Bounds for Point and Steady-State Availability: an Algorithmic Approach Based on Lumpability and Stochastic Ordering

A. Bušić and J. M. Fourneau

PRiSM, University of Versailles, France

{abusic, jmf}@prism.uvsq.fr

Outline

- Motivation
- Introduction
- Stochastic comparison
- Computing transient bounds: LMSUB
- Avoiding state space generation: LL
- Numerical example
- Conclusion

Motivation

Highly available multicomponent systems:

- huge state space ($> 10^{10}$ states) $\rightarrow$ excludes the generation of the whole state space
- most of the probability mass located in a small portion of the state space

Our goal: to compute the bounds for *both transient and steady-state* dependability measures such as reliability, point and steady-state availability.

Method: algorithmic construction of a numerically tractable model that can be stochastically compared with the original one.

- No graph-structure conditions on the original model (unlike Courtois and Muntz)
- The same bounding model for both transient and steady-state analysis.

Introduction

- $Z = (Z_t)_{t \in [0,\infty)}$ a homogeneous CTMC on a finite state space
  $S = U \cup D$, $U \cap D = \emptyset \rightarrow$ UP and DOWN states
  - Reliability: $R(t) = Pr(Z_s \in U, \forall\ s\ \in\ [0,t))$
  - Point availability: $PAV(t) = Pr(Z_t \in U)$
  - (for Z having a steady state distribution π)
    Steady state availability: $A = \sum_{i|i \in U} \pi(i)$
- $(X_k)_{k \geq 0}$ - DTMC obtained from Z through uniformization
  with transition matrix $P = (Id - Q/\lambda)$, $Q =$ trans. rate matrix of Z

  P_U is the block of P associated to transitions between UP states

  π_0 is the initial distribution of Z (and of X)

Using uniformized chain,

$$R(t) = \sum_{k=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \pi_0 P_U^k 1 \geq \sum_{k=0}^{M} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \pi_0 P_U^k 1, \forall M$$

$$PAV(t) = \sum_{k=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \pi_0 P^k 1_U \geq \sum_{k=0}^{M} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \pi_0 P^k 1_U, \forall M$$

Lower bounds for $R(t)$ and $PAV(t)$ can be derived by using lower bounds for $\pi_0 P_U^k 1$ and $\pi_0 P^k 1_U \Rightarrow$ *DTMC bounding problem.*

- $\pi_0 P^k 1_U = P(X_k \in U)$
- Aggregating all DOWN states into one absorbing state $a \Rightarrow (\tilde{X}_k)_{k \geq 0}$
  Then, $\pi_0 P_U^k 1 = P(\tilde{X}_k \in U)$

Bounds for $P(X_k \in U)$ and $P(\tilde{X}_k \in U) \rightarrow$ *stochastic comparison.*

Stochastic comparison

Definition. (Strong stochastic order) Let S be a (partially) ordered space and X and Y be two random variables on S. Then,

$$X \preceq_{st} Y \text{ if } E[f(X)] \leq E[f(Y)],\ \forall f \text{ non decreasing function},$$

whenever the expectations exist.

Characterisation for $S = \{1, \ldots, n\}$ with $1 < 2 < \ldots < n$.
Let X and Y be r.v. on S with distribution vectors $(p_i)_{i=1}^n, (q_i)_{i=1}^n$. Then,

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

Example:

$$(0.4, 0.1, 0.2, 0.3) \preceq_{st} (0.4, 0.1, 0.1, 0.4)$$

$$(0.4, 0.1, 0.2, 0.3) \not\preceq_{st} (0.4, 0.2, 0, 0.4)$$

- If all the UP states precede all the DOWN states, then 1_D is a non-decreasing function
- For $(Y_k)_{k\geq 0}$ such that $X_k \preceq_{st} Y_k, \forall k$,

$$P(X_k \in U) = 1 - E[1_D(X_k)] \geq 1 - E[1_D(Y_k)] = P(Y_k \in U), \forall k$$

If X and Y have steady-state distributions π and π_Y and if $\pi \preceq_{st} \pi_Y$, then,

$$A = \sum_{i|i\in U} \pi(i) \geq \sum_{i|i\in U} \pi_Y(i).$$

Comparison of two homogeneous DTMCs on $S = \{1, \ldots, n\}$.

