An Algorithmic Approach to Stochastic Bounds

J.M. Fourneau and N. Pekergin

PRiSM, Université de Versailles Saint-Quentin en Yvelines,
45 Av. des Etats Unis, 78000 Versailles, France

Abstract. We present a new methodology based on the stochastic ordering, algorithmic derivation of simpler Markov chains and numerical analysis of these chains. The performance indices defined by reward functions are stochastically bounded by reward functions computed on much simpler or smaller Markov chains. This leads to an important reduction on numerical complexity. Stochastic bounds are a promising method to analyze QoS requirements. Indeed it is sufficient to prove that a bound of the real performance satisfies the guarantee.

1 Introduction

Since Plateau's seminal work on composition and compact tensor representation of Markov chains using Stochastic Automata Networks (SAN), we know how to model Markov systems with interacting components and large state space [29,30, 31]. The main idea of the SAN approach is to decompose the system of interest into its components and to model each component separately. Once this is done, interactions and dependencies among components can be added to complete the model. The stochastic matrix of the chain is obtained after summations and Kronecker (or tensor) products of local components. The benefit of the SAN approach is twofold. First, each component can be modeled much easier compared to the global system. Second, the space required to store the description of components is in general much smaller than the explicit list of transitions, even in a sparse representation. However, using this representation instead of the usual sparse matrix form increases the time required for numerical analysis of the chains [6,15,37,33]. Note that we are interested in performance indices R defined as reward functions on the steady-state distribution (i.e. $R = \sum_i r(i)\pi(i)$) and we do not try to compute transient measures. Thus the numerical computation of the analysis is mainly the computation of the steady-state distribution and then the summation of the elementary rewards $r(i)$ to obtain R. The first step is in general the most difficult because of the memory space and time requirements (see Steward's book [34] for an overview of usual numerical techniques for Markov chains). The decomposition and tensor representation has been generalized to other modeling formalisms as well : Stochastic Petri nets [13], Stochastic Process Algebra [20]. So we now have several well-founded methods to model complex systems using Markov chains with large state space.

Despite considerable works [7,12,15,37], the numerical analysis of Markov chains, is still a very difficult problem when the state space is too large or the eigenvalues badly distributed. Fortunately enough, while modeling high speed networks, it is often sufficient to satisfy the requirements for the Quality of Service (QoS) we expect. Exact values of the performance indices are not necessary in this case and bounding some reward functions is often sufficient.

So, we advocate the use of stochastic bounds to prove that the QoS requirements are satisfied. Our approach differs from sample path techniques and coupling theorem applied to models transformation (see [27] for an example on Fair Queueing delays comparison based on sample-paths), as we only consider Markov chains and algorithmic operations on stochastic matrices. Assume that we have to model a problem using a very large Markov chain. We need to compute its steady-state distribution in order to obtain reward functions (for instance, the cell loss rates for a finite capacity buffer). The key idea of the methodology is to design a new chain such that the reward functions will be upper or lower bounds of the exact reward functions. This new chain is an aggregated or simplified model of the former one. These bounds and the simplification criteria are based on some stochastic orderings applied to Markov processes (see Stoyan [35] and other references therein). As we drastically reduced the state space or the complexity of the analysis, we may now use numerical methods to efficiently compute a bound of the rewards.

Several methods have been proposed to bound rewards : resolution of a linear algebra problem and polyhedra properties by Courtois and Semal [8,9], Markov Decision Process by Van Dijk [38] and various stochastic bounds (see [35,22,32] and references therein). Here we present recent results based on stochastic orders and structure-based algorithms combined with usual numerical techniques. Thus the algorithms we present can be easily implemented inside software tools based on Markov chains. Unlike former approaches which are either analytical or not really constructive, this new approach is only based on simple algorithms. These algorithms can always be applied, even if the quality of the bounds may be sometimes not enough accurate.

We survey the results in two steps : first how to obtain a bounding matrix and a bound of the distributions and in a second step how to simplify the numerical computations. We present several algorithms based on stochastic bounds and structural properties of the chains and some examples to show the effectiveness of the approach. In section 2, we define the “st” and “icx” stochastic orderings and we give the fundamental theorem on the stochastic matrices. We also present Vincent’s algorithm [1] which is the starting point of all the algorithms for the “st” ordering presented here. Then we present, in section 3, several algorithms for “st” bounds based on structures: upper-Hessenberg, lumpability, stochastic complement [26], Class C Matrices. Section 4 is devoted to the analysis of a real problem: the loss rates of a finite buffer with batch arrivals and modulation, Head of Line service discipline and Pushout buffer management. Such systems have been proposed for ATM networks [18]. The example here is only slightly simplified to focus on the algorithmic aspects. The reduction algorithms we have used on this example has divided the state-space by ten for several analysis. Finally, in section 5, we present some algorithms for “icx” ordering.

2 Strong Stochastic Bounds

For the sake of simplicity, we restrict ourselves to Discrete Time Markov Chains (DTMC) with finite state space $E = \{1, \ldots, n\}$ but continuous-time models can be considered after uniformization. Here we restrict ourselves to "st" stochastic ordering. The definitions and results for "icx" ordering are presented in section 5. In the following, n will denote the size of matrix P and $P_{i,*}$ will refer to row i of P.

First, we give a brief overview on stochastic ordering for Markov chains and we obtain a set of inequalities to imply bounds. Then we present a basic algorithm proposed by Vincent and Abuamsha [1] and we explain some of its properties.

2.1 A Brief Overview

Following [35], we define the strong stochastic ordering by the set of non-decreasing functions or by matrix K_{st}.

$$K_{st} = \begin{bmatrix} 1\,0\,0 \ldots 0 \\ 1\,1\,0 \ldots 0 \\ 1\,1\,1 \ldots 0 \\ \vdots\,\vdots\,\vdots\,\ddots\,\vdots \\ 1\,1\,1 \ldots 1 \end{bmatrix}$$

Definition 1 *Let X and Y be random variables taking values on a totally ordered space. Then X is said to be less than Y in the strong stochastic sense, that is, $X <_{st} Y$ if and only if $E[f(X)] \leq E[f(Y)]$ for all non decreasing functions f whenever the expectations exist.*

If X and Y take values on the finite state space $\{1, 2, \ldots, n\}$ with p and q as probability distribution vectors, then X is said to be less than Y in the strong stochastic sense, that is, $X <_{st} Y$ if and only if $\sum_{j=k}^{n} p_j \leq \sum_{j=k}^{n} q_j$ for $k = 1, 2, \ldots, n$, or briefly: $pK_{st} <_{st} qK_{st}$.

Important performance indices such as average population, loss rates or tail probabilities are non decreasing functions. Therefore, bounds on the distribution imply bounds on these performance indices as well. It is important to know that st-bounds are valid pour the transient distributions as well. We do not use this property as we are mainly interested in performance measures on the the steady-state. To the best of our knowledge, such a work has still to be done to link st-bounds and numerical analysis for the computation of transient distributions.

It is known for a long time that monotonicity [21] and comparability of the one step transition probability matrices of time-homogeneous MCs yield sufficient conditions for their stochastic comparison. This is the fundamental result we use in our algorithms. First let us define the st-comparability of the matrix and the st-monotonicity.

Definition 2 *Let P and Q be two stochastic matrices. $P <_{st} Q$ if and only if $PK_{st} \leq QK_{st}$. This can be also characterized as $P_{i,*} <_{st} Q_{i,*}$ for all i.*

Definition 3 *Let P be a stochastic matrix, P is st-monotone if and only if for all u and v, if $u <_{st} v$ then $uP <_{st} vP$.*

Hopefully, st-monotone matrices are completely characterized (this is not the case for other orderings, see [4]).

