Algorithms for an Irreducible and Lumpable Strong Stochastic Bound

J.M. Fourneau, M. Lecoz and F. Quessette
PRiSM, Université de Versailles Saint-Quentin en Yvelines,
45 Av. des Etats Unis, 78000 Versailles, France

Motivation

- Building large Markov chains is easy (composition of sub-models in interaction, SAN, Petri nets, SPA, tensor representation).
- But numerical analysis of chains in steady-state is still difficult.
- Compute performance indices defined as reward functions on the steady-state distribution: $\sum_i r(i)\pi(i)$.
- It is often not necessary to compute the exact value of the distribution.
- Bounding some reward functions is sufficient.
- Here : stochastic bounds and algebraic algorithms on stochastic matrices, numerical analysis.

Methodology

- we have to model a problem with a very large Markov chain and compute its steady-state distribution.
- Design algorithmically a new chain (transition matrix) such that :
  - the new reward functions will be an upper bound of the exact reward functions and must be consistent with the macro states.
  - the new steady-state distribution is an upper bound of the exact steady-state.
  - the new matrix is lumpable (simpler to solve).
- Based on strong stochastic orderings for random variables, and Markov chains (see Stoyan).

Strong Stochastic Bounds

- Restriction (here) : Discrete Time Markov Chains (DTMC) and total order on the state space $E = \{1, \ldots, n\}$ (n is the size of the chain).
- $P_{i,*}$ will refer to row i of P.
- **Definition 1** *If X and Y take values on the finite state space $\{1, 2, \ldots, n\}$ with p and r as probability distribution vectors, then $X <_{st} Y$ if and only if $\sum_{j=k}^{n} p_j \leq \sum_{j=k}^{n} r_j$ for $k = 1, 2, \ldots, n$.*

Fundamental theorem for MC comparison

Theorem 1 *Let $X(t)$ and $Y(t)$ be two DTMC and P and R be their respective stochastic matrices. If*

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

Then, $X(t) <_{st} Y(t), t > 0$.

Relations

- If P is not monotone, R must be...

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

- It is possible to use a set of equalities, instead of inequalities.
- Properly ordered (in increasing order for i and in decreasing order for j in system 2), a constructive way to obtain a stochastic bound (Vincent's algorithm).

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

An example

$$P1 = \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ 0.1 & 0.7 & 0.1 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.5 & 0.2 & 0.0 \\ 0.1 & 0.0 & 0.1 & 0.7 & 0.1 \\ 0.0 & 0.2 & 0.2 & 0.1 & 0.5 \end{bmatrix} \quad \begin{bmatrix} 0.5 & 0.2 & 0.1 & 0.2 & 0.0 \\ & & & & 0.1 \\ & & & & 0.1 \\ & & & & 0.1 \\ & & & & 0.5 \end{bmatrix}$$

Compute column n (st-monotonicity implies that the elements are non decreasing)

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

Finally $\pi_{P1} = (0.180, 0.252, 0.184, 0.278, 0.106)$

$\pi_R = (0.143, 0.190, 0.167, 0.357, 0.143)$

$\pi_{P1} <_{st} \pi_R$

Irreducibility of R

- It may happen that the matrix computed by Vincent's algorithm is reducible, even if P is irreducible
- Due to the $-$ operation, some elements may be zero even if the corresponding elements in P are positive.
- A new algorithm (IMSUB) for the irreducibility
- Ideas : do not delete transitions and add the subdiagonal.
- a necessary and sufficient condition on P to obtain an irreducible upper bound.
- Another algorithm (LIMSUB) for a lumpable and irreducible bounding matrix.

