MRMSolve: Distribution Estimation of Large Markov Reward Models*

Sándor Rácz, Árpád Tari, and Miklós Telek

Technical University of Budapest, Hungary,
{raczs,arpi,telek}@webspn.hit.bme.hu

Abstract. MRMSolve is an analysis tool developed for the evaluation of large Markov Reward Models (MRM). The previous version of MRMSolve [8] provided only the moments of MRMs at arbitrary transient instant of time. This paper presents a new version of MRMSolve with new analysis features and software environment. The most important new functionality of MRMSolve is that it also makes distribution estimation of MRMs. MRMSolve can estimate the distribution of reward measures up to models with $\sim 10^6$ states, and to the best of our knowledge no other algorithm can handle MRMs with more than $\sim 10^4$ states.
Keywords: Markov Reward Models, accumulated reward, completion time, moments based distribution estimation.

1 Introduction

Stochastic reward processes are commonly applied for computer and communication systems' performance analysis. These processes are characterized by a discrete state stochastic process describing the system behaviour and the associated reward structure describing the system performance in different states. When a Continuous time Markov chain (CTMC) describes the system behaviour and there are non-negative *reward rate*s associated with the system states the stochastic reward process is referred to as Markov reward model (MRM).

There are two main subclass of measures associated with MRM. The random amount of reward accumulated during a given time interval is referred to as *accumulated reward*, and the random time to accumulate a given amount of reward is referred to as *completion time*. It is easy to compute the mean of these reward measures based on the transient analysis of the underlying CTMC, instead, this paper is about the distribution of the reward measures. Closed form expressions are known for the distribution of accumulated reward [4] and completion time [5] of MRMs in double Laplace transform domain. Unfortunately, the numerical evaluation of these transform domain expressions are infeasible when the state space of the CTMC is larger than ~ 20 states[1]. A number of numerical

* This work was partially supported by Hungarian Scientific Research Fund (OTKA) under Grant No. T-34972.

[1] The "$\sim$" sign indicates our general the rules of thumb. The complexity of particular procedures depend on several parameters which are not reflected by these general rules.

techniques were proposed for the evaluation of MRMs with larger state spaces. Some of them directly calculate the distribution of reward measures [11, 2, 6, 10], while some other of them provides only the moments of the measures [4, 14]. Table 1 summarizes their complexity and memory requirement, where m denotes the number of moments, t the time, K (K') the number of different (important) reward rates, T the number of transitions, and N the number of states. Based on Table 1 and our experiences we conclude that the numerical methods that directly provide the distribution of reward measures are applicable for MRMs with less than $\sim 10^4$ states while it is possible to evaluate the moments of reward measures up to $\sim 10^6$ states [14]. The MRMSolve tool is aimed for the analysis of large MRMs, hence it applies the numerical method presented in [14] to evaluate the moments of reward measures and it estimates the distribution based on the evaluated moments. Especially, the distribution estimation provides upper and lower bounds of the distribution function.

Method	CPU time	memory	output
Iyer, Donatiello and Heidelberger [4]	$\mathcal{O}(N^4 \cdot m^2)$	n.a.	moments
Smith, Trivedi and Ramesh [11]	$\mathcal{O}(N^3)$	n.a.	distribution
Donatiello and Grassi [2]	$\mathcal{O}(T \cdot K \cdot t^2)$	$\mathcal{O}(K \cdot N \cdot t)$	distribution
Nabli and Sericola [6]	$\mathcal{O}(T \cdot K \cdot t^2)$	$\mathcal{O}(K \cdot N \cdot t)$	distribution
Silva and Gail [10]	$\mathcal{O}(T \cdot K' \cdot t^2)$	$\mathcal{O}(N \cdot t^2)$	distribution
Telek and Racz [14]	$\mathcal{O}(T \cdot m \cdot t)$	$\mathcal{O}(N \cdot m)$	moments
CTMC transient analysis	$\mathcal{O}(T \cdot t)$	$\mathcal{O}(N)$	