Definition 4 *Let P be a stochastic matrix. P is $<_{st}$-monotone if and only if $K_{st}^{-1} P K_{st} \geq 0$ component-wise.*

Thus we get:

Property 1 *Let P be a stochastic matrix, P is st-monotone if and only if for all i, $j > i$, we have $P_{i,*} <_{st} P_{j,*}$*

Theorem 1 *Let $X(t)$ and $Y(t)$ be two DTMC and P and Q be their respective stochastic matrices. Then $X(t) <_{st} Y(t), t > 0$, if*

- *$X(0) <_{st} Y(0)$,*
- *st-monotonicity of at least one of the matrices holds,*
- *st-comparability of the matrices holds, that is, $P_{i,*} <_{st} Q_{i,*}\ \forall i$.*

Thus, assuming that P is not monotone, we obtain a set of inequalities on elements of Q :

$$\begin{cases} \sum_{k=j}^{n} P_{i,k} \leq \sum_{k=j}^{n} Q_{i,k} & \forall\ i,j \\ \sum_{k=j}^{n} Q_{i,k} \leq \sum_{k=j}^{n} Q_{i+1,k} & \forall\ i,j \end{cases} \tag{1}$$

2.2 Algorithms

It is possible to derive a set of equalities, instead of inequalities. These equalities provides, once they have been ordered (in increasing order for i and in decreasing order for j in system 2), a constructive way to design a stochastic matrix which yields a stochastic bound.

$$\begin{cases} \sum_{k=j}^{n} Q_{1,k} = \sum_{k=j}^{n} P_{1,k} \\ \sum_{k=j}^{n} Q_{i+1,k} = max(\sum_{k=j}^{n} Q_{i,k}, \sum_{k=j}^{n} P_{i+1,k}) \quad \forall\ i,j \end{cases} \tag{2}$$

The following algorithm [1] constructs an st-monotone upper bounding DTMC Q for a given DTMC P. For the sake of simplicity, we use a full matrix representation for P and Q. Stochastic matrices associated to real performance evaluation problems are usually sparse. And the sparse matrix version of all the algorithms we present here is straightforward. Note that due to the ordering of the indices, the summations $\sum_{j=l}^{n} q_{i-1,j}$ and $\sum_{j=l+1}^{n} q_{i,j}$ are already computed when we need them. And they can be stored to avoid computations. However, we let them appear as summations to show the relations with inequalities 1.

Algorithm 1 Construction of the optimal st-monotone upper bounding DTMC Q:

```
$q_{1,n} = p_{1,n}$;
for $i = 2, 3, \ldots, n$ do $q_{i,n} = \max(q_{i-1,n}, p_{i,n})$; od
for $l = n-1, n-2, \ldots, 1$, do $q_{1,l} = p_{1,l}$;
    for $i = 2, 3, \ldots, n$, do $q_{i,l} = \max(\sum_{j=l}^{n} q_{i-1,j}, \sum_{j=l}^{n} p_{i,l}) - \sum_{j=l+1}^{n} q_{i,j}$; od
od
```

Definition 5 *We denote by $v(P)$ the matrix obtained after application of Algorithm* 1 *to a stochastic matrix P.*

First let us illustrate Algorithm 1 on a small matrix. We consider a 5×5 matrix for $P1$ and we compute matrix Q, and both steady-state distributions.

$$P1 = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.7 & 0.1 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.5 & 0.2 & 0.0 \\ 0.1 & 0.0 & 0.1 & 0.7 & 0.1 \\ 0.0 & 0.2 & 0.2 & 0.1 & 0.5 \end{bmatrix} \qquad Q = v(P1) = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.6 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.5 & 0.1 & 0.1 \\ 0.1 & 0.0 & 0.1 & 0.7 & 0.1 \\ 0.0 & 0.1 & 0.1 & 0.3 & 0.5 \end{bmatrix}$$

Their steady-state distributions are respectively $\pi_{P1} = (0.180, 0.252, 0.184, 0.278, 0.106)$ and $\pi_Q = (0.143, 0.190, 0.167, 0.357, 0.143)$. Their expectations are respectively 1.87 and 2.16 (we assume that the first state has index 0 to compute the reward $f(i) = i$ associated to the expectation). Remember that the strong stochastic ordering implies that the expectation of f on distribution π_{P1} is smaller than the expectation of f on distribution π_Q for all non decreasing functions f.

It may happen that matrix $v(P)$ computed by Algorithm 1 is not irreducible, even if P is irreducible. Indeed due to the subtraction operation in inner loops, some elements of $v(P)$ may be zero even if the elements with the same indices in P are positive. We have derived a new algorithm which try to keep almost all transitions of P in matrix $v(P)$ and we have proved a necessary and sufficient condition on P to obtain an irreducible matrix (the proof of the theorem is omitted for the sake of readability):

Theorem 2 *Let P be an irreducible finite stochastic matrix. Matrix Q computed from P by Algorithm* 2 *is irreducible if and only if every row of the lower triangle of matrix P contains at least one positive element.*

Even if matrix $v(P)$ is reducible, it has one essential class of states and the last state belongs to that class. So it is still possible to compute the steady-state distribution for this class. We do not prove the theorem but we present an example of a matrix $P2$ such that $v(P2)$ is reducible (i.e. states 0, 1 and 2 are transient in matrix $v(P2)$).

$$P2 = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.7 & 0.1 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.5 & 0.2 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.7 & 0.3 \\ 0.0 & 0.2 & 0.2 & 0.1 & 0.5 \end{bmatrix} \qquad Q = v(P2) = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.6 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.5 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.7 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.5 \end{bmatrix}$$

In the following, ϵ is an arbitrary positive value. And we assume that a summation with a lower index larger than the upper index is 0.

Algorithm 2 Construction of an st-monotone upper bounding DTMC without transition deletion:

```
$q_{1,n} = p_{1,n}$;
for $i = 2, 3, \ldots, n$ do $q_{i,n} = \max(q_{i-1,n}, p_{i,n})$; od
for $l = n-1, n-2, \ldots, 1$, do $q_{1,l} = p_{1,l}$;
    for $i = 2, 3, \ldots, n$, do
        $q_{i,l} = \max(\sum_{j=l}^{n} q_{i-1,j}, \sum_{j=l}^{n} p_{i,l}) - \sum_{j=l+1}^{n} q_{i,j}$;
        if $(q_{i,l} = 0)$ and $(p_{i,l} > 0)$ and $(\sum_{j=l+1}^{n} q_{i,j} < 1)$ then
            $q_{i,l} = \epsilon \times (1 - \sum_{j=l+1}^{n} q_{i,j})$
    od
od
```

2.3 Properties

Algorithm 1 has several interesting properties which can be proved using a max-plus formulation [10] which appears clearly in equation 2.

Theorem 3 *Algorithm* 1 *provides the smallest st-monotone upper bound for a matrix* P*: i.e. if we consider* U *another st-monotone upper bounding DTMC for* P *then* $Q <_{st} U$ *[1].*

However bounds on the probability distributions may still be improved. The former theorem only states that Algorithm 1 provides the smallest matrix. We have developed new techniques to improve the accuracy of the bounds on the steady-state π which are based on some transformations on P [10].

We have studied a linear transformation for stochastic matrices $\alpha(P, \delta) = (1-\delta)I + \delta P$, for $\delta \in (0,1)$. This transformation has no effect on the steady-state distribution but it has a large influence on the effect of Algorithm 1. We have proved in [10] that if the given stochastic matrix is not row diagonally dominant, then the steady-state probability distribution of the optimal st-monotone upper bounding matrix corresponding to the row diagonally dominant transformed matrix is better in the strong stochastic sense than the one corresponding to the original matrix. And we have established that the transformation $P/2 + I/2$ provides the best bound for the family of linear transformation we have considered. More precisely:

Theorem 4 *Let* P *be a DTMC of order* n*, and two different values* $\delta_1, \delta_2 \in (0,1)$ *such that* $\delta 1 < \delta 2$*, Then* $\pi_{v(\alpha(P,\delta 1))} <_{st} \pi_{v(\alpha(P,\delta 2))} <_{st} \pi_{v(P)}$.

One may ask if there is an optimal value of δ. When the matrix is row diagonal dominant (RDD), its diagonal serves as a barrier for the perturbation moving from the upper-triangular part to the strictly lower-triangular part in forming $v(P)$.

Definition 6 *A stochastic matrix is said to be row diagonally dominant (RDD) if all of its diagonal elements are greater than or equal to 0.5.*

Corollary 1 *Let P be a DTMC of order n that is RDD. Then $v(P)$ and $v(\alpha(P))$ have the same steady-state probability distribution.*

Corollary 1 implies that one cannot improve the steady-state probability bounds by choosing a smaller δ value to transform an already RDD DTMC. And $\delta = 1/2$ is sufficient to transform an arbitrary stochastic matrix into a RDD one. This first approach was then generalized to transformations based on a set of polynomials which gives better (i.e. more accurate) bounds [5]. Let us first introduce these transformations and their basic properties.

