Perfect Simulation and Monotone Stochastic Bounds[*]

J.M. Fourneau
INRIA project MESCAL
Laboratoire Informatique de Grenoble
CNRS UMR 5217
Montobonnot, France
jmf@prism.uvsq.fr

I. Kadi
PRiSM CNRS UMR 8144
Université de Versailles Saint-Quentin
45 Av. des Etats Unis, 78000 Versailles, France
ika@prism.uvsq.fr

N. Pekergin
LACL,
Université Paris 12
61 Av. Général de Gaulle,
94010, Créteil, France
nih@prism.uvsq.fr

J. Vienne
INRIA project MESCAL
Laboratoire Informatique de Grenoble
CNRS UMR 5217
Montobonnot, France
jerome.vienne@imag.fr

J.M. Vincent
INRIA project MESCAL
Laboratoire Informatique de Grenoble
CNRS UMR 5217
Montobonnot, France
jean-marc.vincent@imag.fr

ABSTRACT

We combine monotone bounds of Markov chains and the coupling from the past to obtain an exact sampling of a strong stochastic bound of the steady-state distribution for a Markov chain. Stochastic bounds are sufficient to bound any positive increasing rewards on the steady-state such as the loss rates and the average size or delay. We show the equivalence between st-monotonicity and event monotonicity when the state space is endowed with a total ordering and we provide several algorithms to transform a system into a set of monotone events. As we deal with monotone systems, the coupling technique requires less computational efforts for each iteration. Numerical examples show that we can obtain very important speedups.

Keywords

Perfect Simulation, Stochastic Bounds, Coupling from the past, monotone Markov chains

1. INTRODUCTION

Simulation is the most versatile tool to model large and complex systems such as high speed networks or highly dependable computer systems. Unfortunately even if we can represent large systems with simulators, usual techniques require a long time to run when we consider a system with a large number of resources and when we need very accurate results. Drawbacks of simulation are the control of the warm-up period before sampling, the influence of the initial state on the stochastic behavior of the system and the accuracy for the estimation of small events probabilities. Thus it is typically difficult to perform a rare event simulation with a high dependence on the initial state. Usually the initial state of the simulator is an empty network when we model a network of queues or a fully operational states when we deal with reliability problem and these states clearly add a bias to the likelihood. Building an arbitrary state of the model raises new questions:

- is the state really reachable ?
- what is the bias induced by this state ?

And the warm-up period problems remains to be solved for any initial state. Even if we consider Poisson arrival of events, regeneration does not really help as the regenerative points that we can easily identify appear very unlikely (consider for instance the empty state in a network of queue when the load is not light).

In this paper we advocate to use *perfect simulation* and to combine this technique with stochastic monotonicity to speed up the computation. Perfect simulation [16] directly builds steady state samples avoiding the warm-up period and the initial state bias. This method is based on the more general theory of coupling for Markov chains. Let us first review some ideas about coupling. Assume that we compute with the same random sequence of random numbers a sample path beginning at any initial state. If at time t two sample-paths are in the same state (we say that they couple), they will continue forever during all the simulation. When all the sample-paths have coupled, we obtain a sample state. We may use the state to initialize the simulation or consider it as a sample of distribution. it is not necessary anymore to continue the simulation. For instance in Fig. 1 all the sample-paths have coupled in state 1 at time 4. In this drawing, each column is associated to a time instant and the rows contain the states. U_i is the i-th value of the random sequence used to generate the transitions of the chain at time i. This drawing is an example of forward coupling also called coupling in the future.

It is known for a long time that coupling in the future does not provide a sample distributed according to the steady-state. But Propp and Wilson have proved that coupling from the past (CFTP), also called backward-coupling, gives an exact sample of the steady-

[*]This work is supported by SMS, ANR-05-BLAN-009-02 and Checkbound, ANR-06-SETI-002

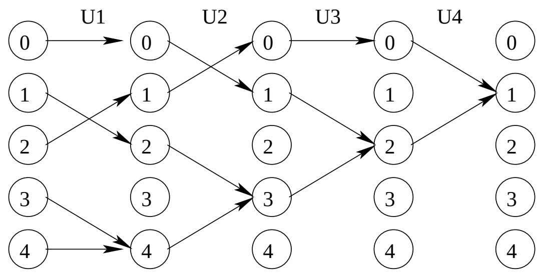

Figure 1: Forward Coupling

state distribution [16]. Coupling from the past is similiar to coupling in the future but the initial time of the simulation will be chosen randomly whereas the final time is deterministic. In other words the Markov chain is not started at time 0 but sufficiently far away in the past such that at time 0 all the paths are coupled.

This method is extremely efficient when the state space is large and has monotone dynamic. It has also be shown recently that we can use antithetic variable technique to speed up the computation of the confidence interval [21]. But many practical and theoretical problems remain to be solved for discrete Markovian systems to obtain a fully versatile technique.

One of the problem we must consider is the number of operations we need to obtain a sample. The general backward algorithm tries to couple sample-paths beginning in every state in the state space. Thus modelling very large state space systems requires some model transformation. Furthermore the number of operations is at least linear in the size of the state space. The monotonicity property of the event structure of the model (which is formally defined in the next section) allows to use a more efficient algorithm which sandwiches all sample-paths to couple into extreme ones.

Many routing techniques for networks of queues are monotone [11], but a large number of models of synchronized systems are not monotone. So we prove a model transformation technique which derives monotone models. The fundamental theory we need here is the stochastic comparison of Discrete-Time Markov Chains (see [14] for a recent presentation of this theory). These models are built to provide bounds on several stochastic estimates we usually want to obtain. Strong stochastic bounds are associated to increasing rewards we compute on transient distributions and on the steady-state when it exists. Thus if we can compare two distributions we can also compare increasing rewards on these distributions. This is usually sufficient for Quality of Service modeling as we just want to prove that some quantities (average delay, loss rates) are smaller than some performance thresholds (see for instance [15] for an application on Fair Queueing disciplines and [12] for algorithms to bound sub-stochastic matrices for point availability models).

The following of the paper is as follows. The next section is devoted to a brief presentation of Perfect Simulation and its complexity. In section III we present the theory of construction of Monotone Markov chains and we prove the relationships between the monotonicity associated to the strong stochastic ordering to compare Markov Chains and the monotonicity of events considered in the sandwiching of sample paths when the state space is assumed to be totally ordered. We also give an algorithm to obtain a set of monotone events from a monotone matrix. In section IV we consider the general problem of model transformation to obtain a monotone representation using matrices and events. Finally numerical results are shown in section V to illustrate that the computation of the monotone bounds may be much faster than in the backward perfect simulation.