Table 1. *Complexity of numerical analysis methods of MRMs*

This paper introduces the new features of MRMSolve and its wide range of applicability. To this end several application examples of the literature of system performance analysis are evaluated. Algorithmic details of the applied computation methods are omitted due to space constraint. The rest of the paper is organized as follows. Section 2 introduces the general structure of MRMSolve. Section 3 summarizes the applied distribution estimation method. Section 4 demonstrates the easy use and usefulness of MRMSolve via a number of application examples. The compact MRMSolve descriptions of models, taken from research papers, are presented together with numerical results. Section 5 discusses the computational and memory complexity of MRMSolve and Section 6 provides concluding remarks.

2 The general structure of MRMSolve

From the users' point of view the analysis of a MRM using MRMsolve is composed by two main steps: i) description of the MRM and the measures of interest, ii) automatic model analysis. To support the model description MRMSolve applies a MRM description language, which is developed particularly for the effective description of large MRMs. Generally, the large MRMs of practical interest exhibit structural regularity. The description language of MRMSolve describes the structural regularity of the models. The simpler is the structure of the model the easier is to describe it with MRMSolve, and the complexity of the model description is independent of the size of the model. The model description contains the system states identification (state space description), the definition of the possible state transitions, the definition of the system performance in each state and the description of the initial distribution. The input of automatic model analysis is the constructed model and the analysis parameters, which are the time points at which the distribution of the accumulated reward is evaluated, the accuracy of the calculation, and the number of evaluated moments (m). The analysis provides the following information about the investigated performance measures: the first m moments of the accumulated reward, the lower and upper bounds of the distribution function, and the estimation of the quantiles.

The major steps of the analysis procedure of MRMSolve are: the interpretation of the model description and generation of the generator matrix, the initial probability vector and the reward vector of the MRMs; calculation of moments of reward measures; distribution estimation of reward measures.

3 Distribution estimation

The problem of inversely characterizing distribution from their moments has been studied for over 100 years. Stieltjes, [12], established necessary and sufficient conditions for the existence of a *real valued*, *bounded* and *non-decreasing* function, for example a distribution function, on the interval $[0, \infty)$ such that its moments match given values. An excellent overview of the moment problem and some variations can be found in [1]. Numerous attempts have been made to obtain continuous or discrete distributions from their moments [1, 13] or to derive upper and lower bounds on the distribution function. In the case of distribution estimation at a given point x we need to find the distribution with the given moments whose cdf function is the less (the most) at point x. Indeed, we need to find the *domain of feasible distribution functions.* It practically means that this method gives upper and lower bounds on the feasible distribution functions (e.g., if the only information from a non-negative random variable is its mean value (μ_1) then the ideal estimator gives the domain $1 \geq F(x) \geq 1 - \frac{\mu_1}{x}$ based on the Markov-inequality). One can realize that this type of estimation methods give the best and the worst cases for the examined measure.

In several practically interesting cases of performance analysis the probability of extreme cases has to be bounded [10]. E.g., the probability that the stress accumulated by the system is larger than a dangerous threshold should be less than 10^{-3}. The analysis of the domain of feasible distributions can provide this kind of limits. Hence, the method applied in MRMSolve is of the second type and is taken from [7]. The method is based on the following main ideas. The Hankel determinant, $\text{Hankel}(a_0, \ldots, a_{2n}) := \mathbf{Det} \begin{pmatrix} a_0 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & a_{2n} \end{pmatrix}$, of the moments of any distribution is non-negative. The extreme case when the Hankel determinant is 0 is provided by a discrete distribution. The applied algorithm calculates the extreme discrete distribution which has the maximal mass at the evaluated point and its Hankel determinant is 0.

4 Numerical examples

4.1 Carnegie-Mellon multiprocessor system

Model construction The system is similar to the one presented in [11]. The system consists of P processors, M memories and an interconnecting network composed by switches which allows any processor to access any memory. The failure rates per hour for the system are set to be μ_P, μ_M and μ_S for the processors, memories and switches respectively.

