Evaluation of reward analysis methods with MRMSolve 2.0

Gábor Horváth, Sándor Rácz, Árpád Tari, Miklós Telek
Technical University of Budapest, 1521 Budapest, Hungary
e-mail: {hgabor,raczs,arpi,telek}@webspn.hit.bme.hu

Abstract

The paper presents the 2.0 release of MRMSolve, a Markov reward model (MRM) analysis tool. This new release integrates recent research results, such as MRMs with partial reward loss, second order MRMs and their combination, with old and widely known reward analysis methods, such as the ones by DeSouza-Gail, Nabli-Sericola and Donatiello-Grassi.

The paper compares the mentioned direct distribution analysis methods with each other and with the moments based reward estimation methods.
Keywords: Markov reward models, Randomization, Numerical analysis.

1. Introduction

The introduction of SRMs initiated a fertile research especially in performance analysis of computer and communication systems through the 80's and the 90's. A number of numerical methods were developed and various modeling formalisms were introduced to make an effective use of SRMs. See [8] for a recent survey of these results.

For an effective use of reward modeling and analysis a series of software tools were developed. The diverse set of reward analysis tools indicates that the concept of reward function can be associated with a wide range of modeling formalisms describing the system behavious, e.g., Tangram-II uses the queueing system formalism [6], SHARPE supports fault tree, reliability block diagram, queueing network [19], Möbius allows high level modeling formalisms including stochastic process algebras [4] and there are other stochastic Petri net based tools, such as SPNP [9] and UltraSAN [20].

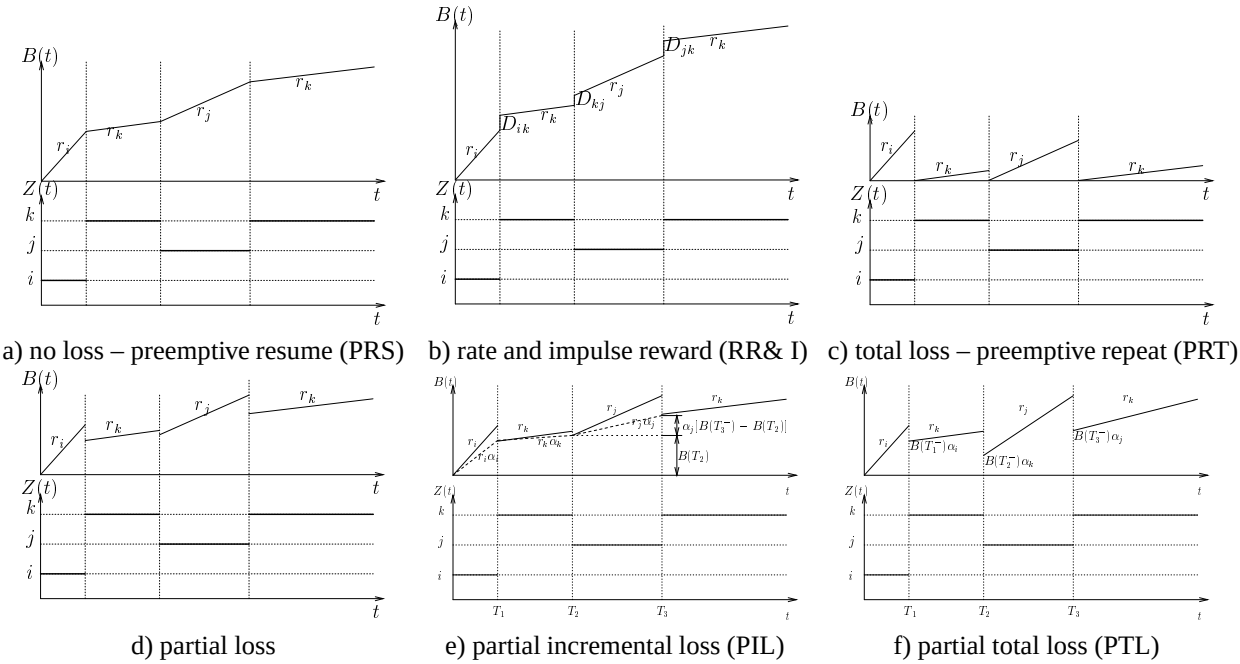
The majority of these tools focus on the analysis of the given modeling formalism and the reward analysis part is treated as an additive modeling and analysis feature. By this reason the reward analysis functions of these tools are restricted to simple reward measures, like mean transient and stationary reward, which are related to the transient and stationary distribution of the structure state process in a straight forward way. More complex reward functions (e.g., second order) and reward measures (e.g., higher moment) are not calculated by these tools. In contrast, MRMSolve focuses on more sophisticated reward analysis. Its modeling formalism is designed for easy definition of potentially large reward models and it integrates (to the best of our knowledge) the largest set of possible reward functions and reward analysis methods. The availability of this wide set of reward analysis methods allows us to compare the experimental properties of these methods.

The rest of this paper is organized as follows. After the introduction of stochastic reward models section 3 summarizes the analysis methods of Markov reward models. Section 4 presents the new architecture of MRMSolve. The Markov reward model of a storage system with probabilistic error checking is introduced in section 5 and analyzed with the built in solvers of MRMSolve in section 6. The paper is concluded in section 7.

2. Classification of stochastic reward models

SRMs can be classified based on the following basic properties.

