})^T = vv(s_i^n)^T e^T + \sum_j vc_j (s_i^n)^T r_j^T.$$

Remember that $v(s_i^n)^T$ is a scalar. So are $c_j(s_i^n)^T$ for all j. Therefore

$$v(s_i^{n+1})^T = v(s_i^n)^T ve^T + \sum_j c_j (s_i^n)^T vr_j^T.$$

Now remember that $ve^T = 1$ and we substitute with γ_i^n and $\beta_{i,j}^n$ to get a recursion:

$$\alpha_i^{n+1} = \gamma_i^n + \sum_j \beta_{i,j}^n vr_j^T.$$

Using a matrix form:

$$R^{(n+1)} = R^{(n)} + \mathbf{B}^{(n)} V.$$

Let us now consider again the recursion equation and let us multiply it by c_l to get:

$$c_l(s_i^{n+1})^T = c_l v(s_i^n)^T e^T + \sum_j c_l c_j (s_i^n)^T r_j^T.$$

With the same properties previously used and taking into account that $c_l e^T = 0$, we get a recursion on $\beta_{i,j}^n$:

$$\beta_{i,l}^{n+1} = \sum_j \beta_{i,j}^n c_l r_j^T = \sum_j \beta_{i,j}^n \mathbf{A}[j,l]$$

Or in matrix form:

$$\mathbf{B}^{(n+1)} = \mathbf{B}^{(n)} \mathbf{A}.$$

By induction we get $\mathbf{B}^{(n)} = \mathbf{B}^{(1)} \mathbf{A}^{n-1}$.

Let us now compute $\mathbf{B}^{(1)}$. By construction:

$$\mathbf{B}^{(1)}[i,j] = c_j (s_i^1)^T,$$

and $s_i^1 = r_i$ for all i. Therefore:

$$\mathbf{B}^{(1)}[i,j] = c_j (r_i)^T = \mathbf{A}[i,j].$$

Finally,

$$\mathbf{B}^{(n)} = \mathbf{A}^n.$$

Let us now return to the recursion on γ_i^m to conclude. We have $A^{(1)} = V$. And,

$$R^{(n)} = V + \sum_{m=1}^{n-1} \mathbf{A}^m V = \sum_{m=0}^{n-1} \mathbf{A}^m V.$$

□

Property 12 *Taking into account the decomposition into a sum of rank* 1 *matrices, once* $\mathbf{A}$ *and* V *have been computed, the computation of the transient distribution at time* n *has a complexity in* $O(ln(n)k^3)$ *with an improved number of matrix multiplications using a fast algorithm based on the binary decomposition of* n*.*

Proof: The most expensive step of computation is the derivation of $\mathbf{A}^n$. We apply the fast exponentiation algorithm [15] to minimize the number of matrix multiplication. We have assumed a basic multiplication of dense matrices with a complexity in $O(k^3)$. □

6 Stochastic monotonicity of a DTMC with a low rank decomposition

Among the matrices with a low rank decomposition, some of them are stochastically monotone. We prove in the following conditions which are easy to check (i.e. with a smaller complexity than Vincent's algorithm). Such a result was previously proved for matrix of class $\mathbf{C^G}$.

Property 13 *[4] A matrix of class* $\mathbf{C^G}$ *is stochastically monotone if vector* r *is non decreasing and vector* $c\mathbf{K_{st}}$ *is non negative.*

Property 14 *If the following conditions hold, matrix* $\mathbf{M}$ *is st-monotone:*

1. *for all* i*, vector* r_i *is non decreasing.*
2. *for all* i*, vector* $c_i\mathbf{K_{st}}$ *is non negative.*

Proof: Consider two arbitrary distributions such that $p \preceq_{st} q$ and compute $p\mathbf{M}$:

$$p\mathbf{M}\mathbf{K_{st}} = pe^T v\mathbf{K_{st}} + \sum_{i=1}^{k} r_i^T c_i \mathbf{K_{st}}.$$

As $pe^T = 1$, we get:

$$p\mathbf{MK_{st}} = v\mathbf{K_{st}} + \sum_{i=1}^{k} pr_i^T c_i \mathbf{K_{st}} = v\mathbf{K_{st}} + \sum_{i=1}^{k} (pr_i^T)(c_i\mathbf{K_{st}}).$$