Three necessary conditions

- **Transition matrix comparison:** $P \preceq_{st} Q \iff P_{i,*} \preceq_{st} Q_{i,*}\ \forall i$.
- **Monotonicity.**
  Def. Q is $\preceq_{st}$-monotone iff for all u and v, $u \preceq_{st} v \Rightarrow uQ \preceq_{st} vQ$.
  Characterization:

$$Q \text{ is } \preceq_{st}\text{-monotone} \iff Q_{i,*} \preceq_{st} Q_{i+1,*},\ \forall i < n.$$

- $X(0) \preceq_{st} Y(0)$

Then $X(k) \preceq_{st} Y(k), \forall k$. If the steady-state distributions π_X and π_Y exist, then

$$\pi_X \preceq_{st} \pi_Y .$$

Construction of an upper bounding DTMC for a given DTMC X
(with P as transition matrix)
Compute Q such that $P \preceq_{st} Q$ (1) and Q is $\preceq_{st}$-monotone (2)

$$\sum_{k=j}^{n} Q_{i,k} \geq \sum_{k=j}^{n} P_{i,k},\ \forall i, j, \tag{1}$$

$$\sum_{k=j}^{n} Q_{i,k} \geq \sum_{k=j}^{n} Q_{i-1,k},\ \forall j,\ \forall i \geq 2. \tag{2}$$

Additionally, Q is forced to be ordinary lumpable (state-space size reduction):
For a given partition $C_l, l \in L$ of the state space,

$$\forall i \in L, \forall a, b \in C_i, \sum_{k \in C_j}^{n} Q_{a,k} = \sum_{k \in C_j}^{n} Q_{b,k},\ \forall j \in L. \tag{3}$$

LIMSUB algorithm, Fourneau et al. 2003: additional irreducibility constraints $\rightarrow$ steady-state bounds.

Computing transient bounds: LMSUB

Theorem

- X a DTMC on S finite and $r()$ non-decreasing rewards on S.
  Partition of S with the states of the same macro-state contiguous.
- Let Y be DTMC such that: $X_0 \preceq_{st} Y_0$, $P \preceq_{st} Q$, Q is $\preceq_{st}$-monotone and lumpable (with respect to the given partition).
  Let Z be the lumped version of Y.
- $s(p) := max_{i \in C_p} r(i)$ (rewards on macro-states)
  (Remark. Do not group in a same macro-state UP and DOWN states!)
- Then,
  - For any instant t, $E_X(r)(t) \leq E_Z(s)(t)$.
  - For the steady-state, $E_X(r) \leq E_Z(s)$.

Remark. Transition matrix of Z is considerably smaller.

LMSUB algorithm (**L**umpable **M**onotone Strong **S**tochastic **U**pper **B**ound) (based on LIMSUB)

- Inspired by transient analysis: no irreducibility constraints
  $\Rightarrow$ can be used to compute bounds even of reducible matrices
- No supplementary movement of probability mass from lower to upper states (coming from transition keeping induced by irreducibility constraints)
- Simpler: based only on transition matrix comparison, monotonicity and lumpability
- Can be used also for steady-state bounds (the bounding chain has only one recurrent class)

Column by column computation (decreasingly in columns and increasingly in lines) in two steps:

- Verifying the comparison and monotonicity constrains for the current column

$$\begin{bmatrix} 0.2 & 0.2 & 0.1\ (0.6) & 0.3\ (0.5) & 0.2 \\ 0.1 & 0.2 & 0.1\ (0.7) & 0.5\ (0.6) & 0.1 \\ 0.0 & 0.2 & 0.6\ (0.8) & 0.1\ (0.2) & 0.1 \\ 0.1 & 0.2 & 0.4\ (0.7) & 0.3\ (0.3) & 0.0 \\ 0.0 & 0.1 & 0.0\ (0.9) & 0.9\ (0.9) & 0.0 \end{bmatrix} \quad \left[\begin{array}{c|ccc} & 0.1\ (0.6) & 0.3\ (0.5) & 0.2 \\ & 0.1\ (0.7) & 0.4\ (0.6) & 0.2 \\ \hline & 0.2\ (0.8) & 0.4\ (0.6) & 0.2 \\ & 0.2\ (0.8) & 0.4\ (0.6) & 0.2 \\ & 0.0\ (0.9) & 0.7\ (0.9) & 0.2 \end{array}\right]$$