Definition 7 *Let $\mathcal{D}$ be the set of polynomials $\Phi()$ such that $\Phi(1) = 1$, Φ different of Identity, and all the coefficients of Φ are non negative.*

Proposition 1 *Let $\Phi()$ be an arbitrary polynomial in $\mathcal{D}$, then $\Phi(P)$ has the same steady-state distribution than P*

Theorem 5 *Let Φ be an arbitrary polynomial in $\mathcal{D}$, Algorithm 1 applied on $\Phi(P)$ provides a more accurate bound than the steady-state distribution of Q i.e.:*

$$\pi_P <_{st} \pi_{v(\Phi(P))} <_{st} \pi_{v(P)}$$

For a stochastic interpretation of this result and a proof based on linear algebra see [5]. Corollary 1 basically states that the optimal transformation if we restrict ourselves to degree 1 polynomials is $\phi(X) = X/2 + 1/2$. Such a result is still unknown for arbitrary degree polynomials, even if it is clear that the larger the degree of Φ, the more accurate the bound $v(\Phi(P))$. This is illustrated in the example below. Let us consider stochastic matrix $P3$ and we study the polynomials $\phi(X) = X/2 + 1/2$ and $\psi(X) = X^2/2 + 1/2$.

$$P3 = \begin{bmatrix} 0.1 & 0.2 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.2 & 0.3 \\ 0.1 & 0.5 & 0.4 & 0 \\ 0.2 & 0.1 & 0.3 & 0.4 \end{bmatrix}$$

First, let us compute $\phi(P3)$ and $\psi(P3)$.

$$\phi(P3) = \begin{bmatrix} 0.55 & 0.1 & 0.2 & 0.15 \\ 0.1 & 0.65 & 0.1 & 0.15 \\ 0.05 & 0.25 & 0.7 & 0 \\ 0.1 & 0.05 & 0.15 & 0.7 \end{bmatrix} \quad \psi(P3) = \begin{bmatrix} 0.575 & 0.155 & 0.165 & 0.105 \\ 0.08 & 0.63 & 0.155 & 0.135 \\ 0.075 & 0.185 & 0.65 & 0.09 \\ 0.075 & 0.13 & 0.17 & 0.625 \end{bmatrix}$$

Then, we apply operators v to obtain the bounds on matrices :

$$v(\phi(P3)) = \begin{bmatrix} 0.55 & 0.1 & 0.2 & 0.15 \\ 0.1 & 0.55 & 0.2 & 0.15 \\ 0.05 & 0.25 & 0.55 & 0.15 \\ 0.05 & 0.1 & 0.15 & 0.7 \end{bmatrix} \quad v(\psi(P3)) = \begin{bmatrix} 0.575 & 0.155 & 0.165 & 0.105 \\ 0.08 & 0.63 & 0.155 & 0.135 \\ 0.075 & 0.185 & 0.605 & 0.135 \\ 0.075 & 0.13 & 0.17 & 0.625 \end{bmatrix}$$

And,

$$v(P3) = \begin{bmatrix} 0.1 & 0.2 & 0.4 & 0.3 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix}$$

Finally, we compute the steady-state distributions for all matrices:

$$\begin{cases} \pi_{v(P3)} = (0.1, 0.2, 0, 3667, 0.3333) \\ \pi_{v(\phi(P3))} = (0.1259, 0.2587, 0, 2821, 0.3333) \\ \pi_{v(\psi(P3))} = (0.1530, 0.2997, 0, 2916, 0.2557) \\ \pi_{P3} = (0.1530, 0.3025, 0, 3167, 0.2278) \end{cases}$$

Clearly, bounds obtained by polynomial ψ are more accurate than the other bounds.

2.4 Time and Space Complexity

It must be clear at this point that Algorithm 1 builds a matrix Q which is, in general, as difficult as P to analyze. This first algorithm is only presented here to show that inequalities 1 have algorithmic implications. Concerning complexity of Algorithm 1 on sparse matrix, we do not have positive results. Indeed, it may be possible that matrix Q has many more positive elements than matrix P and it may be even completely filled. For instance:

$$P4 = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.1 & 0.1 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \quad Q = v(P4) = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.2 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$

More generally, it is easy to build a matrix P with $3n$ positive elements resulting in a completely filled matrix $v(P)$. Of course the algorithms we survey in the next section provide matrices with structural or numerical properties. Most of them do not suffer the same complexity problem.

3 Structure Based Bounding Algorithms for "st" Comparison

We can also use the two sets of constraints of system 1 and add some structural properties to simplify the resolution of the bounding matrix. For instance, Algorithm 3 provides an upper bounding matrix which is upper-Hessenberg (i.e.

the low triangle except the main sub-diagonal is zero). Therefore the resolution by direct elimination is quite simple. In the following we illustrate this principle with several structures associated to simple resolution methods and present algorithms to build structure based st-monotone bounding stochastic matrices. Most of these algorithms do not assume any particular property or structure for the initial stochastic matrix.

3.1 Upper-Hessenberg Structure

Definition 8 *A matrix H is said to be upper-Hessenberg if and only if $H_{i,j} = 0$ for $i > j + 1$.*

The paradigm for upper-Hessenberg case is the $M/G/1$ queue. The resolution by recursion for these matrices requires $o(m)$ operations [34].

Property 2 *Let P be an irreducible finite stochastic matrix such that every row of the lower triangle of P contains at least one positive element. Let Q be computed from P by Algorithm 3. Then Q is irreducible, st-monotone, upper-Hessenberg and an upper bound for P.*

The proof is omitted. The algorithm is slightly different of Algorithm 2. The last two instructions create the upper-Hessenberg structure. Note that the generalization to block upper-Hessenberg matrices is straightforward.

Algorithm 3 An upper-Hessenberg st-monotone upper bound Q:

```
$q_{1,n} = p_{1,n}$;
for $i = 1, 2, \ldots, n$ do $q_{1,i} = p_{1,i}$; $q_{i+1,n} = \max(q_{i,n}, p_{i+1,n})$; od
for $i = 2, 3, \ldots, n$ do
    for $l = n-1, n-2, \ldots, i$ do
        $q_{i,l} = \max(\sum_{j=l}^{n} q_{i-1,j}, \sum_{j=l}^{n} p_{i,l}) - \sum_{j=l+1}^{n} q_{i,j}$;
        if $(q_{i,l} = 0)$ and $(p_{i,l} > 0)$ and $(\sum_{j=l+1}^{n} q_{i,j} < 1)$ then
            $q_{i,l} = \epsilon \times (1 - \sum_{j=l+1}^{n} q_{i,j})$
    od
    $q_{i,i-1} = 1 - \sum_{j=i}^{n} q_{i,j}$
    for $l = i-2, i-3, \ldots, 1$ do $q_{i,l} = 0$ od
od
```

The application of this algorithm on matrix $P1$ already defined leads to:

$$Q = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.6 & 0.1 & 0.1 & 0.1 \\ 0.0 & 0.3 & 0.5 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.2 & 0.7 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.5 \end{bmatrix}$$

3.2 Lumpability

Ordinary lumpability is another efficient technique to combine with stochastic bounds [36]. Unlike the former algorithms, lumpability implies a state space reduction. The algorithms are based on Algorithm 1 and on the decomposition of the chain into macro-states. Again we assume that the states are ordered according to the macro-state partition. Let r be the number of macro-states. Let $b(k)$ and $e(k)$ be the indices of the first state and the last state, respectively, of macro-state A_k. First, let us recall the definition of ordinary lumpability.

Definition 9 (ordinary lumpability) *Let Q be the matrix of an irreducible finite DTMC, let A_k be a partition of the states of the chain. The chain is ordinary lumpable according to partition A_k, if and only if for all states e and f in the same arbitrary macro state A_i, we have:*

$$\sum_{j \in A_k} q_{e,j} = \sum_{j \in A_k} q_{f,j} \quad \forall \;\; macro-state \;\; A_k$$

Ordinary lumpability constraints are consistent with the st-monotonicity and they provide a simple characterization for matrix Q.