The second equation is justified by associativity and is written to emphasize the decomposition approach of the assumptions. Similarly we get:

$$q\mathbf{MK_{st}} = v\mathbf{K_{st}} + \sum_{i=1}^{k} (qr_i^T)(c_i\mathbf{K_{st}}).$$

Remember that $p \preceq_{st} q$. Therefore if vector r_i is non decreasing, we have $pr_i^T \leq qr_i^T$. Finally, all vectors $c_i\mathbf{K_{st}}$ are non negative by assumptions. Thus we get:

$$\sum_{i=1}^{k} (pr_i^T)(c_i\mathbf{K_{st}}) \preceq_{el} \sum_{i=1}^{k} (qr_i^T)(c_i\mathbf{K_{st}}).$$

and $p\mathbf{MK_{st}} \preceq_{el} q\mathbf{MK_{st}}$. □

Example 3 *Let us consider the matrix with the following decomposition of size* 2 *studied in Example 2. One can easily check that vector* r_2 *does not satisfy the assumptions. Furthermore the resulting matrix is not stochastically monotone (clearly seen with the last column).*

We change r_2 *and we obtain a new matrix which satisfies the assumptions of the theorem (new vector* r_2 *is not decreasing),*

$$\begin{array}{cl} & [1,1,1,1,1,1,1,1]^T \\ & [0.1,0.2,0.1,0.1,0.1,0,0.2,0.2] \\ + & [0,0.01,0.01,0.1,0.1,0.2,0.5,0.5]^T \\ & [-0.05,-0.05,-0.1,0.2,0,0,0,0] \\ + & [0,0,0,0,0,0,0.1,0.1]^T \\ & [0,0,0,0,-0.1,0.1,-0.1,0.1]. \end{array}$$

and we can verify that the resulting matrix (see below) is monotone.

$$\begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.1 & 0.1 & 0. & 0.2 & 0.2 \\ 0.0995 & 0.1995 & 0.099 & 0.102 & 0.1 & 0. & 0.2 & 0.2 \\ 0.0995 & 0.1995 & 0.099 & 0.102 & 0.1 & 0. & 0.2 & 0.2 \\ 0.095 & 0.195 & 0.09 & 0.12 & 0.1 & 0. & 0.2 & 0.2 \\ 0.095 & 0.195 & 0.09 & 0.12 & 0.1 & 0. & 0.2 & 0.2 \\ 0.09 & 0.19 & 0.08 & 0.14 & 0.1 & 0. & 0.2 & 0.2 \\ 0.075 & 0.175 & 0.05 & 0.2 & 0.09 & 0.01 & 0.19 & 0.21 \\ 0.075 & 0.175 & 0.05 & 0.2 & 0.09 & 0.01 & 0.19 & 0.21 \end{bmatrix}$$

7 Monotone stochastic bounds with a low rank decomposition

Several algorithms have been presented in the past for the computation of a bounding matrix of class $\mathbf{C}$ or $\mathbf{C^G}$. Let us first present the algorithm introduced in [8] which we will modify to obtain a bound with a size k decomposition. Let us remember the problem for class $\mathbf{C^G}$ and then we present two generalisations based on column or row partition.

We consider a DTMC $\mathbf{P}$ and we want to build a monotone upper bound of $\mathbf{P}$ (say $\mathbf{M}$) in $\mathcal{A}_1$. Thus we have (assuming that row $i-1$ exists):

$$\mathbf{P}(i,*) \preceq_{st} \mathbf{M}(i,*) \quad and \quad \mathbf{M}(i-1,*) \preceq_{st} \mathbf{M}(i,*)$$

Algorithm 3 Monotone Upper Bound in class $\mathbf{C^G}$

Require: $\mathbf{P}$.
Ensure: monotone upper bound in class $\mathbf{C^G}$ described by (v, r, c).

1: $v = \mathbf{P}[1,*]$
2: r[1]=0
3: $w = v\mathbf{K_{st}}$.
4: Compute with Vincent's algorithm max, the maximum for the strong stochastic ordering of the rows of $\mathbf{P}$.
5: $c = max - v$
6: $z = c\mathbf{K_{st}}$
7: r[N] = 1
8: **for** $j = 2$ to $N-1$ **do**
9: $\quad s = \mathbf{P}[j,*]\mathbf{K_{st}}$
10: $\quad h[j] = Max_{z[k]>0} \frac{s[k]-w[k]}{z[k]}$
11: $\quad r[j] = max(h[j], r[j-1])$
12: **end for**

Intuitively, the algorithm works as follows: the bounding matrix has the same first row and the same last row as the bound provided by Vincent's algorithm. The first row is equal to the first row of the initial matrix (Line 1 of the algorithm) and the last row is max, the maximum for the stochastic ordering. Row i (for i between 2 and $N-1$) is chosen such that:

1. it is larger using the stochastic ordering than the corresponding row in the initial matrix (i.e. $\mathbf{P}[i,*]$),
2. it is a linear combination of $\mathbf{P}[1,*]\mathbf{K_{st}}$ and $max\mathbf{K_{st}}$.