- For the first column of each block only: lumpability constraints

$$\left[\begin{array}{c|ccc} & \boxed{0.2}\ (0.7) & 0.3\ (0.5) & 0.2 \\ & 0.1\ (0.7) & 0.4\ (0.6) & 0.2 \\ \hline & \boxed{0.3}\ (0.9) & 0.4\ (0.6) & 0.2 \\ & \boxed{0.3}\ (0.9) & 0.4\ (0.6) & 0.2 \\ & 0.0\ (0.9) & 0.7\ (0.9) & 0.2 \end{array}\right] \quad \begin{bmatrix} 0.3 & 0.7 \\ 0.1 & 0.9 \end{bmatrix}$$

Avoiding computations

- Only the lumped matrix is actually computed and stored.
- Second step computations can be avoided.

  Monotonicity $\Rightarrow$ the maximal value of $\sum_{k=j}^{n} Q_{i,k}$ is obtained for the last row of a block (known already after the first step).
  Only store one value per block.

LMSUB algorithm

for $b = m$ **downto** 1 **do**
  for $j = last(b)$ **downto** $first(b)$ **do**
    for $a = 1$ **to** m **do**
      for $i = first(a)$ **to** $last(a)$ **do**
        $sumQ_{i,j} = \max(sumQ_{i-1,j}, \sum_{k=j}^{n} P_{i,k})$;
        if $b < m$ ***and*** $j = last(b)$ ***and*** $i = first(a)$ **then**
        $sumQ_{i,j} = \max(R_{a,b+1}, sumQ_{i,j})$;
      end
    end
  end
  for $a = 1$ **to** m **do** $R_{a,b} = sumQ_{last(a),first(b)}$;
end

Sparse matrix implementation

- The repairable system models - huge state space but relatively few transitions per state $\rightarrow$ sparse matrix representation
- Adapted data structures to reduce further the computations
  2 vectors:
  1. for partial sums of current column of P
  2. for partial sums of previous/current column of Q
     Only one structure (elements computed in increasing order)

- For vector $sumQ_{*,j}$ ($sumQ_{i,j} := \sum_{k=j}^{n} Q_{i,k}$) we store only the index and the value for the rows this sum strictly increases (monotonicity $\Rightarrow$ increasing)
- Only the elements of $sumQ_{*,j}$ whose value is different from $sumQ_{*,j+1}$ are actually computed.

  Those changes are due to the non-zero elements in the column j in P ($\preceq_{st}$-comparison):
  Between the two non-zero elements of P it is only necessary to update the list containing the information on $sumQ_{*,j+1}$ vector (erase some elements if they are smaller or equal to the last computed element).

Avoiding state space generation: LL

To apply LMSUB algorithm: the transition matrix must be stored on disk. Multicomponent systems - state-space often huge ($> 10^{10}$ states) - generation of the whole state-space excluded

LL (Lumpable and Larger) algorithm

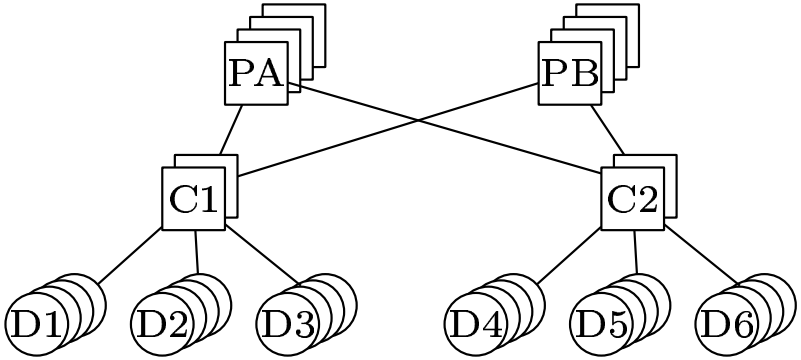
We design the matrix A, lumpable and $P \preceq_{st} A$ and we generate and store the lumped version (B) of A (uniformization performed at this step)
B (not $\preceq_{st}$-monotone) $\rightarrow$ input for LMSUB - two step aggregation

The bound obtained by LMSUB on B is also a bound of the original matrix we are not able to store.