Theorem 6 *Let Q be an st-monotone matrix which is an upper bound for P. Assume that Q is ordinary lumpable for partition A_k and let $Q^{m,l}$ and $P^{m,l}$ be the blocks of transitions from set A_m to set A_l for Q and P respectively, then for all m and l, block $Q^{m,l}$ is st-monotone.*

Indeed, since Q is st-monotone we have:

$$\sum_{j=a}^{n} Q(i,j) \leq \sum_{j=a}^{n} Q(i+1,j) \tag{3}$$

But as Q is ordinary lumpable, if i and $i+1$ are in the same macro-state we have:

$$\sum_{j \in A_r} Q(i,j) = \sum_{j \in A_r} Q(i+1,j) \;\; \forall r$$

So we can subtract in both terms of relation 3 partial sums on the macro state which are all equal due to ordinary lumpability. Therefore, assume that a, i and $i+1$ are in the same macro state A_k, we get

$$\sum_{j \geq a, j \in A_k} Q(i,j) \leq \sum_{j \geq a, j \in A_k} Q(i+1,j)$$

The algorithm computes the matrix column by column. Each block needs two steps. The first step is based on Algorithm 1 while second step modifies the first column of the block to satisfy the ordinary lumpability constraint. More precisely, the first step uses the same relations as Algorithm 1 but it has to take into account that the first row of P and Q may now be different due to the second step. The lumpability constraint is only known at the end of the first step. Recall that ordinary lumpability is due to a constant row sum for the block. Thus after the first step, we know how to modify the first column of the block to obtain a constant row sum. Furthermore due to st-monotonicity, we know that the maximal row sum is reached for the last row of the block. In step 2, we modify the first column of the block taking into account the last row sum. Once a block has been computed, it is now possible to compute the block on the left.

Algorithm 4 Construction of an ordinary lumpable st-monotone upper bounding DTMC Q:

```
$q_{1,n} = p_{1,n}$;
for $x = r, r-1, \ldots, 1$ do
    for $l = e(x)..b(x)$ do $q_{1,l} = \sum_{j=l}^{n} p_{1,l} - \sum_{j=l+1}^{n} q_{1,j}$;
        for $i = 2, 3, \ldots, n$ do
            $q_{i,l} = \max(\sum_{j=l}^{n} q_{i-1,j}, \sum_{j=l}^{n} p_{i,l}) - \sum_{j=l+1}^{n} q_{i,j}$;
        od
        for $y = 1, 2, \ldots, r$ do
            $c = \sum_{j=b(y)}^{e(y)} q_{e(y),j}$ ;
            for $i = b(y), \ldots, e(y)-1$ do $q_{i,b(y)} = c - \sum_{j=b(y)+1}^{e(y)} q_{i,j}$ ; od
        od
    od
od
```

Let us illustrate the two steps on a simple example using matrix $P1$ formerly defined. Assume that we divide the state-space into two macro-states: $(1,2)$ and $(3,4,5)$. We show the first block after the first step (the matrix on the left) and after the second step.

$$\left[\begin{array}{c|ccc} & 0.1 & 0.2 & 0.0 \\ & 0.1 & 0.1 & 0.1 \\ & 0.5 & 0.1 & 0.1 \\ \hline & & & \\ & & & \end{array}\right] \quad \left[\begin{array}{c|ccc} & 0.5 & 0.2 & 0.0 \\ & 0.5 & 0.1 & 0.1 \\ & 0.5 & 0.1 & 0.1 \\ \hline & & & \\ & & & \end{array}\right]$$

This algorithm is used in the next section for the analysis of a mechanism for high speed networks. Most of the algorithms presented here may be applied but the best results, for this particular problem, were found with this last approach.

3.3 Class C Stochastic Matrices

Some stochastic matrices also have a closed form steady-state solution, for instance, the class C matrices defined in [4].

Definition 10 *A stochastic matrix $Q = (q_{i,j})_{1 \le i,j \le n}$ belongs to class C, if for each column j there exists a real constant c_j satisfying the following conditions: $q_{i+1,j} = q_{i,j} + c_j,\ 1 \le i \le n-1$. Since Q is a stochastic matrix, the sum of elements in each row must be equal to 1, thus $\sum_{j=1}^{n} c_j = 0$.*

For instance, the matrix $\begin{bmatrix} 0.45 & 0.15 & 0.4 \\ 0.35 & 0.20 & 0.45 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$ is in class C. It is also st-monotone. These matrices have several interesting properties and we also consider this class for "icx" ordering in section 5. First the steady-state distribution of Q can be computed in linear time:

$$\pi_j = q_{1,j} + c_j \, \frac{\sum_{j=1}^{n} j \; q_{1,j} - 1}{1 - \sum_{j=1}^{n} j \; c_j} \tag{4}$$

The st-monotonicity characterization is also quite simple in this class:

Proposition 2 *Let P be a stochastic matrix belonging to class C. P is st-monotone if and only if $\sum_{k=j}^{n} c_k \geq 0, \quad \forall j \in \{1, \ldots, n\}$.*

The algorithm to obtain a monotone upper bound Q of class C for an arbitrary matrix P has been presented in [4]. First remark that since the upper bounding matrix Q belongs to class C, we must determine its first row $q_{1,j}, \quad 1 \leq j \leq n$, and the columns coefficients $c_j, \quad 1 \leq j \leq n$ rather than all the elements of Q. Within former algorithms the elements of Q are linked by inequalities but now we add the linear relations which define the C class. For instance we have $q_{n,n} = q_{1,n} + n \times c_n$. Therefore we must choose carefully $q_{1,n}$ and c_n to insure that $0 \leq q_{n,n} \leq 1$. Note that x^+ denotes as usual $max(x, 0)$.

Algorithm 5 Construction of a st-monotone upper bounding DTMC Q which belongs to class C:

```
$q_{1,n} = \max_{1 \leq i \leq n-1} \left[ \frac{(n-1)p_{i,n} - (i-1)}{n-i} \right]$
$c_n = \left[ \max_{2 \leq i \leq n} \left( \frac{p_{i,n} - q_{1,n}}{i-1} \right) \right]^+$
for $j = n-1, n-2, \ldots, 2$ do
    $\alpha_j = \max_{2 \leq i \leq n} \left[ \frac{\sum_{k=j}^{n} p_{i,k} - \sum_{k=j}^{n} q_{1,k}}{i-1} \right]$
    $g_i = \frac{n-1}{n-i} \left[ \sum_{k=j}^{n} p_{i,k} - \sum_{k=j+1}^{n} q_{i,k} \right] + \frac{i-1}{n-i} \left[ \sum_{k=j+1}^{n} q_{n,k} - 1 \right]$
    $q_{1,j} = [\max_{1 \leq i \leq n-1} \; g_i]^+$
    $c_j = \max(\frac{-q_{1,j}}{n-1}, \; \alpha_j^+ - \sum_{k=j+1}^{n} c_k)$
od
$q_{1,1} = 1 - \sum_{j=2}^{n} q_{1,j}$
```

Again consider an example: let $P5$ be a matrix which does not belong to class C, and Q its upper bounding matrix computed through algorithm 5.

$$P5 = \begin{bmatrix} 0.5 & 0.1 & 0.4 \\ 0.7 & 0.1 & 0.2 \\ 0.3 & 0.2 & 0.5 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.5 & 0.1 & 0.4 \\ 0.4 & 0.15 & 0.45 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}$$

Since $c_3 = 0.05,\ c_2 = 0.05\ c_1 = -0.1$, Q belongs to class C. The steady-state distributions are :

$$\pi_{P5} = (0.4456, 0.1413, 0.4130) \quad \pi_Q = (0.3941, 0.1529, 0.4529) \quad and \quad \pi_{P5} <_{st} \pi_Q$$

3.4 Partition and Stochastic Complement

The stochastic complement was initially proposed by Meyer in [26] to uncouple Markov chains and to provide a simple approximation for steady-state. Here we propose a completely different idea based on an easy resolution of the stochastic complement. Let us consider a block decomposition of Q: $\begin{pmatrix} A\ B \\ C\ D \end{pmatrix}$, where A, B, C, and D are matrices of size $n_0{*}n_0$, $n_0{*}n_1$, $n_1{*}n_0$ and $n_1{*}n_1$ (with $n_0{+}n_1 = n$). We know that $I - D$ is not singular if P is not reducible [26]. We decompose π into two components π_0 and π_1 to obtain the stochastic complement formulation for the steady-state equation:

$$\begin{cases} \pi_0\ R = 0 \\ \pi_0\ r = 1 \\ \pi_1 = \pi_0\ H \end{cases} \tag{5}$$

where $H = B(I - D)^{-1}$, $R = I - A - HC$ and $r = e_0 + He_1$.

Following Quessette [17], we chose to partition the states such that matrix D is upper triangular with positive diagonal elements. It should be clear that this partition is not mandatory for the theory of stochastic complement. However it simplifies the computation of H. Such a partition is always possible, even if for some cases it implies that n_1 is very small [17].