Finally, Line 11 makes the matrix stochastic monotone.

Example 4 *Consider matrix* $\mathbf{P}$*:*

$$\mathbf{P} = \begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.3 & 0.1 \\ 0.1 & 0.2 & 0.3 & 0.2 & 0 & 0.2 \\ 0.2 & 0.1 & 0.1 & 0.2 & 0.3 & 0.1 \\ 0.1 & 0 & 0.5 & 0 & 0.2 & 0.2 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.4 & 0.2 & 0.1 \end{bmatrix}$$

Vector v *is* $[0.1, 0.1, 0.2, 0.2, 0.3, 0.1]$*. We use Vincent's algorithm to compute* $max = [0, 0.1, 0.3, 0.1, 0.2, 0.3]$*. Vector* c *is* $[-0.1, 0, 0.1, -0.1, -0.1, 0.2]$*.*

Multiplying c *by* $\mathbf{K_{st}}$ *we obtain vector* z *from which we know that we have to compute* h *using columns* 2*,* 3*,* 5 *and* 6*. For instance, we get:*

$$h[2] = max(0/0.1, -0.1/0.1, -0.2/0.1, 0.1/0.2) = 0.5,$$

$$h[3] = max(-0.1/0.1, -0.1/0.1, 0/0.1, 0/0.1) = 0.0,$$

and

$$h[4] = max(0/0.1, 0.1/0.1, 0/0.1, 0.1/0.2) = 1.0,$$

To be complete and allow to compare the results with Algorithm 4 we also report the values of $h[5] = 1$ *and* $h[6] = 0$*. Finally* $r = [0, 0.5, 0.5, 1, 1, 1]$*. The decomposition of the upper bound is:*

$$\begin{aligned} & e^T[0.1, 0.1, 0.2, 0.2, 0.3, 0.1] \\ + \quad & [0, 0.5, 0.5, 1, 1, 1]^T[-0.1, 0, 0.1, -0.1, -0.1, 0.2], \end{aligned}$$

or with an explicit form:

$$\mathbf{M}_1 = \begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.3 & 0.1 \\ 0.05 & 0.1 & 0.25 & 0.15 & 0.25 & 0.2 \\ 0.05 & 0.1 & 0.25 & 0.15 & 0.25 & 0.2 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \end{bmatrix}.$$

For the sake of comparison, we also give the bounds provided by Vincent's algorithm:

$$\begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0 & 0.3 & 0.2 & 0.2 & 0.2 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \end{bmatrix}$$

One can easily check that Vincent's bound is tighter but it does not have any structure which helps for the numerical resolution of transient or steady-state distributions.

7.1 A generalisation based on column partition

We modify Algorithm 3 to obtain a size k decomposition. Lines 1 and 2 are kept unchanged as they compute the first term (i.e. $v.e^T$) which is the same for both decompositions. We just add index i to consider the r_i in the size k decomposition.

We introduce $\mathcal{C}$, the set of indices such that $c[i] \neq 0$. We assume that we can find a partition $\{\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \ldots, \mathcal{C}^{(k)}\}$ of $\mathcal{C}$ in k subsets such that:

- We define the vectors c_i as follows:

$$c_i[j] = c[j] \;\; if \;\; j \in \mathcal{C}^{(i)} \;\; and \;\; c_i[j] = 0 \;\; otherwise. \tag{5}$$

- $\sum_{m \in \mathcal{C}^{(j)}} c_j[m] = 0$, for all j between 1 and k.
- Vectors $c_j \mathbf{K_{st}}$ are non negative, for all j between 1 and k.
- The elements in any subset of the partition are contiguous: If j and k are in the same subset $C^{(i)}$, then there is no state l in $C^{(m)}$ with $m \neq i$ such that $j \leq l \leq k$.

