Bounds based on lumpable matrices for partially ordered state space

Ana Bušić and Jean-Michel Fourneau
PRiSM, Université de Versailles
Saint-Quentin-en-Yvelines
78035 Versailles, France
{abusic, jmf}@prism.uvsq.fr

ABSTRACT

We consider the strong stochastic ("st") ordering on Markov chains whose states are naturally endowed with a partial order and we present an algorithmic derivation of lumpable upper bounding matrix.
The comparison of Markov chains relies on the monotonicity property even if it is only sufficient. When the model is not monotone, we must build a monotone upper bound. Quite often the natural partial order of the model makes the problem monotone with this order. However, if we use a total order consistent with the partial one the model is often not monotone anymore for the total order. Furthermore, the monotonicity property for total order adds a lot of constraints which are irrelevant and make the bound less accurate.
Here we assume that the model is monotone for the natural partial order and we must build a matrix which is larger in the strong stochastic sense and easier to solve. We build a lumpable bound and we store the lumped version to be analyzed.
Lumpable upper bounding matrices may also be useful when they are associated to a total order. We can derive a bound of the Markov chain with a two phase algorithm: first we derive a lumpable upper bound from the specifications, we store the lumped version and we make this lumped matrix monotone in a second phase.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Markov processes; C.4 [**Performance of systems**]: modeling techniques; performance attributes; reliability, availability, and serviceability

General Terms

Algorithms, Performance, Reliability

Keywords

Markov chains, stochastic comparison, lumpability

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SMCtools, October 10, 2006, Pisa, Italy.
Copyright 2006 ACM 1-59593-506-1 ...$5.00.

1. INTRODUCTION

It is well understood now that Markov chains easily model complex systems performance but also reliability and availability problems. The definition and generation of large-scale Markov models from high level specifications are relatively easy and efficient in both time and memory requirements due to numerous approaches and tools. The remaining difficulty is to actually solve the Markov chain and derive useful rewards to study performance or reliability.
Consider an irreducible finite continuous-time Markov chain X whose stochastic transition rate matrix is denoted by A. Then there exists the steady-state distribution of the Markov chain π which is the unique solution of equation $\pi A = 0$. Performance and reliability measures are related to the steady-state and transient distributions of this chain. For instance, the availability measure is defined by separating the states into two classes, UP states and DOWN states. A state is said to be UP if the system is operational for that state; otherwise it is DOWN. Let U denote the set of UP states. The reliability at time t is defined as the probability that the system has always been in the UP states between 0 and t:

$$R(t) = P(X_s \in U, \forall \ \ s \ \in \ [0, t[).$$

The point availability is the probability that the system is operational at time t:

$$PAV(t) = P(X_t \in U)$$

and the steady-state availability is the limit, if it exists, of this probability. It can also be defined as the expectation of a reward on the steady state distribution of X: $\sum_{i|i \text{ is up}} \pi(i)$.
We must compute transient and steady state probability distributions for matrix A. But for many problems these matrices are so huge that this is not even possible to store them in sparse form or to find steady-state or transient distributions. Thus we must use methods which provide a guarantee or a bound on these performance or reliability measures and which are not limited by the size of the state space. As we need to bound both transient and steady state distributions, the polyhedral approach based on Courtois and Semal's results [6], improved by Muntz et al. [15, 10], Carrasco [4] and Mahevas and Rubino [11], does not apply as it only deals with steady-state equations.
Fortunately enough, the stochastic comparison of Markov chains provides bounds for both transient and steady-state measures. Strong aggregation is also consistent with transient and steady-state analysis of a chain and the aggregated chain is smaller and is expected to be easier to analyze. In

[9] a theory for an algorithmic derivation for strong stochastic comparison of Markov chains based on necessary conditions on its transition matrix has been presented. Then algorithms based on lumpability have been proved [7, 3] and a tool has been demonstrated [8].

The comparison of Markov chains requires an order on the state space. Let $\preceq_S$ denote the order on the states and $\preceq_F$ the order on the distributions. The order on the states may be total or only partial. To compare two Markov chains the fundamental theorem (see [14] and the next section) states that it is sufficient that one of the two transition matrices is monotone. Intuitively (see the next section for the definition), a transition probability matrix P is monotone if and only if for all distribution vectors u and v, if $u \preceq_F v$ then $uP \preceq_F vP$. Note that we can choose these two orders ($\preceq_S$ and $\preceq_F$) according to the model requirements: the order on the distributions must be consistent with the rewards we study, while the order on the states has many implications on the monotonicity property and on the set of constraints we must check. Usually, one considers a total ordering on the states and strong stochastic (also denoted as "st" or sample-path) ordering on the distributions for the sake of simplicity. Basically, the set of constraints to derive the monotonicity property is large for an arbitrary partial order $\preceq_S$. However, when the ordering on the states is total, this set can be heavily reduced to finally obtain a linear number of constraints.

Most of the problems we analyze are endowed with a natural partial order. Multi-component systems are often modeled with Queuing Networks, Petri Nets or Stochastic Process Algebra. These formalisms allow to define local behavior and interconnection or composition of these sub-models. States in a sub-model often have a natural order: the number of customers in a queue, the number of tokens in a place of a Petri net are clearly endowed with the natural order on the integers. But the composition of these sub-models is only associated to a partial order. Quite often the natural partial order of the model makes the problem monotone with this order. However, if we use a total order consistent with the partial one the model is not monotone anymore for the total order. For instance some Queuing network routings are monotone when we consider partial order but not for the total order [13]. Forcing the total order monotonicity when bounding a model that is monotone for the natural partial order of the considered system adds a lot of irrelevant constraints that make the bound less accurate.

Here we assume that the model is monotone for the natural partial order and we must build a matrix which is larger in the stochastic sense for this partial ordering of the state space and which is also easier to solve. As in [7, 3] we build the bound to be lumpable and we store the lumped version to be analyzed. However, the approach is quite different of the former algorithms like LIMSUB [7] and LMSUB [3] as we do not need to check the monotonicity property. We just have to build a lumpable matrix which is larger according to the stochastic ordering on partially ordered states. Note that the comparison of matrices is also quite different when the state space is only partially ordered and relies on the theory of increasing sets developed by Massey [12].

When we consider a totally ordered state space, it may be also useful to build a lumpable upper bound of a matrix. Indeed, if the matrix size is quite large it is easier to use a two phase algorithm rather than LMSUB which requires the matrix as input: first we derive a lumpable upper bound from the specifications, we store the lumped version and we make this lumped matrix monotone in a second phase. This two phase algorithm will not require the initial matrix being generated and stored. To illustrate this last point we consider the modeling of highly available multicomponent systems such as the example studied by Muntz et al. [15] and Carrasco [4]. A typical system consists of several disks, CPUs and controllers. We have two types of failures: soft and hard. The failures may occur in batches and all the failed items compete to be repaired. The system is operational (i.e. UP) if there is enough CPUs, disks and controllers. Clearly, if the number of components is large, the state space is huge and the UP states are relatively rare. Furthermore, if the system is highly available, the UP states concentrate most of the probability distribution. For instance, the system depicted in Fig. 1 has more than $9.0\ 10^{10}$ states and 10^{12} transitions. Thus it is even not possible to generate and store the state space and the transition matrix. In [3] we have presented the global approach. Due to the size of the paper it had not been possible to present the algorithmic derivation of lumpable chains. Thus we present here several algorithms to perform this task. We have considered total order on the states to analyze the Muntz's problem and we have also extended the approach to partial orders.