It is quite simple to derive from Algorithm 1 an algorithm which builds a matrix of this form once the partition has been fixed. The algorithm has two steps. The first step is Algorithm 1. Then we remove the transitions in the lower triangle of D and sum up their probabilities in the corresponding diagonal elements of D.

3.5 Single Input Macro State Markov Chain

Feinberg and Chiu [14] have studied chains divided into macro-states where the transition entering a macro-state must go through exactly one node. This node is denoted as the input node of the macro-state. They have developed an algorithm to efficiently compute the steady-state distribution by decomposition. It consists of the resolution of the macro-state in isolation and the analysis of the chain reduced to input nodes. Unlike ordinary lumpability, the assumptions of the theorem are based on the graph of the transitions and do not take into account the real transition rates.

It is very easy to modify Algorithm 1 to create a Single Input Macro State Markov chain. We assume that for every macro state, the input state is the last state of the macro state. Thus the matrix Q looks like this:

$$\left[\begin{array}{c|c|c}
 & \cdots\ \ \cdots & \cdots\ \ \cdots \\
A & \ldots 0 \ldots & \ldots 0 \ldots \\
 & \cdots\ \ \cdots & \cdots\ \ \cdots \\
\hline
\cdots\ \ \cdots & & \cdots\ \ \cdots \\
\ldots 0 \ldots & B & \ldots 0 \ldots \\
\cdots\ \ \cdots & & \cdots\ \ \cdots \\
\hline
\cdots\ \ \cdots & \cdots\ \ \cdots & \\
\ldots 0 \ldots & \ldots 0 \ldots & C \\
\cdots\ \ \cdots & \cdots\ \ \cdots &
\end{array}\right]$$

The algorithm is based on the following decomposition into three types of block : diagonal blocks, upper triangle and lower triangle. The elements of diagonal blocks are computed using the same equalities as in Algorithm 1:

$$\begin{cases} Q_{1,j} &= \sum_{k=j}^{n} P_{1,k} - \sum_{k=j+1}^{n} Q_{1,k} \\ \\ Q_{i+1,j} &= max(\sum_{k=j}^{n} Q_{i,k}, \sum_{k=j}^{n} P_{i+1,k}) - \sum_{k=j+1}^{n} Q_{i+1,k} \end{cases} \tag{6}$$

The elements of blocks in upper and lower triangles have the "single input" structure : several columns of zero followed by a last column which is positive. Furthermore, lower and upper triangles differ because the elements of lower triangle of Q must follows inequalities which take into account the diagonal blocks of Q. Let us denote by $f(i)$ the lower index of the set which contains state i. Then for all i, j in the upper triangle, we just have to sum up the elements of P (take care of the lower index $f(j)$ on the summation of the elements of P):

$$\begin{cases} Q_{1,n} &= \sum_{k=f(n)}^{n} P_{1,k} \\ \\ Q_{1,j} &= \sum_{k=f(j)}^{n} P_{1,k} - \sum_{k=j+1}^{n} Q_{1,k} \\ \\ Q_{i+1,j} &= max(\sum_{k=j}^{n} Q_{i,k}, \sum_{k=f(j)}^{n} P_{i+1,k}) - \sum_{k=j+1}^{n} Q_{i+1,k} \end{cases} \tag{7}$$

And for all i, j in the lower triangle (here the lower index $f(j)$ is also also used in the summation of the elements in the former row of Q):

$$\left\{ Q_{i+1,j} = max(\sum_{k=f(j)}^{n} Q_{i,k}, \sum_{k=f(j)}^{n} P_{i+1,k}) - \sum_{k=j+1}^{n} Q_{i+1,k} \right. \tag{8}$$

The derivation of the algorithm is straightforward. Again let us apply this algorithm on matrix $P1$ with partition into two sets of size 2 and 3 to obtain matrix Q (we also give the values of f for all the indices):

$$f = (1, 1, 3, 3, 3) \quad Q = \left[\begin{array}{cc|ccc} 0.5 & 0.2 & 0.0 & 0.0 & 0.3 \\ 0.1 & 0.6 & 0.0 & 0.0 & 0.3 \\ \hline 0.0 & 0.3 & 0.4 & 0.0 & 0.3 \\ 0.0 & 0.1 & 0.1 & 0.5 & 0.3 \\ 0.0 & 0.2 & 0.0 & 0.3 & 0.5 \end{array}\right]$$

This structure have been used by several authors even if their proofs of comparison are usually based on sample-path theorem [19,24,25].

3.6 Quasi Birth and Death Process

Finally, we have to briefly mention QBD matrices. They have a well-known algorithmic solution [23] but clearly it is not always possible to build an upper bounding st-monotone matrix which is block-tridiagonal. However, it is possible to derive some generalization of Algorithm 1 to get a QBD is the initial matrix has upper bounded transitions to the right (i.e., there exist a small integer k such that for all indices, if $j - i > k$ then $P(i, j) = 0$). The example presented in [24] is partially based on such a structure.

4 A Real Example with Large State Space

As an example, we present the analysis of a buffer policy which combines the PushOut mechanism for the space management and a Head Of Line service discipline. We assume that there exist two types of packets with distinct loss rate requirements. In the sequel, we denote as high priority, the packets which have the highest requirements, i.e., the smallest loss ratio. A low priority packet which arrives in a full buffer is lost. If the buffer is not full, both types of packets are accepted. The PushOut mechanism specifies that when the buffer is full, an arriving high priority packet pushes out of the buffer a low priority one if there is any in the buffer. Otherwise the high priority packet is lost. For the sake of simplicity, we assume that packet size is constant. This is consistent with ATM cells but it is clearly a modeling simplification for other networks. Such a mechanism has been proposed for ATM networks [18]. We further assume that the low priority packets are scheduled before high priority packets (recall that the priority level is based on the access). We assume that the departure due to service completion always takes place just before the arrivals.

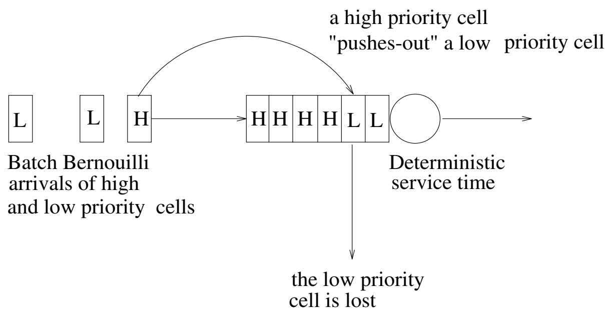

Fig. 1. Push-Out mechanism description

As the buffer size is B, the number of states is $(B + 1)(B + 2)/2$ if the arrivals follow a simple batch process. For the sake of simplicity we assume that the batch size is between 0 and 2. We use the following representation for the state space (T, H) where T is the total number of packets and H is the number of high priority packets. The states are ordered according to a lexicographic non decreasing ordering. It must be clear at this point that the ordering of the state is a very important issue. First, the rewards have to be non decreasing functions of the state indices. Furthermore, as the st-monotone property is based on the state representation and ordering, the accuracy of the results may depend on this ordering. Here, we are interested in the the expected number of lost packet per slot. Let us denote by R^i_M this expectation for type i packets and let $R = R^H + R^L$. The difficult problem here is the computation of R^H. Indeed R can be computed with a smaller chain since the Pushout mechanism does not change the global number of losses. It is sufficient to analyze the global number of packets (i.e without distinction). Such a chain has only $B + 1$ states if we use a simple batch arrival process. For realistic values of buffer size (i.e. 1000), such a chain is very simple to solve with usual numerical algorithms. However for the same value of B, the chain of the HOL+Pushout mechanisms has roughly $5\ 10^5$ states. So, we use Algorithm 4 to get a lumpable bounding matrix. And we analyze the macro-state chain. First let us describe the ordering of the states and the rewards. Let p^H_k be the probability of k arrivals of high priority packets during one slot.

$$R^H = \sum_{(T,H)} \Pi(T, H)\ \ p^H_2\ \ max(0, (H + 2 - B - 1_{T=H}))$$

Where $1_{T=H}$ is an indicator function which states if one high priority packet can leave the buffer at the end of the slot after service completion ($T = H$ that there is no low priority packet). Thus $max(0, (H+2-B-1_{T=H}))$ is the number of packets exceeding the buffer size. For this particular case, due to scheduling of arrivals and service, R^H can be computed in a more simpler expression :

$$R^H = p^H_2 \times \Pi(B, B)$$

Clearly, we have to estimate only one probability and the reward is a non decreasing function which is zero everywhere except for the last state where its value is one. For more general arrival process, the reward function is only slightly different.