Property 15 *If the partition satisfies the constraints then we have the following property for all state m and index i: If $m \notin C^{(i)}$, then $(c_i \mathbf{K_{st}})[m] = 0$.*

Proof: We have only two cases as the states in a subset are contiguous:

- m is larger than any element of $C^{(i)}$,
- m is smaller than any element of $C^{(i)}$.

In the first case, $(c_i \mathbf{K_{st}})[m] = \sum_{l \geq m} c_i[l] = 0$. Indeed, all the entries considered in the summation are not in $C^{(i)}$ and therefore $c_i[l] = 0$ for all l.

For the second case, we write $(c_i\mathbf{K_{st}})[m] = \sum_l c_i[l] - \sum_{l<m} c_i[l]$. The first term is equal to 0 by assumption. The last term is also equal to 0 because all the l considered in the summation are outside $C^{(i)}$.

As a consequence the lower elements of the subsets of the partition are states j such that $c\mathbf{K_{st}}[j] = 0$. This gives a simple heuristic to build a partition. □

Example 5 *Consider again vector c equal to* $[-0.1, 0, 0.1, -0.1, -0.1, 0.2]$*, already used in a previous example. Vector* $c\mathbf{K_{st}}$ *is* $[0, 0.1, 0.1, 0, 0.1, 0.2]$*. One can check that* $c\mathbf{K_{st}}[1] = c\mathbf{K_{st}}[4] = 0$*. This gives the subsets* $\{1, 2, 3\}$ *and* $\{4, 5, 6\}$ *to define the partition.*

We compute vectors h_i using the same technique as in Algorithm 3 but we compute the maximum over the subset $\mathcal{C}^{(i)}$ to get h_i. As a consequence, we have $h = \max_i(h_i)$. This is the main difference with the previous algorithm.

Algorithm 4 Monotone Upper Bound in $\mathcal{A}_k$

Require: $\mathbf{P}$.
Ensure: monotone upper bound $\mathbf{Q}$.

$v = \mathbf{P}[1, *]$
for $i = 1$ to k **do**
  $r_i[1] = 0$
end for
Compute with Vincent's algorithm max, the maximum for the strong stochastic ordering of the rows of $\mathbf{P}$.
$w = v\mathbf{K_{st}}$.
$c = max - v$
$z = c\mathbf{K_{st}}$
Find a partition of $\mathcal{C}$ into k subsets
Define the vectors c_i as in Eq. 5.
for $j = 2$ to $N - 1$ **do**
  $s_j = \mathbf{P}[j, *]\mathbf{K_{st}}$
  for $i = 1$ to k **do**
    $h_i[j] = Max_{z[l]>0, l\in\mathcal{C}^{(i)}} \frac{s[l]-w[l]}{z[l]}$
    $r_i[j] = max(h_i[j], r_i[j-1])$
  end for
end for
for $i = 1$ to k **do**
  $r_i[N] = 1$
end for

Property 16 *For any stochastic matrix* $\mathbf{P}$*, Algorithm 4 has the following properties:*

1. $c_i c_j^T = 0$ *for* $i \neq j$*,*
2. *it computes a size* k *decomposition of a monotone upper bound of matrix* $\mathbf{P}$*,*
3. *This bounding matrix is stochastically smaller than the matrix computed by Algorithm 3,*

Proof: Let $\mathbf{P}$ be the initial matrix, and let $\mathbf{Q}$ be the matrix associated to the decomposition we want to build with Algorithm 4.

- Point 1) is a simple consequence of the definition of c_i. Remember that $c_i[m] = 0$ if $m \notin \mathcal{C}^{(i)}$. As subsets $\mathcal{C}^{(i)}$ form a partition of $\mathcal{C}$, we always have $c_i[m]c_j[m] = 0$ if $i \neq j$.

- First by construction, the matrix is stochastically monotone, as vectors r_i and c_i satisfy the constraints of Property 14.

  We need to show that $\mathbf{P} \preceq_{st} \mathbf{Q}$. This is equivalent to $\mathbf{PK_{st}} \preceq_{el} \mathbf{QK_{st}}$. For $1 \leq j \leq N$, we have $\mathbf{Q}[j,*]\mathbf{K_{st}} = v\mathbf{K_{st}} + \sum_{i=1}^{k} r_i[j]c_i\mathbf{K_{st}}$.

  For $j = 1$, by line 2 of the algorithm, we have $r_i[0] = 0$, $\forall i$, so $\mathbf{Q}[1,*]\mathbf{K_{st}} = v\mathbf{K_{st}} = \mathbf{P}[1,*]\mathbf{K_{st}}$.