The paper is organized as follows. Section 2 contains a short introduction to stochastic ordering on a partially ordered state space. In Section 3, we present the LL (Lumpable and Larger) algorithm for a total order to simplify the presentation. Section 4 is devoted to an example showing that the approach is efficient to handle large problems. Then in Section 5 we consider state spaces endowed with a partial order and we derive new algorithms for computing lumpable bounding matrices.

2. STOCHASTIC ORDERING ON A PARTIALLY ORDERED SPACE

In this section we recall definitions of stochastic comparison on a totally or partially ordered state space and some fundamental results for the algorithmic bounding approach. Recall that a binary relation $\preceq_S$ on a set S is called a partial order on S if it is reflexive, transitive, and antisymmetric. If additionally for all $x, y \in S$ either $x \preceq_S y$ or $y \preceq_S x$ holds, then the relation $\preceq_S$ is called a total order on S. A typical example of a partial order that is not a total order is the product order on a product space. Let S be $\mathbb{N}^I$ or $\mathbb{R}^I$, where I is a countable set. The product order $\preceq_S$ on S is defined by $x, y \in S,\ x \preceq_S y$ if $x_i \leq y_i, \forall i \in I$.

Definition 1. Let $(S, \preceq_S)$ be a partially ordered space and let X and Y be two random variables on S. X is smaller than Y in a strong stochastic sense, $X \preceq_{st} Y$, if

$$\mathbf{E}[f(X)] \leq \mathbf{E}[f(Y)], \text{ for each increasing function } f,$$

provided that the expectations exist.

In the following, we will consider only a finite totally or partially ordered space $(S, \preceq_S)$, and we will use interchangeably $X \preceq_{st} Y$ and $x \preceq_{st} y$, where x and y denote the probability distribution vectors of random variables X and Y. When we consider two different orders $\preceq_A$ and $\preceq_B$ on the same state space S, we will denote by $\preceq_{st,A}$ the strong stochastic comparison on $(S, \preceq_A)$ and by $\preceq_{st,B}$ the strong stochastic

comparison on $(S, \preceq_B)$ to avoid confusion.
Strong stochastic comparison of random variables on a partially ordered set $(S, \preceq_S)$ can be characterized by means of increasing sets. A subset $U \in S$ is called an increasing set if its indicator function $\mathbf{1}_U$ is increasing. It follows that U is an increasing set if and only if $x \in U$ and $x \preceq_S y$ imply $y \in U$. The following characterization is often used as definition of $\preceq_{st}$-order on a partially ordered space S [12]. The proof can be found in [14].

PROPOSITION 1. *$X \preceq_{st} Y$ if and only if $P(X \in U) \leq P(Y \in U)$, for all increasing sets $U \subset S$.*

In the case of a finite totally ordered set $(S, \preceq_S)$, $|S| = n$, there are exactly n different increasing sets $U \neq \emptyset$. For instance, let $S = \{1, 2, \ldots, 5\}$ with the usual ordering relation $1 \preceq_S 2 \preceq_S \cdots \preceq_S 5$, then all the increasing sets $\emptyset \neq U \subset S$ are: $\{5\}$, $\{4, 5\}$, $\{3, 4, 5\}$, $\{2, 3, 4, 5\}$ and S. The $\preceq_{st}$-comparison of two probability vectors can then simply be characterized as:

PROPOSITION 2. *Let X and Y be two random variables on $S = \{1, 2, \ldots, n\}$ with $1 \preceq_S 2 \preceq_S \ldots \preceq_S n$, and let x and y be their probability distribution vectors. Then*

$$X \preceq_{st} Y \iff \sum_{k=j}^{n} x_k \leq \sum_{k=j}^{n} y_k, \ j = 1, 2, \ldots, n.$$

Let us now consider an example of a partial order that is not a total order. Let $S = \{1, 2, 3, 4, 5\}$ and let the ordering relation $\preceq_A$ be defined as $1 \preceq_A i \preceq_A 5$, $\forall i \in S$. Note that states $2, 3, 4$ are not comparable. There are 9 increasing sets $U \neq \emptyset$: $\{5\}$, $\{2, 5\}$, $\{3, 5\}$, $\{4, 5\}$, $\{2, 3, 5\}$, $\{2, 4, 5\}$, $\{3, 4, 5\}$, $\{2, 3, 4, 5\}$ and S. If we consider the random variables X, Y and Z with the following distribution vectors:

$$\begin{aligned} x &= (0.3, 0.3, 0.1, 0.1, 0.2), \\ y &= (0.3, 0.1, 0.2, 0.1, 0.3), \\ z &= (0.1, 0.2, 0.2, 0.1, 0.4), \end{aligned}$$

then we have $X \preceq_{st,A} Z$ and $Y \preceq_{st,A} Z$, but X and Y are not comparable in the $\preceq_{st,A}$-sense since $P(X = 5) = 0.2 < P(Y = 5) = 0.3$ but $P(X \in \{2, 5\}) = 0.5 > P(Y \in \{2, 5\}) = 0.4$. However, if we consider the total order $\preceq_B$ on S, $1 \preceq_B 2 \preceq_B \ldots \preceq_B 5$, then we can notice that $X \preceq_{st,B} Y \preceq_{st,B} Z$.
Note that $\preceq_A \subset \preceq_B$. We say that the total order $\preceq_B$ is consistent with the partial order $\preceq_A$ on S. For an arbitrary ordering relation $\preceq_A$ such that the total order $\preceq_B$ on S is consistent with $\preceq_A$, the number of non-trivial constraints to be satisfied in Proposition 1 is larger. In our simple example there are 4 non-trivial constraints for the totally ordered space $(S, \preceq_B)$ and 8 for the partially ordered space $(S, \preceq_A)$. Thus the total order is far more convenient for the algorithmic approach. For larger partially ordered spaces we are faced with the problem of the combinatorial explosion of the non-trivial constraints to be satisfied. The above example shows that we can easily construct partial orderings on $|S| = n$ with $\theta(2^n)$ non-trivial comparison constraints.
In the following we consider some properties of homogeneous Discrete Time Markov Chains (DTMCs) on a finite partially ordered state space $(S, \preceq_S)$. Our algorithms apply on DTMCs. Continuous time Markov chains (such as the example in Section 4) must be uniformized first. We recall the definitions of monotonicity property and the comparison for transition matrices. Those two properties yield sufficient conditions for stochastic comparison of two DTMCs [14].

Definition 2. A transition matrix P of a homogeneous DTMC $\{X_t\}$ is monotone if for all probability vectors x and y, $x \preceq_{st} y$ implies $xP \preceq_{st} yP$.

The monotonicity property for a transition matrix of a homogeneous DTMC simply states that if the distributions at time t (x and y in the above definition) are ordered, the relation is preserved at time $t + 1$. The monotonicity property strongly relies on the order considered. We will see in Section 5 that a chain that is monotone under a partial order $\preceq_A$ on the state space S does not necessarily need to be monotone under a total order $\preceq_B$ on S even when $\preceq_B$ is consistent with $\preceq_A$.
Let us denote $P_{i,*}$ the row i of the transition matrix P. We have the following characterization of $\preceq_{st}$-monotonicity (see [14]).

PROPOSITION 3. *Let $\{X_t\}$ be a homogeneous DTMC on a partially ordered state space $(S, \preceq_S)$. The transition matrix P of $\{X_t\}$ is $\preceq_{st}$-monotone if for all $i, j \in S$,*

$$i \preceq_S j \implies P_{i,*} \preceq_{st} P_{j,*},$$

i.e. if $\sum_{k \in U} P_{i,k} \leq \sum_{k \in U} P_{j,k}$ for all increasing sets U.

Let P and Q be transition matrices of two homogeneous DTMCs on the same state space.