The key idea to shorten the state space is to avoid the states with large value of low priority packets. So, we bound the matrix with an ordinary lumpable matrix Q with $o(B \times F)$ macro-states where the parameter F allows a trade-off between the computational complexity and the accuracy of the results. More precisely, we define macro-states (T, Y) where Y is constrained to evolve in the range $T..T - F$. If $Y = T - F$, then the state (T, Y) is a real macro-state which contains all the states (T, X) such that $X \leq T - F$. In this case Y is a upper bound of the number of high priority packets in the states which are aggregated. If $Y > T - F$ then the state contains only one state (T, X) where $Y = X$. So, Y represents exactly the number of high priority packets in the buffer (see figure 2). Clearly, if the value of F is large, we do not reduce the state space but we expect that the bound would be tight.

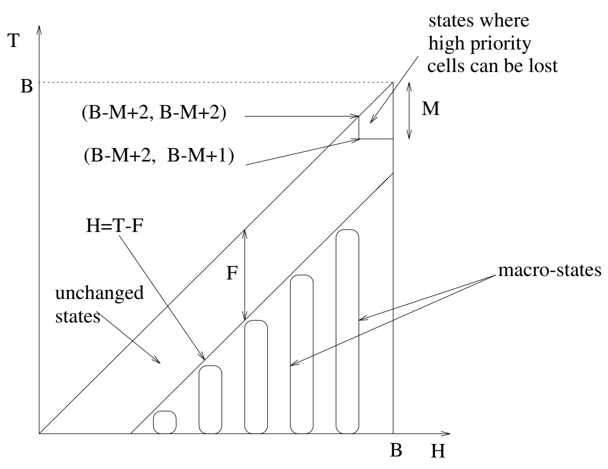

Fig. 2. The aggregated chain

In [16] we have analyzed small buffers to check the accuracy of the bound and large buffers to show the efficiency of the method. Here, we only present a typical comparison of these bounds for a small buffer of size 80 (these small value allows the computation of the exact result for comparison purpose). The load is 0.9 with 2/3 high level packets. With a sufficiently large value for F (typically 10), the algorithm gives accurate results. The exact result for R^H is in this example 8.9 10^{-13}. The bound with $F = 10$ is 9.510^{-13}. Of course if F is too small, the result is worse and can reach 10^{-6} for $F = 2$. The exact chain has 3321 states while the bound with $F = 10$ is computed with a chain of size 798. The number of states is divided by 4 and we only lost few digits. It is worthy to remark that a reduction by an order on the states space implies a reduction by two or three orders on the computation times for the steady state distribution. And the reduction is much more important if the original chain is bigger. Typically, for a buffer size of 1000 and an aggregation factor F equal to 20, the bounding matrix obtained from Algorithm 5 has roughly 20000 states. The original state space is 25 times larger.

The results shows previously are very accurate. We have found several reasons for that property. First the distribution is skewed. Almost all the probability mass is concentrated on the states with a small number of packets. Moreover the first part of the initial matrix is already st-monotone. This property is due to the ordering of the states we have considered. Again, we have to emphasis that the states ordering is a crucial issue for st-bounds [11].

For instance, consider the matrix of the chain for a small buffer of size 4. The chain has 15 states ordered in a lexicographic way: $\{(0), (1,0), (1,1), (2,0), (2,1), (2,2), (3,0), (3,1), (3,2), (3,3), (4,0), (4,1), (4,2), (4,3), (4,4)\}$. Let us denote by p, q and r respectively the arrival probabilities of arrival for a batch of size 0, 1 or 2. And let a be the probability that an arriving packets is a low priority one. Similarly b is the probability for a high level packet. The distribution of packets types in a size 2 batch are respectively c for 2 low level packets, e for two high level, and d for a mixed batch. Independence assumption on the type of packets entering the queue lead to an important reduction of the number of parameter (for instance $c = a^2$). However, it is not necessary to illustrate the effect of Algorithm 5.

$$P = \left(\begin{array}{c|cc|ccc|cccc|ccccc}
p & qa & qb & rc & rd & re & & & & & & & & & \\
\hline
p & qa & qb & rc & rd & re & & & & & & & & & \\
p & qa & qb & rc & rd & re & & & & & & & & & \\
\hline
 & p & & qa & qb & & rc & rd & re & & & & & & \\
 & & p & & qa & qb & & rc & rd & re & & & & & \\
 & & p & & qa & qb & & rc & rd & re & & & & & \\
\hline
 & & & p & & & qa & qb & & & rc & rd & re & & \\
 & & & & p & & & qa & qb & & & rc & rd & re & \\
 & & & & & p & & & qa & qb & & & rc & rd & re \\
 & & & & & p & & & qa & qb & & & rc & rd & re \\
\hline
 & & & & & & p & & & & qa+rc & qb+rd & re & & \\
 & & & & & & & p & & & & qa+rc & qb+rd & re & \\
 & & & & & & & & p & & & & qa+rc & qb+rd & re \\
 & & & & & & & & & p & & & & qa+rc & qb+rd+re \\
 & & & & & & & & & p & & & & qa+rc & bq+rd+re
\end{array}\right)$$

A careful inspection of matrix P shows that the 10 first rows of the matrix already satisfy the st-monotone property. For a bigger buffer model, this property is still true for the states where the buffer is not full. We assume that $F = 2$, the only one non trivial values for such a small example). Thus, we consider two real macro-states : $\{(3,2),(3,3)\}$ and $\{(4,2),(4,3),(4,4)\}$. Note that the initial matrix is already lumpable since the scheduling of service and arrivals imply that some states have similar transitions. For instance states (0), $(1,0)$ and $(1,1)$ can be gathered into one macro-state). We use this property in the resolution algorithm but we do not develop here to focus on the bounding algorithm. Algorithm 5 provides a lumpable matrix with the macro-states already defined which can be aggregated to obtain ($f = min(qb, rc)$ and $g = max(qb, rc)$):

$$\left(\begin{array}{c|cc|ccc|ccc|ccc}
p & qa & qb & rc & rd & re & & & & & & \\
\hline
p & qa & qb & rc & rd & re & & & & & & \\
p & qa & qb & rc & rd & re & & & & & & \\
\hline
 & p & & qa & qb & & rc & rd & re & & & \\
 & & p & & qa & qb & & rc & rd+re & & & \\
 & & p & & qa & qb & & rc & rd+re & & & \\
\hline
 & & & p & & & qa & qb & & rc & rd & re \\
 & & & & p & & & qa & qb & & rc & rd+re \\
 & & & & & p & & & qa+qb & & & r \\
\hline
 & & & & & & p & & & qa+f & g-rc & r \\
 & & & & & & & p & & & qa+f & g+rd+re \\
 & & & & & & & & p & & & q+r
\end{array}\right)$$

5 Algorithms for "icx" Comparison

Stoyan's proof in Theorem 4.2.5 of ([35], p.65]) that the monotonicity and the comparability of transition matrices yield sufficient conditions for chain comparison is not restricted to "st" ordering. Similarly, the definitions of the monotonicity and the comparison of stochastic matrices are much more general than the statements presented in section 2. First let us turn back to the definitions for "icx" ordering which is supposed to be more accurate than the st-ordering.

Definition 11 *Let X and Y be two random variables taking values on a totally ordered space. X is said to be less than Y ($X <_{icx} y$= if and only if $E[f(X)] \leq E[f(Y)]$, for all non decreasing convex functions f, whenever the expectations exist.*

For discrete state space, it is also possible to use a matrix formulation through matrix K_{icx}. Let p and q be respectively the probability distribution vectors of X and Y. $X <_{icx} Y$ if and only if $pK_{icx} \leq qK_{icx}$, where K_{icx} is defined as following :

$$K_{icx} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 2 & 1 & 0 & \dots & 0 \\ 3 & 2 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n & n-1 & n-2 & \dots & 1 \end{bmatrix}$$

This can be rewritten as follows :

$$X <_{icx} Y \Longleftrightarrow \sum_{k=i}^{n} (k-i+1)\; p_k \leq \sum_{k=i}^{n} (k-i+1)\; q_k, \qquad \forall i \in \{1, \dots, n\}$$

Similarly, we can define the increasing concave ordering by the set non-decreasing concave functions. In this case $K_{icv} = -K_{icx}^T$, where A^T denotes the transpose of matrix A.