  Similarly, for $j = N$, we get $\mathbf{Q}[N,*]\mathbf{K_{st}} = v\mathbf{K_{st}} + \sum_{i=1}^{k} c_i\mathbf{K_{st}} = v\mathbf{K_{st}} + c\mathbf{K_{st}} = max\mathbf{K_{st}} \succeq_{el} \mathbf{P}[N,*]\mathbf{K_{st}}$.

  Consider now $2 \leq j \leq N-1$. By construction,$\mathbf{Q}[j,*]\mathbf{K_{st}} = v\mathbf{K_{st}} + \sum_{i=1}^{k} r_i[j]c_i\mathbf{K_{st}}$. The entries of vector $c_i\mathbf{K_{st}}$ are non negative and $0 \leq h_i[j] \leq r_i[j]$. Therefore: $\mathbf{Q}[j,*]\mathbf{K_{st}} \succeq_{el} v\mathbf{K_{st}} + \sum_{i=1}^{k} h_i[j]c_i\mathbf{K_{st}}$.

  Now remark that:

$$h_i[j] = Max_{l \in C^{(i)}, (c\mathbf{K_{st}})[l]>0} \frac{s[l] - w[l]}{c\mathbf{K_{st}})[l]}$$

  A simple consequence of Property 15 is that for any state l and index i, if $l \in C^{(i)}$, then $(c\mathbf{K_{st}})[l] = (c_i\mathbf{K_{st}})[l]$. Therefore

$$h_i[j] = Max_{l \in C^{(i)}, (c_i\mathbf{K_{st}})[l]>0} \frac{s[l] - w[l]}{c_i\mathbf{K_{st}})[l]}$$

  Note that we now have the same definition of h as in Algorithm 3. We use the same argument as in [8] to conclude the proof.

$$\sum_{i=1}^{k} h_i[j]c_i\mathbf{K_{st}} \succeq_{el} s_j - w.$$

  Finally,

$$\mathbf{Q}[j,*]\mathbf{K_{st}} \succeq_{el} w + s_j - w = \mathbf{P}[j,*]\mathbf{K_{st}}.$$

- Point 3) is a simple consequence of the construction of the bound. As for all i, $c_i\mathbf{K_{st}}$ is non negative and r_i is not decreasing and due to the definition of the c_i, we have:

$$\sum_i r_i^T c_i \preceq_{st} s^T \sum_i c_i$$

  where vector s is $\max_i(r_i)$ component-wise. We have already noticed that $h = \max_i(h_i)$. As a consequence $r = \max_i(r_i)$. Finally

$$\sum_i r_i^T c_i \preceq_{st} r^T c$$

This completes the proof of this point.

□

Example 6 *Let us consider the same matrix as in Example 4 and use the following partition:* $\{\{1,3\},\{4,5,6\}\}$. *Note that, as* $c[2]$ *equal to* 0*, we do not have to consider it in the set. As* h_1 *and* h_2 *are not computed on the same set, we get now:*

$$h_1[2]=0,\ h_1[3]=0,\ h_1[4]=1, h_1[5]=1,\ h_1[6]=0,$$

and,

$$h_2[2]=1/2,\ h_2[3]=0,\ h_2[4]=1/2, h_2[5]=1,\ h_2[6]=0,$$

Finally,

$$r_1[1]=0,\ r_1[2]=0,\ r_1[3]=0,\ r_1[4]=1, r_1[5]=1,\ r_1[6]=1,$$

and,

$$r_2[1]=0,\ r_2[2]=1/2,\ r_2[3]=1/2,\ r_2[4]=1/2, r_2[5]=1,$$

$$r_2[6]=1.$$

And the bounding matrix in an explicit form is:

$$\mathbf{M}_2=\begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.3 & 0.1 \\ 0.05 & 0.1 & 0.25 & 0.15 & 0.25 & 0.2 \\ 0.05 & 0.1 & 0.25 & 0.15 & 0.25 & 0.2 \\ 0 & 0.1 & 0.3 & 0.15 & 0.25 & 0.2 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \end{bmatrix}.$$

It can be easily checked than the bound is slightly tighter than $\mathbf{M}_1$ *obtained with Algorithm 3.*

One may expect that we can improve the tightness of the bound if we increase the rank of the decomposition. We were not able to prove such a general result due to the lack of structure between the partitions.