2. PERFECT SIMULATION

In this paper we assume finite state space, ergodic Continuous-Time Markov Chains (CTMC). The first step of the event model is a uniformization of the CTMC. Thus the monotone algorithm is based on Discrete-Time Markov Chain (DTMC), and the strong ordering of these chains.

2.1 Global state iteration

Formally, when all the knowledge of the process dynamics is included in the state description, the system may be described by transition function Φ :

$$X_{n+1} = \Phi(X_n, U_{n+1}); \qquad (1)$$

where X_n is n^{th} observed state of the system, and $\{U_n\}_{n\in\mathbb{Z}}$ the sequence of inputs of the system, typically a sequence of calls to a `Random` function. This type of stochastic recursive sequence has been widely studied in a general framework [2] or [7] and some results related with perfect simulation may be found in [18, 19].

It is clear that, if the $\{U_n\}$ are independent and identically distributed, the process $\{X_n\}_{n\in\mathbb{Z}}$ defined by an initial value X_0 and the recursive equations (1) is a Markov chain. Conversely, given a transition matrix P, it is possible to find many transition functions Φ such that a Markov chain defined by (1) has transition matrix P [20]. Clearly, all these functions do not require the same average number of operations. This is illustrated in the example below.

2.2 Finite Markov Chains

Based on a stochastic recurrent sequence formulation, the following algorithm provides directly a sample of the steady state distribution. Notice that S denotes the finite state space set.

Algorithm 1 Backward-coupling simulation (general version)

```
for all $x \in \mathcal{S}$ do
    $y(x) \leftarrow x$ {choice of the initial value of the vector $y$, $n = 0$}
end for
repeat
    u $\leftarrow$ Random; {generation of $u_{-n}$}
    for all $x \in \mathcal{X}$ do
        $y(x) \leftarrow y(\Phi(x, u))$; {computation of the state at time 0
        of the trajectory issued from $x$ at time $-n$}
    end for
until All $y(x)$ are equal
return $y(x)$
```

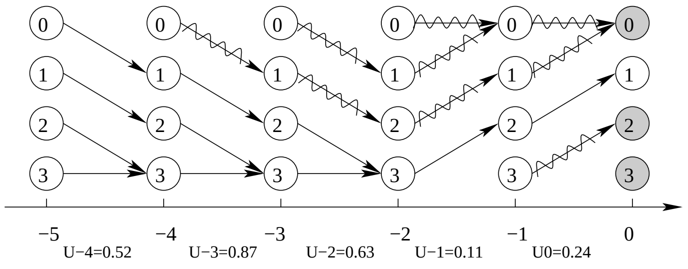

Figure 2: Convergence of Coupling from the Past Algorithm for a non monotone Markov Chain

Provided that the coupling time is almost surely finite (it requires some technical conditions), it is shown that Algorithm (1) generates a state from the steady-state distribution [16, 20]. Let $\mathbb{E}\tau_1$ be the expectation of the coupling time, n be the size of the state space and $op(\Phi)$ be the average number of operations to compute the transition function Φ. Clearly the average number of operations before coupling is $n(\mathbb{E}\tau_1)op(\Phi)$.

Function Φ has a lot of influence on the number of operations. First the way it is implemented has a linear influence because of term $op(\Phi)$. But the problem is much more complex. The coupling time depends on function Φ used to describe the chain. Let us illustrate both points with a toy example.

Consider a simple DTMC with three states a, b and c, an arbitrary initial state and transition probability matrix:

$$P = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$$

Using transform inverse function to represent the transition function, we can write $\Phi(a)$ as:

$$\begin{cases} \text{if } U < 0.5 & \Phi(a) = a \\ \text{elsif } U < 0.75 & \Phi(a) = b \\ \text{otherwise} & \Phi(a) = c \end{cases}$$

In this paper we represent this abstract code by the following entry in the transition table to have a more abstract representation:

	1/4	1/4	1/4	1/4
a	a		b	c

The usual utilization of this table is to perform a linear search from the beginning of the table with comparison on the boundaries. The number of operations we give in the following takes into account this linear search in the entries of the table. Using this table representation, this chain may be associated to the following functions Φ_A and Φ_B.

Φ_A	1/4	1/4	1/4	1/4
a	a		b	c
b	b		c	a
c	c		a	b

Φ_B	1/4	1/4	1/4	1/4
a	a		b	c
b	a	b		c
c	a	b	c	

Function Φ_A implies that the chain never couples because if $U < 0.5$ the state does not change, if $0.5 \leqslant U < 0.75$ the state in increased by one in a circular list and if $U \geqslant 0.75$ the state is decreased by one. Clearly the sample-paths never couple.

But at each step, the probability that the coupling occurs using function Φ_B is larger than 0.5. Thus the expectation of the coupling time is smaller than 2.

Designing algorithms to find the most efficient function Φ to reduce the expectation of coupling time is still an open problem. Some heuristics have been considered in [17].

Let us now consider the number of operations necessary to compute Φ_A and Φ_B. If we use an inverse distribution approach the average number of operations for Φ_A is 1.5 while the computation cost of Φ_B depends on the state: it is 1.5 on a and 1.75 on b and c. Note that this complexity is related to the number of non zero transitions in the chain, their probabilities and the order in which we consider the transitions in the algorithm. It is known for a long time that ordering the transitions in a decreasing order of their probability provides the most efficient implementation for inverse distribution implementation. It is worthy to remark here that trying to optimize the number of operations in Φ will eventually lead to a function Φ which does not have a small coupling time.

On the practical point on view, a better implementation based on the alias method has been used in PSI the software we have developed [20].

This toy model illustrates that the optimization of function Φ to minimize the number of operations before coupling is a complex approach. Hopefully it is also possible to reduce the number of sample-paths if the model is event-monotone.

