Xmgm: Performance Modeling Using Matrix Geometric Techniques

Boudewijn R. Haverkort, Aad P.A. van Moorsel, Dirk-Jan Speelman*

October 29, 1993

Abstract

Over the last two decades a considerable amount of effort has been put in the development and application of matrix geometric techniques for the analysis of queueing systems of which the (embedded) Markov chain exhibits a regular structure. Most of this work however has been presented in either a mathematical context or in a purely application-oriented context.

In this paper we present XMGM*, a performance analysis tool which allows its users to easily specify queueing systems in terms of interarrival and service time distributions.* XMGM *then takes care of the translation of this description to an underlying Markov chain that exhibits a matrix geometric solution. Subsequently, it takes care of the derivation of the measures specified by the user.*

With XMGM *models are specified via C procedure calls. This turns out to be a very flexible approach for modelling queueing systems; it also allows for the easy evaluation of models over parameter ranges.*

1 Introduction

Over the last two decades a considerable amount of research has been done in the field of queueing system analysis. For obtaining the average behaviour of queueing systems many closed-form solutions have been obtained, both for single queueing stations and for networks of them (see, e.g., [7, 16]). However, these solutions generally only apply under a number of restrictions which in practice often do not hold. Especially when present-day communication systems are analyzed, the usual Poisson arrivals and exponential packet lengths are no longer valid assumptions. Furthermore, when finite buffers are considered, many of the proposed techniques fall short.

Some of the above mentioned shortcomings are solved by using numerical techniques or simulation. In the former case, often a Markov chain representation underlying the queueing system is constructed, which is subsequently solved numerically, generally by an iterative procedure [8]. Simulation is always applicable, however, it suffers the drawback that it is rather costly, especially for the more intricate systems and scenarios we want to analyze today.

For a wide class of systems that do not conform the requirements for a "classical" closed-form solution, *matrix geometric methods* might still be applicable and also lead to closed-form solutions [9, 10, 11]. These methods are mathematically very elegant, insightful, and allow for a computationally efficient implementation.

Up till now, matrix geometric methods have been used for practical modelling studies in the field of communication systems, e.g., for the modelling of complex workloads in ATM systems (see, e.g., [1]). These case studies, however, suffer from mathematical complexity. We are only aware of one general purpose tool for matrix geometric methods, i.e., the tool MAGIC developed by Squillante [13]. To really exploit the potential of matrix geometric techniques, it is needed that more modelling tools support these techniques.

In this paper we present XMGM [12], a tool that can be used to easily specify and solve queueing models that fall in the category that can be solved by matrix geometric techniques. XMGM makes extensive use of the language C and is easily extendible with new user-interface components and with new algorithms.

XMGM can best be viewed as the successor of MGMtool [2, 3]. In comparison with MGMtool, XMGM provides a much larger model class, a dedicated X windows user-interface and its mathematical routines are far superior to those of MGMtool.

This paper is organized as follows. In Section 2 we discuss the basic mathematics of the approach followed in matrix geometric methods. The architecture of XMGM is described in Section 3. The modelling primitives available for the user are presented in Section 4. Some implementation aspects and applications are discussed in Section 5 and 6 respectively. Section 7 concludes the paper.

*The authors are with the University of Twente, department of Computer Science, P.O. Box 217, 7500 AE Enschede, the Netherlands (e-mail: haverkor@cs.utwente.nl).

0-8186-5292-6/94 $3.00

2 Matrix geometric methods

In this section we briefly discuss the mathematical aspects of matrix geometric techniques. For an in-depth survey we refer to the work by Neuts [10, 11] or to the tutorial by Nelson [9].

For an M|M|1 queueing model with arrival rate λ and service rate μ, the Markov chain underlying this queueing model is a simple birth-death process where the state variable denotes the total number of packets in the queue and server. The steady-state probability distribution of the number of packets N in the queue has a geometric form when expressed as a function of the utilization $\rho = \lambda/\mu < 1$:

$$z_i = \Pr\{N = i\} = \begin{cases} z_0\rho^i, & i = 1, 2, \cdots \\ 1-\rho, & i = 0. \end{cases} \quad (1)$$

From the distribution z_i, $i = 0, 1, \cdots$, we can derive various performance metrics, such as the average number of packets in the queue, the average response time (via Little's law), or the probability of at least j packets in the queue.

