Stochastic Bounds for Partially Generated Markov Chains: An Algebraic Approach

Ana Bušić[1] and Jean-Michel Fourneau[1,2]

[1] INRIA Grenoble - Rhône-Alpes
51, Av. J. Kuntzmann, 38330 Montbonnot, France
[2] PRiSM, Université de Versailles-St-Quentin
45, Av. des Etats-Unis, 78035 Versailles, France

Abstract. We propose several algorithms to obtain bounds based on Censored Markov Chains to analyze partially generated discrete time Markov chains. The main idea is to avoid the generation of a huge (or even infinite) state space and to truncate the state space during the visit. The approach is purely algebraic and provides element-wise and stochastic bounds for the CMC.

1 Introduction

Even if it is simple to model systems with Markov chains, the analysis of such chains is still a hard problem when they do not exhibit some regularity or symmetry which allow analytical techniques or lumping. Furthermore, some transitions rates may be unknown. An alternative approach is to compute bounds on the rewards we need to check against requirements. We first bound the steady-state or transient distributions at time t. We define the elementary reward for all states and compute the expected reward by a simple summation of the product of the elementary rewards by the state probabilities. The main difficulty is to obtain a bound of the steady state or transient distributions. The key idea is to derive a smaller chain which provides a bound. Several algorithms have been proposed to obtain some stochastic bounds on Discrete Time Markov Chains (DTMC). Most of these algorithms have used the lumpability approach to reduce the size of the chain [1, 6, 7, 16]. Stochastic comparison of DTMC can also be applied when some transition probabilities are unknown [2, 10]. Recently a new approach based on Censored Markov Chain (CMC) have been proposed [4, 8] to deal with large or infinite DTMC. Here we present new algorithms based on CMC when only some parts of the matrix are known. Indeed, when the state space is very large or infinite, we have to truncate the chain during the generation and only some parts of the matrix are computed [5]. CMCs provide an efficient way to describe such a truncated generation.

Consider a DTMC $\{X_t : t = 0, 1, \ldots\}$ with a finite state space S. Suppose that $S = A \cup A^c$, $A \cap A^c = \emptyset$. Suppose that the successive visits of X_t to A take place at time epochs $0 \leq t_0 < t_1 < \ldots$ Then the chain $\{X^A_u = X_{t_u}, u = 0, 1, \ldots\}$ is called the censored chain with censoring set A [17]. Let Q denote the transition probability matrix of chain X_t. Consider the partition of the state space to obtain a block description of Q:

$$Q = \begin{array}{c} \\ \end{array}\begin{bmatrix} Q_{AA} & Q_{A*} \\ Q_{*A} & Q_{**} \end{bmatrix} \begin{array}{l} A \\ A^c \end{array}$$

The censored chain only observes the states in A. Assume that the chain is ergodic (it may be finite or infinite). It can be proved [17] that the stochastic matrix of the censored chain is:

$$S_{AA} = Q_{AA} + Q_{A*}(I - Q_{**})^{-1}Q_{*A} = Q_{AA} + Q_{A*}\left(\sum_{i=0}^{\infty}(Q_{**})^i\right)Q_{*A}. \qquad (1)$$

The second term of the right-hand side represents the probabilities of paths that return to set A through states in A^c.

Censored Markov chains have also been called restricted or watched Markov chains. They are also strongly related to the theory of stochastic complement [11]. Note that it is not necessary for censored Markov chains to be ergodic and we can study for instance the absorption time [8]. However, we assume in this paper that the chains are ergodic and that the CMC is finite.

In many problems, initial chain Q can be large or even infinite or some transition rates may be unknown. Therefore, it is difficult or even impossible to compute $(I - Q_{**})^{-1}$ to finally get S_{AA}. Deriving bounds of S_{AA} from Q_{AA} and from some information on the other blocks is thus an interesting alternative approach. Note that we may have various interesting cases:

- *Partial Generation:* Q_{**} and Q_{*A} are difficult to build or contain unknown rates while Q_{AA} is easy to compute from the specifications.
- *Complete Generation:* all the blocks are easy to compute but $(I - Q_{**})$ is difficult to invert because of its size.

Our major concern is the difficulty to obtain a complete description of the block Q_{*A} from a high-level specification framework such as a Stochastic Process Algebra model or a set of stochastic equations. They provide a continuous time Markov chain which can be uniformized to obtain a DTMC. All these formalisms are very efficient in describing forward transitions (i.e. transitions from state x to any state of the chain). Thus we can easily obtain the rows of matrix Q_{AA}. The first problem is to find the reachable state space to define A^c (the set of reachable states which are not censored). Remember that the reachability problem is a time consuming question in many high-level specification languages. Let us now turn to the block Q_{*A}. We have typically four problems :

- *Reachability.* Even with a tensor based approach it is hard to find the reachable state space. For a Stochastic Automata Network (SAN) we define a product space which contains the reachable state space [14]. It is simple to find the column of the matrix associated to a SAN or to any tensor based model with an algorithm developed by Sbeity in [9], but we build the column of the matrix for the product state space which is a superset of the reachable state space. We must remove the rows associated to non reachable states to obtain block Q_{*A} which is difficult because of the reachability problem.

– *Inversion of a stochastic equation.* For some high-level specification languages, transitions out of x are described by a stochastic recurrence equation $X_{n+1} = f(X_n, U)$, where U is a random variable. This is typically the case when one describes queues. But the transitions entering state x are described by function f^{-1}. This problem is very similar to the computation of the inverse function of a distribution. For some functions f, it is well known in simulation that the complexity of the computation of the inverse of function f is highly dependent on the state where we invert the function.
– *Infinite State Space.* When the chain is infinite, Q_{*A} has an infinite number of rows and it is not possible to generate all of them.
– *Unknown rates.* Assume that some rates of a transition from y to x in A are unknown. Assume that y is not a censored state. Then the transition from y to x is in column x of Q_{*A}. Here we consider that if a rate is missing in a column, the complete column of the block is unknown.

In [15], Truffet has proposed a two-level algorithm for Nearly Completely Decomposable (NCD) chains by using aggregation and stochastic ordering to compute bounding distributions. In [13], Truffet's approach has been combined with state reordering to improve the accuracy of a component-wise probability bounding algorithm. In these works, before employing the aggregation of blocks, the slack probabilities $\beta_i = 1 - \sum_{j \in A} Q[i,j]$, $i \in A$ (which are small due to the NCD structure) are included in the last column for the upper bound and in the first column for the lower bound. Clearly Truffet's approach is optimal when only the block Q_{AA} has been computed. Indeed the bound is tight in that case. For general Markov chains (i.e. not NCD), Dayar, Pekergin, and Younes proposed recently an algebraic approach to dispatch slack probabilities when blocks Q_{AA} and Q_{*A} are known [4]. In this paper, we will refer to their algorithm as DPY. DPY exhibits a desirable feature: under some algebraic conditions it provides the exact result (see [4] for a proof and some examples):

Property 1. If block Q_{*A} has rank 1, then the bound given by DPY is exact.