Stochastic process: The stochastic behaviour of the structure-state process, $Z(t)$, has a significant importance in SRM analysis. $Z(t)$ $(t \geq 0)$ is a stochastic process defined over a discrete and finite state space Ω of cardinality M. Due to their analytical tractability, SRMs with underlying (time homogeneous) continuous time Markov chains (CTMC) gain the most attention [5, 7, 13, 23]. SRM with underlying semi-Markov provess (SMP) [12] or Markov regenerative process (MRP) [22] still allows a compact analytical description, but in Laplace transform domain. SRMs with underlying CTMCs are referred to Markov reward models. MRMSolve 2.0 and this paper is restricted to various classes of MRM only. The analysis of MRMs with underlying (time) inhomogeneous continuous time Markov chains (IH-CTMC) requires the application of a different analytical treatment, the use of forward differential equations [21].



a) no loss – preemptive resume (PRS) b) rate and impulse reward (RR& I) c) total loss – preemptive repeat (PRT)

d) partial loss e) partial incremental loss (PIL) f) partial total loss (PTL)

Figure 1. Change of accumulated reward at state transitions

Reward accumulation: The two considered ways of reward accumulation are the time proportional and the immediate reward accumulation. The time proportional reward is accumulated during a sojourn in a given state. It is deterministic in first order models (fig. 1.a) and normal distributed in second order (SO) models (fig. 2). Traditionally the first one is referred to as rate reward (RR) accumulation. Additional to the time proportional reward accumulation immediate, so called, impulse (I) reward accumulation is possible at state transition instances (fig. 1.b).

Reward loss: Reward functions might allow instantaneous reward loss as well. Traditionally reward functions with no loss (preemptive resume - PRS, fig. 1.a) and with total reward loss (preemptive repeat - PRT, (fig. 1.c)) were applied [2]. Partial loss reward models were introduced in [1, 14] (fig. 1.d). In partial increment loss (PIL) models the amount of lost reward is proportional to the reward accumulated during the sojourn in the last state (fig. 1.e), and in partial total loss (PTL) models the amount of lost reward is proportional to the overall reward (fig. 1.f), The existence of different reward loss policies in a single model increases the modeling power, but it also increases the complexity of the model description and analysis. MRMSolve is restricted to SRMs with unique reward loss policy, but a wide set of possible policies are considered.

Evaluated measure: The analysis of SRMs means indeed two analysis problems: the evaluation of the distribution of the accumulated reward (AR) and of the completion time (CT). The **accumulated reward**, $B(t)$, is a random variable which represents the accumulation of reward in time. The distribution of the accumulated reward is $Pr(B(t) < w)$. The **completion time** $C(w)$ is a random variable representing the time to accumulate w amount of reward. $C(w) = \min[t \geq 0 : B(t) \geq w]$ Both problems are considered in MRMSolve 2.0. Monotone increasing $B(t)$ functions provides a nice relation of these measures.

$$Pr\{B(t) \leq w\} = Pr\{C(w) \geq t\} \qquad (1)$$

In case of non-monotone reward accumulation there are qualitative differences in the analysis of accumulated reward and completion time measures (e.g., the distribution of accumulated reward exhibit a closed form transform domain expression while the distribution of the completion time does not).

In applied performance analysis the required random performance parameter (AR or CT) can be characterized by its distribution function (d) or by its moments (m) depending on the particular application. The availability of moments based distribution estimation methods allow to bound the distribution when only the moments are known.

A different set of methods are available for the direct analysis of the distribution and of the moments of reward measures. An essential goal of this paper is to compare the properties of these approaches. Note that

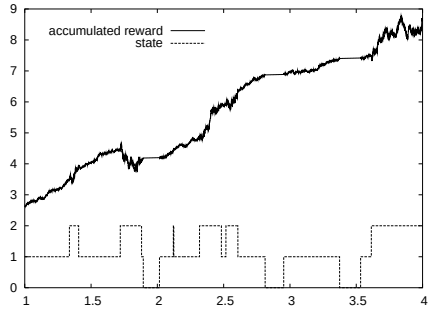


Figure 2. A sample realization of a second order (SO) MRM

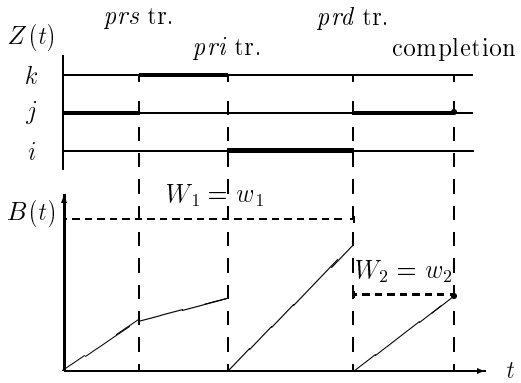


Figure 3. Resampling the reward bound at state transitions

(1) does not relate the moments of the accumulated reward with moments of the completion time.

Resampling of work requirement: In case of completion time analysis of a reward model with total reward loss and random work requirement we have two cases. If the same task has to be completed after a state transition with reward loss the work requirement remains the same (preemptive repeat identical - PRI, second state transition in fig. 3), and if a new task is considered after the state transition the work requirement is resampled (preemptive repeat different - PRD, third state transition in fig. 3).

Table 1 summarizes the possible combination of the mentioned cases, the availability of analysis results in the literature and the methods implemented in MRMSolve 2.0. The model classes are identified with the abbreviation introduced in this section. Letter d and m refers to the distribution and moments of performance measure, respectively. A capital letter (D,M) means that the procedure is implemented in MRMSolve 2.0 based on the given reference(s). If the solution of the class is not available in MRMSolve 2.0 we distinguish the following cases. $\emptyset$ denotes that the class is meaningless (apart from very special application). Empty blocks indicate that we are not aware of any result for the given classes. Letter d[$\bullet$] (m[$\bullet$]) means that there is known solution method for the distribution (moments) of the given class in the associated reference. The superscript * indicate that this solution is not efficient, i.e., it cannot be applied for models with more than 100 states.