Viewing the interconnecting network as S switches and modeling the system at the processor-memory-switch level, the system performance depends on the minimum of the number of operating processors, memories and switches. Each state is thus specified by a triple $(a; b; c)$ indicating the number of operating processors, memories and switches, respectively. We augment the preventive maintenance with state F. Events that decrease the number of operational units are the failures and events that increase the number of operational elements are the repairs. When a component fails, a recovery action must be taken (e.g., shutting down the failed processor, etc.).

Two kinds of repair actions are possible, preventive maintenance is initiated with rate μ_G per hour which restores the system to state $(N; M; S)$ with rate λ_G per hour from state F and local repair which can be thought of as a repair person beginning to fix a component of the system as soon as a component failure occurs. We assume that there is only one repair person for each component type. Let the local repair rates be λ_P, λ_M and λ_S for processors, memories and switches, respectively.

The system starts from the perfect state $(P; M; S)$. The performance of the system is proportional to the number of cooperating processors and memories, whose cooperation is provided by one switch. The system performance (processing power) in a given state is defined as the minimum number of the operational processors, memories and switches. The minimal operational configuration is supposed to have one processor, one memory and one interconnecting switch. We consider the *processing power* of the system averaged over a given time interval, i.e., the reward accumulated in $(0, t)$ is divided by t. Hence the processing power of the system is always between 0 and $min(N, M, S)$ unit.

Model description and automatic analysis For describing system states the tool allows us to use state variables. Extra states can be represented by state variables with associated logical conditions. A state can be identified by the values of its state variables, e.g., in our example a system state is identified by a triple $(a; b; c)$ or we can use some extra states as state F which cannot match to the state variables description. In general, the modeler makes some restrictions for the feasible values of state variables. There are two ways to do this, to limit the value of state variables and to add complex logical constraints to the state space description. In our example the state space is:

$$\begin{aligned}\mathcal{S} &= \{(a;b;c)\ :\ 0 \le a \le P,\ 0 \le b \le M,\ 0 \le c \le S\ \} \ \bigcup\ \{F\} \\ &= \{(0;0;0),(0;0;1),(0;0;2),\dots,(P;M;S-1),(P;M;S),F\}\end{aligned}$$

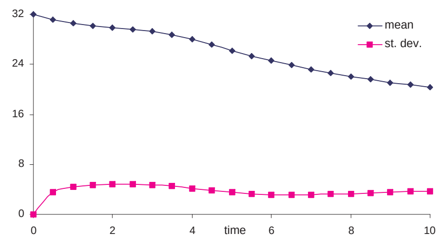
whose cardinality is $(P+1)\ (M+1)\ (S+1)+1$. Table 2 contains the model description of the examined Carnegie-Mellon multiprocessor system, where the model parameters are as follows: number of processors, memories and switches: $P = 128$, $M = 128$, $S = 32$, failure rate of a processor, a memory and a switch: $\mu_P = 0.5$, $\mu_M = 0.05$, $\mu_S = 0.02$, preventive maintenance rate: $\mu_G = 0.1$, repair rate of processors, memories and switches: $\lambda_P = 2$, $\lambda_M = 1$, $\lambda_S = 0.5$, preventive repair rate: $\lambda_G = 1$, enable/disable flag of preventive maintenance: $GR = \textbf{True}$. This system has 549154 states.

The example was evaluated with the following set of analysis parameters: time points: $t = 0.5 \dots 10$, calculated moments: $m = 6$, and the distribution of the accumulated reward were calculated in $k = 20$ points from the lower limit $F_l = 0.01$ to the upper limit $F_u = 0.99$ of the estimated distribution and a threshold curve is calculated at $Th = 0.01$ quantile of the estimated distribution.