However, DPY needs both Q_{AA} and Q_{*A} to be known. Bounds of S_{AA} have also been derived in [8] in a completely different way by applying graph algorithms. This approach requires that Q_{AA} is computed and some parts (not necessary all elements) of Q_{*A}, Q_{**} and Q_{A*} are known. Here we propose a new approach and several algorithms which require less information on the blocks.

The paper is organized as follows. In Sect. 2 we present a brief introduction to stochastic bounds and CMC. Sect. 3 is devoted to the main concept and the first algorithm we obtained when Q_{*A} is known and satisfies some algebraic constraints. We also show that this first algorithm also gives exact result when Q_{*A} has rank 1. In Sect. 4, we present new algorithms when some columns of Q_{*A} are unknown, based on various assumptions on Q_{*A}. Due to the number of algorithms proposed, we do not have enough space to present a large example. Instead we show on small matrices how the algorithms perform.

2 Some Fundamental Results on Stochastic Bounds

We give first the definition of strong stochastic ordering of random variables on a finite state space $\{1, \ldots, n\}$. Let X and Y be two random variables with probability vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ ($p_k = P(X = k), q_k = P(Y = k), \ \forall k$). Throughout the paper, all the vectors are column vectors, $\boldsymbol{v}^t$ denotes a transposed vector, and $\preceq_{el}$ element-wise comparison of two vectors (or matrices).

Definition 1. *$X \preceq_{st} Y$ if $\sum_{k=j}^{n} p_k \leq \sum_{k=j}^{n} q_k, \ \forall j$.*

Let $\{X_t\}_{t\geq 0}$ and $\{Y_t\}_{t\geq 0}$ be two DTMC with transition probability matrices P and Q. Then we say that $\{X_t\}_{t\geq 0} \preceq_{st} \{Y_t\}_{t\geq 0}$ if $X_t \preceq_{st} Y_t$ for all $t \geq 0$. Sufficient conditions for comparison of two DTMC are given by the following classical theorem [12]:

Theorem 1. *$\{X_t\}_{t\geq 0} \preceq_{st} \{Y_t\}_{t\geq 0}$ if $X_0 \preceq_{st} Y_0$ and there exists a transition probability matrix R such that:*

- *$P \preceq_{st} R \preceq_{st} Q$, i.e. $P[i,*] \preceq_{st} R[i,*] \preceq_{st} Q[i,*], \ \forall i$ (comparison),*
- *$R[i-1,*] \preceq_{st} R[i,*], \ \forall i > 1$ (monotonicity).*

Furthermore, if both chains are ergodic, then $\pi_P \preceq_{st} \pi_Q$ (where π_P and π_Q are the steady-state distributions).

The above conditions can be easily checked algorithmically. Furthermore, it is also possible to construct a monotone upper bound for an arbitrary stochastic matrix P [3]. We define operators r and v as in [3]:

- $r(P)[i,j] = \sum_{k=j}^{n} P[i,k], \ \forall i, j,$
- $v(P)[i,j] = \begin{cases} r(P)[1,j], & \text{if } i = 1 \\ \max\{v(P)[i-1,j], r(P)[i,j]\}, & \text{if } i > 1 \end{cases}, \ \forall j.$

Remark 1. It it worthy to remark that $P \preceq_{st} Q$ is equivalent to $r(P) \preceq_{el} r(Q)$.

Proposition 1. *(Vincent's algorithm [3]) Let P be any stochastic matrix and $Q = r^{-1}v(P)$, where r^{-1} denotes the inverse of r. Then Q is $\preceq_{st}$-monotone et $P \preceq_{st} Q$, therefore (by Theorem 1) Q is a transition probability matrix of an upper bounding DTMC. Furthermore, if $P_1 \preceq_{st} P_2$, then $r^{-1}v(P_1) \preceq_{st} r^{-1}v(P_2)$.*

2.1 CMC and Stochastic Bounds

Let us now consider CMC and it's transition probability matrix given by (1):

$$S_{AA} = Q_{AA} + \underbrace{Q_{A*}(I - Q_A)^{-1}Q_{*A}}_{Z}$$

Z is a sub-stochastic matrix which shows how the missing transition probability must be added to Q_{AA} to obtain S_{AA}. Truffet proposed in [15] an algorithm for the case when we know only the block Q_{AA}. An upper bound for S_{AA} can be obtained by adding first the slack of probability mass to the last column, and then applying operator $r^{-1}v$ to compute a monotone bound. More formally, let θ be the operator which transforms a sub-stochastic matrix M into a stochastic matrix by adding in the last column of M all the probability missing in M:

$$\theta(M)[i,j] = \begin{cases} M[i,j], & \text{if } j < n \\ M[i,j] + \beta_i, & \text{if } j = n \end{cases}, \ \forall i,$$

where $\beta_i = 1 - \sum_{j=1}^{n} M[i,j]$, $\forall i$. Of course, if M is stochastic, then $\theta(M) = M$. The upper bound for S_{AA} proposed in [15] is given by $r^{-1}v(\theta(Q_{AA}))$.

Remark 2. Similarly, a monotone lower bound for S_{AA} is given by $r^{-1}w(\phi(Q_{AA}))$, where operator ϕ adds the slack of probability mass to the first column and

$$w(P)[i,j] = \begin{cases} r(P)[n,j], & \text{if } i = n \\ \min\{w(P)[i+1,j], r(P)[i,j]\}, & \text{if } i < n \end{cases}, \ \forall j.$$

Notice that operator $r^{-1}w$ corresponds to the maximal st-monotone lower bound.

Suppose now that we have some partial information on Z, given by positive matrices L and U such that $L \preceq_{el} Z \preceq_{el} U$. Furthermore, we know that $Z\boldsymbol{e} = \boldsymbol{\beta}$, where $\boldsymbol{e}$ denotes a vector with all components equal to 1.

In the following we describe how we can use matrices L and U to construct an upper and a lower stochastic bound for S_{AA} that is more accurate than Truffet's bound (that uses only the information contained in Q_{AA}). Once we have obtained bounds on S_{AA}, we apply Vincent's algorithm to check the monotonicity and analyze the resulting chain to get steady-state or transient distributions (see [4, 8] for more details). Here we only present the computations of the bounds of S_{AA} under various assumptions on the knowledge of Q_{*A}.