Definition 3. For transition matrices P and Q we say that $P \preceq_{st} Q$ if

$$P_{i,*} \preceq_{st} Q_{i,*} \text{ for all } i \in S,$$

i.e. if $\sum_{k \in U} P_{i,k} \leq \sum_{k \in U} Q_{i,k}$ for all increasing sets U.

We give now the classical comparison theorem for two homogeneous DTMCs. The proof of this theorem can be found in [14].

THEOREM 1. *Let $(S, \preceq_S)$ be a partially ordered space and let $\{X_t\}_{t \geq 0}$ and $\{Y_t\}_{t \geq 0}$ be two DTMCs and P and Q their respective transition matrices. If*

- *$X_0 \preceq_{st} Y_0$,*
- *at least one transition matrix P or Q is $\preceq_{st}$-monotone,*
- *$P \preceq_{st} Q$,*

then $X_t \preceq_{st} Y_t$, for all $t \geq 0$. If $\{X_t\}$ and $\{Y_t\}$ have steady-state distributions π_X and π_Y, then $\pi_X \preceq_{st} \pi_Y$.

3. LUMPABLE AND LARGER (LL) ALGORITHM

Let $S = \{1, 2, \ldots, n\}$ be a totally ordered space with the usual ordering relation $\leq$. We suppose an event based description of a DTMC model. A DTMC $X = \{X_t\}$ on S is fully defined by $(\mathcal{E}^X, \mathcal{A}^X, \Phi^X, \mathcal{D}^X)$, where $\mathcal{E}^X$ is the list of all the possible events in the system, and $\mathcal{A}^X = \{A_e^X \mid e \in \mathcal{E}^X\}$, $\Phi^X = \{\phi_e^X \mid e \in \mathcal{E}^X\}$ and $\mathcal{D}^X = \{d_e^X \mid e \in \mathcal{E}^X\}$ are defined as follows:

- $A_e^X \subset S$ denotes the states in which event e occurs with a strictly positive probability.

- $\phi_e^X : A_e^X \to (0,1]$.
  For $s \in A_e^X$, $\phi_e^X(s)$ is the strictly positive probability of event e when the system is in state s.

- $d_e^X : A_e^X \to S$.
  For $s \in A_e^X$, $d_e^X(s)$ is the new (destination) state of the chain after the occurrence of event e in the system that was in state s.

Denote by $S = \cup_{i=1}^r M_i,\ M_i \cap M_j = \emptyset,\ i \neq j$, a partition of the state space S into r macro-states.

Hypothesis 1. The states of the same macro-state are successive, i.e. for $i, j \in \{1, 2, \dots, r\}$ such that $i \leq j$,

$$\forall v \in M_i, \forall w \in M_j, v \leq w.$$

Let $g : S \to \{1, 2, \dots, r\}$ be the projection function defined by the partition $M_1, \dots, M_r$,

$$g(s) = i \Leftrightarrow s \in M_i, \forall s \in S.$$

Then g^{-1} denotes the generalized inverse of g,

$$g^{-1}(i) = \{s \in S \mid g(s) = i\},\ 1 \leq i \leq r. \qquad (1)$$

3.1 LL algorithm

Suppose that we can easily obtain the following information for each macro-state M_i:

1. The list of all the events that may occur in macro-state M_i,
$$\mathcal{E}_i^X = \{e | A_e^X \cap M_i \neq \emptyset\}.$$
2. A function $m_i^X : \mathcal{E}_i^X \to \{1, 2, \dots, r\}$, where $m_i^X(e)$ is the index of the greatest macro-state M_j that can be reached from states of macro-state M_i by the occurrence of event $e \in \mathcal{E}_i^X$:
$$m_i^X(e) = \max_{s \in A_e^X \cap M_i} \left\{ g(d_e^X(s)) \right\}. \qquad (2)$$
3. The maximal values of ϕ_e^X within each macro-state i:
$$\phi_{e,i}^X = \max_{s \in A_e^X \cap M_i} \phi_e^X(s). \qquad (3)$$

Then we can easily build a chain Y on S such that Y is lumpable under the partition $M_1, \dots, M_r$ of the state space S and that the transition matrix of Y is $\preceq_{st}$-larger than the transition matrix of X. We will actually construct only the chain Z, the aggregated version of Y. Denote by $(\mathcal{E}^Z, \mathcal{A}^Z, \Phi^Z, \mathcal{D}^Z)$ the chain description of Z. The construction of Z is given in Algorithm 1.

Proposition 4. *Let X be an arbitrary DTMC and Z the chain obtained by Algorithm 1. Then there exists a DTMC Y on S such that:*

- *Y is lumpable under the partition $M_1, \dots, M_r$ of S,*
- *Z is the aggregated version of Y,*
- *$P^X \preceq_{st} P^Y$, where P^X and P^Y denote the transition matrices of DTMCs X and Y.*

Algorithm 1: LL

($\mathcal{E}^Z$ and $A_e^Z, \forall e$ are initialized to $\emptyset$)
for $i = 1$ **to** r **do**
  Sort $\mathcal{E}_i^X$ decreasingly in $m_i^X(\cdot)$
  Denote by e_j the j^{th} event in the sorted list
  $p \leftarrow 0,\ j \leftarrow 1$
  while $p < 1$ **do**
    $e \leftarrow e_j$
    $\mathcal{E}^Z \leftarrow \mathcal{E}^Z \cup \{e\},\ A_e^Z \leftarrow A_e^Z \cup \{i\}$
    $d_e^Z(i) \leftarrow m_i^X(e)$
    if $\phi_{e,i}^X + p < 1$ **then** $\phi_e^Z(i) \leftarrow \phi_{e,i}^X$
    else $\phi_e^Z(i) \leftarrow 1 - p$
    $p \leftarrow p + \phi_e^Z(i),\ j \leftarrow j + 1$
  end
end

Proof. Denote by $upp(i)$ the greatest state of macro-state M_i. Consider the chain Y given by $(\mathcal{E}^Y, \mathcal{A}^Y, \Phi^Y, \mathcal{D}^Y)$, where:

$$\begin{cases} \mathcal{E}^Y = \mathcal{E}^Z \\ \forall e \in \mathcal{E}^Y : \qquad A_e^Y = \cup_{i \in A_e^Z} M_i \\ \forall e \in \mathcal{E}^Y,\ s \in A_e^Y \cap M_i : \\ \qquad \phi_e^Y(s) = \phi_e^Z(i), \qquad d_e^Y(s) = upp(d_e^Z(i)). \end{cases} \qquad (4)$$

It is obvious that Y is lumpable and that Z is the aggregated version of Y. It remains us to show that $P^X \preceq_{st} P^Y$, i.e. (see Definition 3 and Proposition 2)

$$\sum_{u=v}^{n} P_{s,u}^X \leq \sum_{u=v}^{n} P_{s,u}^Y, \forall s, v \in S. \qquad (5)$$

Let us define the subsets $C_{s,v}^X \subset \mathcal{E}^X,\ C_{s,v}^Y \subset \mathcal{E}^Y,\ s, v \in S$ and $C_{i,j}^Z \subset \mathcal{E}^Z,\ i, j \in \{1, \dots, r\}$ as follows:

$$\begin{aligned} C_{s,v}^X &= \{e \in \mathcal{E}^X \mid s \in A_e^X \text{ and } d_e^X(s) \geq v\}, \\ C_{s,v}^Y &= \{e \in \mathcal{E}^Y \mid s \in A_e^Y \text{ and } d_e^Y(s) \geq v\}, \\ C_{i,j}^Z &= \{e \in \mathcal{E}^Z \mid i \in A_e^Z \text{ and } d_e^Z(i) \geq j\}. \end{aligned}$$