3. Reward analysis methods

3.1. Steady-state analysis of Markov reward models

First of all we restrict our attention to time homogeneous MRMs. With this restriction we need to distinguish two main cases: $\mathcal{B}(t)$ tends to infinity with a positive probability, or not. In the first case $E(\mathcal{B}(t)/t)$ tends to a constant

$$\lim_{t\to\infty} E(\mathcal{B}(t)/t) = \underline{P}_{SS}\,(\mathbf{R} + \mathbf{Q} \odot E(D))\,\mathbb{1}\ ,$$

where $\underline{P}_{SS}$ is the limiting distribution of the underlying *CTMC* with generator $\mathbf{Q}$ (which can be reducible or irreducible), $\mathbf{R}$ is the diagonal matrix of the reward rates, the $E(D)$ matrix contains the mean of the impulse reward associated with the state transitions, $\odot$ denotes the elementwise matrix multiplication ($[\mathbf{A} \odot \mathbf{B}]_{ij} = a_{ij} \cdot b_{ij}$) and $\mathbb{1}$ is the column vector of ones.

If $\mathcal{B}(t)$ is finite with probability 1 we have the following main cases:

- There is PRS or PIL reward accumulation, the underlying CTMC is reducible and the reward rate, the variance and the impulse reward associated with the absorbing sets are zero. In this case the transient analysis for sufficiently large t results the limiting reward distribution.
- There is PRT or PTL reward accumulation and the underlying CTMC is reducible. In this case the stationary behaviour can be obtained using the method presented in [1].

The solution of the combination of these cases requires a combination of the mentioned solution methods.

3.2. Transient analysis of Markov reward models

The transient distribution of reward measures without reward loss can be described with a set of partial differential equations, e.g., the transient behavior of

		CTMC				IH-CTMC			
		RR	RR&I	SO	SO&I	RR	RR&I	SO	SO&I
AR	PRS	**D**[5, 7, 13] **M**[23]	d[5] **M**[15]	**M**[10]		m[21]	m[21]		
	PRT	**M**[11]	∅		∅		∅		∅
	PTL	d*[1]	∅		∅		∅		∅
	PIL	d*[1] **M**[11]	∅		∅		∅		∅
CT	PRS	**D**[5, 7, 13] **M**[23]	d[5] m[16]			d*[21] m[21]	d*[21] m[21]		

Table 1. *Classification of Markov reward models*

the accumulated reward fulfills the following partial differential equation

$$\frac{\partial \underline{B}(t,w)}{\partial t} + \mathbf{R}\,\frac{\partial \underline{B}(t,w)}{\partial w} - \frac{1}{2}\,\mathbf{S}\,\frac{\partial^2 \underline{B}(t,w)}{\partial w^2} = \mathbf{Q} \odot \mathbf{D}(w) \cdot \underline{B}(t,w), \quad (2)$$

where the $\underline{B}(t,w)$ is composed by the elements $B_i(t,w) = Pr(B(t) < w \mid B(0) = 0, Z(0) = i)$, the i,j element of matrix $\mathbf{D}(w)$ is the distribution of the impulse reward earned at a state transition from state i to state j and $B_i(0,w) = U(w)$, where $U(\cdot)$ is the unit step function.

The double Laplace-Stieltjes transform of this partial differential equation has the following form.

$$\underline{B}^{\sim\sim}(s,v) = s\left(s\mathbf{I} + v\mathbf{R} - \frac{v^2}{2}\,\mathbf{S} - \mathbf{Q}\odot\mathbf{D}^{\sim}(v)\right)^{-1}\cdot\underline{h}\,.$$

The lack of the explicit solution of (2) initiated a research activity for efficient numerical analysis of $\underline{B}(t,w)$ and the moments of the reward measures $\int_w w^n d\underline{B}(t,w)$. It seems that the most efficient methods are based on randomization. Table 2 compares some randomization based MRM analysis algorithms with respect to their computational complexity and memory requirements. The complexity of the *CTMC* transient analysis is used as a reference point. In the table t refers to the time point of the analysis, M is the cardinality of the state space, T is the number of state transitions, K is the number of different reward rates and n is the number of computed moments. The $\mathcal{O}(\cdot)$ expressions have to be handled with care. These expressions do not say anything about the real complexity of the methods, since the $\mathcal{O}(\cdot)$ expressions explain only the tendencies, but not the particular values. The experimental analysis in section 6 indicates how complex behaviour might occur behind an $\mathcal{O}(\cdot)$ expression. The following subsections summarizes the methods of Table 2.

Randomization based reward analysis The common root of randomization based reward analysis is to decompose the behaviour of the underlying CTMC into a discrete time Markov chain (DTMC) with transition matrix $\mathbf{P} = \mathbf{Q}/q + \mathbf{I}$ and a Poisson process with parameter q. Based on this decomposition

$$Pr(\text{reward measure} < x) = \sum_{k=0}^{\infty} Pr(\text{reward measure} < x \mid \text{k steps of the DTMC}) \cdot Pr(\text{k steps of the DTMC}),$$

where $Pr(\text{k steps of the DTMC}) = \frac{(qt)^k}{k!}e^{-qt}$. The various analysis methods differ in the evaluation of the conditional probability $Pr(\text{reward measure} < x \mid \text{k steps of the DTMC})$ or the conditional moments $E(\text{reward measure}^n \mid \text{k steps of the DTMC})$, in case of moments analysis.

Method of Donatiello and Grassi The algorithm presented in [7] proposes to calculate the conditional probability $B_i(t, w \mid k)$ in the following form:

$$B_i(t,w \mid k) = \alpha_i^{(k)} u[w - r_i t] + \sum_{h=1}^{k}\sum_{j=1}^{M} \binom{k}{j-1} \beta_i^{(k)}(j,h) \left(\frac{w - r_j t}{t}\right)^{k-h+1} u[w - r_j t]$$

where coefficients $\alpha_i^{(k)} \in \mathbb{R}$ and $\beta_i^{(k)}(j,h) \in \mathbb{R}$ are independent of t and w. [7] presents recursive expressions for the calculation of those coefficients. These recursive expressions contain numerically sensitive subtractions.