The generator matrix $\mathbf{Q}$ of the Markov chain underlying the M|M|1 queue has the following tridiagonal form:

$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda & 0 & \cdots \\ \mu & -(\lambda+\mu) & \lambda & \cdots \\ 0 & \mu & -(\lambda+\mu) & \cdots \\ \vdots & 0 & \mu & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \quad (2)$$

Observe that, apart from the first column, all the columns are basically the same, except that "they are shifted down one row". We call the first column a *boundary column* and the other ones *repeating columns*. The fact that the steady-state distribution of the number of packets N in the M|M|1 queue exhibits a geometric solution has turned out to be intimately related to the regular structure of the matrix $\mathbf{Q}$.

When we consider a PH|PH|1 queue, i.e., a generalization of an M|M|1 queueing model in which both the service and the interarrival times have a phase-type distribution, the underlying Markov chain again has this regular structure. Suppose that the interarrival and service time distributions are given by the representations $(\underline{\alpha}, \mathbf{T})$ (size$(\mathbf{T}) = m_a$) and $(\underline{\beta}, \mathbf{S})$ (size$(\mathbf{S}) = m_s$) respectively (see [10, Chapter 2], the generator matrix again exhibits a tridiagonal form:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{B}_{00} & \mathbf{B}_{01} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}_{10} & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \quad (3)$$

The square matrices $\mathbf{A}_i$ are of size $m_a m_s \times m_a m_s$ and can be directly expressed in terms of the representations of the service and interarrival time distributions [10, Chapter 3]. A similar remark can be made for the matrices $\mathbf{B}_{ij}$. The set of states in the above CTMC for which the number of customers present in the queue is fixed to some i is called a *level*; states within a level only differ in the phase of the interarrival and service time distribution.

Let $\underline{z}_i$ denote the vector of lexicographically ordered steady-state probabilities of states in level i. Then it can be shown that

$$\underline{z}_i = \underline{z}_1 \mathbf{R}^{i-1}, \; i = 1, 2, \cdots. \quad (4)$$

The length of the vectors $\underline{z}_i$, $i = 1, 2, \cdots$, is $m_a m_s$. The square matrix $\mathbf{R}$ of size $m_a m_s \times m_a m_s$ follows from the matrix quadratic equation

$$\sum_{i=0}^{\infty} \mathbf{R}^i \mathbf{A}_i = \mathbf{R}^2\mathbf{A}_2 + \mathbf{R}^1\mathbf{A}_1 + \mathbf{R}^0\mathbf{A}_0 = \mathbf{0}. \quad (5)$$

To start the recursive relation (4), the following boundary equations must be solved to obtain $\underline{z}_0$ and $\underline{z}_1$:

$$(\underline{z}_0, \underline{z}_1) \begin{pmatrix} \mathbf{B}_{00} & \mathbf{B}_{01} \\ \mathbf{B}_{10} & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 \end{pmatrix} = (\underline{0}, \underline{0}), \quad (6)$$

under the usual normalization condition:

$$\sum_{i=0}^{\infty} \underline{z}_i \underline{1} = \underline{z}_0 \underline{1} + \sum_{i=1}^{\infty} \underline{z}_i \underline{1} = \underline{z}_0 \underline{1} + \underline{z}_1 (\mathbf{I} - \mathbf{R})^{-1} \underline{1} = 1. \quad (7)$$

The boundary equations follow from the "boundary" of the global balance equation for the steady-state behaviour of the Markov chain. Since these linear systems are normally not too large, they can be easily solved using Gaussian elimination. The matrix $\mathbf{R}$ is calculated iteratively. Starting with $\mathbf{R}(0) = -\mathbf{A}_0\mathbf{A}_1^{-1}$, we obtain successive approximations of $\mathbf{R}$ as follows:

$$\mathbf{R}(k+1) = -\mathbf{A}_0\mathbf{A}_1^{-1} - \mathbf{R}^2(k)\mathbf{A}_2\mathbf{A}_1^{-1}, \; k = 0, 1, \cdots \quad (8)$$

The iteration is stopped whenever $||\mathbf{R}(k) - \mathbf{R}(k+1)|| < \epsilon$. Once the matrix $\mathbf{R}$ and the two boundary vectors $\underline{z}_0$ and $\underline{z}_1$ have been obtained, all types of interesting performance metrics can be derived in a way similar to the M|M|1 case.