2.3 Monotone perfect sampling

We suppose that the underlying system is governed by a finite set of events. Thus the system is described by a transition function with events. Following the Poisson calculus methodology [3], events are driven by homogeneous independent Poisson processes and the dynamic of the system is defined by a Poisson process (uniformization of all the Poisson processes) and a transition function $\Phi(x, e)$ defined for each state x and each event e occurring on a Poisson process. It is convenient to include the fact that some events could not be applied to a state (not allowed transitions) inside the transition function. For example, the event *end of service* could be executed only if the number of customers in a queue is greater than one. In a queueing network, a customer arrival, the end of a service and the following routing, a customer departure, are typical events in networks. Firings in Petri nets or Cooperation is Stochastic Algebra processes may be handled as well and this allows a versatile description of models.

Let us now give formal definitions.

DEFINITION 1 (EVENT). *An event e is an application defined on S, that associates to each state $x \in S$ a new state denoted by $\Phi(x, e)$. Φ is called the **transition function by events** of the system.*

DEFINITION 2 (EXECUTION). *An **execution** of the system is defined by an initial state $x_0 \in S$ and a sequence of events $e = \{e_n\}_{n \in \mathcal{N}}$. The sequence of states $\{x_n\}_{n \in \mathcal{N}}$ defined by the recurrence $x_{n+1} = \Phi(x_n, e_{n+1})$ for $n \geqslant 0$ is called a **trajectory**.*

DEFINITION 3 (MONOTONE EVENTS). *An event e is said to be monotone, if it preserves the partial ordering $\leqslant$ on S :*

$$\forall (x, y) \in S \;\; x \leqslant y \rightarrow \Phi(x, e) \leqslant \Phi(y, e)$$

If all events are monotone, the global system is said to be event-monotone.

When the operator Φ is event-monotone, the algorithm could be simplified by making iteration only on maximum and minimum values of the state space. For instance, in an open queuing network, there is an unique minimum (all queues are empty) and an unique maximum (all queues are full). Then we only iterate simultaneously 2 trajectories and the time reduction is in the order of the size of the state space. This concept of event-monotone models will clearly help to reduce the computation cost to obtain a sample.

We give in the following backward-coupling for event-monotone models. We consider a set of p events with rates $\lambda_1, \lambda_2, \cdots, \lambda_p$. Let Λ be $\sum_{i=1}^{p} \lambda_i$.

Let us turn now to the expectation of the coupling time for event-monotone systems. It has been shown [16] that the mean coupling time is optimal when steps in the past are multiplied by 2 when

Algorithm 2 Backward-coupling simulation (event-monotone version)

```
n=1;
E[1]=Generate-event()
**repeat**
  n=2n;
  **for all** $x \in M \cup m$ **do**
    $y(x) \leftarrow x$ {choice of the initial value of the vector $y$, $n = 0$}
  **end for**
  **for** i=n downto n/2+1 **do**
    E[i]=Generate-event() {generate event $-i$ according to distribution $(\frac{\lambda_1}{\Lambda}, \cdots, \frac{\lambda_p}{\Lambda})$}
    **for all** $x \in M \cup m$ **do**
      $y(x) \leftarrow \Phi(y(x), E[i])$ {apply the transition given by event $E[i]$ }
    **end for**
  **end for**
  **for** i=n/2 downto 1 **do**
    {event $-i$ has already been generated in a previous step}
    **for all** $x \in M \cup m$ **do**
      $y(x) \leftarrow \Phi(y(x), E[i])$
    **end for**
  **end for**
**until** All $y(x)$ are equal
return $y(x)$
```

trajectories issued from maximum and minimum states have not coupled at time 0. In the algorithm M (resp. m) denotes the set of maximal (resp. minimal) elements in the state space. We give in the following

This algorithm has the same convergence properties as Algorithm (1). The doubling period of each scheme ensures that the coupling time for an event-monotone system is less than the coupling time for Algorithm (1) multiplied by 2. Thus the expected number of operations is $2\mathbb{E}\tau_1 op(\Phi_A)$.

The algorithms and the drawings of the sample paths clearly show that an iteration of Algorithm (2) is much simpler than an iteration of Algorithm (1) as it builds less sample-paths. So if one can transform a model to imply event-monotonicity, one can use Algorithm (2) instead of Algorithm (1) and obtain an exact sample of the new model much more efficiently. We propose to transform the model to obtain an upper bound of the Markov Chain using the stochastic comparison approach. Note however that we do not have any information on the expectation of the coupling time in the initial model compared to the expectation of the coupling time in the upper bounding model. We just know that on the same model the algorithms have roughly the same average number of iterations. Numerical results show very important speedups and the theoretical approach proves that both expectations are upper bounded by the same geometric delay. To the best of our knowledge, we are only able to obtain an upper bound on the expectation or a stochastic upper bound on the distribution.

3. STOCHASTIC ORDERING ON TOTALLY ORDERED STATE SPACE

As the first step of the analysis consists in the uniformization of the process using the sum of the rates, we restrict ourselves to Discrete Time Markov Chains (DTMC) with finite state space $S = \{1, \ldots, n\}$ endowed with a total order. We consider the strong stochastic ordering (denoted "st" in the following). 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 Abu Amsha and Vincent [1]. In the following, $P_{i,*}$ will refer to row i of P.

3.1 A brief overview

Following [14], 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 4. *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, $X \leqslant_{st} Y$ if and only if $E[f(X)] \leqslant E[f(Y)]$ for all non decreasing functions f whenever the expectations exist.*

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

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. Moreover stochastic bounds are also valid for transient distributions. We do not use this property as we are mainly interested in performance measures on the the steady-state. It is known for a long time that monotonicity [14] 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 matrices and the st-monotonicity of a matrix.