Then,

$$\sum_{u=v}^{n} P_{s,u}^X = \sum_{e \in C_{s,v}^X} \phi_e^X(s) \text{ and } \sum_{u=v}^{n} P_{s,u}^Y = \sum_{e \in C_{s,v}^Y} \phi_e^Y(s),$$

for all $s, v \in S$. From (4) it follows that

$$\begin{aligned} \textstyle\sum_{e \in C_{s,v}^Y} \phi_e^Y(s) &= \textstyle\sum_{e \in C_{s,upp(M_{g(v)})}^Y} \phi_e^Y(s) \\ &= \textstyle\sum_{e \in C_{g(s),g(v)}^Z} \phi_e^Z(g(s)),\ s, v \in S. \end{aligned}$$

Equation (5) can be therefore written as:

$$\sum_{e \in C_{s,v}^X} \phi_e^X(s) \leq \sum_{e \in C_{g(s),g(v)}^Z} \phi_e^Z(g(s)), \forall s, v \in S. \qquad (6)$$

The events in Algorithm 1 are considered in decreasing order with the respect to their maximal destination $m_{g(s)}^X(\cdot)$. Denote by

$$\delta = \min_{e \in A_e^Z} m_{g(s)}^X(e).$$

For $g(v) \leq \delta$, $\sum_{e \in C_{g(s),g(v)}^Z} \phi_e^Z(g(s)) = 1$, thus (6) trivially holds. By construction of chain Z in Algorithm 1 and Hypothesis 1 which implies that g is increasing, if $g(v) > \delta$, then for each event $e \in C^X_{s,v}$ we have $m^X_{g(s)}(e) \geq g(d^X_e(s)) \geq g(v) > \delta$, thus $e \in C^Z_{g(s),g(v)}$ (see lines 8 and 9 of Algorithm 1) and $\phi^Z_e(g(s)) = \phi^X_{e,g(s)} \geq \phi^X_e(s)$ (see line 10 of Algorithm 1 and (3)). Therefore (6) holds for all $s, v \in S$. Recall that (6) is equivalent to (5), thus $P^X \preceq_{st} P^Y$. $\square$

3.2 Example

We will illustrate LL-algorithm on a toy example of a repairable multicomponent system (see also Section 4). A typical repairable multicomponent system consists of different types of components and a certain number of units per component type. Suppose that we have a system that consists of only 2 component types and K units of each component type. The state space of our model is then

$$S = \{(x_1, x_2) \mid 0 \leq x_i \leq K, i = 1, 2\}$$

where x_i denotes the number of failed components of type i. We consider a total order $\preceq_S$ on S that is consistent with the usual product order $\preceq_A$ on S: $x \preceq_A y \Leftrightarrow x_1 \leq y_1$ and $x_2 \leq y_2$. The detailed description of $\preceq_S$ is irrelevant for this example. Only one component of each type is active, if there is any operational component of that type, and only active components can fail. Both failures and reparations are exponentially distributed. There are three types of failures: failure of active component of type 1 with rate λ_1, failure of active component of type 2 with rate λ_2 and the failure of all active components in the system with rate λ_3. The third type failure may occur if there is at least one operational component ($x_1 + x_2 < 2K$) and induces two new failures in the case when there is at least one operational component of each type ($x_1 < K$ and $x_2 < K$) or only one new failure otherwise. There is only one repairman with reparation rate μ and if there are failed components of both types he chooses to repair a component of type 1 with probability $\frac{1}{2}$. We suppose that $\mu > \lambda_1 + \lambda_2 + \lambda_3$. Denote the uniformization constant by Δ. Then $a_i = \frac{\lambda_i}{\Delta}$, $i = 1, 2, 3$, and $b = \frac{\mu}{\Delta}$ denote failure and reparation probabilities in uniformized chain X:

- $\mathcal{E}^X = \{f_1, f_2, f_3, r_1, r_2, e_{norm}\}$:
  - f_1: failure of active component 1,
  - f_2: failure of active component 2,
  - f_3: failure of all active components,
  - r_1: reparation of component 1,
  - r_2: reparation of component 2,
  - e_{norm}: event corresponding to uniformization.
- $\mathcal{A}^X$:
  - $A^X_{f_1} = \{x \in S \mid x_1 < K\}$,
  - $A^X_{f_2} = \{x \in S \mid x_2 < K\}$,
  - $A^X_{f_3} = A^X_{f_1} \cup A^X_{f_2}$,
  - $A^X_{r_1} = \{x \in S \mid x_1 > 0\}$,
  - $A^X_{r_2} = \{x \in S \mid x_2 > 0\}$,
  - $A^X_{e_{norm}} = S$.
- Φ^X and $\mathcal{D}^X$:
  - $\phi^X_{f_1}(x) = a_1,\ d^X_{f_1}(x) = (x_1 + 1, x_2),\ \forall x \in A^X_{f_1}$,
  - $\phi^X_{f_2}(x) = a_2,\ d^X_{f_2}(x) = (x_1, x_2 + 1),\ \forall x \in A^X_{f_2}$,
  - $\phi^X_{f_3}(x) = a_3,\ \forall x \in A^X_{f_3}$,
    $$d^X_{f_3}(x) = \begin{cases} (x_1 + 1, x_2 + 1), & x \in A^X_{f_1} \cap A^X_{f_2} \\ (x_1 + 1, x_2), & x \in A^X_{f_1} \backslash A^X_{f_2} \\ (x_1, x_2 + 1), & x \in A^X_{f_2} \backslash A^X_{f_1} \end{cases}$$
  - $\phi^X_{r_1}(x) = \begin{cases} \frac{b}{2}, & x \in A^X_{r_1} \cap A^X_{r_2} \\ b, & x \in A^X_{r_1} - A^X_{r_2} \end{cases}$
    $d^X_{r_1} = (x_1 - 1, x_2),\ \forall x \in A^X_{r_1}$,
  - $\phi^X_{r_2}(x) = \begin{cases} \frac{b}{2}, & x \in A^X_{r_1} \cap A^X_{r_2} \\ b, & x \in A^X_{r_2} \backslash A^X_{r_1} \end{cases}$
    $d^X_{r_2} = (x_1, x_2 - 1),\ \forall x \in A^X_{r_2}$,
  - $\phi^X_{e_{norm}}(x) = 1 - a_1 \cdot \mathbf{1}_{\{x \in A^X_{f_1}\}} - a_2 \cdot \mathbf{1}_{\{x \in A^X_{f_2}\}} - a_3 \cdot \mathbf{1}_{\{x \in A^X_{f_3}\}} - b \cdot \mathbf{1}_{\{x \in A^X_{r_1} \cup A^X_{r_2}\}}$,
    $d^X_{e_{norm}} = x,\ \forall x \in S$.