2.2 Bounds for a Family of Positive Matrices

In a recent paper [10], Haddad and Moreaux have proposed an algorithm to build a stochastic bound from two element-wise bounding matrices. More precisely, they are interested in absorption time and they only consider finite transient Markov chains. They assume that they do not know exactly the stochastic matrix P they need to analyze (because some terms are difficult to compute), but they know two positive matrices L and U such that $L \preceq_{el} P \preceq_{el} U$. In [10], matrix P is supposed transient (i.e. the last state is absorbing). Let $\mathcal{P}_{L,U}$ be the set of stochastic matrices which satisfy these constraints. They derived an algorithm to compute the smallest (in the st sense) transient matrix in $\mathcal{P}_{L,U}$. We give in Algorithm 1 a generalization of that algorithm: a) we don't need to have any absorbing state; b) we consider positive matrices (not necessarily stochastic). Let $\mathcal{M}_{L,U,\boldsymbol{\beta}}$ be a family of positive matrices given by element-wise upper and lower bounds L and U, and a positive vector of normalization constants $\boldsymbol{\beta}$:

$$\mathcal{M}_{L,U,\boldsymbol{\beta}} = \{M \ : \ L \preceq_{el} M \preceq_{el} U \text{ and } M\boldsymbol{e} = \boldsymbol{\beta}\}.$$

We will use the following operator: $\ell(M)[i,j] = \sum_{k=1}^{j} M[i,k]$, $\forall i,j$.

Proposition 2. *Algorithm 1 computes matrices $\overline{M}$ and $\underline{M}$ in $\mathcal{M}_{L,U,\boldsymbol{\beta}}$ such that:*

$$r(\underline{M}) \preceq_{el} r(M) \preceq_{el} r(\overline{M}), \ \forall M \in \mathcal{M}_{L,U,\boldsymbol{\beta}}.$$

Proof. Notice that $L \preceq_{el} M \preceq_{el} U$ implies $r(L)[i,j] \leq r(M)[i,j] \leq r(U)[i,j]$ and $\ell(L)[i,j] \leq \ell(M)[i,j] \leq \ell(U)[i,j], \forall i,j$. The proof follows easily from the fact that $r(M)[i,j] = \beta_j - \ell(M)[i,j-1]$, $\forall M \in \mathcal{M}_{L,U,\boldsymbol{\beta}}$. We omit the technical details. □

Algorithm 1: r-maximal ($\overline{M}$) and r-minimal ($\underline{M}$) elements for a family of positive matrices $\mathcal{M}_{L,U,\boldsymbol{\beta}} = \{M \ : \ L \preceq_{el} M \preceq_{el} U \text{ and } M\boldsymbol{e} = \boldsymbol{\beta}\}$.

```
Input      : $\boldsymbol{\beta}$ - positive vector; $L, U$ - positive matrices : $0 \preceq_{el} L \preceq_{el} U \preceq_{el} \boldsymbol{e}\boldsymbol{\beta}^t$
Notation : $n$ - number of lines; $m$ - number of columns
for $i = 1$ to $n$ do
    for $j = m$ downto 2 do
        $\overline{H}[i,j] = \min\{r(U)[i,j], \beta_j - \ell(L)[i,j-1]\}$;
        $\underline{H}[i,j] = \max\{r(L)[i,j], \beta_j - \ell(U)[i,j-1]\}$;
    end
    $\overline{H}[i,1] = \beta_j$; $\underline{H}[i,1] = \beta_j$;
    $\overline{M} = r^{-1}(\overline{H})$; $\underline{M} = r^{-1}(\underline{H})$;
end
```

Let us now go back to CMC problem. We assumed that we know matrix Q_{AA} and element-wise lower and upper bounds L and U for $Z = Q_{A*}(I - Q_A)^{-1}Q_{*A}$. Denote by $\boldsymbol{\beta} = Q_{AA}\boldsymbol{e}$. Algorithm 1 gives matrices $\overline{M}$ and $\underline{M}$ such that $r(\underline{M}) \preceq_{el} r(Z) \preceq_{el} r(\overline{M})$. Denote by $\underline{S} = Q_{AA} + \underline{M}$ and $\overline{S} = Q_{AA} + \overline{M}$.

Theorem 2. *Matrices $\underline{S}$ and $\overline{S}$ satisfy:*

$$\phi(Q_{AA}) \preceq_{st} \underline{S} \preceq_{st} S_{AA} \preceq_{st} \overline{S} \preceq_{st} \theta(Q_{AA}).$$

Proof. Follows directly from Remark 1. □

Since operator $r^{-1}v$ (resp. $r^{-1}w$) preserves the $\preceq_{st}$-comparison, the upper (resp. lower) bound obtained by taking into account the partial information on Z is more accurate than the bounds proposed in [15]. In the following section we propose how to compute element-wise lower and upper bounds for Z.

3 Element-Wise Bounds for Matrix Z

First let us define the following binary relation on positive real vectors:

Definition 2. *Let $\boldsymbol{x}, \boldsymbol{y}$ be two positive real vectors. Vector $\boldsymbol{y}$ supports $\boldsymbol{x}$ if there exist $\alpha > 0$ and $\gamma \geq 0$ such that $\alpha\, \boldsymbol{y} \preceq_{el} \boldsymbol{x} \preceq_{el} (\alpha + \gamma)\, \boldsymbol{y}$.*

Remark 3. Vector $\boldsymbol{y}$ supports vector $\boldsymbol{x}$ if and only if they have the same support (the non zero elements have the same indices in both vectors). Note that if vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are colinear we have $\gamma = 0$.

Property 2. Relation “supports” is reflexive, symmetric and transitive. Thus the binary relation “supports” is an equivalence relation.