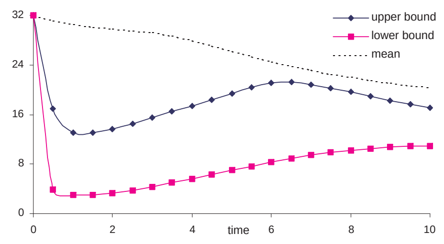
In Figure 1, we compare the system with allowing preventive maintenance (setting GR to **True**) and the system without preventive maintenance (setting GR to **False**). Allowing preventive maintenance results in higher processing power for larger t but causes very high variance.

State space	
$a\ :\ 0$ **To** P	number of active processors
$b\ :\ 0$ **To** M	number of active memories
$c\ :\ 0$ **To** S	number of active switches
Extra state $= \{\ F\ \}$	extra state
State-transition rates	
$(a;b;c) \to (a+1;b;c) = \lambda_P$	processor repair
$(a;b;c) \to (a;b+1;c) = \lambda_M$	memory repair
$(a;b;c) \to (a;b;c+1) = \lambda_S$	switch repair
$GR\ :\ (F) \to (P;M;S) = \lambda_G$	global repair
$(a;b;c) \to (a-1;b;c) = a \cdot \mu_P$	processor failure
$(a;b;c) \to (a;b-1;c) = b \cdot \mu_M$	memory failure
$(a;b;c) \to (a;b;c-1) = c \cdot \mu_S$	switch failure
$GR\ :\ (a;b;c) \to (F) = \mu_G$	global failure
Reward rates	
$(a;b;c) = \textbf{Min}(a,b,c)$	processing power in state (a;b;c)
Initial distribution	
$(P;M;S) = 1$	starting form the perfect state

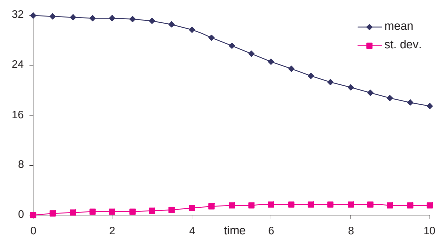
Table 2. *MRMSolve description of the Carnegie-Mellon multiprocessor system*



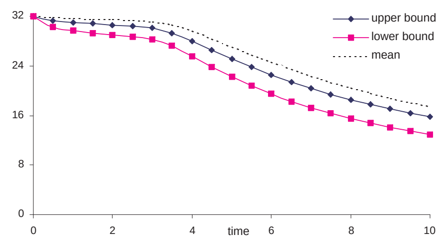
Moments of the processing power ($GR = 1$)



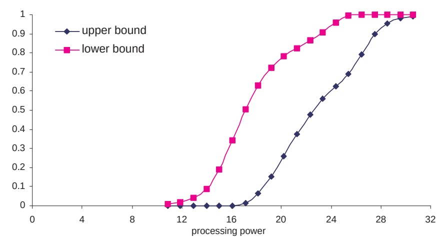
Bounds of the 1% threshold ($GR = 1$)



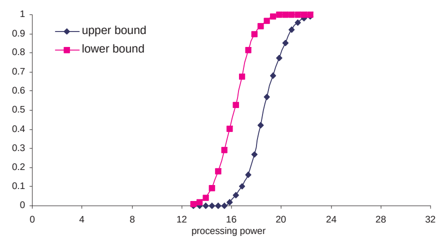
Moments of the processing power ($GR = 0$)



Bounds of the 1% threshold ($GR = 0$)



Distr. of processing power ($t = 10, GR = 1$)



Distr. of processing power ($t = 10, GR = 0$)