Algorithm IMSUB

ϵ is an arbitrary positive value strictly smaller than 1.0.

step 1 : **For** $l = 1, 2 \ldots, n$ **Do**
  $r_{1,l} = p_{1,l}$
End For

step 2 : **For** $i = 2, 3, \ldots, n$ **Do**
  $r_{i,n} = \max(r_{i-1,n}, p_{i,n})$
End For

step 3 : **For** $l = n-1, n-2, \ldots, 1$ **Do**
  For $i = 2, 3, \ldots, n$ **Do**
step 3.a : $r_{i,l} = \max(0, \max(\sum_{j=l}^{n} r_{i-1,j}, \sum_{j=l}^{n} p_{i,l}) - \sum_{j=l+1}^{n} r_{i,j})$
step 3.b : **If** $(r_{i,l} = 0)$ and $(\sum_{j=l+1}^{n} r_{i,j} < 1)$ and $((p_{i,l} > 0) or (i = l-1))$ **Then**
    $r_{i,l} = \epsilon \times (1 - \sum_{j=l+1}^{n} r_{i,j})$
  End If
  End For
End For

- **Theorem 2** *Let P be an irreducible finite stochastic matrix. Matrix R computed from P by IMSUB algorithm is irreducible if and only if $P(1,1) > 0$ and every row of the lower triangle of matrix P has at least one positive element.*

Lumpable Bounds

- Lumpability implies a state space reduction. (decomposition of the chain into macro-states)
- **Definition 2** *Let X be a finite DTMC, Q its matrix, A_k be a partition of the states. X is ordinary lumpable according to A_k, iff for all states e and f in A_i, we have:*

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

- Ordinary lumpability is consistent with monotonicity.
- Assume that states are ordered according to the macro-state partition and that the rewards are consistent.
- The algorithm computes the matrix column by column with some particular work for block boundaries.

Ideas for algorithm LIMSUB

- Each block needs two steps.
- The first step is based on algorithm IMSUB but the first row of P and R may now be different due to the second step.
- The second step modifies the first column of the blocks to satisfy the ordinary lumpability constraint.
- Ordinary lumpability = constant row sum for the block
- Due to st-monotonicity, the maximal row sum is reached for the last row of the block at the end of the first step.
- During step 2, we modify the first column of the block to obtain constant row sum.
- Once a block has been computed, it is now possible to compute the block on the left, or below.

Example for the first block

- On matrix $P1$ formerly defined, with a partition into two macro-states: $(1, 2)$ and $(3, 4, 5)$.
- First step

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

- Second step

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

Complexity

- Space for LIMSUB : only two vectors of size n in memory
- Space for IMSUB : two linked lists in memory (smaller than n)
- Number of Operations :
  - difficult because we add and delete transitions
  - an upper bound can be computed before computing the matrix (based on the indices of the first and last elements in a row)
  - smaller than n^2
  - nf for a band matrix with f diagonals

An example with large state space

- Analysis of a buffer policy which combines PushOut mechanism for the space management and Round Robin service discipline.
- Two types of packets with distinct loss rate requirements.
- High priority: packets which have the smallest loss ratio requirements
- Pushout : a low priority packet which arrives in a full buffer is lost. If the buffer is not full, both types of packets are accepted. When the buffer is full, an arriving high priority packet pushes out of the buffer a low priority one if there is any in the buffer. Otherwise the high priority packet is lost.
- Poisson arrivals, Exponential service
- The buffer size :B, Number of states is in $O(B^2)$

- Representation of states: (T, H, RR) where T is the total number of packets, H is the number of high priority packets, and RR is the state of the
- The states are ordered according to a lexicographic non decreasing ordering.
- Computing the expected number of lost packets per slot. $(\Pi(B, B, L) + \Pi(B, B, H)$.
- If $B = 1000$: analysis a large chain (10^6 states).
- The reward is a non decreasing function which is zero everywhere except for the last two states where its value is one.

The key idea to shorten the state space is to avoid the states with large value of low priority packets.