Method of Nabli and Sericola The main theorem of [13] has the following form

$$B_i(t,w) = 1 - \left[\sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \sum_{k=0}^{n}\sum_{j=1}^{K} \binom{n}{k} s_j^k\,(1-s_j)^{n-k}\; b^{(j)}(n,k)\mathbb{I}_{\{r_{j-1}t \le w < r_j t\}}\right],$$

where $\mathbb{I}_{\{\bullet\}}$ is the indicator, $s_j = \dfrac{w - r_{j-1}t}{(r_j - r_{j-1})t}$ and the coefficient $b^{(j)}(n,k)$ is given by an explicit recursive expression. $b^{(j)}(n,k)$ is independent of t and w. The states with the same reward rate are collected into reward classes. The reward rates associated with the classes are $r_0 < r_1 < \ldots < r_K$. Due to the

Method	CPU time	memory	output	class
Donatiello and Grassi [7]	$\mathcal{O}(T \cdot K \cdot t^2)$	$\mathcal{O}(K \cdot M \cdot t)$	distr.	RR, PRS
Nabli and Sericola [13]	$\mathcal{O}(T \cdot K \cdot t^2)$	$\mathcal{O}(K \cdot M \cdot t)$	distr.	RR, PRS
DeSouza and Gail [5]	$\mathcal{O}(T \cdot t^2)$	$\mathcal{O}(M \cdot t^2)$	distr.	RR, PRS
Rácz and Telek [15, 23]	$\mathcal{O}(T \cdot n \cdot t)$	$\mathcal{O}(M \cdot n)$	moments	RR&I, PRS
Horváth, Rácz, Telek [10]	$\mathcal{O}(T \cdot n \cdot t)$	$\mathcal{O}(M \cdot n)$	moments	SO, PRS
Horváth and Telek [11]	$\mathcal{O}(T \cdot n \cdot t)$	$\mathcal{O}(M \cdot n)$	moments	RR, PIL
CTMC transient analysis	$\mathcal{O}(T \cdot t)$	$\mathcal{O}(M)$		

Table 2. *Complexity of randomization based numerical analysis methods of MRMs*

$r_{j-1}t \leq w < r_j t$ condition the considered s_j coefficient is $0 \leq s_j \leq 1$ and $0 \leq b^{(j)}(n,k) \leq 1$ as well. The convex combination of $0 \leq b^{(j)}(n,k) \leq 1$ terms insures the numerical stability of the method.

Method of De Souza e Silva and Gail The algorithm proposed in [5] computes the distribution of the accumulated reward of *MRM*s with rate and impulse reward. The procedure calculates the distribution of the normalized reward $ACIR(t) = \mathcal{B}(t)/t$ using the same reward classes.

The final form of the algorithm is

$$Pr(ACIR(t) > w) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \sum_{i:r_i > r} \| \Upsilon[i,n,n] \|,$$

where $\Upsilon[i,n,n] \in \mathbb{R}$ is calculated from an explicit recursion. $\Upsilon[i,n,n] \in \mathbb{R}$ is a function of w, hence it has to be recalculated as many times as many w points are required. The $\Upsilon[i,n,n]$ terms are unbounded, and the recursive formula contains numerically sensitive subtractions.

Moments analysis methods The moments analysis methods in [23, 15, 10] has the form

$$\underline{m}^{(n)}(t) = \sum_{k=0}^{\infty} \frac{(qt)^k}{k!} e^{-qt}\, \underline{U}^{(n)}(k)\ ,$$

where $\underline{U}^{(n)}(k) \geq 0$ is given by explicit recursive formula. The complexity of this formula depends on the particular MRM. It is n times more expensive in case of second order and/or impulse reward MRMs. The recursive formula contains only multiplication and summation of positive numbers.

4. Basic features of the MRMSolve 2.0 software tool

MRMSolve 2.0 behaves like a framework; it has tools to edit, check and visualize models, to investigate the effect of model parameters on the distribution and on the moments of the accumulated reward or completion time. Behind the user interface different solver algorithms for different types of reward models are integrated and further solution methods can be added easily.

The graphical interface is written in Java (see a screenshot on Figure 4), thus it is platform independent by its nature. The implemented solver algorithms are written in c++ for efficiency reasons. We used the GNU g++ compiler, which exists under a large number of platforms, therefore also the solver algorithms can be compiled on many platforms without any changes.

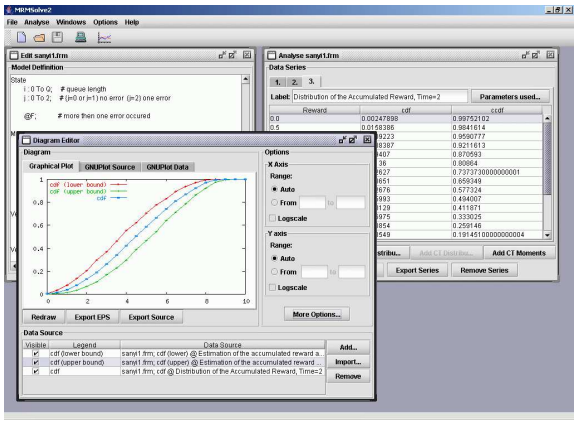


Figure 4. MRMSolve 2.0 screenshot

MRMSolve 2.0 contains the following solvers:

- Preemptive resume MRM. This solver supports the computation of the moments of the accumulated reward and completion time [23], and it can compute the distribution of the accumulated reward using the mentioned 3 methods.
- Second Order MRM with partial increment loss (combination of [10] and [11]). Only the computation of moments of the accumulated reward is supported.