Aggregation rules for the first (LL) step (multicomponent system analysis):

- Do not aggregate the UP states (concentration of the probability mass)
- Aggregate the DOWN states with the same total number of faults.

Numerical example



- Muntz et al. 1989, Carrasco 1999
- UP states:
  - at least one processor unit (PA or PB)
  - at least one controller of each set (C1 and C2)
  - at least three disks of each cluster (D1 and D2 and . . . and D6)

  are operational
- Failures and reparations of components are exponentially distributed

- Only one processor of each type is active and only the active processor can fail;
  A failure of PA contaminates the active PB with probability 0.1
- Two failure modes (soft and hard) - occur with equal probability
- One repairman that chooses at random (uniformly) the failed component

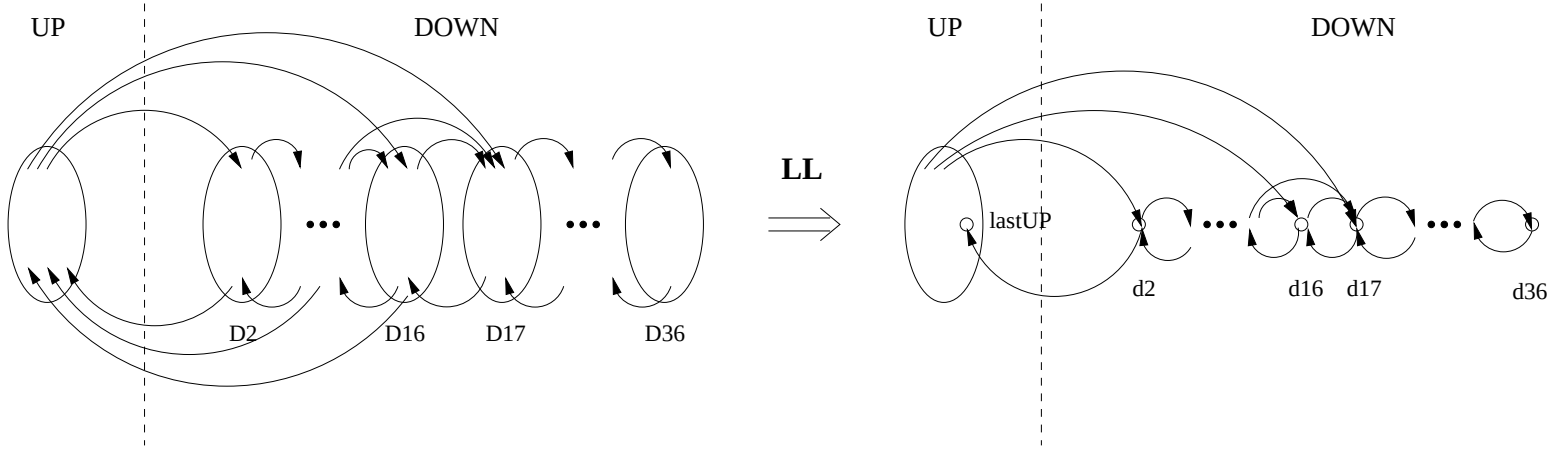
Failure rates	
PA, PB, C1	1/2000
C2	1/4000
D1	1/6000
D2	1/8000
D3	1/10000
D4	1/12000
D5	1/14000
D6	1/16000

Repair rates	soft	hard
system UP	$0.1h^{-1}$	$0.05h^{-1}$
system DOWN	$1h^{-1}$	$0.5h^{-1}$

Only 36 components of 10 types yet the state-space is of order of 9.0 10^{10}.

Availability bounds

- LL step: 1 312 235 states (1 312 200 UP) and 25 754 089 transitions.