DEFINITION 5 (MATRIX STOCHASTIC COMPARISON). *Let P and Q be two stochastic matrices. $P \leqslant_{st} Q$ if and only if $PK_{st} \leqslant QK_{st}$. This can be also characterized as $P_{i,*} \leqslant_{st} Q_{i,*}$ for all i.*

DEFINITION 6 (MONOTONE MATRIX). *Let P be a stochastic matrix, P is st-monotone if and only if for all u and v, if $u \leqslant_{st} v$ then $uP \leqslant_{st} vP$.*

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

PROPERTY 1. *Let P be a stochastic matrix. P is $\leqslant_{st}$-monotone if and only if $K_{st}^{-1} P K_{st} \geqslant 0$ component-wise.*

PROPERTY 2. *Let P be a stochastic matrix, P is st-monotone if and only if for all i,we have $P_{i,*} \leqslant_{st} P_{i+1,*}$*

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

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

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

$$\left\{ \begin{array}{lll} \sum_{k=j}^{n} P_{i,k} & \leqslant & \sum_{k=j}^{n} Q_{i,k} \quad \forall\, i,j \\ \sum_{k=j}^{n} Q_{i,k} & \leqslant & \sum_{k=j}^{n} Q_{i+1,k} \quad \forall\, i,j \end{array} \right. \tag{2}$$

3.2 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 3), a constructive way to design a stochastic matrix which yields a stochastic bound.

$$\left\{ \begin{array}{l} \sum_{k=j}^{n} Q_{1,k} = \sum_{k=j}^{n} P_{1,k} \\ \sum_{k=j}^{n} Q_{i+1,k} = max(\sum_{k=j}^{n} Q_{i,k}, \sum_{k=j}^{n} P_{i+1,k}) \quad \forall\, i,j \end{array} \right. \tag{3}$$

The following algorithm [1, 9] constructs the optimal 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 2.

Algorithm 3 Construction of the optimal st-monotone upper bounding DTMC Q:

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

First let us illustrate Algorithm (3) on a small matrix. We consider a 5×5 matrix for $P1$ and we compute matrix Q, and both steady-state distributions.

$$P = \left[\begin{array}{ccccc} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.7 & 0.1 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.5 & 0.2 & 0.0 \\ 0.1 & 0.0 & 0.1 & 0.7 & 0.1 \\ 0.0 & 0.2 & 0.2 & 0.1 & 0.5 \end{array} \right]$$

$$Q = \left[\begin{array}{ccccc} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.6 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.5 & 0.1 & 0.1 \\ 0.1 & 0.0 & 0.1 & 0.7 & 0.1 \\ 0.0 & 0.1 & 0.1 & 0.3 & 0.5 \end{array} \right]$$

Unfortunately this algorithm may transform an irreducible matrix into a reducible one and we do not have a complete characterization of matrices where this problem occurs. Indeed due to the subtraction operation in inner loops, some elements of Q may be zero even if the elements with the same indices in P are positive. We may use another algorithm (IMSUB) which solves the problem [9]. IMSUB avoids to delete the transitions in the upper triangle and makes positive the elements of the lower subdiagonal.

Algorithm 4 IMSUB

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

The first step is to prove the relations between the strong stochastic ordering we use for the DTMC and the event-monotonicity we consider for the coupling from the past algorithm.

3.3 Stochastic strong monotonicity and event-monotonicity

We suppose a totally ordered state space S and give the relationships between the stochastic monotonicity and event-monotonicity.

THEOREM 2. *When the state space is totally ordered, if the system is event-monotone (see Def.3), it is also stochastically monotone, if the underlying model is homogeneously governed by a finite set of events $E = \{e_1, \cdots e_n\}$.*

Proof :
We must show that for each two states x and y such that $x \leqslant y$, the row x and the row y of the underlying probability transition matrix are comparable in the sense of $\leqslant_{st}$ order:

$$P[x, *] \leqslant_{st} P[y, *]$$

To demonstrate this, we must show that:

$$\sum_{i=k}^{N} P(x, i) \;\leqslant\; \sum_{i=k}^{N} P(y, i) \;, \forall k = 1, 2, ..., N \tag{4}$$

From the event-monotone definition, we have

$$x \leqslant y \rightarrow \Phi(x, e) = z \leqslant \Phi(y, e) = z' \tag{5}$$

Assuming that the same set of events occur in each state, the transition matrix is defined by the event probabilities p_{e_u} for $e_u \in E$, such that $P[x, z] = \sum_{\Phi(x,e_u)=z} p_{e_u}$. Thus equation 4 can be written as follows:

$$\sum_{i=k}^{N} \sum_{\Phi(x,e_u)=i} p_{e_u} \;\leqslant\; \sum_{i=k}^{N} \sum_{\Phi(y,e_u)=i} p_{e_u} \;, \forall k = 1, 2, ..., N$$

It follows from Eq. 5 that for each e_u, if p_{e_u} is included in $\sum_{i=k}^{N} \sum_{\Phi(x,e_u)=i} p_{e_u}$, then it is also included in $\sum_{i=k}^{N} \sum_{\Phi(y,e_u)=i} p_{e_u}$. Thus the former inequalities are satisfied and P is $\leqslant_{st}$ monotone.

THEOREM 3. *When the state space is totally ordered, if the system is stochastically monotone, then there exists a finite set of events $e_1, \cdots e_n$, for which the system is event-monotone.*

We give the following algorithm which takes as input a stochastic monotone matrix $P = (p_{i,j})_{1*N,1*N}$ and gives a set of events E and a transition function Φ, such that $\Phi(x, e_u)$ is event-monotone.

Algorithm 5 Stochastic monotonicity → event-monotonicity

```
$S = \{1, 2, 3, ...., N\}$
$E = \emptyset$ {the set of events is initially empty}
$\Phi : S * E \longrightarrow S$
$V = [v_1, v_2, ..., v_N]$ {a vector representing the column index of the rightmost positive values for each row, initialized to N }
$k \leftarrow 0$
repeat
    for $i = 1$ to $N$ do
        $j \leftarrow v_i$
        while $p_{i,j} = 0$ do
            $j \leftarrow j - 1$
        end while
        $v_i \leftarrow j$ {update vector $V$}
    end for
    $k \leftarrow k + 1$ {the next event $e_k$}
    $p_{e_k} \leftarrow min_{1 \leqslant i \leqslant N}\ p_{i,v_i}$ {probability for event $e_k$}
    for $i = 1$ to $N$ do
        $\Phi(i, e_k) \leftarrow v_i$
        $p_{i,v_i} \leftarrow p_{i,v_i} - p_{e_k}$ {update the matrix}
    end for
until $\sum_{e_k \in E} p_{e_k} = 1$
```

Before proceeding with the proof, let us illustrate by an example on a simple matrix the steps of this algorithm. The components of vector V are the index of the elements which are framed in the matrices. These are indeed the column index of the rightmost positive values for each row.

$$P = \begin{bmatrix} 0.5 & 0.2 & 0.1 & |0.2| & 0.0 \\ 0.2 & 0.5 & 0.1 & 0.1 & |0.1| \\ 0.2 & 0.3 & 0.2 & 0.2 & |0.1| \\ 0.1 & 0.0 & 0.1 & 0.7 & |0.1| \\ 0.0 & 0.0 & 0.2 & 0.3 & |0.5| \end{bmatrix} \rightarrow p_{e_1} = 0.1$$