Describing Models There are two ways to describe a model in MRMSolve 2.0. The simple – less flexible – way is called *raw* description. In this case the user has to fill in all the (non zero) elements of the vectors and matrices describing the model. If the system consists of few states only, this is a fast way to describe the model.

The second method for model description uses a compact and flexible *formalism* for easy description of very large (but well structured) systems. For the details of this model description laguage we refer the reader to [17], and we only provide a short summary on the most important features here. Figure 5 shows the model description of the example presented in section 5. This description consists of sections. A convenient property of this syntax is that the model constants are enumerated in section 'Const', so a parameter can be easily changed without redefining all the matrices and vectors. State space variables can be found in section 'States'. With more than one state space variables the user can construct models with multi dimensional structure. The generator matrix and the vectors corresponding to the model are described by rules in sections 'Vector' and 'Matrix', reflecting the structure of the system. A rule basically assigns an expression to a state identifier (or to a transition identifier, in case of a matrix). Before each rule, there is an optional condition that defines the scope of the rule. During the model construction phase the software generates all the possible states, and fills in the vectors and matrices according to the rules with enabled condition.

The tool supports the user in checking the constructed model. Selecting the 'Show State Space' menu item in the 'Analysis' menu the tool shows the number of states, the number of transitions, and enumerates the labels of the states. A new feature of the tool visualizes the state space of the generated model. The 'Draw State Space' menu item displays the state space graph graphically (using the tools of the Graph Visualization Project (of the ATT Reaseach Lab) [18]).

Model Analysis Selecting the 'Analyse' menu (or the corresponding button on the toolbar) opens the analysis window for the model. In this window the user can call the solvers to get different performance measures of the system. The overall set of performance measures are the distribution of the accumulated reward, the moments of the accumulated reward, the distribution of the completion time and the moments of the completion time. Depending on the type of the considered MRM some of these performance measures are not available. For example, in case of second order MRMs only the 'AR moments' is supported.

```
# Model proposed in [3]
type prs;

Const
      lambda = 3.0;    # rate at which access operations arrive
      mu= 5.0;         # access operation service rate %the system
      sigma= 5.0;      # execution rate of errorchecking/recovery
     gamma= 5.0e-7; # error rate, either software or hardware-induced
      q= 0.2;          # probability of error checking per access,  q in [0,1]
      Q=15;            # queue length
State
      i : 0 To Q;     # queue length
      j : 0 To 2;     # j=0 or j=1 no error  j=2 one error

      @F;             # more then one error occurred
Matrix Q
            [i,j]->[i+1,j]   = lambda;
      j==0 : [i,j]->[i-1,j]   = (1-q)*mu;
      j==0 : [i,j]->[i,j+1]   = q*mu;
      j==0 : [i,j]->[i,j+2]   = gamma;
      j==1 : [i,j]->[i-1,j-1] = sigma;
      j==2 : [i,j]->[i,j-1]   = q*mu;
      j==2 : [i,j]->[@F]      = gamma+(1-q)*mu;
Vector R
      i>0 & j==0 : [i,j] = (1-q)*mu;
      i>0 & j==1 : [i,j] = sigma;
Vector P0
      i==0 & j==0 : [i,j] = 1;      # starting state
```

Figure 5. Source code of example 1

If the moment type measures have been selected (AR or CT moments), series of runs can be generated automatically, where the moments are computed as a function of a model parameter appearing in the 'Const' section of the model description. With the 'Approx Distribution' button the distribution estimator starts and it calculates the lower and upper bounds of the distribution of the reward measure based on the evaluated moments.

Visualizing the Results The 'Diagram Editor' menu starts the diagram editor tool. With this tool it is possible to draw and compare all the available results (already computed in the 'Analyse' window) of all the open models. The tool calls 'gnuplot' to create the required plots. The user can set and change the most important plot options, like the range of the x and y axis, the style of lines, the linear and logarithmic scaling of the axes, or the position of the legends. The graphical results can be exported to encapsulated postscript files.

5. Example of a storage system

In [3] the authors analyze a storage system with probabilistic error checking procedures. They consider the most commonly used error correction type: the 1-correctable.

The storage system can process only one request at a time. The storage system behaves as an atomic en-

tity so that a request is either fully serviced or not serviced at all. Each access operation may be followed by an error checking operation which occurs with probability q. If the data accessed by this operation contains an error, it will be detected and corrected by the 1-correctable error checking/recovery code and the operation is still supplied with correct information.

The behavior of the storage system depends on two quantities, the queue length and the current error level. The queue length describes the queueing behavior of the system while the error level determines whether the system is functionally correct or not. In a functionally correct state the system serves one of the arrived access requests with rate $(1-q)\mu$ and in an error checking state with rate σ. This reward model allows to calculate the service throughput, i.e., the number of operations that can be served within a time limit. Figure 5 presents the description of the model in MRMSolve input language and Figure 6 presents its state space.