3 Xmgm architecture

As stated already in Section 1, one of the reasons why matrix geometric techniques are not that widely used for applied performance analysis studies, might be the fact that the required mathematics is non-trivial. We therefore aim at providing high-level constructs for the definition of queueing systems. These constructs should allow users to easily specify a queueing system, without having to bother about the underlying mathematical engine. Furthermore, the users should have a large flexibility in specifying experiment series. The translation to the underlying solution modules should be both automatic and transparant. The output of the tool should be given in terms of the model as specified by the user. If more specialized users want so, they should be able to control the underlying mathematical engine and to view intermediate results [4]. Keeping these requirements in mind, a performance analysis tool based on matrix geometric methods can be thought to consist of four functional parts.

In the **specification part** modelling primitives are provided that allow users to specify phase-type interarrival and service time distributions, the number of servers, the size of the buffers and the measures that must be calculated. Furthermore, it allows to control internal quantities such as the required accuracy or the maximum number of iterations.

In the **transformation part** procedures are provided that transform the user description of the queueing system to the $\mathbf{A}_i$ and $\mathbf{B}_{ij}$ matrices in Equation (3), characteristic for the matrix geometric approach.

In the **solution part** procedures are provided that derive the matrix $\mathbf{R}$ and the boundary vectors $\underline{z}_0$ and $\underline{z}_1$ from the already derived matrices $\mathbf{A}_i$ and $\mathbf{B}_{ij}$, using Equations (6) through (8).

Finally, in the **evaluation part** procedures are provided that derive "user-oriented performance measures" from the matrix $\mathbf{R}$ and the boundary vectors $\underline{z}_0$ and $\underline{z}_1$, such as average queue lengths, average waiting times, etc.

The four parts and their relation are depicted in Figure 1 in which boxes represent active entities (programs) and ovals represent passive entities (files, data structures)[1].

4 Xmgm modelling primitives

XMGM has been built as a library of C procedures

[1] In a sense, the specification is a file, and should therefore be represented as an oval. On the other hand, in the implementation of , the specification is an executable program that generates the input to the transformation part. From that viewpoint, the specification should be represented as a box.

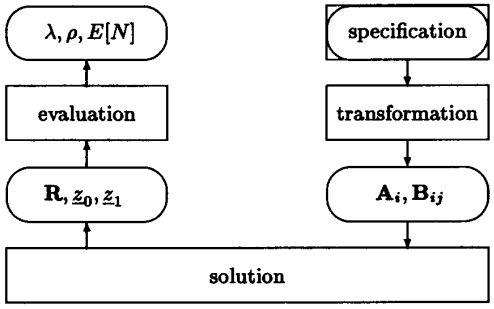

Figure 1: XMGM architecture

and functions that can be used in a C program to specify queueing systems and the desired performance measures. We introduce the modelling primitives by giving an example of the specification of an M|PH|1 queue in Section 4.1. We then discuss in more detail the possibilities of the specification primitives in Section 4.2.

4.1 Specificing a M|PH|1 queue

In the description that follows, we continuously refer to the C program in Figure 2. A C program meant as input for XMGM may include any normal C code, but no `main()`. It should, however, include a procedure `project()`. In the body of `project()` the actual model specification takes place.

First, phase-type representations of the interarrival and service time distributions have to be given. To accomplish this, three real variables are declared (`lambda`, `mu1`, and `mu2`), as well as PH representations of the interarrival and the service time distributions (`Arr` and `Ser` respectively). Then, the variables are assigned their value, followed by the assignment of a value to the matrices. Here, the interarrival time distribution is of exponential type, with rate `lambda`, and the service time distribution is of hypo-exponential type, with two phases. The first phase has rate of completion `mu1`, and the second `mu2`.

The measures that are to be derived by the tool are specified with the procedure `COMPUTE()`. This procedure has a variable length parameter list that should end with the special parameter `END`.