We will consider the following partition of the state space into $2K + 1$ macro-states:

$$M_i = \{x \mid x_1 + x_2 = i\}, \quad i \in \{0, 1, \ldots, 2K\}.$$

The chain Z obtained by Algorithm 1 has the same set of events: $\mathcal{E}^Z = \mathcal{E}^X$. Macro-states M_0 and M_{2K} are single-state macro-state so we need only to sum the probabilities of all the events going to the same macro-state. Note that $\mathcal{E}^X_0 = \{f_1, f_2, f_3, e_{norm}\}$ and $\mathcal{E}^X_{2K} = \{r_1, r_2, e_{norm}\}$. For a macro-state M_i, $0 < i < 2K$ we have:

- $\mathcal{E}^X_i = \mathcal{E}^X$
- $m^X_i(\cdot)$ and $\phi^X_{\cdot,i}$:
  - $m^X_i(f_1) = i + 1,\ \phi^X_{f_1,i} = a_1$
  - $m^X_i(f_2) = i + 1,\ \phi^X_{f_2,i} = a_2$
  - $m^X_i(f_3) = \begin{cases} i + 2, & i \leq 2K - 2 \\ 2K, & i = 2K - 1 \end{cases},\ \phi^X_{f_3,i} = a_3$
  - $m^X_i(r_1) = m^X_i(r_2) = i - 1$,
    $\phi^X_{r_1,i} = \phi^X_{r_2,i} = \begin{cases} b, & 0 < i \leq K \\ \frac{b}{2}, & i > K \end{cases}$
  - $m^X_i(e_{norm}) = i$,
    $$\phi^X_{e_{norm},i} = \begin{cases} 1 - a_1 - a_2 - a_3 - b, & i < K \\ 1 - \min(a_1, a_2) - a_3 - b, & K \leq i \leq 2K - 1 \end{cases}$$
    Indeed, in a macro-state M_i, $K \leq i \leq 2K - 1$ there is at least one state x such that $x_1 = K$, i.e. $x \notin A^X_{f_1}$ and thus $\phi^X_{e_{norm}}(x) = 1 - a_2 - a_3 - b$. Symmetrically, there is also at least one state $y \in M_i$ such that $x_2 = K$, i.e. $x \notin A^X_{f_2}$ with $\phi^X_{e_{norm}}(x) = 1 - a_1 - a_3 - b$. Therefore, $\phi^X_{e_{norm},i} = 1 - \min(a_1, a_2) - a_3 - b$.

Suppose that $\mu > 2\max(\lambda_1, \lambda_2)$. Then Algorithm 1 yields chain Z:

- $\mathcal{E}^Z = \mathcal{E}^X$
- $A^Z_{f_1} = A^Z_{f_2} = A^Z_{f_3} = \{i \mid 0 \leq i < 2K\}$,
  $A^Z_{r_1} = \{i \mid 1 \leq i \leq 2K\}$, $A^Z_{r_2} = \{i \mid K + 1 \leq i \leq 2K\}$
  $A^Z_{e_{norm}} = \{i \mid 0 \leq i \leq 2K\}$

- $\phi^Z_{f_1}(i) = a_1,\ i \in A^Z_{f_1},\ \phi^Z_{f_2}(i) = a_2,\ i \in A^Z_{f_2}$
  $\phi^Z_{f_3}(i) = a_3,\ i \in A^Z_{f_3}$
  $$\phi^Z_{r_1}(i) = \begin{cases} b, & 1 \le i < K \\ b - \max(a_1, a_2), & i = K \\ \frac{b}{2}, & K+1 \le i \le 2K \end{cases}$$
  $\phi^Z_{r_2}(i) = \frac{b}{2} - \max(a_1, a_2),\ i \in A^Z_{r_2}$
  $$\phi^Z_{e_{norm}}(i) = \begin{cases} 1 - a_1 - a_2 - a_3, & i = 0 \\ 1 - a_1 - a_2 - a_3 - b, & i < K \\ 1 - \min(a_1, a_2) - a_3 - b, & \\ & K \le i \le 2K-1 \\ 1 - b, & i = 2K \end{cases}$$
- $d^Z_e(i) = m^X_i(e), \forall e \in \mathcal{E}^Z$

3.3 Improving LL

In the basic version of LL-algorithm each event $e \in \mathcal{E}^X_i$ is considered separately. This is suitable for models where all the events have constant probability within each macro-state. When the probabilities of events depend on state of the chain this can introduce a significant bounding error. Here we present how we can improve the tightness of bounds by merging the events having the same maximal macro-state destination $m_i(\cdot)$. Let us first illustrate this on an example. Consider for instance the above toy example with the following modification: the individual failures (events f_1 and f_2) may occur also for non-active components. Denote the uniformization constant by $\widetilde{\Delta}$ and $\widetilde{a_i} = \frac{\lambda_i}{\widetilde{\Delta}},\ i = 1,2,3,\ \widetilde{b} = \frac{\mu}{\widetilde{\Delta}}$. We have

$$\widetilde{\phi}_{f_1}(x) = (K - x_1)\widetilde{a_1},\ \widetilde{\phi}_{f_2}(x) = (K - x_2)\widetilde{a_2}.$$

Let us consider a macro-state $M_i,\ 0 < i < K$. Then:

$$\widetilde{\phi}_{f_1,i} = K\widetilde{a_1},\ \widetilde{\phi}_{f_2,i} = K\widetilde{a_2}.$$

Indeed, there is at least one state $x \in M_i$ with no failed components of type 1 ($x_1 = 0$) and at least one state $y \in M_i$ with no failed components of type 2 ($x_2 = 0$). The while loop in Algorithm 1 gives the following for a macro-state $M_i,\ 0 < i < K$:

- Iteration 1: $p = 0 < 1$
  - $e \leftarrow f_3,\ \mathcal{E}^Z \leftarrow \mathcal{E}^Z \cup \{f_3\},\ A^Z_{f_3} \leftarrow A^Z_{f_3} \cup \{i\}$
  - $d^Z_{f_3}(i) = i + 2,\ \phi^Z_{f_3}(i) = \widetilde{a_3}$
  - $p \leftarrow \widetilde{a_3}$
- Iteration 2: $p = \widetilde{a_3} < 1$
  - $e \leftarrow f_1, \ldots$
  - $d^Z_{f_1}(i) = i + 1,\ \phi^Z_{f_1}(i) = K\widetilde{a_1}$
  - $p \leftarrow K\widetilde{a_1} + \widetilde{a_3}$
- Iteration 3: $p = K\widetilde{a_1} + \widetilde{a_3} < 1$
  - $e \leftarrow f_2, , \ldots$
  - $d^Z_{f_2}(i) = i + 1,\ \phi^Z_{f_2}(i) = K\widetilde{a_2}$
  - $p \leftarrow K\widetilde{a_1} + K\widetilde{a_2} + \widetilde{a_3}$
- Iteration 4 ...

Thus, in the basic version of LL (Algorithm 1) in the bounding chain we will have a transition $i \to i+1$ with probability

$$\widetilde{\phi}_{f_1,i} + \widetilde{\phi}_{f_2,i} = K\widetilde{a_1} + K\widetilde{a_2}.$$

However, remark that for all states in macro-state M_i we have only $2K - i$ non failed components. Both f_1 and f_2 induce the transition to M_{i+1}. If we considered events f_1 and f_2 as one event, say f, we can take:

$$\begin{aligned} \widetilde{\phi}_{f,i} &= \max_{x \in M_i} \left(\widetilde{\phi}_{f_1}(x) + \widetilde{\phi}_{f_2}(x) \right) \\ &= K \max(\widetilde{a_1}, \widetilde{a_2}) < K\widetilde{a_1} + K\widetilde{a_2}. \end{aligned}$$

Formally, suppose that we can easily obtain the following information for a group of events $\emptyset \ne \mathcal{G} \subset \mathcal{E}^X_i$:

$$\phi_{\mathcal{G},i} = \max_{x \in M_i} \left\{ \sum_{e \in \mathcal{G}} \phi^X_e(x) \right\}.$$