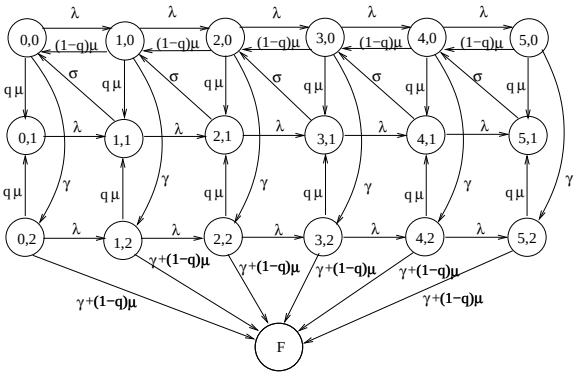


Figure 6. State graph of the storage system with $Q = 5$

6. Numerical experiences

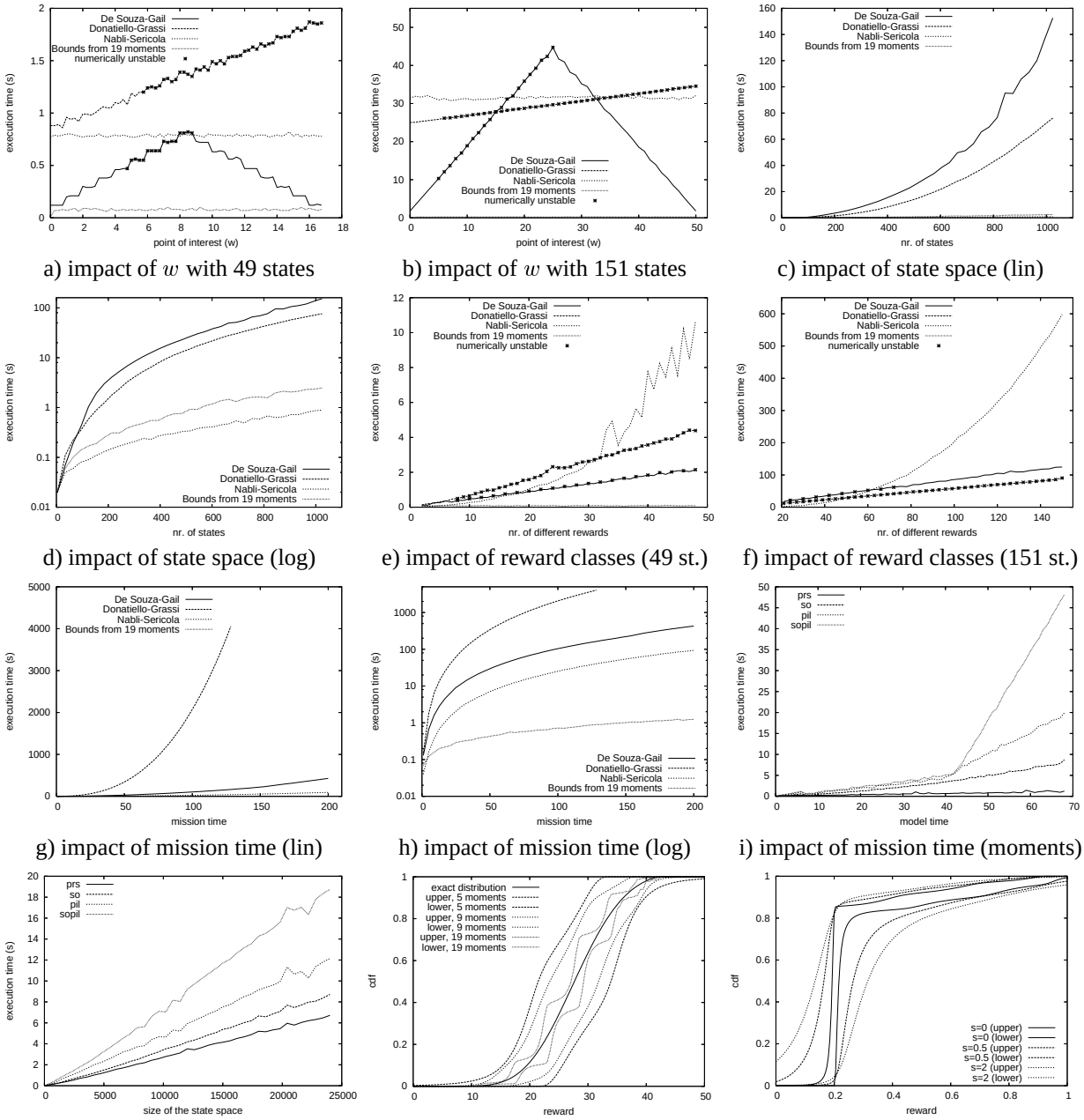
The results presented in this section are collected from our implementation of the given methods. We made a direct implementation of the algorithms of [5, 13, 7] based on the papers presenting them. We intended to create efficient implementations using the available hints of the original papers, even though the efficiency of our implementations might be different from the implementation of the authors. We did not implement steady state check in MRMSolve (yet), but this way the implementation of the algorithms are comparable with respect to the complexity of a given number of iteration steps. The execution time results presented below are measured on a regular PC (Celeron 766 MHz, 256 MB RAM, Linux). We measured the computation time from the start of interpretation of the MRMSolve model description file to the end of calculation of the distribution, or distribution bounds. We evaluated each point of interest separately, even if some of the methods can calculate more points with negligible additional complexity.

In some cases we experienced numerical errors i.e., probability values which are either less than 0 or greater than 1. We indicate these points with "numerical instability". We believe that the numerical errors are due to the numerically sensitive subtractions mentioned at the introduction of the methods. According to our experiences the limit of numerical stability is rather sharp. The results either behave well or extremely badly (extremely large positive or negative numbers, e.g., 10^{100}). Typically, we did not experience slightly bad results (e.g., between 1 and 2). We found that the numerically stable results of the 3 distribution analysis methods match up to the first 6 digits and they are in between the bounds calculated from the moments analysis in each cases.

We used the MRM presented in the previous section for a detailed comparative analysis of the mentioned reward analysis methods. In this model the queue length parameter, Q, determines the size of the state space $M = 3Q + 4$. To study the effect of the model parameters on the properties of the analysis procedures we tune some of the model parameters according to our purposes. Particularly, we analyze the effect of chaining the point of interest (w in $Pr(B(t) < w)$), the size of the state space (M), the number of reward classes and the system time (t).

Point of interest To evaluate the effect of the point of interest we increased the number of reward classes by replacing the Vector R section of the model definition with [i,j] = i+j;. At time $t = 1$ we consider two cases: $Q = 15$ and $Q = 49$ (which results 49 and 151 states, 18 and 52 reward classes, respectively).