The procedure `M_PH_1()` requires as parameters an interarrival and service time distribution. It transforms them in an internal representation which basically consists of the matrices $\mathbf{A}_i$ and $\mathbf{B}_{ij}$, derives the matrix $\mathbf{R}$ and the boundary vectors $\underline{z}_0$ and $\underline{z}_1$, and calculates the measures of interest as specified by

```
project()
{
   double lambda, mu1, mu2;
   PHrep Arr, Ser;
   lambda = 1.0;
   mu1 = 8.0; mu2 = 3.0;
   EXPV(&Arr, lambda);
   HYPOV(&Ser, 2, mu1, mu2);
   COMPUTE(RHO, ETA, EN, END);
   M_PH_1(Arr, Ser);
   TOFILE(RHO, ETA, EN, END);
}
```

Figure 2: Specifying an M|PH|1 queueing system

COMPUTE(). Finally, the procedure TOFILE() specifies which measures have to be included in the standard report of XMGM.

4.2 Specification primitives

XMGM allows for the direct usage of a number of standard PH distributions[2]: exponential (EXPV()), Erlang (ERLV()), hypo-exponential (HYPOV()), hyper-exponential (HYPERV()), Coxian (COXV()) and interrupted Poisson processes (IPP()).

Starting from these standard distributions, more complicated ones can be constructed by expanding any single state in which the residence time is exponentially distributed, in a state in which the residence time is PH distributed (see [2, 12]).

When none of the standard procedures apply, one can resort to the C language, to make procedures that correctly fill the representation matrices of the required PH distributions. In this way, more complex models with a matric geometric solution can be studied.

From the PH representations of the interarrival and service time distributions the matrices $\mathbf{A}_i$ and $\mathbf{B}_{ij}$ are derived. Depending on the type of interarrival and service time distribution, this derivation is more or less complex. The model classes that are currently distinguished are:

- infinite buffer: PH|PH|1, M|PH|1, PH|M|1, PH|M|c, E_m|M|1, E_m|M|c, and M|M|1;
- finite buffer: M|PH|1|K+1, PH|M|1|K+1, PH|M|c|K+c, E_m|M|c|K+c, E_m|M|c|K+c, and M|M|1|K+1.

[2]Note that these procedures employ variable argument lists. This explains the V in their name.

The measures that a user of XMGM wants to be derived have to be indicated in the argument list of the procedure COMPUTE(). The following standard measures are available: the first and second moment of the interarrival and service time distribution (as mean value, rate, second moment, variance and squared coefficient of variation), the utilization (RHO), the largest eigenvalue of $\mathbf{R}$ (ETA), the mean number of packets in the queue (ENq), the mean number of packets in the total system (EN), the mean waiting time (EW), the mean response time (ER), the steady-state probabilities per level (Z+i= $\underline{z}_i\underline{1}$, $i = 0, 1, \cdots$), and the blocking probabilities per level (B+i= $\sum_{j \geq i} \underline{z}_j\underline{1}$, $i = 0, 1, \cdots$).

Apart from these system-oriented measures one can also obtain measures related to the computations themselves: the required CPU time in total or for the individual parts in the computation (CPU, CPUr, CPUz, CPUe), and the number of steps needed to iteratively obtain $\mathbf{R}$ (STEPS).

Finally, the procedure TOFILE() generates an ASCII table of the measures indicated in its argument list. Conversion of this table to a LaTeX table or a Gnuplot plotfile is also supported by XMGM.

5 Xmgm implementation issues

In Section 5.1 we touch upon the implementation environment of XMGM and in Section 5.2 we discuss the user-interface(s) of XMGM.

5.1 The environment of XMGM

XMGM has been developed in C for Sun 4 workstations, running the Unix operating system (or a Sun variant thereof). Basically, whenever a user has completed the specification of a project(), XMGM takes care of the compilation and execution of it. In doing so, XMGM automatically links it with predefined functions from a library.

Experience with XMGM has revealed that the calculation of $\mathbf{R}$ is the main performance determining factor. The size of $\mathbf{R}$ equals $m_a m_s$. In the current implementation of XMGM, this size is limited to $50^2 = 2500$. Special sparse matrix techniques exploiting the row-wise zero structure of $\mathbf{R}$ have been implemented to cope with such large systems [12].