Then we can first merge the events with the same value of $m_i(\cdot)$ into one single event in Algorithm 1. For all $i, k \in \{1, 2, \ldots, r\}$ denote by $\mathcal{G}^X_{i,k}$ the set of all events $e \in \mathcal{E}^X_i$ having the maximal macro-state destination equal to k:

$$\mathcal{G}^X_{i,k} = \{e \in \mathcal{E}^X_i \mid m_i(e) = k\}.$$

Then for each k such that $\mathcal{G}^X_{i,k} \ne \emptyset$ we can consider $\mathcal{G}^X_{i,k}$ as one single event $g_{i,k}$ with maximal macro-state destination k and probability

$$\psi^X_{i,k} = \phi_{\mathcal{G}^X_{i,k},i}. \tag{7}$$

The modified LL-algorithm is given in Algorithm 2 and will be referred in the following as LL-1. Finally, we can improve further the probabilities $\psi^X_{i,k}$. Denote by $\mathcal{H}^X_{i,k}$ the events with $m_i(\cdot)$ greater than k:

Algorithm 2: LL-1

```
1  (E^Z and A^Z_g, ∀g are initialized to ∅)
2  for i = 1 to r do
3      Compute G^X_{i,k}, ∀k
4      L ← {k | G^X_{i,k} ≠ ∅}
5      p ← 0
6      while p < 1 do
7          j ← max{L} , L ← L\{j}
8          G ← G^X_{i,j}
9          g ← g_{i,j}
10         E^Z ← E^Z ∪ {g}, A^Z_g ← A^Z_g ∪ {i}
11         d^Z_g(i) ← j
12         ψ^X_{i,j} ← φ^X_{G,i}
13         if ψ^X_{i,j} + p < 1 then φ^Z_g(i) ← ψ^X_{i,j}
14         else φ^Z_g(i) ← 1 − p
15         p ← p + φ^Z_g(i)
16     end
17 end
```

$$\mathcal{H}^X_{i,k} = \cup_{j \ge k} \mathcal{G}^X_{i,j} = \{e \in \mathcal{E}^X_i \mid m_i(e) \ge k\}.$$

Let $\mathcal{H}^X_{i,r+1} := \emptyset$. Then we can take

$$\psi^X_{i,k} = \phi_{\mathcal{H}^X_{i,k},i} - \phi_{\mathcal{H}^X_{i,k+1},i}. \tag{8}$$

The proof follows easily from the fact that $\preceq_{st}$-comparison on a totally ordered space consists in comparing the tails of distributions (see Proposition 2). In order to take this improvement, called LL-2 algorithm in the following, into account one can just replace lines 8 and 12 of Algorithm 2 with:

- line 8: $\mathcal{H} \leftarrow \mathcal{H}^X_{i,j}$
- line 12: $\psi^X_{i,j} \leftarrow \phi^X_{\mathcal{H},i} - p$

Notice however that it is generally more difficult to compute $\phi_{\mathcal{H}^X_{i,k},i}$ than $\phi_{\mathcal{G}^X_{i,k},i}$ as the event subsets $\mathcal{H}^X_{i,k}$ are larger. Thus, even though LL-2 (see (8)) improves the tightness of bounds, LL-1 (see (7)) may be sometimes preferred for its lower complexity.

3.4 Computation of lower bounds

We can easily adapt Algorithms 1 and 2 to compute an aggregated chain Z of a DTMC Y with a transition matrix P^Y that is lumpable and smaller than the transition matrix P^X of DTMC X. In order to compute lower bounds for increasing rewards on X using Theorem 1, instead of using LIMSUB or LMSUB algorithm in the second step, we can use their equivalents LIMSLB or LMSLB that compute an $\preceq_{st}$-monotone lower bounding chain.
Instead of $m^X_i(e)$ (see (2)) we need to know $l^X_i(e)$, the minimal macro-state destinations of events $e \in \mathcal{E}^X_i$:

$$l^X_i(e) = \min_{s \in A^X_e \cap M_i} \left\{ g(d^X_e(s)) \right\}.$$

Algorithm 3 computes the aggregated chain Z of a DTMC Y such that P^Y is lumpable and smaller than P^X.

Algorithm 3: LS

> ($\mathcal{E}^Z$ and $A^Z_e, \forall e$ are initialized to $\emptyset$)
> **for** $i = 1$ **to** r **do**
>   Sort $\mathcal{E}^X_i$ increasingly in $l^X_i(\cdot)$
>   Denote by e_j the j^{th} event in the sorted list
> $p \leftarrow 0$, $j \leftarrow 1$
>   **while** $p < 1$ **do**
> $e \leftarrow e_j$
> $\mathcal{E}^Z \leftarrow \mathcal{E}^Z \cup \{e\}$, $A^Z_e \leftarrow A^Z_e \cup \{i\}$
> $d^Z_e(i) \leftarrow l^X_i(e)$
>   **if** $\phi^X_{e,i} + p < 1$ **then** $\phi^Z_e(i) \leftarrow \phi^X_{e,i}$
>   **else** $\phi^Z_e(i) \leftarrow 1 - p$
> $p \leftarrow p + \phi^Z_e(i)$, $j \leftarrow j + 1$
>   **end**
> **end**

PROPOSITION 5. *Let X be an arbitrary DTMC and Z the chain obtained by Algorithm 3. Then there exists a DTMC Y on S such that:*

- *Y is lumpable under the partition $M_1, \ldots, M_r$ of S,*
- *Z is the aggregated version of Y,*
- *$P^Y \preceq_{st} P^X$, where P^X and P^Y denote the transition matrices of DTMCs X and Y.*

PROOF. Denote by $low(i)$ the smallest state of macro-state M_i. The proof is similar to the proof of proposition 4 for the chain Y given by $(\mathcal{E}^Y, \mathcal{A}^Y, \Phi^Y, \mathcal{D}^Y)$, where:

$$\begin{cases} \mathcal{E}^Y = \mathcal{E}^Z \\ \forall e \in \mathcal{E}^Y: \quad A^Y_e = \cup_{i \in A^Z_e} M_i \\ \forall e \in \mathcal{E}^Y,\ s \in A^Y_e \cap M_i: \\ \quad \phi^Y_e(s) = \phi^Z_e(i), \qquad d^Y_e(s) = low(d^Z_e(i)). \end{cases} \tag{9}$$

□

Algorithm 2 can be adapted for lower bounds by defining the subsets $\mathcal{G}^X_{i,k} \subset \mathcal{E}^X_i$ as:

$$\mathcal{G}^X_{i,k} = \{e \in \mathcal{E}^X_i \mid l_i(e) = k\}$$

and taking min in line 7 of Algorithm 2:

$$j \leftarrow \min\{\mathcal{L}\}.$$

Finally, for LL-2 algorithm the subsets $\mathcal{H}^X_{i,k}$ should be defined as:

$$\mathcal{H}^X_{i,k} = \cup_{j \le k} \mathcal{G}^X_{i,j} = \{e \in \mathcal{E}^X_i \mid l_i(e) \le k\}.$$

Let $\mathcal{H}^X_{i,0} := \emptyset$. Then (8) should be modified for lower bounds to the following:

$$\psi^X_{i,k} = \phi_{\mathcal{H}^X_{i,k},i} - \phi_{\mathcal{H}^X_{i,k-1},i}.$$

4. THE AVAILABILITY OF REPAIRABLE MULTICOMPONENT SYSTEMS

Muntz et al. [15] (see also [4]) considered the following multicomponent system that consists of several disks, CPUs and controllers (Figure 1). There are two failure modes (soft and hard) which occur with equal probability. The failures of components are exponentially distributed with distinct rates for all the components. The system is operational if at least one of processors PA or PB is operational, at least one controller of each type and at least three out of four disks of each of the six clusters are operational. Only one processor of each type is active and only the active processors can fail. A failure of active processor PA may provoke a failure of the active processor PB with a fixed probability. There is only one repairman who chooses the component to be repaired at random from the set of failed components. The repair times follow exponential distributions. The repair rates depend on the type of components, the type of error (soft or hard) and the availability of the system (UP or DOWN). When the system is DOWN, the rates are 10 times larger as the consequence of the additional precautions to be taken when the system is operational. We model the system with a continuous time Markov chain and we begin by a uniformization process to design a discrete time chain which can be aggregated by algorithm LL. We used LL-2 version of Algorithm 2 (see (8)).