Clearly, the icx-comparison and the icx-monotonicity of stochastic matrices are defined in the same manner as the st-ordering (see definitions 2 and 3). However, the characterization of the $<_{icx}$-monotonicity through matrix K_{icx} must take into account the finiteness of matrix P. Indeed, the conditions $K_{icx}^{-1} P K_{icx} \geq 0$ provide sufficient conditions for the $<_{icx}$-monotonicity. It is known for a long time time that these conditions are also necessary for infinite chains.

For finite chains, the necessary conditions were unknown until recently. Moreover the conditions $K_{icx}^{-1} P K_{icx} \geq 0$ are very restrictive and they lead to a chain whose first and last states are absorbing. Thus, it was not possible to develop an algorithmic approach without an efficient necessary and sufficient condition for monotonicity. Recently, in [2], Benmammoun has proved such conditions for the icx-monotonicity of finite chains. This characterization is based on matrix Z_{icx} which is slightly different from matrix K_{icx}^{-1}.

$$K_{icx}^{-1} = \begin{bmatrix} 1 & 0 & 0 \dots 0 \\ -2 & 1 & 0 \dots 0 \\ 1 & -2 & 1 \dots 0 \\ \vdots & \vdots & \vdots \ddots \vdots \\ 0 & \dots & 1\ -2\ 1 \end{bmatrix} \quad Z_{icx} = \begin{bmatrix} 1 & 0 & 0 \dots 0 \\ -1 & 1 & 0 \dots 0 \\ 1 & -2 & 1 \dots 0 \\ \vdots & \vdots & \vdots \ddots \vdots \\ 0 & \dots & 1\ -2\ 1 \end{bmatrix}$$

5.1 Basic Algorithm

The sufficient conditions to compare Markov chains through the monotonicity and the comparability of matrices (see theorem 1) are also valid for the icx-ordering. Therefore, it is possible to design an algorithm to construct an icx-monotone and upper bounding chain based on Benmammoun's characterization.

Algorithm 6 An icx-monotone upper bound Q:

```
$q_{1,n} = p_{1,n}$;
$q_{2,n} = max(q_{1,n}, p_{2,n})$;
for $i = 3, \dots, n$ do $q_{i,n} = max(p_{i,n}, 2q_{i-1,n} - q_{i-2,n})$; od
for $j = n-1, n-2, \cdots, 2$ do
    $q_{1,j} = \sum_{k=j}^{n}(k-j+1)p_{1,k} - \sum_{k=j+1}^{n}(k-j+1)q_{1,k}$;
    $q_{2,j} = max\left(\sum_{k=j}^{n}(k-j+1)p_{2,k}, \sum_{k=j}^{n}(k-j+1)q_{1,k}\right)$
          $- \sum_{k=j+1}^{n}(k-j+1)q_{2,k}$;

    for $i = 3, 4, \cdots, n$ do
        $q_{i,j} = max\left(\sum_{k=j}^{n}(k-j+1)p_{i,k},\right.$
              $\left.2\sum_{k=j}^{n}(k-j+1)q_{i-1,k} - \sum_{k=j}^{n}(k-j+1)q_{i-2,k}\right)$
              $- \sum_{k=j+1}^{n}(k-j+1)q_{i,k}$;
    od
od
for $i = 1, 2 \cdots n$ do $q_{i,1} = 1 - \sum_{j=2}^{n} q_{i,j}$; od
```

Unfortunately, the output of this algorithm is not always a stochastic matrix as we may obtain elements larger than 1.0. First, we apply this algorithm to matrix $P6$ and the output is a stochastic upper bound Q:

$$P6 = \begin{bmatrix} 0.5 & 0.15 & 0.35 \\ 0.3 & 0.4 & 0.3 \\ 0.45 & 0.1 & 0.45 \end{bmatrix} \quad Q = \begin{bmatrix} 0.5 & 0.15 & 0.35 \\ 0.35 & 0.3 & 0.35 \\ 0.4 & 0.25 & 0.45 \end{bmatrix}$$

However, for matrix $P7$, the output of Algorithm 6 is not a stochastic matrix since $Q_{3,3} > 1$.

$$P7 = \begin{bmatrix} 0.5 & 0.15 & 0.35 \\ 0.3 & 0.0 & 0.7 \\ 0.45 & 0.1 & 0.45 \end{bmatrix}$$

Indeed, the last column of Q is:

$$Q = \begin{bmatrix} 0.35 \\ 0.7 \\ 1.05 \end{bmatrix}$$

Several heuristics may be used to solve this problem. Further researches are still necessary to obtain a simple and efficient algorithm.

5.2 Class C Matrices

Beside, the closed form solution of the stationary distribution, class C matrices have nice properties about stochastic monotonicity. First, we present the stochastic monotonicity characterization for this class and then we show an algorithm to construct an icx-monotone, upper bounding, class C matrix.

Proposition 3

$$P \;\; is \;\; icx - monotone \iff \sum_{k=j}^{n} (k-j+1)\; c_k \geq 0, \qquad \forall j \in \{1, \ldots, n\}$$

Proposition 4 *If P is in class C, then*

$$P \; is \; st - monotone \implies P \; is \; icx - monotone$$

Let us emphasize here by an example, that, in general, st-monotonicity does not imply icx-monotonicity:

$$P = \begin{bmatrix} 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0.6 \\ 0.1 & 0.3 & 0.6 \end{bmatrix}$$

Clearly, P is st-monotone. On the other hand, if we consider $p = [0.2\ 0.4\ 0.4]$ and $q = [0.3\ 0.1\ 0.6]$. $pP = [0.24\ 0.28\ 0.48]$ and $qP = [0.26\ 0.32\ 0.42]$. And $p <_{icx} q$ is true while $pP <_{icx} qP$ is false. Thus P is not icx-monotone.

In the following algorithm we compute an icx-monotone, upper bounding, class C matrix, Q for a given matrix P [4] As in the st-ordering case, this algorithm consists in computing the first row $q_{1,j}, \;\; 1 \leq j \leq n$ and the constant $c_j, \;\; 1 \leq j \leq n$. In fact, these parameters take values within an interval $([\underline{c_j}, \overline{c_j}])$ and $[\underline{q_{1,j}}, \overline{q_{1,j}}])$. Since we construct an upper bound, one must intuitively choose the smallest values for these parameters in order to have elements which are as close as possible to the original ones. Moreover, we define a constant *const* which is greater than 1, but less than $\overline{c_j}$. By doing so, all entries of Q are positive, thus Q is irreducible.

Algorithm 7 Construction of an icx-monotone upper bounding DTMC Q which belongs to class C:

```
$\underline{q_{1,n}} = max_{1 \leq i \leq n-1}[\frac{(n-1)p_{i,n}-i+1}{n-i}]$; $q_{1,n} = \underline{q_{1,n}} + \frac{1-\underline{q_{1,n}}}{const}$;

$\underline{c_n} = max_{2 \leq i \leq n}[\frac{p_{i,n}-q_{1,n}}{i-1}]$; $\overline{c_n} = \frac{1-q_{1,n}}{n-1}$;

if $\underline{c_n} < 0$ then $c_n = 0$ else $c_n = \underline{c_n} + \frac{\overline{c_n}-\underline{c_n}}{const}$;

for $j = n-1, n-2, \cdots, 2$ do
    $f(i,j) = \frac{n-1}{n-i}\left[\sum_{k=j}^{n}(k-j+1)p_{i,k} - \sum_{k=j+1}^{n}(k-j+1)q_{i,k}\right]$
        $- \frac{i-1}{n-i}[1-\sum_{k=j+1}^{n} q_{n,k}]$;

    $\underline{q_{1,j}} = (max_{1 \leq i \leq n-1}\ f(i,j))^+$; $q_{1,j} = \underline{q_{1,j}} + \frac{1-\sum_{k=j+1}^{n} q_{1,k} - \underline{q_{1,j}}}{const}$;

    $\alpha_j = max_{2 \leq i \leq n}(\frac{\sum_{k=j}(k-j+1)p_{i,k} - \sum_{k=j+1}^{n}(k-j+1)q_{i,k} - q_{1,j}}{i-1})$;

    $\underline{c_j} = max(\alpha_j, \frac{-q_{1,j}}{n-1})$; $\overline{c_j} = \frac{1-\sum_{k=j+1}^{n} q_{n,k} - q_{1,j}}{n-1}$;

    if $\underline{c_j} < -\sum_{k=j=1}^{n}(k-j+1)c_k$ then $c_j = -\sum_{k=j+1}^{n}(k-j+1)c_k$
        else $c_j = \underline{c_j}\frac{\overline{c_j}-\underline{c_j}}{const}$;
od
$q_{1,1} = 1 - \sum_{j=2}^{n} q_{1,j}$;
$c_1 = -\sum_{j=2}^{n} c_j$;
```