5.2 Motif user-interfaces

XMGM tool has been designed for usage in an X-window environment (or compatible). In Figure 5.2 we show the basic input window of XMGM which has

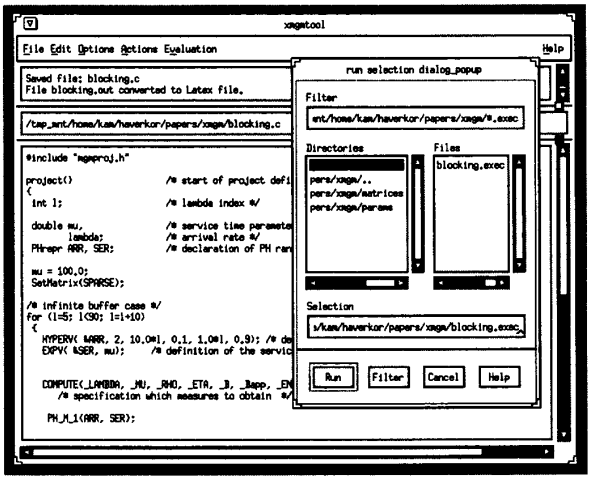

Figure 3: The user-interface of XMGM

been developed using the Motif widget set. Facilities for text-editing, compilation, execution and file-handling are provided. In principle, XMGM can be used without descending to the Unix level.

In order to enhance the compatibility of XMGM, one can also use it without a windowing system. Whenever the model is specified in a file named `project.c`, the computed results will become available in the file `results` after the call `mgm -o results project`.

6 An Xmgm application: buffer dimensioning

Switches in telecommunication systems are, among others, characterized by the size of their buffers. Indeed, the buffer size plays an important role in the performance of a switch and determines to a large extent the blocking probabilities and switch throughput.

Many analytical techniques, however, fall short when finite buffer schemes are introduced. As a consequence of this, many switches are dimensioned using models for infinite-buffer switches. This can, however, incur large errors in the dimensioning as illustrated below.

We will analyse a communication switch under bursty as well as deterministic packet arrivals and exponentially distributed packet lengths. In determining the blocking probabilities, we will make use of three models:

1. **Infinite buffer model.** The blocking probability B_K^1 is defined as the probability of having K or more packets in a PH|M|1 *queue*;

2. **Renormalized infinite buffer model.** The blocking probability B_K^2 is defined as the probability of having exactly K packets in a PH|M|1 *queue*, however, renormalized to account for the finiteness of the modelled system [5, 14, 15];

3. **Finite buffer model.** The blocking probability B_K^3 is defined as the probability of having exactly K packets in a PH|M|1|K+1 *queue*.

ρ	B_K^1	B_K^2	B_K^3
0.55	0.0000	0.0000	0.0000
0.65	0.0002	0.0000	0.0001
0.75	0.0043	0.0016	0.0017
0.85	0.0487	0.0122	0.0124
0.95	0.3905	0.0532	0.0504

Table 1: Blocking in the E_{20}|M|1|10 queue

It should be noted that we misuse the term blocking probability. What we actually mean is the "system full probability". In case of a Poisson arrival stream, these two would equal due to the PASTA property. As we deal with non-Poissonian arrivals, this equality does not hold. Also notice that, given K packets in the queue, there are $K+1$ packets in the queueing system as a whole.

We first address the case where the arrival process is very deterministic. We have chosen an Erlang-20 interarrival time distribution to represent this. Also, $K+1=10$, i.e., there are 9 buffer places and there is one server. In Table 1 we present the three blocking probabilities for increasing utilization ρ. What can be observed is that approximation B_K^1 does not yield accurate results; it highly overestimates the blocking probability. The renormalized blocking probability B_K^2 is a good approximation of the exact blocking probability B_K^3.

Similar conclusions can be drawn for the case when we deal with a more bursty arrival process, i.e. the interarrival times have a 2-phase hyper-exponential distribution with squared coefficient of variation equal to 1.566 (see Table 2). Similar conclusions can be drawn in case the arrival process is even more bursty.