And vector V is $[4, 5, 5, 5, 5]$. The first event e_1 will occur with probability $p_{e_1} = 0.1$. In the sequel we only indicate the event probabilities obtained in each step.

$$P = \begin{bmatrix} 0.5 & 0.2 & 0.1 & |0.1| & 0.0 \\ 0.2 & 0.5 & 0.1 & |0.1| & 0.0 \\ 0.2 & 0.3 & 0.2 & |0.2| & 0.0 \\ 0.1 & 0.0 & 0.1 & |0.7| & 0.0 \\ 0.0 & 0.0 & 0.2 & 0.3 & |0.4| \end{bmatrix} \rightarrow p_{e_2} = 0.1$$

$$P = \begin{bmatrix} 0.5 & 0.2 & |0.1| & 0.0 & 0.0 \\ 0.2 & 0.5 & |0.1| & 0.0 & 0.0 \\ 0.2 & 0.3 & 0.2 & |0.1| & 0.0 \\ 0.1 & 0.0 & 0.1 & |0.6| & 0.0 \\ 0.0 & 0.0 & 0.2 & 0.3 & |0.3| \end{bmatrix} \rightarrow p_{e_3} = 0.1$$

$$P = \begin{bmatrix} 0.5 & |0.2| & 0.0 & 0.0 & 0.0 \\ 0.2 & |0.5| & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.3 & |0.2| & 0.0 & 0.0 \\ 0.1 & 0.0 & 0.1 & |0.5| & 0.0 \\ 0.0 & 0.0 & 0.2 & 0.3 & |0.2| \end{bmatrix} \rightarrow p_{e_4} = 0.2$$

$$P = \begin{bmatrix} |0.5| & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.2 & |0.3| & 0.0 & 0.0 & 0.0 \\ 0.2 & |0.3| & 0.0 & 0.0 & 0.0 \\ 0.1 & 0.0 & 0.1 & |0.3| & 0.0 \\ 0.0 & 0.0 & 0.2 & |0.3| & 0.0 \end{bmatrix} \rightarrow p_{e_5} = 0.3$$

$$P = \begin{bmatrix} |0.2| & 0.0 & 0.0 & 0.0 & 0.0 \\ |0.2| & 0.0 & 0.0 & 0.0 & 0.0 \\ |0.2| & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.1 & 0.0 & |0.1| & 0.0 & 0.0 \\ 0.0 & 0.0 & |0.2| & 0.0 & 0.0 \end{bmatrix} \rightarrow p_{e_6} = 0.1$$

$$P = \begin{bmatrix} |0.1| & 0.0 & 0.0 & 0.0 & 0.0 \\ |0.1| & 0.0 & 0.0 & 0.0 & 0.0 \\ |0.1| & 0.0 & 0.0 & 0.0 & 0.0 \\ |0.1| & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & |0.1| & 0.0 & 0.0 \end{bmatrix} \rightarrow p_{e_7} = 0.1$$

So we can derive an event description of the matrix P, using these seven events.

$$\begin{bmatrix} p_{e_5} + p_{e_6} + p_{e_7} & p_{e_4} & p_{e_3} & p_{e_1} + p_{e_2} & 0.0 \\ p_{e_6} + p_{e_7} & p_{e_5} + p_{e_4} & p_{e_3} & p_{e_2} & p_{e_1} \\ p_{e_6} + p_{e_7} & p_{e_5} & p_{e_4} & p_{e_2} + p_{e_3} & p_{e_1} \\ p_{e_7} & 0.0 & p_{e_6} & p_{e_2} + p_{e_3} + p_{e_4} + p_{e_5} & p_{e_1} \\ 0.0 & 0.0 & p_{e_6} + p_{e_7} & p_{e_5} & p_{e_1} + p_{e_2} + p_{e_3} + p_{e_4} \end{bmatrix}$$

We now give the proof that the transition function Φ given by the Algorithm (5) is event-monotone.

Proof:

The proof is done in two steps. First we demonstrate the existence and the monotonicity of the generated events and secondly we prove that this set of events is finite.

- The principe of this algorithm is to construct the event-monotone model by defining its monotone events one by one until the probability sum is equal to one. This algorithm proposes to choose the first positive value by beginning from the right for each row x. Let us note this value by R_x and the corresponding column by v_x, for all state $x \in S$ ($R_x = P(x, v_x)$). In fact vector V is constituted of $v_x,\ x \in S$.

  It follows from stochastic monotonicity that

  $$\sum_{i=k}^{N} P(x,i) \leqslant \sum_{i=k}^{N} P(y,i)\ , \forall k = 1, 2, ..., N \qquad (6)$$

  Thus for each two states x, y such that $x \leqslant y$:

  $$R_x = P(x, v_x),\ and\ R_y = P(y, v_y) \longrightarrow v_x \leqslant v_y \qquad (7)$$

  We define an event e_u with probability $p_{e_u} = min_{i=1,\cdots N}(R_i)$ such that for all state i, $\Phi(i, e_u) = v_i$. It follows from Eq. 7 that for each two states x, y such that $x \leqslant y$:

  $$\Phi(x, e_u) \leqslant \Phi(y, e_u)$$

  Thus e_u is a monotone event.

  Once this event is defined, we subtract its probability from the transition matrix. It is done by using vector V which means that for each row we subtract p_{e_u} from the right most positive entry. We have from equation 6:

  $$\sum_{i=k}^{N} P(x,i)\ - p_{e_u} \leqslant \sum_{i=k}^{N} P(y,i) - p_{e_u}\ , \forall k = 1, 2, ..., N \qquad (8)$$

Therefore the updated matrix satisfies the stochastic monotonicity inequalities for the next iteration to determine the next event. The other events are similarly determined by using the updated matrix and vector V.

- During the execution of this algorithm, the construction of each event makes null at least one matrix entry. Let M be the number of non zero entries of the underlying matrix. In the worst case after $M \leqslant N^2$ steps all the entries of the transition matrix will be null and the sum of event probabilities will be equal to one.

These theorems state that st-monotonicity and event monotonicity are equivalent when the state space is totally ordered. And Algorithm (5) allows us to obtain a monotone event description of a monotone matrix.