Suppose that vector $\boldsymbol{y} \neq \mathbf{0}$ supports vector $\boldsymbol{x}$ and define $\alpha_{\boldsymbol{x},\boldsymbol{y}}$ and $\gamma_{\boldsymbol{x},\boldsymbol{y}}$ as follows:

$$\alpha_{\boldsymbol{x},\boldsymbol{y}} = \min_{k\,:\,y_k>0} \frac{x_k}{y_k}, \quad \gamma_{\boldsymbol{x},\boldsymbol{y}} = \max_{k\,:\,y_k>0} \frac{x_k}{y_k} - \alpha_{\boldsymbol{x},\boldsymbol{y}}. \tag{2}$$

Lemma 1. *Constants $\alpha_{\boldsymbol{x},\boldsymbol{y}}$ and $\gamma_{\boldsymbol{x},\boldsymbol{y}}$ satisfy $\alpha_{\boldsymbol{x},\boldsymbol{y}}\,\boldsymbol{y} \preceq_{el} \boldsymbol{x} \preceq_{el} (\alpha_{\boldsymbol{x},\boldsymbol{y}} + \gamma_{\boldsymbol{x},\boldsymbol{y}})\,\boldsymbol{y}$. Furthermore, $\alpha_{\boldsymbol{x},\boldsymbol{y}} = \frac{1}{\alpha_{\boldsymbol{y},\boldsymbol{x}}+\gamma_{\boldsymbol{y},\boldsymbol{x}}}$ and $\gamma_{\boldsymbol{x},\boldsymbol{y}} = \frac{1}{\alpha_{\boldsymbol{y},\boldsymbol{x}}} - \alpha_{\boldsymbol{x},\boldsymbol{y}}$.*

Proof. Follows directly from (2). Note that if vector $\boldsymbol{y}$ supports vector $\boldsymbol{x}$ then $\alpha_{\boldsymbol{x},\boldsymbol{y}} > 0$. By symmetry of relation "supports", we also have $\alpha_{\boldsymbol{y},\boldsymbol{x}} > 0$. $\square$

3.1 Main Idea

We introduce here the main idea behind the algorithms for element-wise bounds for matrix Z on a very simple case. The following assumptions will be relaxed in Sect. 4.

Assumption 1. *We assume in this section that all columns of Q_{*A} are in the same class of equivalence for relation "supports" or are null. Furthermore, we suppose that there is at least one non null column.*

In the following, let us note by $C_i(M)$ the i-th column of a matrix M. Let $\boldsymbol{v}$ be any vector that belongs to the same equivalence class as the non null columns of matrix Q_{*A}. Thus, for all i, $C_i(Q_{*A})$ supports $\boldsymbol{v}$ or $C_i(Q_{*A}) = \mathbf{0}$.

For non null columns denote by $\alpha_i := \alpha_{C_i(Q_{*A}),\boldsymbol{v}}$ and $\gamma_i := \gamma_{C_i(Q_{*A}),\boldsymbol{v}}$. For columns i such that $C_i(Q_{*A}) = \mathbf{0}$ we will define $\alpha_i := 0$ and $\gamma_i := 0$ to simplify the formulas. Note that $||\boldsymbol{\alpha}||_1 = \sum_i \alpha_i > 0$ and $||\boldsymbol{\gamma}||_1 = \sum_i \gamma_i \geq 0$.

We will derive simple component-wise upper and lower bounds for columns of matrix Z in Algorithm 2. We use the following trivial property:

Property 3. $C_i(Z) = Q_{A*}(I - Q_{**})^{-1} C_i(Q_{*A})$.

Lemma 2. $\frac{\boldsymbol{\beta}}{||\boldsymbol{\alpha}||_1 + ||\boldsymbol{\gamma}||_1} \preceq_{el} Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v} \preceq_{el} \frac{\boldsymbol{\beta}}{||\boldsymbol{\alpha}||_1}$.

Proof. We have $\alpha_j \boldsymbol{v} \preceq_{el} C_j(Q_{*A}) \preceq_{el} (\alpha_j + \gamma_j)\boldsymbol{v}, \forall j$, which implies $\alpha_j Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v} \preceq_{el} C_j(Z) \preceq_{el} (\alpha_j + \gamma_j) Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}$. Now by summation for all columns j we obtain $||\boldsymbol{\alpha}||_1 Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v} \preceq_{el} \boldsymbol{\beta} \preceq_{el} (||\boldsymbol{\alpha}||_1 + ||\boldsymbol{\gamma}||_1) Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}$. $\square$

Lemma 3. *Assume that vector $\boldsymbol{v}$ supports column i. Then:*

$$\boldsymbol{\beta}\frac{\alpha_i}{||\boldsymbol{\alpha}||_1 + ||\boldsymbol{\gamma}||_1} \preceq_{el} C_i(Z) \preceq_{el} \boldsymbol{\beta}\frac{\alpha_i + \gamma_i}{||\boldsymbol{\alpha}||_1}.$$

Proof. By definition of constants α_i and γ_i we have:

$$\alpha_i \boldsymbol{v} \preceq_{el} C_i(Q_{*A}) \preceq_{el} (\alpha_i + \gamma_i)\boldsymbol{v}.$$

By Property 3, $\alpha_i Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v} \preceq_{el} C_i(Z) \preceq_{el} (\alpha_i + \gamma_i) Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}$, which together with Lemma 2 gives the result. $\square$

Algorithm 2: Element wise upper (U) and lower (L) bounds for matrix Z

```
Input      : $\boldsymbol{\beta} = \boldsymbol{e} - Q_{AA}\boldsymbol{e}$; matrix $Q_{*A}$
Notation : $m$ - right-most non null column of $Q_{*A}$
$\boldsymbol{v} = C_m(Q_{*A})$;
$\boldsymbol{\alpha} = \boldsymbol{0}$; $\boldsymbol{\gamma} = \boldsymbol{0}$;
foreach non null column $j$ of $Q_{*A}$ do
    $\alpha_j = \min_{k\,:\,v_k>0} \frac{Q_{*A}[k,j]}{v_k}$; $\gamma_j = \max_{k\,:\,v_k>0} \frac{Q_{*A}[k,j]}{v_k} - \alpha_j$;
end
$U = \frac{1}{||\boldsymbol{\alpha}||_1}\boldsymbol{\beta}(\boldsymbol{\alpha}+\boldsymbol{\gamma})^t$; $L = \frac{1}{||\boldsymbol{\alpha}||_1+||\boldsymbol{\gamma}||_1}\boldsymbol{\beta}\boldsymbol{\alpha}^t$;
```

Remark 4. We can use Algorithm 1 to obtain matrices $\overline{M}$ and $\underline{M}$ such that $r(\underline{M}) \preceq_{el} r(M) \preceq_{el} r(\overline{M})$, $\forall M \in \mathcal{M}_{L,U,\boldsymbol{\beta}}$. However, element-wise bounds L and U given by Algorithm 2 are rank 1 matrices. Therefore, it is sufficient to compute only two row vectors $\overline{\boldsymbol{w}}$ and $\underline{\boldsymbol{w}}$, the maximal and minimal elements of the family $\mathcal{M}_{U',L',\boldsymbol{\beta}'}$ for $U' = \frac{1}{||\boldsymbol{\alpha}||_1}(\boldsymbol{\alpha}+\boldsymbol{\gamma})^t$, $L' = \frac{1}{||\boldsymbol{\alpha}||_1+||\boldsymbol{\gamma}||_1}\boldsymbol{\alpha}^t$ and $\boldsymbol{\beta}' = 1$. Vectors $\overline{\boldsymbol{w}}$ and $\underline{\boldsymbol{w}}$ can be computed by Algorithm 1 (as a special case of a matrix with only 1 row). Finally, matrices $\overline{M}$ and $\underline{M}$ are obtained as $\overline{M} = \boldsymbol{\beta}\overline{\boldsymbol{w}}$ and $\underline{M} = \boldsymbol{\beta}\underline{\boldsymbol{w}}$.