7.2 A generalisation based on row selection

We give now another algorithm to compute an $\preceq_{st}$-upper bound for a stochastic matrix $\mathbf{P}$. Let $\mathbf{Q}$ be the smallest $\preceq_{st}$-upper bound for $\mathbf{P}$, obtained by 1. The bound given by Algorithm 3 is the matrix with a

size 1 decomposition whose rows are convex combinations of the first and the last row of matrix $\mathbf{Q}$. The main idea of Algorithm 5 is to construct an upper bound with a decomposition of size k, using additional $k-1$ rows of matrix $\mathbf{Q}$.

Algorithm 5 Monotone Upper Bound in $\mathcal{A}_k$

Require: $\mathbf{P}$; $1 = l_1 < l_2 < \ldots < l_k < l_{k+1} = N$.
Ensure: monotone upper bound $\mathbf{R}$.

Compute $\mathbf{Q}$ with Algorithm 1.
for $i = 1$ to $k+1$ **do**
  $v_i = \mathbf{Q}[l_i, *]$
end for
for $i = 1$ to k **do**
  $c_i = v_{i+1} - v_i$
  $z = c_i \mathbf{K_{st}}$
  $w = v_i \mathbf{K_{st}}$
  for $j = 1$ to l_i **do**
    $r_i[j] = 0$
  end for
  for $j = l_i + 1$ to $l_{i+1} - 1$ **do**
    $s = \mathbf{P}[j, *]\mathbf{K_{st}}$
    $h_i[j] = Max_{z[m]>0} \frac{s[m]-w[m]}{z[m]}$
    $r_i[j] = max(h_i[j], r_i[j-1])$
  end for
  for $j = l_{i+1}$ to N **do**
    $r_i[j] = 1$
  end for
end for

Property 17 *For a stochastic matrix* $\mathbf{P}$ *let* $\mathbf{Q}$ *be the bound obtained by Algorithm 1, and* $\mathbf{R}$ *the bound obtained by Algorithm 5.*

1. $\mathbf{R}$ *is a monotone upper bound of matrix* $\mathbf{P}$ *with a decomposition of size* k.
2. $\mathbf{R}$ *is stochastically smaller than the matrix computed by Algorithm 3.*
3. *Rows* $1 = l_1 < l_2 < \ldots < l_k < l_{k+1} = N$ *satisfy* $\mathbf{R}[l_i, *] = \mathbf{Q}[l_i, *]$.
4. *The bounding matrix* $\mathbf{R}'$ *obtained using* $l' = \{l'_1, \ldots, l'_{k'+1}\}$ *such that* $l \subset l'$, *is stochastically smaller than* $\mathbf{R}$. *For* $l = \{1, \ldots, N\}$, $\mathbf{R} = \mathbf{Q}$.

Proof: Note that $v_{i+1} = v_i + c_i = v_1 + \sum_{j \le i} c_i$. Algorithm 5 consist of applying Algorithm 3 to k successive blocks of matrix $\mathbf{P}$, where block i contains rows l_i to l_{i+1}. The proof follows now directly from properties of Algorithm 3. □

Example 7 *Let us consider the same matrix as in Example 4. Set* $k = 2$ *and* $l = (1, 3, 6)$. *We get*

$$c_1 = [0, 0, 0, 0, -0.1, 0.1] \quad \textit{and} \quad c_2 = [-0.1, 0, 0, 0, 0, 0.1].$$

The bound is:

$$\mathbf{M}_3 = \begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \end{bmatrix}$$

For $k = 3$ and $l = (1, 3, 4, 6)$ we obtain Vincent's (optimal $\preceq_{st}$) bound:

$$\mathbf{M}_4 = \begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0 & 0.3 & 0.2 & 0.2 & 0.2 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0 & 0.1 & 0.3 & 0.1 & 0.2 & 0.3 \end{bmatrix}.$$

Remark 5 *It can be easily checked that:*

- $\mathbf{P} \preceq_{st} \mathbf{M}_2 \preceq_{st} \mathbf{M}_1$. *The algorithm based on column partition improves the bound.*
- $\mathbf{P} \preceq_{st} \mathbf{M}_4 \preceq_{st} \mathbf{M}_3 \preceq_{st} \mathbf{M}1$. *The algorithm based on row selection improves the bound.*
- *Both assertions* $\mathbf{M}_3 \preceq_{st} \mathbf{M}2$ *and* $\mathbf{M}_2 \preceq_{st} \mathbf{M}_3$ *are false. In general, we cannot compare the results obtained by the new algorithms.*