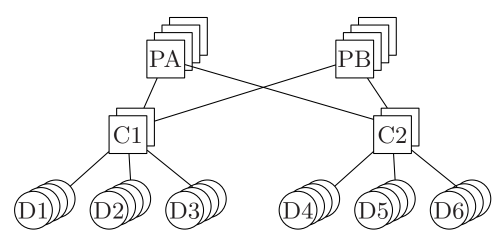

Figure 1: A typical system of duplicated components (Muntz et al.)

Let us now design the macro states we build from the initial states of the chain. A state is described by the number of failed components with hard failures and the number of failed components with soft failures. The model has only 36 components of 10 different types yet the size of the state space is roughly 9.0 10^{10}. Most of the states are DOWN states. Thus we do not aggregate the UP states. We aggregate the DOWN states which have the same total number of faults and then order the macro-states such that the UP states are smaller than the DOWN states. Let us now show the effects of the algorithm on this partition of the states into macro-states. We have three events: single failures, double failures (because a failure of processor PA may provoke a failure of processor PB) and repair. For this example LL algorithm changes the transitions as follows:

- The transition probabilities between simple states do not change.
- The transition from a simple state x to an aggregated state C is the sum of the transition probabilities from x to y, for all y in C.
- For an aggregated state C:
  - If in a macro state C there is a state x such that a double failure can occur in x, then it must now occur with the same probability for all the states in the macro-state. As the probability of a double failure is smaller than the probability of a single failure (this is a modeling assumption), the algorithm adds this transition from C to $C+2$ and do not delete the transition from C to $C+1$ but changes its probability.
  - For a macro-state C with total number of failures smaller than the total number of components (36), there is a transition from C to $C+1$ for all the states in C. The algorithm keeps this transition but it changes its probability such that the sum of transition probabilities to the macro-states $C+1$ and $C+2$ is equal to the maximal sum of those transitions in the original chain for all the states in C.
  - The sum of all the transitions corresponding to repairs is constant for all the states of an aggregated macro-state C, as the repair rate is fixed for the repairman and not for individual components. We distinguish two cases:
    1. There is at least one state x in C such that there are no transitions from x to an UP state. The algorithm deletes all the transitions from C to the UP states and the only transition corresponding to repairs is from C to $C-1$.
    2. For all states in C there is at least one transition to UP states. Algorithm modifies the destination of these transitions to the greatest UP state in the total order considered on the state space.

This partition gives a new model with 1 312 235 macro-states (1 312 200 UP and 35 DOWN macro-states) and 25 754 089 transitions. In [3] we have built this matrix and we have then applied a sparse matrix LMSUB algorithm to make it monotone and to reduce further the state space. Generation of lumped matrix by LL algorithm takes 182 seconds on an ordinary PC (CPU 3.20GHz, 1GB RAM), while the second step takes 152 seconds (reordering of states 95.5s and LMSUB algorithm 56.5s). Note that it was impossible to directly apply LMSUB algorithm because of the size of the original matrix (again roughly 9.0 10^{10}).

We want to emphasize that this method allows computation of bounds for both steady-state and transient increasing rewards such as steady-state availability, point availability and reliability. The numerical results for these rewards can be found in [3].

Deriving a lumped matrix is easier because we do not work on the state space. The algorithm is based on the set of events and the macro-state definition. It is required that the macro states will not be generated during this algorithm to avoid the state space generation. But the events and the transitions (source, destination and rates) must be described in such a way that the operations required are simple to implement. Typically we must:

- Preprocessing: Describe the macro-states
- For each event:
  - Describe the set of initial states
  - Describe the effect of the event
  - Describe the probability of the event
- Main Operation: For each event e and Macro State C_1:
  - Find the macro state C_2 with the largest number which is reached by event e for a state in C_1.
  - Find the maximal probability of event e in C_1.

The main operation has typically a complexity which is the product of the number of events by the number of macro-states which can be reached by an event. Thus the complexity depends on the events but also on the macro-state definition. At the time being the algorithms have been written for specific problems where these events and macro-states are hard coded into the model. Deriving a general algorithm will require a formal definition of events and macro-states.

5. PARTIAL ORDERING AND COMPARISON OF STOCHASTIC MATRICES

Quite often, one can observe that when we use a natural partial ordering, the Markov chain of the model is monotone (see for instance [13] for routing in queuing network and [2] for a model of optical switch). Unfortunately, it is rarely possible to find some total order on the state space under which the model remains monotone.

5.1 Bounding monotone DTMCs on a partially ordered state space using total ordering

Suppose that we want to analyze a DTMC X that is $\preceq_{st,A}$-monotone under some partial order $\preceq_A$ on the state space S. Additionally suppose that S is too big for the exact analysis. The usual approach [9, 8, 3] consists in considering a total order $\preceq_B$ on S. By forcing a total order $\preceq_B$ on the state space one introduces some unnecessary perturbation. Indeed, as the initial chain is no longer $\preceq_{st,B}$-monotone under the total order $\preceq_B$, we need to force the $\preceq_{st,B}$-monotonicity of the bounding chain. The monotonicity constraints under a total order are rather strong and can result by quite loose bounds as $\preceq_B$ is not a natural order on the state space for the considered system. We will illustrate this on a small example.

Example 1. Let us consider the Markov chain with the state space $S = \{1, 2, 3, 4, 5\}$ and transition matrix P:

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

Assume that we have a partial order on the state space defined by $1 \preceq_A 2 \preceq_A 5$ and $3 \preceq_A 4 \preceq_A 5$, then according to Proposition 3 the chain is $\preceq_{st,A}$-monotone.
If, like with the methods presented in [9], we do not know how to take into account a partial order, we must first build a total order on the state space consistent with the partial order $\preceq_A$. In this case, one can consider for instance : $1 \preceq_B 2 \preceq_B 3 \preceq_B 4 \preceq_B 5$. Now the chain is not monotone for this order and the best monotone upper bounding matrix computed by Vincent's algorithm [1] (see also [9]) is:

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

Let us now compute the two steady-state distributions:

$$\pi_P = (0.182, 0.303, 0.115, 0.212, 0.188),$$
$$\pi_Q = (0.125, 0.287, 0.115, 0.245, 0.228).$$

Clearly, using a total order when the problem is based on a partial order makes the result less accurate.

5.2 LL algorithm for an "st"-monotone chain on a partially ordered state space

We propose here another method that exploits fully the monotonicity of the system under the natural partial order on the state space S. Indeed, one can notice that LL-algorithm presented in Section 3 does not rely on the fact that the state space $(S, \preceq_S)$ is totally ordered. We will show here how we can relax this hypothesis. However, in the case of partially ordered space some additional care should be exercised when choosing the partition of the state space.
Let $(S, \preceq_S)$ be a partially ordered state space that admits an upper bound for any two states in S:

Hypothesis 2. For all $x, y \in S$, there exists $z \in S$ such that $x \preceq_S z$ and $y \preceq_S z$.