Example 1. Let us consider a matrix Q_{*A} given as follows:

$$Q_{*A} = \begin{bmatrix} 0.2 & 0.1 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.0 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.2 & 0.0 & 0.1 \end{bmatrix}$$

All the columns are null or support column 4. $\boldsymbol{\alpha}^t = [2, 1, 0, 1]$ and $\boldsymbol{\gamma}^t = [0, 1, 0, 0]$. Thus the upper bound is $(\boldsymbol{\alpha}+\boldsymbol{\gamma})/4$ and the lower bound is $\boldsymbol{\alpha}/5$. The st-upper bound is $\overline{\boldsymbol{w}} = [0.4, 0.35, 0, 0.25]$ and the st-lower bound is $\underline{\boldsymbol{w}} = [0.5, 0.3, 0, 0.2]$. It is interesting to remark that DPY [4] gives the same st-upper bound for this example.

Corollary 1. *If Q_{*A} is of rank 1, then Algorithm 2 gives the exact result.*

Proof. If Q_{*A} is of rank 1, then $\gamma_i = 0$, $\forall i$, so in Algorithm 2, $U = L = Z$. □

4 Finding Bounds Using Incomplete Information

In this section we present different algorithms based on complete or partial information on the block Q_{*A}. Also, we consider here the general case where we can have more than one equivalence class (i.e. when Assumption 1 is not satisfied). In Algorithm 3 we assume less restrictive assumptions and in Algorithm 4 we iterate an approach based on Algorithm 3. In Algorithm 5 we apply Algorithm 2 on groups of columns.

These algorithms apply even if we do not know all the columns of matrix Q_{*A}. It is still possible to obtain in this case an element-wise and an st-upper bound. Lower bounds are much harder to obtain and only some methods will provide lower bounds under stronger assumptions. Finally, in the last two approaches we assume that the sum of the columns of Q_{*A} is known while some columns are unknown. Indeed, for some modeling frameworks it is possible to compute easily the sum of probabilities of a set of events even if the exact transitions are unknown.

Note that it may be possible to use more than one method for some problems and we can combine them to improve the accuracy using a very simple argument:

Lemma 4. *Let $\boldsymbol{\beta}\boldsymbol{\lambda}_1^t$ and $\boldsymbol{\beta}\boldsymbol{\lambda}_2^t$ be two element-wise upper bounds. Then $\boldsymbol{\beta}(\min\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2\})^t$ is a more accurate upper bound (the min operator is element-wise).*

Finally, the preprocessing step of each algorithm is to compute $\boldsymbol{\beta} = \boldsymbol{e} - Q_{AA}\boldsymbol{e}$.

4.1 Less Constrained Matrices

Denote by m the right-most non null column of Q_{*A} and let $\boldsymbol{v}_m = C_m(Q_{*A})$.

Assumption 2. *We assume that some columns (not all of them) of Q_{*A} support $\boldsymbol{v}_m$. We can decompose Q_{*A} as follows:*

$$C_i(Q_{*A}) = \alpha_i \boldsymbol{v}_m + W_i, \tag{3}$$

with $\mathbf{0} \preceq_{el} W_i$ and $\alpha_i \geq 0$. This decomposition is always possible. If column $C_i(Q_{*A})$ supports column $\boldsymbol{v}_m$ we get $\alpha_i > 0$ as usual, otherwise we may have $\alpha_i = 0$ or a positive α_i if we are lucky. So we decompose Q_{*A} into a rank 1 matrix and a positive matrix W whose columns are W_i. The stochastic complement is:

$$Q_{AA} + Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}_m\boldsymbol{\alpha}^t + Q_{A*}(I - Q_{**})^{-1}W.$$

Let $Z_1 = Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}_m\boldsymbol{\alpha}^t$ and $Z_2 = Q_{A*}(I - Q_{**})^{-1}W$. Z_1 and Z_2 are positive matrices.

Some results obtained in the last section for more constrained matrices are still true.

Lemma 5. $C_m(Z) \preceq_{el} \boldsymbol{\beta}\frac{1}{||\boldsymbol{\alpha}||_1}$

Proof. The decomposition implies that: $Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}_m||\boldsymbol{\alpha}||_1 \preceq_{el} \boldsymbol{\beta}$. In the decomposition $W_m = \mathbf{0}$. Thus: $C_m(Z) = Q_{A*}(I - Q_{**})^{-1}\boldsymbol{v}_m$. Combining both relations, we get the result. □

It is also possible to find an upper bound for all the columns which support column m.

Lemma 6. *Assume that column i supports column m. Then $C_i(Z) \preceq_{el} \boldsymbol{\beta}\frac{\alpha_i + \gamma_i}{||\boldsymbol{\alpha}||_1}$. But when the column does not support $\boldsymbol{v}_m$, the upper bound is $\boldsymbol{\beta}$.*

Proof. From Lemma 5, we get $C_m(Z) \preceq_{el} \boldsymbol{\beta}\frac{1}{||\boldsymbol{\alpha}||_1}$, and if column i supports column m we have $C_i(Z) \preceq_{el} C_m(Z)(\alpha_i + \gamma_i)$. Combining both inequalities we get the first result. Finally it is sufficient to remark that $C_i(Z) \preceq_{el} \boldsymbol{\beta}$. $\square$

We introduce two operators, q (quotient) and R (reminder) defined as follows. Let $\boldsymbol{x}$ and $\boldsymbol{y} \neq \boldsymbol{0}$ be two positive vectors:

$$q(\boldsymbol{x}, \boldsymbol{y}) = \min_{k: y_k > 0} \left\{ \frac{x_k}{y_k} \right\}, \quad R(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} - q(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{y}. \tag{4}$$