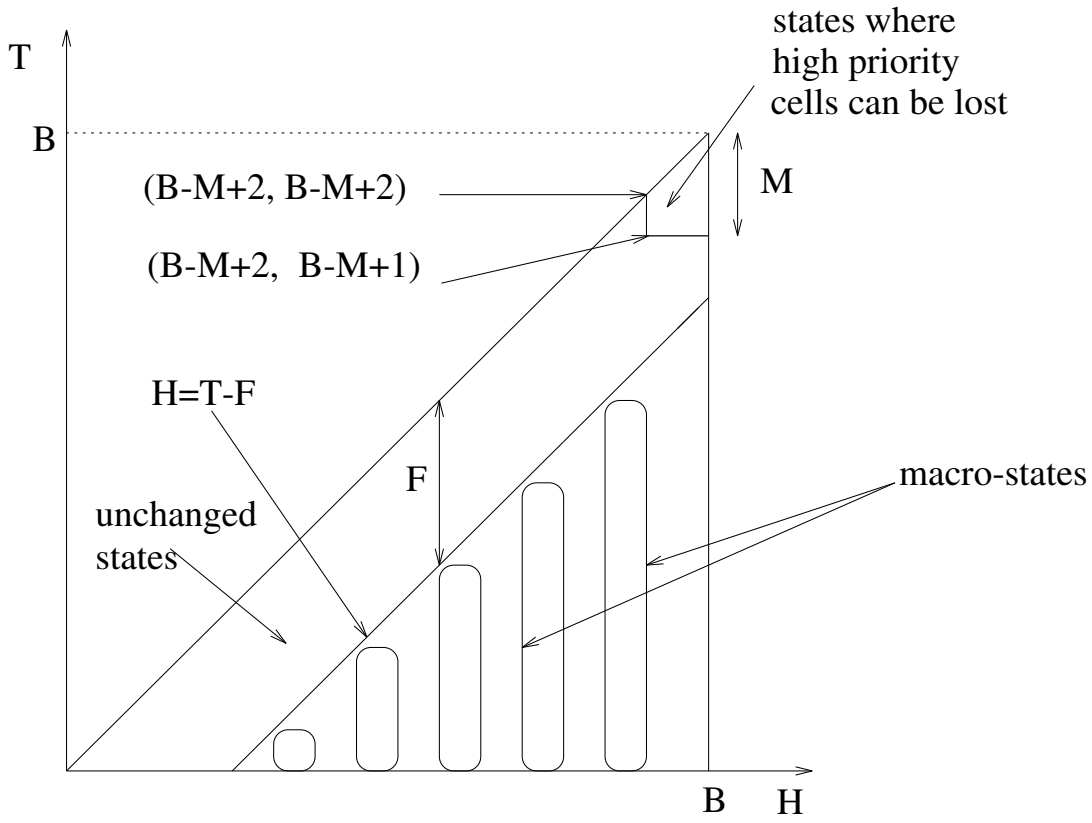

Figure 1: The aggregated chain

B	Size	NZT	Generate	Bound Size	Bound NZT	Reorder	LIMSUB
500	251502	1005003	3.3s	11872	79550	5.9s	38.5s
1000	1003002	4009995	13.5s	23850	148371	24s	165s

- The results look accurate (checked for small values for B). Why ?
- The distribution is skewed. Almost all the probability mass is concentrated on the states with a small number of packets.
- The first part of the initial matrix is already st-monotone (small number of perturbations).
- This is due to the ordering of the states we have considered.
- Small values and Small number of perturbations

Final Remarks

- Discrete Time MC
- For continuous time MC, uniformization
- Total ordering of the state space
- The rewards have to be a non decreasing function and consistent with the structure used (lumpability)
- The accuracy of the results may change with the ordering

Final Remarks

- Stochastic comparison of DTMC by algebraic arguments (not sample-path proofs) leads to Algorithms...
- Algorithms to reduce state-space and obtain upper bounds for almost arbitrary matrix
- Bounds for steady-state distribution and rewards
- Further research to extend the approach to transient rewards
- Easy to program with sparse matrix and a few vectors in memory (Tool available soon, presentation at MASCOTS03).