Then, by induction, there exists an upper bound for an arbitrary subset $M \subset S$. Let $S = M_1, \dots, M_r,\ M_i \cap M_j = \emptyset$ be an arbitrary partition of the state space. Denote by $upp_i \in S$ the upper bound of macro-state M_i:

$$s \preceq_S upp_i,\ \forall s \in M_i.$$

Note that upp_i does not necessarily need to be in M_i. However, it is usually better to take a partition of S such that $upp_i \in M_i$.
Suppose that a DTMC X is monotone on $(S, \preceq_S)$. Then we can use a very simple version (Algorithm 4) of LL-algorithm in order to build a chain Z, the aggregated version of a lumpable chain Y such that $P^X \preceq_{st} P^Y$. Note that, if we know the transition matrix P^X of chain X, then P^Z can be easily obtained as follows:

$$P^Z_{i,j} = \sum_{s \in g^{-1}(M_j)} P^X_{upp_i,s},\ \forall i, j \in \{1, 2, \dots, r\}. \qquad (10)$$

Recall that g^{-1} denotes the generalized inverse of projection function g (see (1)).

Algorithm 4: LL algorithm for an $\preceq_{st}$-monotone chain on a partially ordered state space

($\mathcal{E}^Z$ and $A^Z_e, \forall e$ are initialized to $\emptyset$)
for $i = 1$ **to** r **do**
  for each *e such that* $upp_i \in A^X_e$ **do**
    $\mathcal{E}^Z \leftarrow \mathcal{E}^Z \cup \{e\}$, $A^Z_e \leftarrow A^Z_e \cup \{i\}$
    $d^Z_e(i) \leftarrow g(d^X_e(upp_i))$
    $\phi^Z_e(i) \leftarrow \phi^X_e(upp_i)$
  end
end

Proposition 6. *Let $(S, \preceq_S)$ be a partially ordered space satisfying Hypothesis 2. Let X be an $\preceq_{st}$-monotone DTMC and Z the chain obtained by Algorithm 4. Then there exists a DTMC Y on S such that:*

- *Y is lumpable under the partition $M_1, \dots, M_r$ of S,*
- *Z is the aggregated version of Y,*
- *$P^X \preceq_{st} P^Y$, where P^X and P^Y denote the transition matrices of DTMCs X and Y.*

Proof. Let the chain Y given by $(\mathcal{E}^Y, \mathcal{A}^Y, \Phi^Y, \mathcal{D}^Y)$ be defined as follows:

$$\begin{cases} \mathcal{E}^Y = \mathcal{E}^Z \\ \forall e \in \mathcal{E}^Y : \qquad A^Y_e = \cup_{i \in A^Z_e} M_i \\ \forall e \in \mathcal{E}^Y,\ s \in A^Y_e \cap M_i : \\ \qquad \phi^Y_e(s) = \phi^Z_e(i), \qquad d^Y_e(s) = d^X_e(upp_i). \end{cases} \qquad (11)$$

Note that $g(d^Y_e(s)) = g(d^X_e(upp_i)) = d^Z_e(i)$. Y is obviously lumpable and Z is the aggregated version of Y.
Since X is $\preceq_{st}$-monotone on $(S, \preceq_S)$, it follows that

$$P^X_{x,*} \preceq_{st} P^X_{upp_i,*},\ \forall x \in M_i. \qquad (12)$$

On the other hand, from Algorithm 4 and (11) it follows that

$$P^X_{upp_i,*} =_{st} P^Y_{x,*},\ \forall x \in M_i. \qquad (13)$$

Thus $P^X \preceq_{st} P^Y$ follows now from (12) and (13). $\square$

The lower bounds can be computed using the same method. Let $(S, \preceq_S)$ be a partially ordered space that admits a lower bound for any two states in S:

Hypothesis 3. For all $x, y \in S$, there exists $z \in S$ such that $z \preceq_S x$ and $z \preceq_S y$.

Then for a partition $M_1, \dots, M_r$ of S denote by $low_i \in S$ the lower bound of macro-state M_i:

$$low_i \preceq_S s,\ \forall s \in M_i.$$

As in the case of an upper bound, the states low_i do not necessarily need to be in M_i it is better to take a partition of S for which this is the case. Let X be an $\preceq_{st}$-monotone DTMC on $(S, \preceq_S)$. If we know the transition matrix P^X

of chain X, then P^Z, the aggregated version of a lumpable chain Y such that $P^Y \preceq_{st} P^X$ can be easily obtained as:

$$P^Z_{i,j} = \sum_{s \in g^{-1}(M_j)} P^X_{low_i,s}, \; \forall i,j \in \{1,2,\ldots,r\}.$$

The algorithm that computes chain Z can be obtained from Algorithm 4 by simply replacing upp_i by low_i in lines 3, 5 and 6.

These algorithms are typically useful when we model a system whose rates are modulated by a Markov chain. Consider again the toy problem: if the failure rates and the repair rates are modulated by an external Markov chain, we must add a new component into the description of the state space. Using a partial order allows to only compare the states where the modulating phase has the same value (see [5] for an example of such an application of partial ordering).

6. CONCLUSION

The algorithms we proposed allow to study huge Markov chains without explicitly generating the state space. We must describe the states and the events. The description of states is straightforward but the events must be carefully defined as the main operation of the algorithm consists of finding some maximal transition in the macro-state. Clearly, this operation takes also into account the way we have defined the macro-states we want to consider.

At the time being we have only hard coded some problems to design the lumped chain for some well known hard examples like the availability problem studied by Muntz. To develop a complete algorithm one must first define a complete formalism where we can work on the states, on the events and on the set of events (i.e. macro states). This formalism will certainly be useful for other application as well. Stochastic Model Checking is a promising application for such a formalism.

Finally we want to stress again the importance of the monotonicity property for performance evaluation of systems for discrete-time or continuous-time models. The approach combining partial order and monotonicity is even still more general and many applications in numerical analysis or perfect simulation of Markov chains are still to be derived.

7. ACKNOWLEDGMENT

This research is supported by projects *SurePaths* and *SMS* from ACI Sécurité Informatique and by European Network of Excellence EuroNGI.

8. REFERENCES

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

[2] A. Busic, M. Ben Mamoun, and J.-M. Fourneau. Modeling fiber delay loops in an all optical switch. In *QEST 2006, Riverside, CA, USA*. IEEE Computer Society, 2006.

[3] A. Busic and J.-M. Fourneau. Bounds for point and steady-state availability: An algorithmic approach based on lumpability and stochastic ordering. In *EPEW 2005*, volume 3670 of *LNCS*, pages 94–108. Springer, 2005.

[4] J. A. Carrasco. Bounding steady-state availability models with group repair and phase type repair distributions. *Performance Evaluation*, 35:193–204, 1999.

[5] H. Castel-Taleb, J.-M. Fourneau, and N. Pekergin. Stochastic bounds on partial ordering: Application to memory overflows due to bursty arrivals. In *ISCIS 2005*, volume 3733 of *LNCS*, pages 244–253. Springer, 2005.

[6] P.-J. Courtois and P. Semal. Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition. *Journal of the ACM*, 31(4):804–825, 1984.

[7] J.-M. Fourneau, M. L. Coz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications*, 386:167–185, 2004.

[8] J.-M. Fourneau, M. Le Coz, N. Pekergin, and F. Quessette. An open tool to compute stochastic bounds on steady-state distributions and rewards. In *MASCOTS 2003, Orlando, FL, USA*, page 219, 2003.

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

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

[11] S. Mahevas and G. Rubino. Bound computation of dependability and performance measures. *IEEE Trans. Comput.*, 50(5):399–413, 2001.

[12] W. A. Massey. Stochastic orderings for Markov processes on partially ordered spaces. *Math. Oper. Res.*, 12(2):350–367, 1987.

[13] D. Mattson. *On perfect simulation of Markovian Queueing Networks with Blocking*. PhD thesis, Chalmers, Goteborg University, 2002.

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

[15] R. Muntz, E. de Souza e Silva, and A. Goyal. Bounding availability of repairable computer systems. *IEEE Trans. on Computers*, 38(12):1714–1723, 1989.