The results in figure 7a) and b) indicate that 2 of the 4 methods are sensitive to the point of interest. The DeSouza–Gail method is sensitive to w by its nature, because its major loop takes into consideration only those reward classes whose reward rate is below w. An increasing w increase the number of considered reward classes and so the computational cost. This tendency dominates the first half of the curve. From that point on the procedure calculates the complementary distribution which requires the consideration of the reward classes with reward rate greater than w. This explains the decreasing portion from the middle. (The symmetric shape is a consequence of the equidistant reward rates of the reward classes.) The computational



a) impact of w with 49 states b) impact of w with 151 states c) impact of state space (lin)

d) impact of state space (log) e) impact of reward classes (49 st.) f) impact of reward classes (151 st.)

g) impact of mission time (lin) h) impact of mission time (log) i) impact of mission time (moments)

j) impact of state space (moments) k) moments based bounds l) impact of variance on distr. bounds

Figure 7. Numerical results

cost of the Donatiello–Grassi method increases nearly linearly with w.

The general behaviour of the curves is the same in both cases, but the relations of them are different apart from the moments based one which is the fastest in both cases. The speed of the DeSouza–Gail method depends on w. It is the fastest of the three distribution analysis algorithms for small and large w, but the other might became faster in between. With smaller state space the Donatiello–Grassi method requires the most computational time (particularly at large w) but with the bigger state space it becomes the fastest in the $w \in (14, 32)$ range. The Donatiello–Grassi and the DeSouza–Gail methods are numerically unstable in a wide range of w.

State space size To evaluate the impact of the state space size we use the reward structure defined in figure 5 with $t = 1$, $w = 1$ and vary Q from 0 to 340 (4 to 1024 states).

The complexity of the methods is quadratic with the number of states, but the coefficient of the growth is rather different (figure 7c), d)). The method of the moments and the Nabli–Sericola method grows much slower than the other two. In this particular case there is more than an order of magnitude difference between them. All of the methods are numerically stable in this range.

Number of reward classes We set various number of reward classes with equidistant reward rates to study the impact of the number of different reward rates by adding Rc = 3; #number of reward classes to the Const section and replacing with [i,j] = (Q+1)*3*floor((i*3+j+1)*Rc/((Q+1)*3))/Rc; the Vector R section of the model definition (figure 5). This way, with $Q = 15$ the reward rates are 0, 48; 0, 24, 48; 0, 16, 32, 48; etc. We evaluated the distribution in the middle of the possible range, i.e., $Pr(B(t) < \text{max. reward}/2)$. Similar to the point of interest experiment, we consider two cases: $Q = 15$ and $Q = 49$ (which results 49 and 151 states; 48 and 150 max. reward) with $t = 1$.

The Nabli–Sericola method slows down radically (about quadratically) with the increasing number of different rewards, while the computation time of the other methods grows nearly linearly (figure 7e), f)). With this example the Nabli–Sericola method is the fastest (faster than the moments based method) for small number of reward classes (and 49 states) but it quickly becomes the slowest as the number of reward classes grows.

With this example the Donatiello–Grassi method is numerically incorrect in almost all points. Interestingly, the numerical stability of the DeSouza–Gail method is better with larger state space.

Mission time For the analysis of the impact of mission time we use the model of figure 5 with $w = 1$ and t varying from 1 to 200.

Although the computational time of all the algorithms increases, the mission time has the most crucial effect on the Donatiello–Grassi method (figure 7g), h)). It is hard to read the tendencies from the figure, but logarithmic curves might indicate the $\mathcal{O}(t)$ behaviour of the moments based method against the $\mathcal{O}(t^2)$ behaviour of the distribution analysis methods (see Table 2). In this case, all results are numerically stable.

Comparison of moments based methods Distribution analysis methods are available only for a limited set of MRMs. However, the moments of the accumulated reward can be computed efficiently for a wider set of MRMs and some of these methods are implemented in MRMSolve 2.0 (see Table 1).

Here we compare moments based methods for the preemptive resume (PRS), second order (SO), partial increment loss (PIL), and combined second order partial increment loss (SOPIL) models. In favour of a fair comparison, we used the same number of steps in the randomization. The reward rates are set as [i,j] = (i+j). For the SO models, the variance matrix is defined in the Vector S section of the model description as [i,j] = (i+j)/10 and for the PIL models, the loss vector is defined in the Vector Alpha section as [i,j] = (i+j+1)/(Q+3).

Figure 7i) presents the execution time as the function of mission time when t goes from 0 to 100, $Q = 150$ (454 states) and 9 moments are computed. The first order PRS solver is the simplest and the fastest. The SO solver is a bit slower, because there is an extra vector multiplication in each iteration step. The PIL solver contains a more complex matrix operation. The SOPIL solver is the slowest, since it solves two sets of differential equations parallel.

Figure 7j) presents the dependence on the state space size. In this case the mission time is $t = 1$, the size of the state space vary from 7 to 24007 (with $Q = 1$ to 8001), and 9 moments are computed. The order of the solvers is the same as before. We did not find explanation for the change of the slope (at $t \sim 45$) in figure 7i) and the synchronized waves in Figure 7j)

The major weakness of moments based methods is that they provide distribution bounds only. We investigate the tightness of these bounds in figure 7k) with $Q = 15$ (49 states), $t = 10$. We consider the first order PRS case, because there are distribution analysis methods available for this case. The more moments we use, the tighter the bounds are. The bounds cannot be tightened arbitrarily close. More that 20 moments might cause numerical instability. The implemented moments based distribution estimation procedure does not become unstable because it drops the high moments as long as the estimation is stable. This way bounds can be improved till the limit of numerical stability, and beyond this point additional moments do not improve the bounds.