Fig. 1. Results for Carnegie-Mellon multiprocessor system

4.2 Busy periods in a M/M/1/K system

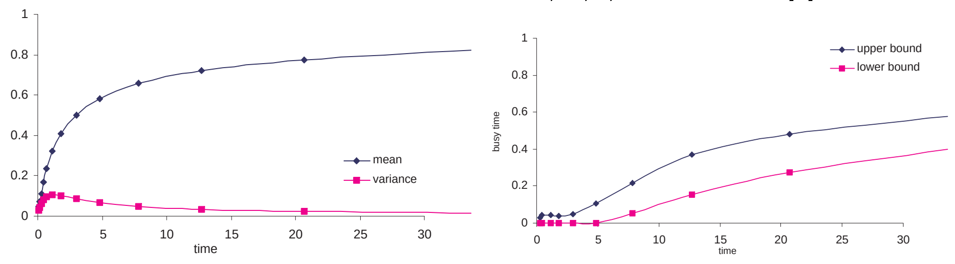
[3] studies the transient behavior of an $M/M/1/K$ queue where the arrival rate to the queue is $\lambda = 1$, the service rate is $\mu = 1$ and $K = 1000$. The queue is empty at time 0. The objective is to find the moments of the busy time of the system during the $[0, t]$ interval relative to the length of the interval and also to provide estimation of the 1% quantile. Table 3 shows the specification of this model for MRMSolve and Figure 2 provides the obtained results. MRMSolve provided the same results as [3].

4.3 Serving rigid, elastic and adaptive traffic

[9] presents the MRM of a transmission link serving peak bandwidth assured rigid, adaptive and elastic traffic with the Partial Overlap policy. There are two kinds of performance criteria considered, which are criteria for average throughput and criteria for throughput threshold. The first one requires only the steady-state analysis of the mean accumulated reward (which is easy to obtain from the steady state distribution of the underlying Markov chain), but the analysis of

State space	
a : 0 **To** 1000	actual number of requests
State-transition rates	
$(a) \to (a+1) = 1$	arrival rate
$(a) \to (a-1) = 1$	departure rate
Reward rates	
a!=0 : $(a) = 1$	busy or idle
Initial distribution	
a==0 : $(a) = 1$	starting form empty state
Parameters of the analysis	
$t = 0.1 \ldots 30$	time points of the analysis
$m = 10$	number of calculated moments
$k = 20$	number of points in distribution estimation
$F_l = 0.01$	lower limit of distribution estimation
$F_u = 0.99$	upper limit of distribution estimation
$Th = 0.01$	threshold value

Table 3. *MRMSolve description of the M/M/1/K system from [3]*



Mean and variance of the busy time in $(0, t)$ Upper and lower limits for the 1% threshold

Fig. 2. Analysis results of the M/M/1/K system

the throughput threshold is based on the transient analysis of a MRM. Table 4 shows the MRMSolve description of the model, with the following parameters: link capacity and its common part: C, C_{Com}, maximum number of adaptive and elastic flows: N_2, N_3, bandwidth demand of rigid, adaptive and elastic flows: b_1, b_3, b_3, minimum allowed bandwidth of adaptive and elastic flows: b_2^{min}, b_3^{min}, arrival intensity of rigid, adaptive and elastic flows: λ_1, λ_2, λ_3 (ideal) service rate of rigid, adaptive and elastic flows: μ_1, μ_2, μ_3. The bandwidth of adaptive and elastic flows in different states ($r_2(a,b,c)$, $r_2(a,b,c)$) are calculated by an external "awk" functions.

4.4 Buffered multiprocessor system

In [2,4] authors consider an optimization problem of a “buffered multiprocessor system”. The system is an N processor fault-tolerant system with a finite buffer of capacity BS. Table 5 presents the MRMSolve description of the model. The numerical results obtained by MRMSolve based on this description are equivalent with the ones presented in [2,4].

State space	
a : 0 **To** C_{Com}/b_1	number of rigid flows
b : 0 **To** N_2	number of elastic flows
c : 0 **To** N_3	number of adaptive flows
$N_2\ b_2^{min} + N_3\ b_3^{min} \leq C - C_{Com}$	guarantee minimal bandwidths
$b \geq 1$	tagging an adaptive flow
State-transition rates	
$(a;b;c) \rightarrow (a+1;b;c) = \lambda_1$	rigid flows arrival
$(a;b;c) \rightarrow (a;b+1;c) = \lambda_2$	adaptive flows arrival
$(a;b;c) \rightarrow (a;b;c+1) = \lambda_3$	elastic flows arrival
$(a;b;c) \rightarrow (a-1;b;c) = a\ \mu_1$	rigid flows departure
$(a;b;c) \rightarrow (a;b-1;c) = (b-1)\ \mu_2$	adaptive flows departure (w/o tagged one)
$(a;b;c) \rightarrow (a;b;c-1) = c\ r_3(a,b,c)\ \mu_3$	elastic flows departure
Reward rates	
$(a;b;c) = b_2\ r_2(a,b,c)$	bandwidth of tagged adaptive flow
Initial distribution	
<file name>	result of steady-state analysis