8 Conclusion

Two important points need to be investigated now. First, as mentioned in the presentation of class $\mathbf{C^G}$ matrices, an algorithm with a low complexity to compute the distribution of the first passage time has been proved in [8]. We still have to find a generalization of this algorithm if the matrix has a decomposition of size k. Second, we have presented two algorithms to find bounding matrices with a decomposition of size k. We must further investigate to understand which algorithm is the more accurate and how to chose the best partitions of the rows or the columns. Finally, we will continue to explore the algorithmic questions on Markov chain models in biology and reliability to find better solutions based on a low rank decomposition.

9 Acknowledgments

Some parts of these works have ween carried on during a cooperation between CNRS and CNRST. This research is also supported by a grant ANR 12 MONU-0019 (project MARMOTE).

References

[1] O. Abu-Amsha and J. M. Vincent. An algorithm to bound functionals on Markov chains with large state space. In *4th INFORMS Conference on Telecommunications*, Boca Raton, Floride, E.U, 1998. INFORMS.

[2] D. Achlioptas and F. McSherry. Fast computation of low rank approximations. In *Proc. of the 33rd Annual Symposium on Theory of Computing*, pages 611–618, 2001.

[3] M. Ben Mamoun, A. Busic, and N. Pekergin. Generalized class C Markov chains and computation of closed-form bounding distributions. *Probability in the Engineering and Informational Science*, 21:235–260, 2007.

[4] M. Ben Mamoun and N. Pekergin. Closed-form stochastic bounds on the stationary distribution of Markov chains. *Probability in the Engineering and Informational Sciences*, 16(4):403–426, 2002.

[5] M. Ben Mamoun and N. Pekergin. Model checking of infinite state space Markov chains by stochastic bounds. In K. Al-Begain, A. Heindl, and M. Telek, editors, *Analytical and Stochastic Modeling Techniques and Applications, 15th International Conference, ASMTA 2008, Nicosia, Cyprus*, volume 5055 of *Lecture Notes in Computer Science*, pages 264–278. Springer, 2008.

[6] M. Ben Mamoun, N. Pekergin, and S. Younès. Class C Markov chains and transient analysis. In *Positive systems Theory and Application conference*, volume 341 of *Lecture Notes in Control and Information Sciences*, pages 177–184. Springer, 2006.

[7] M. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[8] A. Busic and N. Pekergin. Closed form absorption time bounds. In K. Wolter, editor, *Formal Methods and Stochastic Models for Performance Evaluation, Fourth European Performance Engineering Workshop, EPEW 2007, Berlin, Germany, September 27-28, 2007*, volume 4748 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2007.

[9] K. Deng and H. Dayu. Model reduction of markov chains via low-rank approximation. In *American Control Conference (ACC), 2012*, pages 2651–2656, 2012.

[10] J.-M. Fourneau and N. Pekergin. An algorithmic approach to stochastic bounds. In *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, volume 2459 of *LNCS*, pages 64–88. Springer, 2002.

[11] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low rank approximations. In *Proc. of the 39th Symposium on Fondations of Computer Science*, pages 370–378, 1998.

[12] G. Golub and C. V. Loan. *Matrix computations*. John Hopkins University Press, London, Fourth Edition, 2013.

[13] J. Keilson and A. Kester. Monotone matrices and monotone markov processes. *Stochastic Processes and Their Applications*, 5:231–241, 1977.

[14] J. Kleinberg. Authorative sources in a hyperlinked environment. *JACM*, 46:604–632, 1999.

[15] D. Knuth. *The Art of Computed Programming: Semi-Numerical Algorithms, V2*. Addison-Wesley, 1981.

[16] J. C. S. Lui and R. R. Muntz. Computing bounds on steady state availability of repairable computer systems. *Journal of the ACM*, 41(4):676–707, 1994.

[17] J. C. S. Lui, R. R. Muntz, and D. Towsley. Computing performance bounds of fork-join parallel programs under a multiprocessing environment. *IEEE Transactions on Parallel and Distributed Systems*, 9(3):295–311, 1998.

[18] A. Muller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks*. Wiley, New York, NY, 2002.

[19] M. Shaked and J. G. Shantikumar. *Stochastic Orders and their Applications*. Academic Press, San Diego, CA, 1994.