Algorithm 3: Upper bounds for Z when Assumption 1 is not satisfied

```
1. Consider the right-most non null column of $Q_{*A}$, say $\boldsymbol{v}_m$. Set $\alpha_m = 1$, $\gamma_m = 0$
   and $\alpha_i = 0$ for all index $i > m$.
2. For all columns $i$ between 1 to $m$ check if it supports $\boldsymbol{v}_m$:
   (a) If YES compute $\alpha_i$ and $\gamma_i$.
   (b) If NO perform the decomposition described in (3) to obtain $\alpha_i$:
       $\alpha_i = q(C_i(Q_{*A}), C_m(Q_{*A}))$ and $W = R(C_i(Q_{*A}), C_m(Q_{*A}))$.
3. The upper bound is $\frac{1}{||\boldsymbol{\alpha}||_1}\boldsymbol{\beta}(\alpha_i + \gamma_i)^t$ for columns $i$ which support $\boldsymbol{v}_m$, $\boldsymbol{0}$ for the
   null columns and $\boldsymbol{\beta}$ for the remaining ones.
```

Remark 5. All the columns where $\alpha_i > 0$ are used to bound column m. α_i is positive when column i supports column m but this is not necessary. For instance $[1,1,1]^t$ does not support $[1,1,0]^t$ but we obtain $\alpha_i = 1$. Note however we are only able to obtain a non trivial bound for columns which support column m.

Theorem 3. *Algorithm 3 provides an upper bound when we know all columns of Q_{*A}. If some columns of Q_{*A} are unknown, Algorithm 3 provides an upper bound based on known columns. The upper bound for unknown columns is $\boldsymbol{\beta}$.*

Proof. Lemmas 5 and 6 give the answer for known columns. Unknown columns i do not support the column m (the last *known* non null column) so the corresponding $\alpha_i = 0$. $\square$

The following lemma gives a bound of all the colinear columns (it is a simple consequence of Lemma 3).

Lemma 7. *Let i be the index of a column of Q_{*A} which is colinear to $\boldsymbol{v}_m$, then we have $C_i(Z) \preceq_{el} \boldsymbol{\beta}\frac{\alpha_i}{||\boldsymbol{\alpha}||_1}$*

Proof. For a column i colinear to column m, we have $\gamma_i = 0$. $\square$

4.2 Iteration

We decompose the columns according to their equivalence class and we perform a modified version of Algorithm 3 on each class. Then an upper bound for Z can be obtained by taking element-wise minimum of upper bounds obtained for each equivalence class (see Lemma 4).

Algorithm 4: Iteration

```
1. Decompose the columns of $Q_{*A}$ according to the equivalence relation “support”.
2. Upper bounds for columns which are equal to $\mathbf{0}$ are equal to $\mathbf{0}$.
3. For all (non null) equivalence classes:
   (a) Let $\Delta$ be the set of index of the columns that belong to that class.
   (b) Let $\boldsymbol{v}$ be the right-most non null vector in $\Delta$.
   (c) For all $i$, $\alpha_i = q(C_i(Q_{*A}), \boldsymbol{v})$. (Note that for $i \in \Delta$, $\alpha_i = \alpha_{C_i(Q_{*A}),\boldsymbol{v}}$.)
   (d) For all $i \in \Delta$, compute $\gamma_i = \gamma_{C_i(Q_{*A}),\boldsymbol{v}}$.
   (e) Compute $\alpha = \sum_i \alpha_i$.
   (f) The upper bound for column $i \in \Delta$ is $\boldsymbol{\beta}\frac{\alpha_i+\gamma_i}{\alpha}$.
```

Theorem 4. *When all columns of Q_{*A} are known, Algorithm 4 provides an upper bound. Assume now that some columns of Q_{*A} are unknown, Algorithm 4 provides an upper bound based on known columns. The upper bound for unknown columns is $\boldsymbol{\beta}$. (Note that Algorithm 4 gives always better bounds than Algorithm 3.)*

Proof. The proof is similar to the proof of Algorithm 3. It is omitted here for the sake of conciseness. □

4.3 Partial Summations

Assumption 3. *Without loss of generality we assume that $C_n(Q_{*A}) \neq \mathbf{0}$. The main assumption is the following: we can find a partition of $\{1, \ldots, n\}$ into k subsets Γ_1, ..., Γ_k such that for all set index j, $\sum_{i\in\Gamma_j} C_i(Q_{*A})$ supports $C_n(Q_{*A})$. Without loss of generality we assume that $\Gamma_k = \{n\}$. Let us denote by α_j and γ_j the coefficient of the support relation for set Γ_j. Clearly we have: $\alpha_k = 1$ and $\gamma_k = 0$.*

Algorithm 5 consists in building a new matrix where the columns are summed up for all index in the same subset in the partition. This new matrix satisfies the assumptions of Algorithm 2.

Algorithm 5: Partial summations

```
1. Find a partition satisfying the constraints.
2. Sum up the columns of $Q_{*A}$ according to the partition to obtain a new matrix
   with $k$ columns.
3. Apply Algorithm 2 on this matrix to obtain some $\alpha_j$ and $\gamma_j$ for all set index $j$.
4. The element-wise upper bound for an arbitrary column $i$ is $\boldsymbol{\beta}\frac{\alpha_j+\gamma_j}{||\boldsymbol{\alpha}||_1}$ where $j$ in
   the index of the set which contains $i$.
5. The st-upper bound for an arbitrary column $i$ is $\boldsymbol{\beta}\frac{\alpha_j+\gamma_j}{||\boldsymbol{\alpha}||_1}$ if $i$ is the largest
   element of set $\Gamma_j$ and 0 otherwise.
```

Theorem 5. *Algorithm 5 provides an upper bound when all columns are known. Assume now that some columns of Q_{*A} are unknown, Algorithm 5 provides an upper bound based on the known columns. The upper bound for the unknown column is $\boldsymbol{\beta}$.*

Proof. Step 3 and the bound computed for the sum are exactly the same as in Algorithm 2 and we can apply the results we have already proved. Thus $\sum_{i \in \Gamma_j} C_i(Z) \preceq_{el} \boldsymbol{\beta} \frac{\alpha_j + \gamma_j}{||\boldsymbol{\alpha}||_1}$. Step 4 simply states that each element has the same upper bound than the sum. In Step 5 the st-bound is computed from the element-wise upper bound with Algorithm 1. □

4.4 If the Sum of Columns is Known

Assumption 4. *We assume that $\sum_{i=1}^{n} C_i(Q_{*A}) = \boldsymbol{\sigma}$ is known. Only some columns are known. For all the known columns (say with index i) that are not equal to $\mathbf{0}$, we use operators q and R in (4) to get $\boldsymbol{\sigma} = q_i C_i(Q_{*A}) + W_i$, where W_i is non negative.*

Note that it is not necessary to know matrix Q_{*A} to compute $\boldsymbol{\sigma}$, it can be obtained from a high-level specification of the model.