We illustrate the application of this algorithm on matrix $P7$.

$$P7 = \begin{bmatrix} 0.25 & 0.2 & 0.25 & 0.3 \\ 0.15 & 0.1 & 0.65 & 0.1 \\ 0.35 & 0.05 & 0.15 & 0.45 \\ 0.3 & 0.2 & 0.1 & 0.4 \end{bmatrix} \quad Q = \begin{bmatrix} 0.2718 & 0.1962 & 0.162 & 0.37 \\ 0.279 & 0.158 & 0.136 & 0.427 \\ 0.2863 & 0.1199 & 0.1098 & 0.484 \\ 0.2935 & 0.0818 & 0.0837 & 0.541 \end{bmatrix}$$

Matrix Q obtained by this algorithm belongs to class C with $c_1 = 0.00723$, $c_2 = -0.03813$, $c_3 = -0.0261$, $c_4 = 0.057$. Their steady-state distributions are $\pi_P = (0.2755, 0.1497, 0.2354, 0.3393)$ and $\pi_Q = (0.2846, 0.1286, 0.1157, 0.4711)$ and we have $\pi_P <_{icx} \pi_P$.

6 Conclusions

Strong stochastic bounds are not limited to sample-path proofs. It is now possible to compute bounds of the steady-state distribution directly from the chain. This approach may be specially useful for high speed networks modeling where the performance requirements are thresholds. Using the algorithmic approach we survey in this paper, a sample-path proof is not necessary anymore and these algorithms may be integrated into software performance tools based on Markov chains. Generalizations to other orderings or to computation of transient measures are still important problems for performance analysis.

References

1. Abu-Amsha O., VincentJ.-M.: An algorithm to bound functionals of Markov chains with large state space. Int: 4th INFORMS Conference on Telecommunications, Boca Raton, Florida, (1998)
2. Benmammoun M.: Encadrement stochastiques et évaluation de performances des réseaux, PHD, Université de Versailles St-Quentin en Yvelines, (2002)
3. Benmammoun M., Fourneau J.M., Pekergin N., Troubnikoff A.: An algorithmic and numerical approach to bound the performance of high speed networks, Submitted, (2002)
4. Benmammoun M., Pekergin N.: Closed form stochastic bounds on the stationary distribution of Markov chains. To appear in Probability in the Engineering and Informational Sciences, (2002)
5. Boujdaine F., Dayar T., Fourneau J.M., Pekergin N., Saadi S., Vincent J.M.: A new proof of st-comparison for polynomials of a stochastic matrix, Submitted, (2002)
6. Buchholz P.: An aggregation\disaggregation algorithm for stochastic automata networks. In: Probability in the Engineering and Informational Sciences, V 11, (1997) 229–253
7. Buchholz P.: Projection methods for the analysis of stochastic automata networks. In: Proc. of the 3rd International Workshop on the Numerical Solution of Markov Chains, B. Plateau, W. J. Stewart, M. Silva, (Eds.), Prensas Universitarias de Zaragoza, Spain, (1999) pp. 149–168.
8. Courtois P.J., Semal P.: Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition. In: Journal of ACM, V 31 (1984) 804–825
9. Courtois P.J., Semal P.: Computable bounds for conditional steady-state probabilities in large Markov chains and queueing models. In: IEEE JSAC, V4, N6, (1986)
10. Dayar T., Fourneau J.M., Pekergin N.: Transforming stochastic matrices for stochastic comparison with the st-order, Submitted, (2002)
11. Dayar T., Pekergin, N.: Stochastic comparison, reorderings, and nearly completely decomposable Markov chains. In: Proceedings of the International Conference on the Numerical Solution of Markov Chains (NSMC'99), (Ed. Plateau, B. Stewart, W.), Prensas universitarias de Zaragoza. (1999) 228–246
12. Dayar T., Stewart W. J.: Comparison of partitioning techniques for two-level iterative solvers on large sparse Markov chains. In: SIAM Journal on Scientific Computing V21 (2000) 1691–1705.
13. Donatelli S.: Superposed generalized stochastic Petri nets: definition and efficient solution. In: Proc. 15th Int. Conf. on Application and Theory of Petri Nets, Zaragoza, Spain, (1994)
14. Feinberg B.N., Chiu S.S.: A method to calculate steady-state distributions of large Markov chains by aggregating states. In: Oper. Res, V 35 (1987) 282-290
15. Fernandes P., Plateau B., Stewart W.J.: Efficient descriptor-vector multiplications in stochastic automata networks. In: Journal of the ACM, V45 (1998) 381–414.

16. Fourneau J.M., Pekergin N., Taleb H.: An Application of Stochastic Ordering to the Analysis of the PushOut Mechanism. In Performance Modelling and Evaluation of ATM Networks, Chapman and Hall, (1995) 227–244
17. Fourneau J.M., Quessette F.: Graphs and Stochastic Automata Networks. In: Proceedings of the 2nd Int. Workshop on the Numerical Solution of Markov Chains, Raleigh, USA, (1995)
18. Hébuterne G., Gravey A.: A space priority queueing mechanism for multiplexing ATM channels. In: ITC Specialist Seminar, Computer Network and ISDN Systems, V20 (1990) 37–43
19. Golubchik, L. and Lui, J.: Bounding of performance measures for a threshold-based queuing systems with hysteresis. In: Proceeding of ACM SIGMETRICS'97, (1997) 147–157
20. Hillston J., Kloul L.: An Efficient Kronecker Representation for PEPA Models. In: PAPM'2001, Aachen Germany, (2001)
21. Keilson J., Kester A.: Monotone matrices and monotone Markov processes. In: Stochastic Processes and Their Applications, V5 (1977) 231–241
22. Kijima M.: Markov Processes for stochastic modeling. Chapman & Hall (1997)
23. Latouche G., Ramaswami V.: Introduction to Matrix Analytic Methods in Stochastic Modeling. SIAM, (1999)
24. Lui, J. Muntz, R. and Towsley, D.: Bounding the mean response time of the minimum expected delay routing policy: an algorithmic approach. In: IEEE Transactions on Computers. V44 N12 (1995) 1371–1382
25. Lui, J. Muntz, R. and Towsley, D.: Computing performance bounds of Fork-Join parallel programs under a multiprocessing environment. In: IEEE Transactions on Parallel and Distributed Systems. V9 N3 (1998) 295–311
26. Meyer C.D.: Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. In: SIAM Review. V31 (1989) 240–272.
27. Pekergin N.: Stochastic delay bounds on fair queueing algorithms. In: Proceedings of INFOCOM'99 New York (1999) 1212–1220
28. Pekergin N.: Stochastic performance bounds by state reduction. In: Performance Evaluation V36-37 (1999) 1–17
29. Plateau B.: On the stochastic structure of parallelism and synchronization models for distributed algorithms. In: Proceedings of the SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Texas (1985) 147–154
30. Plateau B., Fourneau J.-M., Lee K.-H.: PEPS: A package for solving complex Markov models of parallel systems. In: Modeling Techniques and Tools for Computer Performance Evaluation, R. Puigjaner, D. Potier (Eds.), Spain (1988) 291–305
31. Plateau B., Fourneau J.-M.: A methodology for solving Markov models of parallel systems. In: Journal of Parallel and Distributed Computing. V12 (1991) 370–387.
32. Shaked M., Shantikumar J.G.: Stochastic Orders and Their Applications. In: Academic Press, California (1994)
33. Stewart W.J., Atif K., Plateau B.: The numerical solution of stochastic automata networks. In: European Journal of Operational Research V86 (1995) 503–525
34. Stewart W. J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, (1994)
35. Stoyan D.: Comparison Methods for Queues and Other Stochastic Models. John Wiley & Sons, Berlin, Germany, (1983)
36. Truffet L.: Reduction Technique For Discrete Time Markov Chains on Totally Ordered State Space Using Stochastic Comparisons. In: Journal of Applied Probability, V37 N3 (2000)

37. Uysal E., Dayar T.: Iterative methods based on splittings for stochastic automata networks. In: European Journal of Operational Research, V 110 (1998) 166–186
38. Van Dijk N.: Error bound analysis for queueing networks" In: Performane 96 Tutorials, Lausanne, (1996)