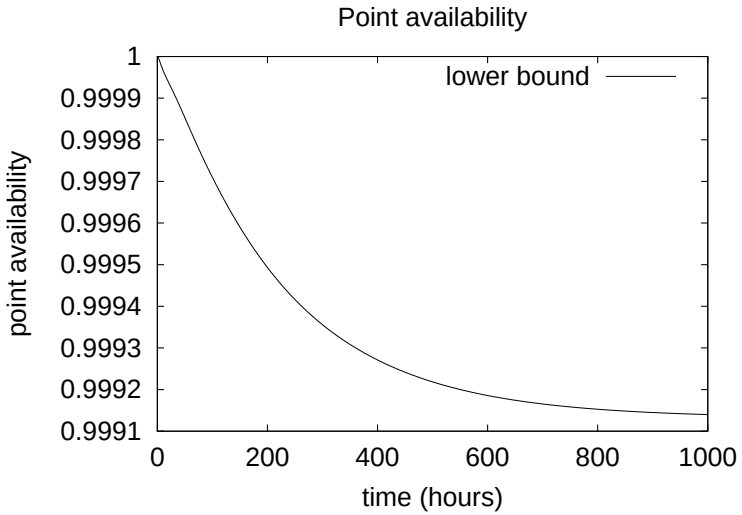


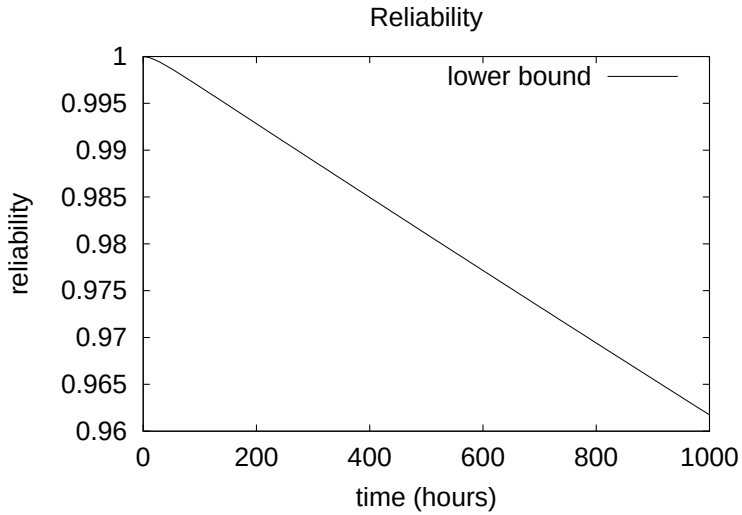
$last_{UP}$: UP state with 15 failures - all hard; only operational processor is PB

- LMSUB step: 172 subsets
  UP states (except $last_{UP}$) are reordered (increasingly) and regrouped by number of failed components followed by number of hard failures; followed by $last_{UP}$ state, then by 35 aggregated DOWN states (cannot be reordered: to remain $\preceq_{st}$ larger)

Reliability bounds: we are only interested in UP states; all the DOWN states aggregated into one absorbing state.

UP states are ordered and regrouped according to the number of failed components followed by the number of hard failures
$\rightarrow$ partition into 137 blocks (1 DOWN)





The lower bound for steady-state availability: 0.999132158

Conclusion

- We have extended the theory of algorithmic stochastic bounds to reliability and availability problems.
  Both transient and steady-state bounds can be computed.
  Markov chains may be reducible.
- More efficient algorithms:
  On presented example our implementation of LMSUB algorithm is approximately twice faster than the previous implementation of LIMSUB (Fourneau et al. 2003): improvement made during the normalization part of the algorithm and the better utilization of the matrix sparse structure.
- High level formalisms may be used to build lumpable matrices which are larger in the stochastic sense.

LL description

Assumption: description of the model is based on events with associated rates (do not depend on states)

Not necessary, other formalisms can be used as well:
LL based on Stochastic Automata Network is currently under development.

Transition rates within aggregated chain:

- Transition rate from a simple state to a macro-state C_p (simple or aggregated): sum of the transition rates from x to y, for all y in C_p.

- For transitions leaving an aggregated state C_p to a macro-state C_q:
  1. label all transitions with the events,
  2. group the transitions and the events according to the number of failures (example: "+1" transition models a new fault, "-2" two new reparations),
  3. if an event is associated to two (or more) numbers of failures: all the transitions labelled with this event must now join the largest state reached by this event from a state in macro-state C_p.
  4. Then do the summation for all the states in C_q.

CPU times in seconds on a PC (CPU 3.20GHz, 1 GB of RAM)

generation	reordering	LMSUB	LIMSUB
182.0	95.52	56.54	107.3