Lemma 8. *As all the columns are non negative (even if they are unknown...) we clearly have $q_i \geq 1$.*

Theorem 6. *Consider $\boldsymbol{\sigma}$ and an arbitrary column index k. Use operators q and R in (4) with input arguments $\boldsymbol{\sigma}$ and $C_k(Q_{*A})$ to obtain q_k and W_k. $C_k(Z)$ is element-wise upper bounded by $\frac{\boldsymbol{\beta}}{q_k}$.*

Proof. We clearly have:

$$\boldsymbol{\beta} = \sum_i C_i(Z) = Q_{A*}(I - Q_{**})^{-1} \sum_i C_i(Q_{*A}) = Q_{A*}(I - Q_{**})^{-1} \boldsymbol{\sigma}. \quad (5)$$

As $\boldsymbol{\sigma} = q_k C_k(Q_{*A}) + W_k$, and as W_k, Q_{A*}, and $(I - Q_{**})^{-1}$ are non negative, we get: $Q_{A*}(I - Q_{**})^{-1} C_k(Q_{*A}) \preceq_{el} \frac{\boldsymbol{\beta}}{q_k}$. As $Q_{A*}(I - Q_{**})^{-1} C_k(Q_{*A}) = C_k(Z)$ we prove the theorem. □

It is even possible to find a bound if we are not able to compute exactly $\boldsymbol{\sigma}$. Assume that we are able to compute $\boldsymbol{\delta}$ such that $\boldsymbol{\delta} \preceq_{el} \boldsymbol{\sigma}$. This is typically the case when we have a high level description of the chain and we are able to classify the transitions according to their destination set.

Theorem 7. *Consider $\boldsymbol{\delta}$ and an arbitrary column index k. Use operators q and R in (4) with parameters $\boldsymbol{\delta}$ and $C_k(Q_{*A})$ to obtain q'_k and W'_k. If $q'_k \geq 1$, column k of Z is element-wise upper bounded by $\frac{\boldsymbol{\beta}}{q'_k}$.*

Proof. As $\boldsymbol{\delta} \preceq_{el} \boldsymbol{\sigma}$, we have $q'_k \leq q_k$ and we apply the former theorem. □

4.5 Examples

We illustrate algorithms proposed in this section on small numerical examples.

Algorithm 3. Let

$$Q_{AA} = \begin{bmatrix} 0.1 & 0.3 & 0.2 & 0.1 \\ 0.1 & 0.4 & 0.2 & 0.0 \\ 0.2 & 0.1 & 0.5 & 0.2 \\ 0.2 & 0.0 & 0.4 & 0.0 \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.0 \\ 0.4 \end{bmatrix} \quad Q_{*A} = \begin{bmatrix} 0.1 & 0.0 & 0.1 & 0.2 \\ 0.0 & 0.1 & 0.0 & 0.0 \\ 0.2 & 0.2 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 & 0.0 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$

Let us consider the columns of Q_{*A}. Clearly, column 3 supports column 4 and $\alpha_3 = 0.5$, $\gamma_3 = 0.5$. Column 2 does not support column 4 and α_2 in the decomposition is 0. Finally column 1 supports column 4 and $\alpha_1 = 0.5$ $\gamma_i = 1.5$. So $||\boldsymbol{\alpha}||_1 = 2$. We find upper bounds for columns 1, 3 and 4 which are respectively $2\boldsymbol{\beta}/2$, $\boldsymbol{\beta}/2$ and $\boldsymbol{\beta}/2$. The upper bound for Z is thus $\boldsymbol{\beta}[1, 1, 1/2, 1/2]$. A strong stochastic bound for Z is $\boldsymbol{\beta}[0, 0, 1/2, 1/2]$ (see Remark 4). The bound provided by DPY is $\boldsymbol{\beta}[0, 1/4, 1/4, 1/2]$, so DPY is better for this example.

Now assume that we are not able to compute column 2.

$$Q_{*A} = \begin{bmatrix} 0.1 & * & 0.1 & 0.2 \\ 0.0 & * & 0.0 & 0.0 \\ 0.2 & * & 0.1 & 0.1 \\ 0.0 & * & 0.0 & 0.0 \\ 0.0 & * & 0.0 & 0.0 \\ 0.1 & * & 0.1 & 0.1 \end{bmatrix}$$

where $*$ denotes that the value is unknown. We are not able to use DPY because the matrix is unknown. But the st-bound with Algorithm 2 is still $\boldsymbol{\beta}[0, 0, 1/2, 1/2]$.

Now assume that we are not able to compute columns 1 and 2. Again it is not possible to use DPY. We still have a support for column 4 from column 3. But as α_1 is unknown, $||\boldsymbol{\alpha}||_1 = 1.5$. The element-wise upper bound is now $\boldsymbol{\beta}[1, 1, 2/3, 2/3]$ and the st-upper bound is $\boldsymbol{\beta}[0, 0, 1/3, 2/3]$.

Algorithm 4. Let

$$\boldsymbol{\beta} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.0 \\ 0.4 \end{bmatrix} \quad Q_{*A} = \begin{bmatrix} 0.4 & 0.1 & 0.1 & 0.2 \\ 0.0 & 0.1 & 0.1 & 0.0 \\ 0.2 & 0.15 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.1 & 0.0 \\ 0.2 & 0.1 & 0.1 & 0.1 \end{bmatrix} \quad H14 = \begin{bmatrix} 0.4 & 0.2 \\ 0.0 & 0.0 \\ 0.2 & 0.1 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.2 & 0.1 \end{bmatrix} \quad H23 = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.15 & 0.1 \\ 0.0 & 0.0 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$$

We have two classes of equivalence in this example, corresponding to matrices $H14$ (columns 1 and 4) and $H23$. For the first class ($H14$), $\boldsymbol{v} = C_4(Q_{*A})$ and the values of α_i are $[2, 0.5, 0.5, 1]$, so $\alpha = 4$. Since columns 1 and 4 are colinear, $\gamma_1 = \gamma_4 = 0$. The corresponding upper bounds for columns 1 and 4 are: $\boldsymbol{\beta}/2$ and $\boldsymbol{\beta}/4$. Thus the upper bound for Z for this class is $\boldsymbol{\beta}[1/2, 1, 1, 1/4]$. For the second class ($H23$), $\boldsymbol{v} = C_3(Q_{*A})$, the values of α_i are $[0, 1, 0, 1]$, $\alpha = 2$, and $\gamma_2 = 0.5$, $\gamma_3 = 0$. The corresponding upper bound for Z is $\boldsymbol{\beta}[1, 3/4, 1/2, 1]$. The final upper bound for Z is then $\boldsymbol{\beta}[1/2, 3/4, 1/2, 1/4]$ and the corresponding strong stochastic bound is $\boldsymbol{\beta}[0, 1/4, 1/2, 1/4]$. For the sake of comparison, the st bound provided by DPY is $\boldsymbol{\beta}[0, 1/2, 1/4, 1/4]$.