3.4 Coupling Time

Consider the time complexity of the perfect simulation algorithm. When the chain is not monotone, the backward iteration of Algorithm (1) should be done for each state. Then the expected time complexity is of the form

$$op(\Phi).n.\mathbb{E}\tau;$$

where $op(\Phi)$ is the time cost of applying the transition function to a single state, n is the size of the state space and τ the number of iterations until coupling occurs.

Denote by $\Phi(x, e_1, \cdots, e_n)$ the state of the system after applying to the initial state x the sequence of events $e_1, \cdots, e_n$. This notation generalizes to the image of a set of states A, $\Phi(A, e_1, \cdots, e_n)$.

DEFINITION 7. *The coupling time τ of the backward scheme is given by*

$$\tau = \inf \{n; \textit{ such that } Card(\Phi(S, e_n, \cdots, e_1)) = 1\}.$$

Provided that there exist with a positive probability an integer k and a specific coupling pattern of events $e_k, \cdots e_1$ such that $Card(\Phi(S, e_k, \cdots, e_1)) = 1$, then τ is almost surely finite and, moreover, upper bounded by a geometric distribution. In the case when the matrix is monotone and the transition function build on the inverse of the probability distribution function, then the sequence $e_1, \cdots, e_1$ (number of events is the size of the state space) is a coupling pattern and the backward scheme converges almost surely.

A first significant time reduction appears when the chain is monotone Algorithm (2). In that case, iterations are done only on the set of extreme states, and in our situation, because the state space is totally ordered just 2 trajectories are derived backward. But this algorithm needs the storage of the event sequence (amount of memory in the order of τ). The doubling scheme in the past ensures that the total number of iterations is less than 2τ.

Previous works [8] have shown that in many practical situations such as queueing networks, the coupling time $\mathbb{E}\tau$ is linear in the number of queues and quadratic in the capacities of the queues. Then it is much smaller than the size of the state space which grows exponentially when the number of queues increases.

A second improvement could be done by the utilization of aggregation function. Truffet [13] has proved that we can combine strong stochastic bounds and aggregation. We have developed two algorithms to build a monotone lumpable upper bounds: LIMSUB [10] and LMSUB [4]. LMBUB is a sparse matrix implementation of Truffet's method while LIMSUB creates an irreducible matrix. Of course in both cases we only build the lumped matrix of the bound to avoid a very large state space generation. This is really useful for numerical computation when one must handle vectors and matrices of the state space size.

Here most of the influence of the size of the state space has been cancelled when one uses monotone models and the sandwich algorithm. But lumping the model still helps. We assume that in the lumpable algorithm, the aggregation function l preserves the order of the initial state space. Then it is easier to simulate the backward scheme on the lumped chain and derive the steady state of lumped process and extend by decomposition of macro-states to the initial state. Denote by τ^l the coupling time of the backward scheme of the lumped chain. Then we have the theorem

THEOREM 4. *If the lumping function l is order preserving ($x \leqslant y$ implies $l(x) \leqslant l(y)$), then*

$$\tau^l \leqslant_{st} \tau.$$

This comes from the fact that if the initial chain have coupled then the lumped one have already coupled.

The estimation of coupling time reduction is known to be hard and counterexamples shows that some aggregation functions have the same coupling time distribution as the initial chain.

Let us now return to input model and discuss how we can transform it to be monotone. We present several algorithms in the next section to deal with various types of Markovian model. In section 5 we present some numerical experiments to show the speedup when the model is monotone.

4. MONOTONE PERFECT SAMPLING FOR NON-MONOTONE MODELS

We need several algorithms because we must consider several types of input models and several ways to transform these models into a set of monotone events. In the MESCAL project we have developed two simulators based on these concepts. PSI is an implementation of the coupling from the past algorithm for general Markov chains (i.e. non monotone). The transitions are implemented with the alias method to be more efficient. The inputs of PSI are a stochastic matrix or a transition rate matrix. PSI2 is based on monotone events and its inputs are a description of monotone events.

In this section we present algorithms to transform the underlying system which is not event-monotone in order to be able to apply monotone perfect sampling using a monotone event representation. We suppose that the underlying system is given as a stochastic matrix P or as a set of non monotone events. We must define a finite set of monotone events to be able to do monotone perfect sampling with PSI2. We present several algorithms based on various inputs and intermediate models: events or stochastic matrix.

4.1 Event transformation

First suppose that the input model is a set of events which is not monotone. We propose the following algorithm to obtain an upper bounding monotone transition function. This algorithm takes a transition function Φ and returns a transition function Φ^{sup} such that

- $\Phi(x, e) \leqslant \Phi^{sup}(x, e) \ \ \forall x, e$
- Φ^{sup} is event-monotone.

The algorithm is simply based on the following approach: Let $\{e_i\}$ be the initial set of events, we modify the events to be monotone and to build an upper bounding matrix as follows:

- the probability of an event does not change.

- the action of an event on a state is changed and we build a new transition function Φ^{sup} to describe this new effect:

$$\Phi^{sup}(e_i, y) = max_{x \leqslant y} \Phi(e_i, x)$$

Algorithm 6 Non monotone event representation $\rightarrow$ monotone event representation

```
$S = \{1, 2, 3, ..., N\}$
$E = \{e_1, e_2, e_3, ..., e_p\}$
$\Phi : S * E \longrightarrow S$
$\Phi^{sup} : S * E \longrightarrow S$
for $j = 1$ to $p$ do
    $\Phi^{sup}(1, e_j) \leftarrow \Phi(1, e_j)$
end for
for $i = 2$ to $n$ do
    for $j = 1$ to $p$ do
        $\Phi^{sup}(i, e_j) \leftarrow max(\Phi^{sup}(i-1, e_j), \Phi(i, e_j))$
    end for
end for
```

Let us now prove that the Algorithm (6) gives an event monotone transition function. Let S be the state space, $S = \{1, 2, \ldots, N\}$ and E be the event space, $E = \{e_1, \ldots, e_p\}$. To prove that the function proposed by Algorithm (6) is event-monotone, we need to introduce the following equivalence.