Table 4. *MRM of the transmission link from [9]*

State space	
a : 0 **To** N	actual number of processors
b : 0 **To** 1	operation condition
State-transition rates	
$b == 1$: $(a;b) \rightarrow (a-1;b) = a\ \mu_P$	
$b == 1$: $(a;b) \rightarrow (a;0) = BS\ \mu_b$	
$b == 0$: $(a;b) \rightarrow (N;1) = \mu_r/(N+1-a)$	
Reward rates	
$(a;b) = TH(a)$	throughput of a $M/M/p/p+B$ system (external 'awk' function)
Initial distribution	
$a == 0$ && $b == 1$: $(a;b) = 1$	starting form perfect state

Table 5. *MRMSolve description of the buffered multiprocessor system of [2, 4]*

5 Notes on computational complexity

The analysis has three main steps, the generation of the reward rate vector, the initial probability vector and the generator matrix of the CTMC; the calculation of the moments of accumulated reward; and the distribution estimation based on the moments. The complexity of the second step is provided in Table 1. The computational complexity of the third step is dominated by the numerical evaluation of the roots of an $\sim m/2$ order polynome, where m is the number of known moments. This step is practically immediate using a currently common computer.

The running time complexity of the MRMSolve tool is demonstrated via the running time data of an example introduced above. E.g., the analysis of the fairly large model of Section 4.1 with $N = 549154$ states and $T = 3244608$ transitions required: 7 min. – to generate the generator matrix and the initial probability and reward vectors; 524 min. – to evaluate the moments of accumulated reward at the required time points; ∼1 min. – to complete the distribution estimation based on the moments; on a normal PC with Celeron 600MHz processor and 256MB memory.

This example coincides with our general experience that for really large models the calculation of moments dominates the running time. The time of the first step (the generation of the generator matrix and the associated vectors) increases faster than linearly with the size of the model, but it is still acceptable for larger models as well. E.g., the first step took only 25 minutes for the same example with twice that many processors which result in twice that many states and transitions.

The nice feature of the second step is that, in contrast with the first and third steps, its complexity is easy to identify. Since the numerical procedure [14] is based on the randomization technique the dominant element of the generator matrix (the one with the largest absolute value), q, and the largest time instant t characterize the number of required iterations. The precision parameter has minor effect on the number of iterations if qt is large ($qt > 100$). In these cases the number of required iterations is between qt and $2qt$.

In the introduced example we calculate 6 moments at 20 time points form 0.5 to 10 and the dominant element of the generator matrix is $q = 64$, hence $qt = 640$ and the number of iterations is ∼1000. In one iteration cycle 6 vector-matrix multiplications, $6 \times 20 \times 2$ vector-vector multiplications and the same number of vector summations are performed. During this computation we had a sparse generator matrix with $(N + T)$ non-zero elements and dense vectors with N non-zero elements. One vector-matrix multiplication of a dense vector and a sparse matrix of $(N + T)$ non-zero elements means $(N + T)$ floating point multiplications and one vector-vector multiplications of dense vectors (N non-zero elements) means N floating point multiplications. The complexity of summation is less than the complexity of multiplication, hence we do not consider it independently, only as an additional factor of the complexity of multiplication.