Known sum. Let

$$\boldsymbol{\beta} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.0 \\ 0.4 \end{bmatrix} \quad Q_{*A} = \begin{bmatrix} 0.1 & 0.0 & 0.1 & 0.2 \\ 0.0 & 0.1 & 0.0 & 0.0 \\ 0.2 & 0.2 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 & 0.0 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix} \quad \boldsymbol{\sigma} = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.6 \\ 0.0 \\ 0.1 \\ 0.4 \end{bmatrix}$$

Then we use operators q and R in (4) to obtain the ratios q_i. Some values of the rests W_i are omitted for the sake of conciseness. We have: $q_1 = 3$, $W_1^t = [0.1, 0.1, 0, 0, 0.1, 0.1]$, $q_2 = 1$, $q_3 = 4$, and $q_4 = 2$. And finally the bounding matrix is $\boldsymbol{\beta}[1/3, 1, 1/4, 1/2]$ and the st bound is $\boldsymbol{\beta}[0, 1/4, 1/4, 1/2]$.

Assume now that we are not able to compute the second column of Q_{*A}. We have: $\boldsymbol{\delta}^t = [0.4, 0, 0.4, 0, 0, 0.3]$. Then $q_1 = 2$, $W_1^t = [0.2, 0, 0, 0, 0, 0.1]$, $q_3 = 3$ and $q_4 = 2$. As we cannot compute another bound for the second column, we keep the simplest one (i.e. 1), so the element-wise bound is $\boldsymbol{\beta}[1/2, 1, 1/3, 1/2]$ and the st bound is $\boldsymbol{\beta}[0, 1/6, 1/3, 1/2]$.

5 Concluding Remarks

In this paper we have presented several algorithms to obtain bounds of the transition probability matrix S_{AA} of a CMC. These methods apply even if the initial chain is infinite. The CMC is obtained after a partial generation of the state space. More precisely, we only know Q_{AA} and some columns of Q_{*A}. When the whole block Q_{*A} is known, it is possible to use both DPY and the algorithms presented here. On many examples DPY and Algorithm 2 provide the same result for the upper bound and lower bound. We have also proved that they give an exact result if matrix Q_{*A} has rank 1. Note however that in general Algorithm 2 requires strong assumptions on the matrix and we have also found some matrices where the bound is worse than the bound provided by DPY even if we have no proof that DPY is always better. However, the aim of our algorithms is to find bounds of S_{AA} when Q_{*A} is only partially known or when Q_{*A} is infinite. In both cases we cannot apply DPY. The algorithms presented here (except Algorithm 2) still apply if some columns of matrix Q_{*A} are unknown. Thus they may be used even when some part of the matrix (or the models) are difficult to compute. We do not have comparison of results for these algorithms (except Algorithm 4 which is always better than Algorithm 3). Indeed, they are not based on the same assumptions. When several algorithms can be applied, the best solution is to use all of them and combine the element-wise upper bounds.

Acknowledgments. This work was partially supported by ANR-05-BLAN-0009-02 SMS and ANR-06-SETIN-002 Checkbound.

References

1. A. Busic and J.-M. Fourneau. Bounds for point and steady-state availability: An algorithmic approach based on lumpability and stochastic ordering. In *EPEW 2005*, LNCS 3670, pages 94–108. Springer, 2005.
2. A. Busic, J.-M. Fourneau, and N. Pekergin. Worst case analysis of batch arrivals with the increasing convex ordering. In *EPEW 2006*, LNCS 4054, pages 196–210. Springer, 2006.
3. T. Dayar, J.-M. Fourneau, and N. Pekergin. Transforming stochastic matrices for stochastic comparison with the st-order. *RAIRO-RO*, 37:85–97, 2003.
4. T. Dayar, N. Pekergin, and S. Younes. Conditional steady-state bounds for a subset of states in Markov chains. In *SMCtools '06*. ACM Press, 2006.
5. E. de Souza e Silva and P. Mejiá Ochoa. State space exploration in Markov models. *ACM SIGMETRICS Perform. Eval. Rev.*, 20(1):152–166, 1992.
6. J.-M. Fourneau, M. Lecoz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications*, 386(1):167–185, 2004.
7. 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*, LNCS 2459, pages 64–88, 2002.
8. J.-M. Fourneau, N. Pekergin, and S. Younes. Censoring Markov chains and stochastic bounds. In *EPEW 2007*, LNCS 4748, pages 213–227. Springer, 2007.
9. J.-M. Fourneau, B. Plateau, I. Sbeity, and W. J. Stewart. SANs and lumpable stochastic bounds: Bounding availability. In *Computer System, Network Performance and Quality of Service*. Imperial College Press, 2006.
10. S. Haddad and P. Moreaux. Sub-stochastic matrix analysis for bounds computation - theoretical results. *Eur. Jour. of Op. Res.*, 176(2):999–1015, 2007.
11. C. D. Meyer. Stochastic complementation, uncoupling Markov chains and the theory of nearly reducible systems. *SIAM Review*, 31(2):240–272, 1989.
12. A. Muller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks.* Wiley, New York, NY, 2002.
13. N. Pekergin, T. Dayar, and D. Alparslan. Compenent-wise bounds for nearly completely decomposable Markov chains using stochastic comparison and reordering. *Eur. Jour. of Op. Res.*, 165:810–825, 2005.
14. B. Plateau, J.M. Fourneau, and K.H. Lee. PEPS: a package for solving complex Markov models of parallel systems. In *Proceedings of the 4th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, Spain*, 1988.
15. L. Truffet. Near complete decomposability: Bounding the error by a stochastic comparison method. *Ad. in App. Prob.*, 29:830–855, 1997.
16. L. Truffet. Reduction technique for discrete time Markov chains on totally ordered state space using stochastic comparisons. *Jour. of App. Prob.*, 37(3):795–806, 2000.
17. Y.Q. Zhao and D. Liu. The censored Markov chain and the best augmentation. *Jour. of App. Prob.*, 33:623–629, 1996.