LEMMA 1. *For each event $e_u \in E$, These two propositions are equivalent:*

- *(1) $\forall i, j \in S :$ if $i \leqslant j$ then $\Phi(i, e_u) \leqslant \Phi(j, e_u)$*
- *(2) $\forall i \in S : \Phi(i, e_u) \leqslant \Phi(i+1, e_u)$*

Proof:

- (1) $\Longrightarrow$ (2) : This implication is evident, because by taking $j = i+1$, the proposition (2) is satisfied.
- (2) $\Longrightarrow$ (1) : From the proposition (2) we have :

  $$\Phi(1, e_u) \leqslant .. \leqslant \Phi(i, e_u) \leqslant \Phi(i+1, e_u) \leqslant .. \leqslant \Phi(N, e_u)$$

  Thus, we can obviously see that for all $i, j \in S$, $if\ i \leqslant j$: $\Phi(i, e_u) \leqslant \Phi(j, e_u)$

PROPERTY 3. $\Phi^{sup}(i, e_u)$ *is event-monotone.*

Proof:
To prove that $\Phi^{sup}(i, e_u)$, $i \in S$ is monotone, we must show that for each two states: $i \in S$, $j \in S$ and $i \leqslant j$: $\Phi^{sup}(i, e_u) \leqslant \Phi^{sup}(j, e_u)$. Indeed we can deduce from lemma 1 that it is sufficient to show that :

$$\forall i \in S : \Phi^{sup}(i, e_u) \leqslant \Phi^{sup}(i+1, e_u)$$

From the algorithm, we have for each event $e_u \in E$, and state $i \in S$:

$$\Phi^{sup}(i+1, e_j) = max(\Phi^{sup}(i, e_u), \Phi(i, e_u))$$

Then, $\Phi^{sup}(i+1, e_u) \geqslant \Phi^{sup}(i, e_u)$. Therefore we conclude that $\Phi^{sup}(i, e_u)$ is event-monotone.

PROPERTY 4. *Let Π (resp. Π_P) be the stationary distribution estimated by doing the monotone perfect sampling of Φ^{sup} (resp. of Φ). We must show that $\Pi_P \leqslant_{st} \Pi$.*

Proof: Let P (resp. Q) be the stochastic matrix associated to Φ (resp. Φ^{sup}). It follows from theorem (2) that event monotonicity implies $\leqslant_{st}$ monotonicity. Thus Q is $\leqslant_{st}$-monotone. The proof of $P \leqslant_{st} Q$ is exactly the proof of Vincent's algorithm (i.e. Algorithm 3). It is omitted here and it can be found in [1]. It is more important to remark that some important properties of Algorithm (3) such as optimality are still true for Algorithm (6).

4.2 Matrix and IMSUB

Assume now that the input of the model is a stochastic matrix. The algorithm we have already presented provides a first way to obtain a monotone set of events for an upper bound. We first apply IMSUB (i.e. Algorithm 4) to construct a $\leqslant_{st}$-monotone upper bounding matrix Q. From theorem 3, it is possible to compute the set of monotone-events. This can be done by means of Algorithm (5(. The stationary distribution of Q (Π_Q) can be estimated with monotone perfect sampling Algorithm (1) through PSI2. Since by construction $P \leqslant_{st} Q$, we obtain a stochastic upper bound on the stationary distribution of P ($\Pi_P \leqslant_{st} \Pi_Q$).

4.3 Matrix and transformation of events

Another solution with the same input is to find a set of events directly from the initial matrix. These events are not monotone as the matrix is not. This algorithm tries to minimize the number of events. Then we build from this set a new set of events which are monotone and which describe an upper bound of the matrix.

We first obtain the transition function Φ for the input matrix P. We give the following algorithm which takes as input a stochastic matrix P and returns a transition function Φ corresponding to P.

Algorithm 7 event representation for a transition matrix

```
$S = \{1, 2, 3, ...., N\}$
$E = \emptyset$ {the set of events is initially empty}
$\Phi : S * E \longrightarrow S$
$V = [v_1, v_2, ..., v_N]$
$k \leftarrow 0$
repeat
    for $i = 1$ to $N$ do
        $v_i \leftarrow 1;$
        for $j = 2$ to $N$ do
            if $p_{i,j} > p_{i,v_i}$ then
                $v_i \leftarrow j;$
            end if
        end for
    end for
    $k \leftarrow k+1$ {the next event $e_k$}
    $p_{e_k} \leftarrow min_{1 \leqslant i \leqslant N}\ p_{i,v_i}$ {the probability for event $e_k$}
    for $i = 1$ to $N$ do
        $\Phi(i, e_k) \leftarrow v_i$
        $p_{i,v_i} \leftarrow p_{i,v_i} - p_{e_k}$ {update the matrix}
    end for
until $\sum_{e_k \in E} p_{e_k} = 1$
```

As the proof that the function given by of Algorithm (7) is event-monotone is very similar to the former proof it is omitted here. Since the underlying matrix P is not $\leqslant_{st}$-monotone Φ is not event-monotone but it is now quite simple to complete the transform. We just have to use Algorithm (6) to obtain a new set which is now consistent with the inputs of PSI2 and which allows to build an upper bound of the original Markov chain.

5. BOUNDS AND PERFECT SIMULATION: SOME EXAMPLE

We consider a slightly modified version of the M/M/1/B queue. The service are exponential and the service rate is μ. We consider the superposition of two independent arrival processes. The first process is a Poisson process of rate λ. The second process is a Poisson batch arrival process. The distribution of the batch is state dependent. At state 0 it has size $k > 2$. For all other states, the batch size is 1 almost surely. The arrival rate is α.

Clearly this system is not monotone because of the transitions out of state 0. Thus we must use Algorithm (1) to obtain some samples We have a matrix based implementation of this algorithm in PSI a software tool developed by some of us at INRIA (see http://www-id.imag.fr/Logiciels/psi/). The input model is a stochastic matrix in sparse form.

We also apply Algorithm (3) to build a monotone upper bound which can be described as follows:

- the transitions due to Poisson arrivals and exponential services (with rate λ and μ are kept unchanged;
- the transitions due to the batch Poisson arrivals still have the same rate, and the size of the batch is still state dependent. However the distribution changes. Assume that we are at state x, if $x < k$ then the chain jumps to state k, otherwise the chain jumps to state $x + 1$. Thus the size of the batch is $max(k - x, 1)$. The description with jumps come from the matrix representation of the chain while the description with a state dependent batch arrival is suitable for an event based model.

We also have an experimental tool called PSI2 to perform the CFTP algorithm for monotone systems. The input model is based on events description.