The bounds in figure 7k) were generated in less than a second, while the Nabli–Sericola method run 8 seconds, the Silva–Gail method run about 80 seconds and the Donatiello–Grassi method run more than 2000 seconds, but the last two methods were unstable.

In case of SO models, the variance of the reward accumulation increases (in general) the distance between the upper and lower bounds as it increases the variance of the reward distribution, especially when the corresponding first order case (with zero variance) is nearly deterministic (see Figure 7l), where $t = 0.2$, $Q = 20$, 19 moments are used and the variance vec-

tor is defined as [i,j] = s * (i+j)/10; with $s = 0,\ 0.5,\ 2$). Indeed the bounds of the SO models get tighter in a small interval around 0.2, because the higher variance of the SO model prevents a sharp increase (or big jump) of the distribution.

7. Conclusions

A new version of MRMSolve is introduced and used for comparing reward analysis methods. Among the three considered distribution analysis methods none of them is faster than the others in all cases. We found the Nabli–Sericola method to be numerically stable, while the DeSouza–Gail and the Donatiello–Grassi methods contain numerically sensitive subtractions which makes them unstable occasionally.

The introduced experiences verified the $\mathcal{O}$ tendencies reported in Table 2. The moments based analysis of reward measures is usually much faster than any direct distribution analysis method and numerically stable. We believe that moments based bounding of reward measure distribution is not needed as long as a direct distribution analysis is feasible, but in a range of models moments based reward analysis is the only feasible analysis method, even if it does not provide the distribution, only its bounds, which might be loose occasionally.

Finally, we hope that MRMSolve 2.0, whose source code is available at http://webspn.hit.bme.hu/∼mrmsolve, promotes the application of MRMs and their solution using both distribution analysis and moments based methods.

References

[1] A. Bobbio, V. G. Kulkarni, and M. Telek. Partial loss in reward models. In *MMR 2000*, pages 207–210, Bordeaux, France, 2000.

[2] A. Bobbio and M. Telek. The task completion time in degradable sytems. *Performability Modelling: Techniques and Tools*, pages 139–161. Wiley, 2001.

[3] Ing-Ray Chen and I. Ling Yen. Analysis of probabilistic error checking procedures on storage systems. *The computer journal*, 38(5):348–354, 1995.

[4] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The mbius modeling tool. In PNPM 2001, pages 241–250, Aachen, Germany, Sept 2001. IEEE CS Press.

[5] E. de Souza e Silva and R. Gail. An algorithm to calculate transient distributions of cummulative rate and impulse based reward. *Commun. in Statist. – Stochastic Models*, 14(3):509–536, 1998.

[6] Edmundo de Souza e Silva and Rosa M. M. Leao. The tangram-ii environment. In *TOOLS 2000*, pages 366–369, Schaumburg, IL, USA, March 2000. Springer, LNCS 1786.

[7] L. Donatiello and V. Grassi. On evaluating the cumulative performance distribution of fault-tolerant computer systems. *IEEE Transactions on Computers*, 1991.

[8] Boudewijn R. Haverkort, Raymond Marie, Gerardo Rubino, and Kishor Trivedi, editors. *Performability Modelling*. Wiley, Chichester, England, 2001.

[9] C. Hirel, B. Tuffin, and K.S. Trivedi. Spnp: Stochastic Petri net package version 6.0. In *TOOLS 2000*, pages 354–357, Schaumburg, IL, USA, LNCS 1786.

[10] G. Horváth, S. Rácz, and M. Telek. Analysis of second-order markov reward models. In *DSN/PDS 2004*, Florence, Italy, June 2004.

[11] G. Horváth and M. Telek. Analysis of markov reward models with partial reward loss based on a time reverse approach. tech. rep., Technical University of Budapest, 2003.

[12] V.G. Kulkarni, V.F. Nicola, and K. Trivedi. The completion time of a job on a multi-mode system. *Advances in Applied Probability*, 19:932–954, 1987.

[13] H. Nabli and B. Sericola. Performability analysis: a new algorithm. *IEEE Transactions on Computers*, 45:491–494, 1996.

[14] V.F. Nicola, R. Martini, and P.F. Chimento. The completion time of a job in a failure environment and partial loss of work. In *MMR'2000*, pages 813–816, Bordeaux, France, July 2000.

[15] S. Rácz and M. Telek. Performability analysis of Markov reward models with rate and impulse reward. *NSMC*, pages 169–187, Zaragoza, Spain, 1999.

[16] S. Rácz and M. Telek. Mission time analysis of large dependable systems. In *IPDS 2000*, pages 13–22, Chicago, IL, USA, March 2000. IEEE CS Press.

[17] S. Rácz, B. P. Tóth, and M. Telek. MRMSolve: Numerical analysis of large Markov reward models. In *Tools 2000*, pages 337–340. LNCS 1786, 2000.

[18] AT&T Labs Research. Graphviz - open source graph drawing software. http://www.research.att.com/sw/tools/graphviz.

[19] R. Sahner, K.S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 1996.

[20] W. H. Sanders, W. D. Obal, M. A. Qureshi, and F. K. Widjanarko. The ultraSAN modeling environment. *Performance Evaluation*, 24(1-2):89–115, 1995.

[21] M. Telek, A. Horváth, and G. Horváth. Analysis of inhomogeneous markov reward models. In *NSMC 2003*, pages 305–322, Urbana, IL, USA, Sept 2003.

[22] M. Telek and A. Pfening. Performance analysis of Markov Regenerative Reward Models. *Performance Evaluation*, 27&28:1–18, 1996.

[23] M. Telek and S. Rácz. Numerical analysis of large Markovian reward models. *Performance Evaluation*, 36&37:95–114, Aug 1999.