7 Concluding remarks

In this paper we have described a performance analysis tool based on matrix geometric methods. After

ρ	B_K^1	B_K^2	B_K^3
0.250	0.0000	0.0000	0.0000
0.353	0.0005	0.0003	0.0003
0.510	0.0062	0.0027	0.0024
0.667	0.0433	0.0126	0.0120
0.824	0.2163	0.0385	0.0374
0.980	0.8529	0.0840	0.0833

Table 2: Blocking in the $H_2|M|1|10$ queue

a short resume of matrix geometric methods we have discussed the general tool architecture as well as various implementation aspects. XMGM is, to the best of our knowledge, the only general performance analysis tool that relies on matrix geometric methods and that does not bother the tool user with mathematical details of this method (thereby excluding its predecessor MGMtool [2, 3]). Instead, it allows users to specify queueing systems by only specifying a PH representation of the interarrival and service time distribution. XMGM then takes care of the underlying mathematics. Also on a relatively high level, the user specifies the desired output measures. Apart from average performance measures, more detailed performance measures can also be derived. With a number of examples we have shown the usability of the tool. Integration with the programming language C has turned out to be very powerful.

For the near future, we see a number of directions that can be taken to enhance the work presented here. The class of models supported by XMGM can still be extended as well as the underlying mathematical engine, e.g., by including routines that allow for the calculation of waiting time distributions. Also, XMGM needs to be used for more realistic modelling studies. We are currently using XMGM for the analysis of ATM connection control mechanisms [6] and are including Markovian arrival processes (MAPs; see, e.g., [1]) in its "modelling world".

References

[1] C. Blondia, "Performance Evaluation of an M/1-stage in an ATM Switching Element", *Performance Evaluation* **15**, pp.1–20, 1992.

[2] A. Dijkstra, *MGMtool: A tool for the performance analysis of* PH|PH|c *queues, based on matrix geometric techniques*, M.Sc. thesis, University of Twente, Department of Electrical Engineering, 1992.

[3] B.R. Haverkort, A.P.A. van Moorsel, A. Dijkstra, "MGMtool: A Performance Analysis Tool Based on Matrix Geometric Methods", in: *Computer Performance Evaluation 1992: Modelling Techniques and Tools*, Eds.: R. Pooley, J. Hillston, Edinburgh University Press, pp.312–316, 1993.

[4] B.R. Haverkort, K.S. Trivedi, "Specification and Generation of Markov Reward Models", *Discrete-Event Dynamic Systems: Theory and Applications* **3**, pp.219–247, 1993.

[5] B.R. Haverkort, "Some Notes on Deriving Blocking Probabilities", in progress, 1993.

[6] G. Heijenk, A.P.A. van Moorsel, I.G. Niemegeers, "Performance of a Connectionless Protocol over ATM", *Memoranda Informatica* 92-02, University of Twente, 1992.

[7] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, John Wiley & Sons, 1975.

[8] U. Krieger, B. Müller-Clostermann, M. Sczittnick, "Modelling and Analysis of Communication Systems Based on Computational Methods for Markov Chains", *IEEE Journal on Selected Areas in Communications* **8**(9), pp.1630-1648, 1990.

[9] R. Nelson, "Matrix Geometric Solutions in Markov Models: A Mathematical Tutorial", *IBM Research Report* RC 16777, 1991.

[10] M.F. Neuts, *Matrix Geometric Solutions in Stochastic Models: An Algorithmic Approach*, Johns Hopkins University Press, Baltimore, 1981.

[11] M.F. Neuts, *Structured Stochastic Matrices of* M|G|1 *Type and Their Applications*, Marcel Dekker Inc., New York, 1989.

[12] D.H.J. Speelman, *Xmgm: A Performance Modelling Tool Based on Matrix Geometric Methods*, M.Sc. thesis, University of Twente, Department of Computer Science, 1993.

[13] M.F. Squillante, "MAGIC: A Computer Performance Modelling Tool Based on Matrix-Geometric techniques", in: *Computer Performance Evaluation: Modelling Techniques and Tools*, Eds.: G. Balbo, G. Serazzi, North-Holland, pp.411–425, 1992.

[14] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach*, John Wiley & Sons, 1986.

[15] H.C. Tijms, "Heuristics for Finite-Buffer Queues", Research Report 1991-29, Free University, Department of Econometrics, Amsterdam, 1991.

[16] K.S. Trivedi, *Probability & Statistics with Reliability, Queueing and Computer Science Applications*, Prentice-Hall, 1982.