We perform the following experiments on an ordinary PC (processor P4 3.0 GHz, 2Gb of memory). We first generate the models and we make the measurements once the models have been stored on disk. The first set of experiments deal with a system with a load smaller than 1.0. The service rate is 1.0 while $\lambda = 0.8$ and $\alpha = 0.1$. We have fixed $k = 12$ for all experiments and we change the value of the queue size to change the number of states in the model. We gather in Table 1 the numerical values obtained to generate 1000 samples. The data are the CPU user times in second measured by the time Linux command. In the second set of experiments, we model a queue with a load equals to 1.5. We only change the input rate λ to 1.4.

B	200	1000	5000	10000
Algo 1	8	187	4600	12200
Algo 2	2.8	16	72	144

Table 1: CPU User Time for 1000 samples

B	200	1000	5000	10000	100000
Algo 1	5.1	127	3000	12000	none
Algo 2	0.8	4.5	21.6	38	295

Table 2: CPU User Time for 1000 samples

In both tables we can see that CFTP for monotone Markov Chain is much more efficient than CFTP for an arbitrary Markov Chain. Algorithm (2) seams to have a linear complexity with the size of the state space. The complexity of Algorithm (1) is more than linear with the state space. A linear part comes from the samples which must begin in every state. The other part comes from the coupling time, the length of which is dependent of the size of the state space.

6. CONCLUSION

In this paper we show how we can combine monotone bounds and coupling from the past to obtain efficiently an exact sample of a strong stochastic bound. One can generalize to other model transformation. For instance, we have developed some algebraic manipulations of the chain which do not change the steady-state distribution and which make some chains monotone or almost monotone [6]. Recently a general algorithm has been presented in [5] which outputs a monotone matrix compliant to a pattern (basically a list of non zero transitions in the matrix). All the patterns presented allow to simplify the numerical computation of the steady-state computation of the chain. In the future we will try to identify patterns which provide a fast coupling time. Note that the theory we develop is for totally ordered Markov chains. When the state space is endowed with a partial order the equivalence between event monotonicity and stochastic monotonicity is much more complex.

Note that it is also possible to use a lower bounding algorithm to obtain a monotone matrix. One can obtain two monotone systems which can be efficiently simulated and gives some information about bounding accuracy as we have lower and upper bounds.

It is also worthy to remark that we can build a stochastic monotone finite Markov Chain for an infinite one. Thus using a model transformation we can obtain a finite Markov Chain which can be simulated using coupling from the past while the usual algorithms do not apply on infinite state space.

7. REFERENCES

[1] O. Abu Amsha and J.-M. Vincent. An algorithm to bound functionals of markov chains with large state space. In *INFORMS*, Boca Raton, 1998.

[2] A. Borovkov and S. Foss. Two ergodicity criteria for stochastically recursive sequences. *Acta Appl. Math.*, 34, 1994.

[3] P. Brémaud. *Markov Chains: Gibbs fields, Monte Carlo Simulation and Queues*. Springer-Verlag, 1999.

[4] A. Busic and J.-M. Fourneau. Bounds for point and steady-state availability: An algorithmic approach based on lumpability and stochastic ordering. In M. Bravetti, L. Kloul, and G. Zavattaro, editors, *Formal Techniques for Computer Systems and Business Processes, European Performance Engineering Workshop, EPEW 2005 and International Workshop on Web Services and Formal Methods, WS-FM 2005, Versailles, France, September 1-3, 2005, Proceedings*, volume 3670 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2005.

[5] A. Busic and J.-M. Fourneau. A matrix pattern compliant strong stochastic bound. In *2005 IEEE/IPSJ International Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops), Trento, Italy*, pages 256–259. IEEE Computer Society, 2005.

[6] T. Dayar, J.-M. Fourneau, N. Pekergin, and J.-M. Vincent. Polynomials of a stochastic matrix and strong stochastic bounds. In *Markov Anniversary Meeting*, pages 211–228, Charleston, 2006. Celebration of the 100th anniversary of Markov.

[7] P. Diaconis and D. Freedman. Iterated random functions. *SIAM Review*, 41(1):45–76, 1999.

[8] J. Dopper, B. Gaujal, and J.-M. Vincent. Bounds for the coupling time in queueing networks perfect simulation. In *Numerical Solutions for Markov Chain (NSMC06)*, pages 117–136, Charleston, 2006. Celebration of the 100th anniversary of Markov.

[9] J. Fourneau and N. Pekergin. An algorithmic approach to stochastic bounds. In *LNCS 2459, Performance evaluation of complex systems: Techniques and Tools*, pages 64–88, 2002.

[10] J.-M. Fourneau, M. L. Coz, F. QuessettD., J. Fourneau, and D. Nott. Algorithms for an irreducible and lumpable strong stochastic bound. *Special Issue on the Conference on the Numerical Solution of Markov Chains 2003, Linear Algebra and its Applications*, 386:167–185, 2004.

[11] P. Glasserman and D. Yao. *Monotone Structure in Discrete-Event Systems*. John Wiley & Sons, 1994.

[12] S. Haddad and P. Moreaux. Sub-stochastic matrix analysis for bounds computation: Theoretical results. *European Journal of Operational Research*, In Press.

[13] J. Ledoux and L. Truffet. Markovian bounds on functions of finite Markov chains. *Advances in Applied Probability*, 33(2):505–519, 2001.

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

[15] N. Pekergin. Stochastic bounds on delays of fair queueing algorithms. In *Proceedings of INFOCOM'99*, pages 1212–1219, New York, NY, 1999.

[16] J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1&2):223–252, 1996.

[17] F. Quessette. Couplage depuis le passé, Feb. 2003. ROADEF'2003, Avignon.

[18] O. Stenflo. *Ergodic theorems fory Iterated Function Systems controlled by stochastic sequences*. Doctoral thesis n. 14, Umea university, 1998.

[19] O. Stenflo. Ergodic theorems for markov chains represented by iterated function systems. *Bull. Polish Acad. Sci. Math*, 2001.

[20] J.-M. Vincent and C. Marchand. On the exact simulation of functionals of stationary markov chains. *Linear Algebra and its Applications*, 386:285–310, 2004.

[21] J.-M. Vincent and J. Vienne. Perfect simulation of monotone systems with variance reduction. In *Proceedings of the 6th Int. Workshop on Rare Event Simulation*, pages 275–285, Bamberg, Oct. 2006.