Based on these data we can relate the obtained calculation time with the performance of the applied computer. We had ~ 1000 iteration cycles in 524 min., which means ∼2 cycle/min. One cycle requires $\sim 1000 \times 6 \times (N + T) + \sim 1000 \times 6 \times 20 \times 2 \times N \approx 140.000.000$ multiplications, hence our computer performed approximately 5.000.000 multiplications with associated data access, data storage and summations in a second.

The memory requirement of the analysis of this model was $(6 + 2) \times 20$ memory blocks for vectors of N floating point numbers and one memory block for the sparse generator matrix of $N + T$ floating point numbers. It is $8 \times 20 \times \sim 500.000 + \sim 3.500.000$ floating point numbers 8 bytes each, hence ∼100 MByte. Our computer has 256 MByte memory, hence this example fit to the memory. If it is not the case the number of multiplications drops to 1/10 of the value above.

6 Conclusion

The paper presents a software tool to analyze the distribution of reward measures of large MRMs. To the best of the authors knowledge the applied analysis approach is the only one that can cope with the distribution of MRMs of more than 10^4 states and this tool is the first one which implements this approach.

A model description language is developed for the effective definition of large MRMs. The distribution of reward measures are estimated based on their moments. The accuracy of this estimation method is poorer at around the mean of the distributions, but it is quite tight at around the extreme values. There are several practical examples, among others from the analysis of safety critical systems, when the goal of the analysis is to evaluate the occurrence of extreme cases.

The present implementation of the tool allows to define MRMs, to run the computation and to post-process the results with a widely applied work sheet managing program named Excel.

References

1. N. I. Akhiezer and M. G. Krein, "Some questions in the theory of moments", *Translations of mathematical monographs*, 2, AMS., Providence, R.I., 1962.
2. L. Donatiello and V. Grassi, "On evaluating the cumulative performance distribution of fault-tolerant computer systems", *IEEE Tr. on Computers*, 1991.
3. W.K. Grassmann, "Means and variances of time averages in markovian environment", *European Journal of Operational Research*, Vol. 31, pp 132 – 139, 1987.
4. R. Iyer, L. Donatiello and P. Heidelberger, "Analysis of performability for stochastic models of fault-tolerant systems", *IEEE Tr. on Comp.*, pp 902 – 907, 1986.
5. V. Kulkarni, V. Nicola and K. Trivedi, "On modeling the performance and reliability of multi-mode computer systems", *J. of Syst. and Sw.*, pp 175 – 183, 1986.
6. H. Nabli and B. Sericola, "Performability analysis: A new algorithm", *IEEE Tr. on Computers*, C-45(4), pp 491 – 494, 1996.
7. S. Rácz, "Numerical analysis of communication systems through Markov reward models", *Ph. D. thesis*, Technical University of Budapest, 2000.
8. S. Rácz, B. P. Tóth and M. Telek, "MRMSolve: A Tool for Transient Analysis of Large Markov Reward Models", In *Tools 2000*, pp 337 – 340, LNCS 1786, 2000.
9. S. Rácz, M. Telek and G. Fodor, "Link Capacity Sharing Between Guaranteed- and Best Effort Service on an ATM Transmission Link under GoS Constraints", *Telecommunication Systems*, Vol. 17:1,2, 2001.
10. E. De Souza e Silva and H.R. Gail, "An algorithm to calculate transient distribution of cumulative rate and impulse based reward", *Stoch. Models*, pp 509 – 536, 1998.
11. R. Smith, K. Trivedi and A.V. Ramesh, "Performability analysis: Measures, an algorithm and a case study", *IEEE Tr. on Computers*, C-37, pp 406 – 417, 1988.
12. T. Stieltjes, "Reserches sur les fractions continues", *Ann. Fac. Sci. Univ. Toulouse*, Vol. 2, pp J 1 – 122 , 1894.
13. Aldo Tagliani, "Existence and stability of a discrete probability distribution by maximum entropy approach", *Applied Math. and Computation*, pp 105 – 114, 2000.
14. Miklós Telek and Sándor Rácz, "Numerical analysis of large Markov reward models", *Performance Evaluation*, Vol. 36&37, pp 95 – 114, August 1999.